Introduction générale

Dans le domaine de la représentation des connaissances, les ontologies ne sont considérées que relatives aux différents domaines de connaissances, elles répondent aux problèmes de représentation et de manipulation des connaissances. Les ontologies sont très utilisées dans la représentation des connaissances sur le Web.

De nos jours, l'ontologie matérialise une connaissance experte d'un domaine. Partant du fait que plusieurs connaissances peuvent prendre des représentations différentes, on trouve de nos jours plusieurs ontologies de domaine pour un même champ d'application. Les techniques d'alignement représentent un cadre général, dans lequel plusieurs ontologies hétérogènes peuvent être exploitées et interopérables [1].

Les expériences sur les ontologies montrent de plus en plus clairement qu'elles ne représentent correctement et de façon consensuelle que des domaines réduits. Ainsi, les ontologies de domaines sont développées pour des applications particulières alors que des ontologies de référence tendent à être utilisées pour fédérer les résultats des applications spécifiques [2].

Dans le but d'exploiter et de partager les connaissances des différents SBCs (ontologies) l'alignement d'ontologies consiste à déterminer, par un processus global, les correspondances (similarité) entre deux ontologies, en utilisant ou en mettant en œuvre des solutions aux différents problèmes d'hétérogénéité, et des méthodes qui permettent de calculer la similarité [60].

Motivations

Le Web sémantique à pour but d'exploiter la connaissance formalisée à l'échelle du web. Il est, en particulier fondé sur les ontologies : des structures définissant les concepts et relations utilisés pour représenter la connaissance. Ces concepts sont utilisés pour décrire les services Web sémantiques, annoter les ressources du web (images, textes, son, etc.) ou pour décrire des flux de données.

Le web sémantique est donc fondé sur un ensemble d'ontologies. Il est cependant prévisible que des sources de connaissance différentes utiliseront des ontologies différentes.

Pour aligner deux ontologies hétérogène et distribuée dans le web, il est donc utile de déterminer si deux ontologies sont proches ou non.

Comment trouver les ontologies proches dans le web sémantique (système ouvert) ?

Les ontologies dans le web sont changeables et modifiables (dynamiques) et cela influence l'alignement représenté. Il faut avoir des mises à jour des alignements et du souplesse pour les changements et les modifications.

Comment peut-on gérer ces modifications au niveau de l'alignement ? [3]

À cause de ces motivations nous proposons une approche basée agent logiciel pour fournir un environnement qui fait sélectionner les ontologies de domaine proches distribuées dans le web et déduire des alignements entre ces ontologies en exploitant les différentes réponses reçues à partir de ces SBCs et en fin de fournir une base d'alignement (médiateur) pour l'interopérabilité sémantique entre SBCs du même domaine.

Problématique

Un même domaine peut avoir plusieurs ontologies qui sont représentées de manière différente, alors si, un utilisateur exprime une requête à partir d'une ontologie A, des systèmes à base de connaissances permet de répondre, mais avec des connaissances de ces ontologies de domaine B. De même pour une question donnée nous pouvons avoir deux réponses syntaxiquement différentes, mais sémantiquement équivalentes.

L'objectif sera donc de développer un environnement assurant une interopérabilité sémantique entre les systèmes de même domaine dont leurs ontologies sont représentées de manière différente et distribuée dans le web.

Plan du mémoire

Le mémoire est organisé en deux parties principales : la première partie comporte trois chapitres sur l'état de l'art en lien avec le cadre de notre mémoire.

La deuxième partie regroupe le détail de notre contribution.

Dans le premier chapitre, nous présentons un état de l'art sur les ontologies. Nous détaillons les principales notions sur les ontologies telles qu'ontologies au sein de représentations des connaissances, les composants d'une ontologie, les langages de représentations des ontologies, recherche future pour le Web sémantique et finalement nous présentons les systèmes multi agents et ontologies comme l'apport des SMA sur les ontologies et l'inverse.

Dans le deuxième chapitre, nous présentons la notion alignement d'ontologies basée principalement sur la mesure de similarité. Nous présentons les différentes techniques utilisées pour mesure de la similarité. Ainsi, dans ce chapitre nous illustrons les domaines d'applications d'alignement d'ontologies. Ces applications montrent l'efficacité de cette technologie pour améliorer l'interopérabilité sémantique entre des SBCs réparties.

Le troisième chapitre est consacré à la présentation de quelques approches d'alignement d'ontologies qui sont conçues pour résoudre le problème d'hétérogénéité sémantique entre deux ou plusieurs ontologies de domaines.

Dans la deuxième partie, nous allons décrire notre contribution qui est la proposition d'une approche basée agent pour l'alignement d'ontologies : « Équivalence sémantique ».

Nous avons présenté dans le chapitre quatre l'architecture générale de notre approche basée sur les agents logiciels puis on va décrire le rôle de chaque agent de système.

Ce chapitre contient aussi le processus concernant le déroulement de système et l'interaction générale entre les agents.

Dans le cinquième chapitre, nous poursuivons l'implémentation des principaux modules de notre architecture sur une étude de cas.

Enfin, nous terminons ce mémoire par une conclusion générale, qui récapitule les travaux réalisés et fait le point sur un ensemble de perspectives envisagées.

I.1. Introduction

Les ontologies sont étudiées par les chercheurs travaillant en Intelligence Artificielle, sur la représentation de la connaissance, et maintenant, sur le Web sémantique.

Le terme « ontologie » vient du domaine de la philosophie, où il signifie « explication systématique de l'existence » (de ontos « l'être, ce qui est », et de logos « discours »).

Dans cette perspective, une ontologie est indépendante du langage utilisé pour sa description [6]. De nombreuses acceptions du mot « ontologie » existent (vocabulaire technique, référentiel métier, terminologie/thesaurus, système de classes d'une représentation par objet, base de connaissances terminologique…).

Bien que la communauté d'I.A. semble s'accorder sur l'utilisation et le sens du mot « ontologie », il n'existe pas de définition formelle unanimement acceptée. Une ontologie est vue comme un moyen de « décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances » [8].

Cette définition est une extension de celle de Gruber [13], décrivant une ontologie comme

« **Une spécification explicite d'une conceptualisation** ». Une ontologie est une modélisation des connaissances du monde, d'informations extra linguistiques. Organisée en un réseau de concepts, elle consiste en un ensemble de définitions de catégories de base (objet, relations, propriétés) qui permettent de décrire les objets du domaine que l'on traite, leurs propriétés et les relations qu'ils entretiennent les uns avec les autres [9].

On les appelle aussi bases de connaissances (knowledge base). Ce modèle du monde est idéalement indépendant des langues traitées par un système ; les mots décrits dans les lexiques de diverses langues pointeront vers les concepts de l'ontologie qu'ils expriment.

Les ontologies ont pour but de saisir la connaissance dans un domaine, d'une façon générale et de fournir une représentation communément acceptée qui pourra être réutilisée et partagée par diverses applications et groupes. Les ontologies fournissent un vocabulaire commun dans un domaine et définissent à différents niveaux de formalisation la signification de termes et leurs relations [8].

« Une ontologie peut prendre une grande variété de formes, mais elle inclura nécessairement une liste des termes et des spécifications sur leurs significations. Cela comprend des

définitions et des indications sur la façon dont les concepts sont reliés entre eux, imposant une structure au domaine et contraignant les interprétations possibles des termes. » [10].

Une ontologie est donc un « modèle conceptuel spécifique élaboré dans le domaine de la gestion du savoir. Une ontologie peut représenter des relations complexes entre des objets et inclure les règles et axiomes manquants dans un réseau sémantique. Une ontologie qui décrit le savoir dans un domaine précis est souvent reliée à des systèmes de prospection des données et de gestion des connaissances. » selon [11].

Les principaux buts, dans le domaine de conception des ontologies, sont de faire des ontologies partageables en développant des formalismes et des outils communs, de développer le contenu des ontologies (conception d'ontologie), et de comparer, de rassembler, de traduire et de constituer diverses ontologies. Le travail récent en conception d'ontologie a produit une gamme des projets divers, des ontologies représentant la connaissance du monde générale, aux ontologies de domaines spécifiques en passant par les ontologies de système de représentation de connaissances qui englobent les « cadres ontologiques ». La communauté de l'« ingénierie ontologique » s'accorde à dire qu'il serait bénéfique de pouvoir intégrer des ontologies de façon à partager et réutiliser les connaissances de chacun [12].

Dans ce chapitre nous allons voir un état de l'art en matière d'ingénierie ontologique, en particulier les acquis du domaine et les nombreux problèmes restant à traiter. Les différents types d'ontologies selon le degré de formalisme, les objets modélisés, la granularité et les ontologies au sein du processus de représentation des connaissances, les composants d'une ontologie et les formalismes de représentation des ontologies sont les sujets de la première partie. La deuxième partie expose les outils pour les ontologies comme les langages de constructions d'ontologies, les moteurs d'inférence, langages d'interrogation d'ontologies et le cycle de vie des ontologies avec ces différentes étapes. Finalement, l'évaluation et l'évolution d'une ontologie, l'utilisation des ontologies et l'apport des systèmes multi agents sur les ontologies et l'inverse clôture ce chapitre.

I.2. La notion d'ontologie

I.2. 1. Définition

Une définition des ontologies consensuelle, précise et complète dans le contexte du Web sémantique n'existe pas encore. Cependant, la définition la plus citée est celle de [13]: « Une ontologie est une spécification explicite et formelle d'une conceptualisation d'un domaine de connaissance ». La conceptualisation est le résultat d'une analyse ontologique du domaine

étudié et donc est l'abstraction du monde de ce domaine. Cette conceptualisation est représentée dans une forme concrète, donc spécification, où les concepts, les relations entre eux et les contraintes pour les employer sont explicitement définis dans un format et langage formel.

Dans notre contexte [Tom Gruber] « Une ontologie est une **spécification formelle**, et **explicite** d'une **conceptualisation partagée** » une ontologie d'un domaine de connaissance informellement, est considéré, comme un vocabulaire commun des termes, qui est bien contrôlés, organisés, cohérents, et partagés. Ces termes correspondent aux concepts et relations entre eux dans le domaine. Les termes sont organisés dans une structure hiérarchique formée par des liens de « spécialisation» (subsomption ou is_a) entre des termes de concepts ou entre des termes de relation. Ce vocabulaire commun est partagé entre des agents qui collaborent dans le domaine en question. Cela assure une interprétation cohérente, précise entre les agents.

On distingue généralement deux entités globales au sein d'une ontologie. La première, à objectif terminologique, définit la nature des éléments qui composent le domaine de l'ontologie en question, un peu comme la définition d'une classe en programmation orientée objet définit la nature des objets que l'on va manipuler par la suite. La seconde partie d'une ontologie explicite les relations entre plusieurs instances de ces classes définies dans la partie terminologique.

Ainsi, au sein d'une ontologie, les concepts sont définis les uns par rapport aux autres , ce qui autorise un raisonnement et une manipulation de ces connaissances.

I.3. Les différents types d'ontologies

Uschold M. [14] détermine trois dimensions pour la classification des ontologies:

Le degré de formalisme du modèle de représentation, le sujet (les objets modélisés) domaine de connaissance, connaissances de raisonnement, connaissances liées au modèle de représentation.

Et enfin, l'objectif opérationnel : communication entre utilisateurs, interopérabilité entre systèmes, application à un problème d'ingénierie (comme la réutilisation de composants), résolution de problèmes…

I.3.1. Selon le degré de formalisme

M. Uschold et M. Grüninger ont identifié quatre types d'ontologies: les ontologies informelles, les ontologies semi formelles et les ontologies rigoureusement formelles [14].

I.3.1.1. Les ontologies hautement informelles : exprimées en langage naturel

I.3.1.2. Les ontologies semi informelles : elles sont exprimées sous une forme limitée, restreinte et structurée du langage naturel (en utilisant des modèles), c'est à dire des patrons ont été mis en oeuvre.

I.3.1.3. Les ontologies semi formelles : exprimées dans un langage défini artificiellement et formellement.

I.3.1.4. Les ontologies rigoureusement formelles : exprimées dans un langage contenant une sémantique formelle, des théorèmes et des preuves de propriétés telles que la robustesse, l'exhaustivité, la complétude et la consistance. La plupart des ontologies sont aujourd'hui implémentées en langage formel (Ontolingua, CycL, Loom, Flogic). Une classification de ce type d'ontologies peut se définir soit en «intension » ou en « extension » :

- Une ontologie formalisée en extension [13], a sa définition formelle avec une sémantique déclarative (vocabulaire, thésaurus, etc.). Elle est constituée d'un ensemble de «lexons », expressions élémentaires construites d'un élément de contexte, de terme et de rôle.
- Une ontologie formalisée en intension [15] possède une définition à travers des «mondes possibles» (thésaurus de domaine vus comme une synthèse organisée des arrangements de termes possibles linguistiquement).

I.3.2. Selon les objets modélisés

Les ontologies ont été aussi regroupées dans [16] en se basant sur les objets modélisés par l'ontologie afin de répondre à un but précis (voir la figure I.1).

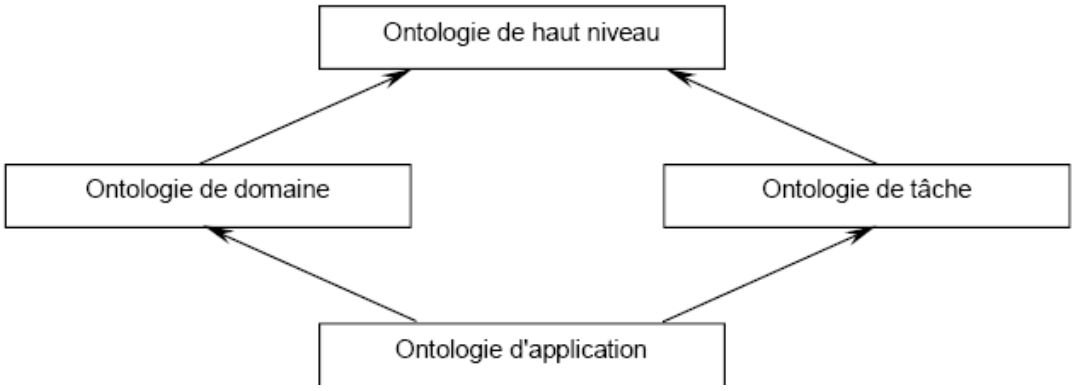


Figure I.1 : Classification des ontologies selon N. Guarino [15].

I.3.2.1. Les ontologies supérieures : Dites aussi de haut niveau « top level ontologies ou Upper Level ontologies **»,** ontologies « génériques ou noyaux d'ontologies », « méta ontologie», «ontologies de sens commun/général » ; elles sont universelles, réutilisables et

référençables à partir des concepts des autres niveaux d'ontologies (voir la figure I.2). Elles comportent des concepts abstraits (généraux) subsumant les concepts existant dans les différents domaines.

Une ontologie de haut niveau est généralement conçue afin de réduire les incohérences des termes définis plus bas dans la hiérarchie. Il n'existe pas pour le moment d'ontologies de haut niveau unifiées.

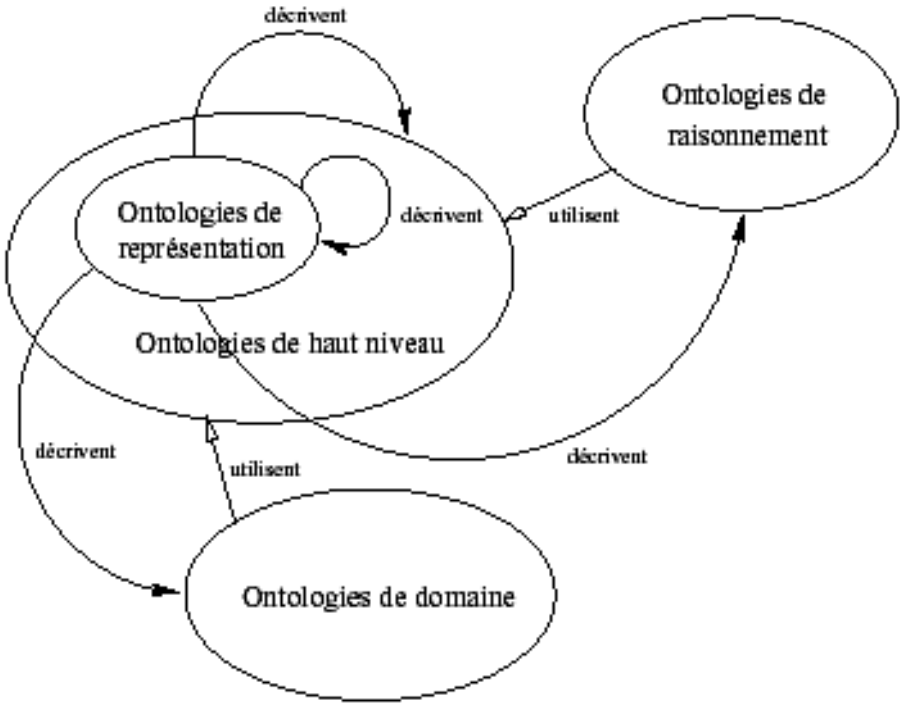


Figure I.2 : Schéma représentatif des relations entre les ontologies de domaine, de raisonnement, de représentation et de haut niveau. [17].

