Modélisation et Exploitation de Profils : Accès Sémantique à des Ressources

Pascaline Tchienehom

Modélisation et Exploitation de Profils : Accès Sémantique à des Ressources

Pascaline Laure TCHIENEHOM

à mes enfants

Résumé

L'accès à des ressources est une vision plus large de l'accès à l'information où les ressources ne sont plus limitées aux utilisateurs et aux documents mais peuvent être étendues à toutes sortes de catégories de personnes, choses ou actions : étudiants, thèses, dispositifs mobiles, logiciels, etc. L'hétérogénéité de ces ressources a conduit au développement de nombreuses méthodes d'accès. Ces méthodes sont basées sur la description des ressources utilisées, que l'on appelle *profil*, et sur la définition de principes d'exploitation de ces descriptions pour la réalisation d'une tâche spécifique : recherche ou filtrage d'information, partage ou échange d'informations, envoi de messages d'alertes pour prévenir d'une situation particulière, etc.

Les modèles de profils et principes d'exploitation de ces derniers diffèrent d'une application à une autre. Afin de faire collaborer différentes applications, il y a un réel besoin de définir un cadre homogène et flexible de modélisation et d'exploitation de profils. Nos travaux visent à proposer des solutions sur ces deux aspects, au travers d'un modèle générique de profil ainsi que de méthodes d'analyse et d'appariement d'instances de ce modèle.

L'objectif de notre contribution est de maintenir une flexibilité au niveau de la modélisation de profil, sans entraver l'appariement entre modèles de profils décrits dans des langages différents. Pour cela, nous ajoutons une dimension sémantique à notre modèle générique. Le but est d'expliciter le sens des éléments descriptifs d'un profil (structure et contenu) et de proposer des moyens de déduction automatique d'éléments de sémantique compatible entre profils différents. La sémantique est instanciée par des métadonnées et n'est pas limitée à un cadre applicatif pré-défini. Elle doit permettre de faire interopérer des modèles de profils issus, éventuellement, d'applications différentes. Cette sémantique permet donc de définir des règles d'interprétation de profils et établit ainsi des *ponts de coopération* ou *langages partagés* entre modèles de profils différents. Pour valider nos propositions, un outil d'aide à la construction, à la visualisation et à l'analyse sémantique de profils a été implémenté. De plus, une évaluation des méthodes d'analyse sémantique et d'appariement de profils proposées a été effectuée.

Mots clés : *Profil, Accès Sémantique, Ressource, Modèle générique, Appariement, Interopérabilité, Métadonnées.*

Abstract

Resources access is a broader view of information access where resources are not limited to users and documents but can be extended to any kind of persons, things and actions such as : students, thesis, mobile devices, software and so on. Heterogeneity of resources has led to the development of several access methods. These methods rely on the description of resources that we call *profile*, and also on the definition of using rules for those profiles in order to achieve a specific task : information retrieval or filtering, sharing or exchanging information, sending alert messages to inform of a particular situation and so forth.

Profiles and their using rules differ from one application to another. For applications cooperation, there is a real need of a flexible and homogenous framework for the modelling and use of profiles. Our research work aims at providing solutions in those two aspects, thanks to a profile generic model and methods for analysing and matching instances of this model.

The goal of our contribution is to keep flexibility at the modelling level without compromising the matching between profiles described by different languages. For that, we add a semantic dimension to our generic model. The aim is to clarify the meaning of profiles descriptive elements (structure and contents) and to provide applications with means for an automatic deduction of compatible semantics elements between different profiles. Semantics is instantiated with metadata and is not restricted to a pre-defined applicative framework. Semantics should allow interoperability between profiles described by different applications. This semantics should then define rules of profiles interpretation and should set *bridges* or *shared languages* among those profiles. In order to validate our proposals, an assistant tool for profiles construction, visualization and semantic analysis has been implemented. Furthermore, an evaluation of methods for profiles semantic analysis and matching has been carried out.

Keywords : *Profile, Semantic Access, Resources, Generic model, Matching, Interoperability, Metadata.*

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les développements que connaissent aujourd'hui Internet, et en particulier le World Wide Web, mais également les intranets et tous les environnements numériques de travail conduisent à la mise à disposition d'une masse sans cesse croissante de ressources. Une ressource peut être de nature très variée : usager (individu, groupe d'usagers, étudiant, enseignant), contexte environnemental d'un usager (situation géographique, environnement matériel et logiciel) ; contexte cognitif ou besoin de l'usager (identifié grâce aux actions de ce dernier : documents jugés, sauvegardés, annotés, etc.) ; informations mises à disposition (articles, thèses, etc.) ; collections ou parties d'informations ; dispositifs matériels ou logiciels ; etc. De nombreuses architectures matérielles pour stocker, partager ou faire communiquer ces ressources existent comme : les systèmes distribués, les systèmes mobiles, les systèmes pervasifs. Cependant, les applications logicielles pour l'exploitation (recherche, filtrage, analyse, stockage, découverte de nouvelles connaissances, etc.) optimale de ces ressources sont difficiles à mettre en œuvre du fait de la complexité et de l'hétérogénéité des ressources.

Si on prend le cas de l'accès à l'information, on remarque une grande diversité au niveau des informations mises à disposition : documents structurés, non structurés ou semi-structurés, documents mono ou multimédia, etc. Ces informations ont des thématiques diverses, leur structure n'est pas toujours clairement définie et elles proviennent généralement de différentes sources : bases de connaissances issues d'internet et du web, d'intranets, de workflows, etc. De même, on observe aussi une population très hétérogène d'usagers de ces informations. Leurs besoins, préférences, objectifs, connaissances et autres sont variés. De plus, ces usagers sont caractérisés généralement par l'inconstance de leurs besoins, préférences et objectifs qui évoluent régulièrement et qui peuvent même changer radicalement dans le temps.

Dans le but de réduire l'hétérogénéité des ressources, des solutions ont été élaborées et consistent, en général, à définir des modèles génériques dont le but est d'offrir un référentiel et un cadre homogène de description de ressources et de faciliter ainsi l'analyse de modèles de ressources différents. Cette analyse pourra être basée sur des modèles génériques (modèles d'une ou de plusieurs classes de ressources) et non plus sur des modèles spécifiques de ressources (modèles d'une instance de ressource). Cependant, des dispa-

rités (hétérogénéités) entre instances de modèles génériques demeurent et réduisent ainsi les possibilités de coopération entre ressources différentes.

Il y a donc un réel besoin d'interopérabilité (ou coopération) entre descriptions de ressources (appelées profils) pour la réalisation d'une tâche spécifique. Dans un contexte collaboratif par exemple, on peut être amené à rechercher des utilisateurs similaires pour un partage d'informations en fonction du contenu et/ou des jugements effectués sur ces dernières. De plus, on peut souhaiter échanger ces informations via des dispositifs mobiles (PDA, téléphone portable, etc.) ou autres. Pour cela, on doit généralement combiner différentes descriptions d'utilisateurs, d'informations et de dispositifs matériels qui vont constituer ainsi les ressources clés de l'application. L'application doit manipuler ces différentes ressources (usagers, informations, matériels, logiciels, etc.) pour répondre de façon personnalisée ou autres aux demandes de chaque usager ou groupe d'usagers. Les applications d'accès à des ressources sont donc confrontées aux problèmes d'intégration (ou coopération) de ressources hétérogènes pour la réalisation d'une tâche.

Ainsi, afin de faciliter cette intégration de ressources hétérogènes, il est nécessaire de définir des modèles de ressources qui aient à la fois des propriétés d'extensibilité, de flexibilité, de ré-utilisabilité et d'interopérabilité. Pour cela, une sémantique doit être associée à la description des ressources. Cette sémantique doit permettre de faire coopérer des modèles différents de façon cohérente. Par exemple, des utilisateurs qui s'intéressent à des *informations récentes* peuvent avoir chacun leur propre définition de cette notion. Ainsi, les résultats des recherches de ces derniers doivent être fortement liés à l'interprétation de la sémantique qu'ils auront associée à cette notion. Pour ce faire, on peut se baser sur des langages de métadonnées.

La solution que nous avons adoptée dans nos travaux est celle de la proposition d'un modèle de profil (description de ressource) qui possède une double dimension : *générique et sémantique*. L'aspect générique permet d'avoir un modèle sous-jacent homogène et la sémantique va atténuer les disparités qui subsistent au niveau des instances de profils pour une plus grande interopérabilité de ces derniers, dans des applications. Notons que la sémantique est également définie de façon générique et est instanciée par des métadonnées sur des exemples de profils. Par la suite, nous définissons une méthode d'exploitation flexible de profils basée sur l'analyse de la sémantique de profils et l'appariement de ces derniers pour l'accès à des ressources.

Dans le chapitre 1, nous présentons une étude bibliographique sur les différentes techniques d'*accès à des ressources* et sur l'utilisation de profils dans ces techniques. L'objectif de ce chapitre est de montrer la richesse et la diversité de ces techniques. Cette richesse et cette diversité sont liées à la combinaison de différents types de ressources pour la réalisation d'une tâche spécifique. De plus, à travers une étude comparative de certains systèmes d'accès existants (systèmes de personnalisation en l'occurrence), nous montrons l'intérêt de la construction d'un modèle générique de profil et celui de

la définition d'une méthode d'appariement de profils basées sur une combinaison d'appariements élémentaires. Nos propositions porteront, par la suite, sur ces aspects.

Dans le chapitre 2, nous présentons un état de l'art sur la *description de ressources*. L'objectif est de réaliser un panorama des différents modèles de ressources existants et d'analyser les possibilités offertes par chacun d'eux. Plusieurs initiatives ont proposé des modèles de ressources et la plus fédératrice à l'heure actuelle est celle regroupée sous l'appellation de *web sémantique*. Ce chapitre nous permet également de justifier de la dimension sémantique de nos propositions aussi bien concernant la modélisation que l'exploitation (analyse ou interprétation et appariement) de profils.

Dans le chapitre 3, nous présentons notre contribution concernant la *modélisation flexible de profils pour l'accès à des ressources*. Nous y décrivons un cadre générique pour l'accès à des ressources à travers :

1. *une architecture d'accès à des ressources* dans laquelle tout élément (ou composant) de l'architecture est décrit de façon détaillée par un profil. L'originalité de cette architecture se situe au niveau de son aspect générique et des nombreuses possibilités d'interactions entre profils complémentaires qu'elle offre ;
2. *un modèle générique de profil* pour la description de classes de profils non pré-définies. Ce modèle générique intègre une dimension sémantique afin de faire interopérer des profils décrits dans des langages différents. Cette sémantique va jouer le rôle de *référentiel commun d'interprétation* de profils.

Dans le chapitre 4, nous décrivons notre contribution relative à l'*exploitation flexible de profils pour l'accès à des ressources*. Pour cela, nous proposons :

1. *un algorithme d'analyse de la sémantique de profils* afin de déduire automatiquement les éléments de sémantique compatible que l'on va pouvoir comparer entre ces profils ;
2. *une méthode d'appariement de profils* qui va permettre de mesurer une degré de ressemblance entre profils en combinant des appariements élémentaires clairement identifiés par l'algorithme d'analyse de profils.

Dans le chapitre 5, nous présentons les expérimentations que nous avons menées pour valider nos propositions à travers :

1. la description du cadre général d'expérimentation ;
2. la présentation de l'outil de construction, de visualisation et d'analyse sémantique de profils que nous avons implémenté ;
3. une évaluation de l'algorithme d'analyse ainsi que de la méthode d'appariement de profils proposés.

Enfin, nous terminons par une analyse de nos contributions et une présentation de nos perspectives de recherche.

PARTIE 1 : État de l'art

Chapitre 1

Accès à des ressources

1.1 Introduction

L'hétérogénéité des ressources existantes (information, usagers, environnements matériels et logiciels, etc.) a soulevé les problèmes de description de ressources et de définition de méthodes d'exploitation de ces descriptions pour pouvoir rechercher, sélectionner, trier ou même partager des ressources. L'ensemble de ces opérations est regroupé sous l'appellation d'*accès à des ressources*. L'accès à des ressources est ici une vue plus large de l'*accès à l'information* (terme plus usité dans la littérature) où les ressources ne sont plus limitées aux informations et aux usagers mais peuvent être étendues à toutes sortes d'éléments (personnes, choses, actions).

Dans la littérature, plusieurs outils d'accès à des ressources ont été développés pour aider l'utilisateur à retrouver ce qu'il recherche. L'évaluation de la pertinence [1] des résultats est au cœur de la mise en œuvre de ces différents outils. Certains de ces outils visent à évaluer la pertinence relativement à un usager ou groupe d'usagers afin d'améliorer une *pertinence système* qui serait évaluée de la même façon pour tous les utilisateurs. Le but est de se rapprocher le plus possible de la pertinence personnelle (ou attentes) des utilisateurs : on parle alors de *personnalisation* ou de *techniques d'accès personnalisé*.

Le développement du web a accentué le besoin de techniques nouvelles pour aider les utilisateurs à trouver ce qu'ils recherchent mais aussi pour faire savoir qu'une ressource existe. Dans ce chapitre, nous présentons un panorama de différentes techniques *d'accès à des ressources* et nous décrivons plus en détail *l'accès à l'information* (personnalisé ou non) au travers de modèles de représentation et de techniques d'appariement. Ces modèles et techniques sont illustrés principalement pour des ressources de types : informations textuelles (ou documents textuels), requêtes et usagers mais peuvent

1. *la pertinence* peut se définir comme un degré de similarité entre le besoin de l'usager et les résultats qui lui sont restitués

être généralisés à d'autres types de ressources. Nous effectuons également une étude comparative de différents systèmes de personnalisation pour situer nos travaux de recherche par rapport à la revue de littérature effectuée.

1.2 Panorama des techniques d'accès à des ressources

Les systèmes de production, de collection, de traitement, d'enregistrement et de diffusion d'informations, communément appelés *systèmes d'information*, se déclinent, de nos jours, en plusieurs catégories [SM03] :

- *les systèmes distribués* qui sont des ensembles d'entités autonomes de calculs interconnectées et qui peuvent communiquer. On peut citer par exemple : un réseau physique de machines (PC, PDA), un logiciel avec plusieurs processus sur une même machine (processus légers) ;
- *les systèmes mobiles* qui sont des systèmes distribués permettant la gestion de la mobilité de certaines entités (PDA, téléphone portable, etc.) ;
- *les systèmes pervasifs* qui sont des systèmes mobiles permettant la gestion de l'ubiquité et du contexte. Les systèmes ubiquitaires sont des systèmes dont les applications sont accessibles de n'importe où (lieu, dispositif de connexion) [SLP04]. La gestion du contexte dans des applications, communément appelées *context-aware applications*, est la prise en compte du contexte d'exécution de ces dernières [BHH04]. Ce contexte peut être relatif à la situation géographique de l'usager, au type de dispositif de connexion de l'usager, etc.

Les composantes de ces systèmes d'information sont variés : usagers, matériels, logiciels, réseaux. Ces composantes vont constituer des ressources clés dans de nombreuses applications d'accès à des ressources :

1. *l'accès à l'information* qui à travers des processus de recherche, de filtrage ou de recommandation, consiste à décrire le besoins de l'usager et à rechercher les informations qui sont similaires à ce besoin [Rij79] [BYRN99]. Notons le cas particulier des *interfaces d'accès à l'information* où les critères de la recherche sont pré-définis pour une restitution de résultats qui correspondent aux critères renseignés par l'usager sur l'ensemble des critères disponibles [CAF00] ;
2. *les entrepôts de données (respectivement de documents)* qui permettent de stocker dans un répertoire partagé, d'interroger et de procéder à des analyses multidimensionnelles de données [KR03] (respectivement de documents [Khr04]) ;
3. *la navigation* dans une collection d'informations qui est généralement très fortement liée à des techniques *de visualisation* (hiérarchies, graphes, usage de couleurs et de formes, usage du zoom, etc.) [LK03]

[CJ01] et *d'annotations* [KKPS02] qui ont pour but d'aider l'utilisateur à parcourir de façon plus efficace la collection.

Par ailleurs, notons également l'usage de plus en plus fréquent sur les pages web de flux RSS ou fil RSS [BDBD+00] (*RSS feed* en anglais), qui est un format de *syndication* de contenu Web. RSS est le sigle de *Really Simple Syndication* (syndication vraiment simple) ou de *Rich Site Summary* (Sommaire d'un site enrichi). Un fil RSS est un fichier XML dynamique dont votre lecteur RSS (Safari, Mozilla Firefox, Mozilla Thunderbird ou encore Opera) affiche le contenu qui est mis à jour en permanence. Ce système est très utilisé pour diffuser les nouvelles des sites d'information (actualité, sciences, informatique, etc.) ou des blogs [2], ce qui permet de consulter ces dernières sans visiter le site ;

4. *les web services* qui sont des technologies permettant à des applications de dialoguer à distance via Internet et ceci indépendamment des plates-formes et des langages sur lesquelles elles reposent. Le but des services web est donc de faciliter l'interopérabilité entre différentes applications. Comme exemple de service web, on peut citer un *service d'agence de voyages* qui vend des formules de vacances et utilise d'autres services web de carte de crédit, de réservation d'hôtel, de compagnie aérienne. Pour faire interopérer ces applications, les services web s'appuient sur un ensemble de protocoles standardisant les modes d'invocation mutuels de composants applicatifs. Ceci est possible notamment grâce aux langages WSDL (Web Service Description Language) [CMRW05] pour la description de services web, SOAP (Simple Object Access Protocol) [Ved05] [Mit03] pour la communication avec un service web et autres extensions de ces langages comme AWSDL [LV05]. Ces langages sont basés sur XML et ses technologies satellites. Notons également l'usage de UDDI [3] (Universal Discovery Description and Integration) pour l'enregistrement et la publication de services web.

   De plus, les web services sont de plus en plus basés sur des architectures de type *peer-to-peer* (P2P ou pair à pair). Le P2P [Leu02a] [Leu02b] est un modèle de communication dans lequel chaque partie peut initialiser une session de communication et dispose des mêmes capacités (une application peut être simultanément client et/ou serveur) ;

5. etc.

L'hétérogénéité des ressources mises à disposition soulève de nouveaux

2. Un weblog est un site web sur lequel une ou plusieurs personnes s'expriment librement, sur la base d'une certaine périodicité. Dans son usage francophone comme anglophone, weblog est fréquemment raccourci en blog

3. *cf.* url(http ://www.uddi.org)

challenges. De nombreuses applications ont vu le jour pour fournir différents outils de stockage (hiérarchies de signets, par exemple), d'analyse (outils de visualisation d'informations sous forme de graphes ou de tables, outils de découverte de nouvelles connaissances, etc.) et d'interrogation (systèmes des questions réponses, systèmes de gestion de bases de données, entrepôts de données ou de documents, recherche d'information basée sur des requêtes en langage naturel, etc.) de collections de ressources. De nombreuses problématiques sont liées à l'élaboration de ces outils : représentation des ressources à manipuler, appariement de ressources, combinaison de ressources, principe de sélection des résultats à présenter à l'usager, modèles de restitution des résultats, évaluation de ces outils. Dans la section suivante, nous décrivons les modèles de représentation et d'appariement de ressources dans les techniques d'accès à l'information. Les modèles sont principalement illustrés avec des ressources de type document et utilisateur mais l'on peut utiliser le même raisonnement avec d'autres types de ressources.

1.3 Techniques d'accès à l'information

Les techniques d'accès à l'information permettent à un individu d'obtenir des informations répondant à ses besoins. Nous pouvons les regrouper en deux grands groupes :

- celles qui reposent sur une approche *service au comptoir* ou *pull* et qui consistent à renvoyer des informations répondant à une demande explicite d'un individu. C'est le cas de la Recherche d'Information (RI) ;
- celles qui reposent sur une approche *service à domicile* ou *push* et qui consistent à renvoyer automatiquement à un individu des informations qui pourraient l'intéresser, sans qu'il n'en ait fait explicitement la demande. Pour cela, on utilise une représentation à priori de l'usager. C'est le cas du Filtrage (et/ou Recommandation) d'Information (FI).

Les sections suivantes, 1.3.1 et 1.3.2, présentent différentes techniques d'accès à l'information au travers des processus de recherche (pull) et de filtrage d'information (push).

1.3.1 Recherche d'Information

Le processus de Recherche d'Information repose sur l'expression du besoin d'un individu au travers d'une requête formulée dans un langage libre plus ou moins structuré. En réponse à cette requête, un appariement est réalisé entre les termes (ou mots-clés) d'indexation de la requête et ceux des informations pré-indexées par le système. La recherche d'information est principalement basée sur le principe d'un appariement optimal, de type vectoriel (*cf.* section 1.4.3.2) ou probabiliste (*cf.* section 1.4.3.3) [Rij79] [BYRN99]. Enfin, le système propose traditionnellement à l'individu les informations

pertinentes sous forme d'une liste ordonnée selon leur degré de pertinence décroissant.

Cependant en Recherche d'Information, l'intention réelle de l'utilisateur n'est pas toujours évidente dans sa manière de formuler sa requête et cela peut générer des ambiguïtés au niveau du sens des mots qu'elle contient. De nombreuses solutions existent pour préciser le sens d'une requête et on peut citer en particulier :

— les techniques d'expansion de requêtes via des *thésaurus*[4] [XC96] ou des *ontologies*[5] [Voo94] [BAGB03] ;

— les techniques de reformulation de requêtes dans des *processus de personnalisation de recherche* à travers une recherche personnalisée individuelle ou collaborative.

La *recherche individuelle personnalisée* va consister à :

1. utiliser des jugements de pertinence (ou non-pertinence) d'un utilisateur sur un ensemble d'informations pour reformuler sa requête et affiner ainsi la recherche. C'est la méthode de réinjection de pertinence ou relevance feedback [Roc71] [BCSD99] [KV02] ;
2. utiliser la notion de profil long terme (profil construit sur une période relativement importante) des besoins de l'usager et la notion de profil court terme (profil construit sur une période courte) de ses besoins, pour aider à l'interprétation de requêtes afin de réévaluer et de réordonnancer les résultats d'une recherche [BBB03a] [BBB04] ;
3. utiliser la notion de contextualisation (profil court terme) et d'individualisation (caractéristiques propres à l'usager quel que soit le contexte) pour la personnalisation de la recherche [PSC+02] ;
4. etc.

La *recherche collaborative* [KGB98] quant à elle va consister à utiliser la notion de groupe pour répondre aux besoins des utilisateurs. Ainsi, on va pouvoir reformuler la requête d'un utilisateur avec les termes des documents validés par des utilisateurs de profils similaires au sien, lesquels documents ont été obtenus suite à des requêtes ou situations de recherche similaires [JRP01].

4. Un thesaurus est une sorte de dictionnaire hiérarchisé ; un vocabulaire normalisé sur la base de termes génériques et de termes spécifiques à un domaine. Il ne fournit qu'accessoirement des définitions, les relations des termes et leur choix l'emportant sur les significations *[fr.wikipedia.org/wiki/Thésaurus]*

5. En informatique, une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être : des relations sémantiques ; des relations de composition et d'héritage (au sens objet) *[fr.wikipedia.org/wiki/Ontologie_(informatique)]*

1.3.2 Filtrage d'Information

Alors que la Recherche d'Information (RI) est une tâche très interactive, celle du Filtrage d'Information (FI) est relativement passive [BC92] car l'utilisateur ne formule pas explicitement ses besoins au travers d'une requête (ou expression d'un besoin ponctuel) comme c'est le cas en RI. En Filtrage d'Information, on utilise plutôt une représentation de l'utilisateur appelé *profil utilisateur* pour lui envoyer des informations. Ces informations proviennent généralement d'un flux dynamique. Elles sont ensuite comparées aux différents profils disponibles pour déterminer ceux auxquels elles correspondent. Il existe plusieurs méthodes de filtrage [MLR03] :

- *le filtrage cognitif ou basé sur le contenu* qui utilise la description du contenu des informations pour déterminer à quels profils utilisateurs elles correspondent [Lie95] [Mla96] [PMB96]. Le profil utilisateur, en filtrage cognitif, décrit les centres d'intérêt durables ou récurrents de l'individu qui sont représentés communément par une liste de mots-clés pondérés [Kor97]. Ce profil est obtenu manuellement ou automatiquement en indexant (*cf.* section 1.4.2.1), par exemple, les informations sauvegardées par l'utilisateur lors de ses sessions de recherche [Che02] ;
- *le filtrage social ou collaboratif* qui utilise les jugements (ou feedback) d'un ensemble d'utilisateurs concernant un ensemble d'informations pour effectuer des recommandations. On utilise une mesure de similarité entre jugements d'individus pour déterminer si une information correspond à un individu donné [GNOT92] [KMM+97] [RP97]. La description du contenu réel des informations est ignorée. Le tableau Tab. 1.1 représente des exemples de profils de jugements utilisateurs en filtrage collaboratif. Le «+» signifie que l'information intéresse l'utilisateur, le «-» qu'elle ne l'intéresse pas et le « ?» que l'information n'a pas encore été jugée par un utilisateur et pourrait donc être recommandée à ce dernier. Les utilisateurs *utilisateur1* et *utilisateur3* peuvent être considérés comme similaires car ils ont effectué les mêmes jugements. On peut donc recommander le document *document4* à l'utilisateur *utilisateur3* car l'utilisateur *utilisateur1* l'a déjà jugé comme étant intéressant ;

Utilisateurs	Jugements de documents			
	document1	document2	document3	document4
utilisateur1	+	-	+	+
utilisateur2	-	-	-	?
utilisateur3	+	-	+	?

Table 1.1 – Exemples de profils de jugements utilisateurs en filtrage collaboratif

Notons également l'existence de méthodes de filtrage collaboratif orientées vers les communautés qui ne se focalisent plus sur le calcul de corrélation et de prédiction (similarités entre jugements pour faire des recommandations) mais qui intègrent des actions diverses des membres de la communauté [DBGLN04] : envoi (recommandation) de documents, déclenchement périodique ou après une évaluation ou à la demande du système de filtrage collaboratif ;

— *le filtrage démographique* qui utilise les données démographiques des utilisateurs (sexe, âge, profession, ville d'origine, etc.) pour les classer par groupes [Kru97] [NDB06] et leur faire des recommandations. Pour cela, on se base sur une catégorisation des informations en fonction des données démographiques des individus. Cette catégorisation permet de déterminer quel type d'information est apprécié par un type d'utilisateur particulier (relativement à leurs données démographiques). Pour cela, on peut procéder à une catégorisation manuelle ou on peut se baser, par exemple, sur les jugements des utilisateurs pour déduire le type d'individu (groupe) auquel correspond une information [Paz99]. Le tableau Tab. 1.2 représente des exemples de profils utilisateurs en filtrage démographique. Le «+» signifie que l'information intéresse l'utilisateur, le «-» qu'elle ne l'intéresse pas et le « ?» que l'information n'a pas encore été jugée par un utilisateur et pourrait donc être recommandée à ce dernier. On peut déduire du tableau Tab. 1.2, trois groupes de personnes du fait de la similarité de leurs jugements : les femmes de moins de 18 ans, les femmes de plus de 25 ans et les hommes de moins de 18 ans. On peut donc faire des recommandations relativement aux jugements effectués dans les groupes identifiés. Notons que l'on ne peut rien dire concernant les utilisateurs *utilisateur7* et *utilisateur8* car leurs jugements sont complètement disjoints.

Utilisateurs	Données démographiques		Jugements de documents		
	Sexe	Âge	document1	document2	document3
utilisateur1	F	15	+	-	+
utilisateur2	F	18	+	-	?
utilisateur3	F	25	-	-	+
utilisateur4	F	28	-	-	?
utilisateur5	M	16	+	-	+
utilisateur6	M	18	+	-	?
utilisateur7	M	25	-	-	-
utilisateur8	M	32	+	+	?

Table 1.2 – Exemples de profils utilisateurs en filtrage démographique

Ces approches ne sont pas exclusives et différentes méthodes hybrides, combinant ces différents types de filtrage, ont été développées [GSK+99] [Paz99]. L'utilisation des approches hybrides permet d'améliorer la pertinence des résultats des systèmes de filtrage en palliant certaines limites des types de filtrage présentés précédemment comme [BS97] : la sur-spécialisation en filtrage basé sur le contenu ; l'obtention des jugements qui est une tâche coûteuse pour les utilisateurs, etc.

Comme exemples d'approches de filtrage hybride, on peut citer :

— *le filtrage collaboratif via le contenu* [Paz99] qui va permettre de déterminer des similarités entre utilisateurs via leur profil de besoins (centres d'intérêt), construit à partir du contenu des informations qu'ils ont jugées. Ainsi, pour identifier des groupes d'utilisateurs on ne se basera plus uniquement sur une mesure de similarité entre jugements utilisateurs. L'intérêt particulier de ce type de filtrage hybride est qu'il va permettre de faire des recommandations à un nouvel utilisateur, en l'affectant à un groupe via son profil des besoins. En filtrage collaboratif pur, il aurait fallu attendre que cet utilisateur ait effectué des jugements (sur des informations) pour pouvoir l'associer à d'autres utilisateurs et lui faire des recommandations. Cela nécessite en général un certain temps : c'est le problème de l'*entonoir* (ou boîte noire ou démarrage à froid) qui se pose généralement pour le démarrage d'un filtrage collaboratif. Notons que ce problème d'entonoir peut être détecté via des approches de contrôle de la qualité des systèmes de filtrage collaboratif qui mesurent le taux d'utilisation (jugements des usagers entre autres) de ces systèmes par les usagers [GLBD04] ;

— *les approches réclusives* [Yag02] qui sont basées sur la recherche d'une similarité entre objets en comparant leur description respective (ou contenu respectif). Ainsi, on pourra recommander une information si sa description est similaire à une autre information qui elle a déjà été validée (c'est-à-dire jugée intéressante) par l'utilisateur. L'intérêt de cette approche hybride est que l'on va pouvoir recommander une information qui n'a pas encore été jugée. En filtrage collaboratif pur, il faut attendre qu'une information soit jugée par au moins un utilisateur pour pouvoir la recommander ;

— etc.

Pour résumer, les différentes techniques d'accès à l'information partagent le même objectif qui est d'aider l'utilisateur à obtenir les informations qu'il recherche ou dont il peut avoir besoin. Pour cela, on doit décrire les informations manipulées par les processus de recherche et de filtrage d'information. Cette description des informations est désignée sous le nom de profil (ou modèle ou représentation). L'appariement (ou mesure de similarité) entre profils va permettre de décider de la restitution ou non des informations aux usagers.

Le filtrage cognitif ou basé sur le contenu peut-être considéré comme le processus dual de la recherche d'information, comme l'illustre la figure FIG. 1.1 décrivant les *modèles en U* de la RI et du FI. Cependant, quand on est dans un *contexte collaboratif* où plusieurs utilisateurs concourent à la restitution d'un résultat donné, les appariements ou comparaisons de profils ne se font plus uniquement entre les informations mises à disposition et les besoins des usagers de ces informations mais aussi entre informations, entre usagers et jugements d'usagers. C'est le cas typique de la recherche collaborative, du filtrage collaboratif, du filtrage démographique et des approches hybrides d'accès à l'information qui utilisent ces techniques.

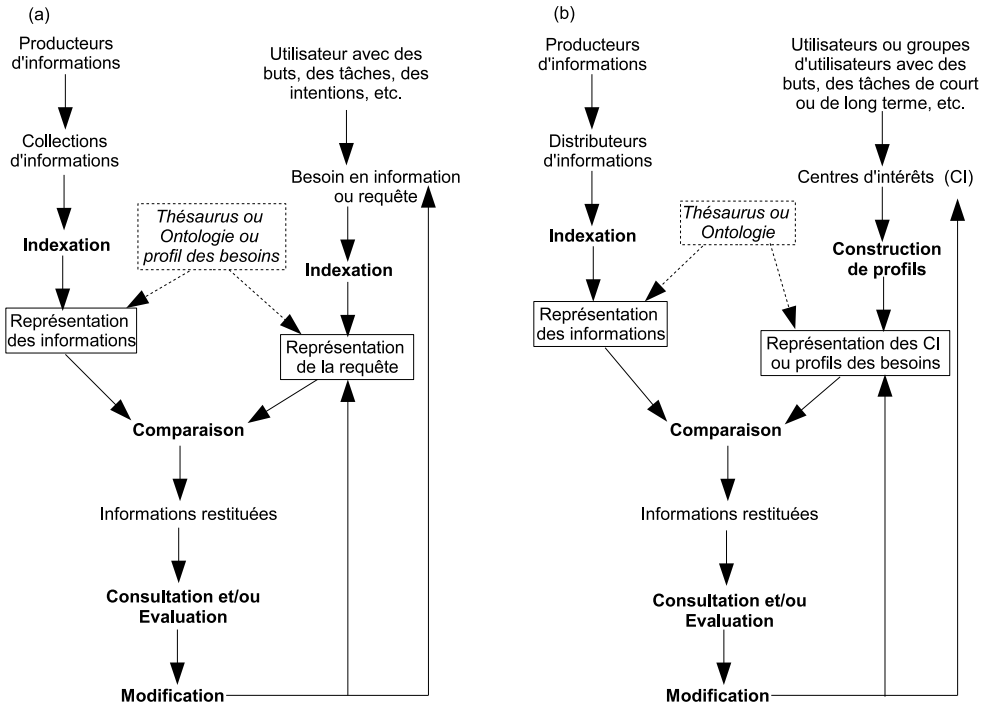

FIGURE 1.1 – (a) Modèle général en U de la Recherche d'Information (b) Modèle général en U du Filtrage d'Information basé sur le contenu

Ainsi, on va distinguer, dans les techniques d'accès à l'information, différentes combinaisons de profils pour effectuer des appariements :

- représentation (ou profil) d'une requête reformulée ou pas et représentation (ou profil) d'une information à restituer ;
- représentation d'une requête a et représentation d'une requête b ;
- profil des besoins d'un utilisateur a et profil des besoins d'un utilisateur b ;
- profil des besoins d'un utilisateur et profil d'une information mise à disposition ;

— profil d'une information a mise à disposition et profil d'une information b mise à disposition ;
— profil des jugements d'un utilisateur a et profil des jugements d'un utilisateur b ;
— profil des données démographiques d'un utilisateur a et profil des données démographiques d'un utilisateur b.

Le tableau TAB. 1.3 représente pour chaque technique d'accès à l'information, les combinaisons de profils possibles pour les appariements.

Techniques d'accès	Combinaisons de profils pour les appariements
Recherche individuelle	1. représentation de requête reformulée ou pas et profils des informations mises à disposition.
Recherche collaborative	1. représentation de requête et profils des informations mises à disposition 2. représentation de la requête d'un utilisateur et représentations de requêtes d'autres usagers ; 3. profil des besoins d'un utilisateur et profils des besoins d'autres usagers.
Filtrage cognitif	1. profil des besoins d'un utilisateur et profils des informations mises à disposition.
Filtrage collaboratif	1. profil des jugements d'un utilisateur et profils des jugements d'autres usagers.
Filtrage démographique	1. profil des données démographiques d'un utilisateur et profils des données démographiques d'autres usagers ; 2. profil des jugements d'un utilisateur et profils des jugements d'autres usagers.
Filtrage hybride	1. profil des besoins d'un utilisateur et profils des besoins d'autres usagers ; 2. profil des jugements d'un utilisateur et profils des jugements d'autres usagers ; 3. profil d'une information et profils d'autres informations mises à disposition.

TABLE 1.3 – Techniques d'accès à l'information et combinaisons de profils associées pour les appariements

Dans la section suivante, nous présentons avec plus de détails la notion de profil telle qu'elle est utilisée dans les différentes techniques d'accès à l'information.

1.4 Profils : types, méthodes de construction, modèles de représentation et appariements

De façon générale, le profil d'un objet est un ensemble de caractéristiques permettant d'identifier ou de représenter cet objet. Nous avons étudié les profils dans les techniques d'accès à l'information sous différents angles : types, méthodes de construction, modèles de représentation et appariements de profils.

1.4.1 Types de profils

Les profils utilisés dans les techniques d'accès à l'information sont de nature très variée et on peut les classifier en deux grands groupes :

— ceux relatifs aux informations mises à disposition ;

— ceux relatifs aux utilisateurs de ces informations.

1.4.1.1 Profils relatifs aux informations mises à disposition

Le profil des informations mises à disposition correspond à la description de ces dernières qui est souvent réduite, en RI ou FI, à une liste de mots-clés pondérés décrivant le contenu effectif de ces informations. Plusieurs travaux permettent actuellement de décrire les informations en utilisant également d'autres critères que ceux liés à leur contenu effectif. On peut citer par exemple les métadonnées du Dublin Core [6], pour la description de ressources. Nous pouvons également citer les travaux de Lainé-Cruzel [LC99] qui permettent de définir des propriétés liées à l'ensemble d'un document (profession de l'auteur, type de document, etc.) ainsi que celles relatives à des parties de documents (type d'unité documentaire, forme discursive, style, etc.) afin de restreindre les documents pertinents (du point de vue du sujet dont ils traitent) aux seuls documents exploitables et réellement utilisables, du point de vue de l'usager. De même, une liste non exhaustive de métadonnées pour l'annotation qualitative de documents est donnée par Berti-Equille [BE02] [BE03] dans le contexte de la recommandation multi-critères.

Notons que les informations à restituer par les processus de RI ou FI peuvent être de différents niveaux de granularité : *collections* de documents [GGMT99], *documents* et *granules ou parties* de documents [INEX] [7], [TREC] [8]. De plus, les profils de ces informations peuvent être composés soit uniquement de mots-clés pondérés décrivant leur contenu, soit de mots-clés pondérés et d'autres caractéristiques (ou critères) souvent décrites par des métadonnées (*cf.* Fig. 1.2).

6. *cf.* http ://dublincore.org/documents/dces/
7. *cf.* http ://qmir.dcs.qmw.ac.uk/INEX/index.html
8. *cf.* http ://icl.pku.edu.cn/icl_groups/iregroup/trec/trec.nist.gov/

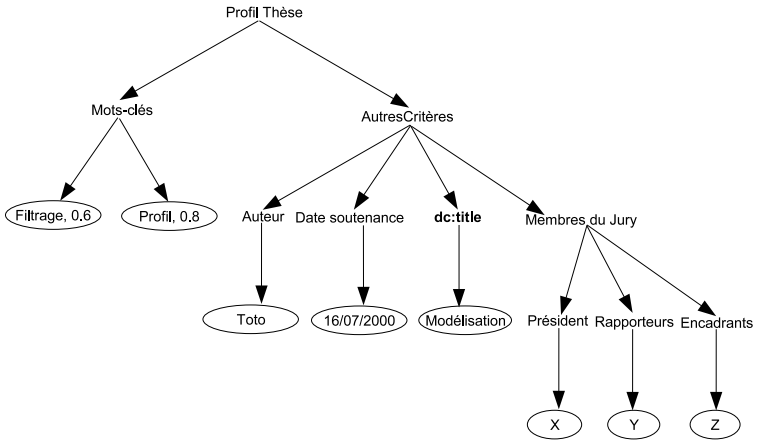

FIGURE 1.2 – Exemple de profil de thèse

1.4.1.2 Profils relatifs aux utilisateurs

Le profil utilisateur est une banque de données qui regroupe les différents sujets ou thèmes susceptibles d'intéresser un utilisateur donné [BMRM96]. Il peut également être vu comme une collection d'informations diverses sur l'utilisateur (*cf.* FIG. 1.3). Cette collection va permettre d'illustrer un ensemble de caractéristiques avec des valeurs associées [Mar02] contenant par exemple ce que l'utilisateur préfère, ses thèmes ou centres d'intérêts, ses données démographiques, etc.

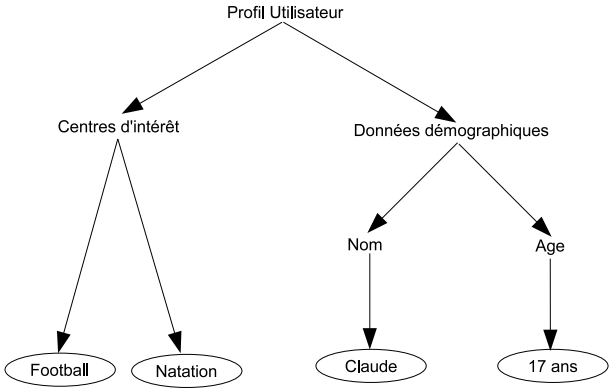

FIGURE 1.3 – Exemple de profil utilisateur

Les utilisateurs peuvent être étudiés également selon différents niveaux de granularité [PSC+02] : *individu*, *groupe* d'utilisateurs ou *population* représentant tous les utilisateurs. Les profils utilisateurs peuvent donc décrire des individus ou des groupes d'individus. Ils peuvent également être de différents

types, chacun décrivant une facette (ou vue) de l'utilisateur comme :

- *les profils court terme ou long terme* qui sont liés au temps d'apprentissage considéré pour l'obtention des informations du profil [WIY99] [MT02a] [MT02b]. Le profil court terme peut être, par exemple, le profil de l'utilisateur durant une session de recherche. Il peut être assimilé au contexte de recherche de ce dernier. Le profil long terme peut correspondre, quant à lui, au profil (description) de l'utilisateur construit sur plusieurs sessions de recherche. Ainsi, le profil court terme va permettre de préciser l'objectif à court terme d'un utilisateur tandis que le profil long terme permet de préciser l'objectif à priori de l'utilisateur indépendamment de sa session de recherche. Le profil court terme est très important car il permet de se rendre compte des changements de centres d'intérêt ou de préférences d'un utilisateur pour mieux s'adapter à celui-ci. En résumé, le profil long terme est obtenu après un temps d'apprentissage important contrairement au profil court terme ;
- *les profils positifs ou négatifs* qui permettent de préciser ce que l'utilisateur aime ou recherche et ce qu'il n'aime pas ou ne recherche pas [HKNH00]. La notion de profil négatif est née du fait que la plupart des systèmes de filtrage d'information emploient des valeurs de seuil assez élevées pour éviter de sélectionner des documents non pertinents. Cette approche engendre dans certains cas, la non-sélection de documents pertinents dont la valeur de similarité avec le profil est inférieur au seuil. Pour résoudre ce problème, Hoashi [HKNH00] introduit la notion de profil négatif. Pour cela, il effectue d'abord un premier filtrage avec le profil positif de l'utilisateur pour une valeur de seuil pas très élevée et par la suite, un second filtrage avec le profil négatif de ce dernier.

En résumé, la structure d'un profil quelconque, en RI ou FI, peut être composée :

- *d'un seul critère* qui est lié au contenu des informations à savoir : *mots-clés* pour les profils d'informations et *centres d'intérêt* pour les profils utilisateurs. Pour ces derniers, on parle généralement de *profils des besoins utilisateurs* ;
- *de plusieurs critères*. Dans ce cas, on a un profil étendu (*cf.* FIG. 1.2 et FIG. 1.3).

La typologie structurelle d'un profil peut donc être mono-critère [PMB96] [Amm03] ou multi-critères [LC99] [BE02] [Tch04a] [AVF06]. Par ailleurs, selon ce que les profils représentent au niveau sémantique, on va distinguer dans la littérature différents types de profils. Les tableaux TAB. 1.4 et TAB. 1.5 représentent des typologies sémantiques de profils.

Notons que les différents *types de profils d'informations mises à disposition*, du tableau TAB. 1.4, peuvent être utilisés pour décrire soit des collections de documents (collections CLEF 2002, etc.), soit des documents, soit

Types de profils des informations mises à disposition	Sémantique
profil du contenu de l'information	description du contenu de l'information (mots-clés ou termes d'indexation)
profil étendu de l'information	description du contenu de l'information augmentée d'autres caractéristiques : langue, taille, ...

TABLE 1.4 – Typologie sémantique de profils d'informations mises à disposition

des granules ou parties de documents (paragraphes, sections, etc.). De même, les différents *types sémantiques de profils utilisateurs*, du tableau TAB. 1.5, peuvent être utilisés pour décrire soit un individu, soit un groupe d'individus.

Types de profils utilisateurs	Sémantique
profil des besoins	centres d'intérêt
profil des jugements	types de jugements : pertinence, non-pertinence, ...
profil étendu	combinaison de différents critères de description : centres d'intérêts, jugements, ...
profil court terme	profil construit sur une période courte : deux heures, une session de recherche, ...
profil long terme	profil construit sur une période assez longue : plusieurs sessions de recherche, ...
profil positif	profil représentant ce qui est recherché par l'usager
profil négatif	profil représentant ce qui n'est pas recherché par l'usager

TABLE 1.5 – Typologie sémantique de profils utilisateurs

Dans la section suivante, nous présentons les principales méthodes utilisées pour construire des profils en RI et FI.

1.4.2 Méthodes de construction de profils

En général, on distingue deux groupes de méthodes de construction de profils :

— *les méthodes manuelles* où l'humain renseigne lui-même les valeurs de certains critères descriptifs de profils ;

— *les méthodes automatiques ou semi-automatiques* de construction (ou d'apprentissage) de profils comme : l'indexation automatique, le profiling, les approches par stéréotypes, etc.

1.4.2.1 Indexation automatique

Les mots-clés *significatifs* décrivant le contenu d'une information et leurs poids sont obtenus en général par une opération d'indexation [Rij79] [SM83]. L'indexation va permettre de déterminer les mots-clés décrivant le contenu des informations. Elle est applicable à différents niveaux de granularité des informations (document, granules de document, collections de documents). L'indexation est également utilisée pour déduire les centres d'intérêts (ou mots-clés décrivant des besoins) des utilisateurs à partir des informations manipulées (sites visités, informations jugées ou sauvegardées, etc.) par ces derniers durant des sessions de recherche.

Les différentes étapes, généralement suivies, pour l'indexation sont :

1. *le nettoyage* qui consiste à supprimer tout ce qui peut avoir des répercussions inattendues sur les traitements qui vont suivre. Par exemple, dans le cas de documents textuels, on devra supprimer : les images, les formules mathématiques, etc. ;
2. *la segmentation* qui consiste à découper le texte en unités lexicales ou *termes* (mots simples ou groupes de mots) ;
3. *la suppression des mots outils* qui consiste à éliminer les mots qui n'ont pas de sens propre comme les articles, les adverbes, etc. Pour cela, on utilise des listes pré-définies de ces mots outils ou mots vides [9] ;
4. *l'analyse morphologique des termes* qui recouvre deux notions à savoir [Dai98] [10] :
   (a) *la morphologie flexionnelle* où la forme du mot change du fait des accords grammaticaux ou de la conjugaison (genre, nombre, personne, temps, mode). Ce type d'analyse morphologique, encore appelé *lemmatisation*, permet de déterminer la forme neutre d'un mot ou *lemme*. Ainsi, le lemme du mot *amies* est : *ami*. Comme exemple de lemmatiseur, on peut citer FLEMM [11] pour le français,

9. *cf.* http ://bll.epnet.com/help/ehost/Stop_Words.htm
10. *cf.* http ://www.sciences.univ-nantes.fr/info/perso/permanents/daille/
11. *cf.* http ://www.univ-nancy2.fr/pers/namer/Telecharger_Flemm.htm

et pour l'anglais on peut utiliser l'étiqueteur morpho-syntaxique de BRILL[12] ;

(b) *la morphologie dérivationnelle* qui est liée à des règles de suffixation et de préfixation. Ainsi, du nom *volcan*, on pourra dériver l'adjectif *volcanique*.

La *radicalisation* est l'analyse morphologique qui va permettre de déterminer le *radical* ou *racine* d'un mot en éliminant les *flexions* et/ou *dérivations* existantes. Ainsi, le mot *volcaniques* aura pour lemme *volcanique* et pour radical *volcan*. Pour déterminer les racines des mots en anglais, on utilise généralement l'algorithme de Porter [Por80]. Des versions françaises de cet algorithme ont été également proposées [PFL+02].

Notons qu'il existe, dans les pré-traitements d'informations textuelles, d'autres types d'analyses à savoir :

(a) l'*analyse syntaxique* qui cherche à structurer la phrase en dépendances syntaxiques du genre *«sujet verbe complément»* ;

(b) l'*analyse sémantique* qui consiste à enrichir le texte avec d'autres informations comme par exemple la classe sémantique (ou catégorie) de chaque mot qui peut-être : objet, humain, etc. ;

5. *la pondération et la sélection des termes significatifs* : les termes significatifs sont des termes de fréquence d'apparition intermédiare. Ils sont obtenus en appliquant la loi de Luhn [Luh58] et de Zipf [Zip49] (*cf.* FIG. 1.4) :
   — selon Zipf, si les termes sont rangés dans l'ordre décroissant de leur fréquence d'apparition, le produit de cette fréquence par le rang est quasiment constant : $Frequence.Rang \simeq Constante$;
   — selon Luhn, les termes peuvent être regroupés en trois classes, en fonction de leur fréquence d'apparition, par la définition d'un seuil minimal S_m et d'un seuil maximal S_M de signification. Ces seuils permettent de caractériser les termes très fréquents et les termes très rares.

En déduction de la loi de Zipf et de la théorie de Luhn, un terme d'indexation (radicalisé ou pas) représente le contenu des documents dans lequel il apparaît (*pouvoir sémantique*) et en même temps il distingue le contenu d'un document du reste de la collection (*pouvoir discriminatoire*), d'où les deux fréquences :

— *fréquence absolue* (notée df_i) ou fréquence d'apparition d'un terme t_i dans une collection de documents ;
— *fréquence relative* (notée tf_{ij}) ou fréquence d'apparition d'un terme t_i dans un document d_j.

12. *cf.* http ://www.cs.jhu.edu/ brill/

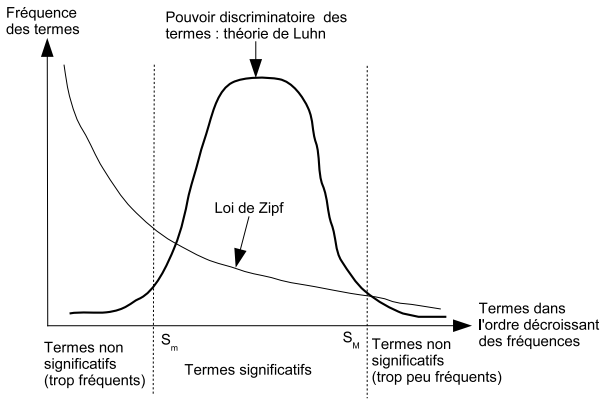

FIGURE 1.4 – Pouvoir discriminatoire des termes d'indexation

Le poids sémantique d'un terme t_i dans un document d_j (noté w_{t_i,d_j}) est proportionnel à sa fréquence relative tf_{ij} et inversement proportionnel à sa fréquence absolue df_i. L'inverse de la fréquence absolue (df_i) est notée idf_i.

En conséquence, la fréquence relative tf_{ij}, l'inverse de la fréquence absolue idf_i et le poids sémantique w_{t_i,d_j} d'un terme peuvent être calculés comme suit :

$$tf_{ij} = \frac{f(t_i, d_j)}{Max[f(t, d_j)]} \tag{1.1}$$

où :

$f(t_i, d_j)$ est la fréquence d'apparition du terme t_i dans le document d_j ;

$Max[f(t, d_j)]$ est la fréquence maximale des termes dans le document d_j ;

$tf_{ij} \in [0, 1]$.

$$idf_i = \log(N/n) + 1 \tag{1.2}$$

où :

N est le nombre de documents de la collection ;

n est le nombre de documents contenant t_i ;

$idf_i \in \Re_+^*$.

$$w_{ti,d_j} = tf_{ij} \cdot idf_i \tag{1.3}$$

D'autres formules pour calculer w_{t_i,d_j} ont également été proposées afin d'améliorer la formule de base de K. Sparck Jones (*cf.* formule 1.3) [Jon72]. Ces formules sont explicitées dans de nombreux ouvrages [FY92] [BYRN99] [SD01b]. Parmi les variantes du calcul du poids sémantique (w_{t_i,d_j}) d'un terme, on peut citer :

1. la formule de *Salton et Buckley 90* [SB90] :

$$w_{t_i,d_j} = \frac{\log(tf_{ij}) + 1}{\sqrt{\sum_{i=1}^{N}(\log(tf_{ij}) + 1)}} \tag{1.4}$$

où :

tf_{ij} est la fréquence d'apparition du terme t_i dans le document d_j ;

N est le nombre total de documents dans la collection.

Notons que $\log(tf_{ij}) + 1$ permet de réduire l'impact des termes trop fréquents lors du processus de comparaison des requêtes et des documents.

2. la formule de *Singhal et al. 97* [SMB97] :

$$w_{t_i,d_j} = \frac{\log(tf_{ij}) + 1}{0,7 + 0,3 \times \frac{l_j}{L_M}} \tag{1.5}$$

où :

tf_{ij} est la fréquence d'apparition du terme t_i dans le document d_j ;

l_j est la longueur du document d_j ;

L_M est la longueur moyenne des documents de la collection.

3. la formule *Okapi-BM25* de *Robertson et al. 99* [RWB99] :

$$w_{t_i,d_j} = \sum_{T \in Q} w^{(1)} \cdot \frac{(k_1+1)tf_{ij}}{K + tf_{ij}} \cdot \frac{(k_3+1)qtf_i}{k_3 + qtf_i} + k_2 \cdot |Q| \cdot \frac{avdl - dl_j}{avdl + dl_j} \tag{1.6}$$

où :

Q est une requête ;

T est l'ensemble des termes de la requête Q ;

$w^{(1)}$ est soit la formule de poids de *Robertson et Sparck Jones 76* [RJ76] des termes T de la requête Q, ci-dessous (*cf.* la formule 1.7)

$$\log \frac{(r + 0,5)/(R - r + 0,5)}{(n - r + 0,5)/(N - n - R + r + 0,5)} \tag{1.7}$$

soit une version plus générale qui prend en compte aussi bien la pertinence que la non-pertinence des informations [RW97] (*cf.* formule 1.8, ci-dessous)

$$\frac{k_5}{k_5 + \sqrt{R}}(k_4 + \log \frac{N}{N-n}) + \frac{\sqrt{R}}{k_5 + \sqrt{R}} \log \frac{r + 0,5}{R - r + 0,5}$$
$$- \frac{k_6}{k_6 + \sqrt{S}} \log \frac{n}{N-n} - \frac{\sqrt{S}}{k_6 + \sqrt{S}} \log \frac{s + 0,5}{S - s + 0,5} \tag{1.8}$$

N est le nombre de documents de la collection ;

n est le nombre de documents contenant le terme t_i ;

R est le nombre de documents connus comme étant pertinents pour une requête spécifique ;

r est le nombre de documents pertinents contenant le terme t_i ;

S est le nombre de documents connus comme étant non-pertinents pour une requête spécifique ;

s est le nombre de documents non-pertinents contenant le terme t_i ;

tf_{ij} est la fréquence d'apparition du terme t_i dans le document d_j ;

qtf_{ij} est la fréquence d'apparition du terme t_i dans la requête Q ;

dl_j est la longueur du document d_j ;

$avdl$ est la longueur moyenne des documents de la collection ;

$K = k_1((1-b) + b.dl_j)/avdl$;

k_1, k_2, k_3, b, k_4, k_5 et k_6 sont des paramètres dépendant de la nature des requêtes et éventuellement de la collection. Les valeurs de ces paramètres ont été obtenues par des séries d'expérimentations sur des collections de tests.

4. la formule de *Boughanem et al. 00* utilisé dans le système connexionniste *Mercure* [BJMSD00]. Cette formule a été inspirée des travaux de Robertson sur le projet Okapi [RWB99]. Sa formule est la suivante :

$$w_{t_i,d_j} = \frac{(\log(tf_{ij}) + 1) \cdot (h_1 + h_2 \cdot \log(\frac{M}{n_i}))}{h_3 + h_4 \cdot \frac{dl_j}{\Delta}} \tag{1.9}$$

où :

tf_{ij} est la fréquence d'apparition du terme t_i dans le document d_j ;

M est le nombre de documents dans la collection ;

n_i est le nombre de documents contenant le terme t_i ;

dl_j est le nombre de termes d'indexation du document d_j ;

Δ est le nombre moyen de termes dans un document (longueur moyenne des documents de la collection).

Notons que les formules qui prennent en compte la longueur des documents [SMB97] [RWB99] [BJMSD00] essayent de compenser les fréquences élevées de termes dans le cas de documents longs conduisant généralement à la restitution des documents les plus longs au détriment des documents les plus courts. Ces mesures sont de ce fait bien adaptées à des collections de documents hétérogènes en longueur (nombre de termes d'indexation, etc.).

5. la formule de *Crampes 80* pour la pondération de textes courts. La pondération des termes extraits de textes courts de type *titre* ou *résumé* est un cas particulier. La fréquence relative dans ce contexte est généralement binaire. Des formules spécifiques ont alors été proposées dont celle de *Crampes 80* [Cra80], définie comme suit :

$$w_{t_i,d_j} = (1 - \frac{n_i}{N})^{\sqrt{N}} \tag{1.10}$$

où :

N est le nombre de documents (textes courts) dans la collection ;

n_i est le nombre de documents (textes courts) contenant le terme t_i.

1.4.2.2 Clustering et Approches par stéréotypes

Le *clustering* ou *classification* [LC96] [Dun00] identifie et classe les objets sur la base de la similarité des caractéristiques qu'ils possèdent. Le clustering cherche à minimiser la variance à l'intérieur d'un groupe et à maximiser la variance entre les groupes. Le résultat du clustering devrait être un nombre de groupes hétérogènes avec des contenus homogènes.

Les méthodes de classification et/ou d'apprentissage automatique [13] sont nombreuses [Mit97]. En général, on peut les classer en deux catégories :

— *les méthodes statistiques* qui reposent sur du calcul de corrélations, d'analyse de données, d'occurrences et de co-occurrences de données comme : la méthode de *classification par analyse de vraisemblance de liens* [Ler91] ; les algorithmes du *bayésien naïf* [NVB+01] et du *C4.5* [Qui93] ; etc. ;

— *les méthodes symboliques* qui s'opposent aux méthodes statistiques. Dans ces méthodes, on va utiliser des symboles, des patterns ou des combinaisons de symboles ou patterns que l'on va essayer de reconnaître au sein d'un objet donné. On peut citer, par exemple, les méthodes de classification basées sur de la *programmation logique inductive* [MdR94] [Fla98].

On peut également classer les méthodes de classification et/ou d'apprentissage en :

— *méthodes d'apprentissage supervisé* ou apprentissage à partir d'exemples comme : les algorithmes du *bayésien naïf* [NVB+01] et du *C4.5* [Qui93] ; la *programmation logique inductive* [MdR94] [Fla98] ; etc. ;

— *méthodes d'apprentissage non supervisé* si l'on n'a pas recourt à la connaissance d'un expert du domaine à travers une base d'exemples. On peut citer, dans ce cas, la méthode de *classification par analyse de vraisemblance de liens* [Ler91].

13. L'apprentissage automatique consiste à apprendre les critères discriminatoires d'un ensemble d'objets pour pouvoir par la suite reconnaître automatiquement un objet donné qui correspond à ces critères

Ainsi, avec des profils individuels d'utilisateurs, on peut créer des groupes d'utilisateurs par classification en regroupant, par exemple, des utilisateurs de centres d'intérêts communs. De même, pour les informations mises à disposition, on va pouvoir les regrouper par classes sur la base, par exemple, de la similarité de leur contenu.

L'*approche par stéréotype* [SSH97] est une approche particulière de clustering qui est fondée sur l'identification (généralement manuelle) de groupes appelés stéréotypes et sur la détermination des caractéristiques clés de chaque groupe. Les groupes et les caractéristiques de chaque groupe sont pré-définis et les différents objets (utilisateurs ou informations) à classer sont affectés à ces groupes en fonction du degré de ressemblance de leur profil individuel aux différents stéréotypes.

1.4.2.3 Apprentissage de profils utilisateurs par profiling

L'apprentissage de profils utilisateurs peut se faire en « scrutant » les habitudes des utilisateurs et en analysant leurs réactions vis à vis des documents qui leur sont présentés : c'est du profiling. Le système détermine le besoin que l'utilisateur semble exprimer en notant, par exemple, la fréquence d'apparition de certains termes dans les requêtes ou documents visités.

Le profiling [CKK02] [BBB03b] consiste donc à scruter, enregistrer et analyser les actions et successions d'actions d'un utilisateur durant des sessions de recherche pour déterminer son profil. Pour ce faire, un sous-système de modélisation ou un *agent* observe l'utilisateur au travers d'une interface et apprend le profil de ce dernier à partir de ses actions (visite d'un site Internet de manière récurrente, achat d'un produit en ligne, sauvegarde de documents, etc.). Il s'agit ici de ce que l'on appelle *profil implicite*. Il est souvent difficile pour un utilisateur d'exprimer clairement ses besoins, mais il lui est plus facile d'identifier ses besoins en partant des documents susceptibles d'y répondre. On peut donc utiliser le jugement d'un utilisateur sur des documents qui lui sont proposés (résultats d'une recherche par exemple) pour déterminer ses besoins en informations.

Le principe du profiling est d'enregistrer et d'analyser les comportements et actions des visiteurs pour déduire par apprentissage leurs besoins. Il est possible d'utiliser les différentes informations que l'utilisateur examine pour essayer d'en extraire ses thèmes de recherche ou centres d'intérêts [CKK02] [BBB03b]. Pour cela, on peut se baser sur :

- l'usage fait d'une information : lue ou ignorée ;
- le temps d'étude d'une information : une information non pertinente ne devrait pas être étudiée longtemps par l'utilisateur, au contraire des informations pertinentes sur lesquelles l'utilisateur va s'attarder ;
- les différentes commandes exécutées à la suite de l'étude d'une information : si l'utilisateur sauvegarde l'information sur son ordinateur ou répond à ses mails, on peut alors supposer que l'information ou le

mail en question répond à ses besoins. Dans le même ordre d'idées, certains systèmes comptabilisent le nombre de « clicks » effectués par l'utilisateur lors de la consultation d'une information pour déduire l'intérêt que l'usager porte à cette information ;

— les annotations sur des informations : jugements, commentaires, etc.

Le profiling permet également d'opérer des regroupements entre des profils similaires et de produire des offres personnalisées aux membres de chaque groupe.

En résumé, il existe aujourd'hui une multitude d'approches qui permettent la construction automatique de profils. La plupart d'entre elles réalisent un apprentissage basé sur des techniques neuro-mimétiques [CP43], sur des algorithmes génétiques [Hol75], sur des processus de classification [Mit97], etc. Cependant, toutes ces méthodes ne sont pas forcément exclusives et peuvent être combinées pour obtenir un résultat optimal [Cal98] [BCSD99] [BHM01].

Dans la section suivante, nous présentons les principaux modèles de représentation du contenu de profils en RI et FI et nous expliquons comment ces modèles sont utilisés pour apparier des profils ou mesurer la similarité entre profils.

1.4.3 Modèles de représentation de contenu et appariements de profils

Il existe plusieurs modèles de représentation de contenu de dimensions (ou critères descriptifs) de profils en RI/FI et chacun d'entre eux permet de définir un principe d'appariement (ou de comparaison) de profils. Parmi ces modèles, les plus utilisés sont : le modèle booléen, le modèle vectoriel et le modèle probabiliste. Nous illustrons ces différents modèles de représentation à travers un *modèle de document* et un *modèle de requête* mais cela peut-être généralisé à tout type de profils.