I.3.2.2. Les ontologies de domaine : ce type décrit un vocabulaire appartenant à un domaine générique donné tel que la médecine. Elles ne sont pas propres à une tâche précise et présentent une bonne précision et se rapportent à un certain type d'artefacts.

I.3.2.3. Les ontologies de tâche : spécifiques à une tâche générique, telle que la vente, et indépendamment du domaine d'application.

I.3.2.4. Les ontologies d'application : Correspondent à l'exécution d'une tâche particulière et leur domaine d'application est restreint. Elles sont souvent des spécialisations des ontologies de domaine et des ontologies de tâche.

I.3.2.5. Ontologies de représentation : Ce type d'ontologies est un cas particulier d'ontologies supérieures qui regroupe des concepts déjà utilisés pour formaliser les connaissances. Indépendamment des domaines [18] puisqu'elles décrivent des primitives cognitives communes. Parmi les ontologies de représentation, on trouve la « Frame-Ontology» qui définit,

de manière formelle, les concepts utilisés principalement dans des langages à base de frames : classes, sous classes, attributs, valeurs, relations et axiomes [13].

I.3.2.6. Les ontologies de raisonnement : regroupe les processus de raisonnement appliqués aux connaissances qui forment eux-mêmes un domaine de connaissance. On parle particulièrement d'ontologies développées pour représenter des connaissances génériques mises en oeuvre lors de la résolution automatique de problèmes.

I.3. 3. Selon la granularité

La classification suivante est en fonction du degré de granularité, c'est-à-dire quel niveau de détail des objets de la conceptualisation est préconisé.

En fonction de l'objectif opérationnel, une connaissance plus ou moins fine du domaine est nécessaire et des propriétés considérées comme accessoires dans un contexte peuvent se révéler indispensables dans un autre :

I.3.3.1. Granularité fine : correspondant à des ontologies très détaillées, possédant ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche **[17]**.

I.3.3.2. Granularité large : correspondant à un vocabulaire moins détaillé. Les ontologies de haut niveau ont une granularité large, du fait que les notions sur lesquelles elles portent peuvent être raffinées par des notions plus spécifiques **[17]**.

I.3.4. Les ontologies au sein de représentations des connaissances

Cela met en place une classification d'ontologie sur la base de la quantité et le type de structure de conceptualisation, desquels il définit des ontologies d'information telles que les schémas de bases de données, des ontologies de modélisation de connaissances et des ontologies terminologiques telles que les lexiques [19]. C'est ce dernier type que l'on peut considérer comme la base des représentations du langage. Une ontologie est semblable à un dictionnaire ou à un glossaire, mais avec plus de détails et une structure qui permet à des ordinateurs de traiter son contenu.

Les ontologies sont classifiées ici sur la base de leur force d'expression, c'est-à-dire sur la base de l'information que l'ontologie doit exprimer.

Tamma **[36]** et D. McGuinness **[20]** considèrent dans leur taxonomie que **les Lexiques contrôlés** (une liste finie de termes, un ensemble de sens lexicaux associés à des traits

syntaxiques, morphologiques et sémantiques) , les **Glossaires** (des listes de termes avec leurs significations), les **Thesauri** (ajoutant aux glossaires la sémantique ressortant des définitions des relations entre les termes tels que la relation de synonymie et ne fournissant pas de structure hiérarchique explicite) et les **Hiérarchies Is-a informelles** (une notion vague de généralisation et de spécialisation bien que ce ne soit pas une hiérarchie stricte de sous-classes) sont des ontologies avec un degré d'expression plus ou moins faible, ne répond pas la définition d'une ontologie.

La figure I.3 montre une échelle des degrés de formalisation et du niveau d'engagement sémantique qui permet de préciser le sens des concepts sémantiques de manière non ambiguë, à travers des principes différentiels **[22]** en représentation de connaissances.

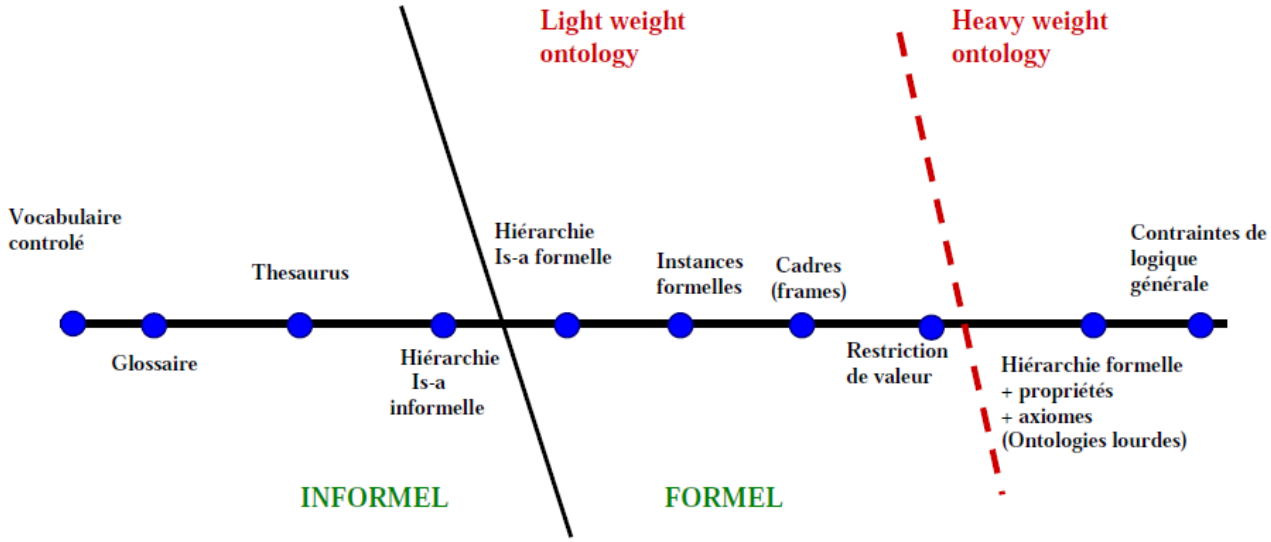


Figure I.3 : Les différents niveaux de formalisme et d'engagement sémantique en représentation des connaissances **[20]**.

Cette échelle va du vocabulaire contrôlé aux ontologies lourdes (heavy-weight ontologies), comprenant les termes désignant les concepts et relations du domaine, leurs propriétés conceptuelles, et toutes les connaissances nécessaires à la description de la sémantique du domaine.

Une autre taxonomie **[20]** considère que la famille des ontologies comprend :

I.3.4.1. Les hiérarchies Is-a formelles

Ici, les concepts sont organisés selon une hiérarchie de sous-classes stricte**.** Le concept d'héritage est toujours applicable dans ce type d'ontologies qui peut inclure uniquement des noms de classe. Elle appartient à la catégorie des ontologies légères.

I.3.4.2. Les instances formelles

Les ontologies incluant des relations d'instances formelles sont une extension naturelle des ontologies appliquant une structure de hiérarchie stricte.

I.3.4.3. Les cadres

Les cadres introduisent des propriétés caractéristiques pour décrire les concepts. Chose intéressante dans la mesure où l'on peut appliquer le principe d'héritage sur ces propriétés.

I.3.4.4. Les restrictions de valeur

Ces ontologies permettent d'appliquer des restrictions aux valeurs associées aux propriétés (Nombre maximum de noms pour décrire le concept).

I.3.4.5. Les contraintes de logique générale

Ces ontologies sont celles qui ont la plus grande force d'expression et qui sont de ce fait écrites dans un langage d'ontologie très expressif (tel que Ontolingua). Ces ontologies peuvent, par exemple, être basées sur des équations mathématiques qui utilisent des valeurs d'autres propriétés pouvant être exprimées comme des énoncés logiques.

I.4. Les composants d'une ontologie

Une ontologie débute par une taxinomie, un arrangement structuré d'informations en classe qui catégorisent un sujet et ses dépendants **[4]**.

La taxinomie est la partie centrale de la plupart des ontologies. Cependant, l'organisation taxinomique peut varier d'une façon très importante elle aussi.

Selon Gruber [13], la formalisation d'une ontologie se met en place grâce à 5 types de composants (« Modelling primitives »).

I.4.1. Les classes / les concepts

Une classe ou un concept représente un type d'objet dans l'univers. Les classes sont habituellement organisées en taxinomies auxquelles on applique des mécanismes d'héritage.

Tous les concepts peuvent être organisés en une large taxinomie. Il peut aussi y avoir un grand nombre de hiérarchisations plus petites, ou bien pas de taxinomie explicite du tout.

Les concepts sont utilisés dans leur sens large. Ils peuvent être abstraits ou concrets, élémentaires (électrons) ou composés (atome), réels ou fictifs.

Il arrive que les définitions des ontologies aient été diluées, en ce sens que les taxinomies sont considérées comme des ontologies complètes.

En résumé, un concept peut être tout ce qui peut être évoqué et, partant de là, peut consister en la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc.

Selon [21] Il est souvent fait référence aux concepts en tant qu'union de classes et d'instances, alors que chacun des constituants de l'ontologie est considéré comme un fragment de connaissance (knowledge pièce).

I.4.2. Les relations

Les relations représentent un type d'interaction entre les notions d'un domaine. Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensemble. Des exemples de relations binaires sont sous-classe-de, connecté-à.

I.4.3. Les rôles

Selon Sowa [23], « un rôle caractérise une entité par quelque rôle qu'elle joue dans sa relation à une autre entité. Le type « Humain », par exemple, est un type de phénomène qui dépend de la forme interne de l'entité ; mais la même entité peut être caractérisée par des rôles du type, Mère, Employé ou Piéton. ».

I.4.4. Les fonctions

Les fonctions sont aussi des cas particuliers de relations dans lesquelles le $n_{ième}$ élément de la relation est défini à partir des n-premiers. Comme exemple de fonctions binaires il y a la fonction mère-de ou carré-de, comme fonction ternaire, le prix d'une voiture usagée sur lequel on peut se baser pour calculer le prix d'une voiture d'occasion en fonction de son modèle, de sa date de construction et de son kilométrage.

I.4.5. Les axiomes

Les axiomes sont utiles à la structuration de phrases qui sont toujours vraies. Ils permettent de contraindre les valeurs de classes ou d'instances.

I.4.6. Les instances

Les instances sont utilisées pour représenter des éléments dans un domaine.

I.5. Les formalismes de représentation des ontologies

De nombreux formalismes ont été développés pour représenter les connaissances, de la logique des prédicats jusqu'aux langages sophistiqués basés sur des structures de données appelées schémas.

I.5.1. Les formalismes logiques

Dans une représentation logique, la base de connaissances consiste en un ensemble d'axiomes décrivant une situation, un état de choses, sur lesquels des règles d'inférence opèrent et fournissent de nouvelles formules que l'on peut considérer comme valides. Celles-ci constituent alors de nouveaux états de choses dans la base. Le langage de programmation Prolog est fondé directement sur ces principes.

I.5.2. Les réseaux sémantiques

Un réseau sémantique est un modèle de représentation du contenu sémantique des concepts sous forme de graphe. Un graphe est formé de noeuds, représentant les concepts, reliés par des arcs décrivant les relations entre eux.

En I.A., Quillian fut le premier à développer de tels réseaux en tant que modèles de la mémoire associative humaine. Actuellement, la théorie des graphes conceptuels de [23], représentant les relations sémantiques, constitue le formalisme le plus répandu pour conceptualiser les ontologies.

Les concepts y sont organisés en un treillis de types, reliés par une relation d'ordre correspondant au lien *sorte-de*.

I.5.3. Les primitives

Les primitives ont été utilisées dans la théorie de la dépendance conceptuelle de Schank en 1972. Elle propose un modèle de représentation conceptuelle fondé sur l'organisation de la mémoire et insiste sur l'universalité des primitives qui reflètent la pensée plutôt que la langue. Étant universelles, les primitives sont définies d'une façon très générale et en petit nombre, ce qui pose problème pour la définition de domaines spécifiques. Schank distingue six classes de concept,

dont deux pour le contexte ; objets physiques, actions, attributs d'objets, attributs d'actions, lieu et temps [4]. De même, il définit onze actions de base permettant de représenter toutes les actions possibles et six rôles conceptuels décrivant les relations entre entités et actions.

I.5.4. Les schémas

Un schéma est une structure de données qui regroupe un ensemble d'informations concernant un concept particulier. Les concepts sont généralement organisés en treillis suivant la relation sorte-de. À chaque noeud est associé un schéma qui décrit les propriétés du concept dont héritent les concepts descendants.

La notion de schéma dont on trouve l'origine chez Minsky en 1974 a été reprise en I.A. pour développer des formalismes de représentation des connaissances, par exemple les langages KRL (1977) et KL-ONE (1985). Ces formalismes intègrent souvent des stratégies d'inférence spécifiques, comme le raisonnement par défaut et l'héritage automatique des propriétés en suivant les arcs d'une hiérarchie de concepts.

Définition d'un « frame » donnée par Minsky :

« Un frame est une structure de données représentant une situation stéréotypée, comme se trouver dans un certain type de salon ou se rendre à un goûter d'anniversaire d'un enfant.

Divers types d'informations sont associés à chaque frame. Certaines d'entre elles concernent l'utilisation de ce frame. D'autres portent sur ce que l'on s'attend à ce qu'il arrive par la suite. D'autres encore portent sur ce qu'il faut faire si ces attentes ne sont pas confirmées. »

I.5.5. Les scripts

La notion de scripts ou scénarios a été introduite par Schank et Abel en 1977, sur le modèle des frames pour le traitement du langage naturel. Les frames servent alors à représenter des séquences d'actions stéréotypées appelées scénarios/scripts. Un scénario consiste en un ensemble d'actions élémentaires, ou de références à d'autres scénarios, ordonnées selon leur déroulement dans le temps.

Les scripts décrivent la chronologie et décompose les procédures. On y décrit les objets et les acteurs, les conditions initiales et les résultats, des scènes de la vie courante. Le traitement d'un texte consiste à reconnaître un événement décrit en accédant au scénario adéquat et à l'interpréter, c'est-à-dire à en extraire des prédictions (ou résultats). Reste le problème des situations inattendues ainsi que des situations faisant intervenir plusieurs scénarios simultanément.

[19] définit d'ailleurs sa typologie sur le sujet de la conceptualisation et sur le type de représentation utilisé dans la conception d'ontologie. Ses « ontologies de représentation » regroupent les primitives de représentation utilisées dans la formalisation des connaissances dans les paradigmes de représentation de connaissances. (Ex : Frame-ontolgy dans Ontolingua Server).

I.6. Outils pour les ontologies

Pour que les différents systèmes puissent utiliser et échanger des informations, il faut spécifier un modèle pour celles-ci. Différents langages, du simple XML au riche OWL+SWRL, sont utilisés pour modéliser l'information. Ces derniers permettent d'ajouter une sémantique plus ou moins forte à l'information. Une fois modéliser ces ontologies peut être exploitées par des moteurs d'inférences et interrogées par des langages de requêtes.

I.6.1. Langages de construction d'ontologies

Pour que les différents agents logiciels puissent utiliser et échanger des informations, il faut spécifier un modèle pour celles-ci. Différents langages, du simple XML au riche OWL+SWRL, sont à l'heure actuelle utilisés pour modéliser l'information.

Ces derniers permettent d'ajouter une sémantique plus ou moins forte à l'information. Une fois modéliser nos ontologies peut être exploitées par des moteurs d'inférences et interrogées par des langages de requêtes.

Dans cette partie, nous présentons différents langages pour créer des ontologies.

Ces langages ont été conçus pour répondre notamment aux besoins du web sémantique.

I.6.1.1. Le web sémantique

Depuis peu, l'émergence d'une tendance appelée Web Sémantique [24] a fait son apparition dans le World Wide Web (WWW). À l'inverse du Web d'aujourd'hui, qui est d'abord conçu pour l'interprétation et l'utilisation humaine, le Web Sémantique est une vision d'un Web qui serait non ambiguë et compréhensible par une machine. Le Web sémantique reste entièrement fonde sur le Web "classique" et ne le remet pas en cause.

Cela reste un moyen de publier et consulter des documents, mais les documents traits par le Web sémantique contiennent non pas des textes en langage naturel (français, espagnol, chinois, etc.) mais des informations formalisées pouvant être traitées automatiquement par des agents logiciels. Ceci est réalisé par l'annotation du contenu du Web, par la description de propriétés et de relations. Cette annotation est réalisée avec un langage raisonnablement expressif, possédant une sémantique bien définie et permettant le partage ainsi que la réutilisation des données au travers de différentes applications. Le consortium W3C a défini un cadre général pour ce type

de descriptions se base sur RDF (Ressource Description Framework) pour la représentation de connaissance et OWL (Ontology Web Language) pour la modélisation sémantique de connaissances.

I.6.1.2 eXtended Markup Language et XML Schema

L'eXtended Markup Language [25] (XML) est un langage de description et d'échange de documents structures, issu de SGML (Standard Generalized Markup Language) et défini par le consortium Web. XML permet de décrire la structure arborescente de documents à l'aide d'un système de balises permettant de marquer les éléments qui composent la structure et les relations entre ces éléments. XML ne pose aucune contrainte sémantique sur la description des informations, il ne constitue donc pas un langage de modélisation d'ontologies à lui seul.

Avec XML nous pouvons décrire une Personne Mohamed âge de 37 ans qui habitent zaàtcha rue, Biskra. Pour un agent l'exemple ci-dessous correspond est compris comme une simple arborescence de chaînes de caractères.

```
<Personne id=‘ Mohamed’>
<Adresse> zaàtcha rue, Biskra </Adresse>
<Age>37</Age>
</Personne>
```

XML Schéma [26] (XML-S) est un outil de définition de grammaires caractérisant des arborescences de documents (notion de validité syntaxique). Avec les schémas XML, il est possible de contraindre la structure arborescente d'un document, mais pas la sémantique des informations contenues dans ce document.

I.6.1.3 Resource Description Framework et RDF Schéma

Le Resource Description Framework [27] (RDF) est un modèle pour la représentation de meta-données à propos de ressources. Cette représentation est faite sous la forme d'un triplet:

- Sujet : La ressource que l'on définit
- Prédicat : la propriété de la ressource, qui est une liaison étiquetée et orientée du sujet vers l'objet.
- Objet : La valeur de la propriété pouvant être une autre ressource ou bien un littéral.

RDF est à la base exprime sous la forme d'un graphe oriente, mais on peut le d'écrire par la syntaxe XML. On parle alors de RDF/XML qui par abus de langage peut être appelé RDF. En

prenant l'exemple défini pour XML, nous pouvons définir avec RDF un graphe dont l'agent comprendra comme un concept Mohamed qui possède les propriétés adresse et age.

Si RDF fournit une capacité d'échange de connaissances, il ne permet pas à l'utilisateur de définir le vocabulaire des termes à utiliser, ni d'établir la sémantique des objets utilisés.

```
<rdf :RDF>
<rdf :Description about='Mohamed'>
<rdf :Property about='adresse'>
  zaàtcha rue, Biskra
 </rdf :Property>
<rdf :Property about='age'>
 37
</rdf :Property>
</rdf :Description>
</rdf :RDF>
```

Figure 1.4. Representation RDF/XML de Mohamed.

RDF Schéma [28] ou RDFS est un langage permettant de définir des propriétés sémantiques pour les ressources par un schéma. Dans un schéma on peut définir de nouvelles ressources comme des spécialisations d'autres ressources. Les schémas contraignent aussi le contexte d'utilisation des ressources. Avec RDFS de nouvelles notions sémantiques apparaissent.

La principale est la distinction entre une classe (concept d'une ontologie) et une instance (individu d'une ontologie). Voici d'autres notions définies dans RDFS :

– La notion de classe (rdfs : Class) qui peut être rapprochée de la notion de concept d'une ontologie.

– La notion de sous-classe (rdfs :subClassOf) qui est une spécialisation d'une classe déjà définie.

– La notion de type (rdf :Type) : les instances d'une classe propriété et sous propriété (prédicat ou rôle de l'ontologie)

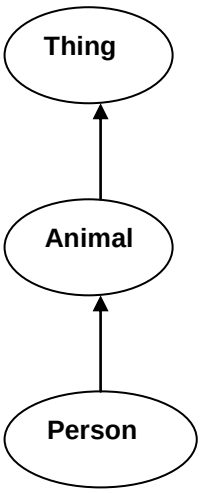
– Les notions de source (rdfs :domain) et de cible (rdfs :range) d'une propriété.

La hiérarchie de subsomption ou taxinomie peut être décrite en utilisant la propriété rdfs :subClassOf. Cette hiérarchie définit les relations de spécialisation d'une classe Ressource par rapport a une classe Objet. Cette relation s'applique également aux relations par la propriété rdfs :SubPropertyOf.

Soit P une propriété de rdfs :range R et de rdfs :domain D, alors une propriété S est une rdfs :SubPropertyOf de P si : le rdfs :range de S est R ou une sous-classe de R et le rdfs :domain de S est D ou une sous-classe de D.

Sur l'exemple de Mohamed, avec le schéma RDF nous définissons le concept de Personne(figure 1.4) avec les types des objets de chaque propriété, une taxinomie de concepts ou hiérarchie de subsomption ainsi que l'instance Mohamed (figure 1.5). Nous avons vu que RDF et RDFS permettent de définir sous la forme d'un graphe de triplets des données ou des meta-donnees. Cependant, il est impossible de raisonner sur ces représentations car la sémantique (hors subsomption) reste très limitée. C'est ce manque de sémantique que

```
<rdf :RDF>
<rdfs :Class rdf :about='Animal'>
<rdfs :subClassOf rdf :resource='Thing'/>
</rdfs :Class>
<rdfs :Class rdf :about='Personne'>
<rdfs :subClassOf rdf :resource='Animal'/>
</rdfs :Class>
<rdf :Property about='age'>
<rdfs :domain rdf :resource='Animal'/>
<rdfs :range rdf :resource='xsd :integer'/>
</rdf :Property>
<rdf :Property about='adresse'>
<rdfs :domain rdf :resource='Personne'/>
<rdfs :range rdf :resource='xsd :string'/>
</rdf :Property>
</rdf :RDF>
```



Taxinomie des concepts

Figure I.5 – Représentation RDFS de Mohamed

I.6.1.4. Ontology Web Language

OWL (Ontology Web Language[29]) est un langage fonde sur la syntaxe RDF/XML et héritier des travaux de DAML+OIL. OWL n'est pas simplement une extension de RDF, il introduit l'aspect sémantique lui manquant, comme les outils de comparaison de propriétés et de classes (identité, équivalence, contraire, cardinalité, symétrie, etc.). Ainsi par ses primitives

plus riches, OWL offre une capacité d'interprétation plus grande à la machine que RDF et RDFS.

OWL se décompose en trois sous-langages OWL Lite, OWL DL et OWL Full, offrant des capacités d'expression croissantes et donc destine à des différentes utilisations.

OWL Lite est le sous langage de OWL le plus simple, car son utilisation est contrainte.

Voici la liste des constructeurs proposent par OWL Lite :

Catégories	Constructeurs
Axiome de Classe	oneOf (enumeration), dataRange, disjointWith
Expressions booléennes	unionOf, complementOf
Cardinalites (0,n) Individu cible d'une propriété	minCardinality, maxCardinality, cardinality hasValue

Table I.1. Liste des constructeurs proposés par OWL Lite.

OWL Full est la version la plus complexe d'OWL, mais également celle qui permet le plus haut niveau d'expressivité. Son utilisation n'est contrainte que par le langage RDF, mais elle ne garantit pas la complétude et la décidabilité des calculs lies à l'ontologie.

La syntaxe ne subit pas de modification par rapport à OWL DL mais OWL Full permet une utilisation sans contrainte sur l'utilisation des constructeurs.

Ex. : utiliser un concept a la place d'un individu Adulte : IntersectionOf (Personne, hasValue(age,Majorite)), Majorite étant un concept remplaçant l'individu de type Age et de valeur 18.

Les 3 niveaux d'OWL présentent une hiérarchie sur la validité des ontologies :

– une ontologie OWL Lite valide est également une ontologie OWL DL valide

– une ontologie OWL DL valide est également une ontologie OWL Full valide

OWL est un langage base sur du XML, nous permettant de décrire de manière riche des informations. Nous présentons dans la prochaine partie les logiques de description qui forment la base théorique des langages d'ontologie comme OWL.

I.6.2. Les logiques de description

Les logiques de description [30] ou DL (Description Logics), appelées aussi logiques terminologiques, sont une famille de formalismes conçue pour décrire et raisonner sur les connaissances d'un domaine.

Comme ses prédécesseurs, les schémas (ou frames [31]) et les réseaux sémantiques, les logiques de description ont pour but d'obtenir une représentation plus proche de la représentation humaine que les formules de la logique des prédicats. Néanmoins, toute formule écrite en DL, peut être réécrite en formule de la logique des prédicats (PL).

Il existe de nombreuses logiques de descriptions offrant plus ou moins d'expressivité.

Plus une logique de description est expressive, plus sa complexité est élevée. C'est pourquoi nous pouvons distinguer deux générations de logiques de description en se basant sur leur calculabilité et leur expressivité ([32]) :

"Les premiers travaux sur les LD commencèrent au début des années 1980 avec des

Systèmes à base de connaissances telles que KL-ONE, BACK et LOOM [30]. Ces premières implantations résolvent des problèmes d'inférence en temps souvent polynomial (indécidable pour KL-ONE [33], par le biais d'une catégorie d'algorithmes de vérification de subsomption de type normalisation/comparaison (structural subsumption algoritms). Ces algorithmes ne s'appliquent qu'à des LD peu expressives, sans quoi ils sont incomplets, c'est-à-dire qu'ils sont incapables de prouver certaines formules vraies."

"Dans les années 1990, une nouvelle classe algorithmes est apparue : les algorithmes de vérification de satisfiabilité à base de tableaux tableau-based algorithms. Ces derniers raisonnent sur des LD dites expressives ou très expressives, mais en temps exponentiel. Cependant, en pratique, le comportement des algorithmes est souvent acceptable [30].

L'expressivité accrue a ouvert la porte à de nouvelles applications telles que le Web sémantique [30]. Le terme logiques de description expressive (LDE) désigne l'ensemble des LD qui ont émergé pendant cette période."

Les logiques permettent de définir des connaissances suivant deux niveaux de description.

Le premier, le niveau terminologique ou TBox, décrit les concepts et les rôles entre les concepts d'un domaine alors que le second, le niveau assertionnel ou ABox, décrit les individus. Un individu est une assertion de concepts identifie par un nom unique et pouvant avoir des rôles avec d'autres individus. Par exemple : Tout employé est une personne (Employé v Personne) sera définie dans la TBox, alors que Mohamed est un employé (Mohamed : Employé) sera défini dans la ABox .

I.6.3 Les moteurs d'inférences

La plupart de ces moteurs sont conçus pour raisonner sur les logiques de descriptions, mais acceptent en entrée des fichiers OWL. Une fois l'ontologie chargée, ces moteurs effectuent les inférences sur la TBox et la ABox.

Nous présenterons plus en détail dans la suite de ce mémoire Pellet et Racer qui sont deux moteurs d'inférence, permettant le raisonnement sur la ABox et la TBox et exploitent des ontologies possédant un niveau d'expressivité en logique de description et en OWL satisfaisant.

I.6.3.1 Racer

Racer [34] est le moteur d'inférence le plus connu et l'un des plus utilises dans le domaine pour ces performances et sa stabilité. Il est commercialisé par Racer Systems GmbH & Co. KG, fonde en 2004 par Volker Haarslev, Kay Hidde, Ralf Moller et Michael Wessel qui travaillaient à l'université de Hambourg. Racer travaille sur les ontologies modélisées par son langage, mais il accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer. Ce moteur d'inférence possède également son propre langage de requête nRQL (new Racerpro query Language) pour interroger les ontologies sur la ABox et la TBox. Racer possède quelques atouts :

– La documentation sur Racer est importante, provenant des concepteurs et des utilisateurs.

– Racer permet l'utilisation d'un mécanisme d'abonnement à un concept qui permet d'être informé de la création de nouvelles instances de ce concept.

– Il permet de créer avec ce mécanisme d'abonnement un monde ferme local. Dans le monde ouvert, il n'est pas possible de s'abonner au concept de Livre sans Auteur, car un livre peut avoir un auteur décrit dans le futur. Racer permet de le faire. En effectuant préalablement 2 souscriptions la première (A) sur le concept de Livre, la seconde (B) sur le concept de Livre avec au moins un Auteur. L'abonnement sur le concept défini par A − B est donc un abonnement sur les livres sans auteur.

– Racer permet l'ajout d'assertions et d'individus dans les ABox après le chargement de l'ontologie.

– Racer permet l'utilisation de règles SWRL.

Racer possède quelques points négatifs :

– Racer suppose que toutes les propriétés sur les datatypes sont fonctionnelles (pas de valeurs multiples pour un datatype property)

– Racer ne permet pas l'utilisation de type de données utilisateur (type défini par l'utilisateur), car il possède ces propres types de données et il effectue une conversion avec les types de base.

– Racer est un produit commercial, il n'existe pas de version libre d'utilisation. Cependant, il est possible d'obtenir une licence gratuite dans le cadre de la recherche scientifique.

I.6.3.2 Pellet

Le moteur Pellet [35] est beaucoup plus récent. Pellet est un des projets du MINDSWAP Group, un groupe de recherche sur le web sémantique de l'université du Maryland.

Il est disponible en Open Source et offre des évolutions fréquentes. Pellet travaille sur des ontologies décrites en RDF ou OWL et permet les requêtes avec RDQL et SPARQL sur la ABox et la TBox. Comme pour Racer, nous allons présenter quelques atouts et points négatifs de Pellet.

– Pellet possède une documentation pauvre en comparaison de celle de Racer. En effet racer est le plus utilise et donc le plus documente par des particuliers. De plus, la documentation officielle de Pellet reste assez pauvre en comparaison de celle de Racer.

– Actuellement Pellet ne permet pas l'utilisation de règles SWRL (la prochaine version l'inclura).

– Pellet n'offre pas de système de souscription à un concept.

Les atouts de Pellet :

– Pellet est open-source et développe en Java.

– Pellet est un raisonneur OWL DL complet.

– Pellet propose en cas d'incohérence dans l'ontologie des réparations possibles, ainsi qu'une heuristique permettant d'obtenir les informations à ajouter dans l'ontologie pour passer au sous-langage OWL inférieur (**OWL Full > OWL DL > OWL Lite**).

– Pellet permet l'utilisation des E-Connections?? et des types de données utilisateurs.

Une fois modélisée et chargée par un moteur d'inférence, une ontologie peut être interrogée par des langes de requêtes. Suivant les moteurs d'inférence employés le choix du langage de requêtes est restreint. En effet ces langages interrogent l'ontologie en passant par le moteur d'inférence.

I.6.4 Langages d'interrogation d'ontologies

Dans cette partie, nous présentons les trois langages RDQL, SPARQL et nRQL. Ces trois langages sont des langages d'interrogation basés principalement sur la reconnaissance de

graphe RDF. Le principe est de décrire un graphe RDF à l'aide de variables. Les résultats d'une requête étant les valeurs des variables pour lesquelles il existe un graphe RDF dans l'ontologie correspondant au graphe défini par la requête. RDQL et SPARQL sont les langages d'interrogation utilisables sur Pellet et nRQL est le langage d'interrogation de Racer.
RDQL (RDF Data Query Language[29]) est un langage d'interrogation de données définies en RDF. Ce langage n'est pas standardiste, il existe de nombreuses implémentations bien que la soumission W3C définit une base commune. Sa syntaxe est très proche de SQL :

SELECT variable [, variable]*
FROM documents rdf [, documents rdf]*
WHERE modèle de triplets
AND restrictions booléennes
USING définition des raccourcis

– La clause SELECT définit la liste des variables que l'on désire obtenir. Une variable est composée de caractères alphanumériques avec le ‘ ’ et commence par un ‘ ?’. Les variables d'une requête sont séparées par un espace (et/ou) une virgule. La valeur * signifie que l'on désire obtenir l'ensemble des variables utilisées dans la requête.
– La clause FROM définit l'emplacement des documents RDF utilise pour la requête.
– La clause WHERE définit les triplets RDF (sujet - prédicat - objet), les éléments de ce triplet sont décrits soit par les valeurs de l'ontologie interrogée soit par des variables.
Plusieurs triplets peuvent être définis dans une même clause WHERE, cette succession de triplets indiquant la conjonction de chacun des termes.
– La clause AND définit les restrictions booléennes de la requête. Une restriction booléenne est constituée de valeurs ou variables compose a l'aide d'opérateurs :
– opérations numériques : +, −, _, /
– conjonction, disjonction : ||, && .
– comparateurs sur les numériques : <, >, <=, >=, = = .
– comparateurs sur les chaînes de caractères : EQ pour l'égalité, NE pour la différence.
– La clause USING permet l'utilisation de simplification de nommage a d'alias
WHERE (?email, <http ://www.w3.org/2001/vcard-rdf/3.0/EMAIL>, ”Dujardit@lifl.fr”)
WHERE (?email, vcard :EMAIL, ”Dujardit@lifl.fr”)
USING vcard FOR <http ://www.w3.org/2001/vcard-rdf/3.0/>
Voici un exemple de requête RDQL, qui sélectionne l'age d'une personne dans le document

annuaire.rdf sachant que la personne possède un age inférieur ou égal a 50 ans, que son nom est Dujardin et son adresse email est Dujardit@lifl.fr.

```
SELECT ?name ?email, ?age
FROM <http ://www.lifl.fr/ANNUAIRE/annuaire.rdf>
WHERE ( ?email, vcard :EMAIL, ‘‘Dujardit@lifl.fr’’)
   AND ( (10 + ?age)*2 <= 60/2) && ( ?name EQ ‘‘Dujardin’’) )
WHERE ( ?email, vcard :EMAIL, ‘‘Dujardit@lifl.fr’’)
 USING vcard FOR <http ://www.w3.org/2001/vcard-rdf/3.0/>
```

SPARQL (SPARQL Protocol And RDF Query Language [29]) est une amélioration de RDQL, ce langage est en cours de standardisation au niveau du W3C. SPARQL offre une syntaxe quasi identique à RDQL, mais y ajoute notamment les opérateurs UNION et OPTIONAL dans la clause WHERE. L’opérateur UNION définit la disjonction de triplets RDF.

opérateur OPTIONAL définit des triplets RDF facultatifs pour le résultat de la requête

nRQL (new Racer pro Query Language[34]) est le langage d’interrogation de Racer.