1.4.3.1 Modèle booléen et booléen étendu

Le modèle booléen est basé sur la présence ou l'absence des termes de la requête dans les représentations des documents (termes d'indexation associés). Il permet d'effectuer une recherche stricte réalisée à partir d'une comparaison exacte entre le besoin en information, décrit dans la requête à l'aide d'opérateurs logiques (ET, OU, NON ou SAUF) et les termes représentant les documents d'une collection. Par exemple, soit la requête booléenne suivante : *«initiation ET informatique ET (algorithmique OU programmation) ET NON gestion»*, les documents restitués devront comporter :

— les trois termes *initiation*, *informatique* et *algorithmique* mais pas le terme *gestion* ;

— les trois termes *initiation*, *informatique* et *programmation* mais pas le terme *gestion* ;
— les quatre termes *initiation*, *informatique*, *algorithmique* et *programmation* mais toujours pas le terme *gestion* ;

Soit Q l'ensemble des requêtes booléennes, D l'ensemble des documents d'une collection, T l'ensemble des termes d'indexation et F la fonction de similitude telle que :

$$F : D \times Q \rightarrow \{0,1\}$$

$$(d,q) \rightarrow \{0,1\}$$

La ressemblance $F(d_k, q)$ entre un document d_k et une requête q dépend de l'équation de cette requête qui peut avoir les formes basiques suivantes :

$$F(d_k, t_i) = \begin{cases} 1 & si\ t_i \in d_k \\ 0 & sinon \end{cases}$$

$$F(d_k, t_i ET t_j) = MIN(F(d_k, t_i), F(d_k, t_j)) = F(d_k, t_i) * F(d_k, t_j)$$

$$F(d_k, t_i OU t_j) = MAX(F(d_k, t_i), F(d_k, t_j))$$
$$= F(d_k, t_i) + F(d_k, t_j) - F(d_k, t_i) * F(d_k, t_j)$$

$$F(d_k, NON t_i) = 1 - F(d_k, t_i)$$

Le modèle booléen a l'avantage d'être simple à mettre en œuvre mais engendre plusieurs inconvénients, notamment le fait de ne pas pouvoir ordonner les documents restitués. De plus, l'écriture de la requête peut-être complexe et une mauvaise formulation peut engendrer des résultats erronés.

Une extension du modèle booléen a été introduite pour essayer de pallier le premier inconvénient (ordonnancement des résultats) en s'appuyant sur la théorie des ensembles flous proposée par Zadeh. Dans ce modèle, la fonction de similitude retourne une valeur dans un intervalle $[0,1]$:

$$F : D \times Q \rightarrow [0,1]$$

$$(d,q) \rightarrow [0,1]$$

Le modèle *booléen étendu* prend en compte, en plus de l'absence ou de la présence des termes de la requête dans le document, les poids sémantiques de chacun de ces termes. L'inconvénient majeur de cette méthode de recherche est que l'évaluation peut proposer des valeurs différentes pour des expressions équivalentes.

1.4.3.2 Modèle vectoriel

Le modèle vectoriel préconise une représentation commune de la requête et des unités d'informations dans un même espace vectoriel engendré par les termes d'indexation [SM83]. Ainsi, si on pose $\overrightarrow{q_i}$ le vecteur représentant la requête q_i et $\overrightarrow{d_j}$ le vecteur représentant le document d_j :

$$\overrightarrow{q_i} = (w_{t_1,q_i}, w_{t_2,q_i}, w_{t_3,q_i}, \ldots, w_{t_n,q_i})$$

$$\overrightarrow{d_j} = (w_{t_1,d_j}, w_{t_2,d_j}, w_{t_3,d_j}, \ldots, w_{t_n,d_j})$$

où :

n est le nombre de termes d'indexation de la base ;

w_{t_n,q_i} est le poids du terme t_n dans la requête q_i ;

w_{t_n,d_j} est le poids du terme t_n dans le document d_j.

La figure FIG. 1.5 illustre, par exemple, la représentation vectorielle d'un document dans un espace à deux dimensions (termes t_1 et t_2).

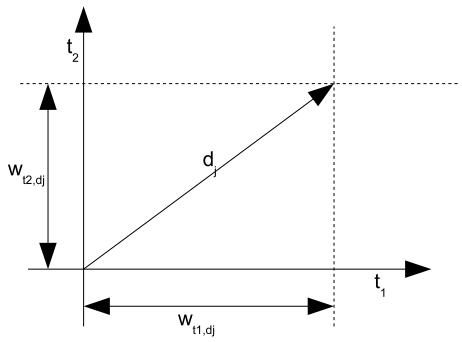

FIGURE 1.5 – Représentation vectorielle d'un document dans l'espace des termes t_1 et t_2

L'hypothèse de l'indépendance des termes (et donc de l'orthogonalité des vecteurs termes) est généralement faîte. Cette hypothèse permet de simplifier toutes les opérations de représentation et de recherche d'information à partir des termes d'indexation : calcul de poids, calcul de similarité, etc.

La majorité des modèles de recherche utilisés en informatique documentaire sont basés sur cette représentation. Afin de pouvoir apparier ou comparer les vecteurs de requête et les vecteurs de documents, des mesures de similitude ont été élaborées : *produit scalaire*, *mesure du cosinus*, *distance métrique*.

1. Le *produit scalaire* est la fonction mathématique classique utilisant les coordonnées des vecteurs (poids sémantique des termes). La mesure

de similarité entre le vecteur $\overrightarrow{q_i}$ et le vecteur $\overrightarrow{d_j}$, notée $sim(q_i, d_j)$ correspond au produit des vecteurs :

$$sim(q_i, d_j) = \sum_{k=1}^{n} w_{t_k,q_i} \cdot w_{t_k,d_j} \tag{1.11}$$

où :

n est le nombre de termes d'indexation de la base ;

w_{t_k,q_i} est le poids du terme t_k dans la requête q_i ;

w_{t_k,d_j} est le poids du terme t_k dans le document d_j.

2. La *mesure du cosinus* est la plus répandue dans ce type de modèle pour évaluer la ressemblance des documents et de la requête. Elle mesure l'angle entre les vecteurs. Elle est équivalente au produit scalaire des vecteurs normalisés. La mesure du cosinus est donnée comme suit [SM83] :

$$sim(q_i, d_j) = \frac{\sum_{k=1}^{n} w_{t_k,q_i} \cdot w_{t_k,d_j}}{(\sum_{k=1}^{n} w_{t_k,q_i}^2)^{1/2} (\sum_{k=1}^{n} w_{t_k,d_j}^2)^{1/2}} \tag{1.12}$$

où les poids des termes de la requête w_{t_k,q_i} et du document w_{t_k,d_j} sont ceux définis précédemment pour l'équation 1.11.

La pertinence d'une information par rapport à une requête est alors rattachée à la mesure de similarité des vecteurs correspondants. Le cosinus de deux vecteurs appartient toujours à l'intervalle $[0, 1]$ et plus les vecteurs sont proches, plus leur similarité (valeur du cosinus de l'angle entre les vecteurs) est grande. Ainsi, *si* $sim(q_i, d_j) = 1$ *alors* $\overrightarrow{q_i} = \overrightarrow{d_j}$. A chaque information ou document, il va correspondre une mesure de similarité ou degré de ressemblance qui va permettre de sélectionner les informations à restituer selon un seuil donné. Les informations sélectionnées peuvent ensuite être classées selon un degré de pertinence décroissant.

3. La *distance métrique* entre les vecteurs à apparier se calcule comme suit :

$$dist(q_i, d_j) = ||q_i, d_j|| = \Big(\sum_{k=1}^{n} |w_{t_k,q_i} - w_{t_k,d_j}| \Big)^{1/p} \quad avec \quad p \geq 1 \tag{1.13}$$

Dans ce cas, plus la distance est grande, plus les vecteurs sont différents. Ainsi, *si* $dist(q_i, d_j) = 0$ *alors* $q_i = d_j$.

L'inconvénient majeur du modèle vectoriel est qu'il ne permet pas de modéliser les associations entre termes d'indexation : chaque terme est considéré comme indépendant des autres (principe d'orthogonalité des vecteurs termes).

1.4.3.3 Modèle probabiliste

Dans le modèle probabiliste, on suppose que lorsque les représentations de la requête et d'un document sont suffisamment similaires, la probabilité correspondante de pertinence est suffisante pour restituer le document en réponse à la requête. Deux probabilités conditionnelles sont utilisées :

— $P(t_i/Pert)$: probabilité que le terme t_i apparaisse dans un document donné sachant que ce document est pertinent pour la requête ;

— $P(t_i/NonPert)$: probabilité que le terme t_i apparaisse dans un document donné sachant que ce document n'est pas pertinent pour la requête.

En utilisant une formule établie par Bayes (probabilités conditionnelles) et en supposant l'indépendance des variables *document pertinent* et *document non pertinent*, la fonction de recherche peut-être obtenue en calculant la probabilité de pertinence $P(Pert/d_j)$ d'un document d_j donné. Soit le document $d_j = (t_1, t_2, t_3, \ldots, t_n)$ où :

$t_i = 1$ si le terme t_i indexe le document d_j ;

$t_i = 0$ sinon.

La probabilité de pertinence d'un document $P(Pert/d_j)$ sachant sa description et la probabilité de non pertinence $P(NonPert/d_j)$ de ce document se calculent comme suit :

$$P(Pert/d_j) = \frac{P(d_j/Pert) \cdot P(Pert)}{P(d_j)} \tag{1.14}$$

$$P(NonPert/d_j) = \frac{P(d_j/NonPert) \cdot P(NonPert)}{P(d_j)} \tag{1.15}$$

Pour les équations 1.14 et 1.15 on a :

$P(d_j) = P(d_j/Pert).P(Pert) + P(d_j/NonPert).P(NonPert)$;

$P(d_j/Pert)$ (respectivement $P(d_j/NonPert)$) est la probabilité de restituer le document d_j sachant qu'il est pertinent (respectivement non pertinent) ;

$P(Pert)$ (respectivement $P(NonPert)$) est la probabilité à priori pour qu'un document soit pertinent (respectivement non pertinent).

Si on considère l'indépendance des termes :

$$P(d_j/Pert) = \prod_{i=1}^{n} P(t_i/Pert) \tag{1.16}$$

$$P(d_j/NonPert) = \prod_{i=1}^{n} P(t_i/NonPert) \tag{1.17}$$

Pour les équations 1.16 et 1.17 on a :

$P(t_i/Pert) = \frac{r_i}{R}$;
$P(t_i/NonPert) = \frac{n_i - r_i}{N - R}$;
r_i est le nombre de documents pertinents dans lesquels le terme t_i apparaît ;
R est le nombre de documents pertinents pour la requête (dans la collection) ;
n_i est le nombre de documents non pertinents dans lesquels le terme t_i apparaît ;
N est le nombre de documents non pertinents pour la requête (dans la collection).

Pour caractériser l'occurrence des termes d'indexation dans les documents, on utilise une loi de distribution (comme la loi de poisson). Cette occurrence est déduite d'un échantillon de documents.

Pour la restitution, les documents sont classés en fonction de $P(Pert/D)$. Le principe d'ordonnancement probabiliste stipule que cet ordonnancement est optimal en ce sens que, quel que soit le pourcentage de documents pertinents qui sont restitués, le pourcentage de documents restitués qui sont effectivement pertinents est maximisé.

1.4.3.4 Autres modèles de représentation d'informations

Les modèles de représentation booléen, vectoriel et probabiliste sont les plus utilisés dans les techniques d'accès à l'information. Notons cependant qu'il existe également d'autres modèles de représentation parmi lesquels nous pouvons citer :

— *le Latent Semantic Analysis ou Indexing (LSA ou LSI)* qui est une variante du modèle vectoriel. Il convertit un exemple représentatif de documents en une matrice de *terme-par-document* dans laquelle chaque cellule indique la fréquence avec laquelle chaque terme (lignes) apparaît dans chaque document (colonnes). Ainsi, un document devient un vecteur colonne et peut être comparé à une requête utilisateur représentée comme un vecteur de même dimension. Le LSA ou LSI étend le modèle vectoriel en modélisant les relations terme-document via une approximation réduite de l'espace ligne et colonne. Cette approximation est calculée par une *décomposition en valeurs singulières* ou *SVD pour Singular Value Decomposition* de la matrice de *terme-par-document* [DDF90] [Dum94]. Dans l'approche LSI, on va essayer de tenir compte du contenu sémantique (mots-clés) et conceptuel des documents (relations *terme-document* déduite par *SVD*) ;

— *les réseaux inférentiels Bayésiens* qui permettent de prendre en compte la dépendance entre les termes, contrairement au modèle vectoriel [TC91] ;

— *les approches connexionnistes* [CP43], [BJMSD00] ;
— *les graphes conceptuels* [Sow84] ;
— *les algorithmes génétiques* [Hol75].

Nous allons présenter maintenant des notions relatives à la personnalisation en particulier. La personnalisation est devenue une approche incontournable et ceci s'illustre par le développement des nombreuses méthodes d'accès personnalisé à des ressources.

1.5 Personnalisation de l'accès à des ressources

La personnalisation et l'adaptation des processus sont des concepts très souvent liés dans des applications. Nous les explicitons, dans cette section, à travers des définitions et des exemples d'applications. Ensuite, nous effectuons une classification plus générale des techniques de personnalisation. Enfin, une étude comparative de systèmes de personnalisation nous permet de situer nos travaux de recherche par rapport à la revue de littérature effectuée.

1.5.1 Personnalisation et adaptation des processus

De façon générale, personnaliser l'information signifie s'adapter aux buts, préférences et capacités de chaque utilisateur :

— *l'adaptation aux buts* consiste à prendre en considération le but que cherche à atteindre l'utilisateur, l'objectif de sa recherche. Cette prise en compte du but de l'utilisateur est une des préoccupations fondamentales en personnalisation. Cet objectif est généralement déterminé en scrutant et en analysant les actions de l'usager lors de ses sessions de recherche [BBB03a] ou en lui posant des questions à travers des formulaires ou des demandes de jugements [BCSD99] sur des informations par exemple ;
— *l'adaptation aux préférences* de l'utilisateur existe de façon répandue dans les interfaces adaptables et paramétrables par l'utilisateur, grâce à un ensemble d'options de menus de type *« personnaliser »*. Un exemple d'une forme plus « automatisée » d'adaptation est donnée par la configuration d'un environnement graphique particulier et l'exécution de certaines applications utiles à l'utilisateur, lorsqu'il se connecte à un système en s'identifiant ;
— *l'adaptation aux capacités* de l'utilisateur consiste à lui délivrer de l'information dans une forme utilisable et dans des délais acceptables par lui. Par exemple, cela peut consister à limiter l'attente de l'utilisateur suite à ses requêtes. Cela peut se concrétiser également par l'adaptation aux moyens matériels et/ou logiciels dont disposent l'utilisateur pour visualiser l'information fournie (exemple : taille d'écran, débit du réseau, etc.) [KRW+04] [HBS02].

Notons également que la personnalisation ne peut pas être regardée du seul point de vue d'un individu. Elle s'applique aussi pour un groupe d'individus ayant une ou plusieurs caractéristiques communes. La personnalisation doit permettre de fournir à chaque utilisateur ou groupe d'utilisateurs des informations qui correspondent à son profil. Ainsi, d'un usager à un autre on n'aura pas forcément le même ensemble de résultats pour un même besoin (ou demande ou requête) en information.

L'adaptation, quant à elle, va permettre de modifier le comportement d'un système. Elle recouvre deux notions qui sont : *l'adaptabilité et l'adaptativité*. L'*adaptabilité* est la capacité d'un système à être modifié par des usagers. L'*adaptativité* est la capacité du système à modifier son comportement sans intervention explicite de l'usager en apprenant des interactions de ce dernier. Ainsi, l'adaptativité est liée à l'évolution automatique, dans le temps, des profils des usagers [Tma02].

Un système d'accès à l'information peut être :

— *personnalisable uniquement*, c'est à dire qu'il prend en compte le profil de chaque utilisateur (ce profil n'évolue pas forcément) pour lui renvoyer de l'information [LC99] [BE02] ;

— *adaptatif uniquement*, c'est à dire qu'il modifie son comportement en fonction des déductions qu'il fait des interactions des usagers. Cependant, il ne fait pas de distinction entre un usager ou un groupe d'usagers particulier. Ici, l'usager représente l'ensemble de la population des utilisateurs (c'est-à-dire tout le monde). On peut citer par exemple, le système Alexa [14] qui est un site internet américain qui audite et rend publique la fréquentation des sites internet ;

— *personnalisable et adaptatif*, c'est à dire qu'il identifie chaque usager et chaque groupe d'usagers de façon unique. Il définit alors une stratégie d'accès à l'information qui correspond à chaque utilisateur ou groupe d'utilisateurs et qui évolue en fonction des interactions de chacun (individu ou groupe) [Lie95] [Mla96] [PMB96] [Jac98] [Che02] ;

— *non personnalisable et non adaptatif*, comme le système Walden Path [15] qui est un outil qui permet (à un enseignant, par exemple) d'organiser des informations afin qu'elles soient facilement utilisables (par des élèves ou étudiants, par exemple). Il permet de collecter, d'annoter et de stocker des informations provenant de sources variées.

Dans la section suivante, nous allons présenter une classification des techniques de personnalisation de façon générale.

1.5.2 Classification de techniques de personnalisation

Il existe différentes techniques de personnalisation que l'on peut regrouper en deux groupes :

14. http ://http ://www.alexa.com/
15. http ://www.csdl.tamu.edu/walden/

1. les plus simples qui permettent à l'utilisateur de personnaliser, par exemple, une page web ou l'interface d'une application en définissant manuellement des couleurs, des polices de caractères, etc. On parle aussi de *customisation* ;
2. les plus complexes qui se basent sur une modélisation automatique ou semi-automatique de l'utilisateur appelé *profil utilisateur*. Parmi ces techniques, on peut citer :
   (a) *les tuteurs intelligents* [Bru95] : qui prennent en compte le niveau des apprenants ainsi que leurs connaissances pour dispenser un cours de façon adaptée à chacun d'entre eux ;
   (b) *les interfaces adaptatives* [Che02] : qui permettent d'assister visuellement l'utilisateur à travers un module de recommandation, par exemple, lors de la navigation. Ce module de recommandation pourra mettre en évidence : l'importance d'un document par rapport à la navigation en cours, la liste des répertoires de l'utilisateur qui contiennent un document ou signet donné, etc.
   (c) *les systèmes basés sur les actions des usagers en phase de recherche* [CKK02] [Sha00] : qui construisent un modèle utilisateur à partir d'un certain nombre d'actions pré-définis (sauvegardes, jugements, clicks, etc.). Ces systèmes permettent de construire un profil utilisateur de façon transparente à l'usager (profil implicite) ;
   (d) *les systèmes de personnalisation à base d'annotations* [CB02] : qui s'intéressent particulièrement au partage d'annotations pour faciliter les tâches collaboratives au sein d'une communauté d'utilisateurs. Une nouvelle annotation d'un utilisateur ainsi que la ressource annotée peuvent être intéressantes pour les autres usagers de profil similaire au sien ;
   (e) *les systèmes de personnalisation de « services web »* comme :
      — *le commerce électronique* qui est un processus d'achat et de vente sur internet [CKK02]. Pour cela, il développe également des techniques de recherche de nouveaux clients, de fidélisation de clients existants [YK02], en analysant leurs achats et comportements sur des sites de vente en ligne ;
      — mais aussi *le courrier électronique* [GNOT92] ; *les listes de discussions* (Usenet News) [KMM+97] ; *les guides de programmes télévisés* personnalisés et basés sur le web [CS00] ; etc. ;
   (f) *les systèmes de personnalisation pour les dispositifs mobiles* comme les téléphones portables, les PDA ou *Personnal Digital Assistant*, etc. Ces dispositifs sont caractérisés par une taille d'affichage réduite, une bande passante limitée et une mémoire restreinte. Les techniques de personnalisation doivent donc prendre en compte ces particularités afin de s'adapter à ce type d'oultils qui sont de

plus en plus utilisés pour accéder à l'information via le WAP ou *Wireless Application Protocol* [RFC 2757]. Dans ce contexte, la personnalisation s'avère particulièrement indispensable pour réduire la quantité d'informations à renvoyer aux usagers.
Les contenus web sont affichés comme des *pages WML*. Le WML [16] ou *Wireless Markup Language* est l'équivalent du HTML pour les systèmes compatibles WAP. Comme exemple de système personnalisé basé sur le WAP, on peut citer : les guides de programmes télévisés personnalisés [CS00]. Notons que pour les téléphones mobiles et les téléphones fixes, il existe également des *systèmes d'envois de SMS (Short Message Service ou Service de messages courts) ou MMS (Multimedia Message Service) personnalisés* pour des annonces diverses : offres, publicités, soldes de comptes bancaires, etc. De plus, on peut aujourd'hui coupler : les pages Web qui sont régies par le protocole HTTP ; les e-mails qui fonctionnent avec les protocoles POP et SMTP ; les pages WML qui sont régies par le protocole WAP et les SMS [17] ;

(g) *les systèmes de personnalisation basés sur la notion de contexte* qui recouvre des aspects relatifs :
- *au contexte des besoins ou contexte cognitif* de l'usager pour préciser le sens d'une requête ou déterminer un centre d'intérêt ponctuel de l'utilisateur lors d'une recherche. Le contexte des besoins fait généralement appel à la notion de *profils court terme* [BBB03a] [BL01a] (pouvant être positif ou négatif). Le profil long terme peut intervenir également, dans la détermination du contexte, pour expliciter une requête (ou un centre d'intérêt ponctuel) ayant une certaine similarité avec ce dernier. La combinaison de centres d'intérêts long terme et court terme va permettre de définir un contexte robuste de l'utilisateur [KC03] [BL02] ;
- *au contexte environnemental* de l'usager et/ou des informations mises à disposition en tenant compte par exemple : de la nature de l'information disponible, des applications (ou logiciels) utilisées, du matériel utilisé (PC, téléphone portable, etc.), de la situation géographique de l'usager [KYS05] [BD03] [HW04], etc. Le but est de déterminer un ensemble de conditions en corrélation qui se produisent au cours d'une activité ou à un moment donné et qui peuvent influer sur la *qualité* des résultats restitués [PSC+02].

Il est à noter que des combinaisons des techniques citées précédemment sont possibles. Dans la section suivante, nous allons faire une étude compa-

16. *cf.* http ://thewml.org/docs/
17. *cf.* http ://www.mctel.fr/

rative de différents systèmes de personnalisation dans le domaine de l'accès à l'information (*cf.* section 1.3) afin de positionner nos travaux de recherche.

1.5.3 Étude comparative de systèmes de personnalisation pour l'accès à l'information

Le tableau TAB. 1.6 illustre une étude comparative entre systèmes de personnalisation pour les critères suivants :

- *contexte d'étude* ou type d'accès à l'information ;
- *typologie sémantique des profils utilisés* (*cf.* section 1.4.1, TAB. 1.4 et TAB. 1.5) ;
- *typologie structurelle des profils* (*cf.* section 1.4.1) ;
- *méthode d'évaluation de la pertinence utilisateur* pour restituer de façon personnalisée des informations à chaque usager (individu ou groupe).

Le tableau TAB. 1.6 nous permet de mettre en évidence la diversité qui existe au niveau des systèmes de personnalisation de l'accès à l'information pour les différents critères de comparaison que nous avons choisis. Dans nos travaux de recherche, nos contributions portent sur ces différents aspects.

1. Les trois premiers critères montrent que chaque application définit la typologie sémantique ou structurelle de ses profils selon l'objectif qu'elle veut atteindre. C'est pour cela que nous nous sommes fixés comme premier objectif celui de définir un *cadre générique pour la conception de tout type d'applications d'accès à des ressources.* Ce cadre doit fournir une structure de base homogène à partir de laquelle on doit pouvoir concevoir des applications d'accès. Le cadre générique que nous proposons comprend :
   - *une architecture générique d'accès à des ressources à base de profils* ;
   - *un modèle générique de profil* pour la description de tout type de profil et qui est instantiable dans différentes applications.

   Plusieurs travaux ont proposé des modèles génériques d'utilisateurs [Kay95] [Sha00] [Kob01]. Dans nos travaux, nous ne nous intéressons pas qu'aux usagers mais à tout type de ressources (matériels, logiciels, documents, etc.) ainsi qu'aux interactions entre différents modèles de ressources. *La spécificité de notre architecture se situe au niveau des nombreuses possibilités de combinaisons de profils complémentaires décrivant différents espaces de ressources.*

   Par la suite, nous nous sommes aussi intéressés aux méthodes d'exploitation de profils pour améliorer l'accès aux ressources à travers des *méthodes d'analyse et d'appariement de profils.*
2. Le but de *l'appariement de profils* pour l'accès à des ressources est la restitution de ressources adaptées aux attentes des utilisateurs. Parmi les méthodes d'appariement on peut citer :

— *la correspondance aux besoins de l'utilisateur* qui mesure un degré de similarité entre le profil des besoins de l'utilisateur ou sa requête (éventuellement reformulée) et une information mise à disposition [PMB96] [Paz99] [BBB03a] ;
— *la sélection multi-critères* qui permet de restituer à l'utilisateur les informations qui correspondent à un sous ensemble de critères pré-définis (langue, taille, date, etc.) ;
— *l'interrogation à relance* [Ame01] qui consiste à évaluer dans un premier temps la correspondance aux besoins de l'usager, puis à effectuer une sélection multi-critères sur les résultats obtenus.

Les méthodes d'évaluation de la *pertinence utilisateur* lorsqu'elles prennent en compte plusieurs critères sont basées généralement sur des appariements de *type booléen* ou de *type attribut-valeur*. Dans ce cas, on ne renvoie à l'utilisateur que les informations qui correspondent aux critères (ou attributs) choisis et qui respectent les contraintes imposées sur ces critères. Cette approche pose certains problèmes :

— il y a un risque d'avoir un ensemble vide de résultats, si aucune ressource ne correspond exactement aux critères définis ;
— il y a également le risque d'augmenter *le silence* informationnel en ne restituant pas des ressources qui seraient intéressantes pour un sous-ensemble important de critères de l'ensemble pré-définis ;
— cette méthode donne la même importance aux différents critères, ce qui ne correspond pas toujours aux objectifs de l'utilisateur. Ainsi, certaines ressources, qui seraient intéressantes pour un critère très important pour l'utilisateur et moins intéressantes pour les autres critères choisis, risquent de ne pas être restituées ;
— on ne peut pas ordonner les résultats selon une mesure de similarité décroissante.

Pour pallier ce handicap, il est nécessaire de proposer des méthodes d'évaluation de la pertinence utilisateur basées sur plusieurs critères, qui calculeraient un degré de ressemblance et non une valeur binaire. L'approche que nous proposons s'inscrit dans cette problématique qui est aussi celle des méthodes de recherche de solutions optimales pour les problèmes de décision basés sur des attributs multiples [ZP04]. *La particularité de notre méthode d'appariement de profils est qu'elle va calculer « un degré de ressemblance » entre profils, relativement à un ensemble de critères pré-définis et pondérés.*

1.6 Conclusion

L'accès à des ressources conduit nécessairement à modéliser les différentes ressources à manipuler. De nombreux travaux, présentés dans ce chapitre,

ont été menés dans ce sens. Ils sont très variés et ont pour préoccupation la satisfaction des usagers. L'étude comparative effectuée entre différents systèmes de personnalisation (*cf.* section 1.5.3) nous permet de souligner l'intérêt qu'il y a à définir un cadre générique pour l'accès à des ressources, ainsi que l'intérêt de la définition de méthodes d'analyse et d'appariement de profils pour optimiser cet accès.

La définition d'un cadre générique est un premier pas vers la gestion de l'hétérogénéité des ressources. Cependant l'instanciation du modèle générique de profils, par les applications, va faire resurgir (dans une moindre mesure) ce problème de disparité entre modèles. Les structures spécifiques des instances de profils vont être, en général, hétérogènes même si elles vont se baser sur le même modèle générique. Afin de pallier cela, nous allons intégrer une *dimension sémantique* à notre modèle générique. Pour mieux expliquer l'intérêt de cette sémantique, nous présentons dans le chapitre suivant, un état de l'art sur la *description de ressources* qui a pour but de présenter les modèles et langages existants de description pour l'intégration de ressources hétérogènes.

Systèmes de personnalisation	Contexte	Typologie sémantique de profils	Typologie structurelle de profils	Évaluation de la pertinence utilisateur
Syskill & Webert 96 [PMB96]	Filtrage cognitif	1. profil du contenu de document 2. profil individuel des besoins	mono-critère	mesure la correspondance au profil des besoins de l'utilisateur
Pazzani 99 [Paz99]	Filtrage hybride	1. profil du contenu de document 2. profil individuel des besoins, des jugements, des données démographiques 3. profil de groupe des besoins, des jugements, des données démographiques	mono-critère	Combinaison des cinq premiers résultats des trois types d'approches de filtrage
Amerouali 01 [Ame01]	Recherche	1. profil étendu de document 2. profil individuel étendu	multi-critères	Interrogation à relance
Berti-Equille 02 [BE02]	Recommandation	1. profil étendu de document 2. profil individuel étendu	multi-critères	Recommandation multi-critères
Pitkow 02 [PSC+02]	Recherche	1. profil étendu de document 2. profil individuel étendu 3. profil de groupe étendu	multi-critères	Reformulation de requête en pré-recherche en utilisant le profil des besoins et ordonnancement multi-critères des résultats
Bottraud et al. 03 [BBB03a]	Recherche	1. profil du contenu de document 2. profil individuel des besoins	mono-critère	Reformulation de la requête en post-recherche, réévaluation des résultats et réordonnancement

TABLE 1.6 – Étude comparative de systèmes de personnalisation pour l'accès à l'information

Chapitre 2

Description de ressources

2.1 Introduction

De multiples normes [1] et standards [2] pour la description de ressources ont vu le jour. Ils permettent de décrire une ressource en prenant en compte différents aspects : adressage, structure logique, sémantique, etc. L'objectif de ces normes et standards est d'optimiser la représentation, l'exploitation, l'interopérabilité, et le partage de ressources entre applications.

La majorité de ces normes et standards sont regroupés sous l'appellation de *web sémantique*. Le *web sémantique* est une extension du web actuel dans lequel l'information a un sens bien défini permettant ainsi aux ordinateurs et aux personnes de travailler en coopération [BLHL01]. Il a pour objectif de définir des normes et standards d'échange et de structuration de ressources (information, usager, service, etc.) sur le web. Ces normes et standards vont faciliter l'intégration de ressources hétérogènes car ils permettent de définir des modèles de ressources : extensibles, flexibles, ré-utilisables et interopérables. L'architecture du web sémantique est encore en cours de construction et de nombreuses applications ont déjà montré l'intérêt de sa mise en œuvre. Cette technologie est applicable dans des contextes autres que celui du web où l'on a besoin de faire coopérer des ressources hétérogènes entre elles.

Dans ce chapitre, nous faisons un état de l'art sur la description de ressources à travers un panorama des modèles génériques existants de description de ressources et une présentation plus détaillée de différents modèles et langages de description de ressources dont la majorité ont été fédérés dans l'architecture du web sémantique. Cet état de l'art nous permet de situer nos travaux par rapport à l'existant et de justifier la dimension sémantique de nos propositions.

1. Une norme est un ensemble de règles de conformité, édictées par un organisme de normalisation au niveau national ou international

2. Un standard est un ensemble de recommandations émanant d'un groupe représentatif d'utilisateurs réunis autour d'un forum comme : l'IETF (Internet Engineering Task Force), le W3C (World Wide Web Consortium), le Dublin Core, etc.

2.2 Panorama des modèles génériques de description de ressources

Il existe différents types de modèles génériques de description de ressources que l'on peut classer en trois grandes catégories :

1. Le modèle le plus simple est le modèle *attribut-valeur ou encore clé-valeur* [SMLP01] [SAW94]. La ressource est décrite par une liste de couples *attribut-valeur*. Ce modèle est facile a gérer mais possède un inconvénient majeur lié à l'absence de structure pour représenter des informations complexes (hiérarchies par exemple) et effectuer ainsi des analyses plus fines liées à la structure. Les attributs sont indépendants les uns des autres. De plus, il est impossible d'avoir deux attributs de même nom car le nom de l'attribut constitue la clé d'un l'élément descriptif de profil, c'est-à-dire d'un couple *attribut-valeur* ;
2. Les modèles basés sur *une structure logique et un contenu associé* [BK05] [CKR04] [Khr04] qui permettent une structuration de la description de profils. Cette structuration est généralement sous la forme d'arbre. Ces modèles vont permettre des analyses plus fines car on va pouvoir définir, par exemple, des hiérarchies d'éléments descriptifs de profils. Plusieurs nœuds peuvent donc avoir le même nom mais des chemins d'accès différents dans la structure. Ainsi, on peut avoir un attribut *langue* qui décrit une donnée démographique de l'utilisateur (représentant les langues courantes de l'usager valables quel que soit le contexte de ce dernier) et un autre attribut de même nom *langue*, qui décrit plutôt ses préférences en langues applicables dans un contexte précis (comme un travail de recherche, par exemple). Ces deux attributs seraient décrits par des nœuds différents de la structure du profil, ce qui ne peut être le cas avec le modèle clé-valeur où un nom d'attribut n'est défini qu'une fois. Cependant, les modèles basés sur la structure logique peuvent être assez propriétaires car leur interprétation va être liée à l'application qui les a définis. Ainsi, d'une application à une autre, des caractéristiques identiques peuvent être décrites par des structures logiques différentes et inversement. Il va donc être difficile sans une *sémantique consensuelle* (normes ou standards de langages de métadonnées) de faire inter-opérer des modèles de profils d'applications différentes ;
3. Les modèles basés sur *de la sémantique via des langages de métadonnées*. Ces modèles vont permettre plus d'interopérabilité car la sémantique qui y est associée permet de s'affranchir des opérations de similarités entre structures logiques pour conclure à une ressemblance éventuelle entre deux éléments de profils donnés. On peut leur reprocher cependant d'être définis pour une classe de profils donnée et non pour toutes classes de profils. Par exemple, les standards :

(a) *FOAF (Friend Of A Friend)* qui sont des modèles d'applications basés sur l'espace de noms décrit à l'adresse `http://xmlns.com/foaf/0.1/` [BM05]. Cet espace de noms permet de créer et d'utiliser des pages personnelles interprétables par machine qui décrivent des personnes, les liens qui existent entre elles et ce qu'elles font via des métadonnées. Ces pages respectent le formalisme RDF et peuvent donc être interrogées via des langages d'interrogation de documents RDF ;

(b) *CC/PP (Composite Capability/Preference Profiles) et CSCP (Comprehensive Structures Context Profiles)* (*cf.* sections 2.3.3.7 et 2.3.3.7) [KRW+04] [HBS02] permettent de décrire uniquement des informations sur le contexte environnemental des utilisateurs comme : leur situation géographique, leur matériel de connexion, les logiciels qu'ils utilisent, etc.

Les travaux de recherche s'orientent actuellement vers la définition de modèles génériques décrivant la structure logique, le contenu mais également la sémantique du modèle. Pour cela, ces travaux se basent généralement sur des modèles et langages de descriptions de ressources que nous décrivons dans les sections suivantes.

2.3 Modèles et langages de description de ressources

Dans cette section, nous présentons les modèles qui sont les plus développés et les plus utilisés pour décrire l'*adressage*, la *structuration* et la *sémantique* des ressources. La plupart de ces travaux ont été menés par le *World Wide Web Consortium ou W3C* (*cf.* `http://www.w3c.org`).

2.3.1 Adressage des ressources

L'une des premières tâches du W3C a été de définir des systèmes d'adressage des ressources sachant que l'on voudrait pouvoir décrire tout type de ressources dans toutes les langues. L'adressage pose donc les problèmes de codification des caractères et de gestion de conflits de noms. Les techniques qui ont été mises en œuvre dans cet objectif sont :

— Universal Resource Identifier (URI), Internationalized Resource Identifier (IRI) et Unicode ;
— Espaces de noms ou Namespaces ;
— XML Path Language ou XPath.

2.3.1.1 Universal Resource Identifier, Internationalized Resource Identifier et Unicode

Le *World Wide Web* (WWW) est défini comme un espace universel contenant tout l'Internet et toutes les autres ressources référencées par les *Iden-*

tificateurs Uniformes de Ressource ou *Universal Resource Identifier* (URIs, parfois communément appelés « URLs »). Le WWW peut être vu comme un service internet sur lequel vont se greffer d'autres services.

Dans la première implémentation du web, relativement peu de technologies étaient mises en œuvre. Il s'agissait des URIs, du protocole HTTP (HyperText Transfer Protocol) et du langage hypertexte HTML (HyperText Markup Language).

Les URIs sont de simples chaînes de caractères qui référencent tout type de ressources Internet telles que : documents, personnes, applications, etc.

Les IRIs (*Internationalized Resource Identifiers*) étendent et renforcent les URIs, en permettant aux personnes d'identifier des ressources web dans leur propre langue (prise en compte de caractères accentués, chinois, etc.). Notons qu'à chaque ressource on associe une et une seule URI ou IRI. Les spécifications des URIs sont décrites dans les documents [RFC 3986] et celles des IRIs dans les documents [RFC 3987].

Les URIs et les IRIs sont basés sur Unicode [3]. Unicode est un système dans lequel les signes ou éléments de toutes les cultures d'écriture connues sont fixés. Par ce système, il devient possible de dire à un ordinateur quel signe on veut voir représenté. La condition préalable est naturellement que l'ordinateur ou le programme exécuté connaisse le système Unicode.

Unicode s'efforce d'englober complètement tous les signes connus des cultures d'écriture présentes et passées. Les caractères sont catalogués par classe et reçoivent une valeur unique en base 2. Tous les caractères et sortes de caractères imaginables sont concernés. Il existe des unicodes pour les caractères de contrôle tels que les traits d'union, les espaces imposés ou les tabulations. Les caractères de formules mathématiques sont représentés tout autant que les signes représentant des syllabes ou des mots des cultures de l'écriture de l'Extrême Orient. Même les différentes parties de caractères comme par exemple les accents ou trémas sur les caractères français ont leur propre unicode. Les caractères peuvent être aussi combinés de façon dynamique. Ainsi il existe bien un « é » français, mais la même lettre peut être créée à partir de « e » et de l'accent sur le caractère. Notons que Unicode est aussi synchronisé avec la norme internationale [ISO/IEC 10646] sur laquelle sont basés HTML depuis la version 4.0 ainsi que XML depuis la version 1.0.

L'adressage des ressources en XML doit aussi permettre la cohabitation de langages différents au sein d'un même document XML tout en gérant les problèmes de conflits de noms identiques entre langages différents. Ceci est possible grâce à l'usage des *espaces de noms*.

3. *cf.* http ://www.unicode.org/

2.3.1.2 Espaces de noms

Les espaces de noms ou namespace[4] [BHL99] permettent de regrouper des métadonnées autour d'un nom de base unique. Pour cela, une adresse URI est associée à chaque espace de noms.

Les espaces de noms sont généralement déclarés dans l'élément racine d'un document XML, par l'intermédiaire de l'attribut *xmlns*. Par exemple, la déclaration de l'espace de noms du langage défini par la recommandation Dublin Core (*cf.* 2.3.3.4), dans un document XML/RDF, est la suivante :

$$< rdf : RDFxmlns : dc = "http : //purl.org/dc/elements/1.1/" >$$

Un espace de noms est généralement représenté par un préfixe : *dc pour dublin core, rdf pour resource description framework, etc.* Les préfixes d'espaces de noms se placent au sein d'un marqueur ou balise XML, avant le nom de la métadonnée. Le préfixe et la métadonnée sont séparés par le caractère deux-points « : ». Le préfixe doit être constitué de lettres (a-z, A-Z et les caractères accentués), de caractères de soulignement (_), de chiffres (0-9) ou de tout autre caractère autorisé par le W3C. Ainsi, la métadonnée *title* du Dublin Core peut apparaître dans un document XML sous les deux formes suivantes :

- *dc : title* qui fait usage du préfixe de l'espace de noms ;
- *http : //purl.org/dc/elements/1.1/title* qui fait usage de l'adresse URI de l'espace de noms associé, complétée par le nom de la métadonnée. Notons que cette forme représente l'adresse URI complète de la métadonnée.

Les espaces de noms multiples permettent d'accueillir plusieurs préfixes différents. On peut donc faire coopérer différents langages au sein d'un même document. Les métadonnées des différents langages seront différenciés par leur préfixe. Ainsi, deux métadonnées de noms identiques ne peuvent être définies dans un même espace de noms. Par contre, des métadonnées portant un nom identique peuvent être définies dans des espaces de noms différents.

On peut également définir un *espace de noms par défaut*. Son objectif est d'éviter de préfixer systématiquement toutes les métadonnées. Ceci permettrait alors de rattacher automatiquement toutes les métadonnées non préfixées à cet espace de noms.

XML donne la possibilité d'utiliser des balises provenant de différents langages à balises au sein d'un même document grâce à la notion d'espace de noms. Cependant, lorsque l'on structure un document XML, on peut répéter la même balise, du même espace de nom, plusieurs fois dans différentes parties. Si l'on souhaite accéder à une balise particulière se trouvant dans une partie spécifique du document, il se pose le problème de conflit de noms de balises identiques, provenant d'un même espace de noms, dans un même

4. *cf.* http ://www.w3.org/TR/REC-xml-names/

document. Ce problème est résolu avec la description des chemins d'accès via des langages comme *XPath ou XML Path Language.*

2.3.1.3 XML Path Language

XML Path Language (XPath [5]) [CD99] est un langage permettant d'adresser des parties de documents XML. Pour cela, on va décrire les chemins d'accès aux différentes parties du document.

La structure d'un document XML est sous la forme d'un arbre où les nœuds sont les différentes balises du document. On peut donc définir des chemins absolus (partant de la racine de l'arbre) ou relatifs (partant d'un nœud quelconque de l'arbre) pour accéder à une balise particulière du document. On peut faire usage également de *chemins de location ou Location Paths* qui utilisent certaines notions de parcours d'arbres comme : *child (fils), descendant (descendant), ancestor (ancêtre), etc.*

Dans la section suivante, nous présentons la structuration (ou organisation) des ressources à travers différentes recommandations et normes.

2.3.2 Structuration des ressources

La définition de normes telles que : SGML, HTML et XML et les travaux du World Wide Web Consortium (W3C) ont permis d'initier de nouvelles perspectives concernant l'exploitation de documents numériques et en particulier des informations textuelles. Grâce aux normes de production et d'échange définies, un document numérique devient alors un ensemble d'objets plus ou moins complexes et non plus seulement une chaîne de caractères. La définition de ces normes est partie d'une étude préalable sur la typologie des structures de documents de façon générale, sachant que tout document a une structure même si celle-ci n'est pas explicite.

2.3.2.1 Typologie de structures

Pour rendre les informations, ou documents XML décrivant des ressources, homogènes et leur exploitation automatique, il est nécessaire de dissocier :

- *le contenu de l'information* ;
- *la structure logique de l'information* : qui correspond à l'organisation hiérarchique de son contenu, mettant ainsi en évidence ses composants. Cette organisation est établie lors de la phase de conception ;
- *la structure physique de l'information* : qui correspond à l'agencement des composants logiques sur le support de restitution. Cet agencement dépend alors essentiellement du média de diffusion : écran, papier, etc.

5. *cf.* `http://www.w3c.org/TR/xpath`

Au concept de structure logique, il convient de distinguer *la structure générique et la structure spécifique* [Sed98] [CMS00]. La *structure générique* exprime l'organisation logique commune à plusieurs informations ou à une classe d'informations. La *structure logique spécifique* à une information est conforme à une structure générique mais est associée à une instance de cette structure générique.

Avec le développement de nouveaux médias de communication et d'accès aux informations, il convient de rajouter à cette typologie les concepts de :

— *structure hypertexte* d'une information qui regroupe l'ensemble des liens que contient cette information (liens intra et extra document par exemple) ;

— *structure spatio-temporelle* qui exprime les mécanismes et les spécifications liés à la nature temporelle et spatiale des objets composant l'information (par exemple, la version d'un document).

Le tableau Tab. 2.1 représente la typologie des structures d'informations.

Types de structures				
Logique		Physique	Hypertexte	Spatio-temporelle
Générique	Spécifique			

Table 2.1 – Typologie des structures d'informations

Partant de ces concepts sur les structures d'informations, trois classes d'informations peuvent être distinguées :

— *les informations structurées* dont la structure logique est explicitement et rigoureusement déclarée (documents SGML ou XML) ;

— *les informations semi-structurées* qui contiennent des données structurelles implicitement déclarées ou partiellement définies dans l'information (documents HTML, par exemple) ;

— *les informations non structurées* qui contiennent peu d'informations sur leur structure logique (un texte brut, par exemple) ;

Une information structurée ou semi-structurée intègre des données additionnelles, telles que les balises, qui renseignent sur les différents éléments de structure qu'elle peut contenir et éventuellement des annotations à caractère sémantique, etc. Plusieurs normes de structuration d'informations ont été ou sont proposées par le W3C à savoir :

1. *SGML (Standard Generalised Markup Language) [ISO-8879, 1986]* jugé assez complexe ;
2. *HTML (HyperText Markup Language)* dont l'absence d'une structure logique explicite cause d'énormes problèmes pour le traitement automatique et la recherche d'information : taux de rappel très faibles. De plus, le mélange des balises contrôlant l'apparence et celles décrivant la structure logique du document rend la réutilisation du texte très

difficile. Ainsi, la DTD (Document Type Definition) HTML4[6] (*cf.* section 2.3.2.3) à laquelle sont conformes tous les documents HTML, n'est pas utilisée avec la rigueur qui serait nécessaire à une extraction simple de la structure logique des documents du web ;

3. *XHTML*[7] *(eXtended HyperText Markup Language)* qui a été définie pour faciliter le passage d'HTML vers XML ;
4. *XML*[8] *(eXtensible Markup Language)* qui est un compromis entre les normes HTML et SGML.

La section suivante présente XML qui est la norme par excellence du W3C pour la structuration des informations.

2.3.2.2 eXtensible Markup Language

eXtensible Markup Language (XML) [BPSM+04] est un langage adapté pour l'échange d'information structurée provenant de sources hétérogènes. XML reprend les caractéristiques de SGML en laissant de côté celles jugées complexes et qui rendaient SGML difficile d'usage. La structure d'un document XML est la suivante :

— un prologue qui est un ensemble de déclarations dont la présence est facultative mais conseillée ;
— un arbre d'éléments qui forme le contenu proprement dit du document ;
— des commentaires et des instructions de traitement dont la présence est facultative et peuvent être, soit dans le prologue, soit dans l'arbre d'éléments.

Avec XML, la séparation de la description structurelle des informations et de la description de leurs réalisations physiques offre d'énormes avantages en terme de facilité d'échange et de production coopérative d'informations et surtout une possibilité énorme en terme de traitement automatisé des informations. La puissance de XML réside dans le fait que toutes les données encodées avec la DTD (*cf.* section 2.3.2.3) sont très fortement structurées, adaptées à l'applicatif et très facilement manipulables par un outil de visualisation. En effet, XML utilise des balises et des attributs comme HTML mais laisse à l'utilisateur la possibilité de définir ses propres balises en fonction de ses besoins.

La norme XML est vouée à devenir le standard d'échange d'informations sur Internet mais est également utilisable dans d'autres cadres. Ainsi, un document XML peut contenir : une base de données Oracle, des tableaux de bord complexes, etc.

6. *cf.* `http://www.w3.org/TR/html4/strict.dtd`
7. *cf.* `http://www.w3.org/TR/2002/REC-xhtml1-20020801/`
8. *cf.* `http://www.w3c.org/TR/2004/REC-xml-20040204/`

La norme XML a été conçue pour être utilisée de deux manières différentes :

1. on peut utiliser une DTD (*cf.* section 2.3.2.3) ou un schéma XML (*cf.* section 2.3.2.4). La DTD ou le schéma XML spécifie la structure logique générique d'une classe de documents et définit les balises à utiliser pour identifier les entités de cette structure. Un document XML faisant appel à une DTD ou à un schéma XML et s'y conformant est dit document *valide* ;
2. un document XML peut être écrit sans DTD ou schéma XML. Il est alors dit document *bien formé*. Dans ce cas, le document doit respecter la syntaxe de la norme XML. Il ne peut comporter aucune ambiguïté dans le balisage : tous les éléments doivent posséder une balise ouvrante et fermante, peuvent être vides et les attributs doivent être entre guillemets. Un document valide est donc avant tout bien formé.

Le tableau Tab. 2.2 illustre un exemple de document XML.

```
<? xml version="1.0" encoding="UTF-8" ?>
<Téléphone_Portable>
      <Marque> Nokia </Marque>
      <Couleur> grise </Couleur>
      <Modèle>
            <nomModèle> 3610 </nomModèle>
            <Poids> 300g </Poids>
            <Autonomie> 2h </Autonomie>
      </Modèle>
</Téléphone_Portable>
```

Table 2.2 – Exemple de document XML

L'objectif d'une DTD ou d'un schéma XML est de définir des contraintes sur des classes de documents conformes à un même modèle (ou structure générique). La norme XML n'impose pas l'utilisation d'une DTD ou d'un schéma pour un document XML mais elle impose le respect des restrictions de structure de la norme XML.

2.3.2.3 Document Type Definition

La norme XML définit la Document Type Definition (DTD [9]) [Ref00] comme une grammaire permettant de vérifier la conformité d'un document XML. Une DTD définit un modèle d'organisation hiérarchique de documents XML en utilisant :

9. *cf.* `http://www.xmlfiles.com/dtd/`

1. *des éléments* organisés hiérarchiquement avec la syntaxe suivante :
   <!ELEMENT nomElément (modèleContenuElément)>.
   Un modèle de contenu est défini pour chaque élément. Le modèle de contenu peut faire appel aux notions d'organisation d'éléments entre eux pour traduire :
   (a) *la séquence* : matérialisée par le symbole de la virgule «,» ;
   (b) *le choix* : matérialisé par le symbole de la barre verticale «|» ;
   On peut utiliser aussi des indicateurs d'occurrence pour traduire :
   (a) *l'optionalité* : matérialisé par le symbole « ?» ;
   (b) *la répétitivité* : matérialisée par le symbole «+» ;
   (c) *l'optionalité et la répétitivité* : matérialisés par le symbole «*».
   Notons que la présence de données dans le contenu d'un élément s'indique par le mot-clé *#PCDATA* qui signifie *Parsed Character Data* ;
2. *des attributs* qui permettent d'ajouter des propriétés à un élément particulier. Un attribut peut-être vu comme une partie du contenu d'un élément. La syntaxe pour définir un attribut est la suivante :
   <!ATTLIST nomElément nomAttribut typeAttribut valeurParDéfaut>.
   Le tableau Tab. 2.3 illustre les différents types d'attributs que l'on peut définir dans une DTD.

Types	**Descriptions**
`CDATA`	la valeur est une chaîne de caractères
`(Val1|Val2|...)`	la valeur est une énumération ou une liste de valeurs possibles
`ID`	la valeur est un identifiant unique
`IDREF`	la valeur est un identifiant d'un autre élément
`IDREFS`	la valeur est une liste d'IDs
`NMTOKEN`	la valeur est un nom XML valide
`NMTOKENS`	la valeur est une liste de noms XML valides
`ENTITY`	la valeur est une entité
`ENTITIES`	la valeur est une liste d'entités
`NOTATION`	la valeur est un nom de notation
`xml:`	la valeur est pré-définie

Table 2.3 – Types pré-définis d'attributs dans une DTD

Les valeurs par défaut des attributs (valeurParDéfaut) peuvent être :
(a) *#DEFAULT value* : l'attribut a une valeur par défaut. Cette valeur est spécifiée entre guillemets ;
(b) *#REQUIRED* : la valeur de l'attribut doit être incluse dans l'élément ;

(c) *#IMPLIED* : la valeur de l'attribut ne doit pas forcément être incluse dans l'élément ;

(d) *#FIXED value* : l'attribut a une valeur fixe.

Le tableau Tab. 2.4, illustre une DTD qui pourrait être utilisée pour décrire la structure logique du document XML du tableau Tab. 2.2.

```
<!– Début de la DTD –>
<!ELEMENT Téléphone_Portable (Marque|Couleur|Modèle)>
<!ATTLIST Téléphone_Portable Marque CDATA #REQUIRED
Couleur CDATA #DEFAULT "grise" Modèle CDATA #REQUIRED>
<!ELEMENT Marque (#PCDATA)>
<!ELEMENT Couleur (#PCDATA)>
<!ELEMENT Modèle (nomModèle|Poids|Autonomie|Prix ?)>
<!ATTLIST Modèle nomModèle CDATA #REQUIRED
Poids CDATA #REQUIRED Autonomie CDATA #REQUIRED
Prix CDATA #REQUIRED>
<!ELEMENT nomModèle (#PCDATA)>
<!ELEMENT Poids (#PCDATA)>
<!ELEMENT Autonomie (#PCDATA)>
<!ELEMENT Prix (#PCDATA)>
<!– Fin de la DTD –>
```

Table 2.4 – Exemple de DTD

La syntaxe utilisée pour décrire des DTD est concise et simple, mais présente l'inconvénient important de ne pas être celle utilisée pour écrire des documents XML. Donc, elle ne peut être analysée par des outils XML standards. C'est une des raisons qui amènent à la définition de nouveaux modèles comme les schémas XML.

2.3.2.4 Schéma XML

Un schéma XML [10] [FW04] indique les éléments qui peuvent être utilisés dans des documents XML et la structure [11] [TBMM04] que ces éléments doivent suivre afin d'être valides pour ce schéma spécifique, en utilisant une syntaxe XML. Les schémas peuvent typer les données [12] [BM04], en documenter leur sens et leur utilisation. Un fichier XML conforme à un schéma particulier est appelé *instance de document*.

Chaque schéma contient deux types de balises :

— *des définitions* : qui permettent de créer de nouveaux types de données simples ou complexes ;

10. *cf.* `http://www.w3c.org/TR/xmlschema-0/`
11. *cf.* `http://www.w3c.org/TR/xmlschema-1/`
12. *cf.* `http://www.w3c.org/TR/xmlschema-2/`

— *des déclarations* : qui décrivent la structure des éléments et des attributs des documents XML qui seront des instances pour le schéma.

Le schéma spécifie les éléments et les attributs qui apparaissent dans un document XML conforme pour ce schéma :

— un *attribut* est une paire (nom, valeur) à l'intérieur d'une balise d'un élément qui précise certaines caractéristiques de l'élément. « nom » est le nom de l'attribut et « valeur » est la valeur de l'attribut. La syntaxe d'un attribut est la suivante : nom="valeur" ;

— un *élément* est une construction structurelle en XML, composée par une balise de début, une balise de fin et par l'information entre les balises. Cette information désigne le *contenu* ou la *valeur* d'un élément. Chaque élément a un type et peut contenir un ensemble d'attributs. La syntaxe d'un élément est de la forme : *<element liste-attributs/>*. Par exemple, l'élément *auteur* peut-être décrit comme suit : *<element name="auteur" type="string"/>*.

Les types de données (DT) sont des composants d'un schéma XML utilisés comme base pour des composants plus grands. Ils peuvent être définis comme des triplets : $DT = (VS, LS, F)$, où :

— VS : l'espace des valeurs (Value Space) est un ensemble de valeurs littérales distinctes ;

— LS : l'espace lexical (Lexical Space) est un ensemble de représentations lexicales sous forme de chaînes de caractères pour les valeurs littérales appartenant à l'espace des valeurs ;

— F : les facettes (Facets) sont des propriétés de l'espace de valeurs. Une facette est une propriété appartenant à un type de donnée qui le distingue d'autres types de données et qui aide à déterminer l'ensemble de valeurs pour un type simple.

La figure FIG. 2.1 présente la hiérarchie des différents types de données pré-définis pour un schéma XML. Avec XML schéma, il existe deux catégories de type de données :

1. *les types simples* : ils ne peuvent contenir ni sous-éléments, ni être qualifiés par des attributs. On distingue différents types de « types simples » :

   (a) les types simples atomiques ou types simples de base : ce sont des types simples comme les entiers, les chaînes de caractères, les dates, etc. (*cf.* FIG. 2.1) ;

   (b) les types simples dérivés par restriction : ce sont des types simples que l'on définit en réduisant le domaine de définition des types simples atomiques existants. Ainsi, le type simple *typeISBN* peut-être défini par restriction du type chaîne de caractères (*xsd : string*), comme dans le tableau TAB. 2.5 ;

   (c) les types listes : ce sont des listes de types simples atomiques ou de types simples dérivés par restriction. Le tableau TAB. 2.6

FIGURE 2.1 – Hiérarchie des types de données pré-définis pour XML schéma

illustre un exemple de création d'un *type liste*. Il s'agit d'une liste de « typeISBN » qui est un type simple résultant d'une restriction du type simple « xsd : string » décrit dans le tableau TAB. 2.5.

Notons que l'on ne peut créer de nouveaux types listes ni à partir des types listes existants, ni à partir des types complexes. Par ailleurs, plusieurs facettes (ou propriétés) peuvent être appliquées au type liste : *length, minLength, maxLength, enumeration.* Par exemple, on peut définir une liste d'exactement six numéros d'ISBN de livre en utilisant la facette « length » ;

(d) les types unions : ce sont des listes de types résultant de l'union (ou combinaison) de types atomiques et/ou de types listes et/ou de types complexes.

Les types atomiques et listes permettent à un élément (ou une valeur) d'attribut d'être composé de une (cas d'un élément de type atomique) ou plusieurs (cas d'un élément de type liste) instances

```
<xsd : simpleType name="typeISBN">
        <xsd : restriction base="xsd : string">
                <xsd : pattern value="[0-9]{3}[-][0-9]{3}[-][0-9]{3}"/>
        </xsd : restriction>
</xsd : simpleType>
```

TABLE 2.5 – Exemple de création d'un type simple par restriction

```
<xsd : simpleType name="listOfMyIsbnType">
        <xsd : list itemType="typeISBN"/>
</xsd : simpleType>
```

TABLE 2.6 – Exemple de création de «type liste»

d'un type atomique. Par contre, un type union permet à un élément (ou valeur) d'attribut d'être composé d'une ou plusieurs instances d'un type résultant de l'union de plusieurs types atomiques et/ou listes. Pour donner un exemple, nous allons créer un type union pour représenter *les numéros d'ISBN de livres d'une bibliothèque par catégorie de livre*. Dans notre exemple du tableau TAB. 2.7, le type union appelé *ISBNUnion* est construit à partir : d'un *type atomique* nommé *BookCategory* qui est obtenu par restriction du type *xsd : string* ; d'un *type liste* nommé *listOfMyIsbnType* qui est défini dans tableau TAB. 2.6.

```
<xsd : simpleType name="ISBNUnion">
        <xsd : union memberTypes="BookCategory listOfMyIsbnType"/>
</xsd : simpleType>
```

TABLE 2.7 – Exemple de création de «type union»

On peut donc créer de nouveaux types simples par dérivation de types simples existants. Cette dérivation peut se faire par : *restriction, liste ou union* ;

2. *les types complexes* : ce sont des types composés d'éléments XML. Ils peuvent contenir des sous-éléments et être qualifiés par des attributs (*cf.* TAB. 2.8). Pour créer un type complexe, on peut donc procéder par combinaison d'éléments xml de type simple ou complexe. On peut également procéder par dérivation de types complexes existants. Cette dérivation peut se faire par : restriction (suppression d'éléments) ou extension (ajout de nouveaux éléments dans le type).

   Le tableau TAB. 2.8 permet de définir le type complexe *typeLivre* en combinant différents éléments de type simple.

```
<xsd : complexType name="typeLivre">
        <xsd : sequence>
                <xsd : element name="titre" type="xsd :string"/>
                <xsd : element name="auteur" type="xsd :string"/>
        </xsd : sequence>
        <xsd : attribute name="isbn" type="typeISBN"/>
</xsd : complexType>
```

TABLE 2.8 – Exemple de création d'un type complexe

Ainsi, un schéma XML correspondant à notre exemple de document XML sur le téléphone portable (*cf.* TAB. 2.2) est illustré dans le tableau TAB. 2.9.

Notons que dans les tableaux TAB. 2.5, TAB. 2.8, TAB. 2.6, TAB. 2.7, et TAB. 2.9, le préfixe *« xsd: »* représente l'espace de noms (*cf.* section 2.3.1.2) de XML schéma défini à l'adresse : `http://www.w3.org/2001/XMLSchema/`.

2.3.2.5 Document Type Definition vs XML Schéma

Un schéma est fonctionnellement équivalent à une DTD mais est écrit en XML. Un des avantages de l'utilisation d'un schéma à la place d'une DTD est l'extensibilité. Un schéma offre la possibilité de créer de nouveaux types tandis qu'une DTD ne permet de spécifier un nouvel élément qu'en terme de chaîne de caractères ou d'autres éléments. Une étude comparative sur l'utilisation d'un schéma par rapport à une DTD est illustrée dans le tableau TAB. 2.10.

XML structure les données comme un langage de programmation orienté objet structure les programmes. Il existe donc un modèle objet pour les documents XML appelé DOM (Document Object Model).

2.3.2.6 Document Object Model

Document Object Model (DOM [13]) [CKR04] est une interface de programmation (API : Application Programming Interface) pour des documents, indépendante des langages de programmation et de script. Il spécifie comment les documents XML sont présentés sous la forme d'objets, de manière à ce qu'ils puissent être utilisés dans des programmes orientés objets pour la mise à jour des contenus, structures et styles de documents (*cf.* section 2.3.2.8). En d'autres termes, le modèle de document objet définit les outils nécessaires à l'accès et à la manipulation de documents structurés sous forme d'arbres d'objets typés.

La séparation stricte entre le contenu, la structure et la mise en forme d'un document a de nombreux avantages. Avec le DOM, on peut par exemple

13. *cf.* `http://www.w3.org/TR/2004/REC-DOM-Level-3-Val-20040127/`

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="Téléphone_Portable"
    type="typeTéléphone_Portable"/>

    <xsd:complexType name="typeTéléphone_Portable">
        <xsd:sequence>
            <xsd:element name="Marque" type="xsd:string"/>
            <xsd:element name="Couleur" type="xsd:string"/>
            <xsd:element name="Modèle" type="typeModèle"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="typeModèle">
        <xsd:sequence>
            <xsd:element name="nomModèle" type="xsd:string"/>
            <xsd:element name="Poids" type="xsd:string"/>
            <xsd:element name="Autonomie" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

TABLE 2.9 – Exemple de schéma XML

effectuer des mises en forme sur une partie d'un document pendant que les autres parties de ce même document sont affichées en réponse à une requête utilisateur.

La représentation DOM de l'exemple de document XML du tableau TAB. 2.2 est décrite dans la figure FIG. 2.2.

La racine d'un DOM est composée de plusieurs classes ou objets typés. Chacune de ces classes a des instances contenant les valeurs correspondantes. Rappelons que deux concepts peuvent être utilisés pour définir une classe de documents XML :

— la DTD (*cf.* section 2.3.2.3) ;
— le schéma XML (*cf.* section 2.3.2.4) qui est l'équivalent d'un schéma de base de données orienté objet qui précise la structure et les types de données.

De plus, la structure DOM d'un document peut faire apparaître plusieurs nœuds de même nom mais auxquels on accède par des chemins différents. La définition d'un chemin menant à un nœud spécifique est appelé XPath (*cf.* section 2.3.1.3). Ce chemin permet de désambiguïser les conflits de noms de balises qui peuvent survenir dans un même document XML.