Comme RDQL et SPARQL, nRQL est basé sur la recherche de graphes RDF. Sa syntaxe est proche des deux autres, sauf pour sa notation préfixe des opérateurs Ci-dessous, nous présentons deux requêtes cherchant tous les oncles (variable z écrite ?z ou $?z) dans l’ontologie myOntology. La première requête est en RDQL (la version SPARQL est identique) et la deuxième en nRQL.

```
SELECT ?z from < myOntology >
WHERE ( ?x < aEnfant >?y),( ?z < estFrereDe >?x)
RETRIEVE ($ ?z)
(AND ( $ ?x $ ?y |myOntology#aEnfant|)
($ ?z $ ?x |myOntology#estFrereDe|))
```

I.7. Le cycle de vie des ontologies

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement doit s’appuyer sur les mêmes principes que ceux appliqués en génie logiciel **[5]**.

En particulier, les ontologies doivent être considérées comme des objets techniques évolutifs et possédant un cycle de vie qui nécessite d'être spécifié. Les activités liées aux ontologies sont d'une part des activités de gestion de projet (planification, contrôle, assurance qualité), et d'autre part des activités de développement (spécification, conceptualisation, formalisation) ; s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration [37]. Un cycle de vie inspiré du génie logiciel .Il comprend une étape initiale d'évaluation des besoins, une étape de construction, une étape de diffusion, et une étape d'utilisation. Après chaque utilisation significative, l'ontologie et les besoins sont réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite.

La Figure I.6 [40] représente les différentes activités qui expliquent que le cycle de vie préconisé est un cycle par prototypes : la vie d'une ontologie passe par les états suivants : spécification des besoins, conception (normalisation, formalisation et opérationnalisation) déploiement et diffusion, utilisation, évaluation et enfin évolution et maintenance. Le cycle de vie par évolution de prototypes permet à l'ontologiste de retourner de n'importe quel état à n'importe quel autre si une certaine définition manque ou est erronée. Ainsi, ce cycle de vie permet l'inclusion, le déplacement ou la modification de définitions n'importe quand durant le cycle de vie de l'ontologie. L'acquisition, la documentation et l'évaluation de connaissances sont des activités de support qui sont effectuées pendant la majorité de ces états.

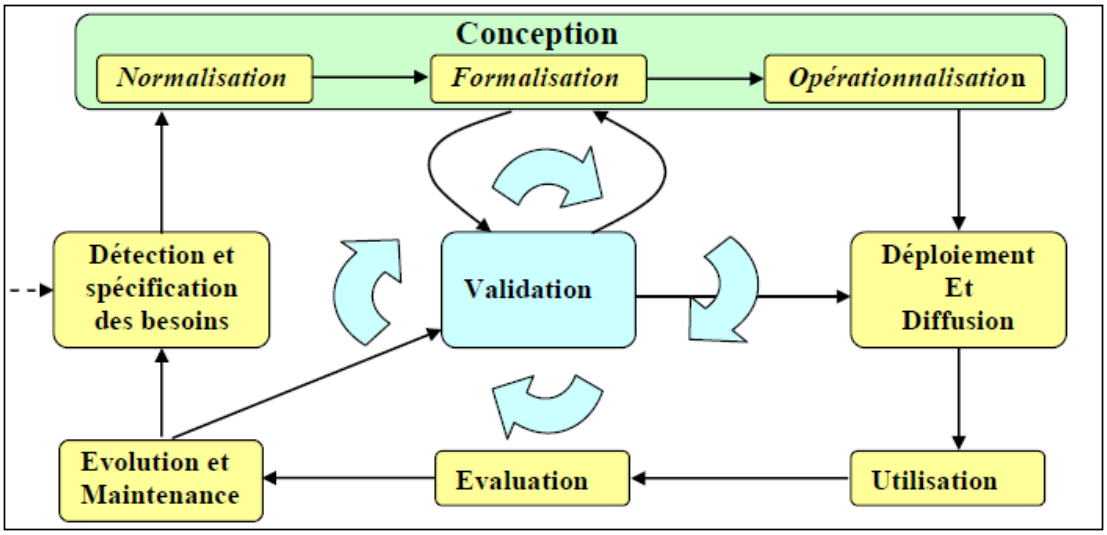


Figure I.6 le cycle de vie d'une ontologie.

Fernandez et ses collègues [39] insistent sur le fait que les activités de documentation et d'évaluation sont nécessaires à l'étape du processus de construction d'ontologie, l'évaluation précoce permettant de limiter la propagation d'erreurs.

I.8. L'évaluation et l'évolution d'une ontologie

L'évaluation d'une ontologie se fait a priori par des tests correspondants à l'objectif opérationnel de l'ontologie **[5]**.

Cette méthode est en particulier préconisée par M.GRUNINGER et M.S.FOX qui proposent d'utiliser des questions de compétences permettant de tester l'ontologie. Si cette dernière répond aux attentes, un système qui l'implémente doit donner les réponses prévues aux questions de compétences [41]. Il est cependant difficile de traduire le but d'une application en quelques questions dont on sera certain qu'elles couvrent l'ensemble du contexte d'usage.

De plus, la validation de l'ontologie en amont de son opérationnalisation est souhaitable. Elle évite de propager des erreurs qui, si les réponses fournies par le système aux questions de compétences se révèlent fausses, peuvent être difficilement repérables. La validité des hiérarchies doit donc être testée dès la phase d'ontologisation, aussi bien du point de vue formel que du point de vue sémantique. La validation formelle consiste à vérifier s'il n'y a pas de cycle, c'est-à-dire de définition en boucle, s'il n'y a pas redondance de concepts ou de relations, si chaque hiérarchie est bien connexe, c'est-à-dire s'il n'y a pas de concept ou de relation isolé des autres et donc sans aucun sens [42]. Des vérifications liées aux choix de modélisation sont également à effectuer, par exemple la détection de l'héritage multiple.
La validation sémantique permet de contrôler que la structure des hiérarchies est correcte vis-à-vis des principes différentiels utilisés. En particulier, la cohérence d'une ontologie vis-à-vis des principes énoncés par B.BACHIMONT peuvent être facilement contrôlée [43]. Les méta-propriétés proposées par C. WELTY et N.GUARINO permettent de vérifier la cohérence sémantique de l'ontologie puisque ces méta-propriétés imposent des contraintes sur les liens de subsomption [44].

Si l'évaluation de l'ontologie a échoué, il est nécessaire de la faire évoluer. Se pose alors le problème délicat consistant à modifier certaines parties de l'ontologie en s'assurant de ne pas provoquer de nouvelles erreurs. Il paraît raisonnable de remonter s'il le faut jusqu'à l'étape de conceptualisation afin d'éviter des modifications qui ne respectent pas la sémantique du domaine. C'est dans ce cadre que les documents produits lors des différentes phases du processus de construction de l'ontologie vont s'avérer précieux.

Le deuxième cas nécessitant l'évolution d'une ontologie est celui où les objectifs changent, c'est-à-dire que le contexte d'usage est modifié, ou que le domaine de connaissance est élargi.

Là aussi, l'ajout de nouvelles connaissances est un processus délicat, compliqué par le fait qu'une ontologie va croître par agrégation de connaissances dans toutes les directions, et non par ajout d'une couche de connaissances [39]. On peut décider soit de construire une nouvelle ontologie avec les connaissances à ajouter et l'intégrer dans l'ontologie déjà constituée, soit d'agréger directement les nouvelles connaissances dans l'ontologie existante. Dans le premier cas, les problèmes auxquels on va devoir faire face sont ceux posés par la fusion d'ontologies, qui reste un thème de recherche encore peu exploré.

I.9. L'utilisation des ontologies

Le champ d'application des ontologies est très vaste ; elles sont utilisées dans les Systèmes à base de connaissance (SBC), où elles peuvent servir de ressources de base pour le raisonnement, de spécification permettant le contrôle de la sémantique d'une base de connaissances ou de conceptualisation commune d'un domaine assurant la communication entre deux agents logiciels ou entre un système et un utilisateur.

Les principaux domaines d'application des SBC sont : **[7]**

- Les systèmes coopératifs qui accompagnent un agent humain dans la réalisation d'une tâche ou la résolution d'un problème en contrôlant la pertinence de ses actions et/ou en proposant des aides.
- Les systèmes de gestion des connaissances qui permettent de sauvegarder, transmettre et partager entre plusieurs utilisateurs des connaissances sur support informatique.
- La résolution automatique de problèmes dans un domaine donné.
- L'enseignement assisté par ordinateur où un système informatique encadre l'apprentissage d'un élève, accompagne ses acquisitions de connaissances, évalue son travail, tout en étant une source de connaissances.

Une des applications phares des ontologies est le Web sémantique, le Web constitue un terrain idéal de mise en oeuvre des ontologies considérées en tant que spécifications partagées de connaissances, les pages Web représentant une énorme masse de connaissances difficilement exploitable sans une assistance automatique.

Cette masse augmente sans cesse ainsi que le nombre d'utilisateurs qui veulent pouvoir trouver facilement les informations qu'ils y recherchent.

L'introduction des ontologies dans le Web, consiste à outiller le Web pour lui adjoindre une couche de connaissances accroissant son efficacité et la puissance des applications qui s'y déploient.

Les pages Web représentant une masse de connaissances aussi énormes que disparates. Cette masse augmente sans cesse ainsi que le nombre d'utilisateurs qui veulent pouvoir trouver facilement les informations qu'ils y recherchent. L'éventail des thèmes traités dans les différentes pages Web est tel qu'une recherche syntaxique par mot-clés retourne quasi systématiquement des pages qui ne portent pas toutes sur le même domaine de connaissances.

L'exploitation efficace des ressources du Web suppose donc que les moteurs de recherche puissent accéder à la thématique de chaque page, et à son sens. De plus, la variété des sources d'information sur le Web (textes, images, etc.) plaide pour un traitement de l'information qui soit indépendant des formes sous lesquelles elle est stockée, c'est-à-dire pour un traitement au niveau conceptuel. Une partie des manipulations d'informations actuellement assurées par les utilisateurs pourra ainsi être prise en charge par les machines.

L'ajout d'une couche sémantique au dessus de la couche HTML, qui ne peut servir qu'à décrire formellement les pages Web, est donc nécessaire. Chaque page doit ainsi intégrer une représentation des connaissances qu'elle contient. De plus, les liens *sémantiques* entre chaque page doivent être spécifiés, ce que ne permet pas de faire l'hypertexte classique. Les différentes applications internent (moteurs de recherche, services, etc.) pourront alors accéder à la sémantique des connaissances intégrées aux pages Web, du moins à une représentation de ces connaissances.

Dans ce cadre, les ontologies vont servir à l'interprétation de ces connaissances en spécifiant la sémantique de la représentation utilisée.

I.10. Système multi agent et Ontologie

Donnons tout d'abord une définition d'un agent ; un agent est, selon [45], une entité physique ou virtuelle, autonome, située dans un environnement, capable de le percevoir, capable d'y agir, de communiquer avec d'autres agents, et éventuellement de se reproduire. De plus, un agent possède un objectif individuel (fonction de satisfaction), des ressources, une représentation partielle de l'environnement, des compétences et il offre des services.

Son comportement est fonction de ses observations, de ses connaissances, de ses croyances, de ses compétences, de ses interactions.

L'autonomie d'un agent peut se définir par trois capacités :

Les agents ont une existence propre, indépendante de l'existence des autres ;

Les agents sont capables de maintenir leur viabilité dans des environnements dynamiques sans contrôle extérieur

La prise de décision interne des agents quant au comportement à avoir est uniquement fonction des perceptions, connaissances et représentations du monde propre aux agents.

donc un agent considéré comme un programme ayant des propriétés particulières ; parmi ces propriétés signale celles qui sont les plus significatives, à savoir l'autonomie, la communication avec d'autres programmes par l'intermédiaire d'envois de messages et le raisonnement à partir d'une base des connaissances et d'événements extérieurs

Les agents peuvent être conçus pour différents types de systèmes caractérisés par les

actions qui seront attendues des agents, leur niveau de raisonnement, leur sociabilité, etc.…

On distingue donc des agents cognitifs, intentionnels, rationnels, réactifs et bien d'autres encore. [46]

Un *système multi-agent* [47] est un système distribué composé d'un ensemble d'agents. Les S.M.A sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de *coopération*, de *concurrence* ou de *coexistence*.

Un S.M.A est généralement caractérisé [47] par :

1. chaque agent a des informations ou des capacités de résolution de problèmes limitées, ainsi chaque agent à un point de vue partiel;

2. il n'y a aucun contrôle global du système multi-agent;

3. Les données sont décentralisées ;

4. le calcul est asynchrone.

Les S.M.A sont des systèmes idéaux pour représenter des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples résolveurs. Ces systèmes possèdent les avantages traditionnels de la résolution distribuée et concurrente de problèmes comme la modularité, la vitesse (avec le parallélisme), et la fiabilité (dûe à la redondance).

Les S.M.A sont à l'intersection de plusieurs domaines scientifiques : informatique répartie et génie logiciel, intelligence artificielle, vie artificielle. Ils s'inspirent également d'études issues d'autres disciplines connexes notamment la sociologie, la psychologie sociale, les sciences cognitives et bien d'autres. C'est ainsi qu'on les trouve parfois à la base des [47]

- systèmes distribués;
- interface personnes-machines;

- bases de données et bases de connaissances distribuées coopératives;
- systèmes pour la compréhension du langage naturel ;
- protocoles de communication et réseaux de télécommunications;
- programmation orientée agents et génie logiciel;
- robotique cognitive et coopération entre robots;
- applications distribuées comme le web, l'Internet, le contrôle de trafic routier, le contrôle aérien, les réseaux d'énergie, etc.

I.10.1. Ontologie pour SMA

Dans un système multiagent ouvert, les agents sont hétérogènes, c'est-à-dire qu'ils n'ont pas forcément été conçus au même moment, ni par les mêmes organisations. Ainsi, pour qu'un agent puisse découvrir et exploiter les services disponibles, il doit être capable de communiquer avec d'autres agents, est à dire interopérer.

C'est dans ce contexte qu'étudie l'apport de la notion d'ontologie pour améliorer l'interopérabilité dans un système ouvert.

Parmi les solutions pour gérer cette interopérabilité l'utilisation des ontologies qui sont des modélisations formelles des connaissances d'un domaine, compréhensibles et exploitables par les agents logiciels fournis un contexte de domaine construit par langage standardisé tel que OWL partageable entre les différents agents hétérogènes.

Donc l'ontologie de domaine est la solution clé pour assurer l'interopérabilité sémantique entre les systèmes multi-agents ouverts.

I.10.2. SMA pour les ontologies

Les S.M.A sont des systèmes idéaux pour représenter des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples résolveurs. Ces systèmes possèdent les avantages traditionnels de la résolution distribuée et concurrente de problèmes comme la modularité, la vitesse (avec le parallélisme), et la fiabilité (due à la redondance).

L'apport des SMA pour les ontologies parvient sur plusieurs axes comme la construction d'ontologie à partir du texte grâce à SMA on peut automatiser cette tache et aussi dans l'alignement d'ontologies dans le web sémantique le problème traité est distribué et les taches réalisées sont indépendantes pour cela nous exploitons les Systèmes multi agent pour arriver à des résultats meilleurs avec efficacité grâce à l'autonomie et flexibilité des agents logiciels.

Et aussi pour gérer les alignements dans le web sémantique l'exploitation des SMA devient indispensable.

I.11. Conclusion

Dans ce chapitre, nous avons présenté l'état de l'art sur les ontologies débutant par sa notion inspirée sur la philosophie et Gruber, et dans le domaine d'informatique (IC), puis les différents types d'ontologies organisées selon le degré de formalisme, selon les objets modélisés, selon la granularité et les ontologies au sein de représentations du connaissances, ensuite les composants d'une ontologie après les formalismes de représentation.

Puis nous avons vu les outils permettant de construire, d'inférer et d'interroger une ontologie,

le cycle de vie d'une ontologie et comment elle évalue et elle est évolue, de même nous avons vu l'apport des SMA pour les ontologies et enfin les recherches futures sur les ontologies tel que l'alignement et fusion des ontologies .

Dans un contexte ouvert, chaque agent est construit de manière indépendante ainsi que l'ontologie qu'il utilise. Nous pouvons donc déduire que dans un système ouvert, chaque ontologie représente une conception différente même si ces dernières sont construites pour un même domaine. Pourtant, il est possible pour un humain de trouver les équivalences qui peuvent exister entre deux ontologies, en utilisant la sémantique de chaque concept. C'est sur cette constatation que nous nous basons pour faire résoudre l'hétérogénéité sémantique de ces systèmes possédant plusieurs ontologies de manière automatique grâce à la sémantique de chaque concept.

Dans le chapitre suivant, nous présentons un état de l'art sur l'alignement des ontologies pour résoudre le problème de l'hétérogénéité sémantique entre des ontologies de domaine.

II.1. Introduction

Dans le domaine de la représentation des connaissances, les ontologies ne sont considérées que relatives aux différents domaines de connaissances.

Elles répondent aux problèmes de représentation et de manipulation des connaissances. Les ontologies sont très utilisées dans la représentation des connaissances sur le Web [62]. De nos jours, l'ontologie matérialise une connaissance experte d'un domaine. Partant du fait que plusieurs connaissances peuvent prendre des représentations différentes, on trouve de nos jours plusieurs ontologies de domaine pour un même champ d'application. Les techniques d'alignement représentent un cadre général, dans lequel plusieurs ontologies peuvent être exploitées [75].

Dans ce chapitre, nous allons présenter comment est né le besoin à l'alignement des ontologies, quelques domaines d'application de l'alignement des ontologies, la définition de l'alignement avec quelques précisions terminologiques, Dimensions de l'alignement (input, processus de l'alignement, output) puis nous allons présenter les techniques et les méthodes utilisées dans la littérature qui attaque le problème de recherche de la similarité, de la dissimilarité ou de la correspondance entre deux entités en général, qu'elles apparaissent dans des schémas, ou dans des ontologies.

II .2. Comment est né le besoin à l'alignement des ontologies ?

Une ontologie est un ensemble d'entités (concepts, relations, restrictions, instances) modélisées pour représenter un domaine particulier de la manière la plus consensuelle possible. Cependant, comme n'importe quelle autre représentation explicite des connaissances, une ontologie dépend toujours d'assomptions implicites comme les objectifs des concepteurs, leurs capitaux connaissances, rendant l'objectivité de la représentation un but difficile à concrétiser.

Ces connaissances implicites sont à l'origine de différentes formes d'hétérogénéité entre les ontologies, même entre les ontologies décrivant le même domaine.

L'alignement des ontologies est la solution pour surmonter ces problèmes d'hétérogénéité. Il existe plusieurs classifications d'hétérogénéité entre les ontologies dans ce qui suit, nous présentons une classification [48] qui résume les principales formes d'hétérogénéité en quatre niveaux, à savoir :

II.2.1. Le niveau syntaxique : il s'agit de toutes les formes d'hétérogénéité relatives au choix du format de représentation. En effet il existe différentes façons de représenter les ontologies (OWL, KIF …) et chaque langage est basé sur une syntaxe différente.

II.2.2. Le niveau terminologique : l'hétérogénéité, à ce niveau, intervient dans le fait d'affecter des noms aux entités (classes, propriétés, relations …) qui constituent une ontologie. Nommer une entité revient à lui associer un objet linguistique à partir d'un langage public.
Des exemples de ce type d'hétérogénéité :

1- Différents noms utilisés pour designer une même entité (synonymie)

2-Le même nom utilisé pour designer deux entités distinctes (polysémie)

3- Des mots provenant de différentes langues (français, anglais, italien…) utilisées pour designer une même entité.

4- Des variations syntaxiques du même mot (différentes prononciations, abréviations, utilisation des préfixes et des suffixes …)

II .2.3. Le niveau conceptuel : les divergences à ce niveau peuvent être résumées en trois aspects :_ la couverture : la différence entre deux ontologies peut être au niveau de la portée de la couverture du domaine décrit. Elles peuvent couvrir des parties différentes (du monde réel ou d'un domaine) ou alors des parties qui se chevauchent, par exemple : une ontologie sur le sport couvre le sport de la course automobile qu'une autre ignorerait complètement.

- **La granularité** : deux ontologies peuvent d'écrire les mêmes entités avec des niveaux de détail différents, par exemple : une ontologie concernée par les comptabilités va considérer le concept générique du document alors qu'une ontologie décrivant le domaine des bibliothèques va distinguer entre les différents types de documents : romans, nouvelles, biographies, manuscrits …

- **La perspective** : deux ontologies peuvent décrire un domaine de deux points de vue différents. Par exemple : le concept de la chaleur chez un Norvégien sera forcément différent du même concept chez un Sénégalais.

II.2.4. Le niveau sémiotique ou pragmatique: ce type d'hétérogénéité intervient lorsqu'il y a différence d'interprétation de la même ontologie par différentes personnes ou différentes communautés. Ces différences d'interprétations sont souvent liées au choix du formalisme de représentation des connaissances, par exemple, des clauses de la logique du premier ordre et

une représentation hiérarchique de classes, sont-ils équivalents, est-ce qu'ils véhiculent la même connaissance ?

En résumé, comprendre les différentes formes d'hétérogénéité des ontologies est primordial pour la réussite de l'alignement de ces dernières, car il est très risqué de réaliser un alignement entre des entités, en se basant seulement sur les liens sémantiques, les autres dimensions doivent être prises en compte.

II.3. Quelques domaines d'application de l'alignement des Ontologies

Plusieurs domaines exploitent l'alignement des ontologies en cours de conception ou en cours d'exécution des systèmes les plus connues sont : **[1]**

II.3.1. L'ingénierie ontologique

C'est un contexte ou les concepteurs d'ontologies sont confrontés à l'hétérogénéité de ces dernières, plus précisément, par rapport aux applications suivantes :

II.3.1.1. La construction d'ontologies : ces dernières années, le maître mot dans la démarche de construction des ontologies est la réutilisation d'ontologies déjà existantes, car la construction d'ontologies à partir de zéro (from scratch) est un processus long, coûteux et très laborieux, parallèlement, elle accentue le phénomène de l'hétérogénéité des ontologies, multipliant le nombre d'ontologies décrivant le même domaine (surtout lorsqu'on sait que l'objectif ultime du web sémantique est d'arriver à instaurer une ontologie de référence pour chaque domaine). Dans ce contexte, l'alignement des ontologies est la solution pour réaliser l'intégration et le rapprochement de ces différentes structures.

II.3.1.2. L'évolution des ontologies : Beaucoup d'ontologies sont en continuelle évolution comme la Geneontology, et de ce fait, plusieurs versions de la même ontologie sont disponibles, mettant les développeurs et les ingénieurs de la connaissance dans la confusion, ne sachant pas ce qui a changé, l'alignement va permettre d'identifier les différences entre deux versions : les entités qui ont été ajoutées, supprimés ou renommés.

II.3.2. L'intégration d'information

C'est une application classique de l'alignement d'ontologies, elle comprend l'intégration des schémas, les entrepôts de données, l'intégration des données et l'intégration des catalogues.

La figure II.1 présente un scénario général de l'intégration d'information, étant donnée, un ensemble de sources locales d'information (ontologies locales LO1, …LOn) qui stocke leurs informations dans des formats différents (SQL, XML, RDF), fournissent aux utilisateurs une

interface de requêtes uniforme à travers une ontologie globale ou commune (CO) à toutes les sources d'information. Ce qui permet aux utilisateurs d'adresser des requêtes directement à l'ontologie globale.

Les sources de données sont transformées en ontologies locales qui sont alignées par rapport à une ontologie globale, *les alignements obtenus aident à générer les médiateurs* qui, à leurs tours, transforment les requêtes adressées à l'ontologie globale en requêtes pour les sources d'information locale et traduisent les réponses dans l'autre sens.

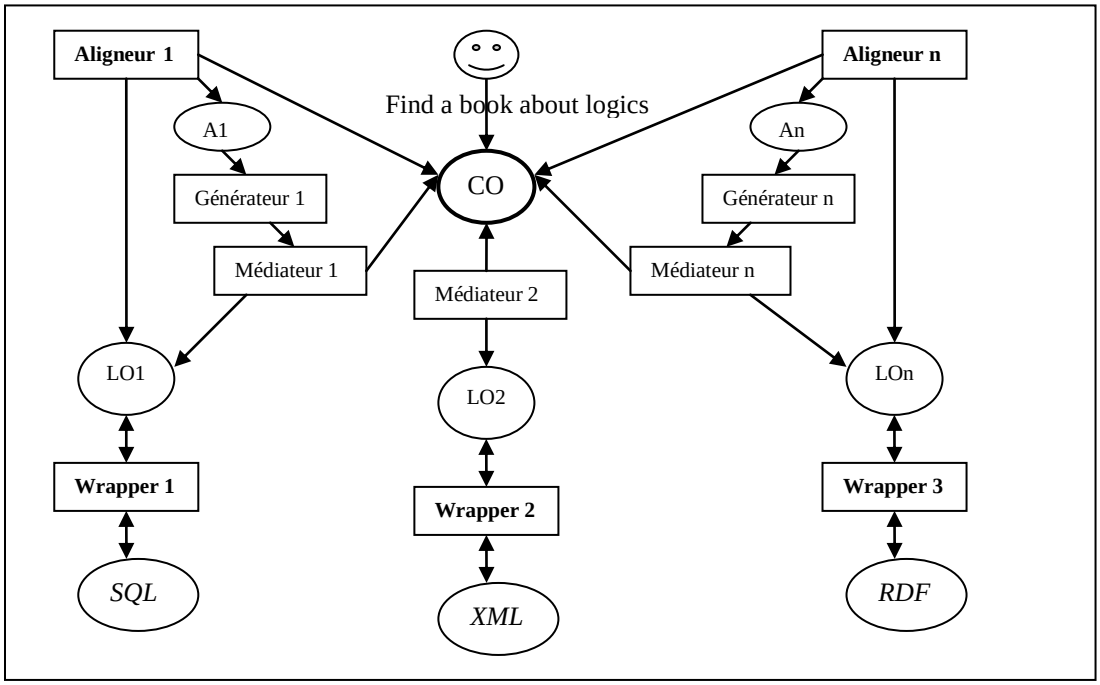


Figure II.1 Scénario centralisé d'intégration d'information **[49]**

II.3.3. Le Partage des informations dans les systèmes pair-à-pair (P2P)

P2P est un modèle de communication distribué dans lequel les pairs ont des capacités fonctionnelles équivalentes dans les échanges de données et de services [50].

Les pairs doivent s'échanger des informations alors qu'ils utilisent des terminologies différentes (les problèmes de traduction des requêtes et de leurs réponses). Une des étapes pour résoudre ce problème serait d'identifier les relations qui existent entre leurs ontologies respectives, autrement dit, réaliser un alignement d'ontologies.

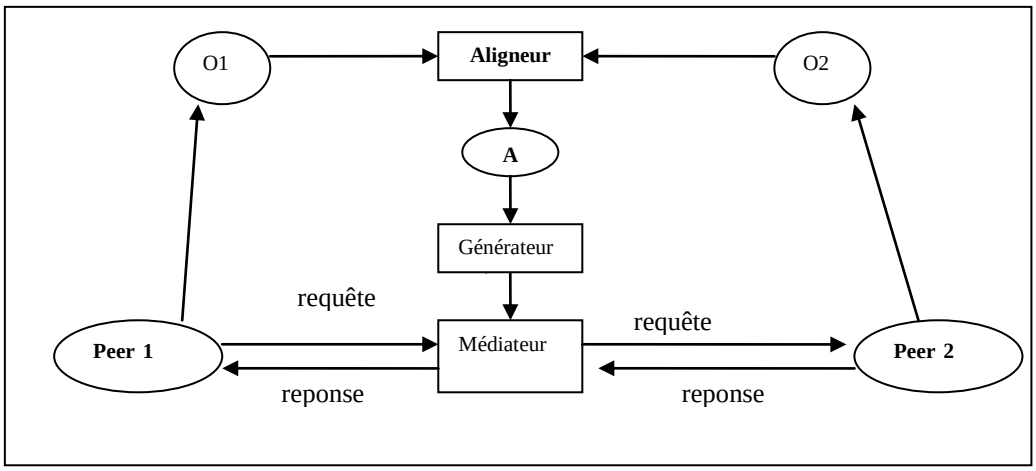


Figure II.2 Réponse à une requête dans un système P2P [49]

II.3.4. La Composition des services web

Les services web sont des processus qui exposent leurs interfaces aux utilisateurs du web qui les invoquent. Les services web sémantiques fournissent un moyen plus riche et plus précis de décrire les services à travers les langages de représentation des connaissances et des ontologies [51].

Par exemple, un service web fournit la description de son output à l'aide d'une ontologie et un autre service web utilise une seconde ontologie pour d'écrire son input, aligner ces deux ontologies permettrait de vérifier si ce qui a été délivré par le premier service correspond à ce qui était attendu par le second service et cela grâce à un médiateur entre ces deux services, généré à partir de l'alignement des deux ontologies précédemment citées.

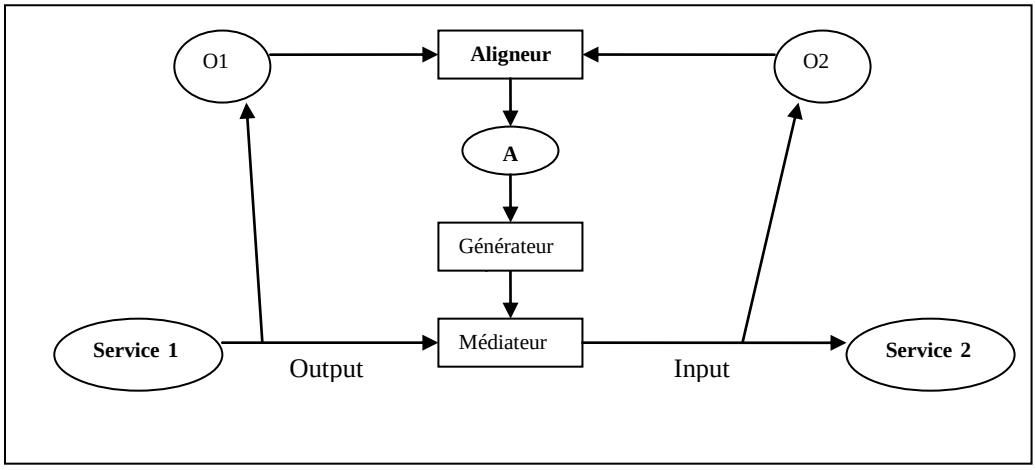


Figure II.3 composition d'un service web **[49]**

II.3.5. La communication entre agents

Lorsque deux agents autonomes et conçus indépendamment se rencontrent, ils ont la possibilité de s'échanger des messages, mais peu de chances pour se comprendre s'ils ne partagent pas le même langage et la même ontologie. L'alignement de leurs ontologies respectives intervient à ce niveau pour traduire les messages ou bien intégrer des passerelles entre leurs axiomes dans le modèle propre à chaque agent (pour pouvoir interpréter les messages).

II.4. Alignement d'Ontologies

II.4.1. Définition: l'alignement d'ontologies consiste à déterminer, par un processus global, les correspondances (similarité) entre deux ontologies, en utilisant ou en mettant en œuvre des solutions aux différents problèmes d'hétérogénéité, et des méthodes qui permet de calculer la similarité [60].

Les liens sémantiques comprennent les relations : d'équivalence ($=$), de généralisation/spécialisation ($\supseteq$, $\subseteq$), de chevauchement (**II**) ou encore d'incompatibilité ($\perp$)

Voici quelques précisions terminologiques : [52]

- **Mapping**

C'est l'expression formelle d'une relation sémantique entre deux entités appartenant à deux ontologies différentes. Lorsque la relation est orientée, elle devient une restriction de la signification mathématique du mapping, qui est la fonction.

- **Coordination d'ontologies**

C'est un terme large, appliqué à chaque fois qu'il faut utiliser la connaissance à partir de deux ontologies ou plus, en même temps, et d'une manière significative, autrement dit, leurs relations mutuelles s'avèrent pertinentes (par exemple pour atteindre le même objectif).

- **Transformation d'ontologies**

C'est un terme général utilisé pour faire référence à n'importe quel processus qui démarre à partir d'une ontologie et lui applique une fonction de transformation pour obtenir une nouvelle ontologie (comme le passage vers une nouvelle version, appelée également versioning).

- **Fusion d'ontologies**

Création d'une nouvelle ontologie à partir de deux ontologies différentes (contrairement à la fusion, qui altère les deux ontologies en entrées pour en obtenir une autre, l'intégration garde intacte une ontologie généralement la plus grande et modifie celle qui va être incluse dans l'autre, c'est un cas de figure possible lorsque les ontologies comprennent des parties qui se chevauchent).

- **Réconciliation d'ontologies**

Processus qui harmonise le contenu de deux ontologies ou plus et qui exige la transformation d'au moins une des ontologies [63].

II.5. Dimensions de l'alignement

L'alignement regroupe trois dimensions : l'input, le processus d'alignement et l'output .

II.5.1. L'input : est constitué essentiellement des structures destinées à être alignées et qui peuvent être, comme énoncé précédemment, des schémas XML, des schémas relationnels, des ontologies.

Remarque : l'input peut être enrichi par un alignement en entrée (qui aurait besoin d'être complété par une nouvelle itération d'alignement).

II.5.2. Le processus d'alignement : peut être considéré comme une fonction **f**, qui à partir d'une paire d'ontologies **O** et **O'**, un alignement en entrées **A** (optionnel), un ensemble de paramètres **p** (ex : paramètres de pondération, seuils …) et un ensemble de ressources externes **r** (ex : thésaurus, lexique…), détermine un alignement **A'** entre ces deux ontologies.