Caractéristiques	DTD	Schéma XML
spécifie le modèle du contenu pour un ensemble de documents XML	oui	oui
peut être partagé	oui	oui
offre une description formalisée et complète du vocabulaire XML	oui	oui
utilise la syntaxe XML	non	oui
permet l'utilisation des éléments globaux et locaux	non	oui
offre la possibilité de spécifier les types de données	non	oui
est extensible	non	oui

TABLE 2.10 – Comparaison entre une DTD et un schéma XML [Scu02]

Par ailleurs, la définition des hyperliens dans un document XML se fait à l'aide du langage XLL (eXtensible Link Language). Les hyperliens vont permettre de décrire la structure hypertexte d'un document XML.

2.3.2.7 Structure hypertexte de documents XML : eXtensible Link Language

eXtensible Link Language (XLink ou XLL [14]) [DMO01] est le langage de description des liens hypertextes en XML. XLL étend les hyperliens définis par HTML en permettant, entre autres :

— des liens qui pourront être bidirectionnels ou gérés dans un fichier extérieur à l'instance de document ;

— des liens vers des cibles sur Internet non balisées au préalable.

Des attributs ajoutés aux liens permettent de définir le type du lien (lien vers une définition, lien extérieur, etc.). Ainsi, un lien XML peut être une URL, comme dans HTML ou un XPTR (eXtended PoinTer).

Un XPTR [GMM03b] [GMM03a] [DDMM03] ressemble à une URL mais il permet d'exprimer des liens plus complexes car au lieu de pointer sur des documents web, il pointe sur des éléments de données au sein d'un fichier XML.

Par ailleurs, l'affichage ou l'impression des données XML impose de définir la façon dont les données s'affichent (écran) ou s'impriment (papier). Pour cela, on utilise des feuilles de styles XSL (eXtensible Stylesheet Language) qui permettent de définir la structure physique d'un document XML.

14. *cf.* `http://www.w3.org/TR/xlink/`

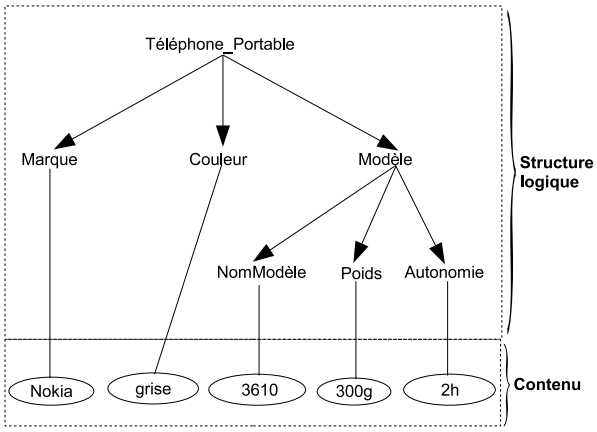

FIGURE 2.2 – Représentation DOM d'un exemple de document XML

2.3.2.8 Structure physique de documents XML : eXtensible Stylesheet Language

eXtensible Stylesheet Language (XSL [15]) [Ber04] est le langage utilisé pour la définition des feuilles de style qui sont associées aux documents XML. C'est le fichier XSL qui permet de définir les propriétés typographiques et graphiques des éléments d'un document : police, couleur, etc.

Une feuille de style XSL se compose de règles de construction décrivant comment les éléments du fichier XML doivent être transformés (*cf.* XSLT [16] [Kay05]) vers le format de sortie qui est par exemple un document HTML ou un texte ASCII.

Les règles de construction de documents XML se divisent en deux parties :

— *la partie modèle* qui identifie le type des éléments sources (XML Schéma, DOM, etc.) ;

— *la partie traitement* qui décrit ce qu'il faut faire avec les éléments qui correspondent à *la partie modèle* (XSL et XSLT).

XML fournit donc une syntaxe de structuration de données. Cependant, il ne fournit aucune information pour décrire la sémantique des balises. Pour pallier cette limite, des langages de métadonnées ont été mis sur pied afin d'exprimer le sens des balises XML.

15. *cf.* `http://www.w3c.org/TR/xsl11/`
16. *cf.* `http://www.w3.org/TR/xslt20/`

2.3.3 Sémantique des informations : langages de métadonnées

Une métadonnée est un terme publié auquel est associé une définition unique. Ce terme peut décrire soit un *concept* soit une *relation entre concepts.* Il existe de nombreux langages de métadonnées pour définir la sémantique de ressources. Ces langages sont généralement définis dans des espaces de noms et peuvent ainsi être combinés pour la description de ressources. Certains de ces langages (RDF, RDFS, OWL, Dublin-Core, WordNet, etc.) sont présentés dans les sections suivantes.

2.3.3.1 Resource Description Framework : modèle et syntaxe

Le W3C travaille également sur la normalisation d'applications génériques permettant de décrire des graphes de documents explicitant des relations sémantiques. C'est le cas de RDF (Resource Description Framework) qui constitue un outil très puissant pour l'indexation et la recherche de documents.

En RDF [17] [KC04] [Hay04], la sémantique est codée sous la forme d'un triplet du genre sujet (ou ressource), verbe (ou prédicat ou propriété) et complément (ou objet ou valeur) d'une phrase élémentaire : notée *[sujet, verbe, complément]* ou *[sujet, prédicat, objet]* ou *[ressource, propriété, valeur].* Ce triplet peut être écrit à l'aide de balises XML.

Un document RDF fait l'assertion que des choses particulières (personne, page web ou autres) ont des propriétés (comme *est la soeur de*, *est l'auteur de*, etc.) avec certaines valeurs (*une autre personne, une autre page web, etc.*). Notons qu'une valeur peut-être une autre ressource décrite par des triplets également. Cette structure est une manière naturelle de décrire la majorité des informations traitées par des machines (et même par l'homme). Ainsi, on pourra exprimer par exemple le fait que : *monsieur X soit l'auteur de la page web Y.*

Le sujet et le complément, s'il s'agit d'une autre ressource, sont identifiés par une *URI (Universal Resource Identifier)* exactement comme un lien dans une page web (les URLs, *Uniform Resource Locations*, sont des types d'URI). Les verbes sont également identifiés par des URIs. De plus, il est possible de définir un nouveau *concept* (ou un nouveau verbe) quelque part sur le web, en associant une URI à ce concept. Le conflit possible entre les noms de concepts est résolu ici par les URI et en faisant usage des espaces de noms. Ainsi, à chaque concept il sera assigné une et une seule URI dont la base (ou préfixe) est l'URI de l'espace de noms associé, ce qui permettra de faire la différence entre les concepts ayant le même nom. Les URIs et/ou espaces de noms assurent que les concepts ne sont pas que des mots dans un document

17. *cf.* `http://www.w3.org/TR/rdf-concepts/` et `http://www.w3.org/TR/rdf-mt/`

mais qu'ils sont associés à une définition unique que tout le monde peut trouver sur le web.

La figure FIG. 2.3 représente un graphe RDF et la liste de ses triplets, associé à l'exemple de document XML du tableau TAB. 2.2, présenté plus haut.

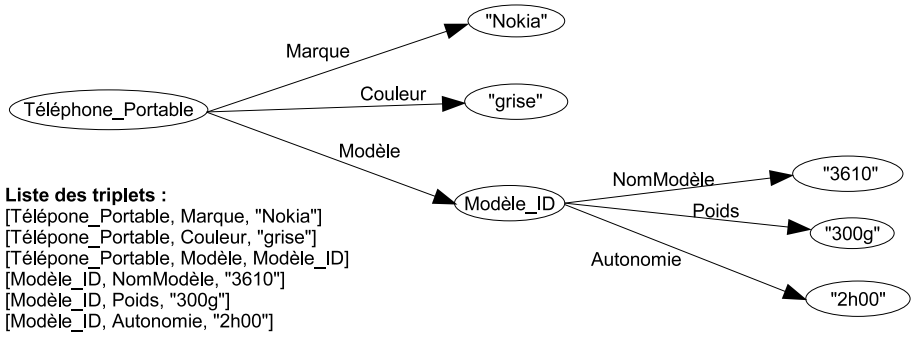

FIGURE 2.3 – Exemple de graphe RDF et liste de triplets associés

Notons que RDF fournit un vocabulaire de base qui est décrit dans son espace de noms à l'adresse suivante : « `http://www.w3.org/1999/02/22-rdf-syntax-ns#` ». Le vocabulaire RDF comprend les noms réservés suivants [LS99] :

- *noms de syntaxe* (pas de concepts) : RDF, Description, ID, about, parseType, resource, li, nodeID, datatype ;
- *noms de classes* : Seq, Bag, Alt, Statement, Property, XMLLiteral, List ;
- *noms de propriétés* : subject, predicate, object, type, value, first, rest, _n où n est un entier plus grand que zéro ;
- *noms de ressources* : nil.

Tous ces noms ont une référence URI. Par exemple, `rdf:type` a pour *RDF URI reference* : « `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` ». Le vocabulaire RDF permet entre autre :

- de décrire une ressource ou une valeur d'un triplet avec le concept `rdf:description` ;
- de préciser une valeur représentée par un symbole, c'est-à-dire une valeur non associée à une URI, avec le concept `rdf:ID` ;
- de préciser une valeur associée à une URI avec le concept `rdf:about` ;
- etc.

Le vocabulaire RDF peut être utilisé pour décrire des documents RDF (en XML). Ainsi, le document RDF associé au graphe de la figure FIG. 2.3 est décrit dans le tableau TAB 2.11. Avec *« ex : »*, le préfixe de l'espace de noms associé aux concepts propres à notre exemple et *« rdf : »* le préfixe de l'espace de noms associé au langage RDF.

```
<rdf:RDF xmlns:ex="http://mon_url.fr/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description
         rdf:about="#Téléphone_Portable">
         <ex:Marque>
            <rdf:Description rdf:ID="Nokia"/>
         </ex:Marque>
         <ex:Couleur>
            <rdf:Description rdf:ID="grise"/>
         </ex:Couleur>
         <ex:Modèle>
            <rdf:Description
               rdf:about="#Modèle_ID">
               <ex:NomModèle>
                  <rdf:Description rdf:ID="3610"/>
               </ex:NomModèle>
               <ex:Poids>
                  <rdf:Description rdf:ID="300 g"/>
               </ex:Poids>
               <ex:Autonomie>
                  <rdf:Description rdf:ID="2 heures"/>
               </ex:Autonomie>
            <rdf:Description/>
         </ex:Modèle>
      <rdf:Description/>
</rdf:RDF>
```

TABLE 2.11 – Exemple de document RDF

Le vocabulaire de RDF est étendu à travers le langage *RDF Schéma*.

2.3.3.2 RDF Schema

RDF Schema (RDFS [18]) [BG04] va permettre de typer les différents éléments d'un triplet RDF. On peut regrouper le vocabulaire RDFS en différents groupes :

1. *description de classes (sujet ou objet d'un triplet RDF)* :
   — `rdfs:Resource` : qui permet de définir une ressource. Toute chose décrite par rdf est une ressource et donc est une instance de `rdfs:Resource`. `rdfs:Resource` est une instance de `rdfs:Class` ;

18. *cf.* `http://www.w3.org/TR/rdf-schema/`

— `rdfs:Class` : c'est la classe des ressources qui sont décrites en RDF ;
— `rdfs:Literal` : c'est la classe des valeurs d'un littéral comme par exemple *integer* pour les entiers ou *string* pour les chaînes de caractères. `rdfs:Literal` est une instance de `rdfs:Class` et une sous-classe de `rdfs:Resource`. Notons que une classe a est une sous-classe d'une classe b si toutes les instances de a sont des instances de b ;
— `rdfs:Datatype` : c'est la classe des types de données. `rdfs:Datatype` est à la fois une instance et une sous-classe de `rdfs:Class`. De plus, chaque instance de `rdfs:Datatype` est une sous-classe de `rdfs:Literal` ;
— `rdf:XMLLiteral` : c'est une classe des valeurs des littéraux XML. `rdf:XMLLiteral` est une instance de `rdfs:Datatype` et une sous-classe de `rdfs:Literal` ;
— `rdf:Property` : c'est la classe des propriétés RDF. `rdf:Property` est une instance de `rdfs:Class`.

2. *description de propriétés ou prédicats* :
— `rdf:type` : c'est une instance de `rdf:Property` qui est utilisée pour dire qu'une ressource est une instance d'une classe. Par exemple, R `rdf:type` C, traduit le fait que C est une instance de `rdfs:Class` et R une instance de C ;
— `rdfs:subClassOf` : C1 `rdfs:subClassOf` C2 traduit le fait que C1 est une instance de `rdfs:Class`, C2 est une instance de `rdfs:Class` et C1 est une sous-classe de C2. La propriété `rdfs:subClassOf` est transitive.
— `rdfs:subPropertyOf` : P1 `rdfs:subPropertyOf` P2 traduit le fait que P1 est une instance de `rdf:Property`, P2 est une instance de `rdf:Property` et P1 est une sous-propriété de P2. La propriété `rdfs:subPropertyOf` est transitive ;
— `rdfs:domain` : c'est une instance de `rdf:Property` qui est utilisée pour signifier que toute ressource ayant une certaine propriété est une instance d'une ou de plusieurs classes. Par exemple, P `rdfs:domain` C traduit le fait que P est une instance de la classe `rdf:Property`, que C est une instance de la classe `rdfs:Class` et que les sujets des triplets dont le prédicat est P sont des instances de la classe C ;
— `rdfs:range` : c'est une instance de `rdf:Property` qui est utilisée pour dire que les valeurs d'une propriété sont des instances d'une ou de plusieurs classes. Par exemple, P `rdfs:range` C permet de dire que P est une instance de la classe `rdf:Property`, C est une instance de la classe `rdfs:Class` et que les objets des triplets qui ont pour prédicat P sont des instances de la classe C ;

— `rdfs:label` : R `rdfs:label` L traduit le fait que L est une étiquette de R (label) compréhensible par l'homme ;
— `rdfs:comment` : R `rdfs:comment` L traduit le fait que L est une description de R compréhensible par l'homme.

3. *Conteneur de classes et de propriétés* : `rdfs:Container`, `rdf:Bag`, `rdf:Seq`, `rdf:Alt`, `rdfs:ContainerMembershipProperty`, `rdfs:member` ;
4. *description de collections RDF* : `rdf:List`, `rdf:first`, `rdf:rest`, `rdf:nil` ;
5. *Réification de vocabulaire* : `rdf:Statement`, `rdf:subject`, `rdf:predicate`, `rdf:object` ;
6. *Propriétés d'utilité* : `rdfs:seeAlso`, `rdfs:isDefinedBy`, `rdf:value`.

L'espace de noms associé à RDFS se trouve à l'adresse : « `http://www.w3.org/2000/01/rdf-scheme#` ». Notons également l'existence de *« RDF Semantics*[19] *»* [Hay04] qui est une spécification précise de la sémantique de RDF et RDFS. Il permet l'interprétation du vocabulaire de RDF et RDFS.

RDFS permet d'enrichir le vocabulaire de base de RDF. Cependant, les relations sémantiques de RDF/RDFS ne sont pas assez puissantes pour permettre de définir certaines propriétés ou contraintes sur des ensembles de données comme : les cardinalités, la disjonction, etc. Pour pallier cet handicap, le langage *Ontology Web Language ou OWL* a été mis sur pied.

2.3.3.3 Ontology Web Language

Web Ontology Language (OWL[20]) [BvHH+04] est une révision de *Darpa Agent Markup Language* et d'*Ontology Interchange Language* (en abrégé DAM-OIL[21]) [CvHH+01]. OWL ajoute plus de vocabulaires pour la description de propriétés et de classes : relations entre classes (*disjonction, équivalence, égalité, etc.*), cardinalité des propriétés (*exactement un, un ou plusieurs, etc.*), typage plus riche des propriétés (*symétrie, transitivité, etc.*), définition de classes énumérées, etc. OWL a été conçu pour répondre au besoin d'un langage d'ontologie pour le web.

L'espace de noms associé à OWL se trouve à l'adresse : « `http://www.w3.org/2002/07/owl#` » et son vocabulaire peut être subdivisé également en différentes catégories :

1. *la description de classes* qui permet de décrire :
   — une énumération avec le mot réservé `owl:oneOf` où chaque élément de l'énumération est décrit avec le mot réservé `owl:Thing` ;

19. *cf.* `http://www.w3.org/TR/rdf-mt/`
20. *cf.* `http://www.w3.org/TR/owl-ref/`
21. *cf.* `http://www.w3.org/TR/daml+oil-reference`

— des restrictions sur des propriétés en explicitant des contraintes : sur des valeurs avec les mots réservés `owl:allValuesFrom`, `owl:someValuesFrom`, `owl:hasValue` ; sur des cardinalités avec les mots réservés `owl:maxCardinality`, `owl:minCardinality`, `owl:cardinality` ;
— une intersection avec le mot réservé `owl:intersectionOf`, une union avec le mot réservé `owl:unionOf` et un complément avec le mot réservé `owl:complementOf` ;
— des axiomes sur des classes comme : l'équivalence avec le mot réservé `owl:equivalentClass`, la disjonction avec le mot réservé `owl:disjointWith`, etc.

2. *la description de propriétés* qui permet :
— de réutiliser certains mots de RDFS comme : `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range` ;
— de définir des relations entre propriétés avec les mots réservés : `owl:equivalentProperty`, `owl:inverseOf` ;
— de définir des restrictions globales sur les cardinalités des propriétés avec : `owl:FunctionalProperty`, `owl:InverseFunctionalProperty` ;
— de définir des caractéristiques logiques sur des propriétés au travers des mots réservés :
`owl:TransitiveProperty`, `owl:SymmetricProperty` qui sont des sous-classes de `owl:ObjectProperty`.
Pour définir ces caractéristiques sur des propriétés, cela implique de considérer une propriété comme une ressource décrite par des triplets. Ainsi, pour définir une propriété P comme étant symétrique, on aurait le triplet : *[P, rdf:type, owl:SymmetricProperty]*.
Le tableau Tab. 2.12 décrit en OWL une propriété symétrique nommée *friendOf*.
Notons qu'une propriété peut être sujet, verbe voire objet d'un triplet. Dans ce cas, c'est l'URI qui permet de dire qu'il s'agit du même élément, même s'il occupe indifféremment les places de sujet, verbe ou objet dans la description d'un document. Par ailleurs, on peut être amené à utiliser des ressources vides pour remplacer des sujets ou objets non-définis de certains triplets.

```
<owl:ObjectProperty rdf:ID="friendOf">
    <rdf:type rdf:ressource="&owl;SymmetricProperty">
    <rdfs:domain rdf:ressource="#Human">
    </rdfs:range rdf:ressource="#Human">
</owl:ObjectProperty>
```

Table 2.12 – Illustration de la définition d'une propriété symétrique avec le langage OWL

3. *la description des types de données* qui permet de spécifier des types de données : les types de données RDF, la classe RDFS `rdfs:Literal`, le type énuméré (en utilisant le constructeur `owl:oneOf`) ;
4. *la description de particularités* qui sont des faits sur des classes *d'individus* ou *des valeurs de propriétés* ou encore sur *des entités individuelles* avec les mots réservés : `owl:sameAs`, `owl:differentFrom`, `owl:AllDifferent` ;
5. *les autres descriptions* qui sont relatives aux annotations, aux versions des informations, etc.

OWL est compatible avec la syntaxe XML/RDF et ré-utilise donc les mots du vocabulaire RDF/RDFS. Notons cependant que les propriétés *d'union, d'intersection et de complément sont des opérations* et donc ne sauraient être décrites dans un graphe RDF qui représente essentiellement des informations statiques. Ces opérations vont donc être décrites dans un graphe RDF par les résultats de leur évaluation. Ainsi, si l'on veut exprimer la propriété d'union, on peut la définir comme dans le tableau Tab. 2.13. Dans ce tableau, on décrit une classe pour laquelle l'extension (ou liste des éléments) comprend : *Livre1, Livre2 et Livre3.*

```
<owl : Class>
    <owl : unionOf rdf : parseType="Collection">
        <owl : Class>
            <owl : oneOf rdf : parseType="Collection">
                <owl : Thing rdf : about="#Livre1" />
                <owl : Thing rdf : about="#Livre2" />
            </owl : oneOf>
        </owl : Class>
        <owl : Class>
            <owl : oneOf rdf : parseType="Collection">
                <owl : Thing rdf : about="#Livre3" />
                <owl : Thing rdf : about="#Livre1" />
            </owl : oneOf>
        </owl : Class>
    </owl : unionOf>
</owl : Class>
```

Table 2.13 – Illustration de la propriété d'union avec le langage OWL

Comparaisons entre OWL Full, OWL DL et OWL Lite

Le langage OWL complet est appelé *OWL Full* pour marquer la différence avec les sous-ensembles du langage que sont : *OWL Lite et OWL DL*

(où DL signifie Description Logic). On peut dire que OWL Full est une généralisation de OWL DL qui est à son tour une généralisation de OWL Lite. OWL Full allège les contraintes de OWL DL (il est donc plus flexible) afin de rendre disponibles certains éléments qui sont très utilisés dans les bases de données et les systèmes de représentation de connaissances. Cependant, cette flexibilité autorise de violer certaines contraintes des systèmes basés sur la logique de description.

OWL Full et OWL DL supportent le même ensemble de constructions OWL. Leur différence repose sur les restrictions d'utilisations de certains éléments. OWL Full permet une coopération libre entre éléments OWL et éléments de RDF Schema (RDFS). Par contre, OWL DL fixe des contraintes sur le principe de coopération avec RDF et oblige une disjonction entre classes, propriétés, individus et types de données. Cette disjonction implique que des prédicats définis pour un type d'éléments donné ne peuvent être utilisés sur d'autres types d'éléments. Ainsi, avec OWL DL l'ensemble des propriétés d'objets est disjoint de celui des propriétés de types de données. Les quatre *caractéristiques de propriétés* suivantes : *inverse of, inverse functional, symmetric et transitive* ne peuvent donc jamais être spécifiées pour des propriétés de types de données (par exemple, les types de XMLSchema). La principale raison de l'existence de OWL DL est liée au fait que des constructeurs d'outils ont développé des systèmes puissants de raisonnement qui supportent des ontologies contraintes par les restrictions imposées par OWL DL. L'objectif de OWL DL est donc de permettre une ré-utilisation simple du patrimoine existant de systèmes de raisonnement basés sur de la logique de description. La définition formelle des différences entre OWL Full et OWL DL est détaillée dans un document intitulé : *OWL Semantics and Abstract Syntax* [PSHH04].

OWL Lite est un sous-langage de OWL DL qui supporte seulement un sous-ensemble de construction OWL. OWL Lite cible particulièrement les développeurs d'outils qui veulent utiliser OWL mais souhaiteraient commencer avec un ensemble simple et relativement basique de caractéristiques du langage. OWL Lite supporte les mêmes restrictions sémantiques que OWL DL, cependant elle introduit de nouvelles restrictions sur certains éléments. Par exemple, OWL Lite interdit l'utilisation de : *owl : oneOf, owl : unionOf, owl : complementOf, owl : hasValue, owl : disjointWith, owl : DataRange*. De plus, OWL Lite restreint le type des sujets (*rdfs : domain*) ou objets (*rdfs : range*) de certaines propriétés. La description formelle du sous-ensemble du langage OWL autorisé par OWL Lite est décrit dans un document intitulé : *OWL Semantics and Abstract Syntax document* [PSHH04].

En résumé, OWL Lite a été conçu pour une implémentation facile et pour donner à l'utilisateur un sous-ensemble fonctionnel afin de commencer à utiliser le langage OWL. OWL DL, quant à lui, a été conçu pour supporter les contraintes de la *logique de description* et pour fournir un langage qui a des propriétés interprétables par des systèmes informatiques de raisonnement

existants. Les sous-langages de OWL permettent de garantir la convergence des inférences tandis qu'avec OWL Full, toutes les inférences ne sont pas forcément décidables.

Ainsi, le choix entre ces différents langages OWL va être dicté par les priorités de l'application à mettre en œuvre : la flexibilité est davantage garantie avec OWL Full, la décidabilité avec OWL DL et le prototypage rapide avec OWL Lite.

Compatibilités entre XML, RDF, RDFS et OWL

De façon générale, on a les compatibilités suivantes entre documents XML, RDF, RDFS et OWL :

- un document RDF/RDFS/OWL est un document XML ;
- un document RDF/RDFS est un document OWL Full ;
- un document OWL (Full, DL ou Lite) peut être transformé en un document RDF/RDFS ;
- un document XML n'est pas toujours un document RDF/RDFS/-OWL.

Un autre langage de métadonnées, appelé Dublin Core, a également été proposé pour décrire des documents. Ce langage est actuellement considéré comme une norme.

2.3.3.4 Dublin Core

Le Dublin Core [22] est un schéma descriptif normalisé né à l'issu d'un meeting sur les métadonnées. Il entre dans les nouvelles propositions de normes destinées à améliorer la RI sur internet et sur le web. Il s'agit d'une initiative convoitant la consolidation de la normalisation des métadonnées.

Le Dublin Core a pour objectif d'être assez simple pour pouvoir être compris par des non spécialistes et assez souple pour pouvoir être étendu dans le futur. Il est encore peu utilisé sur le web alors qu'il peut être exploité avec des technologies existantes, notamment les éléments <META> d'HTML (exploitables par les moteurs de recherche comme Google, Alta Vista, etc.). En effet, dans les documents HTML, les balises META (pour dire METADATA) permettent de définir un certain nombre d'informations supplémentaires sur le contenu d'une page web.

Le Dublin Core propose une quinzaine de propriétés descriptives de base (métadonnées), relatives pour les unes au contenu de la ressource, pour d'autres à la propriété intellectuelle de ce contenu et aux caractéristiques physiques de la ressource : *title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights.* Ces métadonnées ont été étendues par raffinement des métadonnées de base existantes. Ainsi, la métadonnée *date* peut-être raffinée par les métadonnées :

22. *cf.* `http://dublincore.org/documents/dces/`

dateSubmitted pour préciser la date de soumission d'un document, *dateCopyrighted* pour préciser la date de prise en compte du copyright, etc. D'autres métadonnées ont également été ajoutées aux métadonnées de base du Dublin Core (*cf.* `http://dublincore.org/documents/dcmi-terms/#H3`) : *abstract, available, audience, etc.* Ces métadonnées sont compatibles avec les langages : RDF/RDFS/OWL.

Les espaces de noms associé aux métadonnées du Dublin Core se trouvent aux adresses suivantes :

- *http ://purl.org/dc/elements/1.1/* qui décrit les quinze éléments de base du Dublin Core. Cet espace de noms est généralement préfixé par *« dc : »* ;
- *http ://purl.org/dc/terms/* qui décrit des métadonnées créées par raffinement des éléments de base du Dublin Core ainsi que d'autres métadonnées. Cet espace de noms est généralement préfixé par *« dcterms : »* ;
- *http ://purl.org/dc/dcmitype/* qui décrit des métadonnées permettant de typer les ressources : *text, event, image, sound, dataset, etc.* Cet espace de noms est généralement préfixé par *« dctype : »*.

La ré-utilisation de métadonnées du Dublin Core se fait en incluant ces espaces de noms dans un document et en faisant précéder les métadonnées par les préfixes ou l'URI des espaces de noms associés.

Par ailleurs, pour décrire des métadonnées dans des documents multimédia, la norme la plus usité actuellement est la norme MPEG-7 du groupe MPEG (*Moving Picture Experts Group ou MPEG*).

2.3.3.5 MPEG-7

MPEG-7 [Mar04] est un standard ISO/IEC développé par le comité MPEG pour la description de contenu de données multimédia qui supporte un certain degré d'interprétation du sens des informations. MPEG-7, formellement appelé *Multimedia Content Description Interface*, fournit un ensemble d'outils standards pour décrire des contenus multimédia. Les principaux éléments de MPEG-7 sont :

- *les outils de descriptions* composés : de descripteurs (D) qui définissent la syntaxe et la sémantique de chaque caractéristique ou métadonnée (par exemple : Video Segment, Ball, Player, GoalKeeper, etc.) ; de schémas de description (DS) qui spécifient la structure et la sémantique des relations (par exemple : IsCloseTo, RightOf, SameAs, etc.) qui existent entre composants MPEG-7 i.e entre des descripteurs et/ou des schémas de description ;
- *un langage de définition de description (DDL)*, décrit avec XML-schéma, qui définit la syntaxe des outils de description de MPEG-7. Il permet la création de nouveaux schémas de description et de nouveaux descripteurs mais également autorise l'extension et la mo-

dification des schémas de description existants ;

— *un système d'outils* qui supporte la représentation binaire pour : le stockage, les mécanismes de transmission, la synchronisation des descriptions avec le contenu, la gestion et la protection de la propriété intellectuelle, etc.

Par ailleurs, on observe actuellement la transformation de langages de métadonnées existants ou d'ontologies existantes au format RDF/OWL comme l'ontologie de domaine WordNet.

2.3.3.6 WordNet : représentation RDF/OWL

Les versions les plus récentes de la conversion de Wordnet au format RDF/OWL sont disponibles à l'adresse suivante : *http ://wordnet.princeton.edu/wn/*. Cette adresse constitue un espace de noms de Wordnet préfixé généralement par *« wn : »*. Un autre espace de noms qui correspond à la version 2.0 de Wordnet est décrit plutôt à l'adresse : `http://wordnet.princeton.edu/wn20/` et est généralement préfixé par *« wn20 : »*.

Le concept de base de Wordnet est le *synset*. Un synset est un ensemble de mots synonymes comme : *{car, auto, automobile, machine, motorcar}*. Un mot peut appartenir à plusieurs synsets mais un sens pour un mot correspond à un et un seul synset. Certains mots peuvent donc être polysémiques. De même, un synset peut représenter plusieurs sens de mots. Le *sens d'un mot (Word Sense)* est représenté lexicalement par un et un seul mot. Il existe 4 types de synset disjoints : les noms, les verbes, les adjectifs et les adverbes. Notons qu'il existe un type particulier de synset appelé *adjectif satellite (adjective satellite)*.

Par ailleurs, Wordnet définit 17 relations dont : 10 entre synsets (*hyponymy ou spécialisation, entailment ou implication, meronymy ou partie de, etc.*), 5 entre des sens de mots ou WordSense (*see also, antonym, etc.*), la relation *gloss* entre un synset et une phrase et la relation *frame* entre un synset et un verbe. Le schema RDF/OWL de Wordnet comporte donc trois classes : *Synset, WordSense et Word*. Les classes Synset et WordSense ont des sous-classes : *NounSynset, VerbSynset, AdjectiveSynset, AdverbSynset, AdjectiveSatelliteSynset, NounWordSense, VerbWordSense, etc.* Notons que Wordnet contient près de 155 327 mots (Words), 117 597 Synsets et 207 016 WordSenses. La figure Fig. 2.4 illustre un extrait du schéma RDF de Wordnet.

Il existe une version complète de Wordnet mais également une version de base qui permet uniquement l'exploitation des synsets. Les fichiers pour la représentation RDF/OWL de Wordnet peuvent être téléchargés à l'adresse suivante `http://www.w3.org/2006/03/wn/wn20/`. Ces fichiers contiennent la version prolog de Wordnet ainsi que le programme prolog *« convertwn20.pl »* qui permet de convertir les fichiers prolog de Wordnet en fichiers RDF. Les indications relatives à la procédure de conversion sont disponibles en en-tête

du fichier *« convertwn20.pl »*. Le fichier obtenu après conversion, nommé *wordnet_ synset.rdf*, contient 1 721 882 triplets RDF et 464 783 ressoures ou sujets différents de triplets.

Notons que l'on peut également télécharger une version windows ou unix de Wordnet, sous la forme d'une application pour la recherche de mots (synsets d'un mot et glossaire de chaque synset), sur le site de l'université de Princeton à l'adresse suivante : `http://wordnet.princeton.edu/obtain`.

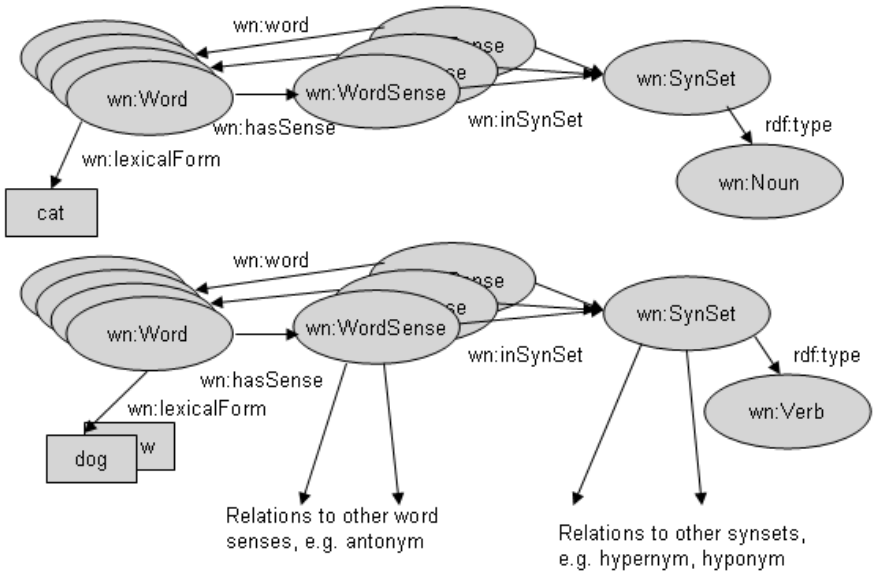

FIGURE 2.4 – Extrait du schéma RDF de WordNet

Dans le contexte de la personnalisation à travers la définition de profils, il existe également des standards pour la description de *profils utilisateurs sémantiques* décrivant des ressources pré-définies.

2.3.3.7 Standards de définition de profils utilisateurs sémantiques

Certains standards pour la description de *profils utilisateurs sémantiques* basés sur leur contexte environnemental ont vu le jour. Ce sont les standards : *Composite Capability/Preference Profiles (CC/PP)* et *Comprehensive Structured Context Profiles (CSCP)*.

Composite Capability/Preference Profiles

Le formalisme (ou standard) *CC/PP*[23] (Composite Capability/Preference Profiles) [KRW+04] permet de décrire des préférences utilisateur. Ces pré-

23. *cf.* http ://www.w3.org/TR/CCPP-struct-vocab/

férences sont considérées comme étant liées uniquement aux capacités de son environnement matériel ou logiciel. Il s'agit généralement des capacités de réception de son dispositif de connexion (téléphone portable, PDA, etc.) qui vont être utilisées pour guider l'adaptation du contenu délivré, aux caractéristiques du dispositif en question.

Le formalisme CC/PP s'appuie sur RDF et permet de définir pour chaque composant hardware ou software, la liste de ses propriétés (par exemple : *name, version, vendor, displayWidth, displayHeight, etc.*). L'inconvénient majeur de CC/PP est qu'il fournit une hiérarchie fixe [SLP04] qui n'offre que deux niveaux structurels de description : *le niveau composant et le niveau propriétés du composant.* Il est donc difficile de décrire des informations contextuelles complexes dont la structure ne respecterait pas celle pré-définie par CC/PP (par exemple, une hiérarchie de propriétés ou de composants). En effet, tous les composants et toutes les propriétés d'un composant sont de même niveau. Le formalisme *Comprehensive Structured Context Profiles* tente de pallier cet handicap.

Comprehensive Structured Context Profiles

Le standard *CSCP* (Comprehensive Structured Context Profiles) [HBS02] est basé également sur RDF et permet la description d'informations sur le contexte environnemental de l'utilisateur. Contrairement à CC/PP, CSCP ne définit pas d'hiérarchie fixe. On peut définir autant de niveaux d'imbrication que l'on souhaite au niveau de la structure. De plus, il permet de définir :

— des ordres de priorité sur des propriétés. Par exemple, on peut définir qu'une propriété est plus importante qu'une autre. Pour cela, on affecte à chaque propriété une valeur de priorité représentée par un nombre réel compris entre 0 et 1 ;

— des conditions liées à certaines propriétés. La syntaxe de la description d'une condition est la suivante : «*opérateurDeComparaison (opérande_1, opérande_2)*». Ainsi, pour décrire la condition qui évalue si la hauteur d'un écran est inférieure à une borne donnée (par exemple : 0,2), on pourra écrire [BHH04] :

$$< (http://...dev : hardware.dev : display.dev : dotPitch.dev : x, 0, 2)$$

où :

l'opérateur de comparaison est $<$;

la première opérande est *http ://...dev : hardware.dev : display.dev : dotPitch.dev : x.* C'est une métadonnée qui décrit la taille d'un écran en DPI (Dot Per Intch ou Point Par Pouce) ;

la seconde opérande est *0,2.*

Si cette condition est vraie, on peut décider que le texte sur cet écran doit être affiché avec une taille bien spécifique.

En résumé, il y a donc un réel besoin d'avoir plus d'informations sémantiques sur les ressources afin de permettre une exploitation optimale de ces dernières. Ceci explique la création de toutes ces normes de structuration et de description de la sémantique de documents décrivant des ressources, ainsi que l'engouement actuel des chercheurs pour le *web sémantique*.

2.4 Présentation du web sémantique

Selon A. Michard [Mic01], *« le web sémantique désigne un système d'information distribué, ouvert, décentralisé, dans lequel les objets du monde réel et les ressources numériques sont décrits à l'aide d'un nombre arbitraire de systèmes descriptifs interopérables, autorisant les références croisées, les recherches transversales et le filtrage »*. Le projet du web sémantique a pour objectif de définir des normes et standards d'échange et de structuration des informations. Le web sémantique s'articule autour de cinq points principaux :

1. la sémantique des données du web ou l'expression du sens ;
2. la représentation des connaissances ;
3. les ontologies ;
4. les agents et les services ;
5. l'évolution des connaissances.

2.4.1 Sémantique des données du web

Le contenu du web d'aujourd'hui est lisible par des humains et par des machines mais il n'est compréhensible que pour les hommes. Le web sémantique vise à ajouter des annotations sémantiques qui soient exploitables par des logiciels (agents). Il s'agit de rendre le contenu du web compréhensible également pour les ordinateurs afin qu'ils puissent les traiter automatiquement et effectuer des inférences pour découvrir ou déduire de nouvelles connaissances : on parle de *sémantique computationnelle* par opposition à la *sémantique cognitive* qui n'est exploitable que par l'homme [CCZC02]. La sémantique computationelle doit donc être sous la forme de bases de données, de programmes, d'indicateurs. A ce jour, le web s'est développé plus rapidement comme médium de documents destinés à des personnes plutôt qu'en tant que données et informations pouvant être traitées de manière automatique. Pour pallier cet handicap, on développe des systèmes d'annotations structurées qui permettraient une exploitation plus approfondie des données sur le web. Ces annotations sont généralement faîtes manuellement par création ou par ré-utilisation de métadonnées existantes à travers des langages de métadonnées comme [FH04] : le Dublin Core, RDFS, OWL, etc. Le web sémantique va utiliser les métadonnées pour donner du sens au contenu des

informations, en créant un environnement où les agents logiciels en parcourant les pages pourront réaliser rapidement des tâches compliquées pour les utilisateurs.

Pour que le web sémantique fonctionne, les ordinateurs doivent avoir accès à des collections structurées d'informations et à des ensembles de règles d'inférences qu'ils peuvent utiliser pour parvenir à un raisonnement automatisé. Les chercheurs en intelligence artificielle ont étudié ces systèmes longtemps avant que le web ne se développe. Ces systèmes sont basés sur la représentation de la connaissance.

2.4.2 Représentation des connaissances

Le challenge du web sémantique est de fournir un langage de représentation de connaissances qui exprime à la fois les données et les règles de raisonnement sur ces données tout en permettant aux règles des systèmes de représentation de connaissances existants d'être exportées sur le web. Deux technologies sont déjà en place :

— *eXtensible Markup Language (XML)* qui donne la possibilité à chaque utilisateur de créer ses propres balises mais ne fournit aucun outil pour exprimer le sens de ces dernières ;

— *Resource Description Framework (RDF)* qui permet de donner un sens aux balises XML.

Avec RDF, tout le monde est autorisé à créer un nouveau concept (décrivant une balise XML qu'il aurait créée) auquel sera associé une définition unique sur le web via l'utilisation d'*Identifiant de Ressource Universelle (URI)* et d'*espaces de noms* (*cf.* sections : 2.3.1 et 2.3.1.2). En effet, le langage humain s'accommode de l'utilisation d'un terme identique pour désigner des choses différentes, mais pas le traitement automatique des données. L'utilisation d'une URI différente pour chaque concept spécifique résout ce problème.

Par ailleurs, deux bases de données peuvent utiliser des identificateurs différents pour ce qui est en fait un même concept. Un programme devant comparer ou combiner les informations de ces deux bases doit savoir que ces deux termes sont utilisés pour dire la même chose. Idéalement, le programme doit avoir une façon de découvrir ces sens communs quelles que soient les bases de données. La solution à ce type de problème est donnée par le troisième composant du web sémantique : *les ontologies*.

2.4.3 Ontologies

Une ontologie est un document ou un fichier qui définit formellement *une taxonomie et un ensemble de règles d'inférences*. La taxonomie définit des classes d'objets (ou concepts) et des relations entre elles. Les règles d'inférences, quant à elles, apportent une information supplémentaire dans le but de permettre aux programmes de faire des déductions. Les ordinateurs

ne comprennent pas vraiment ces informations (règles d'inférences) mais peuvent les manipuler plus efficacement, d'une façon utile et compréhensible par l'homme. Avec les ontologies, les solutions aux problèmes d'hétérogénéité de terminologies, thésaurus, dictionnaires, lexiques et autres commencent à être trouvées. L'ontologie va permettre d'établir des ponts entre vocabulaires hétérogènes.

1. *ontologie et terminologie* : d'après Felber [Fel84], la terminologie est un domaine du savoir interdisciplinaire et transdisciplinaire ayant trait aux notions et à leurs représentations (termes, symboles, etc.). La terminologie vise à faciliter l'accès aux vocabulaires des langues de spécialité en classant les divers concepts, en les définissant et en les structurant d'une manière logique. Une terminologie peut être vue comme une classification de concepts, tandis que les ontologies vont avoir un but plus général : décrire les entités et leurs propriétés, les relations de contraintes entre entités, les comportements ;
2. *ontologie et thésaurus* : un thésaurus est un vocabulaire formellement organisé et dont les relations entre les concepts sont à priori faites de façon explicite [ISO 2788, 1986 : 2]. Un thésaurus peut-être utilisé comme langage d'indexation contrôlé. L'ontologie va être plus formelle qu'un thésaurus avec des liens plus riches (et pas toujours explicites) entre les concepts en exploitant par exemple des règles de symétrie, de transitivité, de généralisation (héritage), etc. ;
3. *ontologie et dictionnaire ou lexique* : Les dictionnaires ou lexiques sont des vocabulaires décrits en langage naturel tandis que l'ontologie va permettre d'avoir un langage formel (utilisation d'expressions logiques construites avec des primitives et des connecteurs, relations de composition et d'héritage au sens objet, etc.). Ainsi, la signification de termes ou de codes XML utilisés sur une page web peut être définie par un pointeur de la page vers une ontologie. Les problèmes de synonymie peuvent être gérés en définissant des relations d'équivalence ou d'égalité.

L'ontologie s'exprime dans un langage et repose sur une théorie (sémantique) garante des propriétés de l'ontologie en termes de consensus, cohérence, réutilisation et partage. Les ontologies sont un des moyens envisagés aujourd'hui pour ajouter de la sémantique à des modèles divers. Par exemple, avec les index, on peut définir des relations formelles entre termes (synonymie, antinomie, etc.) et entre termes et granules documentaires (fréquence d'apparition, méta-information, etc.).

Notons que la partie taxonomie des ontologies est construite entièrement de façon manuelle à l'heure actuelle et que la partie règles d'inférences est très souvent occultée.

On va distinguer différents types d'ontologies [SBF98] :

1. les *ontologies de domaine* de type terminologie et thésaurus qui décrivent le vocabulaire d'un domaine spécifique (médical, mécanique, etc.) [RSB98] ;
2. les *ontologies de représentation* qui ne décrivent aucun domaine particulier mais plutôt un ensemble de primitives pour décrire des ontologies. On peut citer par exemple : PROTÉGÉ[24] [NSD+01] qui est un éditeur d'ontologies et de bases de connaissances ; et ONTOLINGUA[25] [FFR96] qui est un environnement collaboratif distribué pour créer, éditer, modifier, parcourir et utiliser des d'ontologies ; etc. ;
3. les *ontologies génériques ou de haut niveau* [GGM+02] qui peuvent être instanciées ou ré-utilisées car décrivant des connaissances valides pour plusieurs domaines ;
4. les *ontologies de tâches ou de méthodes* qui modélisent une tâche spécifique comme : la gestion de mails [YTA04], la conception d'un projet de recherche [PBH+03], le traitement de commandes clients, etc. Ici, le rôle joué par chaque concept dans une méthode particulière est rendu explicite ;
5. les *ontologies d'applications* qui peuvent être vues comme une double spécialisation d'une ontologie de domaine et d'une ontologie de tâche [YTA04] [PDAB05]. Elles modélisent les connaissances nécessaires pour un domaine d'application.

Notons cependant que les ontologies existantes doivent être transformées au format RDF/OWL pour pouvoir être exploitées par des applications du web sémantique.

2.4.4 Agents et services

Les agents sont des programmes spécialisés dans une tâche précise [Fer95]. Leur intérêt réside dans leur capacité à remplir et à automatiser des tâches à la place des utilisateurs. Ils suivent à la lettre la définition du terme agent : « entité agissant pour le compte de quelqu'un ». Les agents du web sémantique eux sont des programmes qui devront être capables d'exploiter la structure et la sémantique des documents du web sémantique. Une facette importante des agents sera l'échange de *preuves* écrites dans le langage d'unification du web sémantique. Ce langage exprime les inférences logiques faites en utilisant des règles et des informations spécifiées par des ontologies. Le web sémantique devra disposer de moteurs de règles capables de faire des déductions. Il sera également important de disposer de dispositifs de signature numérique afin de permettre aux ordinateurs et aux agents de vérifier que l'information a été fournie par une source sûre. Pour déterminer quel agent

24. *cf.* `http://protege.stanford.edu/`
25. *cf.* `http://www.ksl.stanford.edu/software/ontolingua/`

est spécialisé dans une tâche précise ou dans la découverte d'un type de service donné, on s'appuiera sur une architecture de base [Pri04] et on utilisera un langage commun de communication et d'échange [Ved05] [Mit03]. Par la suite, les services et les agents pourront publier leur fonction en déposant une description de cette dernière dans un répertoire analogue aux *pages jaunes*. Les agents pourront ainsi communiquer en échangeant des ontologies et ils pourront également amorcer de nouveaux raisonnements en découvrant de nouvelles ontologies. Ces agents du web sémantique sont dits *intelligents*.

2.4.5 Évolution des connaissances

Si le web sémantique est bien conçu, il pourra assister l'évolution de la connaissance humaine en favorisant la communication entre différents groupes à travers l'utilisation d'un même langage (les ontologies). Le web sémantique facilitera cette évolution via le nommage des concepts par une URI ou Universal Resource Identifier, et en permettant d'exprimer de nouveaux concepts avec un minimum d'effort.

Le travail qu'effectue le W3C sur le web sémantique est important car il permettra de définir les concepts sur lesquels le web sémantique sera architecturé. La figure FIG. 2.5 présente l'architecture en couches et en *« cours... »* du web sémantique [BL01b] sur laquelle sont basés les différents aspects du web sémantique précédemment présentés.

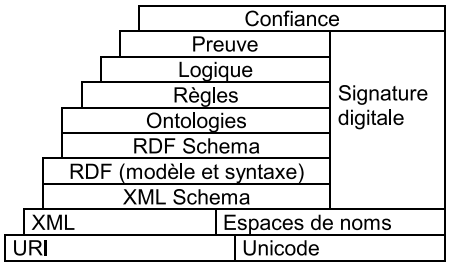

FIGURE 2.5 – Architecture en couches et en *« cours... »* du web sémantique

Les normes ou standards des couches de base de l'architecture du web sémantique sont : URI et unicode ; espaces de noms ; XML ; XML schéma ; RDF modèle et syntaxe ; RDF schéma ; ontologies au format RDF/OWL.

Ces couches de base sont relatives à la description de l'adressage, de la structuration et de la sémantique des ressources. Elles sont complétées par d'autres couches qui ont été définies pour l'exploitation des ressources. Ce sont des couches relatives à l'accès et à l'analyse des modèles de ressources. Pour cela, elles se basent sur des règles, de la logique, sur la sécurité et l'évaluation de la fiabilité des données. Notons cependant que ces dernières couches sont les moins développées du web sémantique, à l'heure actuelle.

2.4.6 Couches d'exploitation du web sémantique : règles, logique, preuves, signature digitale, confiance

Les règles (couche *« règle »*) doivent permettre de raisonner sur des ontologies et d'en déduire de nouvelles connaissances. Ces règles sont définies à travers des langages d'interrogation de documents RDF basés sur des inférences logiques (*cf.* annexe) [HBEV04]. Ces règles doivent suivre une logique formelle (couche *« logique »*) et l'on devrait pouvoir prouver (couche *« preuve »*) leur validité et éventuellement leur décidabilité.

Par ailleurs, le web sémantique doit garantir un minimum de sécurité (couche *« signature digitale »*) sur des données que l'on voudrait protéger. De plus, de façon générale, on devrait pouvoir estimer la confiance (couche *« confiance »*) que l'on a envers les résultats fournis par une application du web sémantique. Pour cela, on peut utiliser des métadonnées qui renseigneraient sur cette confiance [CBHS05] ou expliqueraient éventuellement comment les résultats ont été obtenus.

Cependant, concevoir des normes ou standards viables prend énormément de temps, et il faudra très certainement autant avant que ces standards soient exploités. Il est donc clair, à la vue de la complexité du travail à mener et de la formalisation à réaliser, que le seul moyen d'espérer travailler avec un web sémantique à court ou moyen terme est de procéder par étapes successives et pallier ainsi de façon artificielle au manque de sémantique du web actuel. Alors, une communauté de volontaires contribuant à *sémantiser* le web actuel pourrait ainsi créer des embryons de web sémantique. On peut ainsi noter l'usage d'autres formalismes pour l'intégration de ressources hétérogènes comme les Topic Maps et la logique de description [OVF05].

2.5 Conclusion

Expliciter davantage la sémantique de données manipulées par les applications permet de décupler les capacités de ces dernières à raisonner et à faire des déductions. Elles gagnent ainsi en autonomie et s'illustrent comme des assistants intelligents pour différentes tâches.

Nous allons utiliser des concepts du web sémantique pour expliciter la sémantique de profils manipulés par les applications d'accès à des ressources en proposant un modèle générique et sémantique de profil permettant de décrire des ressources non pré-définies, dans le chapitre 3. Notre modèle héritera donc, par voie de fait, des propriétés du web sémantique que sont : *l'extensibilité, la flexibilité, la ré-utilisabilité et l'interopérabilité*. Les profils ainsi définis seront exploités pour un appariement flexible de profils différents au travers de méthodes d'analyse et d'appariements de profils que nous définissons dans le chapitre 4. Enfin, dans le chapitre 5, nous décrivons un outil d'aide à la construction, à la visualisation et à l'analyse sémantique de profils que nous avons implémenté afin de valider notre modèle générique. Nous

présentons également les résultats d'évaluation des méthodes d'exploitation de profils proposées (analyse sémantique et appariement de profils).

Conclusion état de l'art : Problématiques visées

La gestion de l'hétérogéneité des ressources et l'amélioration de l'accès à ces dernières imposent de pouvoir faire interopérer des ressources différentes. Pour ce faire, on peut se baser sur une sémantique définie au travers de métadonnées. La sémantique peut donc être vue comme un langage consensuel et non ambiguë puisqu'à chaque métadonnée, il sera associé une définition unique. Les langages de métadonnées peuvent être utilisés pour la description de ressources afin d'en faciliter l'interprétation et l'interopérabilité. Pour cela, ces langages de métadonnées peuvent être utilisés comme langages de description de ressources ou comme langages pivots (ou référentiels communs) entre des modèles de description de ressources (ou profils) décrits différemment.

On peut également se servir d'un modèle générique de profil dont le rôle va être de fournir un cadre homogène de description de ressources à travers un schéma général. Ce schéma qui décrit la structure logique, le contenu et la sémantique d'un profil va servir de référence pour définir des méthodes d'exploitation de profils. Ces méthodes ont pour but de garantir un *accès sémantique et optimal* aux ressources mises à disposition.

Dans l'état de l'art, nous avons présenté différents modèles génériques de description de ressources existants. Ces modèles sont comparés dans le tableau Tab. 2.14. L'inconvénient majeur de ces modèles est qu'ils ne garantissent pas une séparation stricte entre la structure logique, le contenu et la sémantique. Ainsi, les traitements à effectuer sur une partie peuvent être fortement dépendants des autres. Ceci limite la flexibilité de description et d'usage des profils.

Le modèle générique de profil que nous proposons doit garantir la séparation stricte entre la structure logique, le contenu et la sémantique d'un profil. Il doit donc permettre une flexibilité de description et d'utilisation de profils. La flexibilité de description garantit la possibilité de décrire différents types de ressources ainsi que le choix et l'organisation non pré-définis des éléments descriptifs de profils. La flexibilité d'utilisation de profils, quant à elle, permet de détecter automatiquement les appariements possibles entre

	Modèles basés sur des langages de métadonnées [HBS02]	Modèles basés sur une structure logique [CKR04]	Modèles clé-valeur [SAW94]
généricité	oui	oui	oui
classes de ressources	pré-définies	non pré-définies	non pré-définies
structuration	arbre ou graphe	arbre	pas de structure
sémantique	oui	non	non
séparation structure logique et contenu	oui	oui	non
séparation structure logique, contenu et sémantique	non	non	non

TABLE 2.14 – Modèles génériques de description de profils

profils grâce à une analyse de la sémantique de chaque profil. Ainsi, la *partie sémantique* du modèle générique va permettre la description du sens des éléments de structure logique et de contenu. Dans certains systèmes existants, modèles basés sur de la structure logique ou modèles clé-valeur, cette partie est généralement considérée comme implicite.

Dans la section suivante, nous illustrons l'intérêt de la sémantique au travers d'un exemple et nous mettons en exergue les problématiques qui y sont inhérentes et que nous abordons dans nos propositions.

Illustration de l'intérêt de la sémantique et problématiques

La figure FIG. 2.6 illustre deux profils qui décrivent tous deux des caractéristiques de téléphones portables comme suit :

- le premier profil (a), nommé *Téléphone_portable_x*, utilise une hiérarchie pour décrire les caractéristiques de l'écran. Pour cela, il utilise les attributs feuilles *Taille* et *Couleur* de l'attribut père *Ecran*. L'attribut feuille *Taille* est associé à la valeur de contenu *2* (pouces) ;
- le second profil (b), *Téléphone_portable_y*, utilise les attributs feuilles *TailleEcran* et *CouleurEcran* pour décrire l'écran d'un téléphone portable. L'attribut feuille *TailleEcran* est associé à la valeur de contenu *4* (centimètres).

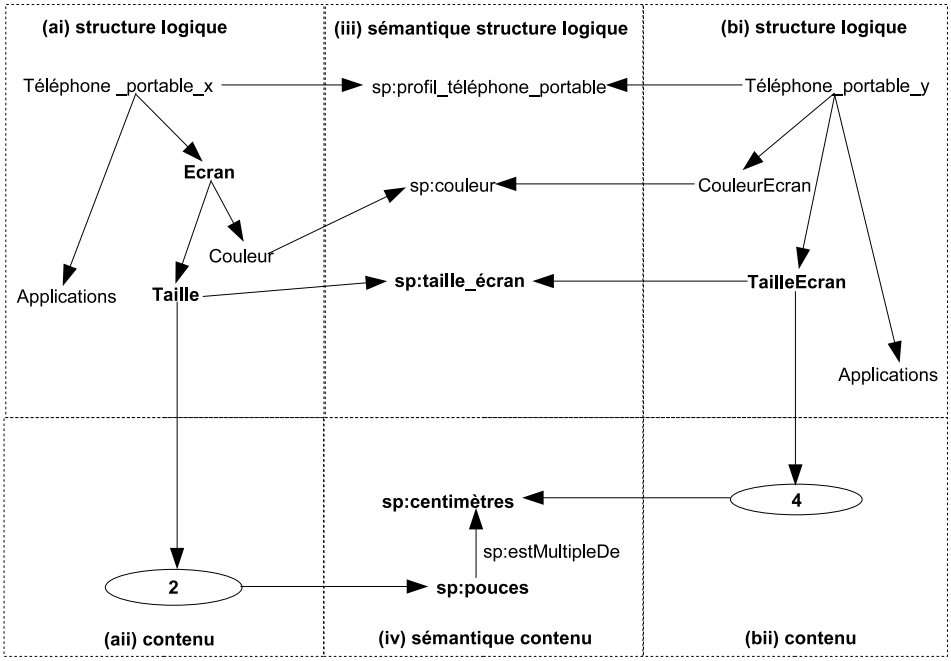

FIGURE 2.6 – Profils et sémantique

On se rend compte à travers cet exemple, qu'il est nécessaire de définir une sémantique non ambiguë non seulement pour préciser le sens des noms d'attributs mais également pour préciser le sens de leur contenu. Les attributs et la structure logique de ces derniers, dans l'exemple précédent, sont différents même s'ils décrivent les mêmes caractéristiques. De plus, les valeurs de contenu qui y sont associées ne sont pas décrits dans le même référentiel car l'un est exprimé en pouces et l'autre en centimètres. La définition d'une sémantique et/ou la ré-utilisation de langages de métadonnées existants devraient permettre de déduire par exemple que les attributs feuilles *Taille* et *TailleEcran*, des profils respectifs *Téléphone_portable_x* et *Téléphone_portable_y*, décrivent la même caractéristique et peuvent donc être appariés.

Ainsi, l'analyse de la sémantique de modèles de profils doit permettre de déduire les éléments de sens compatibles entre ces profils et donc permettre un appariement cohérent entre profils décrits différemment. Nous avons ainsi identifié deux niveaux d'interprétation de la sémantique de profils :

1. *l'analyse sémantique de la structure logique* qui fait usage des concepts associée aux éléments de structure logique ;
2. *l'analyse sémantique du contenu* qui fait intervenir le type de données des valeurs de contenu. Notons qu'au niveau des valeurs de contenu on peut également être confronté aux problèmes de synonymie, d'abré-

viation ou de traduction d'une valeur de contenu dans des langues différentes. De plus, un élément de contenu peut être multivalué comme la liste des applications disponibles sur un téléphone portable. Il peut également être multivalué et pondéré comme la liste des termes d'indexation du contenu des SMS d'un téléphone portable. Par ailleurs, un élément de contenu peut également être décrit par une contrainte (ou condition). Par exemple, si l'on souhaite décrire des usagers par tranche d'âge : junior (moins de 18 ans), âge_intermédiaire (plus des 18 ans et moins de 50 ans), senior (plus des 50 ans), etc.

L'objectif de nos contributions est de proposer des solutions relatives à ces différents aspects. Pour cela, nous avons défini un modèle générique de profil qui intègre une dimension sémantique tout en garantissant la séparation stricte entre la structure logique, le contenu et la sémantique du profil. Il permet de décrire la sémantique de la structure logique et du contenu d'un profil. Nous avons également proposé des méthodes d'analyse sémantique et d'appariement de profils qui sont basés sur le schéma général de notre modèle générique. Ces méthodes ont pour but d'optimiser l'interopérabilté entre profils, issus éventuellement d'applications différentes.

PARTIE 2 : Contributions

Chapitre 3

Modélisation de profils pour l'accès à des ressources

3.1 Introduction

Une ressource est une personne, une chose ou une action nécessaire à l'amélioration ou au bon déroulement d'une activité. L'accès à des ressources permet la combinaison de ressources pour la réalisation d'une tâche spécifique. Cependant, pour pouvoir accéder à des ressources il va falloir disposer d'une description (ou représentation) de ces dernières qui soit exploitable par des programmes. Cette description de ressources, que nous appelons *profil*, fait l'objet du présent chapitre.

Comme nous l'avons vu dans l'état de l'art, il existe une multitude de techniques d'accès à des ressources. Cette hétérogénéité de techniques, s'ajoutant à celle des ressources, complexifie l'exploitation de profils dans les applications : il se pose alors le problème de la gestion de l'interaction entre profils différents ou entre profils décrivant la même classe de ressources mais qui sont décrits par des taxinomies différentes. Afin de pallier cet handicap, on a besoin de modèles de profils qui soient à la fois : extensibles, flexibles, ré-utilisables et interopérables [BLHL01], c'est le cas de modèles de profils sémantiques, dans le but de faciliter l'intégration de ressources hétérogènes.

Nous allons décrire dans un premier temps une architecture générale d'accès à des ressources. L'intérêt de cette architecture est de montrer que l'on a généralement besoin de combiner des profils différents pour la réalisation d'une tâche et que pour cela, on a également besoin d'une description détaillée et rigoureuse de profils. Ensuite, nous proposons un modèle générique de profil qui s'appuie sur les technologies du web sémantique. L'intérêt de la généricité du modèle est de disposer d'une structure de base homogène pour la description ou dérivation de profils décrivant des classes non pré-définies de ressources. L'objectif ici n'est donc pas de *standardiser* le modèle de profil mais de pouvoir instancier différents profils à partir de cette structure. Ce-

pendant au niveau des instances, certaines disparités entre modèles de profils demeurent et sont liées principalement à l'organisation et/ou au nommage différents des éléments de structure logique. La dimension sémantique du modèle va nous permettre de pallier cela, car elle va offrir de la flexibilité au niveau de la description des profils tout en permettant de créer des ponts, sous forme de langages de métadonnées pouvant être partagés, entre modèles différents. Ces ponts définissent un langage consensuel qui a pour but de faciliter les comparaisons de profils.

3.2 Architecture générale d'accès à des ressources

L'accès à des ressources (personnes, documents, etc.) peut être défini, par exemple, pour des applications de partage et d'échange de données incluant généralement des processus de recherche, filtrage ou recommandation. Selon l'application, les ressources exploitées ne sont pas les mêmes. Il est donc intéressant de pouvoir disposer d'un modèle général d'accès qui pourra être instancié pour différents contextes applicatifs.

3.2.1 Modèle général

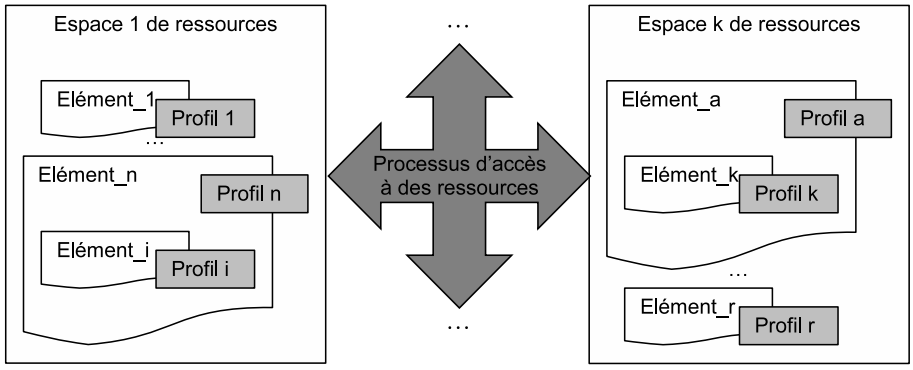

FIGURE 3.1 – Architecture générale à base de profils pour l'accès à des ressources

Le schéma de la figure FIG. 3.1 présente l'architecture générale à base de profils pour l'accès à des ressources que nous proposons. Les processus d'accès pouvant être, par exemple, des processus de recherche, de filtrage ou de recommandation d'information. Une version simplifiée à deux espaces de ressources (espaces des utilisateurs et espace des informations mises à disposition) a été proposée dans [Tch05c], [Tch04b].

Cette architecture résulte de l'analyse de la mise en œuvre de différents systèmes de recherche et de recommandation afin d'en déduire un modèle gé-

néral. Les systèmes existants sont conçus pour atteindre des objectifs particuliers en fonction des spécificités propres à leur contexte : recommandation de pages web en fonction des signets [RP97], filtrage de mails [GNOT92], commerce électronique [CKK02], etc. Contrairement à ces systèmes, notre architecture est assez générale pour servir de modèle à différents types d'applications.

Notre architecture ne s'applique pas à un cadre pré-défini. Elle est constituée d'un ensemble d'éléments qui interviennent dans la mise en œuvre d'un système d'accès à des ressources. C'est à chaque application d'instancier cette architecture générale en définissant les espaces de ressources et les éléments de ces espaces qui l'intéressent. Notre architecture peut être utilisée comme point de départ pour toute construction de systèmes d'accès à des ressources.

Sont mis en évidence, dans cette architecture, les processus d'accès ainsi que la structure générale des éléments manipulés par ces derniers. Ces éléments sont regroupés en différents espaces de ressources qui peuvent être, par exemple : relatifs à *l'espace des utilisateurs*, relatifs à *l'espace des informations mises à disposition*, etc. Notons qu'à chaque type d'élément, on associe un profil qui le décrit de façon détaillée et qui est exploitable par les processus d'accès. De plus, ces éléments peuvent aussi être composés d'un ou de plusieurs sous éléments (granularité des éléments : *cf.* figure FIG. 3.3) eux-mêmes décrits par des profils. Ainsi, notre architecture va donc pouvoir être instanciée dans divers domaines applicatifs.

3.2.2 Instanciation de l'architecture

Les figures FIG. 3.2 et FIG. 3.3 instancient des exemples d'architectures de processus d'accès à des ressources que sont ici *la recherche et la recommandation* dérivées de la figure FIG. 3.1. Elles illustrent les différents types de granularité qui peuvent exister tant au niveau des usagers que des informations mises à dispositions. Cette granularité est traduite schématiquement par la composition ou l'imbrication des éléments de l'architecture. On note également la possibilité d'une juxtaposition d'éléments différents pour un même niveau d'imbrication donné FIG. 3.3. Notre architecture va pouvoir se servir de cette hétérogénéité de structure pour permettre une exploitation maximale de la compléméntarité entre différents profils qui décrivent des éléments de l'architecture.

Les éléments liés à l'espace des utilisateurs sont variés et peuvent être, par exemple :

- *les informations de l'espace de travail des différents utilisateurs* : historique des informations d'usage (requêtes, sites visités, informations jugées, informations transférées, informations sauvegardées, etc.), structure et contenu d'informations diverses (signets, courriers, etc.) ;
- *les informations sur l'environnement de travail des utilisateurs* : environnement logiciel et matériel ;

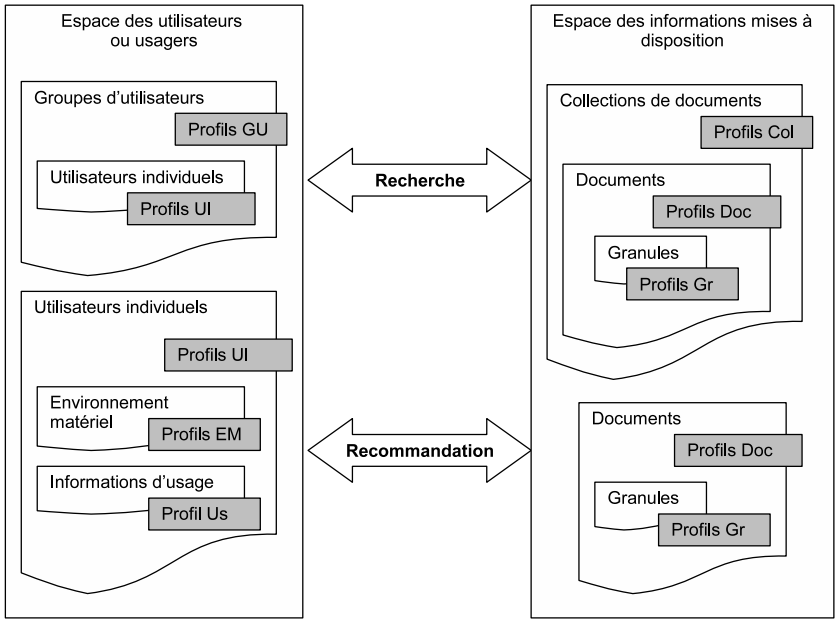

FIGURE 3.2 – Exemple d'architecture à base de profils : illustration de la granularité au niveau des usagers et des informations mises à disposition

— *les informations sur les utilisateurs* : données démographiques, centres d'intérêt, préférences, etc. Ces informations sont obtenues soit manuellement, soit par des méthodes automatiques ou semi-automatiques (*cf.* 1.4.2.3 et 1.4.2.2).

Notons que les éléments de l'espace des utilisateurs peuvent être définis par individu ou par groupe d'individus. Les profils de ces éléments peuvent donc être combinés pour décrire des individus ou groupes d'individus afin de constituer leur profil. Un profil utilisateur peut être : de court terme ou de long terme [WIY99] (*cf.* section 1.4.1.2), positif ou négatif [HKNH00] (*cf.* section 1.4.1.2), etc.

De même, les éléments liés à l'espace des informations mises à disposition sont variés et peuvent être, par exemple :

— *des informations* comme : des documents, des parties ou granules de documents (chapitres, paragraphes, sections, etc.), des collections de documents, des sites ou pages web, des thèses, des articles de journaux, des résumés d'articles scientifiques comme dans la base Medline, etc. Ces informations sont éventuellement annotées par des utilisateurs ou par des experts ou auteurs ;
— *des ressources physiques* comme des serveurs, des périphériques, etc. ;
— *des ressources logicielles.*

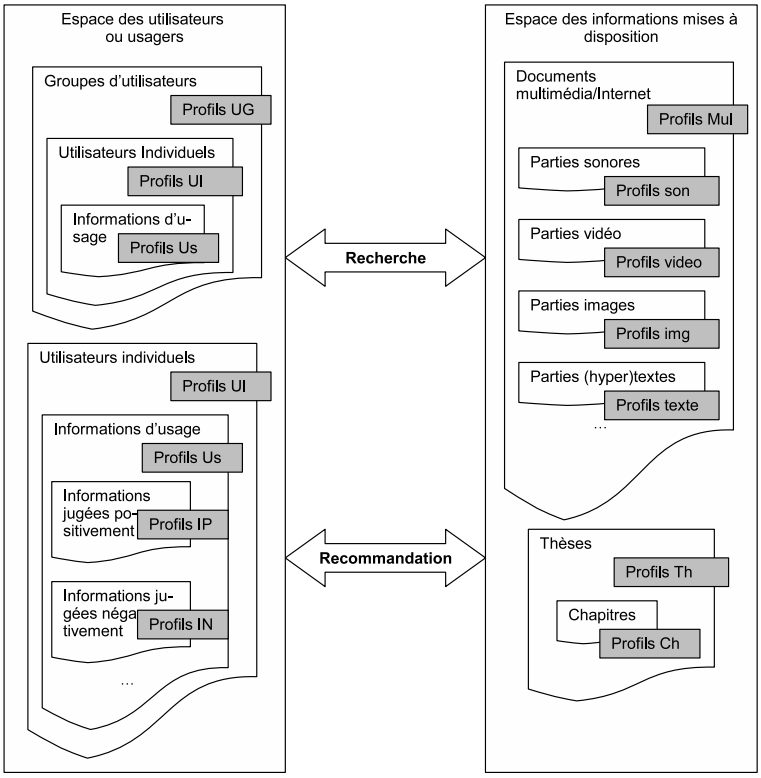

FIGURE 3.3 – Exemple d'architecture à base de profils : illustration de l'hétérogénéité et de la granularité au niveau des éléments de l'architecture

3.2.3 Analyse de l'architecture

L'objectif de l'architecture proposée est de fournir une structure générale pour les applications d'accès à des ressources. L'architecture proposée met en évidence de nombreuses interactions possibles entre différents types de ressources (éléments de l'architecture). Cette coopération se fait par le biais de profils qui sont des descriptions de ces ressources.

De façon générale, un profil va être composé par un ensemble de caractéristiques qui décrivent une ressource donnée : une information (mots clés, langue, taille, date), un usager (thèmes d'intérêt, préférences, données démographiques), un dispositif matériel (mémoire, taille écran), etc. Généralement, ces profils sont construits par des processus d'annotations (de signets, de documents, de fragments de documents), de classification (de signets, de documents, de fragments de documents), d'apprentissage, de création de métadonnées, etc. Par exemple, pour construire un profil utilisateur ou un profil de groupe d'utilisateurs, on peut analyser : les collections d'informations ciblées par ces derniers (signets mémorisés, des résultats validés d'une

recherche antérieure, des pages ou fragments de documents importés), les requêtes formulées par chaque usager ou groupe d'usagers, leurs parcours et habitudes de navigation, etc.

Pour un accès optimal à des ressources, il est nécessaire de décrire avec le plus de détails possibles chaque ressource. Cependant, l'hétérogénéité des applications d'accès et des ressources qu'elles manipulent ainsi que la gestion de leur coopération créent un besoin de modèles de profils extensibles, flexibles, ré-utilisables, inter-opérables. A cet effet, nous présentons, dans la section suivante, un modèle générique de profil qui garantit ces différentes propriétés.

3.3 Modélisation de profil

Le profil d'un objet est l'ensemble des caractéristiques qui permettent de l'identifier ou de le représenter. Les profils utilisés dans les techniques d'accès sont de natures diverses : profil utilisateur, profil de documents, etc. Leur structure peut être composée d'un ou de plusieurs éléments descriptifs (ou attributs) : centres d'intérêts, données démographiques, préférences en langue, mots clés, métadonnées de documents, etc. Cependant, la sémantique des attributs de profils est généralement considérée comme implicite dans les méthodes traditionnelles d'accès à des ressources. Cette sémantique dépend donc fortement de l'application pour laquelle ces profils sont décrits. Ceci a pour conséquence de limiter voire de rendre impossible toute coopération entre profils décrits par des taxinomies différentes (structures et noms des attributs) et/ou issus d'applications différentes. Des problèmes de conflits de sens sur des attributs identiques peuvent donc se poser, de même qu'il sera difficile d'identifier des attributs différents qui décrivent la même caractéristique. De plus, des structures logiques différentes peuvent décrire le même sous-ensemble de caractéristiques. Il y a donc un besoin de modèles génériques [Kob01] et sémantiques de profils [DN03] [BLHL01]. Notre modèle de profil vise à offrir ces différentes propriétés pour améliorer l'accès à des ressources en permettant des interactions flexibles entre profils issus éventuellement d'applications différentes.

Dans la section suivante, nous présentons un modèle générique décrivant la structure logique, le contenu et la sémantique de profils.

3.3.1 Modèle générique de profil

Afin de définir des profils qui soient extensibles, flexibles, ré-utilisables, inter-opérables et multi-facettes, nous proposons un modèle générique de profil qui intègre une dimension sémantique. Il a été conçu pour la description de classes non pré-définies de ressources.

La figure *Fig. 3.4* présente le modèle générique de profil proposé. Il est décrit en utilisant *UML (User Modelling Language)* [MG00] car ce langage

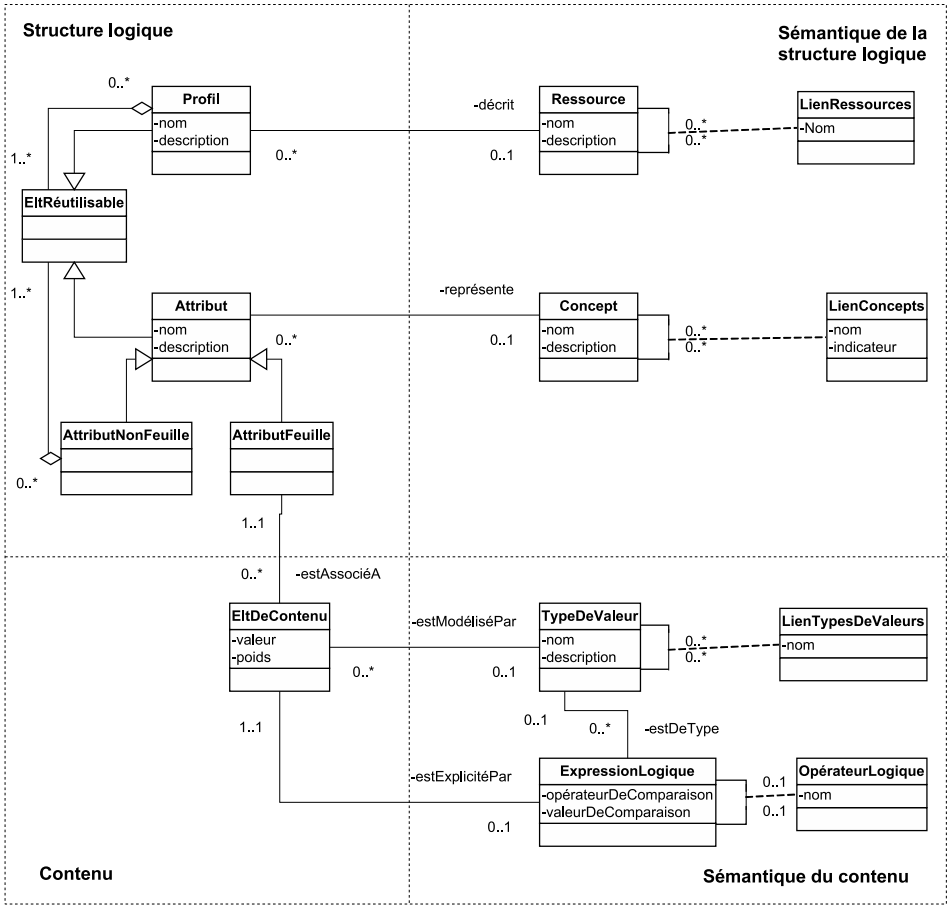

FIGURE 3.4 – Modèle générique de profil [Tch06]

nous fournit un degré d'abstraction intéressant pour décrire des modèles génériques. Notre modèle résulte de l'analyse de différents types de modèles génériques de profils (*cf.* section 2.2). Il permet une flexibilité de description tant au niveau de la structure logique que du contenu grâce à l'usage de la sémantique.

Le modèle générique de la figure *Fig. 3.4* est subdivisé en quatre niveaux : *la structure logique du profil, le contenu du profil, la sémantique de la structure logique du profil et la sémantique du contenu du profil* [Tch05b] [CSDT05b] [Tch06].

3.3.1.1 Structure logique

La structure logique présente la structure générale d'un profil. Cette structure est sous la forme d'une hiérarchie d'éléments réutilisables (instances de la classe *EltRéutilisable*) permettant de caractériser un profil. Cette hiérar-

chie est un arbre dont les nœuds sont soit des profils existants (instances de la classe *Profil*), soit des attributs (instances de la classe *Attribut*) qui décrivent les caractéristiques d'un profil de l'arborescence (ancêtre de type « profil », le plus proche de l'attribut dans l'arborescence). Il existe deux types d'attributs :

1. les attributs pouvant être des nœuds intermédiaires, qui sont des instances de la classe *AttributNonFeuille*, et qui permettent de représenter des catégories ou classes d'attributs (par exemple l'attribut *préférences utilisateurs* peut être composé des attributs *langue*, *taille* et *date*) ;
2. et les attributs qui sont des feuilles de la structure logique (instances de la classe *AttributFeuille*) et auxquels on peut affecter des éléments de contenu (ou valeurs).

Un élément réutilisable peut donc être de type : *Profil*, *AttributNonFeuille* ou *AttributFeuille*. La structure logique d'un profil est donc sous la forme générale d'un arbre.

3.3.1.2 Contenu

Le contenu d'un profil (instances de la classe *EltDeContenu*) est constitué de listes de couples *valeur-poids*. Ces listes peuvent contenir un seul couple valeur-poids (attribut de type monovalué comme la taille d'un document) ou plusieurs couples valeur-poids (attribut de type multivalué comme les mots clés d'un document). La valeur ici est le contenu réel de l'attribut et le poids est une valeur numérique qui décrit à quel point la valeur associée représente l'attribut. Par exemple, si un utilisateur préfère l'anglais au français et le français à l'espagnol, on devrait définir une pondération qui représente cette préférence. La représentation des éléments de contenu est donc basée sur un modèle vectoriel.

Les éléments de la structure logique et du contenu des profils sont organisés de façon hiérarchique en différentes catégories (instances des classes *Profil*, *AttributNonFeuille*, *AttributFeuille*, *EltDeContenu*). Ceci permet de regrouper des éléments similaires dans la même classe et donc de définir une nomenclature (ou taxonomie) de ces différents éléments. Cette taxinomie peut être une manière de définir une partie de la sémantique des éléments de profils.

Cependant, cette organisation structurelle n'est pas suffisante car son interprétation et son exploitation sont fortement liées à l'application qui crée le profil. La coopération (ou interopérabilité) entre profils décrits par différentes taxinomies est donc limitée. Pour améliorer cette interopérabilité, nous ajoutons à notre modèle une dimension sémantique exploitable par différentes applications. Cette sémantique explicite le sens des différents éléments de structure logique et de contenu d'un profil.

3.3.1.3 Sémantique associée à la structure logique

La sémantique de la structure logique de notre modèle générique explicite ce que représente un profil ainsi qu'un attribut de ce profil.

1. La sémantique d'un profil est la description d'une ressource dans un contexte donné. Ainsi, les profils peuvent être relatifs aux utilisateurs (individu ou groupe), aux informations mises à disposition (parties de documents, documents, collections, thèses, etc.), à des dispositifs matériels (téléphones portables, PDAs, etc.), etc. ;
2. La sémantique d'un attribut quant à elle va permettre d'expliciter la caractéristique générique que représente l'attribut (instance de la classe *Concept*). Ainsi, des attributs vont être reliés à des concepts génériques qui sont généralement issus de langages de métadonnées (standards ou normes) existants comme le Dublin Core. Par exemple, l'attribut *mots clés* d'un document pourra être relié à la métadonnée du Dublin Core *dc : subject*.

3.3.1.4 Sémantique associée au contenu

La sémantique du contenu d'un profil permet d'expliciter le modèle de représentation ou type de données (instance de la classe *TypeDeValeur*) des éléments de contenu (*cf.* par exemple, les types de données de XMLSchema). A titre d'exemples, on peut citer les types : entier, chaîne de caractères, date, etc. ; les types dérivés comme les patterns pour décrire : des numéros de sécurité sociale, des codes postaux de régions, des formats de date, etc.

La sémantique de contenu va permettre également de préciser le sens d'un élément de contenu donné au travers d'expressions logiques. Par exemple, on va pouvoir exprimer le fait qu'un usager s'intéresse à des articles publiés après une certaine date. Notons que l'on va également pouvoir combiner plusieurs expressions logiques via les opérateurs logiques : *ET, OU*.

En résumé, la structure logique et le contenu d'un profil sont organisés sous la forme d'arbres hiérarchiques obtenus en appliquant des règles de décomposition sur les instances des classes *Profil* et *AttributNonFeuille*, ou en créant le lien sémantique « estAssociéA » entre une instance de la classe *AttributFeuille* et des instances de la classe *EltDeContenu*. Lorsque l'on ajoute des éléments de sémantique qui sont des instances des classes *Ressource*, *Concept*, *TypeDeValeur* et *ExpressionLogique* éventuellement reliés par des instances des classes d'association *LienRessources*, *LienConcept*, *LienTypesValeurs*, *OpérateurLogique*, alors la structure du profil peut se transformer en graphe.

De façon générale, les profils dérivés du modèle générique peuvent être :
— *extensibles* : on peut y rajouter de nouveaux éléments ;
— *flexibles* : on peut organiser librement la hiérarchie les éléments de profils selon la catégorisation qui sied le mieux au contexte applicatif.

De plus, on peut donner les noms que l'on souhaite aux différents éléments de profils sachant qu'on peut leur rattacher une sémantique plus consensuelle via des langages de métadonnées existants (ontologies, taxinomies, liste de métadonnées), par exemple ;

— *réutilisables et partageables* : un sous-arbre d'un profil peut avoir la structure d'un autre profil existant, provenant éventuellement d'une autre application. Par exemple, un profil utilisateur peut être composé : de ses différents profils d'usage (ou profils court terme) et/ou de profils qui le décrivent dans un contexte environnemental particulier (poste de travail du bureau, poste de travail de la maison, téléphone portable). Un utilisateur selon qu'il se connecte à partir d'un téléphone portable ou de son PC du bureau ou de la maison, par exemple, n'a pas toujours les mêmes centres d'intérêts (thèmes de recherche ou des requêtes), ni les mêmes préferences en termes de qualité de l'information restituée (langue, taille, présentation, etc.) ;

— *multi facettes* : les profils peuvent être analysés sous différents angles (ensemble d'attributs et/ou sous profils). Ainsi, chaque profil ou attribut ou combinaison de profils ou d'attributs de profils peut constituer une facette ou vue de l'utilisateur ;

— *évolutifs* : tous les attributs d'un profil donné ne sont pas forcément renseignés. Les profils (structure logique, contenu et sémantique) peuvent être modifiés et peuvent évoluer dans le temps. Ainsi, un profil peut être partagé et enrichi par différentes applications qui utilisent tout ou partie du profil.

L'intérêt de l'utilisation d'un modèle générique est que la structure de base qu'il propose peut être utilisée par différentes applications afin de définir différentes classes de profils [Tch05a], [CSDT04].

Les différents éléments (classes d'objets, associations et classes d'associations) de notre modèle générique vont être décrits dans les sections suivantes. Le but est de préciser pour chaque élément sa particularité et son rôle dans le modèle.

3.3.2 Description des différentes classes d'objets du modèle générique de profil

Notre modèle générique est composé de *dix classes* dont : *cinq* pour la partie structure logique à savoir les classes *Profil, Attribut, EltRéutilisable, AttributNonFeuille et AttributFeuille* ; *un* pour la partie contenu à savoir la classe *EltDeContenu* ; *deux* pour la partie sémantique de la structure logique à savoir les classes *Ressource et Concept* ; et *deux* pour la partie sémantique de contenu à savoir les classes *TypeDeValeur et ExpressionLogique*.

1. la classe *Profil* permet de décrire un objet identifié dans un contexte donné (un utilisateur x, une information y, etc.). Par exemple : profil de l'utilisateur x lorsqu'il est à la maison, profil de l'utilisateur x

au bureau, profil de l'article y lorsqu'il a été soumis, etc. La classe *profil*, va donc permettre de décrire un même objet dans différents contextes. Par exemple : un usager va pouvoir avoir plusieurs profils différents ; un document (article, programme) va pouvoir être décrit par ses différentes versions.

2. la classe *Attribut* permet de décrire une caractéristique du profil qui est à priori indépendante de l'objet x identifié pour une ressource donnée. C'est une caractéristique qui décrit une classe de ressources mais dans un langage pas nécessairement standardisé. Ainsi, un profil de document pourra être décrit par : sa taille, sa langue, son contenu, ses auteurs, etc. Un profil utilisateur pourra être décrit avec les attributs : centres d'intérêt, langues parlées et/ou écrites, etc. De même, un profil de PC ou de téléphone portable pourra être décrit par les attributs : taille mémoire, taille écran, etc. L'intérêt de la classe attribut est qu'elle va permettre l'exploitation d'une *base d'attributs* que l'on va pouvoir ré-utiliser. De plus, elle donne une totale liberté quant au nommage des instances de cette classe ;
3. la classe *EltRéutilisable* qui permet de préciser que dans un profil on peut ré-utiliser des éléments existants. Notamment on va pouvoir réutiliser des profils ou des attributs (liés ou non à des profils) existants. Par exemple, un profil de thèse peut être décrit par les profils de ses différents chapitres et/ou sections de chapitres ;
4. la classe *AttributNonFeuille* représente les attributs qui décrivent un ensemble d'autres attributs et/ou de profils. Par exemple, l'attribut *image* d'un document multi-média peut être composé des attributs : *taille, position, régions*. De même, l'attribut *centres d'intérêts* peut être décrit par le profil *profil des documents sauvegardés* et les attributs *sport, musique* ;
5. la classe *AttributFeuille* représente un attribut élémentaire ou non-décomposable auquel on peut affecter un contenu. Par exemple, la taille d'un document.

   Cependant, certains attributs selon le contexte peuvent être nœuds intermédiaires ou feuilles. Par exemple, l'attribut *centres d'intérêts* d'un usager peut contenir les mots clés décrivant ses centres d'intérêts, dans ce cas, il s'agit d'un *AttributFeuille*. Il peut également, dans un autre contexte, être composé d'attributs comme *musique* ou *sport* qui eux vont contenir des mots clés décrivant des centres d'intérêts relatifs à la musique ou au sport. Dans ce dernier cas, l'attribut *centres d'intérêts* est un *AttributNonFeuille* ;
6. la classe *EltDeContenu* permet de décrire le contenu effectif d'un profil. Il est décrit par la paire *valeur-poids*. La *valeur* représente le contenu effectif du profil et le *poids* représente une donnée numérique

qui traduit à quel point la valeur *valeur* décrit l'attribut feuille auquel elle est rattachée. Par exemple, si un utilisateur préfère lire les documents en anglais par rapport aux documents en français on va définir une pondération pour traduire cette préférence. Ainsi, les éléments de contenu pour les préférences en langues de cet usager peuvent être : *(anglais, 1)* et *(français, 0.5)*.

Notre modèle de structure logique et de contenu permet donc de hiérarchiser les éléments de description d'un profil afin d'affiner leur description tout en permettant d'avoir une pondération sur les éléments de contenu. Il offre également de la flexibilité car plusieurs attributs peuvent avoir le même nom mais des chemins d'accès différents dans l'arbre ;

7. la classe *Ressource* permet de catégoriser les profils. Ainsi, on peut avoir une classe de profils pour les documents textuels, les documents sonores, les documents vidéo, les images, les usagers individuels, les groupes, etc. Le but de cette classe est également l'aide à la construction de profil par ré-utilisation et éventuellement modification de la description de la catégorie de profil (ou instance de la classe *Ressource*) que l'on souhaite ré-utiliser. A chaque instance de la classe *Ressource*, il va être associé un modèle de profil standard (structure logique et sémantique associée) ré-utilisable par tout profil décrivant ce *type de ressource* ;

8. la classe *Concept* décrit des caractéristiques génériques de ressources issues généralement d'ontologies ou standards existants comme les métadonnées du Dublin Core (*cf.* section 2.3.3.4), de MPEG7 (*cf.* section 2.3.3.5), etc.

   L'intérêt de cette classe est d'augmenter la flexibilité de description de profils dans les applications sans pour autant en compromettre l'interopérabilité. Chaque application pourra décrire ses profils avec le vocabulaire qui lui est propre. Cependant, afin de garantir l'interopérabilité, il faudra rattacher ce vocabulaire à un vocabulaire standard compréhensible par les autres applications. Le but de cette classe est de garder l'identité des applications et leur sémantique implicite tout en créant un *pont d'échange ou langage partagé* avec d'autres applications. Avec notre modèle, il ne s'agit pas de demander au vaste héritage d'applications existantes de se conformer aux nouveaux standards pour plus d'interopérabilité mais plutôt d'étendre leur description initiale en établissant des liens avec ces standards. Ici, les langages de métadonnées (ontologies, standards, normes) sont utilisés comme *un pont de communication ou une sémantique partagée* entre différents langages. Ceci est une solution plus intéressante pour les applications qui ne veulent pas toujours modifier complètement leur système du fait du travail important et du coût que cela représente ;

9. la classe *TypeDeValeur* permet de définir le type ou modèle de représentation d'une valeur. Une instance de la classe *TypeDeValeur* peut être vue comme un type de données de XMLSchema (*cf.* 2.3.2.4) : les types de bases (integer, char, date, etc.) ; les types dérivés de ces types de base (par restriction, liste, union, etc.). L'intérêt de la classe *TypeDeValeur* est qu'elle va permettre de définir une sémantique plus fine sur le contenu des éléments de profils. Par exemple : la longueur d'un document peut-être évaluée en nombre de pages ou en nombre de caractères ou encore en nombre d'octets ; une date peut être représentée selon divers formats (JJMMAAAA, AAAAMMJJ, etc.). Notons qu'il faut connaître ce type d'information pour éviter des erreurs d'analyse ;
10. la classe *ExpressionLogique* va permettre également de préciser la sémantique d'un élément de contenu en définissant des contraintes sur les valeurs de ces éléments de contenu au travers d'expressions logiques. On va pouvoir, par exemple, exprimer le fait qu'une valeur nommée *« moinsRécent »* est inférieure à une borne donnée. Pour cela, cette classe possède deux propriétés : *opérateurDeComparaison* qui représente un opérateur de comparaison quelconque ($=, <, >, \leq, \geq$) ; et *valeurDeComparaison* qui représente la borne de la contrainte. On va pouvoir bien évidemment décrire des expressions logiques plus complexes en utilisant des opérateurs logiques au travers de la classe d'association *OpérateurLogique* présentée dans la section 3.3.3.10 ;

Notons que, de façon générale, les propriétés *nom* et *description* dans les classes où elles apparaissent représentent respectivement :

— un *nom* qui va permettre de nommer une instance de la classe et auquel on va associer une URI permettant une identification unique. L'URI va donc permettre la gestion de conflits de noms ;
— une *description*, généralement textuelle, qui donne des informations plus précises sur l'instance. Cette description est le plus souvent interprétable uniquement par l'humain.

Dans la section suivante, nous allons décrire les différentes associations et classes d'associations entre classes d'objets du modèle générique, au travers d'instances. Nous expliciterons ainsi, les rôles et l'intérêt de chaque association ou classe d'association du modèle pour la modélisation de profils.

3.3.3 Description des différentes associations ou classes d'associations du modèle générique de profil

Notre modèle générique possède *7 types d'associations* différentes et *4 classes d'associations*. Notons qu'au niveau des instances, les classes du modèle générique sont indiquées en gras et italique et l'ensemble des instances de ces classes est précisé avec le prédicat *rdf : type*. Ce prédicat est issu du langage de métadonnées de *RDF* et permet de spécifier qu'une classe est une

instance d'une autre classe. Ces informations sont rajoutées pour faciliter la lecture des exemples.

3.3.3.1 Associations d'agrégation

L'association d'agrégation (généralisation de la composition) entre les classes *Profil et EltRéutilisable* traduit par sa cardinalité le fait qu'un *Profil* est l'agrégation d'au moins une instance de la classe *EltRéutilisable*, c'est-à-dire d'au moins un Profil ou un Attribut (AttributNonFeuille ou AttributFeuille). Par contre, un *EltRéutilisable* peut ne pas décrire de profil existant, dans ce cas il décrit un attribut non feuille.

De même, l'association d'agrégation entre les classes *AttributNonFeuille et EltRéutilisable* traduit par sa cardinalité le fait qu'un *AttributNonFeuille* est l'agrégation d'au moins une instance de la classe *EltRéutilisable*, c'est-à-dire d'au moins un Profil ou un Attribut (AttributNonFeuille ou AttributFeuille). Par contre, un *EltRéutilisable* peut ne pas décrire d'*AttributNonFeuille*, dans ce cas il décrit un profil. Notons qu'un *AttributNonFeuille* peut être constitué d'un seul *EltRéutilisable*. Ceci peut s'expliquer par le besoin de vouloir garder une similitude avec une hiérarchie existante.

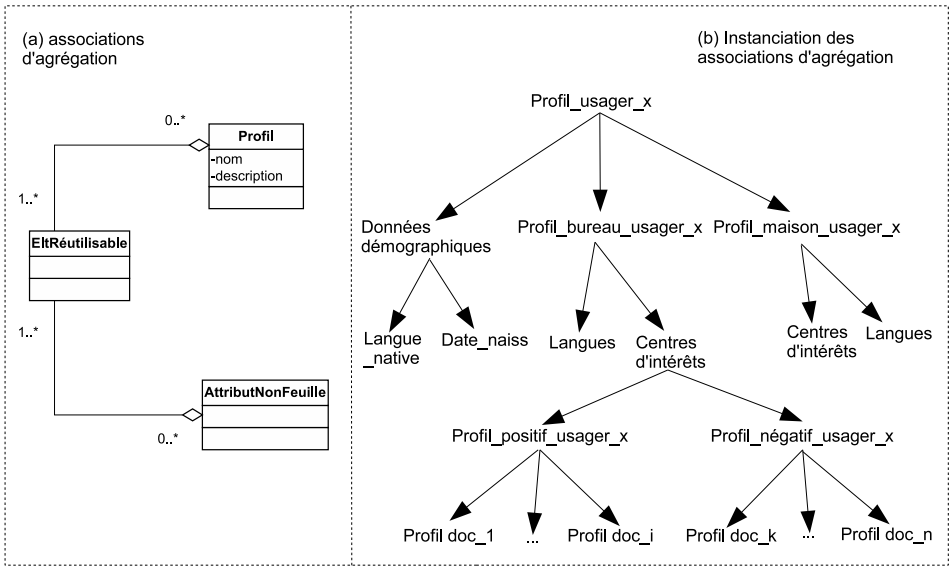

FIGURE 3.5 – Associations d'agrégation du modèle générique et instances

L'intérêt de ces associations d'agrégation est la ré-utilisabilité d'éléments pour la description de profils. La figure *Fig. 3.5* illustre les associations d'agrégation du modèle générique et des instances de ces associations. L'instanciation des associations d'agrégation de cette figure illustre un profil utilisateur composé d'attributs et de profils comme suit :

1. un attribut nommé *Données démographiques* qui est composé des attributs *Langue_ native* et *Date_ naiss* qui décrivent respectivement la langue native et la date de naissance de l'utilisateur. Cette partie illustre la composition d'un *AttributNonFeuille*, représenté ici par *Données démographiques*, par d'autres attributs ;
2. un profil nommé *Profil_ bureau_ usager_ x* composé de deux attributs :
   (a) l'attribut *Langues* qui décrit les préférences en langues de l'utilisateur ;
   (b) l'attribut *Centres d'intérêts* composé :
      i. d'un profil positif qui décrit ce qui intéresse l'utilisateur et qui est composé de tous les profils de documents jugés intéressants par l'usager ;
      ii. d'un profil négatif qui, par contre, décrit ce qui n'intéresse pas l'utilisateur et qui est composé de tous les profils de documents jugés non intéressants par l'usager ;

   Cette partie illustre une composition d'un *AttributNonFeuille* (ici *Centres d'intérêts*) par des instances de la classe *Profil*, puis une composition d'instances de la classe *Profil* par d'autres profils ;
3. un profil nommé *Profil_ maison_ usager_ x* qui lui est composé par les attributs *Centres d'intérêts* et *Langues* qui décrivent respectivement les centres d'intérêts et les préférences en langues de l'usager. Cette partie illustre la composition d'un profil (ici *Profil_ maison_ usager_ x*) par des attributs.

3.3.3.2 Associations d'héritage

Il existe des associations d'héritage entre les classes *Attribut ou Profil* et la classe *EltRéutilisable* et également entre les classes *AttributNonFeuille ou AttributFeuille* et la classe *Attribut*. Elles traduisent, dans notre modèle générique, le fait que (*cf. Fig. 3.6*) :

— les classes *Attribut et Profil* sont des sous-types ou spécialisations de la classe *EltRéutilisable* ;
— et les classes *AttributNonFeuille ou AttributFeuille* sont des sous-types ou spécialisations de la classe *Attribut*.

3.3.3.3 Association *décrit*

L'association *décrit* est définie entre les classes *Profil* et *Ressource*. La cardinalité exprime le fait qu'un *profil* décrit zéro ou une et une seule instance de la classe *Ressource*. Par contre, une *ressource* est décrite par aucun ou plusieurs *profil(s)*. L'intérêt de cette association est la *ré-utilisabilité* de la structure logique et de la sémantique associée, par défaut, à une instance

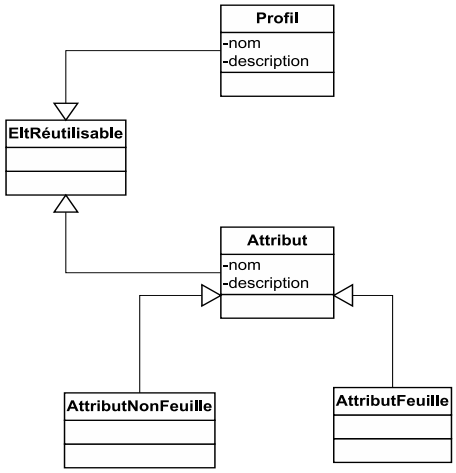

FIGURE 3.6 – Associations d'héritage du modèle générique

donnée de la classe *Ressource* pour l'aide à la création de profils qui décrivent ce type de ressource.

La figure *Fig. 3.7* illustre l'association *décrit* du modèle générique et des instances de cette association. Dans cette instanciation, on remarque que l'on peut combiner au sein d'une même arborescence de profil, des profils qui décrivent différentes ressources.

3.3.3.4 Association *représente*

L'association *représente* est définie entre les classes *Attribut* et *Concept*. La cardinalité exprime le fait qu'un *Attribut* représente zéro ou une et une seule instance de la classe *Concept*. Par contre, un *Concept* est représenté par aucune ou plusieurs instances de la classe *Attribut*. L'intérêt de cette association est la *flexibilité* qu'elle offre au niveau du choix des noms des attributs de profils. On n'est pas contraint par une taxinomie pré-définie de noms d'attributs. Afin de ne pas nuire à l'interopérabilité, on associe les noms d'attributs à des concepts (issus d'ontologies ou standards existants) qui représentent leur sens *générique*.

La figure *Fig. 3.8* illustre l'association *représente* du modèle générique et des instances de cette association. Dans cette instanciation, on remarque qu'au sein d'un même profil, différents attributs peuvent représenter le même concept. Par exemple, les attributs *Centres d'intérêts, Cinéma et Sport* représentent tous le concept *dc : subject*, qui est une métadonnée du Dublin Core.

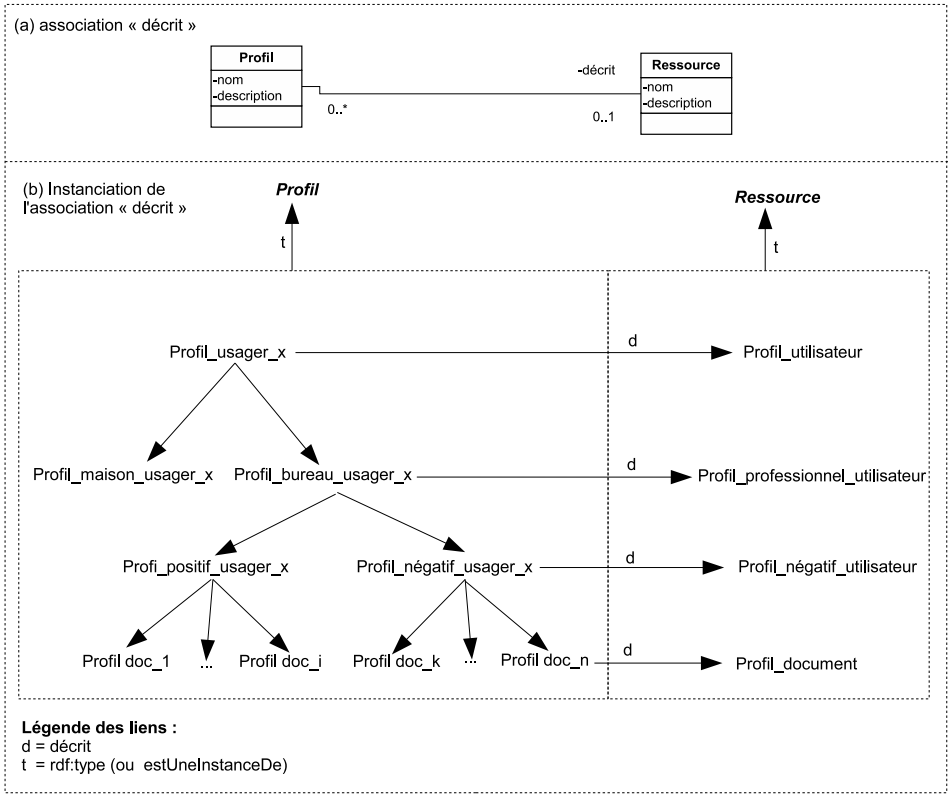

FIGURE 3.7 – Association *décrit* du modèle générique et instances

3.3.3.5 Association *estAssociéA*

L'association *estAssociéA* est définie entre les classes *AttributFeuille* et *EltDeContenu*. La cardinalité exprime le fait qu'un *attribut feuille* est associé à aucune ou plusieurs instances de la classe *EltDeContenu*. Par contre, un *élément de contenu* est lié à un et un seul *attribut feuille*. L'intérêt de cette association est qu'elle permet de dissocier la structure logique et le contenu d'un profil. On peut donc définir une sémantique pour la structure logique et une autre pour le contenu. Cette dissociation va permettre d'analyser plus finement un profil en exploitant à la fois la sémantique de sa structure logique et celle de son contenu (*cf.* chapitre 4).

La figure *Fig. 3.9* illustre l'association *estAssociéA* du modèle générique et des instances de cette association.

3.3.3.6 Associations *estModélisePar*, *estExplicitéPar* et *estDeType*

Les associations *estModélisePar*, *estExplicitéPar* et *estDeType* sont définies comme suit :

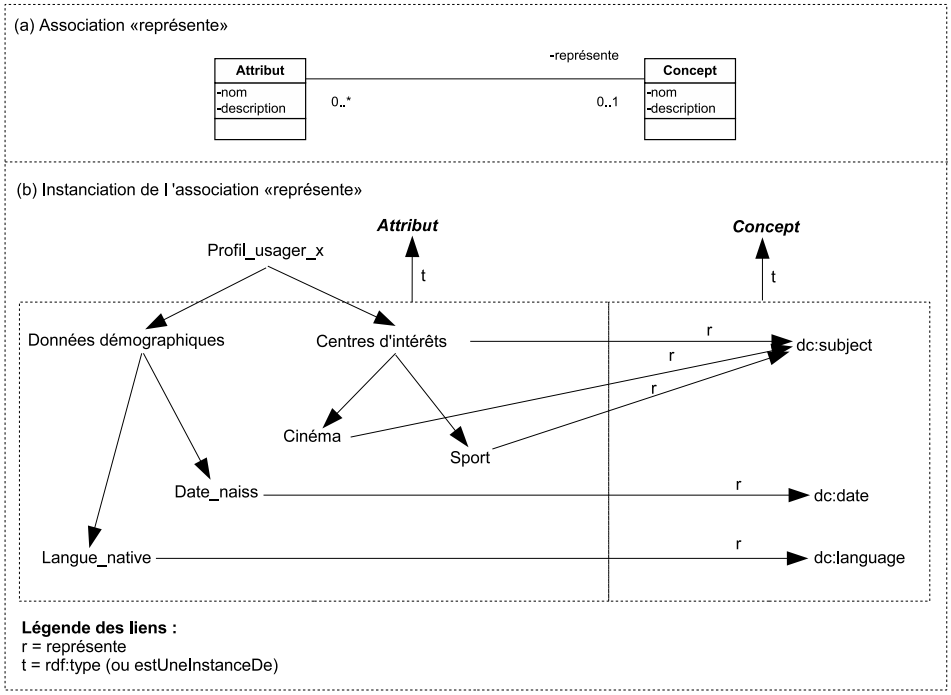

FIGURE 3.8 – Association *représente* du modèle générique et instances

1. l'association *estModéliséPar* lie les classes *EltDeContenu* et *TypeDeValeur*. La cardinalité de cette association traduit le fait qu'un *élément de contenu* (précisément la *valeur* de l'élément de contenu) est modélisé par zéro ou une et une seule instance de la classe *TypeDeValeur*. Par contre, une instance de la classe *TypeDeValeur* est le modèle de représentation d'aucune ou de plusieurs instances de la classe *EltDeContenu*. L'intérêt de cette association est qu'elle permettra de faire des *vérifications de compatibilité de types* entre différents éléments de contenu afin d'éviter certaines opérations erronées (comparaison de dates représentées dans des formats différents) ;
2. l'association *estExplicitéPar* lie les classes *EltDeContenu* et *ExpressionLogique*. La cardinalité de cette association exprime le fait qu'une instance de la classe *EltDeContenu* est explicitée ou clarifiée au travers d'aucune ou d'une et une seule instance de la classe *ExpressionLogique*. Par contre, une *expression logique* est utilisée pour expliciter le sens d'un et un seul *élément de contenu*. L'intérêt de cette association est qu'elle va permettre de préciser le sens de certains éléments de contenu. Par exemple, l'élément de contenu *(récent, 1)* peut être clarifié avec des expressions logiques différentes selon les préférences des usagers. Ainsi, pour un utilisateur donné, un document récent

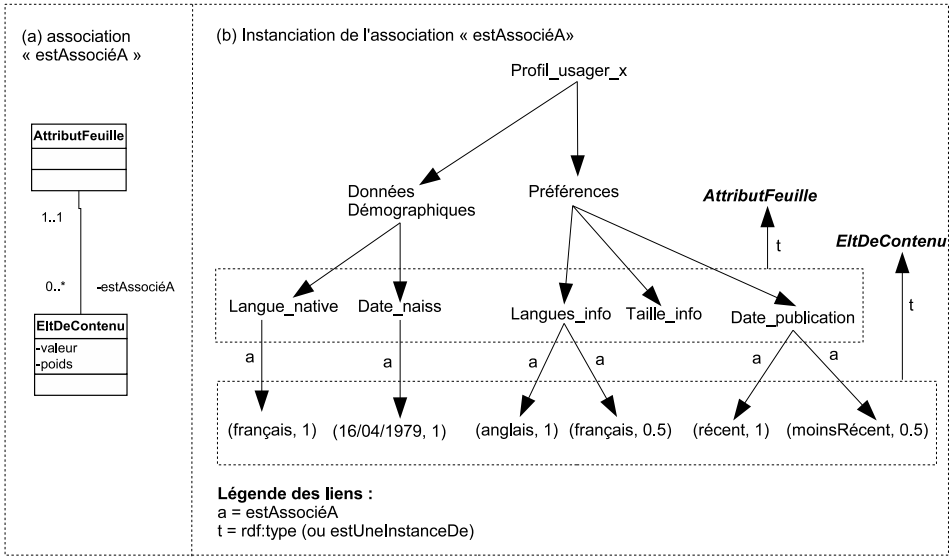

FIGURE 3.9 – Association *estAssociéA* du modèle générique et instances

peut correspondre à un document publié il y a moins de 5 ans et pour un autre, cela peut correspondre à un document publié dans les 12 derniers mois. Il faut être en mesure d'analyser et de prendre en compte ces différentes interprétations. L'analyse de la sémantique de contenu (précisément celle concernant la compatibilité de types) sera optimisée à travers ces expressions logiques car elles vont permettre de déduire un certain nombre de traitements spécifiques (*cf. section 4.3.4*) pour l'appariement d'attributs feuilles ;

3. l'association *estDeType* lie les classes *ExpressionLogique* et *TypeDeValeur*. La cardinalité de cette association exprime le fait qu'une instance de la classe *ExpressionLogique* est définie par une *valeur de comparaison* qui est du type d'aucune ou d'une et une seule instance de la classe *TypeDeValeur*. Par contre, une instance de la classe *TypeDeValeur* définit le type de zéro ou de plusieurs *valeurs de comparaison* d'instances de la classe *ExpressionLogique*. L'intérêt de cette association est qu'elle va permettre de comparer des éléments de contenu associés à des types de données différents mais dont les expressions logiques associés, sont de type identique ou compatible.

La figure *Fig. 3.10* illustre les associations *estModéliséPar*, *estExplicitéPar* et *estDeType* du modèle générique et des instances de ces associations. On remarque que les *éléments de contenu* et les *expressions logiques* ne sont pas associées aux mêmes *types de valeurs*. Par exemple, l'élément de contenu *(récent, 1)* est modélisé par le type *restrictionAnnées* tandis qu'il est explicité par l'expression logique *(>=, 2003)* qui elle est de type *année*. Ainsi,

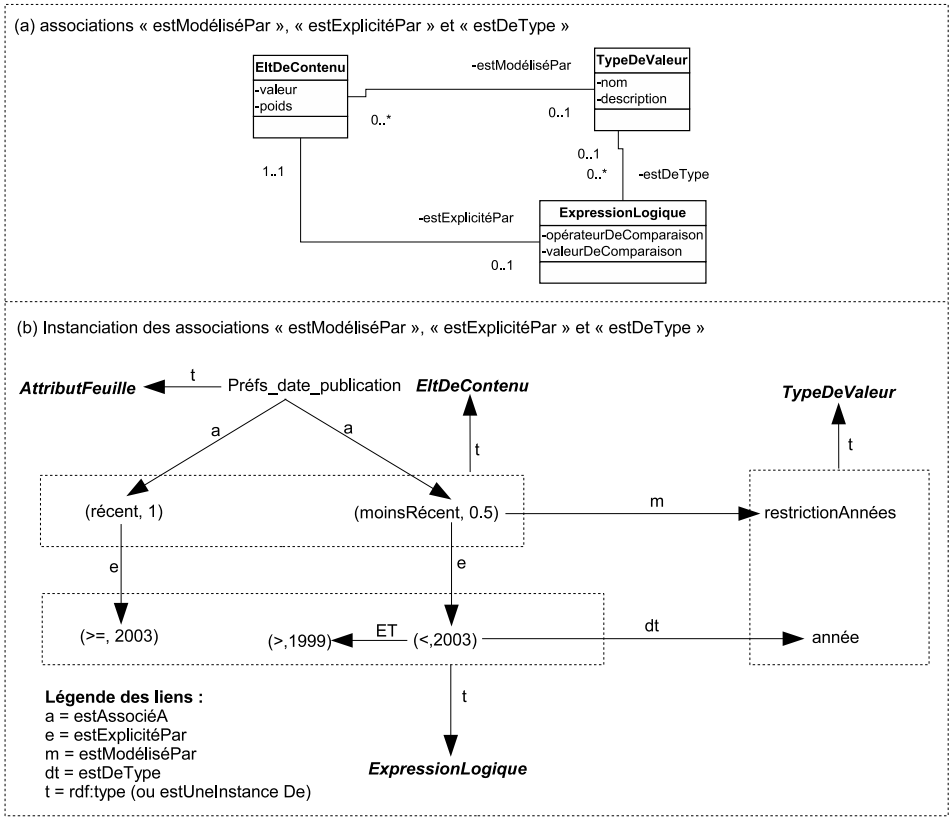

FIGURE 3.10 – Associations *estModéliséPar*, *estExplicitéPar* et *estDeType* du modèle générique et instances

on pourra comparer la date de publication d'un article qui est de type *date* avec les préférences en dates de publication (*Préfs_date_publication*) d'un utilisateur dont le contenu est modélisé par le type *restrictionAnnées* (*cf. Fig. 3.10*) et dont les expressions logiques associées sont de type *année*. Ceci va être possible car on va pouvoir extraire l'année à partir d'une date.

3.3.3.7 Classe d'association *LienRessources*

La classe d'association *LienRessources* est définie réflexivement sur la classe *Ressource*. Elle permet de définir des liens sémantiques entre ressources. C'est l'instanciation de la classe d'association *LienRessources* qui crée ces liens.

Ainsi, sur la figure *Fig. 3.11* qui illustre la classe d'association *LienRessources* du modèle générique, les instances de cette classe association sont : *rdf : type* (noté *t*), *estLaNégationDe* (noté *ng*), *estPlusStableQue* (noté *st*). La figure *Fig. 3.11* illustre des graphes sémantiques de ressources respecti-

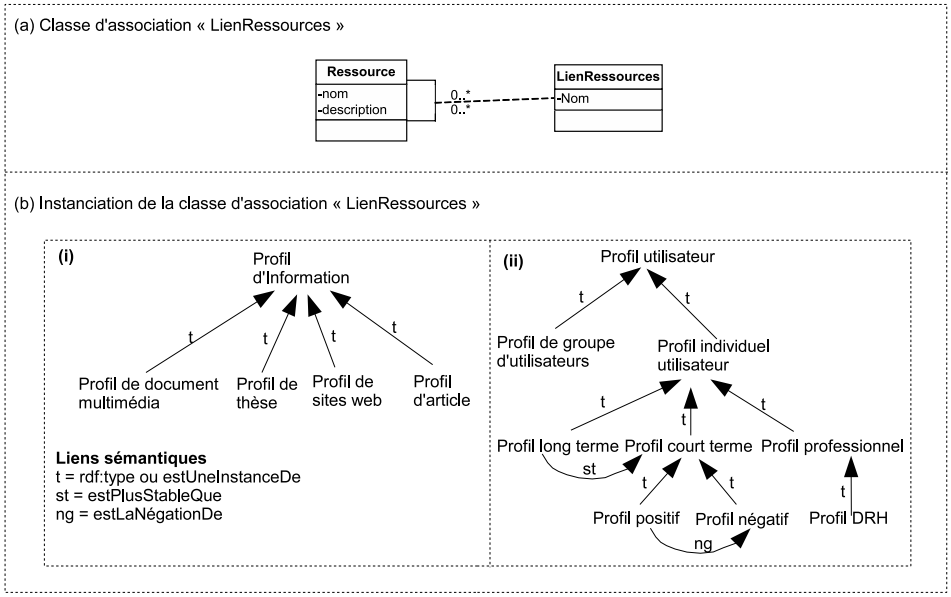

FIGURE 3.11 – Classe d'association *LienRessources* du modèle générique et instances

vement pour une information et pour un utilisateur. On peut exploiter ces graphes pour l'aide à la description de profil en se basant, par exemple, sur la description des ressources similaires à celles que l'on veut décrire.

3.3.3.8 Classe d'association *LienConcepts*

La classe d'association *LienConcepts* est définie réflexivement sur la classe *Concept*. Comme précédemment, elle permet de définir des liens sémantiques entre concepts.

La figure *Fig. 3.12* illustre la classe d'association *LienConcepts* du modèle générique et des instances de cette classe d'association. Les instances de cette classe association sont par exemple : *estComposéDe* (noté c), *estUnePartieDe* (noté p), *rdf : type* (noté t), *seDéplaceVers* (noté dv).

L'intérêt de cette classe d'association est qu'elle va permettre de définir des ponts entre différents langages de métadonnées pour définir si des concepts sont identiques (égalité ou équivalence), s'il y a une relation de généralisation/spécialisation entre eux (*rdf : type* par exemple), s'il y a une relation de partie entre eux (*estUnePartieDe*), etc.

De plus, on va pouvoir indiquer si une instance de cette classe d'association est liée à un sous-arbre de la structure logique d'un profil donné. Pour cela, on va renseigner la propriété *indicateur* de l'instance de la classe d'association *LienConcepts* concernée. Ceci peut s'avérer nécessaire lorsque l'utilisateur définit des liens entre concepts qui n'ont de sens que pour l'in-

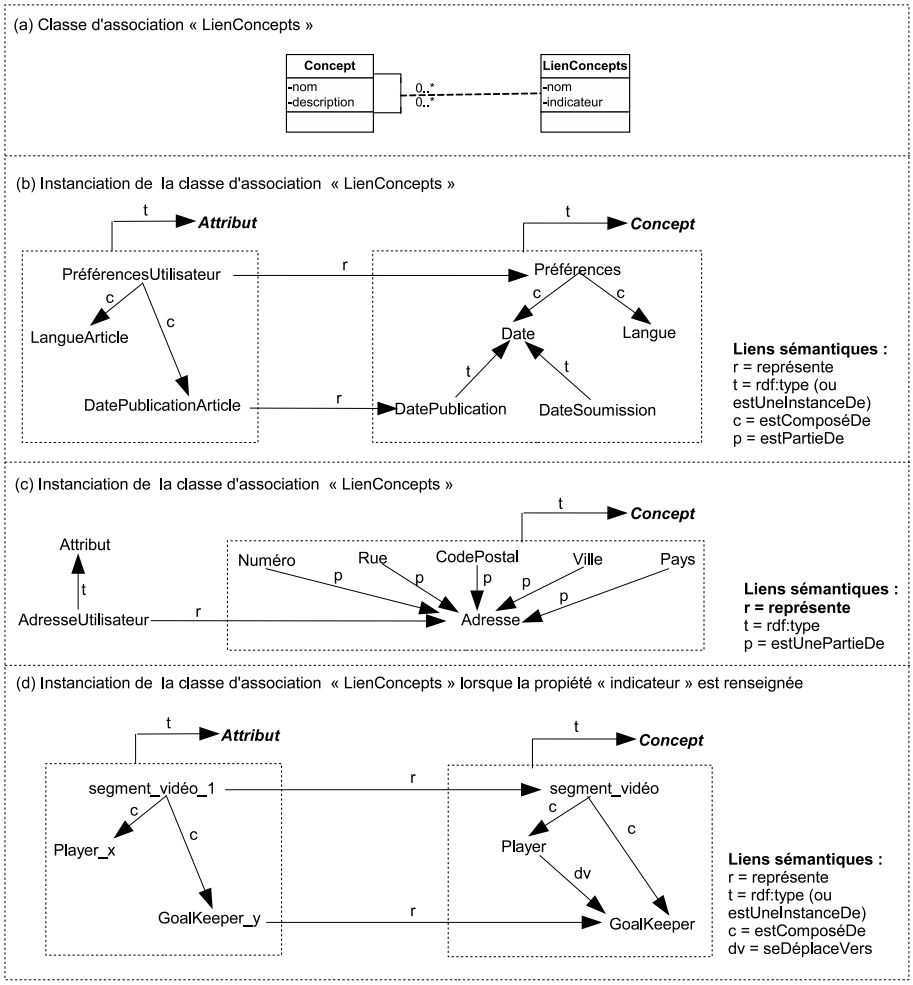

FIGURE 3.12 – Classe d'association *LienConcepts* du modèle générique et instances

terprétation qu'il veut donner à son profil ou à la manipulation de son profil. Par exemple, pour la description de documents multimédia comme dans la figure *Fig. 3.12 (d)*, les instances de la classe *LienConcepts*, à savoir : *estComposéDe* (noté *c*) et *seDéplaceVers* (noté *dv*) doivent avoir la propriété *indicateur* renseignée par l'adresse du nœud *segment_vidéo_1* pour lequel ces liens sont applicables, afin de pouvoir faire le lien avec l'arborescence de ce nœud. Notons que l'arborescence d'un nœud x est définie par l'ensemble de nœuds ayant le nœud x comme ancêtre.

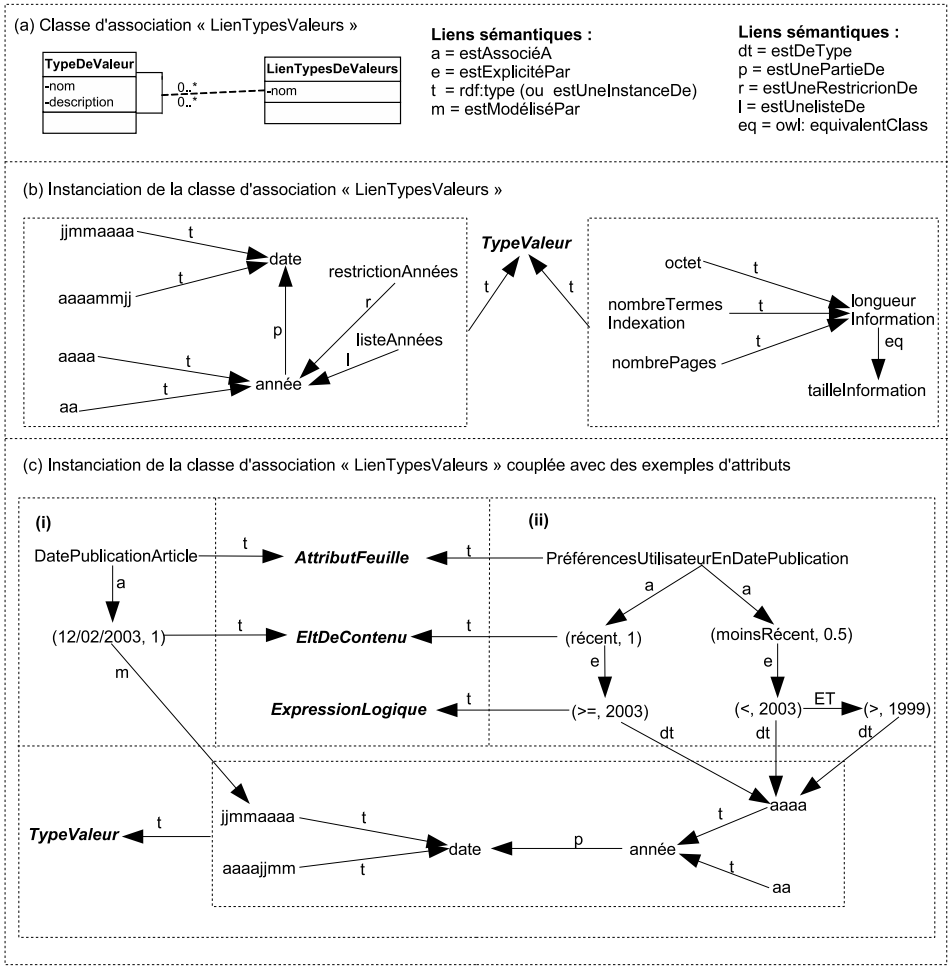

FIGURE 3.13 – Classe d'association *LienTypesDeValeurs* du modèle générique et instances

3.3.3.9 Classe d'association *LienTypesDeValeurs*

La classe d'association *LienTypesDeValeurs* est définie réflexivement sur la classe *TypeDeValeur*. Cette classe d'association permet également de définir des liens sémantiques entre instances de la classe *TypeDeValeur*.

La figure *Fig. 3.13* illustre la classe d'association *LienTypesDeValeurs* du modèle générique et des instances de cette classe d'association. Les instances de cette classe association sont par exemple : *estUnePartieDe* (noté p), *rdf : type* (noté t), etc. Ces liens sémantiques peuvent être exploités pour déterminer des compatibilités entre types de données. Par exemple : le lien *estUnePartieDe* exprime le fait qu'il y a une transformation possible d'un type à un autre par extraction, le lien *rdf : type* ici exprime le fait qu'un type

de donnée est une instance d'un autre type de donnée, etc.