A' = f (O, O', A, p, r)

Ceci peut à être représenté schématiquement de la manière suivante :

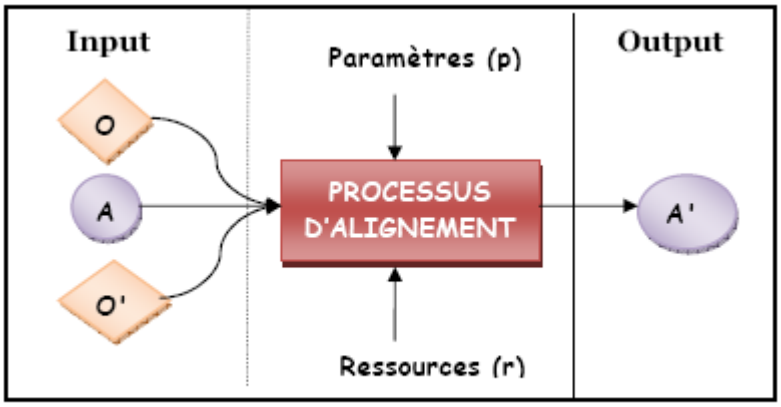


Figure II.4 les trois dimensions de l'alignement

Le processus d'alignement est exécuté selon *une stratégie* ou une combinaison de techniques d'alignement de bases d'écrites dans la suite de ce chapitre.

II.5.3. L'output : est un ensemble d'alignements reliant les entités qui composent les deux ontologies. Un alignement est d'écrit comme un ensemble de cinq éléments :

< id, e, e', r, n >

id : identifiant unique d'un mapping

e : une entité, à aligner, appartenant à **O** (classe, propriété, contrainte, instance)

e' : une entité, à aligner, appartenant à **O'**

r : la relation qui relie **e** à **e'** (**=** , **⊇**, **⊆** , **Π**, **⊥**)

n : la mesure de confiance de la relation **r**, généralement une valeur réelle comprise dans l'intervalle [0,1]. Plus le **n** est proche du 1, plus la relation est considérée comme étant forte.

L'output est caractérisé par :

a) La multiplicité (contraintes sur les relations entre les entités des deux ontologies).

Si on considère, d'une part, les valeurs suivantes:

1 : une et une seule relation
? : de 0 à 1 relation
+ : de 1 à plusieurs relations
***** : de 0 à plusieurs relations

} à partir d'une entité d'ontologie

D'autre part, les deux orientations possibles d'un alignement entre deux ontologies (**O→O'** et **O'→O**) , la multiplicité peut prendre les valeurs suivantes :

1 :1, 1 :?, ?:1, 1 :+, +:1, 1 :*, :1, ?:?, ?:+, +:?, ?:, * :?, + :*, * :+, + :+, * :*.

Voici quelques exemples sur les configurations de multiplicité entre deux ontologies, constitué chacune de trois entités : [64]

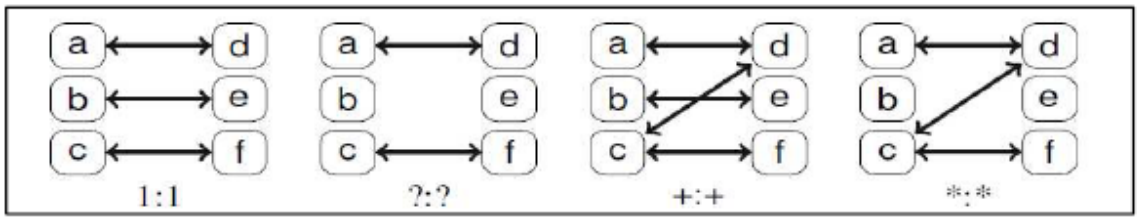


Figure II.5 Exemple de configurations de multiplicité entre 2 ontologies

II. 6 Les méthodes de base pour mesurer la similarité

Pour connaître les méthodes d'alignement, premièrement nous donnons la définition de la similarité, dissimilarité et la distance entre deux entités puis les méthodes de base pour mesurer la similarité entre deux entités.

Avant de présenter les méthodes d'alignement, nous donnons la définition de la similarité.

II.6.1 La similarité

La notion de similarité dans notre contexte n'est pas celle que l'on peut trouver en psychologie ou en mathématiques. En psychologie sociale, la similarité se rapporte à comment les attitudes, les valeurs, les intérêts et la personnalité correspondent entre les personnes.

En mathématiques, plusieurs relations d'équivalence (qui sont des relations binaires réflexives, symétriques et transitives) sont appelées la similarité.

Définition 1 (Similarité) [60]**.** La similarité S : O × O → R est une fonction d'une paire d'entités à un nombre réel exprimant la similarité entre ces deux entités telle que:

- $\forall a, b \in O, S(a, b) \geq 0$ *(positivité)*
- $\forall a, b, c \in O, S(a, a) \geq S(b, c)$ et $S(a, a) = S(a, b) \Leftrightarrow a = b$ *(auto-similarité ou maximalité)*
- $\forall a, b \in O, S(a, b) = S(b, a)$ *(symétrie)*
- $\forall a, b, c \in O, S(a, b) = S(b, c) \Rightarrow S(a, b) = S(a, c)$ *(transitivité)*
- $\forall a, b \in O, S(a, b) \leq \infty$ *(finitude)*

La dissimilarité est parfois utilisée au lieu de la similarité. Elle est définie de manière analogue à la similarité, sauf qu'elle n'est pas transitive :

Définition 2 (Dissimilarité) [60]**.** La dissimilarité DS : O × O → R est une fonction d'une paire d'entités à un nombre réel exprimant la dissimilarité entre ces deux entités telle que:

- $\forall a, b \in O, DS(a, b) \geq 0$ *(positivité)*
- $\forall a, b, c \in O, DS(a, a) \leq DS(b, c)$ et $DS(a, a) = 0$ *(minimalité)*
- $\forall a, b \in O, DS(a, b) = DS(b, a)$ *(symétrie)*
- $\forall a, b \in O, DS(a, b) \leq \infty$ *(finitude)*

La distance est une mesure utilisée aussi souvent que les mesures de similarité.

Elle mesure la dissimilarité de deux entités, elle est inverse de la similarité : si la valeur de la fonction de similarité de deux entités est élevée, la distance entre elles est petite et vice-versa. Elle est donc définie dans [65] comme suit :

Définition 3 (Distance) : La distance D : O × O → R est une fonction de la dissimilarité satisfaisant la définitivité et l'inégalité triangulaire[60] :

- $\forall a, b \in O, D(a, b) = 0 \Leftrightarrow a = b$ *(définitivité)*
- $\forall a, b, c \in O, D(a, b) + D(b, c) \geq D(a, c)$ *(inégalité triangulaire)*

Les valeurs de similarité sont souvent normalisées pour pouvoir être combinées dans des formules plus complexes. Si la valeur de similarité et la valeur de dissimilarité entre deux entités sont normalisées, notées S et DS , alors on a 1 = DS +S .

Définition 4 (Normalisation) : Une mesure est une mesure normalisée si les valeurs calculées par cette mesure ne peuvent varier que dans un intervalle de 0 à 1. Ces valeurs calculées sont appelées valeurs normalisées. Les fonctions du calcul sont appelées fonctions normalisées et notées f.

Les mesures de la similarité, de la dissimilarité, de la distance peuvent être classées selon la nature des entités que l'on veut comparer : des termes, des chaînes de caractères, des structures, des instances (des individus des classes), des modèles théoriques [60].

II.6.2. Les méthodes terminologiques

Ces méthodes se basent sur la comparaison des termes ou des chaînes de caractères ou bien les textes. Elles sont employées pour calculer la valeur de la similarité des entités textuelles, telles que des noms, des étiquettes, des commentaires, des descriptions… Ces méthodes peuvent encore être divisées en deux sous-catégories l'une contient des méthodes qui comparent des

termes en se basant sur les caractères contenus dans ces termes et l'autre utilise certaines connaissances linguistiques.

II.6.2.1 Les méthodes basées sur des chaînes de caractères

Ces méthodes analysent la structure des chaînes de caractères, l'ordre des caractères dans la chaîne, le nombre d'apparitions d'une lettre dans une chaîne pour concevoir des mesures de la similarité. Par contre, elles n'exploitent pas la signification des termes. Par exemple, les mesures dans cette catégorie retournent une grande valeur de similarité (jusqu'à 1) si elles comparent les termes « Voiture » et « voitures », mais une petite valeur, voire la valeur 0, si elles comparent les termes « voiture » et « bagnole ».

Les résultats de la comparaison des chaînes de caractères seront améliorés si ces chaînes sont « nettoyées » ou traitées avant de les fournir aux formules calculant la similarité. Cette phase est appelée la phase de normalisation ou de normalisation textuelle, qui diffère de la normalisation des valeurs de similarité dans un intervalle de [0,1] discuté ci-dessus **(Définition 4)**.

Les différents types de normalisation textuelle sont ceux empruntés au domaine de traitement automatique de la langue naturelle (TALN) :

- Normalisation des caractères : ce type de normalisation convertit toutes les majuscules dans une chaîne de caractères en leurs formes minuscules ou vice-versa. Par exemple, la chaîne de caractères « VoitureS » sera convertie à « voitures » et ensuite, elle est considérée comme égale exactement à l'autre chaîne de caractères « voitures ».
- Normalisation des espaces : ce type de normalisation remplace toutes les séquences consécutives des espaces, des tabulations, des retours de chariot (les caractères CR) trouvées dans une chaîne de caractères par un seul caractère d'espace. Par exemple, l'expression « ma voiture » est normalisée à « ma voiture ».
- Suppression des signes diacritiques ou des accents (aigus, graves…) : ce type de traitement remplace des caractères avec des signes diacritiques par caractères correspondants sans signes diacritiques. Par exemple, le mot « Hanoï » est remplacé par le mot « Hanoi » sans changer la signification du mot, la capitale du Vietnam. Cependant, certaines suppressions changeront la signification du terme : « là » (adverbe de lieu) et « la » (article).
- Suppression des chiffres.
- Élimination des ponctuations.

- Élimination des mots vides (les mots contenant peu d'informations tels que « est », « un » ,« les »…)
- Suppression des affixes (préfixes, suffixes).
- Extension des abréviations (WS=Web Sémantique).
- Tokenisation : exp. PersonEntité devient {Person,Entité}
- Lemmatisation (passer au singulier, à l'infinitif pour les verbes, au masculin pour les adjectifs…)

Il existe plusieurs mesures calculant la valeur de similarité ou la distance entre deux chaînes de caractères dans la littérature telles que la similarité de Jaccard, la distance de Hamming, la distance de Levenshtein… etc.

Nous présentons ici quelques mesures les plus utilisées dans les approches d'alignement d'ontologies dans le cadre du Web sémantique.

Si nous considérons une chaîne de caractère s comme un ensemble de caractères S, la similarité de **Jaccard** entre deux chaînes est définie ainsi :

Définition 5 (Similarité de Jaccard) [60]. Soit s et t deux chaînes de caractères.

Soit S et T les ensembles des caractères de s et t respectivement. La similarité de Jaccard est une fonction de la similarité SJaccard : S x $S \rightarrow [0, 1]$ telle que :

$$\overline{S_{\text{Jaccard}}}(s,t) = \frac{|S \cap T|}{|S \cup T|}$$

Une classe importante des fonctions mesurant la distance entre deux chaînes de caractères s et t, est constituée des fonctions calculant la distance d'édition (edit distance), dans lesquelles la distance est le coût de la meilleure séquence des opérations d'édition qui convertit s en t. Des opérations d'édition typiques sont celles de l'insertion, de la suppression, et de la substitution de caractère, et à chaque opération est affectée à un coût.

La métrique Jaro produit la similarité entre deux chaînes de caractères en se basant sur le nombre et l'ordre des caractères communs entre elles.

Définition 6 (Distance de Jaro) : Soit s et t deux chaînes de caractères. Soit Nc le nombre des caractères communs apparaissant dans les deux chaînes dans une distance de moitié de la longueur de la chaîne la plus courte[60].

Soit Nt le nombre des caractères transposés, qui sont des caractères communs apparaissant dans des positions différentes. La distance de Jaro est une fonction de la dissimilarité DSJaro : S x S → [0, 1] telle que :

$$\overline{DS_{\text{Jaro}}}(s,t) = 1 - \frac{1}{3}\left(\frac{N_c}{|s|} + \frac{N_c}{|t|} + \frac{N_c - N_t/2}{N_c}\right)$$

Il existe aussi des distances qui sont des variantes de la distance de Jaro, telles que la distance Jaro-Winkler :

Définition 7 (Distance de Jaro-Winkler) [60]**.** Soit s et t deux chaînes de caractères. Soit P la longueur du préfixe commun le plus long de s et t. Soit n un nombre positif. La distance de Jaro-Winkler est une fonction de la dissimilarité DSJaroWinkler : S x S → [0, 1] telle que :

$$\overline{DS_{\text{JaroWinkler}}}(s,t) = \overline{DS_{Jaro}}(s,t) - \frac{max(P,n)}{10}\overline{DS_{Jaro}}(s,t)$$

II.6.2.2. Les distances basées sur des tokens

Les mesures présentées ci-dessus s'adaptent bien lorsque l'on veut comparer deux termes ou deux courtes chaînes de caractères.

Il existe aussi des cas où l'on a besoin de comparer des textes longs ou bien des documents textuels. Dans ces cas, ces entités sont découpées en plusieurs morceaux, appelés tokens. Elles deviennent des ensembles des tokens, et la similarité entre elles est produite grâce aux mesures de similarité basées sur des tokens.

Il existe plusieurs mesures de cette catégorie dans la littérature telles que la similarité de Dagan [53], la distance de Jensen-Shannon, la distance de Fellegi-Sunter [54]… Nous présentons ici quelques mesures les plus utilisées dans les approches d'alignement d'ontologies dans le cadre du Web sémantique.

La similarité de Jaccard **(Définition 5)** peut être étendue pour comparer des ensembles des tokens, en définissant la similarité comme le rapport entre la cardinalité de l'intersection des ensembles sur la cardinalité de leur union.

Une mesure qui est largement employée dans le domaine de la recherche d'information, semble convenable ici. Il s'agit du TF/IDF (Term frequency/Inverse Document Frequency). Dans sa conception originale, TF/IDF est employé pour mesurer la pertinence d'un terme dans l'ensemble de documents. La fréquence de terme, TF, dans un document donné montre l'importance de ce terme dans le document en question. La fréquence inverse de document, IDF, est une mesure de l'importance générale du terme dans l'ensemble de documents.

Définition 8 (TF/IDF) [60]. Soit D un corpus des documents, |D| dénote le nombre des documents dans le corpus D. Soit t un terme à considérer, n(t) étant le nombre d'occurrences du terme t dans un document, et N étant le nombre des termes dans ce document, et d(t) étant le nombre des documents qui contiennent au moins une fois le terme t. Les mesures de TF et TF/IDF sont définies comme suivante :

$$TF = \frac{n(t)}{N} \text{ et } TF/IDF = TF * \log\left(\frac{|D|}{d(t)}\right)$$

La similarité des entités textuelles peut être construite comme la valeur cosinus de deux vecteurs représentant ces entités, où chaque dimension correspond à un terme et sa valeur correspond à la valeur **TF/IDF** de ce terme.

[55] fait une excellente comparaison de plusieurs mesures de similarité montrant le point fort de chaque technique pour une tâche particulière.

Chaque mesure de distance ou de similarité s'adapte mieux dans certains domaines d'application.

Le Tableau II.1 résume des domaines d'applications pour des mesures présentées dans cette partie [60].

Mesure de similarité	Domaine d'application
N-gam	Bigams (n=2) est efficace avec des erreurs typographiques mineures
Distance d'édition	Peut-être appliquée aux entités ayant une longueur variable. Pour atteindre une exactitude raisonnable. Les coûts des opérations de modification dépendent de chaque domaine
Distance de Hamming	Utilisée principalement pour les entités numériques ayant des tailles fixes, comme les codes postaux ou les numéros de sécurité sociale
Distance de Monge-Elkan	La meilleure performance au niveau des résultats dans plusieurs expériences. Peut-être employée dans plusieurs domaines.
Distance de Jaro/jaro-winkler	Presque même performance au niveau des résultats que Monge-Elkan mais beaucoup plus rapide.
Distance basée sur TF/IDF	La meilleure pour la comparaison des textes longs (basé sur des tokens).

Table II. 1 Critères principaux d'utilisation des mesures de la similarité.

II.6.2.3. Les méthodes linguistiques

La similarité entre deux entités représentées par des termes peut aussi être déduite en analysant ces termes à l'aide des méthodes linguistiques. Ces méthodes exploitent essentiellement des propriétés expressives et productives de la langue naturelle [56]. Les informations exploitées peuvent être celles intrinsèques (des propriétés linguistiques internes des termes telles que des propriétés morphologiques ou syntaxiques) ou celles extrinsèques (employant des ressources externes telles que des vocabulaires ou des dictionnaires).

II.6.2.3.1 Les méthodes intrinsèques

Une même entité ou un même concept peut être référencé par plusieurs termes (synonymie) ou par plusieurs variantes d'un même terme.

Les méthodes intrinsèques fonctionnent avec le principe de chercher la forme canonique ou représentative d'un mot ou d'un terme (lemme) à partir de ses variantes linguistiques (lexème).

La similarité entre deux termes est donc décidée en comparant leurs lemmes. Par exemple, le résultat de la mesure de similarité exacte de deux mots « ran » et « running » sera égal à 0 (c.-à-d. ils sont différents), alors que le résultat de la même mesure pour les lemmes de ces mots sera égal à 1, ce qui indique que « ran » et « running » sont similaires.

La recherche du lemme d'un mot peut être effectuée dans un dictionnaire.