3.3.3.10 Classe d'association *OpérateurLogique*

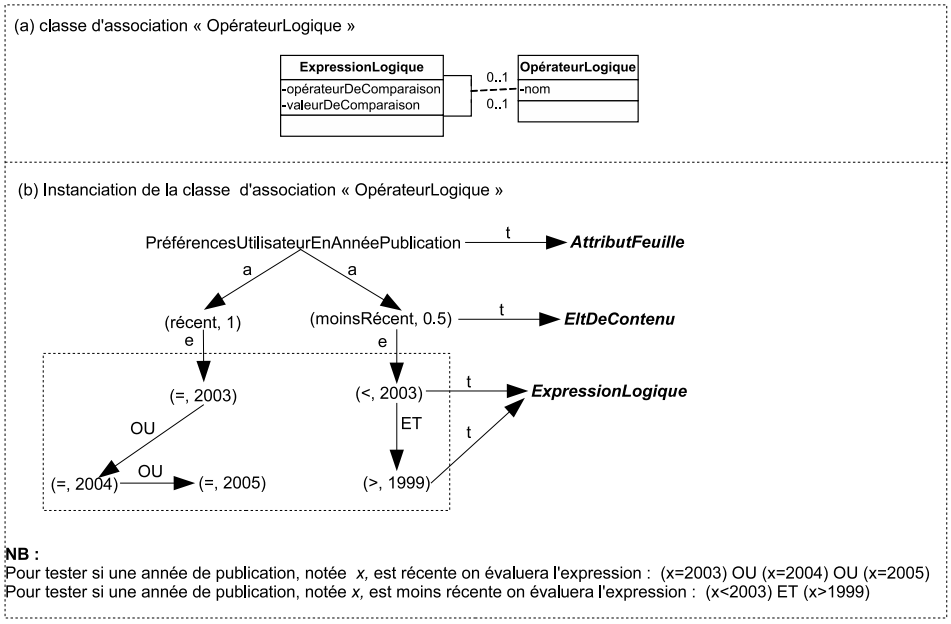

FIGURE 3.14 – Classe d'association *OpérateurLogique* du modèle générique et instances

La classe d'association *OpérateurLogique* est définie réflexivement sur la classe *ExpressionLogique*. Elle va permettre de relier des expressions logiques entre elles par le biais d'opérateurs logiques.

La figure *Fig. 3.14* illustre la classe d'association *OpérateurLogique* du modèle générique et des instances de cette classe d'association. Les instances de cette classe d'association sont les opérateurs logiques : *OU* et *ET*. Ainsi, dans la figure FIG. 3.14 :

- l'élément de contenu *(récent, 1)* est explicité par l'expression logique *(=,2003) OU (=,2004) OU (=,2005)*. Cette expression logique pourra être utilisée pour tester si une année de publication, notée x par exemple, est récente. Pour cela, on évaluera l'expression logique : *(x=2003) OU (x=2004) OU (x=2005)* ;
- l'élément de contenu *(moinsRécent, 0.5)* est explicitée par l'expression logique : *(<,2003) ET (>,1999)*. Dans ce cas, pour tester si une année de publication x est moins récente, on évaluera l'expression logique : *(x<2003) ET (x>1999)*.

Notons que pour être correctement interprétée, une expression logique de notre modèle doit être saisie sous la forme d'une *disjonction de conjonctions*.

Pour ce faire, une expression logique est reliée à une et une seule autre expression logique par un opérateur. L'analyse de la séquence saisie se fait donc de droite vers la gauche. Ainsi, soit la saisie de l'expression suivante :

$(<, 2003)ET(>, 1999)OU(=, 2003)OU(>, 2003)ET(<, 2006)$

Elle correspond, après analyse, à l'expression suivante :

$\{(<, 2003)ET(>, 1999)\}OU(=, 2003)OU\{(>, 2003)ET(<, 2006)\}$

Après avoir présenté les différentes classes, associations et classes d'associations de notre modèle générique au travers d'exemples, nous allons dans la section suivante instancier des modèles de profils en mettant en exergue leur structure logique, leur contenu et la sémantique associée. Ces modèles de profils vont décrire des ressources variées (document textuel, utilisateur, contexte (cognitif, environnemental) d'un utilisateur, fournisseur de service) afin de montrer la généricité de notre modèle et les possibilités de modélisation voire d'exploitation de profils qu'il offre.

3.3.4 Instanciation de profils : structure logique, contenu et sémantique

UML est un langage semi-formel qui nous a permis d'avoir un meilleur rendu visuel de notre modèle générique. Par contre, pour décrire les instances de ce modèle générique, nous avons choisi les formalismes RDF/RDFS/OWL (*cf.* sections 2.3.3.1, 2.3.3.2 et 2.3.3.3). Ces formalismes sont des langages formels, qui sont davantage adaptés pour une description de la sémantique car ils nous fournissent des prédicats de base que nous pouvons ré-utiliser : disjonction (`owl:disjointWith`), équivalence (`owl:equivalentClass`), égalité (`owl:sameClass`), généralisation (`rdfs:subClassOf` et `rdf:type`), etc. De plus, l'usage de RDF/RDFS/OWL nous permet, par la suite, de valider expérimentalement notre modèle en utilisant des interfaces de programmation existantes définies dans le cadre du web sémantique qui permettent d'interpréter et de raisonner sur des représentations RDF/RDFS/OWL. Par ailleurs, notons que UML et RDF ne sont pas des langages disjoints. La brique de base de RDF qui est le triplet *[sujet, prédicat, objet]* existe de façon implicite en UML et dans tous les langages. Ainsi, les associations entre classes et les relations entre une classe et ses propriétés dans un modèle UML donné peuvent être explicitées par des triplets RDF.

Pour le passage de UML à RDF le principe est le suivant :

— pour une association donnée, elle se transforme au niveau de l'instance par le triplet : *[Identifiant_ instance_ classe1, association_ classe1-_ classe2, Identifiant_ instance_ classe2]*. Notons qu'il est associé à chaque identifiant une URI ou une chemin d'accès local à l'application, qui l'identifie de façon unique ;

— pour une classe d'association le principe est le même que pour une association sauf que dans le triplet on parlera plutôt d'instance de classe d'association ;

— pour une classe ayant des propriétés, elle sera décomposée en triplets reliant l'identifiant de l'instance de la classe à ses différentes propriétés sous la forme : *[Identifiant_ instance_ classe1, Propriété_ classe1, Valeur_ Propriété_ classe1]*. Par exemple, soit l'instance suivante *(récent,1)* de la classe *EltDeContenu*, elle sera décomposée par les triplets : *[ID_ récent, valeur, récent], [ID_ récent, poids, 1]*, où ID_récent est l'identifiant de l'instance *(récent,1)*. *valeur et poids* ici sont les propriétés de la classe générique *EltDeContenu*.

Notons que dans les exemples qui vont suivre, nous allons souvent utiliser des métadonnées issues des espaces de noms tels que : Dublin core (noté `dc:`), XMLSchema (noté `xsd:`), RDF (noté `rdf:`), RDFS (noté `rdfs:`), OWL (noté `owl:`) et *SemanticProfile_ NameSpace* qui est l'espace de noms associé à nos propositions (noté `sp:`) et qui est décrit plus en détail dans les sections 4.2.1.1 et 5.2.1.

Dans les sections suivantes, nous allons décrire différentes instances de profils pour montrer la généricité de notre modèle générique.

3.3.4.1 Instanciation de profil utilisateur et de profil d'information textuelle : ré-utilisation de métadonnées existantes

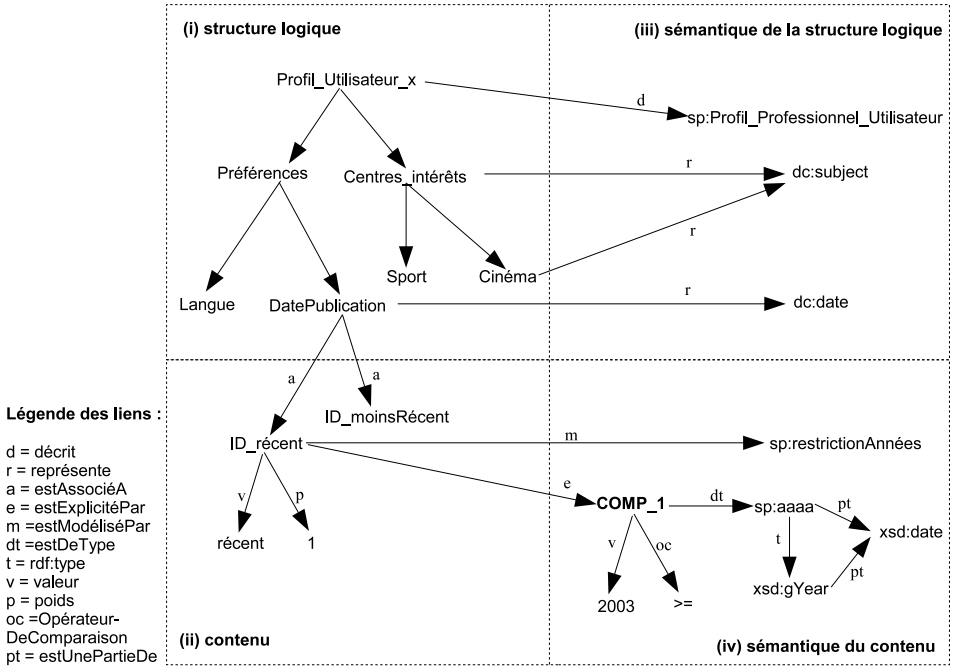

FIGURE 3.15 – Profil utilisateur : structure logique, contenu et sémantique

La *figure 3.15* illustre un profil utilisateur à travers sa structure logique,

son contenu et la sémantique associée qui sont décrits sous la forme d'un graphe RDF. La structure logique et le contenu d'un profil sont toujours décrits par un arbre. La structure de graphe n'est obtenue que lorsque l'on rajoute des éléments de sémantique à la structure logique ou au contenu. Par exemple :

- on peut avoir plusieurs éléments de la structure logique d'un profil qui vont être rattachés au même élément sémantique. De plus, les éléments de sémantique de la structure logique peuvent être également reliés entre eux. Ainsi, dans la *figure 3.15*, nous avons les attributs *Centres_ intérêts* et *Cinéma* qui représentent le même concept générique issu du Dublin Core `dc:subject` ;
- on peut également avoir des liens sous forme de graphe entre éléments de sémantique de contenu. Ainsi dans la *figure 3.15*, on a une instance de la classe générique *ExpressionLogique* nommée *COMP_ 1* qui est de type `sp:aaaa`. Ce type est une instance du type de données de XMLSchema `xsd:gYear` et est, en même temps, une partie du type de données de XMLSchema `xsd:date`. De plus, le type de données `xsd:gYear` est également une partie du type `xsd:date`, ce qui permet d'obtenir un graphe.

Dans cette instance, nous montrons également la ré-utilisabilité de métadonnées existantes dans la description d'un profil. Par exemple :

- la ré-utilisation de métadonnées du *Dublin Core* (préfixées par `dc:`) pour la définition de la sémantique associée aux attributs de la structure logique. Ainsi, les attributs *Centres_ intérêts et Cinéma* représentent le concept `dc:subject`. De même, l'attribut *DatePublication* représente le concept `dc:date` ;
- la ré-utilisation de métadonnées de *XMLSchema* (préfixées par `xsd:`) pour la définition du type des valeurs des éléments de contenu. Par exemple, `xsd:date` et `xsd:gYear`.

La figure FIG. *3.16* illustre un profil d'information textuelle, ici une thèse, à travers sa structure logique, son contenu et la sémantique associée décrits au travers d'un graphe RDF. Dans cet exemple, nous illustrons également :

- la ré-utilisation de métadonnées du *Dublin Core* pour la définition de la sémantique associée aux attributs de la structure logique. Ainsi, les attributs *Contenu, Résumé* et *Chapitre_ 1* représentent le concept *dc : subject*. De même, l'attribut *DateSoutenance* représente le concept *dc : date* ;
- la ré-utilisation de métadonnées de *XMLSchema* pour la définition du type des valeurs des éléments de contenu. Ainsi, l'élément de contenu *ID_ français* associé à l'attribut *Langue* est modélisé avec le type de données *xsd : string* de XMLSchema. De même, l'élément de contenu *ID_ 12/02/03* associé à l'attribut *DateSoutenance* est modélisé par le type *sp :jjmmaaaa* qui est une spécialisation du type *xsd : date* de

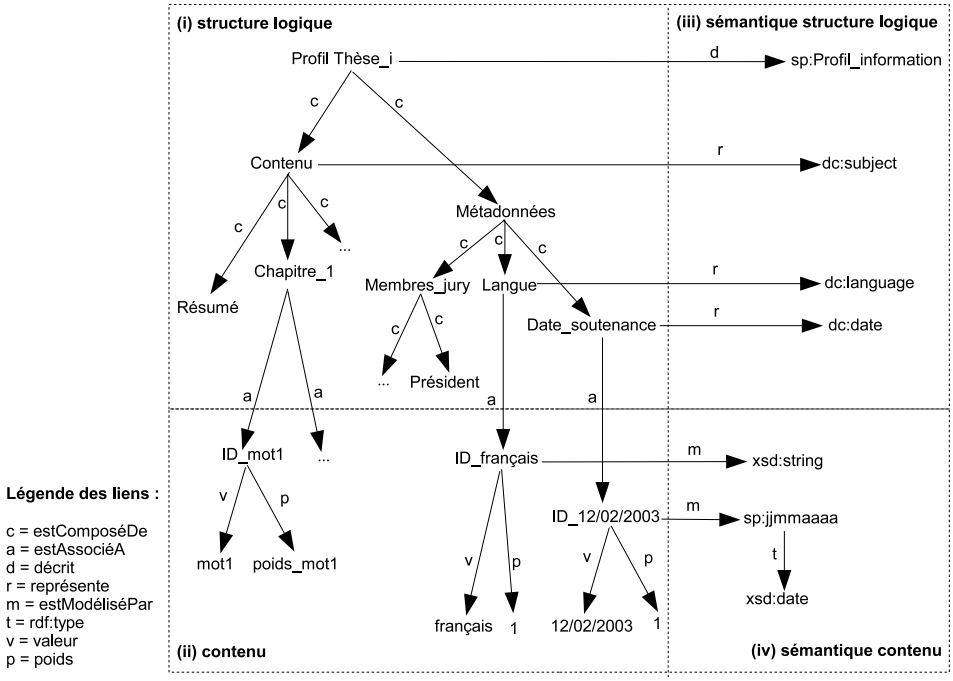

FIGURE 3.16 – Profil d'un document textuel : structure logique, contenu et sémantique

XMLSchema.

L'objectif de ces exemples (*cf.* FIG. 3.15 et 3.16) est de montrer un cas concret de ré-utilisation de métadonnées (ou concepts) existantes par instanciation de notre modèle générique de profil. Cette ré-utilisation de métadonnées permet de faciliter l'interprétation de modèles car elle limite la re-définition de concepts ou types de données et donc limite également la création de liens sémantiques d'équivalence ou d'égalité entre concepts ou entre types de données.

3.3.4.2 Instanciation de profil utilisateur contextuel

La figure *Fig. 3.17* illustre un profil utilisateur lorsque ce dernier se connecte sur un réseau avec un dispositif mobile comme son téléphone portable ou son PDA. Ce profil décrit, dans ce contexte, les centres d'intérêts de l'usager, le profil du dispositif mobile de ce dernier et les préférences de l'usager liées à son dispositif mobile.

L'intérêt de la figure *Fig. 3.17* est aussi qu'elle montre qu'un profil peut être décrit à partir d'autres profils, permettant ainsi la ré-utilisation de profils. Ici, le profil de l'utilisateur est décrit avec le profil de son dispositif mobile. Ainsi, on peut également imaginer décrire le profil du contexte cognitif

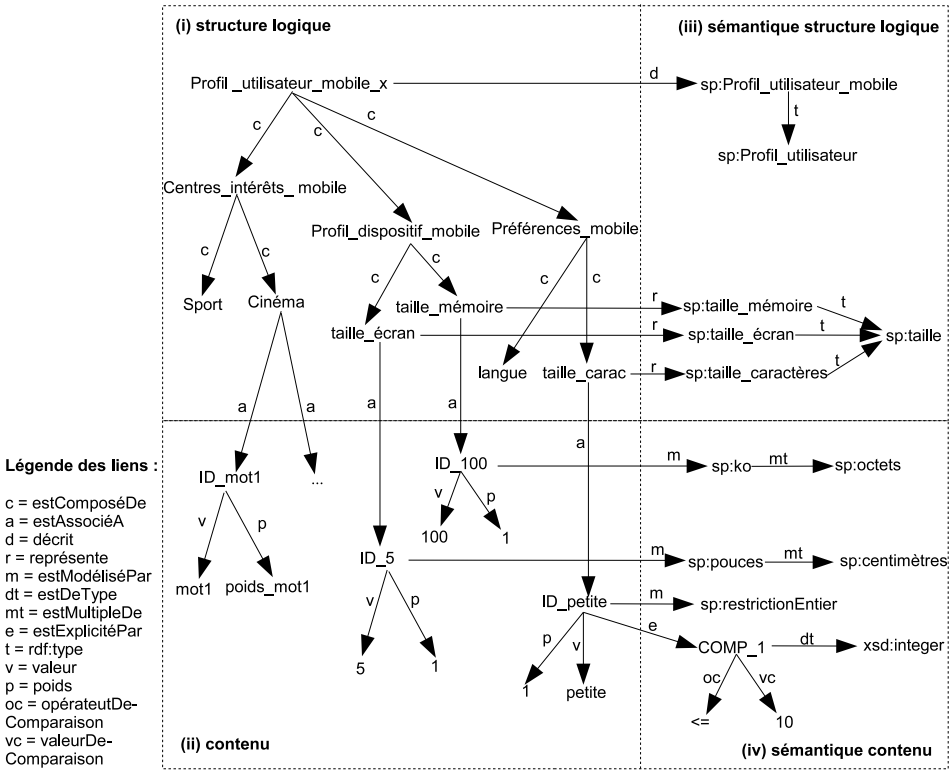

FIGURE 3.17 – Profil contextuel d'un utilisateur : structure logique, contenu et sémantique

(ou contexte des besoins) d'un utilisateur à partir du profil des documents jugés, sauvegardés ou annotés par ce dernier [CSDT05a].

Par ailleurs, ce profil peut être également utilisé pour filtrer des services (services web par exemple) qui sont capables de s'adapter aux préférences de l'usager [LV05] et aux capacités de son dispositif mobile.

3.3.4.3 Instanciation de profil de fournisseur de service

La figure *Fig. 3.18* illustre un profil de fournisseur de service. Avec ce profil, on peut vérifier la taille d'écran ou la catégorie de la taille d'écran d'un dispositif donné qui accède au service de ce fournisseur (comparaison des attributs *taille_ écran* du profil de la figure *Fig. 3.17* et *taille1_ écran* du profil de la figure *Fig. 3.18*) [CSDT06a]. En fonction de ce résultat, le fournisseur de service peut alors développer une stratégie d'adaptation qui consiste à décider de la taille des caractères des documents à envoyer à l'usager. Il peut procéder de la même manière pour déterminer si un dispositif est capable de recevoir des MMS, etc. Le but de cet exemple est de montrer que notre

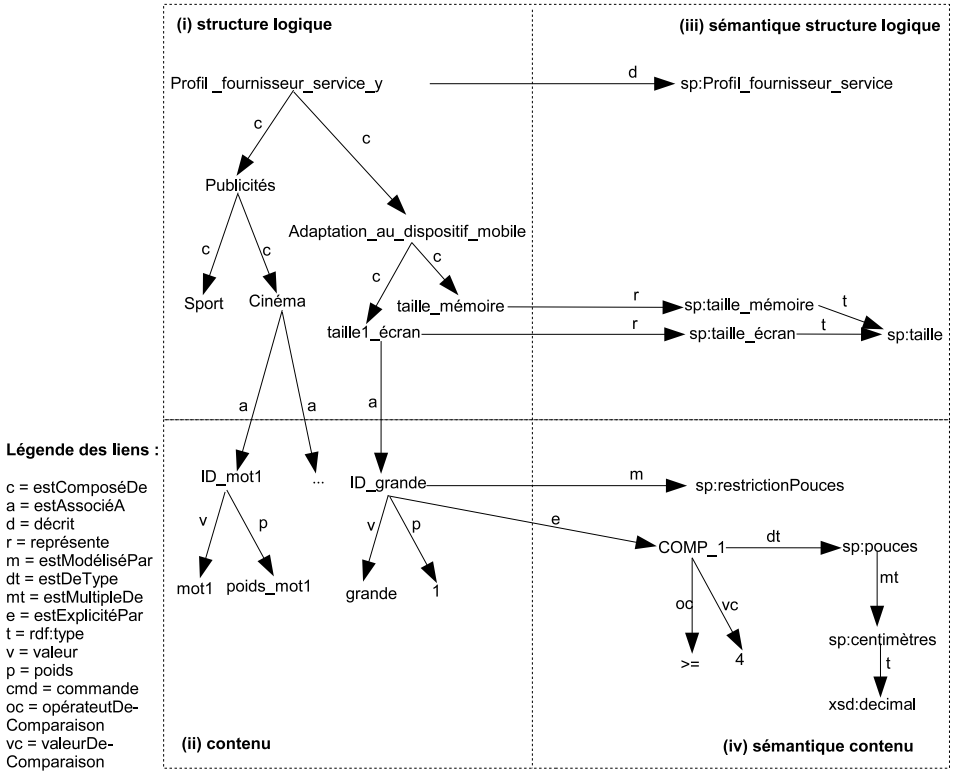

FIGURE 3.18 – Profil de fournisseur de service : structure logique, contenu et sémantique

modèle peut également être utilisé pour la description de services.

Par analogie à la norme CSCP (*cf.* 2.3.3.7), nous pouvons donc définir des contraintes ou conditions sur des valeurs grâce au lien sémantique *estExplicitéPar*. Cependant, dans notre modèle, seulement une partie de la condition (ici comparaison) est décrite, contrairement à la norme CSCP qui décrit complètement la condition à évaluer. Notre modèle définit uniquement l'opérateur de comparaison, ainsi que l'une des opérandes correspondant à la borne de comparaison. L'attribut à comparer (opérande manquante de la condition) est omis et sera complété automatiquement. Pour cela, il faut vérifier la compatibilité de la sémantique associée à l'attribut à comparer avec celle de l'attribut lié à la contrainte (ou condition) définie. L'avantage de notre approche (pour la définition de conditions) est d'éviter de pré-définir la valeur de l'attribut à comparer qu'il faudrait connaître au préalable comme c'est le cas avec le modèle CSCP. Notre modèle permet ainsi de garder une séparation stricte entre structure logique et sémantique ce qui n'est pas le cas du modèle CSCP. D'un point de vue général, il n'est pas toujours évident d'identifier cette partie manquante à l'avance puisque nos profils peuvent

être décrits différemment (il n'y a pas de taxinomie pré-établie). Notons également que notre modèle permet la définition de conditions complexes via une combinaison d'opérateurs de comparaison et d'opérateurs logiques.

Dans la section suivante, nous faisons une analyse de notre modèle générique de profil au travers d'une étude comparative de modèles de description de ressources existants. Nous effectuons également une analyse des formalismes utilisés.

3.3.5 Analyse du modèle générique de profil

Le modèle générique de profil proposé permet de décrire la structure logique, le contenu et la sémantique d'un profil. L'originalité de ce modèle est qu'il permet de décrire différents types de ressources. De plus, notre modèle va permettre également une description du sens des éléments de la structure logique et du contenu d'un profil : *partie sémantique*. Dans les systèmes existants cette partie est généralement considérée comme implicite.

	Semantic-Profile	CC/PP [KRW+04]	CSCP [HBS02]	structure logique et contenu [CKR04]	modèle clé-valeur [SAW94]
généricité	oui	oui	oui	oui	oui
classes de ressources	non pré-définies	contexte environnemental	contexte environnemental	non pré-définies	non pré-définies
structuration	arbre ou graphe	arbre ou graphe	arbre ou graphe	arbre	pas de structure
sémantique	oui	oui	oui	non	non

TABLE 3.1 – Étude comparative de notre modèle de profil nommé *SemanticProfile* par rapport à l'existant

Le tableau TAB. 3.1 compare notre proposition de modèle de profil, nommé *SemanticProfile*, par rapport à l'état de l'art. Ce tableau met en évidence le fait que notre modèle permet d'avoir une structure de profil sous forme d'arbre ou de graphe. De plus, notre modèle n'est pas conçu pour une classe pré-définie de profils. Il garantit également une séparation stricte entre structure logique, contenu et sémantique, ce qui lui confère plus de flexibilité dans la description et dans l'usage de profils.

Par ailleurs, notre modèle permet de combiner différents formalismes de représentation que sont : UML, RDF, RDFS et OWL.

UML versus RDF, RDFS, et OWL

UML dispose d'un ensemble d'éléments de base pour décrire des modèles (classes, associations, cardinalités, etc.) tandis que RDF ne dispose que d'un seul élément de base pour représenter un modèle. Cet élément est le *triplet [sujet, prédicat, objet]*. Cependant, nous remarquons que cette notion de triplet est un élément implicite dans le langage UML et d'ailleurs dans tout langage (langage naturel, etc.). Ces deux formalismes ne sont donc pas disjoints car on peut transformer toute association et toute classe d'un modèle UML en triplet(s) RDF (*cf.* section 3.3.4). Cependant, nous avons choisi le formalisme UML pour représenter notre modèle générique de profil et le formalisme RDF pour les instances. Ceci s'explique par les raisons suivantes :

1. l'élément de base de RDF, qui est le *triplet*, permet de décrire toute relation existante entre deux éléments. RDF a donc un niveau de détail intéressant pour représenter des instances. De plus, étant donné que les analyses de sémantiques se font sur des instances, il est intéressant pour nous d'utiliser les technologies du web sémantique qui est le cadre le plus élaboré et prometteur actuellement pour ce type de traitements ;
2. les éléments de base de UML permettent d'avoir un niveau d'abstraction élevé ce qui correspond bien à l'esprit d'un modèle générique. On aurait pu représenter notre modèle générique en RDF, cependant on n'aurait pas eu un modèle assez compact, facilement lisible et interprétable comme celui obtenu avec UML (*cf.* Fig. 3.4). En RDF, toutes les relations implicites (ou encapsulées) auraient dues être explicitées (par exemple, les liens entre une classe et ses propriétés).

   A tout cela s'ajoute le fait que l'usage d'un langage objet nous permet déjà, au niveau conceptuel, de visualiser le squelette ou encore la structure de l'application qui devra implanter notre modèle. On peut donc déjà intuitivement imaginer des méthodes associées à des classes et des interactions entre différentes classes génériques, ce qui serait plus fastidieux à déceler dans un modèle RDF.

Par ailleurs, pour pouvoir décrire certaines propriétés ou opérations ou contraintes, on va faire appel à d'autres langages de métadonnées : RDFS et OWL. RDFS nous permet de typer les différents éléments d'un triplet avec certains éléments de vocabulaire pré-définis (*cf.* section 2.3.3.2) : *rdf : Resource, rdfs : property, rdf : type, etc.* OWL va permettre de définir principalement des contraintes entre classes pour assurer la cohérence du modèle avec également des éléments de vocabulaire pré-définis (*cf.* section 2.3.3.3) : *owl : disjointWith, owl : sameAs, owl : SymmetricProperty, etc.* Ces éléments de vocabulaire ont un sens pré-défini qui est interprétable par des outils comme *Jena*, une API java pour le web sémantique, qui intègre entre autres : une langage d'interrogation de document RDF, un moteur d'inférence pour faire

des déductions.

UML nous fournit donc une première approche modulaire qui sera combinée avec les spécificités des technologies du *web sémantique* comme RDF, RDFS et OWL, ce qui nous permet de profiter des avantages de chacun de ces formalismes. D'un point de vue visuel et niveau d'abstraction, UML permet un meilleur rendu qu'RDF. Par contre, au niveau exploitation des instances et des relations sémantiques entre éléments de l'instance, RDF/RDFS/OWL nous fournissent des outils plus efficaces dans un cadre éprouvé pour les traitements sémantiques qu'est le *web sémantique*.

3.4 Conclusion

Dans ce chapitre nous avons présenté et justifié notre modèle générique de profil. Ce modèle permet de décrire différentes classes de ressources en explicitant à la fois leur structure logique, leur contenu et la sémantique associée tout en gardant une séparation stricte entre ces différents éléments. La dimension sémantique du modèle permet d'avoir plus de flexibilité dans la description (la structure des profils n'est pas pré-définie) et dans l'exploitation de profils. Ainsi, on va pouvoir faire inter-opérer des profils qui ne suivent pas forcément la même taxonomie.

L'instanciation du modèle générique décrit les profils sous forme de documents RDF ré-utilisant les vocabulaires de : RDFS, OWL, XMLSchema, DC. Ces instances de profils sont donc des *graphes RDF* sur lesquelles on va pouvoir faire des déductions en analysant leur sémantique. Pour cela, nous allons exploiter des propriétés de certains prédicats comme : l'égalité, l'équivalence, la symétrie, la transitivité, la généralisation, etc.

Les profils définis doivent pouvoir être exploités dans des applications d'accès à des ressources. Comme dans le cas de web services, on ne sait pas à priori quels types de profils doivent être appariés. On a donc besoin d'une méthode flexible permettant de faire interagir tout type de profils. Dans le chapitre suivant, nous décrivons une méthode d'exploitation de profils pour l'accès à des ressources qui met en évidence l'interopérabilité de ces derniers.

Chapitre 4

Exploitation de profils pour l'accès à des ressources

4.1 Introduction

L'exploitation de profils dans des applications d'accès à des ressources va consister à les faire inter-opérer en analysant leur structure, leur contenu et la sémantique associée. L'objectif est de détecter les éléments sur lesquels seront basés cette interopérabilité afin de garantir des appariements cohérents entre profils. Pour pouvoir accéder à des ressources particulières, dans le cadre d'une tâche donnée, on est généralement obligé de les comparer afin de sélectionner et éventuellement ordonner celles qui répondent à la tâche définie.

Les informations sémantiques vont servir de *ponts* entre profils différents et permettre ainsi une flexibilité au niveau de l'appariement de profils. Les appariements ne seront donc pas pré-définis mais détectés par analyse des profils à comparer. On va pouvoir comparer des profils décrits dans des langages différents et donc retrouver éventuellement d'autres ressources intéressantes pour une tâche considérée. L'exploitation de la sémantique élargit le champ des ressources à considérer et va, de ce fait, permettre une exploitation plus complète de ces dernières et une coopération maximale entre différentes ressources issues éventuellement de différentes applications. Par contre, l'analyse des informations sémantiques va rajouter une étape supplémentaire à l'opération d'appariement.

Dans ce chapitre, nous allons tout d'abord définir notre environnement général d'exploitation de profils pour l'accès à des ressources. Ensuite, nous allons expliciter le processus d'analyse de la sémantique de profils qui permet d'identifier les éléments appariables qui existent entre des profils que l'on souhaite comparer. Enfin, nous allons décrire le principe d'appariement proprement dit.

4.2 Environnement général d'exploitation de profils

L'environnement général d'exploitation de profils décrit les éléments qui concourent à améliorer l'interopérabilité de profils pour l'accès à des ressources. Cette interopérabilité n'est possible que si l'on peut analyser ou interpréter des profils différents. On pourra ainsi en déduire des éléments de sémantique compatible que l'on va pouvoir comparer. Dans les sections suivantes, nous décrivons notre architecture générale d'exploitation de profils et nous expliquons, à travers des études comparatives, les choix de langages de description et d'analyse des différents éléments de cette architecture.

4.2.1 Architecture générale d'exploitation de profils

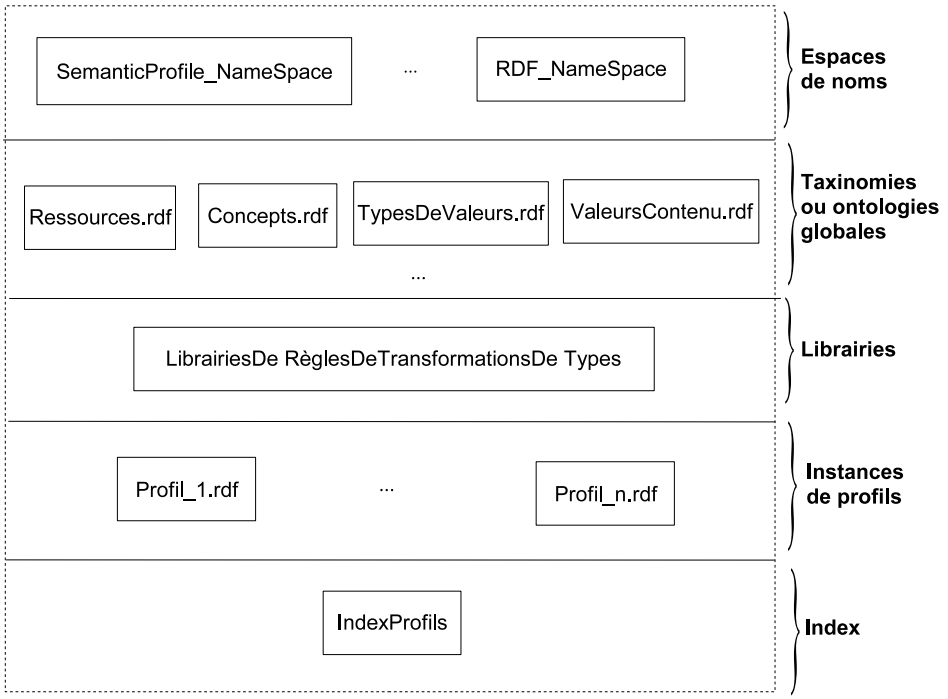

FIGURE 4.1 – Architecture générale d'exploitation de profils

La figure *Fig. 4.1* présente notre architecture générale d'exploitation de profils sur laquelle nous allons nous baser pour analyser et apparier des profils dans des processus d'accès à des ressources. Cette architecture comporte différents types d'informations : *espaces de noms, ontologies globales, librairies de règles de transformations de types de données, instances de profils et index de profils.*

4.2.1.1 Espace de noms

Nous allons utiliser les métadonnées d'espaces de noms pour décrire les différents fichiers RDF de la figure *Fig. 4.1* : instances de profils, ressources, concepts, etc. Les principaux espaces de noms que nous utiliserons sont ceux liés aux langages de métadonnées suivants : Dublin Core (préfixé par `dc:`), RDF (préfixé par `rdf:`), RDFS (préfixé par `rdfs:`), OWL (préfixé par `owl:`), XMLSchema (préfixé par `xsd:`).

Nous avons également défini un espace de noms, nommé *SemanticProfile_NameSpace*, pour prendre en compte principalement les notions relatives à notre modèle générique de profil. Cet espace de noms est préfixé par `sp:` et se subdivise en différentes parties :

1. les noms des classes génériques : Profil, EltRéutilisable, Attribut, AttributNonFeuille, AttributFeuille, EltDeContenu, Ressource, Concepts, ExpressionLogique, TypeDeValeur. Ces classes sont toutes disjointes les unes des autres ;
2. les noms des associations entre classes génériques : estComposéDe, estAssociéA, décrit, représente, estExplicitéPar, estModéliséPar, estDeType ;
3. les noms des classes d'associations entre classes génériques : LienRessources, LienConcepts, LienTypesDeValeurs, OpérateurLogique ;
4. des noms d'instances de la classe générique *Ressource* : profil_information, profil_utilisateur, profil_professionnel_utilisateur, etc. ;
5. des noms d'instances de la classe générique *Concept* : taille_caractères, taille_écran, taille_mémoire, etc. ;
6. des noms d'instances de la classe générique *TypeDeValeur* : jjmmaaaa (format de dates), aaaa (format d'années), hhmmss (format d'heures), pouces, centimètres, ko (kilo octets), etc. ;
7. des noms d'instances des classes d'associations du modèle générique : estUnePartieDe, estComposéDe, rdf : type, rdfs : subClassOf, owl : equivalentClass, etc.

4.2.1.2 Ontologies globales

Les ontologies globales sont indépendantes de toute instance de profil. Leur sémantique est donc applicable à l'analyse de tout profil de l'environnement d'appariement. Ces ontologies vont regrouper des métadonnées existantes (issues des standards ou normes) ainsi que de nouvelles métadonnées qui seront créées par diverses applications éventuellement. Ces ontologies comprendront une taxinomie (organisation des métadonnées) avec des liens sémantiques entre termes de cette taxinomie qui sont interprétables par les outils existants du web sémantique comme : l'équivalence, l'égalité, la transitivité, la symétrie, la généralisation/spécialisation, etc.

Les ontologies globales vont résulter généralement de l'union entre plusieurs ontologies. Dans notre architecture, elles sont réparties en quatre groupes :

1. *ontologie de ressources* représentée par le fichier *Ressources.rdf*, qui va définir principalement des propriétés de généralisation/spécialisation (*rdf : type et rdfs : subClassOf*) entre classes de ressources. On peut également y retrouver d'autres propriétés comme l'équivalence (*owl : equivalentClass*), l'égalité (*owl : sameAs*), la disjonction (*owl : disjointWith*), etc. Ces propriétés représentent des instances de la classe d'association *LienRessources* et sont définies exclusivement entre instances de la classe *Ressource* de notre modèle générique de profil ;
2. *ontologie de concepts* représentée par le fichier *Concepts.rdf*, qui va définir des liens entres concepts. Ces concepts sont issus majoritairement de la norme Dublin Core car cette dernière a été créée pour décrire des ressources. Cependant, ils peuvent également provenir d'autres ontologies existantes. Si ces ontologies ne sont pas des documents RDF, il faut les transformer en fichier RDF afin de les inclure dans l'ontologie de concepts. Ceci est une tâche relativement facile puisque toute relation entre deux concepts peut être traduite sous-forme de triplet RDF ;
3. *ontologie de types de valeurs* représentée par le fichier *TypesDeValeurs.rdf*, qui va décrire des liens entre types de données. Ces types de données sont principalement issus de l'espace de noms associé à XMLSchema ainsi que de l'espace de noms que nous avons créé (*SemanticProfile_NameSpace*). Les liens entre types peuvent également être des liens de : généralisation/spécialisation, d'équivalence, etc. ;
4. *ontologie de valeurs* représentée par le fichier *ValeursContenu.rdf*, qui va illustrer des liens entres *valeurs* ou mots pouvant décrire le contenu ou la sémantique du contenu d'un profil (*cf.* section 3.3.2 sur la description des propriétés *valeur* et *valeurDeComparaison* des classes *EltDeContenu* et *ExpressionLogique* de la figure *Fig.* 3.4). Les liens entre valeurs sont des liens issus généralement d'ontologies de domaine existantes comme WordNet, ou des liens définis dans notre espace de noms (*SemanticProfile_NameSpace*) : la synonymie, l'abréviation, etc.

4.2.1.3 Librairies de règles de transformations de types

Notre environnement d'exploitation de profils va disposer d'une (ou de plusieurs) librairie(s) de règles de transformations de types de données. Il s'agit d'une liste de fonctions permettant la transformation d'une valeur, d'un type de données à un autre type de données. Chaque fonction est nommée par la concaténation des noms des types de données séparés par le caractère de soulignement : « _ ». Ainsi, le nom *Type1_Type2* est le nom de la fonction

permettant de transformer une valeur de type *Type1* en une valeur de type *Type2*.

4.2.1.4 Instances de profils

Les différentes instances de profils vont être définies avec le formalisme RDF, tout en ré-utilisant principalement le vocabulaire des langages RDFS, OWL, XMLSchema, DC. Ces profils seront tous des fichiers d'extension RDF, ce qui signifie qu'ils peuvent être décrits sous forme de triplets RDF. De plus, ils vont respecter le modèle générique de profil que nous avons proposé (*cf.* sections 3.3.1, 3.3.2 et 3.3.3).

4.2.1.5 Index de profils

En recherche d'information classique, un index du contenu des documents mis à disposition est construit afin d'optimiser le temps d'accès aux documents à partir d'une requête. Il se pose le même problème lorsque l'on veut comparer un profil avec une liste de profils sur des attributs feuilles ayant une liste importante d'éléments de contenu. Pour pallier cela, nous définissons des index de profils.

La structure de ces index est similaire à celle des index traditionnellement utilisés en RI. Elle a la forme suivante : *mot + nombre de profils dans lesquels il apparaît + liste des profils dans lesquels il apparaît avec à chaque fois son poids dans ces profils.*

Pour les expérimentations, les index ont été définis uniquement pour les attributs de profils reliés au concept *dc : subject* car leur nombre d'éléments de contenu est généralement important. Ainsi, on aura par exemple dans un index, les lignes suivantes :

```
mot_a 3 profil_c 0,25 profil_e 0,5 profil_f 0,7
mot_b 3 profil_j 0,32 profil_l 0,4 profil_m 0,36
mot_c 4 profil_d 0,25 profil_g 0,12 profil_f 0,42 profil_n 0,53
mot_d 2 profil_h 0,25 profil_k 0,87
```

Notons qu'un index est généralement trié par ordre alphabétique des mots pour optimiser la recherche au sein de l'index. Par ailleurs, on va pouvoir le coupler avec des ontologies qui décrivent des relations entre mots par des liens de synonymie, d'abréviation, etc. Pour cela, on exploite le fichier *ValeursContenu.rdf*.

L'index va permettre également d'éviter des comparaisons inutiles car on ne considère un profil dans l'appariement que s'il contient au moins une valeur d'élément de contenu en commun avec le profil de base de la comparaison. Le profil de base est le profil qui est comparé à une liste d'autres profils.

Dans la section suivante, nous allons présenter la compatibilité qui existe entre RDF et OWL à travers une étude comparative des différents ensembles de langages de OWL à savoir : OWL Full, OWL DL et OWL Lite. Le but de cette section est d'expliquer le choix de l'extension ou du type des fichiers de notre architecture, qui est : « *.rdf* ».

4.2.2 RDF et Ontologies OWL : OWL Full vs OWL DL vs OWL Lite

En général, *un document RDF est OWL Full* à moins qu'il ne soit spécifiquement construit pour être compatible avec OWL DL ou Lite. Nos ontologies sont des fichiers RDF, ce qui nous permet d'utiliser OWL Full sans nous interdire de nous restreindre à OWL DL ou Lite. OWL Full est donc conçu pour une compatibilité maximale avec RDF. OWL Full est ainsi le point de départ naturel pour les utilisateurs de RDF. Notons qu'avant de choisir OWL DL ou OWL Lite, on doit se demander si les avantages de OWL DL/Lite (support de raisonnement fiables) sont plus importants que les restrictions sur l'usage des constructions OWL et RDF. Dans notre cas de figure, nous souhaitons le maximum de flexibilité et en fonction des résultats des expérimentations (*cf. chapitre 5*), on peut être amené à introduire éventuellement plus de contraintes.

Afin, de pouvoir analyser et faire des inférences sur nos profils, nous allons utiliser un langage d'interrogation de documents RDF nommé : *SPARQL* [PS05]. Le choix de ce langage se justifie par le fait qu'il offre un meilleur compromis, par comparaison aux autres langages d'interrogation de documents RDF. Cette étude comparative de langages d'interrogation de documents RDF ainsi que la description du langage SPARQL sont présentées en annexe de ce document.

Après avoir décrit l'environnement général d'exploitation de nos profils, nous allons présenter, dans les sections suivantes, l'usage effectif de ces profils dans les processus d'accès. L'opération clé de l'accès à l'information est l'appariement, cependant afin d'éviter toute incohérence, du fait de la flexibilité de description de nos profils, nous allons également procéder (préalablement à l'appariement) à une phase d'analyse sémantique de profils. Cette phase vise à déterminer les couples d'éléments de profils que l'on va pouvoir comparer, du fait de la compatibilité de leur sémantique.

4.3 Détermination de couples d'éléments appariables

L'étape préalable à l'appariement de profils est celle de la détermination de couples d'éléments appariables des profils à comparer. Ces éléments doivent avoir une sémantique compatible pour que l'opération d'appariement

ait un sens. La sémantique va donc définir le niveau d'interopérabilité (ou capacité d'interaction) entre profils, qui correspond ici au nombre de couples d'éléments de sémantique compatible.

4.3.1 Illustration de l'interopérabilité de profils

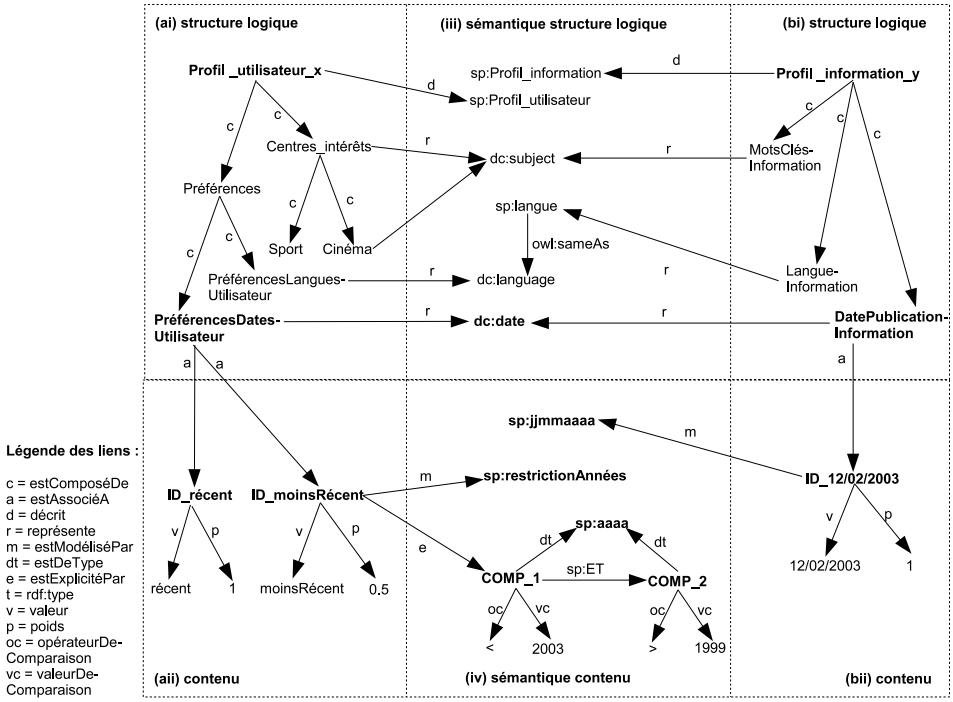

FIGURE 4.2 – Illustration de l'interopérabilité de profils

On peut définir l'interopérabilité comme étant la capacité de plusieurs systèmes, qu'ils soient identiques ou radicalement différents, à communiquer sans ambiguïté et (co-)opérer (ou interagir) ensemble pour la réalisation d'une tâche. Pour faciliter l'interopérabilité entre profils pour l'accès à des ressources, il est nécessaire de définir la sémantique de chaque élément de la structure logique et du contenu de ces profils mais aussi les règles permettant d'analyser cette sémantique. Ces règles vont permettre d'identifier les couples d'attributs de base entre profils que l'on va pouvoir comparer du fait de la compatibilité de leur sémantique. Ces couples vont donc définir un niveau d'interopérabilité possible entre profils pour leur appariement.

Par ailleurs, les résultats de comparaison de ces couples d'attributs vont être agrégés pour calculer un degré de ressemblance entre les profils que l'on souhaite apparier. Notons que les attributs de base ici sont des attributs feuilles parce qu'ils sont directement associés à des éléments de contenu. La

similarité entre attributs non feuilles ou entre profils est donc tout simplement la résultante d'une combinaison d'appariements entre attributs feuilles.

La figure *Fig. 4.2* illustre l'interopérabilité qui peut exister entre un profil utilisateur et un profil d'information. Cette interopérabilité implique :

1. *une analyse de la structure logique et de la sémantique associée* : elle permet de se rendre compte de la compatibilité entre les concepts associés aux attributs des deux taxinomies. L'analyse de la sémantique de la structure logique va permettre de savoir quels couples d'attributs de ces arborescences de profils décrivent des concepts identiques ou du moins compatibles et donc sont potentiellement appariables. Par exemple, dans la figure *Fig. 4.2*, les couples d'attributs : *MotsClésInformation et Cinéma*, *LangueInformation et PréférencesLanguesUtilisateur*, *DatePublicationInformation et PréférencesDatesUtilisateur* représentent les mêmes concepts génériques même s'ils ne portent pas les mêmes noms. Ces concepts génériques sont respectivement : *dc : subject, dc : language, dc : date*. L'analyse de la structure logique et de la sémantique associée définit une *règle nécessaire d'appariement* entre des attributs feuilles : ces attributs peuvent être considérés comme étant *nécessairement appariables ou nécessairement compatibles* ;
2. *une analyse du contenu et de la sémantique associée* : elle permet de se rendre compte de la compatibilité qui existe entre les types de données des éléments de contenu d'attributs feuilles. L'analyse de la sémantique du contenu va permettre de savoir si des attributs feuilles associés à des concepts compatibles sont effectivement appariables. Par exemple, dans la figure *Fig. 4.2*, les attributs feuilles *DatePublicationInformation et PréférencesDatesUtilisateur* sont nécessairement appariables car ils représentent le même concept générique *dc : date*. Par contre, la sémantique de leurs éléments de contenu n'est pas identique. Les préférences de l'utilisateur sont décrites sous forme de restrictions d'années : par exemple, l'élément de contenu *ID_ moinsRécent* est modélisé par le type de données *sp : RestrictionAnnées* et est explicité par des expressions logiques dont les valeurs sont de type sp : aaaa (format d'année). La date de publication de l'information décrite, quant à elle, est représentée par un format de date : l'élément de contenu *ID_ 12/02/2003* est modélisé par le type de données *sp : jjmmaaaa*. Ce qui serait intéressant ici, c'est de pouvoir analyser cette sémantique et de déduire que l'on va pouvoir comparer ces attributs (étant donné que les types de données *sp : jjmmaaaa et sp : aaaa* sont compatibles) moyennant des transformations qui auraient été clairement identifiées : transformation de type de données (extraction de l'année à partir d'une date), changement de la combinaison linéaire ou du modèle vectoriel de représentation des éléments de contenu. Ainsi, l'analyse du contenu et de la sémantique associée d'attributs feuilles,

va compléter la règle nécessaire d'appariement pour la rendre, d'une certaine façon, suffisante.

Ces deux niveaux d'analyse sont nécessaires car on peut avoir des attributs feuilles qui représentent des concepts compatibles mais dont les types de données des éléments de contenu sont différents. Par exemple, l'attribut *PréférencesDatesUtilisateur* d'un utilisateur et l'attribut *DatePublicationInformation* d'une information décrits dans la figure *Fig. 4.2*. De même, des attributs ayant des types de données de contenu identiques ou compatibles ne représentent pas forcément des concepts compatibles. Par exemple, la langue d'une information et les centres d'intérêts d'un usager peuvent avoir, tous les deux, leur contenu décrit par des chaînes de caractères. Cependant, l'appariement entre ces deux attributs pose un problème de cohérence car ils ne décrivent pas des concepts compatibles (*dc : language et dc : subject*, par exemple).

En résumé, l'intérêt de la sémantique dans la gestion de l'interopérabilité de profils, comme dans la figure FIG. 4.2, est qu'elle fournit de la flexibilité dans la description de profils. Elle permet de donner les noms que l'on souhaite aux éléments d'un profil sans perturber ni restreindre leur exploitation. L'analyse de cette sémantique permet de s'affranchir d'une compatibilité sémantique implicite (définie à priori) entre attributs. Elle permet également de réduire les incohérences et les silences liés à une recherche de compatibilité sémantique entre éléments de profils basée uniquement sur la structure logique, surtout si l'on se projette au niveau inter-applications. On va donc pouvoir comparer des profils décrits par des taxinomies différentes et éventuellement issus d'applications différentes tout en garantissant une certaine fiabilité des résultats.

L'étape préalable et essentielle à cet appariement va consister à déterminer les couples d'attributs feuilles que l'on va pouvoir comparer. Pour cela, on va utiliser un parser RDF qui, étant donné un document RDF, renvoie l'ensemble de ses triplets. Ensuite, l'ensemble des triplets des profils à comparer est analysé afin d'en déterminer les couples d'attributs feuilles de sémantiques compatibles. Les règles d'analyse vont être décrites mathématiquement mais également en utilisant un langage d'interrogation de documents RDF nommé *SPARQL* [PS05] qui nous permet de faire des inférences et de déduire ainsi certaines informations implicites (attributs de sémantique compatible). Les sections suivantes décrivent donc formellement l'algorithme général d'analyse sémantique de profils pour la détermination d'éléments appariables ainsi que les différentes étapes de cet algorithme.

4.3.2 Algorithme général d'analyse sémantique de profils

La figure *Fig.* 4.3 [CSDT06a] illustre l'algorithme général d'analyse sémantique de profils avec ces différentes étapes, les paramètres en entrée et

ceux obtenus en sortie. En entrée de cet algorithme, nous avons les deux profils à comparer et en sortie, nous obtenons une liste de couples d'attributs appariables avec pour chacun d'eux les combinaisons linéaires (ou espaces vectoriels de représentation) du contenu de chacun des attributs du couple.

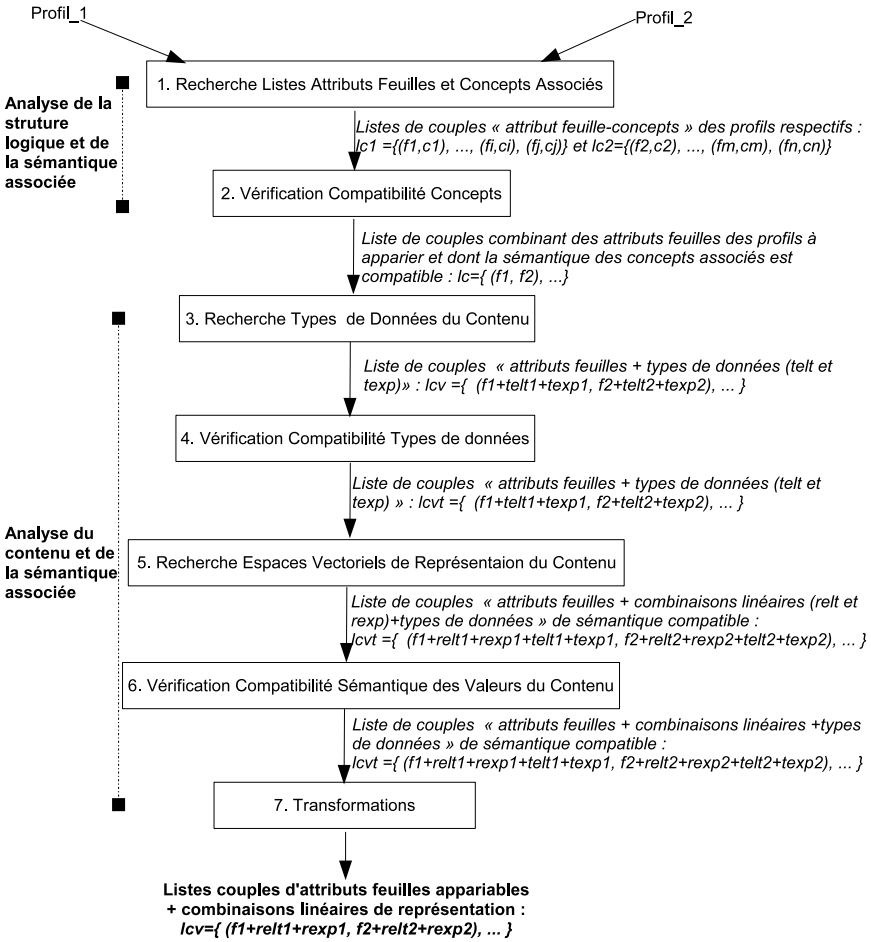

FIGURE 4.3 – Algorithme général d'analyse de profils à apparier

Les étapes de l'algorithme sont regroupées en deux catégories : celles relatives à l'analyse de la structure logique et de la sémantique associée (étapes 1 et 2) ; celles relatives à l'analyse du contenu et de la sémantique associée (étapes 3 à 7). Ces étapes peuvent également être subdivisées en trois classes :

— celles relatives au parcours des graphes de profils pour la recherche d'éléments spécifiques (étapes 1, 3 et 5) ;
— celles relatives au raisonnement (ou à des inférences) pour des vérifications de compatibilité (étapes 2, 4 et 6) ;

— celle relative à des transformations (étape 7) pour uniformiser les représentations de contenu en vue de l'appariement.

Ainsi, l'analyse de la structure logique permet de déterminer la liste des couples d'attributs feuilles qui représentent des concepts compatibles. Cette analyse définit la *règle nécessaire* d'appariement entre deux attributs feuilles issus de profils différents. Ensuite, cette règle est complétée par l'analyse de la sémantique de leur contenu qui permet de s'assurer de la compatibilité sémantique effective (suffisante) des couples d'attributs feuilles à apparier. Les combinaisons linéaires du contenu des attributs feuilles obtenues au terme de l'algorithme seront utilisées pour mesurer un degré de similarité entre les attributs de chaque couple.

4.3.3 Analyse basée sur la structure logique et la sémantique associée

L'analyse de la structure logique d'un profil et de la sémantique associée est subdivisée en différentes étapes :

— identification de la liste d'attributs feuilles et des concepts associés pour chaque profil à apparier ;

— vérification de la compatibilité entre concepts pour toute paire possible d'attributs feuilles des profils à comparer.

4.3.3.1 Détermination de listes d'attributs feuilles et des concepts associés

La liste des attributs feuilles d'un profil donné, nommé $profil_1.rdf$ par exemple, est déterminée en parcourant sa structure logique (qui est sous la forme d'un arbre) et en vérifiant chaque fois, pour un élément donné du profil, s'il appartient à l'ensemble des instances de la classe *AttributFeuille*. L'arbre de la structure logique est décrit par des triplets RDF où le sujet est un nœud père, l'objet un nœud fils et le prédicat l'arc étiqueté qui relie le nœud père au nœud fils.

Par ailleurs, l'analyse de la sémantique des attributs feuilles d'un profil passe par la détermination des concepts auxquels ces attributs sont rattachés. Ces concepts explicitent les caractéristiques génériques décrites par ces derniers et sont reliés à l'attribut feuille par le prédicat *sp : représente*.

Formellement, soit G la liste des triplets (ou graphe) d'un profil, F la liste des attributs feuilles de ce profil, C la liste des concepts associés à ce profil et nf un attribut quelconque, la liste des attributs feuilles (notés f) et des concepts associés (notés c) de ce profil est déterminée ainsi :

$\forall[nf, \text{sp} : \text{estComposéDe}, f] \in G$, si $\exists[f, \text{rdf} : \text{type}, AttributFeuille]$

et $\exists([f, \text{sp} : \text{représente}, c], [c, \text{rdf} : \text{type}, \text{sp} : \text{Concept}]) \Rightarrow f \in F$ et $c \in C$

Le tableau Tab. 4.1 décrit, sous forme d'une requête SPARQL, la liste des attributs feuilles (notés f) d'un profil avec pour chacun d'eux le concept

(noté c) qu'il représente.

Le mot clé *OPTIONAL*, qui définit un chemin optionnel dans un graphe, permet ici de retourner également des attributs feuilles dont les concepts associés n'auraient pas été définis.

```
PREFIX rdf :     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sp :      <http://www.irit.fr/SIG/.../SemanticProfile>

SELECT  ?f  ?c
FROM <profil_1.rdf>
WHERE { [] sp:estComposéDe ?f .
          ?f rdf:type sp:AttributFeuille .
OPTIONAL { ?f sp:représente ?c .  ?c rdf:type sp:Concept }
          }
```

TABLE 4.1 – Sélection de la liste des attributs feuilles et des concepts associés d'un profil

4.3.3.2 Vérification de compatibilité entre concepts

Pour déterminer les couples d'attributs feuilles appariables entre deux arborescences de profils, il faudrait au préalable vérifier la compatibilité des concepts associés à chaque paire possible d'attributs feuilles entre ces profils. Cette vérification peut être vue comme une *règle nécessaire* bien que non suffisante pour l'appariement de deux attributs feuilles. Cette règle s'énonce comme suit : *deux attributs feuilles sont appariables s'ils représentent des concepts compatibles.* Pour cela, ils doivent être connectés au même concept générique (par exemple : *dc : subject, dc : language, etc.*) par le prédicat *représente* (noté r) ou être connectés à des concepts différents (c'est-à-dire définis par des URI différentes) mais compatibles car *égaux* (c'est-à-dire décrivant la même chose) ou *équivalents* (c'est-à-dire ayant les mêmes instances).

Cette règle nommée *règle nécessaire* se définit formellement comme suit : soit F_1 et F_2 les ensembles respectifs des attributs feuilles des profils à comparer, C l'ensemble des concepts rattachés à ces attributs feuilles, G l'ensemble des triplets des profils, G_c le graphe des concepts de référence de l'architecture générale d'appariement (*cf.* fichier *Concepts.rdf* de la figure FIG. 4.1), $f_1 \in F_1$ et $f_2 \in F_2$, f_1 et f_2 peuvent être appariés si et seulement si :

- soit $\exists$ *[*f_1*, sp : représente, c]* $\in G$ *et* $\exists$ *[*f_2*, sp : représente, c]* $\in G$ *et* $c \in C$;
- soit $\exists$ *[*f_1*, sp : représente,* c_1*]* $\in G$, $\exists$ *[*f_2*, sp : représente,* c_2*]* $\in G$, $c_1 \in C$, $c_2 \in C$ *et* $\exists$ *[*c_1*, owl : equivalentClass,* c_2*]* $\in G_c$ *ou* $\exists$ *[*c_1*, owl : sameAs,* c_2*]* $\in G_c$.

Lorsque les concepts rattachés aux attributs feuilles sont différents (proviennent de différents espaces de noms ou ont des noms différents), on peut donc vérifier s'ils sont équivalents ou identiques. Pour cela, on peut utiliser une requête SPARQL similaire à celle du tableau TAB. 4.2. Notons que les symboles « && » et « || » représentent respectivement le *ET* et le *OU* logique et que c_{f_1} et c_{f_2} sont des noms de concepts trouvés pour les attributs feuilles f_1 et f_2 des profils respectifs $profil_1$ et $profil_2$ que l'on souhaite comparer.

De plus, les propriétés *owl : sameAs et owl : equivalentClass*, traduisant respectivement l'égalité et l'équivalence, sont définies comme étant symétriques et transitives. La symétrie va permettre de déduire automatiquement les triplets : *?*c_2 *owl : equivalentClass ?*c_1 et *?*c_2 *owl : sameAs ?*c_1 même si ces derniers n'existent pas explicitement dans le fichier *Concepts.rdf*. De même, la transitivité va permettre de déduire par exemple un triplet *?*c_1 *owl : equivalentClass ?*c_3 alors qu'effectivement, il n'existe que les triplets *?*c_1 *owl : equivalentClass ?*c_2 et *?*c_2 *owl : equivalentClass ?*c_3.

Par ailleurs, le mot clé *UNION*, qui permet de définir une alternative entre des chemins de graphes, est employé ici car entre deux concepts donnés, on peut avoir l'une ou l'autre des propriétés *owl : sameAs* ou *owl : equivalentClass*.

```
PREFIX owl :      <http://www.w3.org/2002/07/owl#>

SELECT  ?c1  ?c2
FROM <Concepts.rdf>
WHERE { ?c1 rdf : type sp : Concept  .
          ?c2 rdf : type sp : Concept  .
        { ?c1 owl : equivalentClass ?c2 } UNION { ?c1 owl : sameAs ?c2 } .
FILTER ( ( ?c1=<cf1 > || ?c1=<cf2 >) &&
           ( ?c2=<cf1> || ?c2=<cf2>) ) }
```

TABLE 4.2 – Vérification de la compatibilité entre concepts

De façon générale, si on veut identifier tous les couples d'attributs feuilles qui vérifient la *règle nécessaire* d'appariement entre deux profils donnés $profil_1$ *et* $profil_2$, on doit tout d'abord déterminer pour chaque attribut feuille (noté f_i) le concept auquel il est rattaché (noté c_i) (*cf.* section 4.3.3.1) et par la suite, vérifier pour toute paire possible d'attributs feuilles des profils à comparer, la compatibilité entre concepts associés.

Une fois l'analyse de la structure logique et de la sémantique associée effectuée, il faut vérifier la compatibilité de la sémantique de contenu associé aux couples d'attributs feuilles obtenus par évaluation de la *règle nécessaire*.

Pour l'analyse de la sémantique du contenu d'un profil, nous considérons que tous les éléments de contenu d'un attribut feuille sont de même type c'est-à-dire qu'ils sont tous modélisés par la même instance de la classe *Ty-*

peDeValeur. Par exemple, soit l'attribut feuille *PréferencesDatesUtilisateur* associé aux éléments de contenu suivants : *(moinsRécent, 0.5) et (récent, 1)*, les valeurs (*moinsRécent* et *récent*) de ces éléments de contenu sont toutes modélisées par le même type de données *sp : restrictionAnnées*. De la même manière, toutes les instances de la classe *ExpressionLogique* qui explicitent les éléments de contenu d'un attribut sont également du même type de données. Ainsi, soient les expressions logiques suivantes : *(<,2003) et (>=, 2003)* qui explicitent respectivement les éléments de contenu *(moinsRécent, 0.5) et (récent, 1)* de l'attribut *PréferencesDatesUtilisateur*, les valeurs de comparaison (*2003* et *2003*) de ces expressions logiques sont toutes du même type de données *sp : aaaa*. Notons que ces contraintes s'expliquent par le fait qu'un attribut feuille est un élément de description élémentaire et donc possède, de ce fait, un contenu homogène.

Par contre, ces contraintes n'apparaissent pas clairement sur le modèle générique de la figure FIG. 3.4 où le lien sémantique *estModélisé Par* est relié plutôt à la classe *EltDeContenu* et pas à la classe *AttributFeuille*. Ceci s'explique par le fait que le lien *estModélisé Par* explicite la sémantique de la valeur d'un élément de contenu et non celle de l'attribut feuille. De plus, l'avantage de la position de ce lien est qu'il permet d'avoir une séparation stricte entre la structure logique, le contenu, la sémantique de la structure logique et la sémantique du contenu. Ceci permet d'avoir une vision modulaire et indépendante des différentes parties de notre modèle générique.

4.3.4 Analyse basée sur le contenu et la sémantique associée

L'analyse du contenu et de la sémantique qui y est associée va permettre de compléter la règle nécessaire définie précédemment, pour déterminer la liste effective des couples d'attributs feuilles de profils à comparer. Cette analyse va donc rendre la règle nécessaire et *suffisante* pour la détermination de couples d'attributs feuilles appariables.

Ainsi, pour vérifier la compatibilité des contenus de deux attributs feuilles *nécessairement appariables*, c'est-à-dire qui vérifie la règle nécessaire d'appariement, nous allons :

- rechercher les types de données utilisés pour représenter le contenu associé à chaque attribut feuille ;
- vérifier la compatibilité des types de données ainsi obtenu ;
- déterminer les *combinaisons linéaires ou espaces vectoriels* de représentation du contenu de ces attributs ;
- vérifier la compatibilité sémantique des valeurs du contenu de ces attributs.

Cette vérification de compatibilité de contenus va permettre aussi d'identifier un ensemble de transformations éventuelles à appliquer sur les contenus des attributs feuilles, afin d'homogénéiser leur représentation, avant de les apparier.

4.3.4.1 Recherche des types de données

L'analyse de la sémantique associée au contenu passe par la recherche des types de données des éléments de contenu associés à un attribut, ainsi que par la recherche du type de données des expressions logiques qui sont rattachées, éventuellement, à ces éléments de contenu. Rappelons que tous les éléments de contenu d'un attribut feuille sont modélisés par le même type de données. De même, toutes les expressions logiques qui explicitent les éléments de contenu d'un même attribut feuille sont également de même type.

Ainsi, la requête SPARQL qui permet de déterminer formellement les types de données du contenu d'un attribut feuille f (nommé *Préférences-DatesUtilisateur*) est illustrée dans le tableau TAB. 4.3, où t_{elt_f} est le type des éléments de contenu et t_{exp_f} le type des expressions logiques. Le mot clé *OPTIONAL*, qui définit un chemin optionnel dans un graphe, permet ici de retourner également des attributs feuilles dont le contenu n'est pas explicité par des expressions logiques mais décrit uniquement avec des éléments de contenu.

```
PREFIX rdf :     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sp :      <http://www.irit.fr/SIG/.../SemanticProfile>

SELECT  ?f  ?t_elt_f  ?t_exp_f
FROM <profil_1.rdf>
WHERE { [] sp:estComposéDe ?f .
          ?f rdf:type sp:AttributFeuille .
          ?f sp:estAssociéA ?elt .
          ?elt rdf:type sp:EltDeContenu .
          ?elt sp:estModéliséPar ?t_elt_f .
OPTIONAL { ?elt sp:estExplicitéPar ?exp .
          ?exp rdf:type sp:ExpressionLogique .
          ?exp sp:estDeType ?t_exp_f . }
FILTER ( ?f =<PréférencesDatesUtilisateur>) }
```

TABLE 4.3 – Détermination des types de données du contenu de l'attribut feuille *PréférencesDatesUtilisateur*

Formellement, les types de données t_{elt_f} et t_{exp_f} d'un attribut feuille f sont décrits comme suit : $\forall$ *[f, sp : estAssociéA, elt]* $\Rightarrow \exists$ *[elt, sp : estModéliséPar,* t_{elt_f}*]* *et si* $\exists$ *[elt, sp : estExplicitéPar, exp]* $\Rightarrow \exists$ *[exp, sp : estDeType,* t_{exp_f}*]* .

4.3.4.2 Vérification de la compatibilité des types de données

Pour apparier deux attributs feuilles, il faut également vérifier la compatibilité des types de données associés à leur contenu respectifs. Cette vérification consiste à parcourir la librairie des règles de transformations de types de données et vérifier s'il existe, pour deux types de données t_{f1} et t_{f2}, des attributs f_1 et f_2 respectifs que l'on souhaite comparer, une méthode permettant de transformer une valeur de type t_{f1} en une valeur de type t_{f2} ou inversement. Ainsi, si $\exists$ *une méthode nommée* $t_{f1\_}t_{f2}$ *ou* $\exists$ *une méthode nommée* $t_{f2\_}t_{f1}$ dans la librairie de notre architecture générale (*cf.* sections 4.2.1 et 4.2.1.3), alors ces types sont compatibles.

Notons que l'on peut vérifier cette compatibilité en analysant et en interprétant les liens sémantiques qui existent entre les types de données. Cependant, cette alternative peut s'avérer assez coûteuse et fastidieuse du fait de la variété des liens qui peuvent exister entre types de données : *estUnePartieDe, rdf : type, etc.* De plus, cette analyse serait inutile si la méthode n'a pas réellement été définie. Nous avons donc choisi la solution qui consiste à dire tout simplement que si la méthode de transformation de types n'est pas implémentée alors cette transformation n'est pas possible même si en réalité les types de données en question sont compatibles. Par conséquent, nous n'exploitons pas vraiment le fichier *TypesDeValeurs.rdf* de notre architecture générale d'appariement (*cf.* sections 4.2.1 et 4.2.1.2).

Au terme de cette étape (étape 4), nous pouvons déjà conclure de la possibilité effective d'appariement entre les attributs des couples obtenus. Notons que les étapes 2 et 4 précédemment décrites agissent comme des filtres qui éliminent les éléments qui ne respectent pas les contraintes de ladite étape. Par contre, les étapes suivantes vont servir principalement à déduire le modèle de représentation du contenu des attributs sélectionnés, afin de préparer l'appariement.

4.3.4.3 Recherche des espaces vectoriels de représentation du contenu

La liste des valeurs des éléments de contenu ainsi que celles des expressions logiques, qui explicitent les éléments de contenu associés à un attribut feuille, vont permettre de définir le(s) espace(s) vectoriel(s) de représentation du contenu de ce dernier. Pour déterminer ces différents espaces vectoriels, deux cas de figures peuvent se présenter :

1. *les éléments de contenu de l'attribut feuille ne sont pas explicités au travers d'expressions logiques.* Dans ce cas, cet attribut est représenté dans un espace vectoriel qui est défini par la liste des valeurs de ses éléments de contenu. Ces valeurs vont donc constituer les vecteurs de base de l'espace vectoriel de représentation et leurs poids vont servir à exprimer la combinaison linéaire des éléments de contenu

de l'attribut dans son espace de représentation. Ainsi, soit l'attribut feuille *PréférencesDatesUtilisateur* associé aux éléments de contenu *(récent, 1) et (moinsRécent, 0.5)*, l'espace vectoriel de représentation est défini par les vecteurs (ou valeurs) *récent et moinsRécent*. Ainsi, *PréférencesDatesUtilisateur* peut s'écrire sous la forme de la combinaison linéaire suivante :

$$\overrightarrow{Préférences Dates Utilisateur} = 1 \,.\, \overrightarrow{récent} + 0,5 \,.\, \overrightarrow{moins Récent}$$

Formellement dans ce cas de figure, l'espace vectoriel (noté par exemple ε_1) associé au contenu d'un attribut feuille f est défini comme suit :

$\forall$ *[f, sp : estAssociéA, elt]* $\Rightarrow \exists$ *[elt, sp : valeur,* v_{elt_f}*]*, $\overrightarrow{v_{elt_f}} \in \varepsilon_1$;

2. *les éléments de contenu de l'attribut feuille sont explicités au travers d'expressions logiques.* Dans ce cas, l'attribut feuille peut être représenté dans différents espaces vectoriels : un espace vectoriel défini par les éléments de contenu et un autre espace vectoriel défini par l'ensemble des expressions logiques qui explicitent ces éléments de contenu. Ainsi, soit l'attribut feuille *PréférencesDatesUtilisateur* dont les éléments de contenu *« (récent, 1) et (moinsRécent, 0.5) »* sont explicités via des expressions logiques comme suit :
   — et l'élément de contenu *(récent, 1)* est explicité par l'expression logique *« (>=, 2003) »* ;
   — l'élément de contenu *(moinsRécent, 0.5)* est explicité par l'expression logique *« (<, 2003) ET (>, 1999) »*.

   L'espace vectoriel associé aux éléments de contenu est le même que celui défini précédemment et est nommé ε_1. Par contre, l'espace vectoriel associé aux expressions logiques est défini par l'ensemble des valeurs de comparaison de ces dernières, précédées de leur opérateur de comparaison et éventuellement reliées entre elles par des opérateurs logiques. Pour notre exemple, on obtient la liste de vecteurs suivante : *« >=2003 », « <2003 ET >1999 »*. Ici, le *poids* des vecteurs (éléments de la liste d'expressions logiques) est affecté automatiquement et est toujours égal à *« 1 »* puisqu'il s'agit d'expressions logiques vérifiées par les éléments de contenu qu'elles explicitent. Ainsi, soit ε_2 ce nouvel espace vectoriel, l'attribut *PréférencesDatesUtilisateur* (noté ici P) s'y écrit sous la forme de la combinaison linéaire suivante :

$$\overrightarrow{P} = 1 \,.\, \overrightarrow{GE\_2003} + 1 \,.\, \overrightarrow{LT\_2003\_ET\_GT\_1999}$$

où :

les opérateurs de comparaisons sont remplacés par deux caractères : LT pour « < », EQ pour « = », GT pour « > », LE pour « <= », GE pour « >= » ;

et les éléments d'une expression logique (opérateur de comparaison, valeur de comparaison et éventuellement opérateur logique) sont reliés par le caractère de soulignement.

Formellement, le vecteur décrivant une expression logique associée à un élément de contenu d'un attribut feuille f est défini comme suit :

— d'abord on recherche le premier élément de l'expression logique de chaque élément de contenu (noté *elt*) d'un attribut donné (noté f). Ce premier élément (noté *exp*) qui décrit une comparaison, est tel que : $\forall$ *[f, sp : estAssociéA, elt]* $\Rightarrow \exists$ *[elt, sp : valeur,* v_{elt_f}*] tel que :* $\forall$ *[elt, sp : estExplicitéPar, exp]* $\Rightarrow \exists$ *[exp, sp : valeur,* v_{exp_f}*] et* $\exists$ *[exp, sp : opérateurDeComparaison, opc]*

— ensuite, on reconstitue l'expression logique complète de chaque élément de contenu (noté *elt*) de l'attribut f. Pour cela, on parcourt tous les patterns du graphe de profil issu du nœud nommé *exp*, associé au nœud *elt*, trouvé précédemment.

PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sp : <http://www.irit.fr/SIG/.../SemanticProfile>

SELECT ?f ?elt ?v_{elt_f} ?p ?exp ?opc ?v_{exp_f}
FROM <$profil_1.rdf$>
WHERE { [] sp : estComposéDe ?f .
?f rdf : type sp : AttributFeuille .
?f sp : estAssociéA ?elt .
?elt sp : valeur ?v_{elt_f} .
?elt sp : poids ?p .
OPTIONAL { ?elt sp : estExplicitéPar ?exp .
?exp sp : valeur ?v_{exp_f} .
?exp sp : opérateurDeComparaison ?opc . }
FILTER (?f = <PréférencesDatesUtilisateur>) }

TABLE 4.4 – Détermination des éléments des espaces vectoriels de représentation du contenu de l'attribut feuille *PréférencesDatesUtilisateur*

La requête SPARQL qui permet de déterminer les éléments des espaces vectoriels associés à un attribut feuille d'un profil donné est illustrée dans le tableau TAB. 4.4. Notons qu'il faut, par la suite, analyser le résultat de cette requête pour extraire les différentes combinaisons linéaires de représentation de l'attribut. De plus, la valeur *exp* associée à chaque élément de contenu, si elle n'est pas vide, sera utilisée pour reconstituer l'expression logique complète par programme.

4.3.4.4 Vérification complémentaire de la compatibilité de la sémantique des valeurs de contenu

On pourrait également être amené à vérifier la sémantique des valeurs des éléments de contenu ou d'expressions logiques pour plus de flexibilité, en utilisant le fichier *ValeursContenu.rdf* de notre architecture générale d'appariement. Un exemple de ce type de vérification est décrite dans le tableau TAB. 4.5.

Notons que, dans le tableau TAB. 4.5, les propriétés *sp : estUneTraductionDe et sp : estUnSynonymeDe*, traduisant respectivement la traduction d'un mot dans une autre langue et la synonymie entre mots, sont définies comme étant symétriques et transitives dans le fichier *ValeursContenu.rdf*. De plus, val_1 et val_2 sont deux valeurs d'éléments de contenu (ou d'expressions logiques) associées aux attributs feuilles f_1 et f_2 des profils $profil_1$ et $profil_2$ respectivement que l'on souhaite apparier.

```
PREFIX sp:    <http://www.irit.fr/SIG/.../SemanticProfile/>

SELECT ?v1 ?v2
FROM <ValeursContenu.rdf>
WHERE { { ?v1 sp:estUneTraductionDe v2 }
        UNION { ?v1 sp:estUnSynonymeDe v2 }
        UNION { ?v1 sp:estUneAbréviationDe v2 } .
FILTER ( ?v1 = <val1> && ?v2 = <val2>) }
```

TABLE 4.5 – Vérification de la compatibilité des valeurs de contenu d'attributs feuilles à apparier

L'intérêt de cette vérification est qu'elle permet de définir un *espace vectoriel de représentation sémantiquement commun* aux attributs à comparer et permet ainsi d'obtenir un résultat de similarité qui traduit plus fidèlement la ressemblance entre les attributs. Par exemple, on peut avoir les valeurs *fr* et *français* dans la représentation du contenu des attributs f_1 et f_2 que l'on souhaite apparier, avec la valeur *fr* décrivant une abréviation de *français*. Il serait intéressant, dans ce contexte, de représenter ces deux valeurs par le même terme car elles représentent la même sémantique.

Pour compléter cette vérification, on peut également utiliser la représentation RDF/OWL de WordNet pour extraire tous les synonymes d'un mot donné. Les synonymes d'un mot sont représentés par un *synset*. Par exemple, un des synsets associés au mot anglais *car* est : *{car, auto, automobile, machine, motorcar}*. Pour extraire les synsets associés à un mot, on peut utiliser la requête SPARQL décrite dans le tableau TAB. 4.6. Ensuite, on analyse les résultats de cette requête pour vérifier si deux mots sont synonymes (par exemple, les mots *car et automobile*).

```
PREFIX wn:    <http://wordnet.princeton.edu/wn20/>

SELECT  ?aSynset
FROM <wordnet.rdf>
WHERE { ?aSynset wn:containsWordSense ?aWordSense .
        ?aWordSense wn:word ?aWord .
        ?aWord wn:lexicalForm "car"@en }
```

TABLE 4.6 – Listes des synonymes d'un mot

4.3.4.5 Transformations à effectuer sur la représentation du contenu

En général, pour identifier les transformations à effectuer il faut :

1. vérifier la compatibilité des types de données du contenu des attributs à comparer afin d'effectuer, éventuellement, des conversions de types par exécution de la méthode de transformation de type de données correspondante (*cf.* section 4.3.4.2). Ainsi, la vérification de la compatibilité des types de données du contenu d'attributs feuilles à apparier peut conduire à un certain nombre de transformations nécessaires, afin de pouvoir effectuer de façon cohérente l'appariement. Par exemple dans la figure FIG. 4.2, pour apparier les attributs *DatePublicationInformation* et *PréférencesDatesUtilisateur*, il est nécessaire d'avoir une règle qui permette de passer du type *jjmmaaaa* au type *aaaa* ;
2. vérifier la compatibilité sémantique des valeurs des éléments de contenu. Pour cela, on peut exploiter les relations de synonymie, de traduction ou même d'abréviation entre valeurs (*cf.* section 4.3.4.4). Par exemple, soient deux utilisateurs dont les préférences en langues sont exprimées dans des langues différentes : *LanguesUtilisateur1(anglais, 1), (français, 0.5) et LanguesUtilisateur2(english, 1), (french, 0.5)* . On peut décrire ces préférences dans un espace sémantique commun comme suit : *LanguesUtilisateur1(anglais, 1), (français, 0.5) et LanguesUtilisateur2(anglais, 1), (français, 0.5)* . On procède ici par substitution de valeur ;
3. vérifier les espaces vectoriels de représentation du contenu de chaque attribut feuille (listes des valeurs des éléments de contenu ou d'expressions logiques) à apparier. Il peut y avoir une *disjonction*, une *inclusion*, un *recouvrement* entre valeurs (ou vecteurs) des espaces vectoriels des attributs feuilles à comparer. Dans ces cas, pour effectuer l'appariement, il peut être nécessaire de procéder à :

   (a) un changement de combinaison linéaire (ou espace) de représen-

tation par *extension des vecteurs de base* afin de représenter les attributs à comparer dans la même dimension ;

(b) un changement de combinaison linéaire (ou espace) de représentation par *changement des vecteurs de base* si l'un des attributs feuilles à comparer peut être représenté dans différents espaces vectoriels. C'est le cas des attributs qui ont des éléments de contenu explicités par des expressions logiques. Notons que dans ce cas, le type de données des valeurs des éléments de contenu et de leurs expressions logiques sont généralement différents.

Formellement, l'identification du type de changement d'espace de représentation est défini comme suit :

— soient F_1 et F_2 les ensembles des attributs feuilles des profils respectifs $profil_1$ et $profil_2$ à comparer (*cf.* section 4.3.3.1) ;
— soit G l'ensemble des triplets du graphe sémantique de ces profils ;
— soit *recherche(*f*,* t_{elt_f}*,* t_{exp_f}*)* une méthode, décrite dans la section 4.3.4.1, qui recherche les types de données t_{elt_f} et t_{exp_f} qui sont associés au contenu d'un attribut feuille noté f. Notons que si un type n'est pas trouvé, sa valeur est alors vide ;
— soit $t_{f_1}\_t_{f_2}(v,\ v^{'})$ une méthode qui permet de convertir une valeur v de type t_{f_1} en une valeur $v^{'}$ de type t_{f_2} (*cf.* sections 4.2.1.3 et 4.3.4.2) ;
— et soient $f_1 \in F_1$ et $f_2 \in F_2$. Si f_1 et f_2 sont deux attributs feuilles nécessairement appariables, ces derniers peuvent être effectivement apparier, moyennant certaines transformations, si et seulement si : *recherche(*f_1*,* $t_{elt_{f_1}}$*,* $t_{exp_{f_1}}$*) et recherche(*f_2*,* $t_{elt_{f_2}}$*,* $t_{exp_{f_2}}$*)* renvoient des types de données compatibles.
De façon générale, les transformations à effectuer sur la représentation des contenus des attributs feuilles f_1 et f_2 à apparier, en fonction de la compatibilité des types des données de ces contenus, sont clairement identifiées comme suit :

1. on effectue un changement de combinaison linéaire de représentation par *extension des vecteurs de base initiaux* afin de représenter les attributs f_1 et f_2 à comparer dans la même dimension, si :

   (a) $t_{elt_{f_1}} = t_{elt_{f_2}}$

   (b) ou $\exists\ t_{elt_{f_1}}\_t_{elt_{f_2}}(v_{f_1},\ v^{'}_{f_1})$

   (c) ou $\exists\ t_{elt_{f_2}}\_t_{elt_{f_1}}(v_{f_2},\ v^{'}_{f_2})$

   Par exemple, si l'on souhaite comparer la langue d'un document représenté par l'élément de contenu *« (français, 1) »* aux préférences en langues d'un usager représentées par les éléments de contenu *« (anglais, 1), (français, 0.5), (espagnol, 0.25) »*, il faut étendre la représentation de la langue du document, de sorte qu'elle soit décrite dans le même espace de représentation que celui des

préférences de l'usager. Ainsi, la langue du document sera décrite par les éléments de contenu suivants : *« (anglais, 0), (français, 1), (espagnol, 0) »*. Le poids des nouveaux vecteurs (ou valeurs) est égal à 0.

2. on effectue un changement de combinaison linéaire de représentation par *changement des vecteurs de base initiaux* des attributs f_1 et f_2, si :
   (a) $t_{exp_{f_1}} = t_{exp_{f_2}}$
   (b) ou $\exists\ t_{elt_{f_1}}\_t_{exp_{f_2}}(v_{f_1}, v'_{f_1})$
   (c) ou $\exists\ t_{elt_{f_2}}\_t_{exp_{f_1}}(v_{f_2}, v'_{f_2})$
   (d) ou $\exists\ t_{exp_{f_1}}\_t_{exp_{f_2}}(v_{f_1}, v'_{f_1})$
   (e) ou $\exists\ t_{exp_{f_2}}\_t_{exp_{f_1}}(v_{f_2}, v'_{f_2})$

   Par exemple, dans la figure FIG. 4.2, il est nécessaire d'exprimer l'élément de contenu de l'attribut *DatePublicationInformation*, ici *« (12/02/2003, 1) »*, dans le même espace de représentation que celui des éléments de contenu de l'attribut *PréférencesDatesUtilisateur* qui sont : *« (moinsRécent, 0.5), (récent, 1) »*. Pour cet exemple, il faut changer les vecteurs de base initiaux car on a des éléments de contenu qui sont explicités par des expressions logiques. De plus, les types de données nécessitent que l'on transforme la valeur *12/02/2003* en la valeur *2003* par exécution de la méthode nommée *jjmmaaaa_ aaaa*. Cet exemple est illustré clairement dans la section suivante (*section 4.3.4.6*).

   Notons que dans les cas (a), (d) et (e), il va être nécessaire de gérer le problème de *recouvrement d'intervalles* afin de garantir un appariement correct. Il s'agira de déterminer les intervalles communs entre les expressions logiques des attributs à comparer. S'il n'existe pas d'intervalles communs alors le résultat de l'appariement entre ces attributs est nul.

4.3.4.6 Exemple de transformations de représentation de contenu

Soient les deux attributs feuilles de sémantique compatible suivants :

— *PréférencesDatesUtilisateur* dont le contenu est *{ (récent,1), (moinsRécent, 0.5) }*. Avec la valeur *récent* qui représente les années supérieures ou égales à 2003 (*>=2003*) et la valeur *moinsRécent* qui représente les années inférieures à 2003 et supérieures à 1999 (*<2003 ET >1999*) ;

— *DatePublicationArticle* dont le contenu est *(12/02/2003, 1)*.

Les résultats de l'analyse de contenu sont constitués des combinaisons linéaires de ces deux attributs relatives à leurs éléments de contenu ainsi qu'aux expressions logiques qui les explicitent.

— Pour *PréférencesDatesUtilisateur (noté a_{ut})*, on obtient :
$a_{ut_{exp}} = 1.\overrightarrow{\text{GE\_2003}} + 1.\overrightarrow{\text{LT\_2003\_ET\_GT\_1999}}$
$a_{ut_{elt}} = 1.\overrightarrow{\text{récent}} + 0.5.\overrightarrow{\text{moinsRécent}}$
— Pour *DatePublicationArticle (noté a_{inf})*, on obtient après conversion du type de données $a_{inf_{elt}} = 1.\overrightarrow{\text{EQ\_2003}}$ et après changements d'espaces de représentation on a :
$a_{inf_{exp}} = 1.\overrightarrow{\text{GE\_2003}} + 0.\overrightarrow{\text{LT\_2003\_ET\_GT\_1999}}$ (on procède ici à une extension des vecteurs de base)
$a_{inf_{elt}} = 1.\overrightarrow{\text{récent}} + 0.\overrightarrow{\text{moinsRécent}}$ (on procède ici à un changement des vecteurs de base)

Ainsi, ces deux attributs seront appariés en utilisant les vecteurs poids de leurs représentations, relatifs aux éléments de contenu $a_{ut_{elt}}$ et $a_{inf_{elt}}$. Pour cela, on peut appliquer, par exemple, la mesure du cosinus comme décrit dans le tableau Tab. 4.7.

expression de l'attribut a_{inf} dans la base *elt*		
base *elt*	**récent**(t_1)	**moinsRécent**(t_2)
base *exp*	**GE_2003** (v_1)	**LT_2003_ET_GT_1999** (v_2)
Vecteurs poids des expressions logiques après transformations		
$a_{ut_{exp}}$	1	1
$a_{inf_{exp}}$	1	0
Vecteurs poids des éléments de contenu après transformations		
$a_{ut_{elt}}$	1	0.5
$a_{inf_{elt}}$	1	0
Similarité	$sim(a_{ut_{elt}}, a_{inf_{elt}}) = 0.894$ (formule du cosinus)	

Table 4.7 – Changement d'espace de représentation et calcul de similarité

Dans la section suivante, nous illustrons un autre exemple de transformation de représentation de contenu dans le contexte d'un fournisseur de service qui souhaite s'adapter aux caractéristiques des dispositifs mobiles de ses usagers.

4.3.4.7 Contexte mobile : exemple de transformations de représentation de contenu

Soient les deux attributs feuilles de sémantique compatible suivants :
— *taille1_ écran_ fournisseur* dont le contenu est *(moyenne, 1)*. Avec la valeur *moyenne* qui représente la restriction aux tailles d'écrans supérieures ou égales à 2 pouces et inférieures à 4 pouces ;
— *taille_ écran_ mobile* dont le contenu est *(3, 1)* et où la valeur 3 est définie en pouces.

Les résultats de l'analyse de contenu donnent deux combinaisons linéaires pour ces deux attributs : l'une associée aux valeurs des éléments de contenu (noté a_{elt}) et l'autre associée aux valeurs des expressions logiques (noté a_{exp}). Ainsi :

— pour l'attribut *taille1_ écran_ fournisseur* (noté a_{fs}), on obtient :
$a_{fs_{exp}} = 1.\overrightarrow{\text{GE\_2\_ET\_LT\_4}}$
$a_{fs_{elt}} = 1.\overrightarrow{\text{moyenné}}$

— pour l'attribut *taille_ écran_ mobile* (noté a_m), on obtient $a_{m_{exp}} = 1.\overrightarrow{\text{GE\_2\_ET\_LT\_4}}$ après évaluation de la condition « >= 2 *ET* < 4 », et après changement d'espace de représentation on a :
$a_{m_{elt}} = 1.\overrightarrow{\text{moyenné}}$ (changement des vecteurs de base)
Notons que GE remplace le symbole >= et LT le symbole <.

Les deux attributs *taille1_ écran_ fournisseur* et *taille_ écran_ mobile* seront appariés en utilisant leur représentation relative aux éléments de contenu $a_{fs_{elt}}$ et $a_{m_{elt}}$. Ce type de comparaison va servir à vérifier les tailles de dispositifs mobiles et permettre ainsi au service de s'y adapter, notamment en terme de présentation de l'information.

En résumé, afin d'apparier deux attributs, il faut vérifier la cohérence de leur sémantique (sémantique de la structure logique et sémantique de contenu). Pour cela, nous avons besoin de définir des méthodes de transformation de représentation de contenu (conversion de types, changement de l'espace de représentation) pour les attributs de sémantique compatible.

Notons cependant que les requêtes de raisonnement des étapes 2 et 6 de l'algorithme d'analyse sémantique de profils, définies précédemment, décrivent principalement le squelette général de ces requêtes. Si les instances de profils et les ontologies globales s'enrichissent de nouveaux types de liens sémantiques, ces requêtes (ou règles) peuvent être modifiées afin de prendre en compte les nouveaux liens.

Par ailleurs, l'avantage d'utiliser une API java est qu'elle va nous permettre de combiner les aspects de programmation aux aspects de raisonnement liés au web sémantique. Dans les sections suivantes, nous allons décrire plus en détails les principes d'appariement proprement dits basés sur les espaces vectoriels de représentation des contenus, afin d'évaluer un degré de similarité entre profils.

4.4 Appariement de profils

Dans cette section, nous allons proposer une méthode d'appariement de profils basée sur la combinaison d'un ensemble d'appariements élémentaires, ou appariements d'attributs feuilles, pour l'accès à des ressources. L'appariement de profils est lié aux appariements des attributs feuilles qui décrivent ces profils car ce sont ces derniers qui sont associés à des éléments de contenu. La combinaison de certains de ces appariements va permettre de sélection-

ner les résultats qui correspondent à la tâche définie ou tout simplement de ré-ordonnancer ces résultats.

4.4.1 Appariement d'attributs feuilles de profils

L'appariement cohérent d'attributs feuilles de profils différents nécessite que ces attributs aient une sémantique compatible (*cf.* sections 4.3.3 et 4.3.4). Nous avons identifié différents types d'appariements d'attributs feuilles de profils :

— *appariement de type booléen* ;
— *appariement de type RI* ;

4.4.1.1 Appariement de type booléen

L'appariement de type booléen ou de type base de données est utilisé lorsque les deux attributs feuilles à apparier sont mono-valués. Un exemple de ce type d'appariement consisterait à comparer l'âge de la population cible d'un document à l'âge d'un usager. Le résultat de ce type d'appariement est binaire.

4.4.1.2 Appariement de type RI

Ce type d'appariement est utilisé lorsqu'au moins un des attributs feuilles à apparier est multi-valué. Dans ce cas, on représente les différents attributs feuilles à apparier dans un même espace vectoriel dont la dimension est donnée par la taille du vocabulaire (*cf.* section 4.3.4.5). A chaque vecteur de *valeurs* (communément appelé termes en RI), noté par exemple $d = (t_1, t_2, \ldots, t_n)$, est associé un vecteur de poids réel ou booléen, noté $p_d = (w_{d,t_1}, w_{d,t_2}, \ldots, w_{d,t_n})$, qui permettra de calculer un degré de similarité entre les attributs feuilles à comparer.

Comme exemples de ce type d'appariement, on peut citer :

— *la correspondance d'une information aux besoins d'un usager* : il s'agit d'une mesure de similarité entre le vecteur des poids des termes représentant les besoins de l'utilisateur (requête éventuellement reformulée ou centres d'intérêts) et celui des termes représentant le contenu d'une information (document, granule de document, collection de documents, etc.). Les poids, dans ce cas, sont généralement calculés avec les formules de *tf* (*cf.* formule 1.1) ou *tf.idf* (*cf.* formule 1.3) et la similarité avec la formule du cosinus (*cf.* formule 1.12) ;
— *la compatibilité d'une information aux préférences multi-valuées d'un utilisateur* : il s'agit de mesurer la similarité d'une information pour un attribut donné (langue, format, etc.) aux préférences de l'utilisateur pour cet attribut. Le tableau Tab. 4.8 illustre un exemple d'évaluation de cet appariement entre l'attribut langue d'un document et les préférences en langues d'un usager.

Vecteur de termes f	Anglais(t_1)	Français(t_2)	Espagnol(t_3)
Vecteur poids p_d, de la langue du document	$w_{d,t_1} = 1$	$w_{d,t_2} = 0$	$w_{d,t_3} = 0$
Vecteur poids p_u, des préférences en langues de l'usager	$w_{u,t_1} = 1$	$w_{u,t_2} = 0.5$	$w_{u,t_3} = 0.25$
Similarité	$sim(p_d, p_u) = 0.873$ (formule du cosinus)		

TABLE 4.8 – Exemple de calcul de similarité entre la langue d'un document et les préférences en langue d'un usager

— etc.

Une fois l'appariement au niveau des attributs feuilles effectué, le problème qui se pose est de pouvoir évaluer la similarité entre deux profils lorsque ces derniers sont décrits par plusieurs attributs feuilles. Nous proposons, pour cela, une méthode qui combine différents appariements élémentaires effectués sur les attributs feuilles de ces profils.

4.4.2 Appariements de profils

L'appariement de profils (respectivement d'attributs non feuilles de profils) est le résultat d'une combinaison d'appariements de couples d'attributs feuilles, de sémantique compatible, qui décrivent ces derniers.

4.4.2.1 Combinaison d'appariements

Pour combiner différents types d'appariements, ces derniers doivent tout d'abord être effectués séparément. Ainsi, chaque appariement (ou combinaison d'appariements) va représenter un facteur potentiel dans le processus de sélection ou d'ordonnancement des résultats obtenues pour une tâche donnée. On peut donc décrire les ressources (ou résultats) recherchées sous forme de listes de facteurs (ou appariements), notées $a = (f_1, f_2, \ldots, f_n)$, auxquelles vont correspondre des vecteurs d'appariements effectués entre des couples d'attributs feuilles de sémantique compatible. Ainsi, soit u et d des profils à apparier, le vecteur des résultats d'appariements entre ces profils est noté $p_{d,u} = (v_{d,u,f_1}, v_{d,u,f_2}, \ldots, v_{d,u,f_n})$.

Notons qu'une sous-liste de a peut-être utilisée pour la sélection (on la note a_s) et une autre sous-liste ou la même peut-être utilisée pour l'ordonnancement (on la note a_o) des résultats. Un exemple de liste de facteurs de sélection ou d'ordonnancement des résultats peut être : *correspondance d'une information aux besoins d'un usager (pertinence de documents, de granules ou de collections de documents), compatibilité d'une information aux préférences en langues d'un usager, etc.*

De plus, on va associer un vecteur de poids pour une liste de facteurs donnée a' (a' pouvant être a_s ou a_o). Ce vecteur de poids est noté $p_{a',x} = (w_{f_1}, w_{f_2}, \ldots, w_{f_m})$ et décrit le pouvoir discriminant (ou l'importance) des facteurs les uns par rapport aux autres. Ainsi, w_{f_j} est le poids ou l'importance du facteur f_j. Afin de calculer les valeurs des w_{f_j}, des ordres d'importance doivent être donnés pour tous les éléments du vecteur $p_{a',x}$. Considérons les ordres d'importance i, pour une liste de facteurs donnée, définis dans le tableau TAB. 4.9. Le facteur le plus important a pour ordre d'importance 1.

vecteur de facteurs a'	f_1	f_2	f_3	f_4	f_5	f_6	$\ldots$	f_n
Ordres d'importance i	1	1	2	3	3	4	$\ldots$	k
vecteur de poids des facteurs $p_{a',x}$	$w_{f_1} = \alpha_1$	$w_{f_2} = \alpha_1$	$w_{f_3} = \alpha_2$	$w_{f_4} = \alpha_3$	$w_{f_5} = \alpha_3$	$w_{f_6} = \alpha_4$	$\ldots$	$w_{f_n} = \alpha_k$

TABLE 4.9 – Ordres d'importance et poids des facteurs (ou appariements) de sélection ou d'ordonnancement des informations

La méthode de calcul des éléments du vecteur $p_{a',x}$ (poids des facteurs ou appariements) est donnée par :

$$\alpha_i = \beta \cdot \Sigma_{j>i} \alpha_j \tag{4.1}$$

où α_1 et β sont pré-définis et α_i représente le poids des facteurs d'ordre d'importance i (car plusieurs facteurs peuvent avoir le même ordre d'importance). Ainsi, si on a k ordres d'importance, on aura $(k-1)$ équations à $(k-1)$ inconnues à résoudre :

$$\begin{cases} \alpha_1 = \beta(\alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \cdots + \alpha_{k-1} + \alpha_k) \\ \alpha_2 = \beta(\alpha_3 + \alpha_4 + \alpha_5 + \cdots + \alpha_{k-1} + \alpha_k) \\ \alpha_3 = \beta(\alpha_4 + \alpha_5 + \cdots + \alpha_{k-1} + \alpha_k) \\ \cdots \\ \alpha_{k-2} = \beta(\alpha_{k-1} + \alpha_k) \\ \alpha_{k-1} = \beta\alpha_k \end{cases}$$

Pour déterminer les α_i, on peut utiliser la méthode du pivot de Gauss. Notons que si l'on souhaite que les $\alpha_i \in [0,1]$ alors α_1 doit être fixé à 1. De plus, $\beta > 1$ pour garantir des poids qui suivent l'ordre d'importance des différents appariements.

On peut donc calculer un *poids de sélection* p_s et/ou un *poids d'ordonnancement* p_o pour chaque ressource. Ces poids de sélection ou d'ordonnancement seront une fonction de $p_{d,u}$ (résultats des appariements élémentaires) et $p_{a',x}$ (poids des appariements élémentaires). Ils peuvent être évalués à

l'aide de la formule de la moyenne pondérée qui est définie, dans ce contexte, comme suit :

$$f(p_{d,u}, p_{a',x}) = \frac{\sum_j v_{d,u,f_j} \cdot w_{f_j}}{\sum_j w_{f_j}} \tag{4.2}$$

L'utilisation d'une moyenne pondérée, plutôt que celle d'une somme pondérée, permet de normaliser la valeur des poids de sélection ou d'ordonnancement dans l'intervalle des valeurs possibles pour les appariements v_{d,u,f_j}.

Pour la sélection des résultats, il peut être nécessaire de définir un seuil pour décider si le degré de correspondance entre deux profils est assez significatif. On peut utiliser, des méthodes basées sur la distribution des scores [AH01] [BTT04]. On peut également procéder comme suit : ordonner les résultats suivant les scores et choisir les n premiers résultats.

Le principe de combinaison d'appariements peut-être, pour des besoins de performance, légèrement différent selon que l'on veut comparer deux profils ou un profil avec une liste de profils.

4.4.2.2 Cas d'un profil et d'une liste de profils

Lorsqu'il s'agit de comparer un profil à une liste d'autres profils comme c'est souvent le cas en RI, la liste complète de tous les appariements possibles a est généralement déterminée en fonction de l'ensemble des attributs de tous les profils définis pour une application donnée. Les sous-listes a_s et a_o et les ordres d'importance de leurs facteurs ou appariements sont déterminés soit par les utilisateurs, soit par l'application qui les fixe pour ses différentes tâches.

De façon générale, l'algorithme du tableau TAB. 4.10 résume les étapes à suivre pour une application d'accès à des ressources.

`Algorithme général d'accès à des ressources`
1. Choix des profils à utiliser qui décrivent des éléments de l'architecture
2. Détermination des différents appariements en fonction de la liste des attributs de description des profils sélectionnés : $a = (f_1, f_2, \ldots, f_n)$
3. Effectuer les différents appariements : $p_{d,u} = (v_{d,u,f_1}, v_{d,u,f_2}, \ldots, v_{d,u,f_n})$
4. Combiner les différents appariements
4.1 Détermination de sous-listes pour la sélection (a_s) et/ou l'ordonnancement (a_o) des résultats à partir de la liste a
4.2 Détermination des ordres de préférences pour chaque liste
4.3 Détermination des poids des facteurs de chaque liste : $p_{a_s,x} = (w_{f_1}, w_{f_2}, \ldots, w_{f_l})$ et $p_{a_o,x} = (w_{f_1}, w_{f_2}, \ldots, w_{f_n})$
4.4 Calcul des poids de sélection p_s et des poids d'ordonnancement p_o
5. Restituer les résultats

TABLE 4.10 – Algorithme général pour un accès flexible à des ressources

Par ailleurs, étant donné que tous les attributs descriptifs d'un profil ne sont pas toujours renseignés ou que certains attributs ne sont pas définis pour tous les profils de l'architecture générale de l'application, il va se poser le problème de la gestion de ces valeurs vides ou non renseignées dans la restitution des résultats.

4.4.2.3 Discussion sur les valeurs non renseignées

En cas de valeur non renseignée (égale à null), deux choix s'offrent à nous :

— considérer la valeur null comme un zéro (ce qui correspond au pire des cas) et dans ce cas, le dénominateur des formules de poids de sélection ou d'ordonnancement (*cf. formule 4.2* de la section 4.4.2.1) reste le même ;

— ne tenir compte que des valeurs renseignées et dans ce cas, le dénominateur de la formule du poids ne prend pas en compte le poids des facteurs pour lesquels le résultat d'appariement n'existe pas (est égal à null).

Si p_p est le poids obtenu en remplaçant la valeur null par la valeur 0 et p_n le poids obtenu dans le cas contraire, on aura :

$$\begin{cases} \text{numérateur}(p_p) = \text{numérateur}(p_n) \\ et \\ \text{dénominateur}(p_p) > \text{dénominateur}(p_n) \end{cases}$$

On aura donc toujours $p_p < p_n$. Ceci signifie qu'en considérant les valeurs non renseignées, on augmente les chances de restituer un document dont certains attributs ne seraient pas définis. Pour l'exemple illustré dans la section suivante (*cf.* tableau TAB. 4.13), nous n'avons pas remplacé les valeurs vides par des zéros. De ce fait, si on a une valeur égale à null pour un facteur donné, le dénominateur de la formule du poids de sélection ou du poids d'ordonnancement ne prend pas en compte le poids de ce facteur.

4.4.2.4 Illustration de l'appariement de profils et de l'influence des valeurs non renseignées

Dans cette section, nous illustrons la méthode de combinaison d'appariements proposée au travers d'un exemple. Nous allons utiliser deux types de profils :

1. des profils de documents dont le contenu des attributs feuilles est représenté dans le tableau TAB. 4.11 ;
2. et des profils utilisateurs dont le contenu des attributs feuilles est représenté dans le tableau TAB. 4.12.

Profils de documents	(mots clés, poids)	Langue	Taille	Popularité
D1	(landscape, 0.73) ; (nature, 0.59) ; (flower, 0.62)	(anglais, 1)	(normal, 1)	(forte, 1)
D2	(paysage, 0.28) ; (architecture, 0.65)	(français, 1)	(court, 1)	(moy, 1)
D3	(paysage, 0.51) ; (nature, 0.72) ; (faune, 0.63)	(français, 1)	(long, 1)	(forte, 1)
D4	(paisaje, 0.32) ; (naturaleza, 0.23) ; (ciudad, 0.54)	(espagnol, 1)	(normal, 1)	(faible, 1)

TABLE 4.11 – Exemples de profils de documents

Profils utilisateurs	Centres d'intérêt	Préférences en Langue	Préférences en Taille	Préférences en Popularité
U1	(nature, 1) (paysage, 1)	(anglais, 1) ; (français, 0.5) ; (espagnol, 0.25)		
U2	(nature, 1) (paysage, 1)		(long, 0.25) ; (normal, 0.5) ; (court, 1)	

TABLE 4.12 – Exemples de profils utilisateurs

La détermination des couples d'attributs appariables va permettre de définir la liste de facteurs $a = (f_1, f_2, f_3, f_4)$. Les f_j représentent respectivement et par ordre de pouvoir discriminant décroissant : *la correspondance d'un document aux besoins ou centres d'intérêts d'un usager (ordre 1), la compatibilité d'un document aux préférences en langues d'un usager (ordre 1), la compatibilité d'un document aux préférences en tailles d'un usager (ordre 2), la popularité d'un document (ordre 3).* Pour chaque document et pour chaque utilisateur on aura un vecteur de résultats d'appariements $p_{d,u} = (v_{d,u,f_1}, v_{d,u,f_2}, v_{d,u,f_3}, v_{d,u,f_4})$. Dans cet exemple, le principe de restitution des informations consiste à sélectionner les documents selon les facteurs f_1 et f_2 et à ordonnancer les résultats sélectionnés selon tous les facteurs de a. La liste a est donc égale à la liste a_o. Dans cet exemple, les vecteurs $p_{a_s,x}$ (poids des facteurs de sélection) et $p_{a_o,x}$ (poids des facteurs d'ordonnancement) sont les mêmes pour tous les utilisateurs. Les éléments de $p_{a_s,x}$,

à savoir w_{f_1} et w_{f_2}, ont le même ordre d'importance et α_1 est fixé à 1.

Facteurs	f_1 formule du cosinus	f_2 produit scalaire	f_3 produit scalaire	f_4	$p_s = (\sum_j v_{d,u,f_j} \cdot w_{f_j}) / \sum_j w_{f_j},\ j < 3$ $p_o = (\sum_j v_{d,u,f_j} \cdot w_{f_j}) / \sum_j w_{f_j},\ j < 5$
$p_{a_o,x}$	$w_{f_1} = 1$	$w_{f_2} = 1$	$w_{f_3} = 0.333$	$w_{f_4} = 0.165$	
p_{d_1,u_1}	0,829745715	1	null	0.86	$p_s =$ **0,914872858** $p_o =$ 0,910690861
p_{d_2,u_1}	0,27974834	0.5	null	0.42	$p_s =$ **0,38987417** $p_o =$ 0,39217013
p_{d_3,u_1}	0,802226992	0.5	null	0.73	$p_s =$ **0,651113496** $p_o =$ 0,657125631
p_{d_4,u_1}	0,581758201	0.25	null	0.35	$p_s =$ **0,415879101** $p_o =$ 0,410858291
p_{d_1,u_2}	0,829745715	null	0.5	0.86	$p_s =$ **0,829745715** $p_o =$ 0,759776846
p_{d_2,u_2}	0,27974834	null	1	0.42	$p_s =$ **0,27974834** $p_o =$ 0,45530597
p_{d_3,u_2}	0,802226992	null	0.25	0.73	$p_s =$ **0,802226992** $p_o =$ 0,671513346
p_{d_4,u_2}	0,581758201	null	0.5	0.35	$p_s =$ **0,581758201** $p_o =$ 0,538056209

TABLE 4.13 – Calcul des poids de sélection et d'ordonnancement en tenant compte des valeurs non renseignées

Le calcul des éléments du vecteur $p_{a_o,x}$ (facteurs d'ordonnancement) se fait à partir du système d'équations suivant :

$$\begin{cases} \alpha_1 = \beta(\alpha_2 + \alpha_3) \\ \alpha_2 = \beta\alpha_3 \end{cases}$$

Avec $\alpha_1 = 1$ et $\beta = 2$, on obtient $\alpha_2 = 0.333$ et $\alpha_3 = 0.165$.

Ainsi, on a : $w_{f_1} = 1$, $w_{f_2} = 1$, $w_{f_3} = 0.333$ et $w_{f_4} = 0.165$.

Le calcul des poids de sélection (p_s) et des poids d'ordonnancement (p_o) des documents pour les utilisateurs U1 et U2 est décrit dans le tableau TAB. 4.13.

En conclusion, les résultats restitués aux différents utilisateurs, dans l'ordre, sont :

— l'utilisateur U1 reçoit les documents D1 et D3 ;

— l'utilisateur U2 reçoit les documents D1, D3 et D4.

Le document D4 est sélectionné pour l'utilisateur U2 parce que l'usage des valeurs égales à null a permis d'obtenir un poids de sélection plus important que si on avait remplacé ces valeurs vides par des zéros.

Par cet exemple, nous avons illustré le fait que la combinaison de différents appariements permet de restituer des réponses adaptées à chaque usager. Les deux utilisateurs ont le même besoin en information mais leurs ensembles de résultats sont différents. De plus, nous montrons que la prise en compte des valeurs vides permet d'augmenter la probabilité de restitution d'un document qui ne serait pas renseigné pour un attribut donné.

4.5 Conclusion

Dans ce chapitre, nous avons défini une méthode flexible d'analyse sémantique de profils pour l'accès à des ressources. Cette méthode va permettre de comparer des profils décrits différemment en analysant leur sémantique. Pour cela, elle détermine de façon automatique le nombre de couples d'attributs feuilles appariables entre ces profils. On peut ainsi élargir la base de profils exploitée par une application donnée à celles d'autres applications. La méthode d'analyse sémantique de profils proposée permet également une exploitation d'ontologies qui ne se limite pas à un simple usage de taxinomies mais également à l'usage de règles d'inférences permettant de déduire de nouvelles connaissances ou informations non explicitées sur les taxinomies. De plus, nous avons défini une méthode de combinaison d'appariements d'attributs feuilles de profils qui permet de mesurer un degré de similarité entre ces profils.

Dans le chapitre suivant, nous présentons les expérimentations que nous avons effectuées pour valider nos propositions. Le but est de montrer la faisabilité de ces dernières et les opportunités offertes par nos différentes contributions.

Chapitre 5

Expérimentations : construction et exploitation de profils pour l'accès à des ressources

5.1 Introduction

Le but de ce chapitre est de mener des expérimentations pour montrer la faisabilité de nos propositions, tant sur le plan de la modélisation de profils que sur celui de l'exploitation de ces derniers dans des applications. Ces expérimentations ont conduit à la mise en œuvre d'un outil nommé *SemanticProfile* pour la construction, la visualisation et l'analyse sémantique de profils. Cet outil peut être exploitable par différentes applications d'accès à des ressources. Nous avons également procédé à une évaluation des méthodes d'exploitation de profils proposées dans le chapitre précédent à savoir : l'analyse sémantique et l'appariement de profils. Pour cela, nous avons tout d'abord construit un cadre général d'expérimentation.

5.2 Cadre expérimental

Notre cadre expérimental est constitué des différents éléments de notre architecture générale d'exploitation de profils pour l'accès à des ressources. Les instances de profils construites décrivent des documents de la campagne d'évaluation CLEF 2001, ainsi que des profils d'utilisateurs des collections de cette campagne. Dans cette section, nous allons décrire en détail les différents éléments de l'architecture d'exploitation de profils (*cf.* section 4.2.1) relative à nos expérimentations : les espaces de noms, les ontologies, la librairie de règles de transformation de types, les instances de profils et l'index

du contenu des profils.

5.2.1 Description des espaces de noms utilisés

Nous avons utilisé 6 espaces de noms différents :

1. *l'espace de noms de RDF* qui est décrit à l'adresse « `http://www.w3.org/1999/02/22-rdf-syntax-ns#` » et qui est préfixé par « *rdf :* ». Cet espace de noms permet de décrire les différents éléments d'un triplet à travers des mots réservés (*cf.* section 2.3.3.1). Pour nos expérimentations nous avons fait usage, principalement, des mots réservés suivants :
   (a) *Description* pour décrire une ressource ;
   (b) *ID* pour définir un label ou littéral ;
   (c) *about* pour nommer une ressource associé à une URI ;
   (d) *type* pour définir une classe comme étant une spécialisation d'une autre.
2. *l'espace de noms de RDFS* qui est décrit à l'adresse « `http://www.w3.org/2000/01/rdf-scheme#` » et qui est préfixé par « *rdfs :* ». Cet espace de noms permet, principalement, de typer les différents éléments d'un triplet RDF (*cf.* section 2.3.3.2). Par exemple, pour spécifier que toutes les instances d'une classe B sont également des instances d'une classe A, on peut utiliser la propriété *subClassOf*. Ainsi, pour notre modèle générique on va définir cette propriété entre les classes *AttributFeuille et Attribut*, par exemple ;
3. *l'espace de noms de OWL* qui est décrit à l'adresse « `http://www.w3.org/2002/07/owl#` » et qui est préfixé par « *owl :* ». Cet espace de noms permet de définir, principalement, des contraintes entre classes ou ressources d'un modèle RDF (*cf.* section 2.3.3.3). Nous avons utilisé par exemple les contraintes suivantes :
   (a) *disjointWith* pour exprimer la disjonction entre classes (par exemple, entre les classes *AttributNonFeuille et AttributFeuille* du modèle générique) ;
   (b) *equivalenceClass* pour l'équivalence entre concepts ou types de données ;
   (c) *sameAs* pour l'égalité entre concepts ou types de données.
4. *l'espace de noms de XMLSchema* qui est décrit à l'adresse « `http://www.w3.org/2001/XMLSchema/` » et qui est préfixé par « *xsd :* ». Cet espace de noms permet de définir des types de données mais également d'en créer de nouveaux à partir de types existants (*cf.* section 2.3.2.4). Nous avons ré-utilisé, principalement, les types : *string (chaîne de caractères), integer (entier), float (réel), gYear (année)* ;

5. *l'espace de noms du Dublin Core* qui est décrit à l'adresse « `http://purl.org/dc/elements/1.1/` » et qui est préfixé par « *dc :* ». Il est constitué d'une liste de métadonnées pour la description de ressources (*cf.* section 2.3.3.4). Nous avons ré-utilisé, principalement, les concepts suivants : *title* (titre), *creator* (créateur d'une ressource), *description* (texte donnant une description d'une ressource), *subject* (mots-clés ou contenu), *language* (langue), *date*, *source* (origine d'une ressource) ;
6. *l'espace de noms nommé « SemanticProfile_NameSpace »* qui est l'espace de noms que nous avons défini relativement à nos propositions et qui est préfixé par « *sp :* ». Cet espace de noms décrit les classes, les associations et les classes d'associations du modèle générique proposé mais également des instances de ces classes et classes d'associations (*cf.* section 4.2.1.1). *SemanticProfile_NameSpace* peut être visualisé avec l'outil *SemanticProfile* et nous comptons également le publié sur Internet. Un extrait de cet espace de noms est illustré dans la figure Fig. 5.1.

   Chaque élément de l'espace de noms est défini avec le mot réservé `rdf:Description` et par une liste de propriétés provenant de différents espaces de noms. Cet espace de noms décrit environ 108 ressources différentes en utilisant environ 13 prédicats différents. Notons qu'une ressource est soit un sujet d'un triplet soit un objet d'un triplet pouvant être décrit par d'autres triplets.

Les différents espaces de noms présentés précédemment vont être exploités pour la définition d'ontologies globales et d'instances de profils de notre environnement d'appariement.

5.2.2 Description des ontologies globales

Nous avons construit 4 ontologies globales (*cf.* section 4.2.1.2) pour nos expérimentations qui permettent d'améliorer l'analyse sémantique de certaines instances de classes de notre modèle générique :

1. *Ressources.rdf* qui va décrire les relations sémantiques (instances de la classe *LienRessources* du modèle générique) qui existent entre instances de la classe *Ressource* de notre modèle générique. Ces instances sont subdivisées en trois classes principales : celles qui décrivent des utilisateurs, celles qui décrivent des informations et celles qui décrivent des dispositifs matériels. Le fichier *Ressources.rdf* contient 51 triplets et 30 ressources. Comme exemples d'instances de ressources définies, on peut citer : *sp : profil_information*, *sp : profil_article*, *sp : profil_article_LeMonde94*, *sp : profil_utilisateur*, *sp : profil_dispositif_matériel* ;

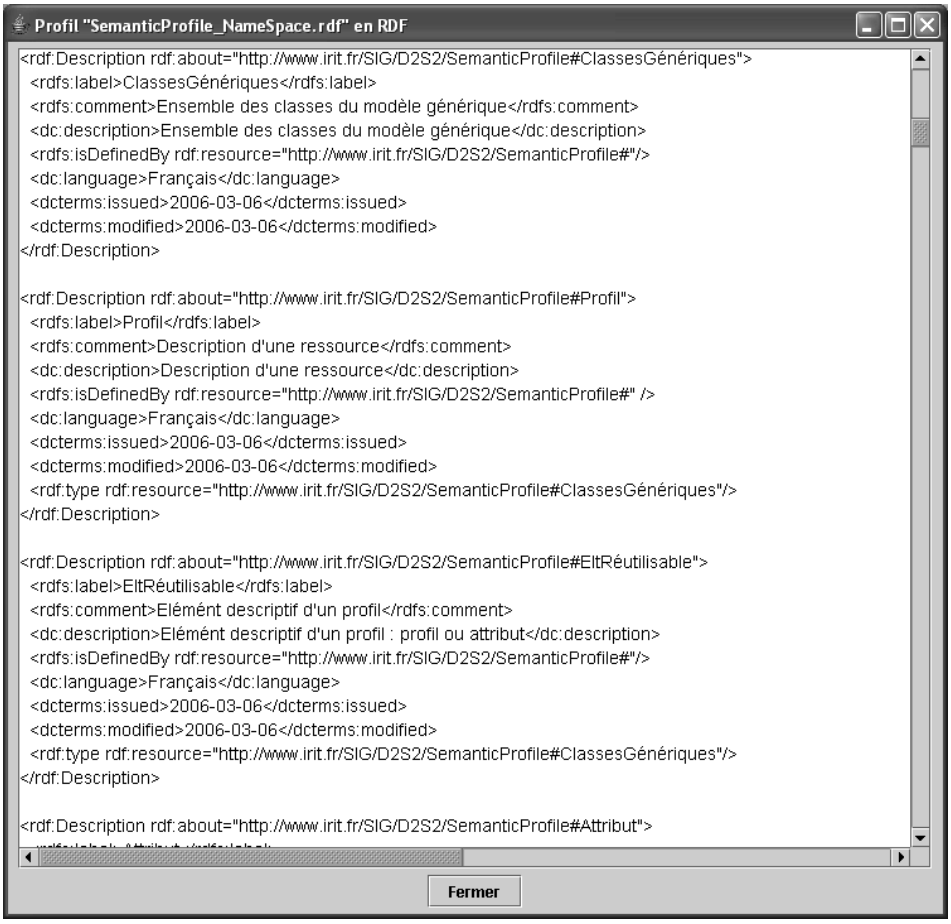

FIGURE 5.1 – Extrait de l'espace de noms *SemanticProfile_ NameSpace*

2. *Concepts.rdf* qui va décrire les relations sémantiques (instances de la classe *LienConcepts* du modèle générique) qui existent entre instances de la classe *Concept* de notre modèle générique. Le fichier *Concepts.rdf* contient 60 triplets et 32 ressources ou instances de concepts. Comme exemples d'instances de concepts définies, on peut citer : *dc : title*, *dc : subject*, *dc : language*, *sp : métadonnées*, *sp : taille* ;
3. *TypesDeValeurs.rdf* qui décrit les relations sémantiques (instances de la classe *LienTypesDeValeurs* du modèle générique) qui existent entre instances de la classe *TypeDeValeur* de notre modèle générique. Le fichier *TypesDeValeurs.rdf* contient 55 triplets et 32 ressources. Comme exemples d'instances de la classe *TypeDeValeur* définies, on peut citer : *xsd : string*, *xsd : float*, *sp : jjmmaaaa* ;
4. *ValeursContenu.rdf* qui va décrire les relations sémantiques entre valeurs de contenu. Ces valeurs correspondent aux propriétés *valeur et*

valeurDeComparaison des classes *EltDeContenu et ExpressionLogique* respectivement, de notre modèle générique. Ce fichier est constitué de 29 triplets et de 27 ressources ou classes de valeurs de contenu que nous avons défini. Ces *valeurs* décrivent : les langues (anglais, français, espagnol) avec leurs abréviations et leur traduction en anglais. Ce fichier n'étant pas suffisant pour gérer toutes les valeurs de contenu, nous avons également fait usage de la version RDF/OWL de Wordnet ;

5. *wordnet_ synset.rdf* qui est le fichier contenant la description RDF/OWL de Wordnet [AGS06]. Nous utilisons ce fichier pour compléter le fichier précédent *ValeursContenu.rdf* avec une terminologie éprouvée comme celle de *Wordnet* (*cf.* section 2.3.3.6).

De façon générale, la description d'une instance des classes *Ressource, Concept, TypeDeValeur* comprend le nom de l'instance, la description succincte de cette dernière et les liens existants avec d'autres instances de la même classe générique (Ressource, Concept ou TypeDeValeur). Les liens pour les instances de ces classes sont généralement des liens de : spécialisation, équivalence, égalité, partie de, etc.

Les ontologies *Ressources.rdf, Concepts.rdf, TypesDeValeurs.rdf et ValeursContenu.rdf* présentées précédemment, peuvent être visualisées avec l'outil *SemanticProfile*. La visualisation peut se faire au travers :

1. d'un fichier texte rdf qui décrit une ontologie (les classes et les relations qui existent entre elles)
2. d'un fichier texte qui décrit une ontologie sous forme de triplets préfixés ou non (espaces de noms représentés par leurs préfixes) ;
3. d'un dessin qui représente l'ontologie sous forme de graphe de nœuds et d'arcs étiquetés.

Notons que l'on peut également avoir des statistiques sur les différentes ontologies globales en terme de : nombre de triplets, nombre de ressources et nombre de prédicats différents contenus dans ces dernières. Le tableau Tab. 5.1 compare statistiquement ces différentes ontologies. Notons aussi que le contenu de ces ontologies globales peut évoluer.

5.2.3 Description de la librairie de règles de transformations de types

La librairie de règles de transformations de types est un package java que nous avons implémenté et qui contient des méthodes permettant de transformer une valeur d'un type *Type1* en une autre valeur de type *Type2*, par exemple. Le nom de la méthode est composé de la concaténation des noms des deux types de données séparés par le caractère de soulignement. Pour l'exemple précédent, le nom de la méthode serait *Type1_ Type2*. Notons

	Res-sources	Con-cepts	Types-DeValeurs	Valeurs-Contenu	wordnet_synset
nombre de triplets	51	60	55	29	1 721 882
nombre de ressources	30	32	32	27	464 783
nombre de prédicats	3	3	3	3	17

TABLE 5.1 – Comparaisons statistiques des ontologies globales

que, pour des raisons de programmation, les caractères spéciaux « : » et « / », des préfixes ou des adresses URI des espaces de noms de types de données, sont remplacés par le caractère « 0 ». Ainsi, chaque classe java de cette librairie contient une méthode *get* qui reçoit la valeur à transformer en paramètre et renvoie le résultat de la transformation. Ainsi, l'appel suivant *sp0jjmmaaaa_sp0aaaa.get("12/02/2003")* retourne la valeur *2003* qui est l'année de la date passée en paramètre.

Par ailleurs, nous avons défini des instances de profils à partir de la collection de la campagne CLEF2001. Pour cela, nous nous sommes également basés sur le vocabulaire des espaces de noms de notre environnement d'exploitation de profils (*cf.* section 4.2), afin de décrire la sémantique associée à la structure logique et au contenu d'un profil.