Une autre approche qui est automatique et plus légère et plus efficace est d'utiliser des stemmers. Un stemmer est un programme ou un algorithme qui détermine la forme radicale à partir d'une forme infléchie ou dérivée d'un mot donné. Les radicaux (stems) trouvés par les stemmers n'ont pas besoin d'être identiques à la racine morphologique du mot. Il suffit que les mots similaires soient associés à un même radical, même si ce radical n'est pas une racine de mot valide. Un stemmer pour le français, par exemple, devrait identifier les chaînes de caractères « maintenaient », « maintenait », « maintenant », ou « maintenir » comme basées sur la racine "mainten".

Une approche plus complexe pour déterminer le radical exact d'un mot est la lemmatisation. Ce processus comprend la détermination de la partie du discours (catégorie lexicologique) d'un mot, et l'application des règles de normalisation différentes pour chaque partie du discours. Cette approche exige la connaissance de la grammaire d'une langue, des règles différentes… Elle est donc lourde, compliquée et difficile à implémenter.

II.6.2.3.2 Les méthodes extrinsèques

Ces méthodes calculent la valeur de similarité entre deux termes en employant des ressources externes telles que des dictionnaires, des lexiques ou des vocabulaires. La similarité est décidée grâce aux liens sémantiques déjà existants dans ces ressources externes tels que des liens synonymes (pour l'équivalence), des liens hyponymes/ hyperonymes (pour la subsomption). Par exemple, à l'aide des ressources des synonymes, «voiture» et «bagnole» sont dites similaires. Typiquement, WordNet2, un système lexicologique, est employé pour trouver des relations telles que la synonymie entre des termes, ou pour calculer la distance sémantique entre ces termes, en utilisant des liens sémantiques dans WordNet, afin de décider s'il existe une relation entre eux.

Les ressources externes utilisées dans les méthodes extrinsèques peuvent aussi être des vocabulaires ou des dictionnaires multilingues, ou d'autres systèmes tels que EuroWordNet3, Polylex4.

II.6.3. Les méthodes structurelles

Ce sont des méthodes qui déduisent la similarité de deux entités en exploitant des informations structurelles lorsque les entités en question sont reliées aux autres par des liens sémantiques ou syntaxiques, formant ainsi une hiérarchie ou un graphe des entités.

Nous appelons méthodes structurelles internes les méthodes qui n'exploitent que des informations concernant des attributs d'entité, et méthodes structurelles externes les autres qui considèrent des relations entre des entités.

II.6.3.1 Les méthodes structurelles internes

Ces méthodes calculent la similarité entre deux entités en exploitant des informations des structures internes de ces entités.

Dans la plupart des cas, ce sont des informations concernant des attributs de l'entité, telle que des informations du codomaine des attributs, celles de la cardinalité des attributs, celles des caractéristiques des attributs (la transitivité, la symétrie), ou celles des autres types de restriction sur des attributs. Par exemple, en considérant l'entité : le concept « Humain », nous pouvons exploiter des informations concernant des attributs de ce concept tels que l'intervalle des valeurs de donnée pour l'attribut « hasAge », à savoir [0, 150] ; la cardinalité de l'attribut « hasSpouse », à savoir 1 ; ou bien la caractéristique transitive de l'attribut « hasAncestor ».

Dans le domaine des bases de données, plusieurs méthodes ont été proposées pour calculer la similarité entre deux éléments de deux schémas de base de données, en se basant sur les contraintes à propos de ces éléments.

II.6.3.2 Les méthodes structurelles externes

Contrairement aux méthodes structurelles internes, qui exploitent des informations des attributs d'entité, les méthodes structurelles externes exploitent des relations entre des entités elles-mêmes, qui sont souvent des relations de subsomption (is-a ou spécialisation) ou de méréologie (part-whole).

Avec ces relations, les entités sont considérées dans des hiérarchies et la similarité entre elles est déduite de l'analyse de leurs positions dans ces hiérarchies. L'idée de base est que si deux entités sont similaires, leurs voisines pourraient également être d'une façon ou d'une autre similaires. Cette observation peut être exploitée de plusieurs manières différentes en regardant des relations avec d'autres entités dans des hiérarchies. Deux entités peuvent être considérées similaires si :

- Leurs super-entités directes (ou toutes leurs super-entités) sont similaires.
- Leurs sœurs (ou toutes leurs sœurs, qui sont les entités ayant la même superentité directe avec les entités en question) sont déjà similaires.
- Leurs sous-entités directes (ou toutes leurs sous-entités) sont déjà similaires.
- Leurs descendants (entités dans le sous-arbre ayant pour racine l'entité en question) sont déjà similaires.
- Toutes (ou presque toutes) leurs feuilles (les entités de même type, qui n'ont aucune sous-entité, dans le sous-arbre ayant pour racine l'entité en question)sont déjà similaires.
- Toutes (ou presque toutes) les entités dans les chemins de la racine aux entités en question sont déjà similaires.

Des combinaisons des heuristiques ci-dessus sont aussi possibles.

Cependant, cette approche peut rencontrer quelques difficultés dans les cas, où les hiérarchies sont différentes au niveau de granularité. Par exemple, si dans une hiérarchie, l'entité « Personne » a deux sous-entités « Enfant » et « Adulte », et si dans une autre hiérarchie, la même entité « Personne » est divisée en deux autres sous-entités « Femme » et « Homme », la déduction que « Enfant » et « Femme » ou « Enfant » et « Homme » sont similaires, est incorrecte dans tous les cas.

Une approche pour déduire la similarité entre deux entités exploite des relations reliant ces deux entités. L'idée est que si l'on a deux entités similaires A et A', et si elles sont connectées par un même type de relation R avec deux autres entités B et B', alors on peut déduire que B et B' sont d'une façon ou d'une autre similaires.

De même, si on sait que A et A' sont similaires, B et B' sont aussi similaires, alors les relations de A-B et de A'-B' peuvent être similaires. L'idée peut être étendue pour un ensemble d'entités et de relations : si on a un ensemble de relations R1…Rn qui sont similaires avec un autre ensemble de relations R1'…Rn', alors les entités qui sont les domaines (ou les co-domaines) de ces relations sont considérées comme similaires.

Cependant, cette approche pose un autre problème : comment peut-on dire que deux relations sont similaires ?

Cette approche est basée sur la similarité des relations pour impliquer la similarité de leurs entités de domaine ou leurs entités de co-domaine.

Des relations entre les entités peuvent être considérées comme d'autres entités, elles peuvent être aussi organisées dans une hiérarchie de relations, et donc, le calcul de la similarité entre les relations est également un grand problème.

II.6.4 Les méthodes extensionnelles

Ces méthodes déduisent la similarité entre deux entités qui sont notamment des concepts ou des classes en analysant leurs extensions, c.-à-d., leurs ensembles des instances.

Dans le cas où les ensembles des instances partagent une partie commune, on peut avoir des mesures qui emploient des opérations de l'ensemble, telles que celles de Hamming ou de Jaccard.

Définition 14 (Distance de Jaccard, version adaptée pour les ensembles des instances) .

Soit S et T deux ensembles. Soit P(x) la probabilité d'une instance aléatoire être dans l'ensemble X. La distance de Jaccard est une fonction de la dissimilarité DS Jaccard :
2^E x 2^E → [0, 1] telle que [60] :

$$\overline{DS_{\text{Jaccard}}}(S,T)=1-\frac{P(S\cap T)}{P(S\cup T)}$$

La mesure ci-dessus produit la similarité de deux entités qui est en fait la similarité entre les deux ensembles de leurs instances en se basant sur la comparaison exacte des éléments dans

deux ensembles. Dans le cas où les ensembles des instances ne partagent aucune partie commune, ces mesures ne sont plus applicables (le résultat retourné sera toujours égal à 1, c.-à-d. les entités à comparer sont toujours différentes).

Une autre mesure pour calculer la similarité entre deux ensembles emploie une technique d'analyse multidimensionnelle dans le domaine de la statistique [61].

L'hypothèse de cette technique est que si deux entités ont les distances très similaires à toutes autres entités, elles doivent être très similaires. Les entités dans des ensembles sont représentées par des vecteurs, dont la valeur d'une dimension est la similarité de l'entité en question avec une autre entité dans les deux ensembles.

II.6.5. Les méthodes sémantiques

On distingue deux techniques de mesure sémantique :

II.6.5.1. Les techniques basées sur les ontologies externes

Lorsque deux ontologies doivent être alignées, il est préférable que les comparaisons se fassent selon un capital de connaissances commun. Ce type de techniques s'intéresse à l'utilisation d'ontologie formelle intermédiaire pour répondre à ce besoin. Cette ontologie va définir un contexte commun [66] pour les deux ontologies à aligner. L'idée est que cette ontologie, avec une couverture appréciable du domaine d'intérêt des ontologies (ou une ontologie encore plus générale comme une ontologie de haut niveau), va permettre de lever le voile sur les ambiguïtés concernant les différentes significations possibles des termes. Des exemples d'ontologies intermédiaires : FMA11 " the Foundational Model of Anatomy ", CYC ontology et SUMO "the Suggested Upper Merged Ontology".

II.6.5.2. Les techniques déductives

Les méthodes sémantiques se basent sur des modèles de logique (tels que la satisfiabilité propositionnelle (SAT), la SAT modale ou les logiques de descriptions) et sur des méthodes de déduction pour déduire la similarité entre deux entités.

Les techniques des logiques de description (telles que le test de subsomption) peuvent être employées pour vérifier des relations sémantiques entre des entités telles que l'équivalence (la similarité est égale à 1) , la subsomption (la similarité est de 0 à 1) ou l'exclusion (la similarité est égale à 0), et permettent donc de déduire la similarité de deux entités.

II.6.6. Les méthodes de combinaison des similarités

Une entité peut être considérée sous plusieurs différents aspects, soit en s'appuyant sur son nom, soit sur ses attributs, ou soit sur ses relations avec d'autres entités. La similarité entre deux entités peut donc être calculée en se basant sur plusieurs aspects. Sur chaque aspect, les caractéristiques d'une entité sont comparées avec les caractéristiques correspondantes d'une autre par une des mesures de similarité de base présentées dans les méthodes de base pour mesurer la similarité, cela retourne une valeur de la similarité (ou de la dissimilarité/distance).

Il faut donc un moyen pour combiner toutes les valeurs de similarité calculées de chaque aspect pour produire une seule valeur de similarité représentative pour deux entités à comparer. Cette partie analyse quelques approches existantes dans la littérature.

Définition 18 (Somme pondérée) [60] : Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit Wi le poids de la dimension i. Soit DS(xi, yi) la dissimilarité de la paire des objets à la dimension i. La somme pondérée entre x et y est une fonction de la dissimilarité DSsp : O x O _ *R* telle que :

$$DS_{sp}(x,y)=\sum_{i=1}^{n} w_i * DS(x_i,y_i) \qquad \sum_{i=1}^{n} w_i = 1$$

En général, la somme des poids est égale à 1 :

dans ce cas, nous avons la version normalisée de DSsp.

Une autre mesure analogue à la somme pondérée est le produit pondéré.

Cependant, un inconvénient de cette mesure est que le résultat sera égal à 0 si une des dimensions est égale à 0.

Définition 19 (Produit pondéré) [60]**.** Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit wi le poids de la dimension i. Soit DS(xi, yi) la dissimilarité de la paire des objets à la dimension i. Le produit pondéré entre x et y est une fonction de la dissimilarité DSpp : O x O →*R* telle que :

$$DS_{sp}(x,y)=\prod_{i=1}^{n} DS(x_i,y_i)^{w_i}$$

Les approches d'alignement des schémas ou des ontologies présentées dans la section suivante emploient une ou plusieurs mesures de similarité présentées dans la section 1.2.1 pour calculer les valeurs de similarité entre des entités, ensuite retournées des alignements entre deux schémas ou deux ontologies en évaluant ces valeurs de similarité.

Toutes les approches, qui combinent des valeurs de similarité calculées par différentes mesures, emploient la méthode de la somme pondérée (Définition 18).

Cependant, certaines approches (par exemple Anchor-PROMPT) déduisent des alignements en examinant des critères heuristiques sans utiliser des méthodes de combinaison des similarités.

La **Figure II.6** résume différentes mesures de la similarité, catégorisées selon les techniques utilisées [60].

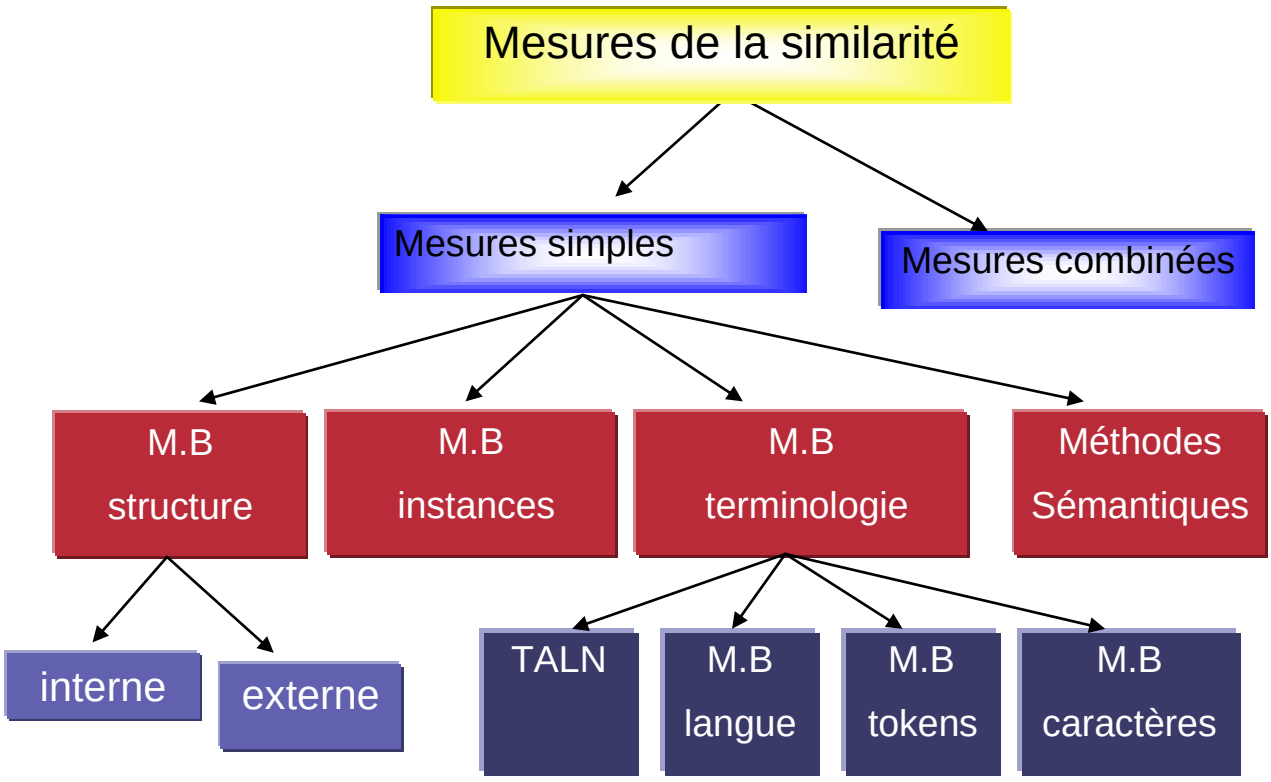


Figure II.6 : Les catégories des mesures de similarité selon différentes techniques.

II.7. Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur l’alignement des ontologies; quelques domaines d’application de l’alignement des ontologies et les dimensions de l’alignement, puis nous avons présenté les techniques et les méthodes utilisées dans la littérature qui attaque le problème de recherche de la similarité, de la dissimilarité ou de la correspondance entre deux entités en général, qui apparaissent dans des ontologies.

L’alignement d’ontologies revêt toute son importance dans des applications nécessitant la prise en compte d’une interopérabilité sémantique. Plusieurs approches d’alignement d’ontologies existent dans la littérature. Elles sont basées sur les mesures de similarités.

Dans le chapitre suivant, nous présentons quelques approches d’alignement des ontologies en utilisant une ou plusieurs mesures de similarité ; pour calculer les valeurs de similarité entre des entités, on s'intéressant sur les ontologies représentées au format OWL-DL et on focalisant sur les approches basant sur les systèmes multi agents.

III.1 Introduction

L'alignement d'ontologies représente un grand intérêt dans le domaine de la gestion des connaissances hétérogènes. L'alignement d'ontologies repose sur le calcul des mesures de similarité.

La littérature du domaine propose plusieurs méthodes d'alignement d'ontologies. Ces méthodes exploitent différents formats de représentations des ontologies et différentes méthodes de mesures de similarité.

Nous nous intéressons aux ontologies décrites dans un même langage de représentation des connaissances (OWL) langage recommandé par W3C .

Dans ce chapitre, nous allons présenter quelques approches existantes dans la littérature qui concernent le problème d'alignement d'ontologies **.**

Ensuite nous présentons quelques propositions existantes pour le partage du sens entre agents basés sur l'alignement d'ontologies.

III.2 Les approche d'alignement d'ontologies

Les techniques d'alignement jouent un rôle capital dans la construction d'un lien sémantique entre les ontologies d'un même domaine. Quelques approches d'alignement considèrent que l'utilisation d'une connaissance sur le domaine est une manière assurant la correspondance sémantique entre la dissimilarité syntaxique des ontologies , l'obtention de la bonne connaissance sur le domaine est primordiale.