5.2.4 Description des instances de profils et index

Les instances de profils utilisées dans nos expérimentations sont subdivisées en deux catégories :

— celles relatives à l'espace des informations mises à disposition et qui, dans notre cas, décrivent les documents des collections de la campagne d'évaluation CLEF 2001 ;

— celles relatives à l'espace des usagers et qui, dans notre contexte, décrivent des utilisateurs à travers leurs données démographiques, leurs préférences et leurs centres d'intérêts.

5.2.4.1 Description des profils d'information

Les documents de la campagne d'évaluation CLEF 2001 sont répartis en 3 collections d'articles de journaux de l'année 1994 à savoir : Los Angeles Times 1994 (LATimes94), Le Monde 1994 (LeMonde94) et l'Agence Télégraphique Suisse 1994 (ATS94). Le tableau TAB. 5.2 compare ces différentes collections sur les critères de : taille mémoire, nombre de documents et langue de rédaction des articles de la collection.

	LATimes94	LeMonde94	ATS94
Taille mémoire	420 Mo	156 Mo	82.1 Mo
Nombre de documents	113 005	44 013	43 178
Langue de rédaction	Anglais	Français	Français

TABLE 5.2 – Collections de CLEF 2001 : LATimes94, LeMonde94, ATS94

Chacune de ces collections possède une DTD qui lui est propre et qui est différente des autres collections. On a donc des documents avec des structures logiques différentes, ce qui nous confère un cadre intéressant pour nos expérimentations. Le tableau TAB. 5.3 illustre un extrait des différences qui existent entre éléments de structure logique de nos différentes collections.

	LATimes94	LeMonde94	ATS94
Titre	–	TITLE	TI
Grandes lignes ou thèmes	HEADLINE	SUBJECTS	KW
Créateurs ou Auteurs	BYLINE	AUTHOR	AU
Date	DATE	DATE	DT
Paragraphe	P	TEXT	TX

TABLE 5.3 – Collections de CLEF 2001 : comparaison des éléments de structure logique

Pour décrire les profils de documents de ces collections, nous avons procédé comme suit :

1. *construction de sous-modèles de profils génériques* qui décrivent les 3 collections, à partir de leur DTD. Pour cela, nous définissons des profils décrivant de façon générale, la structure logique, le contenu et la sémantique des documents de chaque collection. Le principe de cette construction est le suivant :

   (a) *ajout des attributs « langue », « taille » et « source »* aux DTDs des différentes collections. La *langue* est celle utilisée pour rédiger les articles (par exemple, français ou anglais), la *taille* définit le nombre de termes d'indexation de chaque document et la *source* représente le nom de la collection à laquelle appartient un document ;

   (b) *définition de la sémantique des éléments de structure logique des DTDs modifiés pour les différentes collections.* Pour cela, nous avons analysé des instances de documents de ces collections ;

   (c) *définition d'un élément de contenu non renseigné* lié à chaque attribut feuille de chaque DTD modifiée. La valeur de l'élément de

contenu est fixée à la valeur *null* et le poids de cette valeur est de *1*. Il s'agit, par cette astuce, de permettre la définition de la sémantique des éléments de contenu de chaque collection ;

(d) *définition des types de données des éléments de contenu de chaque attribut feuille des DTDs modifiées.* Pour cela, nous avons analysé des instances de documents des différentes collections. Notons que les éléments de contenu de documents ne sont pas explicités par des expressions logiques ;

(e) *transformation des DTDs modifiées en profils* qui décrivent la structure logique, le contenu et la sémantique des documents des différentes collections. Ces modèles de profils définissent des instances de notre modèle générique et constituent ainsi des *sous-modèles de profils génériques* pour nos différentes collections.

2. *construction de profils décrivant chaque document des différentes collections.* Pour cela, nous procédons aux étapes suivantes :

   (a) *indexation du contenu des documents.* Nous utilisons, ici, le principe d'indexation classique de recherche d'information où les termes d'indexation ont été sélectionnés après suppression des mots outils ;

   (b) *construction des profils (structure logique, contenu et sémantique) de chaque document.* Pour cela, on utilise : les *3 sous-modèles de profils génériques* obtenus par modification des DTDs originelles des différentes collections et les documents pour déterminer le contenu de certains attributs comme la date. Notons que le contenu de l'attribut qui décrit les mots clés d'un document n'a pas été ajouté aux profils (cet attribut représente la liste des paragraphes du document). Ceci s'explique par le fait que leur transformation en RDF augmentait considérablement la taille des fichiers. Ainsi, pour rechercher le contenu lié aux mots clés d'un document, les résultats d'indexation sont utilisés en suivant le principe de RI classique.

Les profils de documents ainsi obtenus sont décrits statistiquement, d'après leur représentation RDF, dans le tableau Tab. 5.4. Ce tableau présente le nombre moyen de triplets, de ressources , de prédicats et de littéraux différents sur l'ensemble des documents de chaque collection. Notons qu'il faut exclure de ces calculs les éléments de contenu décrivant les mots clés des documents.

Par ailleurs, rappelons qu'une *ressource* peut être sujet ou objet d'un triplet tandis qu'un *littéral* est toujours objet d'un triplet. La spécificité d'un littéral est qu'il représente une valeur atomique (chaine de caractère, nombre) non décomposable par des triplets contrairement à une ressource.

	LATimes94	LeMonde94	ATS94
Nombre moyen de triplets	120,02	148,12	132,28
Nombre moyen de ressources	43,45	49,44	45,45
Nombre moyen de prédicats	8	8	8
Nombre moyen de littéraux	31,96	42,43	36,91

TABLE 5.4 – Statistiques des profils RDF des collections de CLEF 2001

Les profils de documents ainsi construits définissent des exemples de profils de l'espace des informations mises à disposition. Par ailleurs, nous avons également défini des profils relatifs à l'espace des usagers que nous décrivons dans la section suivante.

5.2.4.2 Description des profils d'utilisateurs

Nous avons défini 10 profils utilisateurs qui décrivent leurs centres d'intérêts, leurs préférences (en dates, langues ou tailles de documents) et leurs données démographiques (nom, sexe, profession). Les profils utilisateurs sont décrits par une structure logique, un contenu et une sémantique. Notons que les éléments de contenu des préférences utilisateurs sont généralement explicités par des expressions logiques. Statistiquement, les profils RDF de ces utilisateurs sont composés d'une moyenne de : 55.63 triplets, 27.63 ressources, 9.95 prédicats et 11.05 littéraux différents (*cf.* TAB. 5.5).

	Utilisateurs
Nombre moyen de triplets	55.63
Nombre moyen de ressources	27.63
Nombre moyen de prédicats	9.95
Nombre moyen de littéraux	11.05

TABLE 5.5 – Statistiques des profils RDF des utilisateurs

Notons que ces profils peuvent être visualisés avec l'outil *SemanticProfile*.

Par ailleurs, un index des instances de profils a été réalisé. Il concerne uniquement les éléments de contenu relatifs aux mots clés des documents des différentes collections.

Dans la section suivante, nous présentons l'outil d'aide à la construction, à la visualisation et à l'analyse sémantique de profils basés sur le modèle générique proposé.

5.3 SemanticProfile : outil de construction, de visualisation et d'analyse sémantique de profils

L'objectif de l'outil *SemanticProfile* est de mettre à la disposition de tout concepteur d'applications d'accès à des ressources, basées sur notre modèle générique de profil, un outil d'aide pour la construction, la visualisation et l'évaluation de l'interopérabilité des profils définis. Cette évaluation a pour objectif de montrer que ces profils sont bel et bien exploitables car pouvant être appariés avec d'autres profils.

5.3.1 Construction de profils

La construction d'un profil va consister à définir sa structure logique, son contenu et sa sémantique. Au terme de cette définition, le profil doit être transformé en fichier RDF. Le principe de construction d'un profil se subdivise donc en différentes étapes :

1. *définition de la structure logique et de la sémantique associée* ;
2. *définition du contenu et de la sémantique associée* ;
3. *transformation du profil ainsi décrit en fichier RDF.*

Afin de mener à bien ces différentes étapes, un système de numérotation de *nœuds importants* d'un profil, a été mis sur pied. Les *nœuds importants* qui ont été identifiés, pour un profil donné, sont :

1. la racine de la structure logique ;
2. les différents éléments de la structure logique (profils et attributs) ;
3. les différents éléments de contenu ;
4. et l'expression logique qui explicite un élément de contenu.

Pour chacun de ces *nœuds importants*, nous définissons une liste d'informations séparées les unes des autres par le caractère « § ». Ces données sont :

1. *le numéro du nœud* qui identifie de façon unique le nœud dans la liste. Ces numéros permettent :
   (a) d'identifier le nœud racine de la structure logique ;
   (b) de définir des liens de type *père-fils* entre instances des classes *Profil, Attribut et EltDeContenu* ;
   (c) et de définir un lien de type *estExplicitéPar* entre une instance de la classe *EltDeContenu* et une instance de la classe *ExpressionLogique*. Notons que le lien de type *estExplicitéPar* est traité comme le lien de type *père-fils* à la seule différence que l'on n'aura qu'un et un seul fils de type *expression logique* pour un élément de contenu donné.

Pour attribuer ces numéros, nous avons défini un système de numérotation dont le principe est le suivant :

(a) fixer une borne maximale des *nœuds importants*. Par exemple : 10, 100, 1000, 10 000, etc. ;

(b) affecter le numéro « 0 » au nœud racine de la structure logique du profil ;

(c) déterminer le numéro des autres *nœuds importants* (forcément nœud fils d'une autre nœud donné) comme suit :

i. si la borne maximale des *nœuds importants* est égale à 10 alors les fils du nœud racine, par exemple, auront pour numéros : 01, 02, 03, etc. De même, les fils du nœud 01 auront pour numéros : 011, 012, 013, etc. ;

ii. si la borne maximale des *nœuds importants* est égale à 100 alors les fils du nœud racine, par exemple, auront pour numéros : 001, 002, 003, etc. De même, les fils du nœud 001 auront pour numéros : 00101, 00102, 00103, etc. ;

iii. si la borne maximale des *nœuds importants* est égale à 1000 alors les fils du nœud racine, par exemple, auront pour numéros : 0001, 0002, 0003, etc. De même, les fils du nœud 0001 auront pour numéros : 0001001, 0001002, 0001003, etc. ;

Notons que pour nos expérimentations, la borne maximale des *nœuds importants* pour les documents des collections a été fixée à 10 000 et celles des profils utilisateurs à 1000.

2. *la valeur du nœud* qui est l'instance de la classe représentant ce nœud. Ainsi, pour un profil ou un attribut on utilise son nom (par exemple : préférencesLangues, préférencesDates, etc.). Pour un élément de contenu, on utilise le couple *valeur-poids* séparé par le caractère « ø » (par exemple : anglaisø1.0, françaisø0.5, etc.).
   Notons cependant une différence pour les expressions logiques qui elles ont pour valeur l'expression logique complète reconstituée en exploitant éventuellement la classe *OpérateurLogique*. Par exemple, soit l'élément de contenu *(moinsRécent, 0.5)* explicité par une expression logique décrivant des années inférieures à 2003 et supérieures à 1999. Avec notre modèle générique, cette expression logique s'écrit avec deux instances de la classe *ExpressionLogique* et une instance de la classe *OpérateurLogique*. Les instances de la classe *ExpressionLogique* sont des couples *opérateurDeComparaison-valeurDeComparaison* séparés par le caractère « ø » (par exemple : <ø2003, >ø1999). Les instances de la classe *OpérateurLogique* (ET, OU) sont séparés des instances des instances de la classe *ExpressionLogique* avec le caractère « ? ». On obtient donc pour l'expression logique associée à l'élément de contenu *(moinsRécent, 0.5)*, la valeur : *<ø2003 ?ET ?>ø1999* ;

3. *la classe générique à laquelle appartient le nœud* qui peut être : Profil, AttributNonFeuille, AttributFeuille, EltDeContenu et ExpressionLogique ;
4. *la sémantique du nœud* qui est une instance d'une des classes suivantes : Ressource, Concept et TypeDeValeur.

Par la suite, la liste des *nœuds importants* est trié sur le numéro des nœuds afin que chaque *nœuds importants* soit suivi par la liste de ses nœuds fils triés par ordre croissant des numéros de nœuds. Un exemple de profil décrit par sa liste triée de *nœuds importants* est illustré dans le tableau Tab. 5.6. Notons que le nombre maximum (ou borne maximale) de nœuds importants, dans cet exemple, est de 10.

```
0§profil_utilisateur_z§sp:Profil§sp:profil_utilisateur
01§centresIntérêts§sp:AttributNonFeuille§dc:subject
011§musique§sp:AttributFeuille§dc:subject
0111§jazzø1.0§sp:EltDeContenu§xsd:string
012§cinéma§sp:AttributFeuille§dc:subject
0121§comédieø1.0§sp:EltDeContenu§xsd:string
013§sport§sp:AttributFeuille§dc:subject
0131§footballø1.0§sp:EltDeContenu§xsd:string
02§préférencesLangues§sp:AttributFeuille§sp:langue
021§anglaisø1.0§sp:EltDeContenu§xsd:string
022§françaisø0.5§sp:EltDeContenu§xsd:string
03§préférencesDates§sp:AttributFeuille§dc:date
031§récentø1.0§sp:EltDeContenu§sp:restrictionAnnées
0311§>=ø2003§sp:ExpressionLogique§sp:aaaa
032§moinsRécentø0.5§sp:EltDeContenu§sp:restrictionAnnées
0321§<ø2003?ET?>ø1999§sp:ExpressionLogique§sp:aaaa
```

Table 5.6 – Exemple de profil décrit par sa liste triée de *nœuds importants*

Le profil décrit par une liste *triée* de *nœuds importants* est alors analysé et transformé en fichier RDF. Ces nœuds importants sont identifiés de façon unique par *leur nom et par leur xpath*, car on peut avoir plusieurs nœuds de nom identique dans un profil. Par simplification, seul le nom apparaît dans les illustrations. De plus, le parsing du schéma xml d'une classe ou catégorie de profils, en utilisant Sax ou DOM parser, peut aider à la création de fichiers XML et RDF de profils.

Le traitement de fichiers RDF (création, lecture, interrogation, etc.) est rendu possible grâce à l'usage de l'API (Application Programming Interface) java *Jena* pour le développement d'applications basées sur le web sémantique. Il fournit : un environnement de programmation pour RDF/RDFS/OWL, le langage d'interrogation SPARQL et un moteur d'inférence. Jena est dispo-

nible à l'adresse suivante : `http://jena.sourceforge.net/`.

Ainsi, la lecture des fichiers RDF de profils va permettre de les visualiser sous différentes formes : textuelle (document RDF ou triplets RDF) ou graphique (dessin du graphe du profil).

5.3.2 Visualisation textuelle de profils

La visualisation textuelle d'un profil va se faire principalement sous deux formes : le document RDF ou la liste des triplets décrivant le profil.

5.3.2.1 Document RDF

Le document RDF d'un profil respecte la syntaxe RDF. Une URI est affectée à toutes les ressources ainsi qu'à tous les prédicats décrivant le profil. Notons que certaines ressources ne possèdent pas d'uri explicite, il s'agit de ressources dont l'uri est laissée comme relative. Dans ce cas, cette uri est définie à partir de l'uri du fichier courant dans lequel les ressources sont décrites. Le nom de ces ressources est précédé du symbole #.

Par conséquent, seuls les littéraux (valeurs atomiques des objets de triplets) ne possèdent pas d'URI. Ceci, s'explique par le fait que les littéraux sont des éléments atomiques donc non décomposables sous forme de triplets. Un littéral va être par exemple : la valeur d'un élément de contenu, le poids de l'élément de contenu, etc.

La figure Fig. 5.2 illustre un extrait d'un exemple de document RDF. Notons que dans ce document les littéraux sont encadrés par des balises comme : *<sp : valeur>anglais</sp : valeur>*, *<sp : poids>1.0</sp : poids>*, *etc.*

Les descriptions de ressources, quant à elles, sont définies par une balise nommée *rdf:Description*. Les ressources peuvent être décrites par une liste de balises éventuellement imbriquées dans la balise *rdf:Description* comme suit : *<rdf : Description rdf : about="uri+nomRessource">listes de balises</rdf : Description>*.

5.3.2.2 Triplets RDF

Pour afficher les triplets RDF d'un document RDF donné, il faut utiliser des méthodes disponibles dans les packages de l'API Jena comme suit :

1. créer un modèle vide, en exécutant l'instruction : *Model model = ModelFactory.createDefaultModel()* ;
2. ouvrir le document RDF en lecture et lire son contenu dans le modèle crée précédemment, en utilisant la suite d'instructions java suivante : *BufferedReader frdf = new BufferedReader (new FileReader(Fich)) ; model.read(frdf, "") ;*

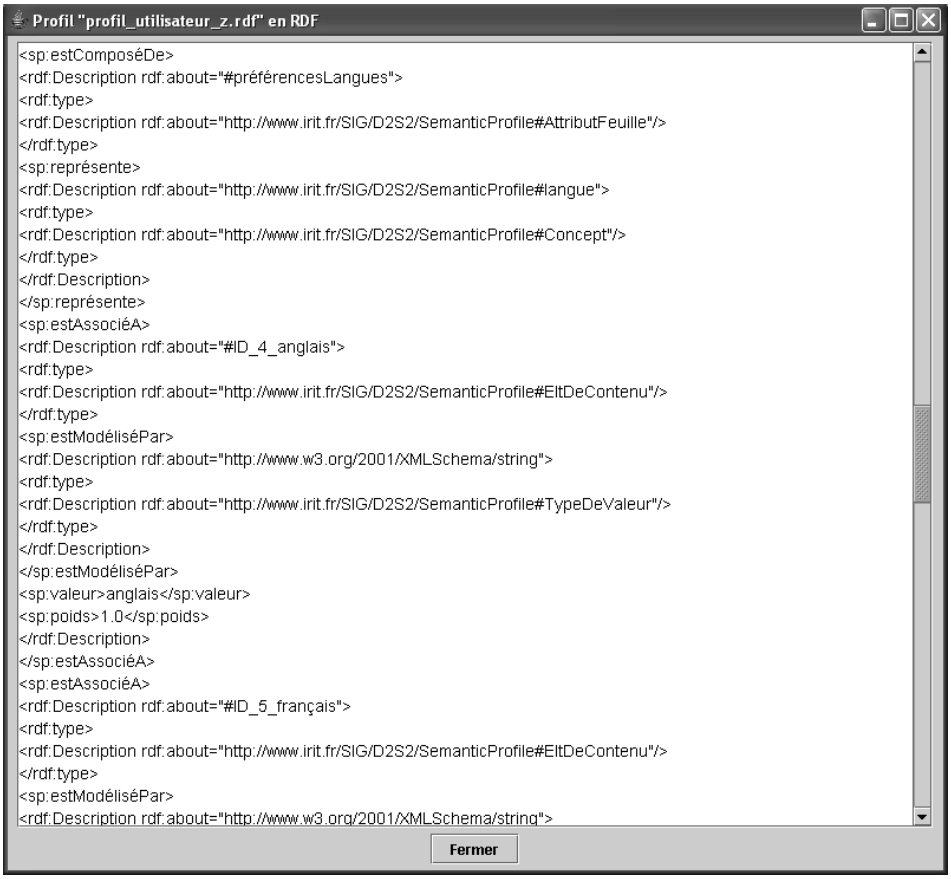

FIGURE 5.2 – Extrait d'un exemple de document RDF d'un profil

frdf.close() ;
Où *Fich* désigne le nom du fichier RDF à lire ;

3. établir la liste de tous les triplets du fichier RDF en exécutant l'instruction : *StmtIterator iter = model.listStatements()* ;
4. déterminer et afficher les différents éléments de chaque triplet.

La figure FIG. 5.3 illustre un extrait de la liste des triplets d'un document RDF. Dans cette figure, les adresses URIs des ressources sont remplacées par les préfixes correspondants. Cependant, la visualisation de profils sous forme de triplets RDF peut se faire également en affichant les adresses URIs complètes à la place des préfixes. Les deux options sont proposées par notre outil.

Par ailleurs, la liste des triplets est classée par groupes de triplets de sujets identiques. Par exemple, les triplets *[#préférencesLangues, sp : estAssociéA, #ID_4_ anglais]* et *[#préférencesLangues, sp:estAssociéA, #ID_ 5_ français]* sont placés côte à côte.

Notons aussi qu'une ressource peut être sujet ou objet d'un triplet. C'est le cas par exemple de la ressource nommée *#ID_4_anglais* dans les triplets :

— *[#préférencesLangues, sp : estAssociéA, #ID_4_anglais]* qui est le premier triplet de la figure FIG. 5.3 ;

— et *[#ID_4_anglais, sp : valeur, "anglais"]* qui est le dernier triplet de la figure FIG. 5.3.

Par contre, un littéral est toujours l'objet d'un triplet. C'est le cas par exemple du littéral *"anglais"* dans le triplet *[#ID_4_anglais, sp : valeur, "anglais"]* de la figure FIG. 5.3. Notons que les littéraux sont facilement identifiables dans la liste des triplets car leur valeur apparaît entre guillemets.

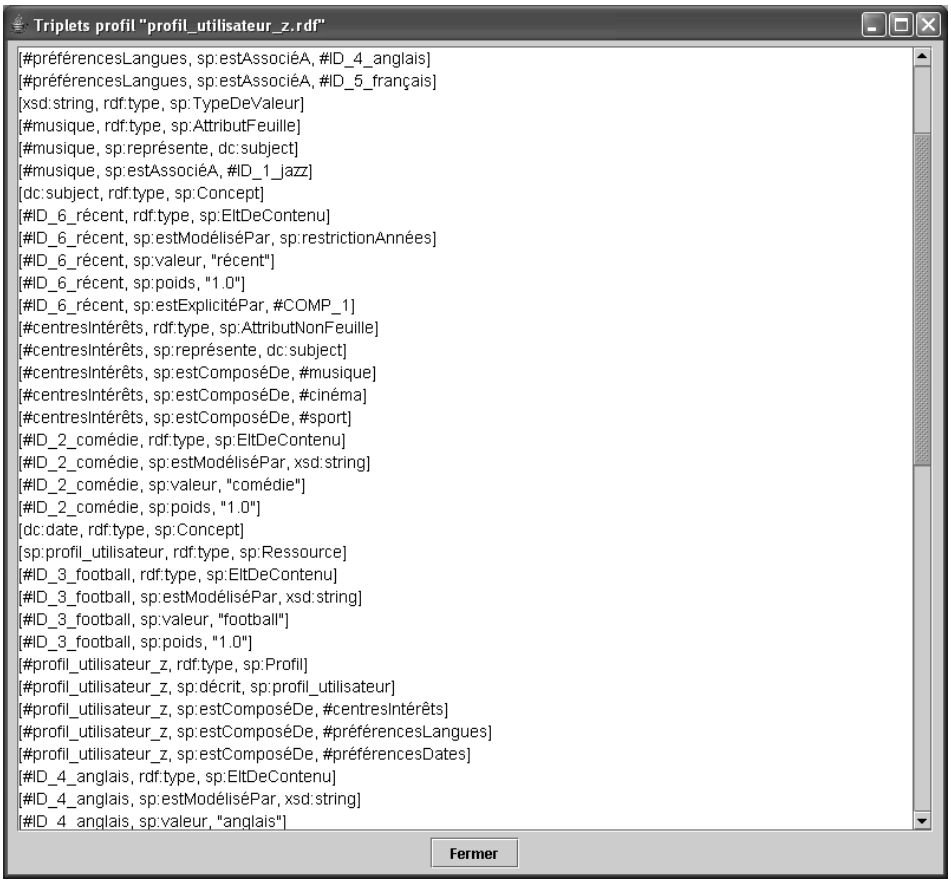

FIGURE 5.3 – Extrait d'une liste de triplets d'un document RDF décrivant un profil

Notons également que l'outil *SemanticProfile* permet de visualiser, sous forme textuelle, les ontologies globales crées, l'espace de noms *SemanticProfile_NameSpace* défini et les documents des collections de CLEF transformés en fichiers RDF et utilisés pour nos expérimentations.

Par ailleurs, la lecture du fichier RDF d'un profil ou de l'ensemble de ses triplets, pour en comprendre la structure et le contenu, peut être très difficile. On peut donc avoir besoin d'une visualisation graphique qui affiche le dessin du graphe ou le dessin d'une partie du graphe d'un profil.

5.3.3 Visualisation graphique de profils

Afin de proposer un outil de visualisation graphique de profils, nous avons utilisé le package java *Jung*, car il nous a fournit un ensemble de possibilités intéressantes de présentation de graphes :

- choix des formes des nœuds ;
- choix des couleurs des nœuds ;
- choix des noms des étiquettes d'arcs et de nœuds ;
- choix des formes et de l'orientation des arcs ;
- possibilité d'usage d'une fonction de *zoom* sur le dessin du graphe ;
- non obligation de définir au préalable la position des nœuds. Le système est capable de proposer une première disposition des nœuds et des arcs que l'on va pouvoir modifier. Il est donc possible d'interagir avec le graphe pour en modifier l'organisation ;

Avec ce package, notre outil va permettre de visualiser le graphe d'un profil ainsi que le graphe de deux profils qui donne une idée sur l'interopérabilité qui existe entre ces derniers.

5.3.3.1 Graphe d'un profil

La visualisation graphique d'un profil nous permet de visualiser tout ou partie de ce profil. La figure FIG. 5.4 illustre un exemple de graphe d'un profil obtenu avec l'outil *SemanticProfile* en utilisant le package Jung.

Le dessin de graphe obtenu possède, dans tous les cas, les caractéristiques suivantes :

- le nœud racine de la structure logique a la forme d'une étoile :
- les nœuds relatifs à la structure logique, exceptés le nœud racine, ont la forme d'ellipses ;
- les nœuds relatifs au contenu ont la forme de carrés
- les nœuds relatifs à la sémantique de la structure logique ont la forme de triangles ;
- les nœuds relatifs à la sémantique de contenu ont la forme de polygones à 5 côtés ;

Notons cependant que les cases à cocher permettent d'annuler ou de rétablir les différentes mises en forme suivantes :

- affichage ou non des *étiquettes des nœuds* ;
- affichage ou non des *étiquettes des arcs* ;
- annulation ou rétablissement des *formes des nœuds*. En cas d'annulation (la case *Formes des nœuds* est décochée), tous les nœuds ont

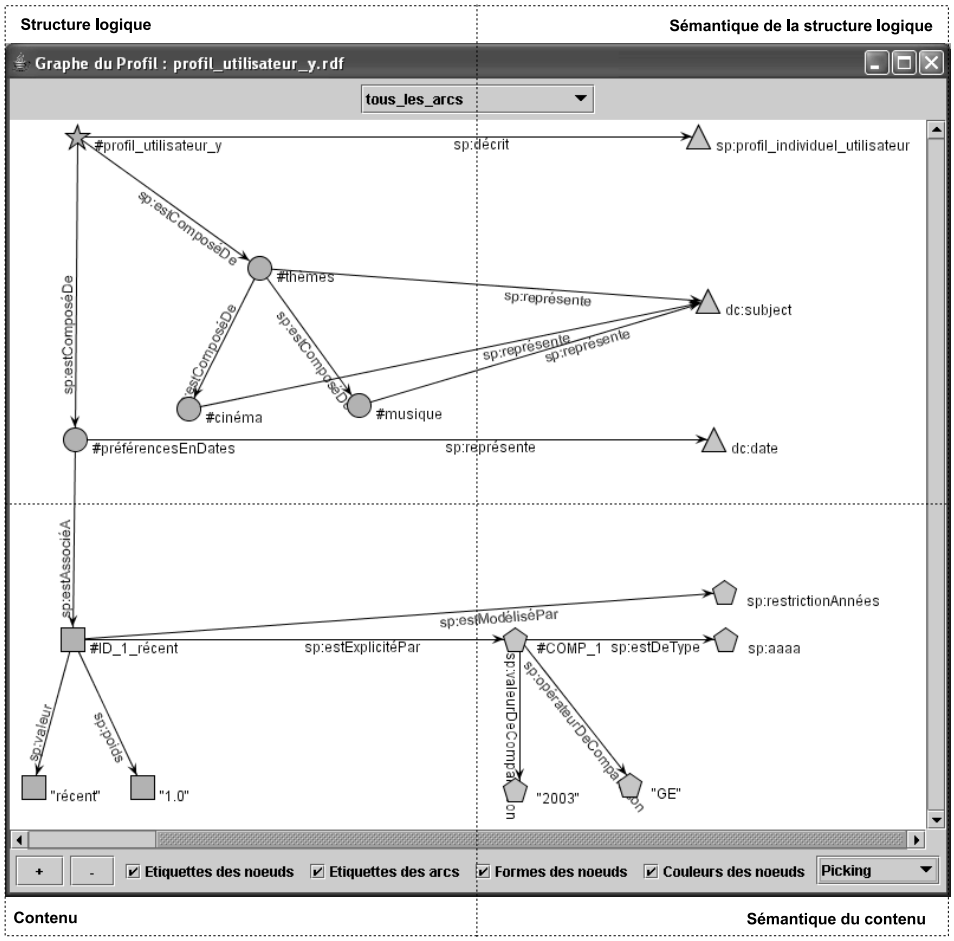

FIGURE 5.4 – Exemple de graphe d'un profil obtenu avec l'outil *Semantic-Profile*

alors la forme d'ellipses ;

— annulation ou rétablissement des *couleurs des nœuds*. En cas d'annulation (la case *Couleurs des nœuds* est décochée), tous les nœuds ont alors la couleur vert fluo ;

Notons que les boutons nommés « + » et « - » permettent d'effectuer des zooms sur le dessin. De plus, il existe une première liste de choix dans laquelle apparaît le terme *tous_ les_ arcs* qui permet de selectionner un nœud du graphe pour n'afficher que les arcs incidents ou sortants de ce nœud. Si l'on souhaite ré-afficher tous les arcs de tous les nœuds, il faut choisir dans la liste le terme *tous_ les_ arcs*. Il existe également une deuxième liste de choix qui contient deux valeurs : *Picking* et *Transforming*. La valeur *Picking*, lorsqu'elle est sélectionnée, permet d'interagir sur chaque nœud du dessin

pour le déplacer tandis que la valeur *Transforming* permet d'interagir sur l'ensemble du dessin afin de le déplacer.

Par ailleurs, l'uri de certaines ressources est laissée comme relative (présence du symbole # uniquement) dans le dessin du graphe de profil pour ne pas le surcharger. Cependant, la base des uris de ces ressources est définie par le chemin d'accès au fichier dans lequel est décrit le graphe de profil. Ce fichier porte généralement le nom du nœud racine de la structure logique de ce profil. La base de l'uri de ces ressources est complétée par le chemin d'accès à ces ressources dans le graphe de profil. Ce chemin d'accès est sous la forme d'une hiérarchie car les uris des ressources laissées comme relatives sont celles des éléments de structure logique, de contenu ou d'expressions logiques et que la structure générale de ces éléments est toujours sous la forme d'un arbre, d'après notre modèle générique de profil.

Ainsi, si on considère l'élément de structure logique nommé *sport* dont le chemin dans le graphe du profil *profil_ a_ ut* est *« profil_ a_ ut/centresIntérêts-/sport »*, son uri peut avoir la forme suivante :

file :///C :/eclipse/workspace/Sem_ profiler/Profils_ all_ rdf/profil_ a_ ut-/centresIntérêts/sport

où *file :///C :/eclipse/workspace/Sem_ profiler/Profils_ all_ rdf/* est le chemin d'accès au fichier qui décrit le profil *profil_ a_ ut*

Notons aussi que :

1. les éléments de contenu sont numérotés pour un profil donné. Ainsi, s'il y a n éléments de contenu dans un profil donné, ils seront numérotés comme suit : *ID_ 1_ valeurEltDeContenu, ID_ 2_ valeurEltDeContenu, ..., ID_ n_ valeurEltDeContenu* ;
2. les expressions logiques sont également numérotées pour un profil donné. Ainsi, s'il y a n expressions logiques dans un profil donné, elles seront numérotées comme suit : *COMP_ 1, COMP_ 2, ..., COMP_ n.*

Chaque élément de contenu ou expression logique est donc identifié de façon unique au sein d'un même profil à cause de sa numérotation. Par ailleurs, les éléments de sémantique eux ont toujours une uri explicite qui référence l'espace de noms auquel ils sont associés.

La visualisation graphique de profils nous offre également la possibilité de visualiser le graphe de deux profils et permet ainsi de se faire une première idée sur l'interopérabilité qui existe entre eux.

5.3.3.2 Interopérabilité de profils : graphe de deux profils

La figure Fig. 5.5 illustre un exemple d'interopérabilité de profils obtenu avec l'outil *SemanticProfile* en utilisant le package Jung. Ce type de visualisation permet de visualiser surtout les similitudes sémantiques qui existent

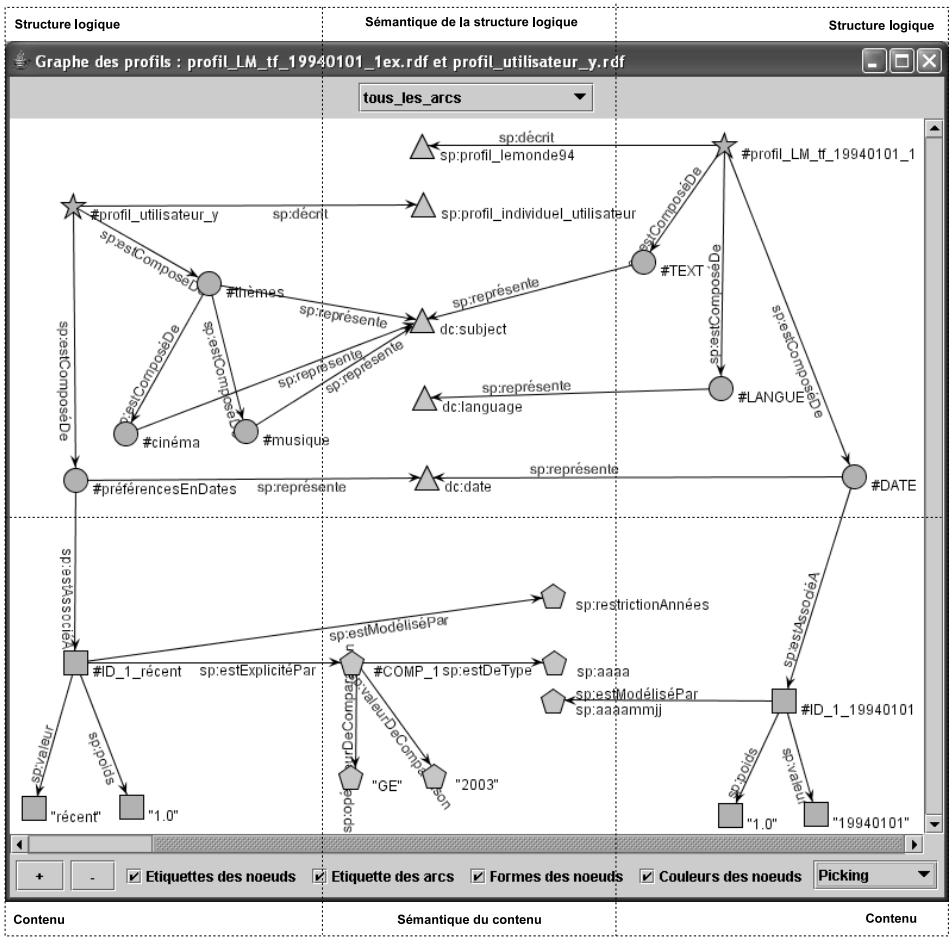

FIGURE 5.5 – Exemple d'interopérabilité de graphe d'un profil obtenu avec l'outil *SemanticProfile*

entre deux profils. Par exemple, dans la figure FIG. 5.5, on voit que les attributs *PréférencesEnDates* et *DATE*, des profils respectifs *profil_ utilisateur_ y* et *profil_ LM_ tf_ 19940101_ 1*, représentent le même concept `dc:date`. De même, les attributs *thèmes, cinéma et musique* du profil *profil_ utilisateur_ y* et l'attribut *TEXT* du profil *profil_ LM_ tf_ 19940101_ 1* représentent le même concept nommé *dc :subject*. Ainsi, ces attributs peuvent donc être éventuellement appariés.

La visualisation de l'interopérabilité de profils permet de se rendre compte des attributs qui pourraient être appariés entre deux profils. Cependant, ce constat doit être effectivement vérifié au travers d'une analyse sémantique plus approfondie des profils. Il faut analyser à la fois la sémantique de la structure logique ainsi que celle du contenu et il peut être nécessaire, dans

les deux types d'analyse, de procéder à des inférences. Nous abordons cet aspect dans la section suivante.

5.3.4 Analyse sémantique de profils : détermination de la liste d'attributs appariables

Les résultats de l'analyse sémantique de deux profils résultent de l'exécution de l'algorithme décrit dans la section 4.3.2. Cette analyse permet de détecter automatiquement la liste de couples d'attributs feuilles appariables entre ces profils. Cette liste comprend pour chaque couple :

1. le nom de chaque attribut du couple ;
2. le contenu de chaque attribut sous forme vectorielle, et éventuellement modifié après des transformations de types ou d'espace de représentation ;
3. des explications précisant les opérations qui ont conduit à la sélection d'un couple donné. Ces explications concernent : la vérification de concepts, la vérification de types de données et les transformations effectuées (types, espaces vectorielles).

La figure FIG. 5.6 illustre un exemple de résultat d'analyse de la sémantique de deux profils avant transformations. La présentation de ce résultat est subdivisée en trois parties :

1. *des généralités* qui présentent le nombre d'attributs feuilles de chaque profil ainsi que le nombre de couples d'attributs feuilles appariables entre eux. Par exemple, dans la figure FIG. 5.6, il y a deux couples d'attributs appariables, tandis les profils *LM_tf_19940101_1b_app.rdf* et *profil_utilisateur_z_app.rdf* à apparier, ont respectivement 9 et 5 attributs feuilles. Ceci peut s'expliquer par des problèmes d'incompatibilité de concepts ou de types, ou par le fait qu'il y a des attributs feuilles dont le contenu n'est pas renseigné ;
2. *les détails sur les différents couples appariables* qui décrivent pour chaque attribut de chaque couple :
   (a) le concept qui y est associé ;
   (b) le type des éléments de contenu de cet attribut ;
   (c) le type des expressions logiques, si elles existent, des éléments de contenu de cet attribut.

   Lorsque l'un de ces éléments n'est pas renseigné, il est remplacé par le caractère « $ ». Notons que chaque couple est numéroté et que la description d'un couple donné est toujours précédé de son numéro. Ainsi, dans la figure FIG. 5.6, le couple *numéro 1* est défini comme suit, pour les profils *LM_tf_19940101_1b_app.rdf et profil_utilisateur_z_app.rdf* respectivement :

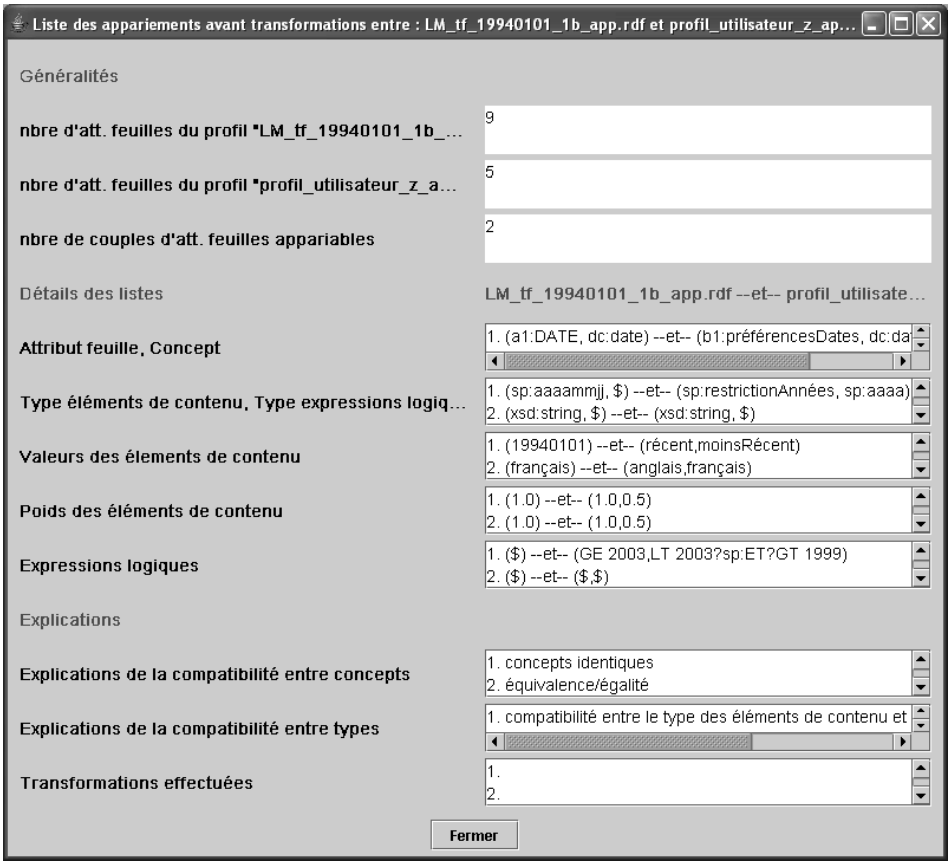

FIGURE 5.6 – Illustration des résultats d'analyse sémantique de profils avant transformations

(a) *Attribut feuille, Concept* : (a1 : DATE, dc : date) –et– (b1 : préférencesDates, dc : date).
Ce qui signifie que l'attribut feuille *a1 : DATE* du profil *LM_ tf_ 19940101_ 1b_ app.rdf* est associé au concept *dc : date* et peut être apparier avec l'attribut feuille *b1 : préférencesDates* du profil *profil_ utilisateur_ z_ app.rdf* qui représente également le concept *dc : date* ;

(b) *Type éléments de contenu, Type expressions logiques* : (sp : aaaammjj, $) –et– (sp : restrictionsAnnées, sp : aaaa).
Ce qui signifie que les éléments de contenu l'attribut *a1 : DATE* sont associés au type de données *sp : aaaammjj* et qu'ils ne sont pas explicités par des expressions logiques. De même, les éléments de contenu de l'attribut *b1 : préférencesDates* sont modélisés avec le type de données *sp : restrictionsAnnées* tandis que les expressions

logiques qui les explicitent sont du type de données *sp : aaaa* ;

(c) *Valeurs des éléments de contenu* : (19940101) –et– (récent, moinsRécent).

Ce qui signifie que l'attribut *a1 : DATE* a un seul élément de contenu qui a pour valeur *19940101*. Par contre, l'attribut *b1 : préférencesDates* a deux éléments de contenu dont les valeurs sont : *récent et moinsRécent* ;

(d) *Poids des éléments de contenu* : (1.0) –et– (1.0,0.5).

Ce qui signifie que l'attribut *a1 : DATE* a un seul élément de contenu qui a pour poids *1.0*. De même, l'attribut *b1 : préférencesDates* a deux éléments de contenu dont les poids sont : *1.0 pour la valeur « récent » et 0.5 pour la valeur « moinsRécent »* ;

(e) *Expressions logiques* : ($) –et– (GE 2003, LT 2003 ?sp : ET ?GT 1999).

Ce qui signifie qu'il n'y a pas d'expressions logiques associées aux éléments de contenu de l'attribut *a1 : DATE*. Par contre, pour l'attribut *b1 : préférencesDates* on a l'expression logique *« GE 2003 »* qui est associée au premier élément de contenu dont la valeur est *récent* tandis que l'expression logique *« LT 2003 ?sp : ET ?GT 1999 »* est associée au second élément de contenu dont la valeur est *moinsRécent* ;

3. *des explications* qui donnent des éléments des renseignements sur des inférences ou sur des transformations qui ont été effectuées.

Les inférences concernent la vérification de compatibilité entre concepts ou types de données. Ainsi lorsqu'il n'y a pas eu d'inférence sur la vérification de concepts, le texte *concepts identiques* s'affiche. De même, lorsque les types de données sont les mêmes le texte *types identiques* s'affiche. Dans le cas contraire, une explication est affichée. Dans le cas d'une inférence de concepts compatibles, il s'agit soit d'une équivalence, soit d'une égalité entre concepts. Le texte *équivalence/égalité* s'affiche alors. Dans le cas d'une compatibilité de types, le texte précise entre quels types de données (types de données des éléments de contenu, type de données des expressions logiques) la compatibilité a été détectée. Par exemple, dans la figure FIG. 5.6, on a le texte *compatibilité entre le type des éléments de contenu (sous-entendu de l'attribut du profil 1, ici LM_tf_19940101_1b_app.rdf) et le type des expressions logiques (sous-entendu de l'attribut du profil 2, ici profil_utilisateur_z_app.rdf)*.

Par contre, les explications concernant les transformations n'apparaissent que si l'on choisit d'afficher les couples appariables après transformations comme l'illustre la figure FIG. 5.7.

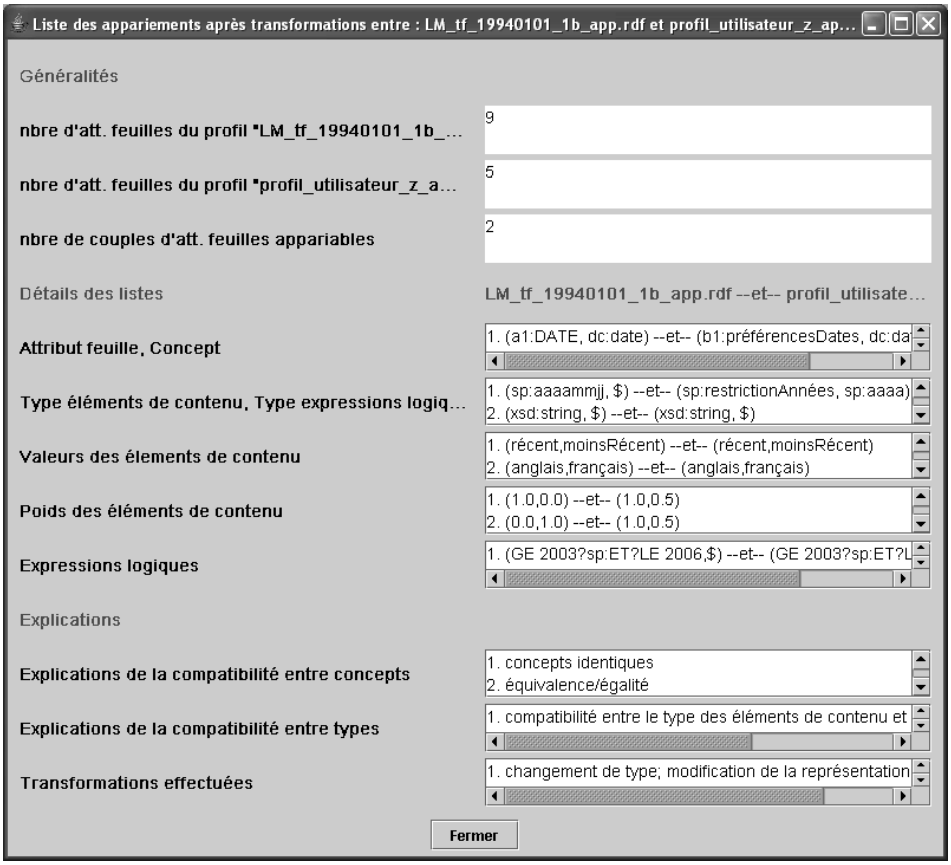

FIGURE 5.7 – Illustration des résultats d'analyse sémantique de profils après transformations

> Notons que cet aspect explicatif est important car il permet de juger manuellement de la fiabilité des résultats obtenus.

La figure FIG. 5.7 illustre un exemple de résultat d'analyse de la sémantique de deux profils après transformations. Les explications relatives aux transformations y apparaissent. Elles concernent des modifications de types de valeurs ainsi que des modifications portant sur la représentation vectorielle du contenu (ajout, suppression ou remplacement d'élément).

L'outil *SemanticProfile* offre de nombreuses fonctionnalités pour l'aide à la définition de profils pour l'accès flexible à des ressources. L'intérêt de cet outil est qu'il permet de procéder à une première évaluation de l'interopérabilité entre profils que l'on souhaite définir. Par la suite, ces profils doivent être introduits dans des applications concrètes. Notre outil permet déjà d'entrevoir les possibilités d'interactions entre profils différents.

Nous avons présenté toutes les fonctionnalités offertes par notre outil *SemanticProfile* et nous allons maintenant discuter, plus en détails, de l'éva-

luation des méthodes d'exploitation de profils proposées pour l'accès à des ressources, à savoir : l'analyse sémantique et l'appariement de profils.

5.4 Évaluation des méthodes d'analyse sémantique et d'appariement de profils

L'algorithme d'analyse sémantique de profils proposés permet de définir le dégré d'interopérabilité qui existe entre deux profils donnés. Ce degré d'interopérabilité est défini par le nombre de couples d'attributs feuilles appariables qui existent entre ces profils.

5.4.1 Algorithme d'analyse sémantique de profils pour la détermination d'éléments appariables

Nous avons évalué l'algorithme d'analyse de profils proposé par rapport à des méthodes d'analyse classiques qui seraient basées sur un modèle de profils de type *attribut-valeur* ou de type *structure logique*. Pour cela, nous avons utilisé les 10 profils utilisateurs définis dans la *section 5.2.4.2*, qui décrivent leurs centres d'intérêts, leurs préférences (en dates, langues ou tailles de documents) et leurs données démographiques (nom, sexe, profession). Ces profils utilisateurs sont décrits par une structure logique, un contenu et une sémantique. Nous avons également utilisé des documents de la campagne d'évaluation CLEF 2001 qui sont constitués d'articles de journaux de l'année 1994 : *Agence Télégraphique Suisse (ATS), Le Monde (LeMonde) et Los Angeles Times (LaTimes)*. La description de ces articles (structure logique, contenu et sémantique) est faîte dans la *section 5.2.4.1*.

Les expérimentations menées ont consisté à détecter le nombre moyen de couples d'attributs feuilles de sémantique compatible entre toute paire possible de profils définis pour nos expérimentations. Pour cela, nous avons appliqué notre algorithme (nommé SemanticProfile). Ensuite, nous avons considéré *le cas où les profils seraient décrits avec le modèle attribut-valeur*, c'est-à-dire par les listes d'attributs feuilles des profils définis où les attributs de sémantique similaire ont le même nom. Enfin, nous avons considéré *le cas où les profils seraient décrits avec un modèle basé sur une structure logique*, c'est-à-dire décrits uniquement par les structures logiques des profils définis où les attributs de sémantique similaire ont le même nom mais ont également le même chemin d'accès dans la structure logique. Le tableau Tab. 5.7 illustre les résultats obtenus pour les différentes méthodes [CSDT06b] [CSDT06a].

Dans ce contexte hétérogène, on remarque que notre algorithme permet de détecter plus d'appariements élémentaires entre profils. La sémantique ajoutée agit comme une partie partagée entre profils et permet ainsi, grâce à des inférences, de détecter des couples d'attributs qui n'ont pas le même nom mais qui partagent la même sémantique, ce qui n'est pas possible avec

Modèles de profils	Semantic-Profile	Attribut-valeur	Structure logique
Nombre moyen de couples de sémantique compatible	4.84	0.92	0.79

TABLE 5.7 – Résultats de détection automatique d'attributs de sémantique compatible

les *modèles de type attribut-valeur ou structure logique*. Par ailleurs, les attributs qui portent le même nom n'ont pas forcément la même sémantique. Ainsi, grâce à la double analyse de la sémantique de la structure logique et de la sémantique du contenu, notre approche garantit des résultats fiables. Notre méthode permet donc, dans un contexte hétérogène, de *réduire le silence informationnel et les incohérences* qui sont généralement inhérentes aux méthodes classiques de représentation et d'exploitation de profils.

Intérêts de l'analyse de la sémantique pour l'accès à des ressources

L'analyse de la sémantique de profils permet de comparer des modèles de ressources différents. Grâce à cette comparaison, on peut décider si une instance de ressource donnée est intéressante par rapport à une tâche prédéfinie. Cette analyse va permettre d'accéder à davantage de ressources décrites différemment, contrairement aux approches classiques.

Par ailleurs, l'analyse de la sémantique va également fournir une alternative à l'interrogation basée sur des éléments de structure logique (générique ou spécifique) qui impose de connaître les structures logiques des ressources qui nous intéressent. Ceci est fort difficile dans le contexte hétérogène actuel et il est donc important de pouvoir évoluer vers des méthodes qui autorisent une connaissance approximative des structures logiques (exploitation de modèles génériques, de chemins flous), voire des méthodes qui ne nécessitent aucune connaissance de ces structures logiques.

Si l'on formule par exemple la requête suivante : *« Liste des articles publiés en 1994 et qui traitent de la coupe du monde de football »*. Avec les approches classiques, il faudrait connaître le nom des balises qui décrivent des dates ainsi que celles qui décrivent le contenu d'un article, voire les chemins ou du moins une partie des chemins qui mènent à ces balises. D'après le tableau TAB. 5.3, qui décrit les éléments de structure logique des 3 collections de la campagne CLEF 2001, on se rend compte de l'hétérogénéité des balises utilisées pour la description de structures logiques. Il faudrait donc connaître toutes ces balises pour assurer une bonne couverture de la base.

Avec une analyse sémantique basée sur des métadonnées, il suffirait de connaître la structure logique d'une collection donnée pour retrouver les éléments de structure logique des autres collections qui ont une sémantique

compatible. On peut également utiliser des éléments sémantiques ou métadonnées pour exécuter cette requête. Ainsi, les dates pourront être représentées par la métadonnée *dc : date* et le contenu des articles par la métadonnée *dc : subject*, par exemple. L'analyse de la sémantique permettrait de retrouver les différents éléments de structure logique correspondants pour chaque collection, comme dans le tableau TAB. 5.8.

Ainsi, l'analyse sémantique permet de garantir une couverture maximale des collections décrites par des DTDs différentes et ceci que l'on procède à une interrogation à partir de métadonnées ou à une interrogation à partir d'éléments de structure logique d'une collection donnée. Une réflexion analogue peut-être menée dans le contexte de l'intégration de schémas de bases de données différentes.

Métadonnées	**Éléments de structure logique**		
	`LATimes94`	`LeMonde94`	`ATS94`
`dc:date`	DATE	DATE	DT
`dc:subject`	HEADLINE, P	SUBJECTS, TITLE, TEXT	KW, TI, TX

TABLE 5.8 – Collections de CLEF 2001 : correspondance entre métadonnées et éléments de structure logique

Par ailleurs, nous avons également procédé à l'évaluation de la méthode d'appariement de profils par combinaisons d'appariements élémentaires d'attributs feuilles de sémantique compatible. Cette évaluation est décrite dans la section suivante.

5.4.2 Méthode de combinaison d'appariements entre profils

L'appariement de profils a pu être testé sur une application de recherche personnalisée d'information. Pour cela, nous avons utilisé les collections de la campagne d'évaluation CLEF 2001. Le principe de cette expérimentation va consister à exécuter les 50 requêtes de la collection CLEF 2001 pour 3 des profils utilisateurs définis dans notre cadre expérimental. Il s'agit de rechercher des documents qui correspondent aux profils des utilisateurs lorsqu'ils formulent ces requêtes. Ainsi, pour les 3 utilisateurs nous avons exécuté 150 requêtes.

Analyse des profils utilisés

Les attributs feuilles des profils d'articles des collections de CLEF 2001 ne sont pas forcément tous renseignés et ne portent pas toujours le même nom. Pour des attributs représentant des concepts compatibles, le contenu n'est pas toujours représenté de la même façon d'une collection à une autre.

La définition d'une sémantique (de la structure logique et du contenu) et son analyse, afin d'identifier les attributs appariables et effectuer correctement les différents appariements, prend donc toute son importance.

Le tableau TAB. 5.9, présente la description des 3 profils d'utilisateurs qui vont formuler chacun les 50 requêtes de la campagne CLEF 2001. Ces profils utilisateurs ainsi que les profils de la campagne CLEF 2001 vont être utilisés pour effectuer une évaluation de notre méthode de combinaison d'appariement. Le but est de mesurer la divergence ou la convergence de nos résultats par rapport aux résultats qui seraient obtenus en se basant uniquement sur les requêtes formulées et le contenu des documents (sujet des documents). Cette analyse va permettre également de juger de *la qualité* de notre méthode d'appariement.

Attributs	**Contenu**		
	`Utilisateur1`	`Utilisateur2`	`Utilisateur3`
`Centres d'intérêt`	requête	requête	requête
`Préférences-EnLangues`	(anglais, 1), (français, 0.5)	(anglais, 0.5), (français, 1)	(anglais, 0), (français, 1)
`Préférences-EnTailles`	(court, 1), (normal, 0.5)	(court, 0.5), (normal, 1)	(court, 1), (normal, 0)
`Préférences-EnDates`	(semestre1, 0), (semestre2, 1)	(semestre1, 1), (semestre2, 0)	(semestre1, 1), (semestre2, 1)

TABLE 5.9 – Profils des 3 utilisateurs définis pour l'évaluation de la combinaison d'appariement

Notons que des restrictions sont définies pour les préférences en tailles d'articles et en dates d'articles, pour les différents usagers. De plus, les préférences en dates concernent la même année pour les différents utilisateurs. L'année choisie est 1994 pour être en conformité avec les documents de notre collection.

Ainsi, un article court est un article dont le nombre de termes d'indexation est inférieur ou égal à 50, dans le cas contraire il s'agit d'un article de taille normale. Par ailleurs, un article publié au semestre 1 est un article dont le mois de publication est inférieur ou égal au mois de Juin, dans le cas contraire il d'agit d'un article publié au semestre 2. Ceci peut être intéressant pour des usagers qui cherchent à se documenter sur des faits d'actualités plus ou moins récent d'une année donnée.

Pour effectuer l'appariement en vue de la restitution des résultats, nous avons procédé, dans un premier temps, à la détermination de la liste des appariements élémentaires (couples d'attributs appariables) entre profils d'utilisateurs et profils d'articles de journaux de la campagne CLEF 2001.

5.4.2.1 Détermination de la liste des appariements élémentaires

Avec la description des articles de journaux et celle des utilisateurs, nous avons identifié une liste d'appariements. Cette liste, avec les ordres d'importance considérés pour la combinaison d'appariements, est présentée dans le tableau TAB. 5.10.

Appariements	`sujet de l'article`	`langue`	`taille`	`date`	`titre`	thème collection
Importance	1	2	3	4	5	6

TABLE 5.10 – Ordres d'importance des facteurs ou appariements élémentaires

Chacun de ces appariements représente un couple d'attributs qui décrivent un article et un usager. Ainsi :

1. le *sujet de l'article* décrit l'appariement entre les centres d'intérêt d'un usager (ici une requête) et le contenu des paragraphes d'un article ;
2. la *langue* décrit l'appariement entre les préférences en langues d'un usager et la langue dans laquelle est rédigée l'article ;
3. la *taille* décrit l'appariement entre les préférences en tailles d'articles d'un utilisateur et la taille d'un article donné ;
4. la *date* décrit l'appariement qui existe entre les préférences en dates d'un utilisateur et la date de publication d'un article donné ;
5. le *titre* décrit l'appariement entre les centres d'intérêt de l'usager et le titre du document ;
6. le *thème de la collection* décrit l'appariement entre les centres d'intérêt d'un usager le contenu de la collection d'un article donné.

Les appariements identifiés vont nous permettre d'évaluer l'évolution des mesures de rappel et de précision classiques, lorsque l'on combine des appariements. Cette analyse va nous permettre de déduire le comportement général de notre méthode d'appariement.

5.4.2.2 Résultats de la combinaison d'appariements

Les figures FIG. 5.8, 5.9 et 5.10 présentent une étude comparative des 30 premiers documents obtenus pour nos différents utilisateurs, sur les 50 requêtes de la campagne CLEF 2001. Sur l'axe des abscisses, les étiquettes représentent respectivement les appariements ou combinaisons d'appariements suivants :

1. sujet de l'article (F1) ;
2. sujet de l'article et langue (F12) ;
3. sujet de l'article, langue et taille (F123) ;

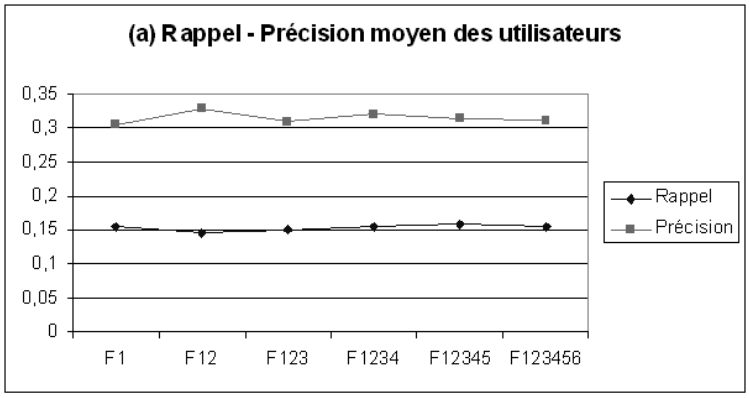

FIGURE 5.8 – Moyenne du rappel-précision moyen des utilisateurs (sur les 30 premiers résultats) pour les 50 requêtes de la collection CLEF 2001

4. sujet de l'article, langue, taille et date (F1234) ;
5. sujet de l'article, langue, taille, date et titre (F12345) ;
6. sujet de l'article, langue, taille, date, titre et thème de la collection à laquelle appartient l'article (F123456).

Les différentes figures FIG. 5.8, 5.9 et 5.10, représentent respectivement les courbes de la moyenne :

1. du rappel-précision moyen des utilisateurs ;
2. du rappel-précision par utilisateur ;
3. des moyennes de variation de rang (sur les 30 premiers résultats) pour les 50 requêtes de la collection CLEF 2001. La variation de rang est la différence entre la position d'un document dans l'ensemble des 30 premiers résultats de l'appariement F1 et sa position dans l'ensemble des 30 premiers résultats d'un autre appariement (F12, F123, etc.), s'il y est toujours présent.

Ces figures décrivent les résultats de recherches des différents utilisateurs avec les poids des différents appariements respectant les ordres d'importance du tableau TAB. 5.10. Ces ordres d'importance ont été choisis différents les uns des autres, pour mieux visualiser l'impact de la prise en compte d'un nouvel appariement élémentaire dans une combinaison d'appariements.

Plus particulièrement, ces courbes nous permettent également de visualiser la perte ou le gain en rappel et précision (*cf.* figures FIG. 5.8 et 5.9), du fait de la prise en compte de différents appariements. Ainsi, les appariements relatifs au contenu des articles dans la liste des appariements identifiés sont : *sujet de l'article, titre, thème*. Les autres appariements permettent juste d'adapter les résultats aux préférences des usagers, mais ne sont pas liés

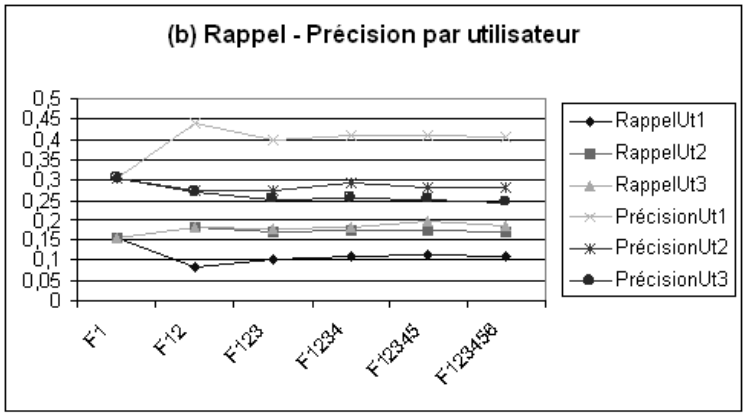

FIGURE 5.9 – Moyenne du rappel-précision par utilisateur (sur les 30 premiers résultats) pour les 50 requêtes de la collection CLEF 2001

à la requête qu'ils ont formulée. Les courbes obtenues vont permettre également de visualiser les variations d'ordonnancement des résultats, du fait de la prise en compte de différents appariements (*cf.* figure FIG. 5.10).

Dans la section suivante, nous analysons ces différents résultats et nous déduisons le comportement général de notre méthode d'appariement.

5.4.2.3 Analyse des résultats

Les résultats obtenus avec nos trois utilisateurs montrent que la combinaison de différents appariements dans l'ensemble des 30 premiers résultats fait varier les mesures de rappel et de précision, calculés relativement à l'appariement F1. Cependant, cette combinaison d'appariements agit surtout sur l'ordonnancement des résultats.

Les figures FIG. 5.8 et 5.9 montrent une augmentation ou une diminution aléatoire du rappel ou de la précision. Cependant, on peut dire que même s'il arrive de perdre en rappel ou en précision (car la courbe de précision n'est pas toujours croissante), cette perte est généralement faible. On constate que les différences de rappels par rapport au rappel obtenu avec l'appariement F1 varient entre $-7,4E-03$ et $5,7E-03$, tandis que les différences de précision varient entre $6,4E-03$ et $2,3E-02$.

Notons que les mesures de rappel et précision sont inadéquates dans notre contexte pour évaluer la qualité des résultats car tous les appariements ne sont pas relatifs au contenu des articles mais font intervenir d'autres caractéristiques de ces derniers comme : la date, la taille et la langue. Cependant l'intérêt de l'observation de l'évolution des mesures de rappel et précision relativement à l'appariement élémentaire F1, nous permet de savoir si notre méthode de combinaison d'appariements détériore considérablement la perti-

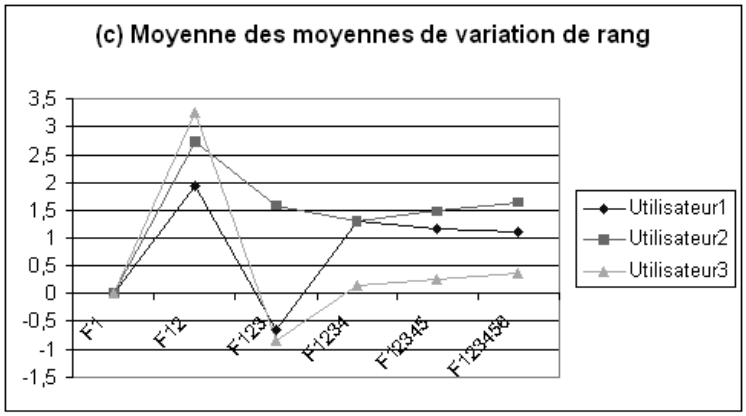

FIGURE 5.10 – Moyenne des moyennes de variation de rang (sur les 30 premiers résultats) pour les 50 requêtes de la collection CLEF 2001

nence relative au contenu des documents, dans le cas où celle-ci est le critère le plus important pour la sélection des résultats. Ceci n'est pas le cas dans nos expérimentations, vu la faiblesse des différences de rappel et de précision (relativement à F1) entre les différentes combinaisons d'appariements.

Par ailleurs, l'ordre des résultats peut être modifié de façon significative, afin de correspondre davantage aux préférences de l'utilisateur. Ainsi, on note des variations de rang pouvant aller jusqu'à des moyennes d'environ 2 à 4 positions parmi les 30 premiers documents (*cf.* figure FIG. 5.10). On peut noter que lorsque la courbe des variations de rang croît, cela signifie qu'un nombre important de documents voit leur position augmenter dans le classement des résultats et lorsqu'elle décroît, cela signifie plutôt qu'un nombre important de documents voit leur position dans le classement diminuer. Ainsi dans notre contexte expérimental, la prise en compte de différents appariements permet d'ajuster les résultats aux *« caractéristiques ou préférences »* de l'utilisateur.

En général, on peut dire que la prise en compte d'un nouvel appariement élémentaire, avec notre méthode de combinaison d'appariements, agit plus sur la modification de l'ordonnancement des résultats. De plus, notre méthode d'appariement permet également d'éviter une divergence importante des résultats obtenus par combinaison d'appariements par rapport aux résultats obtenus avec l'appariement le plus important, qui pour notre expérimentation est nommé *sujet de l'article*.

5.5 Conclusion

Les expérimentations que nous avons présentées dans ce chapitre sont subdivisées en deux parties : la présentation d'un outil d'aide à la description, à la visualisation et à l'analyse sémantique de profils pour l'accès à des ressources nommé *SemanticProfile* et l'évaluation des méthodes d'analyse sémantique et d'appariement de profils. Ces expérimentations ont pour but de montrer la faisabilité de nos propositions, mais surtout de simuler dans un cadre expérimental l'intérêt de nos propositions. Nous envisageons de poursuivre ce premier travail d'expérimentation dans un contexte applicatif réel, comme la mise sur pied d'un système collaboratif au sein d'une communauté de chercheurs, par exemple. Notons cependant que les expérimentations menées ont nécessité beaucoup de recherches. Ceci s'explique par le fait que les outils d'exploitation de documents RDF sont assez nouveaux et sont en perpétuelle évolution.

Conclusion générale et perspectives

Exploiter les ressources mises à disposition en fonction d'une application d'accès et de façon adaptée aux usagers est une tâche complexe qui nécessite d'appréhender, et donc de modéliser l'hétérogénéité et la granularité de ces ressources. Cette modélisation doit être suffisamment flexible pour permettre de faire interopérer de façon optimale différentes ressources (usagers, informations, dispositifs mobiles, etc.). Nos travaux de recherche se situent dans cette perspective et élaborent des contributions sur deux grands axes :

1. *la modélisation de profils* à travers un modèle générique de profils qui permet :
   (a) d'instancier des profils pour des types de ressources non pré-définies (*cf.* section 3.3.4) : profil de thèse, profil utilisateur mobile, profil de fournisseur de services et autres ;
   (b) de définir une structure logique flexible en donnant les noms que l'on souhaite aux éléments de cette structure logique ;
   (c) de définir une sémantique pour chaque élément descriptif d'un profil (élément de structure logique et de contenu) afin de faciliter son interopérabilité avec d'autres profils qui ne suivent pas toujours la même taxinomie.

   Ce modèle générique de profil est basé sur la sémantique et offre donc des propriétés :
   (a) *d'extensibilité* puisque l'on peut définir la structure logique que l'on souhaite. On va pouvoir également modifier cette structure en y ajoutant ou en y supprimant des éléments ;
   (b) *de flexibilité* puisque le type de ressources à décrire n'est pas prédéfini et que l'on peut nommer les éléments descriptifs d'un profil comme on le souhaite, sans que cela nuise à l'interopérabilité de ce dernier. Cette interopérabilité va être possible grâce à la sémantique que l'on va associer à chaque élément descriptif du profil, laquelle va servir de langage pivot pour l'analyse de la compatibilité sémantique de profils différents ;

(c) *de ré-utilisabilité* puisque l'on va pouvoir décrire un profil à partir de modèles ou éléments de modèles de profils existants. De plus, on peut aussi ré-utiliser des vocabulaires issus de langages de méta-données existants (décrits dans des espaces de noms) pour décrire la sémantique des éléments d'un profil. Ainsi, la ré-utilisabilité concerne à la fois la définition de la structure logique et de la sémantique d'un profil ;

(d) *de multi-facettage de profil* puisque l'on peut ne s'intéresser qu'à une partie d'un profil pour une application donnée. Un même profil peut donc être utilisé différemment par différentes applications.

2. *l'exploitation de profils* pour l'accès à des ressources au travers de définitions de méthodes d'analyse sémantique et d'appariement de profils qui permettent de faire interagir, dans des applications, des profils différents. Ainsi :

   (a) *l'analyse sémantique de profils* va permettre de détecter automatiquement les appariements possibles entre profils différents. Les profils sont généralement décrits dans des langages différents et l'on souhaiterait les comparer en interprétant leur sémantique. L'analyse sémantique de profils va donc évaluer *le niveau d'interopérabilité* qui existe entre des profils ;

   (b) *l'appariement de profils* va permettre de mesurer un degré de similarité entre deux profils, en combinant tout ou partie des appariements élémentaires qui ont été détectés par l'étape préalable d'analyse sémantique ces profils. La méthode d'appariement de profils offre de la *flexibilité* au niveau du principe de combinaison d'appariements élémentaires, car elle permet : la sélection et la pondération des appariements élémentaires à utiliser, la prise en compte de valeurs non renseignées.

Des expérimentations ont également été menées pour montrer la faisabilité des propositions faîtes sur les deux aspects résumés précédemment. Ces expérimentations ont conduit au développement d'un outil d'aide à la construction, à la visualisation et à l'analyse sémantique de profils qui peut être intégré dans le processus de développement de diverses applications d'accès à des ressources. De plus, des évaluations des méthodes d'analyse sémantique et d'appariement de profils issus de notre modèle générique ont été faîtes et ces évaluations illustrent l'intérêt de ces méthodes. Ainsi, notre méthode d'analyse sémantique va permettre, dans un contexte hétérogène, de faire interopérer automatiquement des profils différents. Notre méthode d'appariement de profils, quant à elle, va permettre de comparer des profils en combinant différents appariements identifiés par la méthode d'analyse sémantique de profils.

En résumé, nos propositions permettent de garantir un cadre de modélisation et d'exploitation de profils qui soit à la fois : *extensible, flexible, ré-utilisable et interopérable.* Nos contributions sont donc adaptées pour l'intégration de ressources hétérogènes. L'hétérogénéité ici peut être liée à la granularité, à la structure logique, au contenu ou à la sémantique des ressources.

Les perspectives qui découlent de nos travaux de recherche sont nombreuses et concernent différents axes de recherche sur les profils :

1. *Modélisation et exploitation de la notion de croyance.* Notre modèle générique nous permet de définir des poids pour des valeurs d'éléments de contenu d'un attribut feuille donné. Ce poids permet de discriminer les différentes valeurs de cet attribut entre elles en définissant *l'importance* ou pouvoir discriminant de chaque valeurs par rapport aux autres. Un principe identique est appliqué pour les appariements dans la méthode de combinaison d'appariements élémentaires. Le poids peut être calculé de différentes manières : *formules 1.1, 1.2 et 1.3 de tf et idf (cf. section 1.4.2.1), formule 4.1 du calcul de poids des appariements (cf. section 4.4.2.1), etc.*

   Par contre, une notion qui n'est pas prise en compte dans notre modèle est celle de la *croyance* qui permettrait de définir la confiance que l'on a pour une valeur d'un élément de contenu donné ou même pour un élément ré-utilisable (attribut, profil) d'un profil donné. Par exemple, si on a des préférences en langues ou dates de publication de documents qui ont été définies par l'usager lui-même et d'autres qui ont été déduites automatiquement sur l'analyse des interactions de l'usager, alors on peut vouloir attribuer une confiance plus forte sur les données renseignées par l'usager que sur celles déduites par *apprentissage.* Ainsi, nous envisageons de représenter et d'exploiter cette notion de croyance dans des profils pour l'accès à des ressources ;

2. *Exploitation et évolution de profils.* L'architecture générale d'appariement proposée (*cf.* section 4.2.1) met en exergue des ontologies globales ainsi que des instances de profils décrites sous la forme de fichiers RDF. Dans ces ontologies globales tout comme dans les instances de profils, sont décrites des instances de classes de la partie sémantique de notre modèle générique (ressources, concepts, types valeurs) ainsi que les relations sémantiques qui existent entre les instances de ces classes. On a donc une description locale (au niveau des instances de profils) et globale (au niveau des ontologies globales) des relations sémantiques qui existent entre instances de ces classes sémantiques. Dans le chapitre 4, nous prenons en compte uniquement les relations sémantiques qui existent entre instances de ces classes au niveau global.

   Dans la suite de nos travaux, nous envisageons de rajouter la prise en

compte de ces relations sémantiques au niveau des instances de profils (niveau local). Il faudra donc définir la portée de la prise en compte de relations sémantiques entre instances de classes sémantiques si on est au niveau local. Notamment, il faudra décider si une relation sémantique de niveau local peut être répertoriée au niveau global et gérer les problèmes de conflits qui pourront exister entre le niveau local et le niveau global. Ceci pose également de nombreux problèmes :

(a) évolution du système (instances de profils, ontologies globales) du fait de la définition de nouvelles relations sémantiques ;

(b) évolution de l'exploitation de ce système par la prise en compte de nouvelles relations sémantiques dans les règles d'inférence ou requêtes SPARQL ;

(c) évaluation qualitative et quantitative de la fiabilité ou de la confiance que l'on peut accorder aux résultats d'inférence ;

3. *Implémentation et évaluation de profils.* Nous comptons améliorer l'outil proposé en permettant :

   (a) la construction de profils à travers une interface *intelligente* qui masquerait au maximum la complexité du modèle générique et faciliterait ainsi la définition de profils par des utilisateurs lambda ;

   (b) la visualisation *graphique, guidée et explicative* des éléments de sémantique compatible entre profils. Dans un environnement collaboratif, un usager va donc pouvoir visualiser les autres usagers avec lesquels il peut interagir, ainsi que les appariements élémentaires sur lesquels seront basées ses interactions. La visualisation guidée va permettre de passer d'une vue globale des interactions à des vues détaillées de ces dernières. De plus, des explications (via des annotations ou des métadonnées) sur la détection de ces appariements élémentaires vont être données à l'utilisateur ;

   Par ailleurs, nous envisageons à plus long terme d'utiliser notre outil dans une application concrète avec des usagers réels. Par exemple, une application participative et collaborative pour l'accès à des ressources dans le domaine éducatif (laboratoire de recherche, université, etc.) ou autres.

4. *Modélisation de profils, sécurité et sémantique.* Le problème du respect d'informations privées est très important surtout lorsque l'on traite des aspects de personnalisation. Aujourd'hui, il y a une véritable dérive de la divulgation des données privées. Lorsqu'un internaute visite plusieurs sites Internet, ses actions sont suivies à son insu, par exemple au travers de *cookies* où des informations privées le concernant peuvent être recueillies. Un problème sous-jacent au respect des données privées est de pouvoir déterminer ce qu'est une

information privée. Selon la sensibilité de l'utilisateur, la barrière ne se place pas toujours au même niveau. La configuration idéale serait de concevoir des systèmes de personnalisation qui tiennent compte des soucis d'intimité des usagers et des lois de sécurité qui sont appliquées dans différents pays [Kob02]. Il va donc se poser le problème de vérification de la compatibilité de la sécurité entre ressources. L'hétérogénéité des langages de description de cette sécurité est liée aux exigences de chaque usager, application ou pays en matière de sécurité. Pour une vérification flexible de cette sécurité, il est nécessaire de pouvoir se baser sur une sémantique qui optimisera l'interopérabilité *de modèles de sécurité différents* tout en respectant leurs contraintes respectives.

Annexe

Langages d'interrogation de documents RDF

Le formalisme RDF a prouvé sa puissance pour la description sémantique de ressources [DN03]. Par contre, l'interprétation de cette sémantique pour l'intégration de ressources hétérogènes dans des environnements ouverts et/ou distribués [DHNS04] est liée à l'implémentation de *règles d'inférences* au travers éventuellement de *langages d'interrogation*. Il existe plusieurs langages d'interrogation de documents RDF et la qualité de ces langages est liée à la prise en compte d'un certain nombre de concepts de base de RDF [HBEV04] :

1. analyse basée sur les triplets RDF ;
2. raisonnement basé sur la sémantique formelle de RDF qui permet de déduire des informations implicites à partir d'informations explicites (subsomption, symétrie, transitivité, etc.) ;
3. prise en compte des types de données de XML Schema ;
4. flexibilité dans l'analyse des triplets décrivant une ressource qui permettra par exemple de tolérer et de gérer des contradictions ou des incomplétudes.

De façon générale, les langages RDF doivent essayer de respecter les propriétés suivantes [HBEV04] :

1. *l'expressivité* qui indique la puissance des requêtes que l'on peut formuler avec ce langage. Typiquement, un langage devrait au moins être relationnellement complet c'est-à-dire qu'il devrait permettre l'écriture des opérations d'algèbre relationnelle. Les opérations de bases de l'algèbre relationnelle sont : *la sélection, la projection, le produit cartésien, la différence et l'union*. Ces opérations de bases sont combinées pour exprimer d'autres opérations comme : *l'intersection* (combinaison de l'union et de la différence), *la jointure* (combinaison du produit cartésien et de la sélection), etc. Cependant, cette propriété d'expressivité est souvent restreinte pour pouvoir respecter d'autres propriétés comme *la sûreté* et pour permettre une exécution efficace et optimale des requêtes ;

2. *la fermeture* qui exige que les résultats d'une opération soient des éléments du modèle de données. Ce qui signifie que si le langage interroge un graphe, le résultat devrait être un graphe ;
3. *l'adéquation* qui permet de vérifier si un langage utilise tous les concepts du modèle sous-jacent. Cette propriété complète la propriété de fermeture. Pour la fermeture, le résultat d'une requête ne doit pas être extérieur au modèle et pour l'adéquation le modèle doit être exploité dans son ensemble ;
4. *l'orthogonalité* qui exige que toutes les opérations puissent être utilisées indépendamment du contexte d'usage ;
5. *la sûreté* qui garantit un résultat fini (sur un ensemble de données finies) et syntaxiquement correct. Les concepts qui nuisent généralement à cette propriété sont : la récursivité, la négation et les fonctions propriétaires à un environnement de développement particulier.

A l'heure actuelle, il n'existe pas de langage RDF qui vérifie toutes ces propriétés. Le tableau *Tab. 5.11* présente une étude comparative de langages d'interrogation de documents RDF sur certaines propriétés importantes qu'elles devraient respecter. Une liste plus exhaustive de dimensions de comparaisons est donnée par [HBEV04]. Dans le tableau *Tab. 5.11*, le symbole « − » signifie que la propriété n'est pas respectée, le symbole « ∘ » signifie que la propriété est partiellement respectée et le symbole « • » signifie que la propriété est complètement respectée.

Ce tableau montre que le langage qui respecte le plus les propriétés nécessaires à l'interrogation de documents RDF est le langage *RQL*. Ce langage est donc quantitativement le plus complet mais paradoxalement, il est le moins usité à l'heure actuelle du fait de sa complexité. Par contre, RDQL qui est quantitativement moins complet a eu davantage de succès. Ceci est dû à sa simplicité qui a rapidement conduit à son implémentation dans des environnements de développement d'applications pour le web sémantique. Sur le plan qualitatif, RDQL est un langage assez développé pour les applications basées sur le web sémantique car il supporte au moins partiellement des propriétés très importantes pour ces applications comme : *les types de données, les espaces de noms et les inférences*. Ce constat le rend, dans notre contexte, plus intéressant que les langages TRIPLE, N3 et RQL (décrits également dans le tableau *Tab. 5.11*) qui ne prennent pas toujours en compte ces propriétés même partiellement. Par exemple, les types de données de XMLSchema ne sont pas du tout supportés par TRIPLE, N3 et RQL.

Notons que nous avons analysé les profils de notre environnement en nous basant sur le langage *SPARQL* [PS05] qui est une amélioration du langage *RDQL*. De plus, ces langages (*SPARQL et RDQL*) sont implémentés dans un environnement de développement java pour le web sémantique appelé *Jena*[1]

1. *cf.* urlhttp ://jena.sourceforge.net

	Langages RDF			
Dimensions de comparaison	RDQL [Sea04]	TRIPLE [SD01a]	N3 [BLCH03]	RQL [KAC+02]
Usage de chemin de graphe	•	•	•	•
Sélection, projection et produit cartésien	•	•	•	•
Union	–	•	•	•
Différence	–	–	–	•
Fonctions d'agrégation et de groupage	–	–	•	•
Quantificateur universel	–	–	–	•
Récursivité	–	•	•	–
Espaces de noms	○	–	•	•
Types de données (XML Schema)	○	–	–	–
Déduction de données implicites (subsomption, symétrie, etc.)	○	○	○	•

TABLE 5.11 – Étude comparative de langages d'interrogation de documents RDF

que nous avons utilisé également dans nos expérimentations. Les avantages de SPARQL par rapport à RDQL est qu'il permet :

1. l'usage de chemins de graphe optionnels ;
2. l'usage de chemins de graphe alternatifs ;
3. l'usage d'expressions régulières pour comparer des chaînes de caractères ;
4. l'usage de fonctions d'agrégat ;
5. de travailler sur plusieurs graphes dans une seule requête.

Description générale du langage SPARQL

La structure générale d'une requête SPARQL est décrite dans le tableau TAB. 5.12.

Où :

— *liste_de_préfixes* décrit une liste de préfixes pour des espaces de noms utilisés dans la requête. Pour cela, on utilise le mot réservé *PREFIX*

```
liste_de_préfixes
SELECT liste_de_variables
FROM liste_de_fichiers_rdf
WHERE { liste_opérations }
Fonctions d'agrégat
```

TABLE 5.12 – Structure générale d'une requête SPARQL

suivi de l'adresse URI de l'espace de noms considéré. Par exemple : *PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>* ;

— *liste_de_variables* est une liste de variables qui peut représenter indifféremment le sujet, le prédicat ou l'objet d'un triplet. Une variable est composée de caractères alphanumériques et est toujours précédée par le caractère « ? » : *?a, ?personne, ?voiture, ?attribut, etc.* Notons qu'il existe une variable particulière notée *[]* qui est une variable non nommée et qui est utilisée lorsque la valeur de cette variable peut être différente dans une succession de sélections ou de jointures ;

— *liste_opérations* permet de décrire des combinaisons d'opérations entre des patterns de triplets : sélection, jointure, etc.
L'opération de sélection est réalisée en comparant tous les éléments d'un pattern (squelette ou schema) de triplet de la requête avec tous les triplets du graphe interrogé. On peut également définir l'opération de sélection en utilisant le mot réservé *FILTER* suivi de la condition de sélection. Cette condition fait usage d'opérateurs de comparaisons (=, <, <=, >, >=, <>) et d'opérateurs logiques (« && » pour le ET logique et « || » pour le OU logique).
La jointure, quant à elle, est définie par l'usage du même nom de variable dans différents patterns de triplets RDF de la clause WHERE. Notons que entre chemins de graphes, le *ET* est représenté par l'opérateur « . » et le *OU* par les opérateurs *OPTIONAL* pour les chemins optionels et *UNION* pour les chemins possibles.
Ainsi, la liste d'opérations suivante *« (?a sp : estComposéDe ?b) . (?b sp : représente dc : date) »* décrit :

1. des sélections de triplets dont la valeur de prédicat est *sp : estComposéDe* ;
2. des sélections de triplets dont la valeur de prédicat est *sp : représente* et la valeur de l'objet *dc : date* ;
3. une jointure sur la variable ?*b* qui doit être objet des triplets ayant pour prédicat *sp : estComposéDe ET* sujet des triplets ayant pour prédicat *sp : représente* et pour objet *dc : date*.

— *Fonctions d'agrégat* qui permet d'appliquer des opérations sur tout

ou partie des résultats comme : *ORDER BY* pour le tri des résultats.

Notons également que SPARQL a amélioré le parcours de graphes avec l'usage de chemin optionel (grâce au mot réservé *OPTIONAL*) ainsi qu'avec l'usage d'alternatives entre plusieurs chemins possibles (grâce au mot réservé *UNION*). Ceci permet de parcourir avec plus de souplesse un graphe RDF car on n'est plus obligé d'être sûr de la présence d'un pattern de triplet dans un graphe. Si on a un doute sur l'existence d'un pattern de triplet, alors on utilise le mot réservé *OPTIONAL*. Si on est sûr de l'existence d'au moins un pattern de triplet sur une liste de patterns donnée, alors on utilise le mot réservé *UNION*.

Publications de l'auteur

Thèse

[1] P. L. Tchienehom, Modélisation et Exploitation de Profils : Accès Sémantique à des Ressources, Thèse de doctorat en informatique, Université de Toulouse (France), Novembre, 2006.

Revues Nationales

[2] M. Chevalier and C. Soulé-Dupuy and P. L. Tchienehom, Semantics-based Profiles Modeling and Matching for Resources Access. *Journal des Sciences Pour L'Ingénieur (JSPI), (ed.) African Journals OnLine (AJOL)*, vol.1, n°7, pages 54-63, 2006.

Conférences Internationales

[3] M. Chevalier and C. Soulé-Dupuy and P. L. Tchienehom, Profiles Semantics and Matchings Flexibility for Resource Access. *International IEEE conference on Signal-Image Technology & Internet-based Systems (SITIS'05)*, pages 224-231, 2005.

[4] M. Chevalier and C. Soulé-Dupuy and P. L. Tchienehom, A profile-based architecture for a flexible and personalized information access. *IADIS International Conference (IADIS/WWW Internet 2004)*, vol.2, pages 1017-1022, 2004, IADIS - ISBN 972-99353-0-0.

Workshops Internationaux

[5] M. Chevalier and C. Soulé-Dupuy and P. L. Tchienehom, Profile reusability for personalized information access : application to users contexts

determination. *International Workshop on Context-based Information Retrieval (CIR'05)*, in conjonction with Context'05, pages 71-82, 2005.

[6] P. L. Tchienehom, Profiles semantics for personalized information access. *International Workshop on Personalization on the Semantic Web (PerSWeb'05), in conjonction with UM'05*, pages 102-111, 2005.

Conférences Nationales

[7] M. Chevalier and C. Soulé-Dupuy and P. L. Tchienehom, Interopérabilité de profils pour l'accès à des ressources, 24ième Congrès National Inforsid'06, pages 65-80, 2006.

[8] P. L. Tchienehom, Modèle générique de profils pour la personnalisation de l'accès à l'information. *23ième Congrès National Inforsid'05*, pages 269-284, 2005.

[9] P. Tchienehom, Architecture de Recherche et de Recommandation d'Information à base de Profils : définitions, acquisitions et usages de profils. *22ième Congrès National Inforsid'04*, pages 143-159, 2004.

Rapports de recherche

[10] P. L. Tchienehom, Une nouvelle approche pour un accès personnalisé à l'information, *Rapport Interne - Laboratoire IRIT de Toulouse*, n° IRIT/2005_2_R, 2005.

Colloques

[11] P. Tchienehom, Accès à une information Pertinente et Personnalisée : Approche à base de Profils. *Colloque des doctorants Edit'04*, pages 100-104, 2004.

Bibliographie

[AGS06] M. Assem, A. Gangemi, and G. Schreiber, editors. *RDF/OWL Representation Of Wordnet*. Editor's Draft, April 2006. `http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion.html`.

[AH01] A.T. Arampatzis and A. Van Hameren. The score-distributional threshold optimization for adaptive binary classification tasks. *In proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[Ame01] Y. Amerouali. Métadonnées basées sur l'association d'éléments de description de ressources et d'éléments de profil d'utilisateur. *Thèse de Doctorat en sciences de l'information et de la communication*, Mai 2001. Université Claude Bernard Lyon I - ENSSIB (France).

[Amm03] A. Ben Ammar. Profils en recherche d'information : définition, exploitation et adaptation. *Thèse de Doctorat en Informatique*, Avril 2003. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[AVF06] K. Abbas, C. Verdier, and A. Flory. Gestion des connaissances centrée sur le modèle utilisateur. *Gestion et Ingénierie des SystEmes Hospitaliers (GISEH)*, 2006.

[BAGB03] M. Baziz, N. Aussenac-Gilles, and M. Boughanem. Exploitation de liens sémantiques pour l'expansion de requêtes dans un système de recherche d'information. *21ième Congrès Inforsid*, pages 121–134, 2003.

[BBB03a] J.C. Bottraud, G. Bisson, and M.F. Bruandet. An adaptative information research personnal assistant. *In proceedings of Workshop AI2IA (Artificial Intelligence, Information Access and Mobile Computing) IJCAI'03*, 2003.

[BBB03b] J.C. Bottraud, G. Bisson, and M.F. Bruandet. Apprentissage de profils pour un agent de recherche d'information. *Conférence d'Apprentissage (CAp03)*, 2003.

[BBB04] J.C. Bottraud, G. Bisson, and M.F. Bruandet. Expansion de requêtes par apprentissage automatique dans un assistant pour la recherche d'information. *COnférence en Recherche d'Information et Applications (CORIA'04)*, pages 89–105, 2004.

[BC92] N.J. Belkin and W.B. Croft. Information filtering and information retrieval : Two sides of the same coin ? *Communications of the ACM, Information Filtering*, 35(12) :29–38, 1992.

[BCSD99] M. Boughanem, C. Chrisment, and C. Soulé-Dupuy. Query modification based on relevance backpropagation in adhoc environment. elsevier science. *Information Processing & Management Journal*, 35 :121–139, 1999.

[BD03] L. Barhuus and A. Dey. Is context-aware computing taking control away from the user ? three levels of interactivity examined. *In Proceedings of the fifth international conference on Ubiquitous computing (Ubicomp'03)*, 2003. Seatle, Washington (USA).

[BDBD+00] G. Beged-Dov, D. Brickley, R. Dornfest, I. Davis, L. Dodds, J. Eisenzopf, D. Galbraith, R.V. Guha, K. MacLeod, E. Miller, A. Swartz, and E. van der Vlist. Rdf site summary (rss) 1.0. *RSS-DEV Working Group*, 35(12), 2000. http ://web.resource.org/rss/1.0/spec.

[BE02] L. Berti-Equille. Annotation et recommandation collaboratives de documents selon leur qualité. *Ingénierie des Systèmes d'Information, Recherche et Filtrage d'information. RSTI serie ISI-NIS*, 7(2) :125–155, 2002.

[BE03] L. Berti-Equille. Quality-based recommendation of xml documents. *Journal of Digital Information Management*, 1(3) :117–128, 2003.

[Ber04] A. Berglund, editor. *Extensible Stylesheet Language (XSL) Version 1.1.* W3C Working Draft, December 2004. http ://www.w3c.org/TR/xsl11/.

[BG04] D. Brickley and R.V. Guha, editors. *RDF Vocabulary Description Language 1.0 : RDF Schema.* February 2004. http ://www.w3.org/TR/rdf-schema/.

[BHH04] S. Buchholz, T. Hamann, and G. Hubsch. Comprehensive structured context profiles (cscp) : Design and experiences. *In Proceedings of the Workshop on Context Modeling and Reasoning (CoMoRea'04)*, pages 43–47, 2004.

[BHL99] T. Bray, D. Hollander, and A. Layman, editors. *Namespaces in XML.* World Wide Web Consortium (W3C), January 1999. http ://www.w3.org/TR/REC-xml-names/.

[BHM01] A. Benammar, G. Hubert, and J. Mothe. Reformulation automatique des profils utilisant un ensemble local de documents. *Actes de la Conférence Veille Stratégique, Scientifique et Technologique (VSST'01)*, pages 321–329, 2001.

[BJMSD00] M. Boughanem, C. Julien, J. Mothe, and C. Soulé-Dupuy. Mercure at TREC8 : adhoc, filtering, Web and CLIR tasks. *In Harman D.K (ed.) NIST Gaithersburg Proceedings of the eight International Conference on Text REtrieval TREC8*, November 2000. Maryland (USA).

[BK05] M. Bouzeghoub and D. Kostadinov. Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils. *COnférence en Recherche d'Informations et Applications (CORIA'05)*, pages 201–218, 2005.

[BL01a] T. Bauer and D. B. Leake. Real time user context modeling for information retrieval agents. *Conference on Information and Knowledge Management (CIKM'01)*, 2001.

[BL01b] T. Berners-Lee. Axioms, architecture and aspirations. *W3C all-working group plenary Meeting*, 2001. http ://www.w3.org/2001/Talks/0228-tbl/slide5-0.html.

[BL02] T. Bauer and D. B. Leake. Exploiting information access patterns for context-based retrieval. *Communications of the ACM*, 2002. ISBN 1-58113-459-2/02/0001.

[BLCH03] T. Berners-Lee, D. Connolly, and S. Hawke. Semantic web tutorial using n3. *Tutorial presented at the Twelfth International World Wide Web Conference (WWW'03)*, Budapest (Hungary) 2003. http ://www.w3.org/2000/10/swap/doc.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001. `http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`.

[BM04] P. V. Biron and A. Malhotra, editors. *XML Schema Part 2 : Datatypes Second Edition*. W3C Recommendation, October 2004. http ://www.w3c.org/TR/xmlschema-2/.

[BM05] D. Brickley and L. Miller. Foaf vocabulary specification. *Namespace Document, Pages about Things Edition*, July 2005. http ://xmlns.com/foaf/0.1/.

[BMRM96] E. Bloedorn, I. Mani, T. Richard, and McMillan. Representational issues in machine learning of user profiles. *Notes of the AAAI Spring Symposium on Machine Learning in Information Access*, 1996. Stanford University.

[BPSM+04] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, editors. *Extensible Markup Language (XML) 1.0 (Third Edition).* W3C Recommendation, February 2004. http ://www.w3c.org/TR/2004/REC-xml-20040204/.

[Bru95] P. Brusilovsky. Systèmes tuteurs intelligents pour le World-Wide Web. *In R. Holzapfel (ed.) Proceedings of Third International WWW Conference*, pages 42–45, 1995.

[BS97] M. Balabanovic and Y. Shoham. Fab : Content-based, collaborative recommendations. *Communications of the ACM*, 40(3) :66–72, 1997.

[BTT04] M. Boughanem, M. Tmar, and H. Tebri. Filtrage d'information. *Méthodes avancées pour les systèmes de recherche d'informations, Hermes Sciences Publication, Lavoisier*, pages 137–161, 2004. Chapitre 7.

[BvHH+04] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference.* W3C Recommendation, February 2004. http ://www.w3.org/TR/owl-ref/.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* First edition, Addison Wesley, ISBN 0-201-39829-X, 1999.

[CAF00] R. Chbeir, Y. Amghar, and A. Flory. Mims : A prototype for medical image retrieval. *RIAO Conference*, 2000.

[Cal98] J. Callan. Learning while filtering documents. *In Proceedings of the twenty first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 224–231, 1998.

[CB02] N. Correia and M. Boavida. Towards an integrated personalization framework : A taxonomy and work proposals. *In Proceedings of the Workshop on Personalization Techniques in Electronic Publishing on the Web : Trends and Perspectives*, May 2002. Malaga (Spain).

[CBHS05] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs. *Web Semantics : Science, Services and Agents on the World Wide Web 3*, pages 247–267, 2005.

[CCZC02] J. Caussanel, J-P. Cahier, M. Zacklad, and J. Charlet. Les topic maps sont-ils un bon candidat pour l'ingénierie du web sémantique ? *Actes de la conférence Ingénierie des Connaissances*, 2002.

[CD99] J. Clark and S. DeRose, editors. *XML Path Language (XPath) Version 1.0.* W3C Recommendation, November 1999. http ://www.w3c.org/TR/xpath.

[Che02] M. Chevalier. Interface intelligente pour l'aide à la recherche d'information sur le web. *Thèse de Doctorat en Informatique*, Décembre 2002. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[CJ01] M. Chevalier and C. Julien. ISIDORView : une interface de visualisation des résultats de recherche d'informations. *Actes de Extraction et Gestion de Connaissances EGC'01, Extraction des Connaissances et Apprentissage. Editions Hermès Science Publication*, 1(1) :135–147, 2001. ISBN 9-782746-202160.

[CKK02] Y.H. Cho, J. Kyeong, and S.H. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert System with Applications*, 23(3) :329–342, 2002.

[CKR04] B. Chang, J. Kesselman, and R. Rahman, editors. *Document Object Model (DOM) Level 3 Validation Specification Version 1.0.* W3C Recommendation, January 2004. http ://www.w3.org/TR/2004/REC-DOM-Level-3-Val-20040127/.

[CMRW05] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C Working Draft*, 1(1), May 2005. http ://www.w3.org/TR/2005/WD-wsdl20-20050510/.

[CMS00] C. Chrisment, J. Le Maître, and F. Sèdes. Bases de données documentaires. *Technique de l'Ingénieur, Traité Informatique, H 7 248*, Mai 2000.

[CP43] W.S Mc Cullogh and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, (5) :115–133, 1943.

[Cra80] J.B. Crampes. Aide à l'interrogation d'un dictionnaire de données. *RAIRO Informatique/Computer Science*, 14(1) :86–95, 1980.

[CS00] P. Cotter and B. Smith. PTV : Intelligent personalised TV guides. *American Association for Artificial Intelligence (AAAI)*, 2000. www.aaai.org.

[CSDT04] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom. A profile-based architecture for a flexible and personalized information access. *IADIS International Conference (IADIS/WWW Internet 2004)*, 2 :1017–1022, October 2004. IADIS - ISBN 972-99353-0-0.

[CSDT05a] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom. Profile re-usability for personalized information access : application to users contexts determination. *International Workshop*

on Context-based Information Retrieval (CIR'05), in conjonction with Context'05, pages 71–82, Juillet 2005.

[CSDT05b] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom. Profiles semantics and matchings flexibility for resource access. *International IEEE conference on Signal-Image Technology & Internet-based Systems (SITIS'05)*, pages 224–231, Novembre 2005.

[CSDT06a] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom. Interopérabilité de profils pour l'accès à des ressources. *24ième Congrès National Inforsid'06*, pages 65–80, Juin 2006.

[CSDT06b] M. Chevalier, C. Soulé-Dupuy, and P. L. Tchienehom. Semantics-based profiles modeling and matching for resources access. *Journal des Sciences Pour L'Ingénieur (JSPI), (ed.) African Journals OnLine (AJOL)*, 1(7) :54–63, 2006.

[CvHH+01] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *DAML+OIL (March 2001) Reference Description.* W3C Note, December 2001. http ://www.w3.org/TR/daml+oil-reference.

[Dai98] B. Daille. Traitement des corpus, application à l'extraction automatique de terminologie. *La structure du lexique, Ateliers en morphologie. Paul Boucher (ed.) les ateliers* COLEX *(Centre Ouest Lexique)*, pages 159–164, 1998. Université de Nantes.

[DBGLN04] N. Denos, C. Berrut, L. Gallardo-Lopez, and An-Te Nguyen. Une plateforme de filtrage collaboratif orientée vers la communauté. *COnférence en Recherche d'Information (CORIA)*, 2004.

[DDF90] S. Deerwester, S. Dumais, and G.W. Furnas. Indexing by latent semantic indexing. *Journal of American Society for Information Science*, 41 :391–407, 1990.

[DDMM03] S. J. DeRose, R. Daniel, E. Maler, and J. Marsh, editors. *XPointer xmlns() Scheme.* W3C Recommendation, March 2003. http ://www.w3.org/TR/xptr-xmlns/.

[DHNS04] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed e-learning environments. *In proceeding of WWW2004*, 2004.

[DMO01] S. DeRose, E. Maler, and D. Orchard, editors. *XML Linking Language (XLink) Version 1.0.* W3C Recommendation, June 2001. http ://www.w3.org/TR/xlink/.

[DN03] P. Dolog and W. Nejdl. Challenges and benefits of the semantic web for user modelling. *In proceeding of AH 2003*, 2003.

[Dum94] S.T. Dumais. Latent Semantic Indexing (LSI) : TREC-3 report. *In Proceedings of the Third Text Retrieval Conference (TREC-3). NIST Special Publication*, 1994.

[Dun00] M.D. Dunlop. Development and evaluation of clustering techniques for finding people. *In Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM2000)*, October 2000. Basel (Switzerland).

[Fel84] H. Felber. Terminology manual. *UNESCO*, 1984. Paris.

[Fer95] J. Ferber. *Les systèmes multi-agents.* InterEditions, ISBN 2-7296-0572-X, 1995.

[FFR96] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server : A tool for collaborative ontology construction. *In Knowledge Acquisition Workshop*, 1996.

[FH04] H. Folch and B. Habert. Langages de métadonnées pour web(s) sémantique(s). *Chapitre 3 dans Méthodes avancées pour les systèmes de recherché d'informations, sous la direction de Madjid Ihadjadene*, 2004.

[Fla98] P. Flach. The Logic of Learning : A brief Introduction to Inductive Logic Programming. *Department of Computer Science, University of Bristol, Bristol (UK)*, 1998.

[FW04] D. C. Fallside and P. Walmsley, editors. *XML Schema Part 0 : Primer Second Edition.* W3C Recommendation, October 2004. http ://www.w3c.org/TR/xmlschema-0/.

[FY92] W.B Frakes and R.B. Yates. *Information Retrieval : Data Structures & Algorithms.* Ed. Addison Wesley Publishing Company, ISBN 0-134-63837-9, 1992.

[GGM+02] N. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with dolce. *In Proceedings of EKAW2002*, 2002.

[GGMT99] L. Gravano, H. Garcia-Molina, and A. Tomasic. Gloss : Text-source discovery over the internet. *ACM transactions on Database systems*, 24(2) :229–264, 1999.

[GLBD04] L. Gallardo-Lopez, C. Berrut, and N. Denos. Une approche pour le contrôle de la qualité des systèmes de filtrage collaboratif. *International Journal of Information and Communication Sciences for Decision Making, Special Issue ISDM'Majestic*, (13), Février 2004.

[GMM03a] P. Grosso, E. Maler, and J. Marsh, editors. *XPointer element() Scheme.* W3C Recommendation, March 2003. http ://www.w3.org/TR/xptr-element/.

[GMM03b] P. Grosso, E. Maler, and J. Marsh, editors. *XPointer Framework.* W3C Recommendation, March 2003. http ://www.w3.org/TR/xptr-framework/.

[GNOT92] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM, Information Filtering*, 35(12) :61–70, 1992.

[GSK+99] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. *In Proceedings of AAAI. AAAI Press*, 35 :439–446, 1999.

[Hay04] P. Hayes, editor. *RDF Semantics*. February 2004. http ://www.w3.org/TR/rdf-mt/.

[HBEV04] P. Haase, J. Broekstra, A. Eberhart, and R. Volz. Comparison of rdf query languages. *In proceedings of the third International Semantic Web Conference ISWC'04*, 2004. Hiroshima (Japan).

[HBS02] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. *In Proceedings of SCI*, 2002.

[HKNH00] K. Hoashi, M. Kazunori, I. Naomi, and K. Hashimoto. Document filtering method using non relevant information profile. *In Proceedings of the twenty third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Distributed Retrieval*, pages 176–183, 2000.

[Hol75] J. Holland. Adaptation in natural and artificial systems. *Ann Arbor. University of Michingan Press*, 1975. Michingan.

[HW04] D. Halvatzaras and M. H. Williams. A context aware user profile for personalization. *IADIS International Conference (IADIS/WWW Internet 2004)*, 1 :452–460, 2004.

[Jac98] M. Jaczynski. Modèle et plate-forme à objets pour l'indexation des cas par situations comportementales : application à l'assistance à la navigation sur le Web. *Thèse de doctorat en Informatique, Université de Nice*, 1998.

[Jon72] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1) :11–20, 1972.

[JRP01] L. Jeribi, B. Rumpler, and J.M. Pinon. Système d'aide à la recherche et à l'interrogation de bases documentaires, fondé sur la réutilisation d'expériences. *19ième Congrès d'Inforsid*, pages 443–463, 2001.

[KAC+02] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Schol. Rql : A declarative query language for rdf. *In proceedings of the Eleventh International World Wide Web Conference (WWW'02)*, 2002. Honolulu (Hawaii).

[Kay95] J. Kay. The um toolkit for reusable, long term user models. *User Modeling and User-Adapted Interaction*, 4(3) :149–196, 1995.

[Kay05] M. Kay, editor. *XSL Transformations (XSLT) Version 2.0.* W3C Working Draft, February 2005. http ://www.w3.org/TR/xslt20/.

[KC03] H. R. Kim and P. K. Chan. Learning implicit interest hierarchy for context in personalization. *In Proceedings of the 8th international conference on Intelligent user interfaces*, pages 101–108, 2003.

[KC04] G. Klyne and J. J. Carroll, editors. *Resource Description Framework (RDF) : Concepts and Abstract Syntax.* W3C Recommendation, February 2004. http ://www.w3.org/TR/rdf-concepts/.

[KGB98] C. Kurzke, M. Galle, and M. Bathelt. WebAssist : a user profiles specific information retrieval assistant. *Computer Networks*, 30(1-7) :654–655, 1998. http ://www7.scu.edu.au/1903/com1903.htm.

[Khr04] K. Khrouf. Entrepôts de documents : de l'alimentation à l'exploitation. *Thèse de Doctorat en Informatique*, Juillet 2004. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[KKPS02] J. Kahan, M. R. Koivunen, E. Prud'Hommeaux, and R.R. Swick. Annotea : an open rdf infrastructure for shared web annotations. *Computer Networks*, 39(5) :589–608, August 2002.

[KMM+97] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Grouplens : Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3) :77–87, March 1997.

[Kob01] A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction, Kluwer Academic Publishers*, 11 :49–63, 2001. Printed in the Netherlands.

[Kob02] A. Kobsa. Personalized hypermedia and international privacy. *Communications of the ACM, Special Isuue on Adaptive Web-Based Systems and Adaptive Hypermedia, ACM Press*, 2002.

[Kor97] R.R. Korfhage. *Information storage and retrieval.* Wiley computer publishing, ISBN 0-471-14-338-3, 1997.

[KR03] R. Kimball and M. Ross. *Entrepôts de données - Guide pratique de modélisation dimensionnelle.* Vuibert, ISBN 2-7117-4811-1, 2003.

[Kru97] B. Krulwich. Lifestyle finder : Intelligent user profiling using large-scale demographic data. *AI Magazine*, 18(2) :37–45, 1997.

[KRW+04] G. Klyne, F. Raynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, and L. Tran, editors. *Composite Capability/Preference Profiles (CC/PP) : Structure and Vocabularies 1.0.* W3C Recommendation, January 2004. http ://www.w3.org/TR/CCPP-struct-vocab/.

[KV02] M. Karamuftuoglu and F. Venuti. Okapi in tips : The changing context of information retrieval. *In Proceedings of the Workshop on Personalization Techniques in Electronic Publishing on the Web : Trends and Perspectives*, May 2002. Malaga (Spain).

[KYS05] O. Kwon, K. Yoo, and E. Suh. Ubiss : a proactive intelligent decision support system as an expert deploying ubiquitous computing technologies. *Expert Systems with Applications*, pages 149–161, 2005.

[LC96] A. V. Leouski and W. B. Croft. An evaluation of techniques for clustering search results. *Technical Report IR-76*, 1996.

[LC99] S. Lainé-Cruzel. Profildoc : Filtrer une information exploitable. *Bulletin des Bibliothèques de France*, (5) :60–65, 1999. http ://www.enssib.fr/bbf/bbf-99-5/10_lainecruzel.pdf.

[Ler91] Israel-César Lerman. Foundations in Likehood Linkage Analysis Classification Method. *Applied Stochastic models and data analysis*, 7 :69–76, 1991.

[Leu02a] B. Leuf. Intoduction to peer architectures. *informit.com*, 2002. http ://www.informit.com/articles/article.asp ?p=29238&rl=1.

[Leu02b] B. Leuf. *Peer to Peer : Collaboration and Sharing over the Internet.* Addison Wesley, ISBN 0-201-76732-5, 2002.

[Lie95] H. Lieberman. Letizia : An agent that assists web browsing. *In Proceedings of the IJCAI'95*, pages 924–929, 1995.

[LK03] C. H. Lee and T. Kanungo. The architecture of trueviz : grounftruth/metadata editing and visualizing toolkit. *Pattern Recognition*, 36(3) :811–825, March 2003.

[LS99] O. Lassila and R. R. Swick, editors. *Resource Description Framework (RDF) Model and Syntax Specification.* February 1999. http ://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[Luh58] H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2) :159–165, 1958.

[LV05] C. Lopez-Velasco. Awsdl : une extension de wsdl pour des services web adaptés. *23ième Congrès National Inforsid'05*, pages 134–148, 2005.

[Mar02] S. Marel. L'appariement de profils. *Diplôme de Recherche Technologique (DRT)*, Décembre 2002. Université Paul Sabatier

Toulouse 3 - Laboratoire IRIT et entité Hewlett Packard Laboratoire de Grenoble (France).

[Mar04] J. M. Martinez, editor. *MPEG-7 Overview.* International Organisation for Standardisation ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Audio, October 2004. http ://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

[MdR94] S. Muggleton and L. de Raedt. Inductive Logic Programming : Theory and Methods. *Journal of Logic Programming*, 19(20) :629–679, 1994.

[MG00] P.A. Muller and N. Gaertner. *Modélisation objet avec UML.* Deuxième édition, Eyrolles, ISBN 2-212-09122-2, 2000.

[Mic01] A. Michard. Bibliothèques numériques universelles. vers un web sémantique ? *Tutoriaux de la conférence* INFORSID'2001, Mai 2001. Martigny (Suisse).

[Mit97] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[Mit03] Soap version 1.2 part 0 : Primer. *W3C Recommendation*, June 2003. http ://www.w3.org/TR/2003/REC-soap12-part0-20030624/.

[Mla96] D. Mladenic. Personal WebWatcher : design and implementation. *Technical Report IJS-DP-7472*, 1996. Department of Intelligent Systems, Slovenia : J. Stefan Institute.

[MLR03] M. Montaner, B. Lopez, and J.L.D.L Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review. Kluwer Academic Publishers*, 19 :285–330, 2003.

[MT02a] S. Mizzaro and C. Tasso. Ephemeral and persistent personalization in adaptive information access to scholarly publications on the Web. *In Paul De Bra, Peter Brusilovsky, and Ricardo Conejo editors, Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference AH 2002, LNCS 2347*, pages 306–316, May 2002. ISBN 3-540-43737-1.

[MT02b] S. Mizzaro and C. Tasso. Personalization techniques in the tips project : The cognitive filtering module and the information retrieval assistant. *In Proceedings of the Workshop on Personalization Techniques in Electronic Publishing on the Web : Trends and Perspectives*, May 2002. Malaga (Spain).

[NDB06] An-Te Nguyen, N. Denos, and C. Berrut. Exploitation des données « disponibles à froid » pour améliorer le démarrage à froid dans les systèmes de filtrage d'information : une approche expérimentale fondée sur les « espaces de communautés » et la classification par règles. *24ième Congrès National Inforsid'06*, 2006.

[NSD+01] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating semantic web concepts with protege2000. *IEEE Intelligent Systems*, 16(2) :60–71, 2001.

[NVB+01] C. Nédellec, M. O. A. Vetah, P. Bessières, C. Brun, and B. Jacq. Reconnaître les fragments pertinents pour l'extraction d'information dans les textes de génomique, un problème de classification. *Actes de la Conférence francophone d'Apprentissage (CAP 2001)*, 2001.

[OVF05] M. Ouziri, C. Verdier, and A. Flory. Data integration and user modelling : An approach based on topic maps and description logics. *7th International Conference on Enterprise Information Systems (ICEIS)*, 10 :171–175, 2005.

[Paz99] M. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 1999.

[PBH+03] G. Paquette, J. Bourdeau, F. Henri, J. Basque, M. Leonard, and M. Citra. Construction d'une base de connaissances et d'une banque de ressources pour le domaine du téléapprentissage. *Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation*, 10, 2003. http ://sticef.univ-lemans.fr/num/vol2003/paquette-01s/sticef_2003_paquette_01s.pdf.

[PDAB05] G. Pierra, H. Dehainsala, Y. A. Ameur, and L. Bellatreche. Base de données à base ontologique : principe et mise en œuvre. *Ingénierie des Systèmes d'Information (ISI)*, 2005.

[PFL+02] M. Paternostre, P. Francq, J. Lamoral, D. Wartel, and M. Saerens. Carry, un algorithme de désuffixation pour le français. *Projet Galilei*, Juillet 2002. http ://www.galilei.ulb.ac.be/rd/pu_publi/carryfinal.pdf.

[PMB96] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert : Identifying interesting web sites. *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 54–61, 1996.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3) :130–137, 1980.

[Pri04] C. Priest. A conceptual architecture for semantic web services. *In Proceedings of the Third International Semantic Web Conference*, 2004.

[PS05] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. *W3C Working Draft*, July 2005. http ://www.w3.org/TR/2004/rdf-sparql-query/.

[PSC+02] J. Pitkow, N. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search : A contextual computing approach may prove a breakthrough in personalized search efficiency. *Communications of the ACM*, 45(9) :50–55, 2002.

[PSHH04] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, editors. *OWL Web Ontology Language Semantics and Abstract Syntax.* W3C Recommendation, February 2004. http ://www.w3.org/TR/2004/REC-owl-semantics-20040210/. Latest version available at http ://www.w3.org/TR/owl-semantics/.

[Qui93] J. R. Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kauffman, 1993.

[Ref00] J. E. Refsnes. Xml dtd - an introduction to xml document type definitions. *XMLFiles.com*, 2000. http ://www.xmlfiles.com/dtd/.

[Rij79] C.J V. Rijsbergen. *Information Retrieval.* Second edition, Butterworths, 1979.

[RJ76] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May-June 1976.

[Roc71] J.J. Rocchio. Relevance feedback in information retrieval. *In Salton Editor, The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall Inc. Englewood Cliffs (NJ)*, pages 313–323, 1971.

[RP97] J. Rucker and M.J. Polanco. Siteseer : Personalized navigation for the web. *Communications of the ACM*, 40(3) :73–75, 1997.

[RSB98] C. Rosse, L. G. Shapiro, and J. F. Brinkley. The digital anatomist foundational model : Principles for defining and structuring its concept domain. *In Proceeding of American Madical Informatics Association Fall Symposium*, pages 820–824, 1998. http ://sig.biostr.washington.edu/projects/da/.

[RW97] S. Robertson and S. Walker. On relevance weights with little relevance information. *In J. Belkin, A. Desai Narasimhalu and Peter Willet (ed.) proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval. ACM Press*, pages 16–24, May-June 1997.

[RWB99] S. Robertson, S. Walker, and M. Beaulieu. Okapi at trec-7 : Automatic adhoc filtering VLC and interactive track. *Information Processing & Management Journal, Elsevier Science*, pages 130–137, 1999. Gaithersburg (Maryland, USA).

[SAW94] B. N. Schilit, N. L. Adams, and R. Want. Context-aware computing applications. *In IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[SB90] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Science*, 41 :288–297, 1990.

[SBF98] R. Studer, V.R. Benjamins, and D. Fensel. Knowledge engineering : Principles and methods. *Data & Knowledge Engineering*, 25 :161–197, 1998.

[Scu02] M. Scuturici. Contribution aux techniques orientées objet de gestion des séquences vidéo pour les serveurs web. *Thèse de Doctorat en Informatique*, Décembre 2002. Institut National de Sciences Appliquées de Lyon - Laboratoire d'Ingénieurie des Systèmes d'Informations LISI (France).

[SD01a] M. Sintek and S. Decker. Triple : An rdf query, inference and transformation language. *In deductive Databases and Knowledge Management (DDLP'01)*, 2001.

[SD01b] C. Soulé-Dupuy. Bases d'informations textuelles : des modèles aux applications. *Mémoire d'Habilitation à Diriger des Recherches (HDR) en Informatique*, Décembre 2001. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[Sea04] A. Seaborne. Rdql : A query language for rdf. *W3C member submission*, 2004. http ://www.w3c.org/Submission/2004/SUBM-RDQL-20040109.

[Sed98] F. Sedes. Bases documentaires - hyperbases. proposition d'un modèle générique et contribution à la spécification d'un langage pour l'intégration et la manipulation d'informations semi-structurées. *Mémoire d'Habilitation à Diriger des Recherches (HDR) en Informatique*, 1998. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[Sha00] A. Sharma. A generic architecture for user modelling systems and adaptive web services. *Delhi College of Engineering*, 2000. New Delhi.

[SLP04] T. Strang and C. Linnhoff-Popien. A context modeing survey. *In Proceedings of the first International Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp'04)*, 2004.

[SM83] G. Salton and M.J McGill. *Introduction to Modern Information Retrieval*. Second Edition, Mc Graw Hill International Book Company, ISBN 0-07-Y66526-53, 1983.

[SM03] D. Saha and A. Mukherjee. Pervasive computing : A paradigm for the 21st century. *IEEE Computer*, 36(3) :25–31, 2003.

[SMB97] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, 1997. Philadelphia (USA).

[SMLP01] M. Samulowitz, F. Michahelles, and C. Linhoff-Popien. Capeus : An architecture for context-aware selection and execution of services. *In new developments in distributed applications and interoperable systems*, pages 23–39, 2001. Kluver Academic Publishers.

[Sow84] J. Sowa. *Conceptual Structures : information processing in mind and machine*. The System Programming Series, Addison Wesley Publishing Company, 1984.

[SSH97] B. Shapira, P. Shoval, and U. Hanani. Stereotypes in information filtering systems. *Information Processing & Management*, 33(3) :273–287, 1997.

[TBMM04] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, editors. *XML Schema Part 1 : Structures Second Edition*. W3C Recommendation, October 2004. http ://www.w3c.org/TR/xmlschema-1/.

[TC91] H.R Turtle and W.B Croft. Evaluation of inference network-based retrieval model. *ACM Transact. Inf. Syst.*, 9(3) :187–222, July 1991.

[Tch04a] P. Tchienehom. Accès à une information pertinente et personnalisée : Approche à base de profils. *Colloque des doctorants Edit'04*, pages 100–104, Février 2004.

[Tch04b] P. Tchienehom. Architecture de recherche et de recommandation d'information à base de profils : définitions, acquisitions et usages de profils. *22ième Congrès National Inforsid'04*, pages 143–159, Mai 2004.

[Tch05a] P. L. Tchienehom. Modèle générique de profils pour la personnalisation de l'accès à l'information. *23ième Congrès National Inforsid'05*, pages 269–284, Mai 2005.

[Tch05b] P. L. Tchienehom. Profiles semantics for personalized information access. *International Workshop on Personalization on the Semantic Web (PerSWeb'05), in conjonction with UM'05*, pages 102–111, Juillet 2005.

[Tch05c] P. L. Tchienehom. Une nouvelle approche pour un accès personnalisé à l'information. *Rapport Interne - Laboratoire IRIT de Toulouse*, Février 2005. n° IRIT/2005_2_R.

[Tch06] P. L. Tchienehom. Modélisation et exploitation de profils : Accès sémantique à des ressources. *Thèse de doctorat en informatique*, November 2006. Université de Toulouse (France).

[Tma02] M. Tmar. Modèle auto-adaptatif de filtrage d'information : apprentissage incrémental du profil et de la fonction de décision. *Thèse de Doctorat en Informatique*, 2002. Université Paul Sabatier Toulouse 3 - Laboratoire IRIT (France).

[Ved05] A. S. Vedamuthu. Web services description language (wsdl) version 2.0 soap 1.1 binding. *W3C Working Draft*, May 2005. http ://www.w3.org/TR/2005/WD-wsdl20-soap11-binding-20050510/.

[Voo94] E. Voorhees. Query expansion using lexical-semantic relations. *In Proceedings of the 17th Annual Internatinal ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, 1994. Dublin (Ireland).

[WIY99] D.H. Widyantoro, T.R. Ioerger, and J. Yen. An adaptative algorithm for learning changes in user interests. *In Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM'99). ACM Press*, pages 405–412, 1999. New York (USA).

[XC96] J. Xu and W.B. Croft. Query expansion using local an global document analysis. *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, August 1996. Zurich (Switzerland).

[Yag02] R.R. Yager. Fuzzy logic methods in recommender systems. *Fuzzy Sets and Systems. Elsevier*, 136 :133–149, 2002.

[YK02] Benjamin P. C. Yen and Robin C. W. Kong. Personalization of information acces for electronic catalogs on the web. *Electronic Commerce Research and Applications*, 1(1) :20–40, 2002.

[YTA04] J. Yu, J. A. Thom, and L. Alem. Group memory based on the task information. *In proceedings of RIAO'04*, pages 329–346, 2004.

[Zip49] G.K. Zipf. Human behavior and the principle of least effort. *Ed. Addison Wesley Publishing*, 1949.

[ZP04] J. Zhang and P. Pu. Survey of solving multi-attribute decision problems. *EPFL Technical Report No : IC/2004/54*, June 2004.

Index

[FFR96], 77

G

[GGM+02], 77
[GLBD04], 14
[GNOT92], 12, 36, 89
[GSK+99], 14
[GGMT99], 17
[GMM03b], 59
[GMM03a], 59

H

[Hay04], 61, 65
[CBHS05], 79
[HBEV04], 79, 191, 192
[HW04], 37
[HBS02], 34, 45, 73, 82, 117
[HKNH00], 19, 90
[Hol75], 28, 34

J

[Jac98], 35
[JRP01], 11

K

[KC04], 61
[KYS05], 37
[KKPS02], 9
[KV02], 11
[KAC+02], 193
[Kay05], 60
[Kay95], 38
[KC03], 37
[KR03], 8
[KRW+04], 34, 45, 72, 117
[Kob01], 38, 92
[Kob02], 189
[KMM+97], 12, 36
[Kor97], 12
[Kru97], 13
[KGB98], 11
[Khr04], 8, 44

L

[LS99], 62
[Leu02a], 9
[Leu02b], 9
[LV05], 9, 115
[LC99], 17, 19, 35
[LK03], 8
[LC96], 26
[Ler91], 26
[Lie95], 12, 35
[Luh58], 22

M

[Mit03], 9, 78
[Mar02], 18
[Mar04], 70
[CP43], 28, 34
[Mic01], 74
[Mit97], 26, 28
[MT02b], 19
[MT02a], 19
[Mla96], 12, 35
[MLR03], 12
[MdR94], 26
[MG00], 92

N

[NVB+01], 26
[NDB06], 13

N

[NSD+01], 77

O

[OVF05], 79

P

[PSHH04], 68
[PBH+03], 77
[PFL+02], 22
[Paz99], 13, 14, 39, 41
[PMB96], 12, 19, 35, 39, 41
[PDAB05], 77
[PSC+02], 11, 18, 37, 41
[Por80], 22
[Pri04], 78
[PS05], 126, 129, 192

Q

[Qui93], 26

R

[Ref00], 51
[Rij79], 8, 10, 21
[RJ76], 24
[RW97], 24
[RWB99], 24, 25

[Roc71], 11
[RP97], 12, 89

R

[RSB98], 77

S

[SBF98], 76
[Sed98], 49
[SD01b], 23
[SM03], 8
[SM83], 21, 30, 31
[SB90], 24
[SMLP01], 44
[SAW94], 44, 82, 117
[Sea04], 193
[SSH97], 27
[Sha00], 36, 38
[SMB97], 24, 25
[SD01a], 193
[Sow84], 34
[Jon72], 23
[SLP04], 8, 73

T

[TBMM04], 53
[Tch04a], 19
[Tch04b], 88
[Tch05a], 96
[Tch05b], 93
[Tch05c], 88
[Tch06], 93
[Tma02], 35
[TC91], 33

V

[Ved05], 9, 78
[Voo94], 11

W

[WIY99], 19, 90

X

[XC96], 11

Y

[Yag02], 14
[YK02], 36
[YTA04], 77

Z

[ZP04], 39
[Zip49], 22