D'autres approches n'exploitent pas une connaissance sur le domaine et ne réalisent pas un modèle sémantique formel pour l'alignement des structures produites, dans ce cas la structure obtenue est difficile à exploiter, *e.g.*, pour répondre aux requêtes interrogeant les ontologies . En outre, les approches courantes d'alignement d'ontologies sont basées sur les mesures de similarité entre chaînes de caractères et des structures composites. Les ontologies à aligner peuvent être représentées avec différents langages.

Étant donné que le langage OWL est un standard pour les ontologies, toute méthode d'alignement n'exploitant pas ce format présente un inconvénient .

III.2.1 Les approches dans le domaine du Web sémantique

GLUE [57] est une approche dont le but est de trouver semi automatiquement des correspondances entre des schémas pour l'intégration de données. GLUE utilise la technique

d'apprentissage (telle que Naïve Bayes) pour trouver des correspondances entre deux ontologies.

GLUE comprends, plusieurs modules d'apprentissage (learners), qui sont entraînés par des instances des ontologies. En exploitant différentes caractéristiques des instances telles que les valeurs textuelles des instances, les noms des instances, les formats des valeurs…

Les prévisions de ces modules de mise en correspondance sont combinées par un méta module de mise en correspondance en employant la somme pondérée. Le résultat final des correspondances sera déduit à partir des valeurs de similarité agrégées.

Un inconvénient de cette approche est qu'elle se fonde principalement sur les instances des ontologies, qui ne sont pas toujours abondamment disponibles pour plusieurs ontologies [60].

S-Match [58] est un algorithme et un système pour chercher sémantiquement des correspondances, basé sur l'idée d'employer le moteur de la satisfiabilité propositionnelle (SAT) [67] pour le problème de mise en correspondance des schémas. Il prend comme entrée deux graphes des concepts (schémas), et produit en sortie des rapports entre les concepts tels que l'équivalence, overlapping, différence (mismatch), plus générale ou plus spécifique.

L'idée principale de cette approche est [60]:

- utiliser la logique pour coder le concept d'un nœud dans le graphe et d'appliquer SAT pour trouver des rapports. Le concept à un nœud, qui est alors transformé en formule propositionnelle, est la conjonction de tous les concepts des étiquettes des nœuds sur le chemin de la racine du graphe jusqu'au nœud en question.
- Le concept d'étiquette d'un nœud est construit en deux étapes :

(i) la normalisation de l'étiquette: telle que la tokenization, la lemmatisation,

(ii) l'extraction du sens de l'étiquette normalisée (des lemmes) à partir de WordNet .

Ensuite, les relations sémantiques (l'équivalence, plus générale, plus spécifique) entre deux étiquettes de deux schémas sont

(i) calculées grâce aux « matchers », les modules qui calculent la similarité entre deux étiquettes en employant des mesures de similarité de base (1.2.1.2) telle que la similarité des préfixes, des suffixes, la distance d'édition, la similarité de n-gram ;

(ii) déduites en employant des matchers qui exploitent la sémantique dans WordNet, la similarité entre des hiérarchies, la similarité entre des commentaires. Ces relations sémantiques sont aussi encodées en logique.

Enfin, le rapport entre deux concepts qui doit être prouvé est également converti en formule propositionnelle. Le moteur SAT calcule sur l'ensemble de formules propositionnelles pour vérifier si le rapport supposé est vrai ou faux. Cela permet donc de déduire des correspondances entre deux ontologies.

OLA [68] est un algorithme pour aligner des ontologies représentées en OWL. Il essaie de calculer la similarité de deux entités dans deux ontologies en se basant leurs caractéristiques (leurs types : classe, relation ou instance, leurs rapports avec d'autres entités : sous-classe, domaine, co-domaine…) et de combiner les valeurs de similarités calculées pour chaque paire d'entités de manière homogène.
La combinaison est la somme pondérée des valeurs de similarité de chaque caractéristique.
Les poids sont associés suivant le type d'entité à comparer et ses caractéristiques.
Ils sont mis dans une matrice des poids et sont prédéfinis avant l'exécution de l'algorithme.
Les mesures de similarité de base employées dans l'algorithme sont

- l'égalité des chaînes des caractères pour des URIs des entités.
- des mesures de similarité des suffixes ou des chaînes des caractères pour des étiquettes des entités, la similarité (l'égalité) des types des données.
- Pour la similarité entre deux ensembles, le cas très souvent rencontré dans OWL (par exemple, en comparant deux entités, l'algorithme exploite la similarité de deux ensembles d'entités qui sont sous-entités des entités en question), il utilise la mesure de similarité basée sur des correspondances. À partir des valeurs de similarité calculées par des mesures de base,
- L'algorithme applique un calcul du point fixe, avec des itérations pour améliorer la similarité de deux entités. Quand il n'y a plus d'améliorations, des alignements entre deux ontologies sont générés.

L'algorithme **ASCO2** propose un modèle de calcul de similarité sur deux étapes [60] : la similarité partielle et la similarité finale. La similarité partielle entre deux entités des deux ontologies est déduite entre les composantes correspondantes aux entités en question. Ces composantes sont des pièces de connaissance contenues dans les définitions de l'entité en employant des primitives du langage OWL. Les valeurs de similarité partielle sont ensuite agrégées dans un schéma de pondération variable pour obtenir une meilleure valeur de similarité finale de ces deux entités .

Les relations entre les approches d'alignement des schémas et les approches d'alignement des ontologies avec les mesures de similarité utilisées (qui sont résumées et inspiré au Bach Thành Lê **[60]** sont montrées dans la **Figure III.1**.

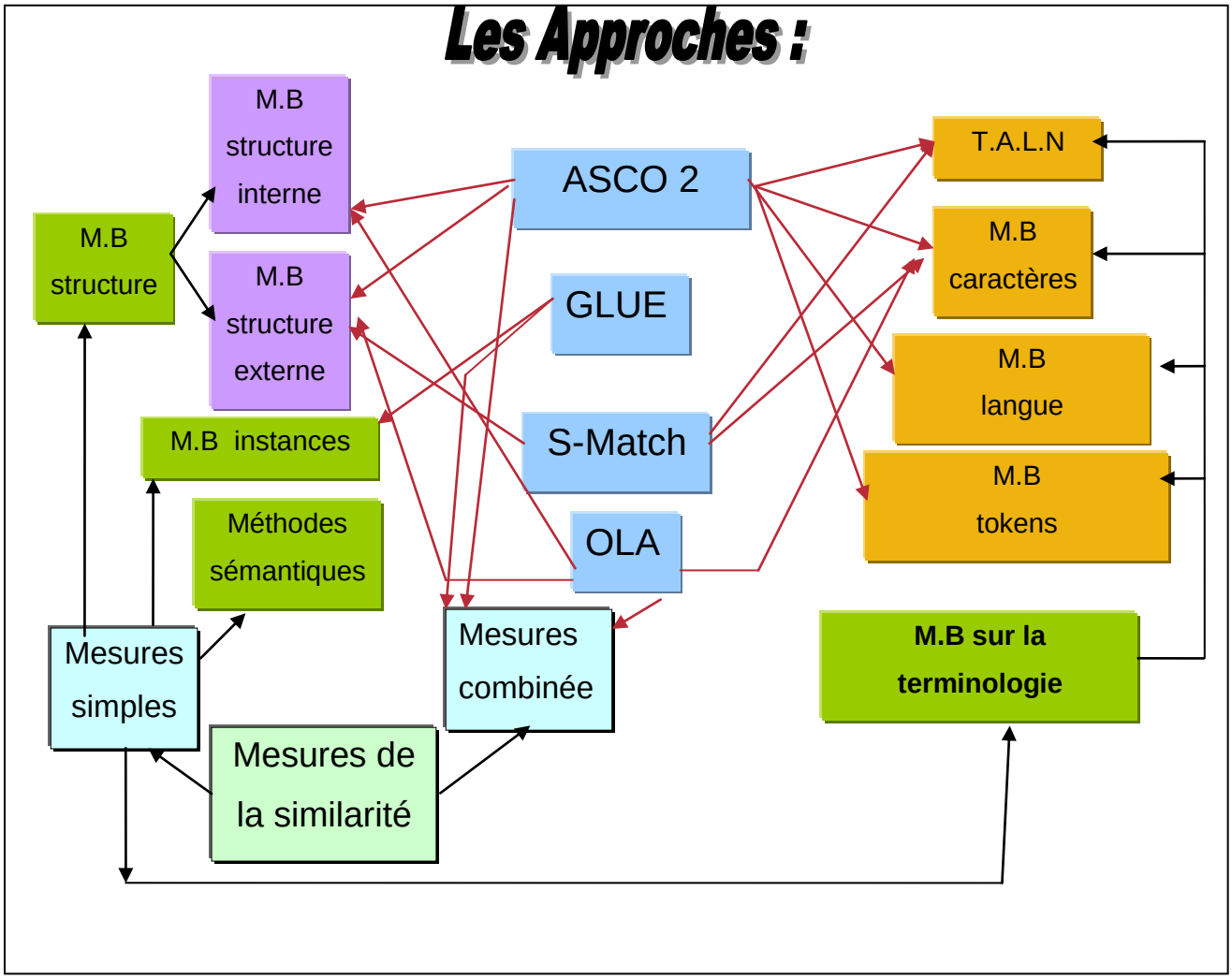


Figure III.1 : Des approches d'alignement des ontologies et leurs relations avec des mesures de la similarité

III.3. Les approches d'alignement basées sur les Systèmes Multi Agents : (Quelques propositions pour le partage du sens entre agents)

Nous présentons différentes approches qui abordent le problème de la compréhension partagée des connaissances au sein des systèmes à base de connaissance, et dans lequel les agents manipulent des ontologies pour représenter leurs connaissances.

III.3.1 L'approche « NASA / Knowledge Evolution, Inc » : ONP

. Le protocole présenté dans ce projet s'attaque à un problème très similaire à celui que nous traitons, à savoir permettre l'exploitation de bases de connaissances distribuées par des agents logiciels. Il traite également de mise en correspondance d'ontologies.**[46]** ONP intervient au niveau de la réception de chaque message échangé entre un agent A et un agent B : l'agent B récepteur va mettre en jeu différents mécanismes pour comprendre le message qu'il reçoit, et ajouter les informations nouvellement acquises à son ontologie. Ce processus contient 4 étapes centrales :

Interprétation : cours de cette phase, un agent récepteur B détermine si un message est correctement interprété : il vérifie d'abord l'existence des termes dans son ontologie. Le cas échéant, il doit interroger une base de synonymes et exécuter, pour chacun des termes inconnus,

une requête de confirmation auprès de l'agent émetteur A en lui proposant les synonymes obtenus.

Si l'ensemble W des termes encore inconnus a un cardinal inférieur à une valeur fixée comme acceptable pour la compréhension du message, alors l'agent **récepteur transmet cet ensemble W à l'agent émetteur A** : commence alors la phase de clarification.

Clarification : Pour chacun des termes de l'ensemble W, l'agent A cherche des synonymes dans sa propre ontologie. Il cherche également des liens possibles grâce à des relations de spécialisation, de généralisation et éventuellement d'autres. Évaluation de la pertinence. C'est l'évaluation des résultats d'une requête par rapport à la requête elle-même. Dans cette approche ONP, les résultats sont des URLs ou des documents ; il est donc facile de comparer les mots clés. Mise à jour. Cette dernière étape a pour objectif la mise à jour de l'ontologie de l'un ou des deux agents : un nouveau concept, une nouvelle distinction ou simplement un nouveau terme peuvent ainsi être introduits dans une ontologie.

Ces étapes sont au coeur du processus de négociation, mais signalons aussi que le protocole global est géré, à un niveau supérieur, par une machine à états dont l'objectif est de déterminer le comportement successif d'un agent.

Notons que lors de la phase d'interprétation nous avons évoqué l'utilisation d'un service de mots externe : il s'agit de la base WordNet qui permet d'obtenir des synonymes et d'autres mots pour guider le processus.

l'objectif de cette approche est la compréhension de messages. ONP suit une démarche de négociation « terme à terme » (et ne concerne que les termes) qui est dépendante des ressources utilisées pour l'interprétation : Il est en effet difficile de conserver la sémantique d'un terme en lui choisissant un synonyme au hasard, même si ce synonyme est ensuite confronté à l'ontologie de laquelle est issu le terme initial.

III.3.2. L'approche « MERIT / Infonomics »

Cette approche, tout comme les autres, s'attaque au problème d'hétérogénéité dans la représentation des connaissances **[46]**.

Le principe de départ est ici que les alternatives existantes nécessitent des connaissances *à priori* trop contraignantes, d'où l'hypothèse de l'apprentissage automatique de mappings : cela permet de s'affranchir de la nécessité d'avoir des concepts partagés,

des ontologies dérivées les unes des autres, et de

l'intervention humaine pour spécifier manuellement les relations entre différentes ontologies.

La méthode se base sur la mise en correspondance d'instances de concepts afin de pouvoir communiquer à propos d'un concept subsumant.

Elle se propose ainsi de résoudre une partie du problème de l'hétérogénéité des sources de connaissances : les conflits de noms, structurels et représentationnels.

Le processus d'apprentissage de cette approche se base sur les jeux de langage développés par Steels [69], et dont l'objectif est d'obtenir un système de communication consensuel en ne partant de rien : il s'agit globalement d'échanger des termes et de les évaluer.

L'utilisation du jeu de langage pour l'apprentissage de *mappings* vise tout d'abord à obtenir une attention conjointe des deux agents prenant part au dialogue : un agent transmet une instance à un autre agent, puis l'agent récepteur recherche le concept dont une instance a un maximum de mots en commun avec l'instance initialement transmise. C'est pour cela qu'il est nécessaire que des instances soient communes aux deux agents, exprimées en fonction de leurs ontologies respectives.

L'un des deux agents, nommons le A, peut ensuite commencer à établir un *mapping* entre les concepts terminaux du concept choisi : pour cela il a besoin que son interlocuteur, l'agent B, lui envoie des concepts terminaux de son concept, avec des instances correspondantes. Notons que les concepts terminaux doivent être identifiés de manière unique pour éviter des conflits

de nommage, car ils sont transmis simultanément : la hiérarchie des concepts n'existe alors plus. L'intérêt est ici de s'affranchir des différences structurelles des ontologies ; les concepts sont mis à plat. L'agent A cherche alors à établir des associations entre ses concepts terminaux et ceux de l'agent B. Il se base pour cela sur la proportion de mots communs dans les instances.

Des envois successifs de ce type permettent à l'agent A de maximiser des forces d'association liant les concepts terminaux. De plus, les associations sont exprimées grâce à des opérateurs spécifiques aux chaînes de caractères (*émerge* et *Split*) pour indiquer la relation entre deux concepts, et donc le moyen de transformer une instance d'un concept en l'instance d'un autre concept.

Le processus s'arrête quand l'agent A considère certaines associations comme correctes. Le *mapping* est alors final : c'est un *mapping* des concepts de l'agent A vers des concepts de l'agent B. Pour cette raison il est dit asymétrique, et ne permet donc pas à l'agent B de disposer de correspondances entre ses concepts et ceux de l'agent A.

Cette approche est une solution générique au problème d'hétérogénéité des ontologies. Elle souffre toutefois de quelques limitations. Tout d'abord, les *mappings* obtenus ne sont pas bidirectionnels. De plus, ils nécessitent un temps de calcul élevé à cause du parcours des concepts, des envois successifs d'ensembles { concepts + instances }, et du traitement des chaînes de caractères. La nécessité que les ontologies ont des instances de concepts en commun pour guider les associations limite également son utilisation dans le cas général.

III.3.3. L'approche « Alignement réciproque d'ontologies »

Cette approche présente une méthode d'alignement également basée sur l'échange d'instances d'ontologies, mais elle utilise, contrairement aux autres, une représentation ensembliste et définit des catégorisations du domaine qui servent à raisonner sur l'alignement à un niveau logique **[46]**.

Lors d'un « jeu d'alignement », des concepts sont échangés entre deux agents. Selon l'appartenance ou non d'un concept à une catégorie les liens entre les concepts sont mis à jour.

Cette méthode repose donc sur l'échange d'objets représentationnels pour mener une action collective coordonnée au sein d'un système multi-agents. L'objectif est la convergence des agents vers une ontologie partagée, à partir des catégorisations.

Description du cycle de base du jeu d'alignement :

Sélection aléatoire de deux agents A et B.

À choisi aléatoirement une catégorie c de son ontologie, et choisi une instance o de c. B reçoit o et repère une catégorie c' telle que o appartienne à c'. B choisi alors o' appartenant à c' (mais o' différente de o) l'envoie à l'agent A. A informe B sur l'appartenance de o' à la catégorie c.

A et B adaptent éventuellement leurs ontologies, ainsi que les matrices d'association.

Le processus décrit dans cette approche, comme dans l'approche MERIT, est basé sur la manipulation d'instances ; il faut que des objets soient connus de part et d'autre. Il s'agit donc d'une approche fondée sur une vision extensionnelle (voir chapitre 1). Il n'est pas non plus possible d'automatiser la communication entre deux agents quelconques, mais le processus permet toutefois l'adaptation d'ontologies et la convergence vers une ontologie commune à un groupe d'agents ; la convergence absolue n'a cependant pas été prouvée de manière formelle.

III.4. Conclusion

Dans ce chapitre, nous avons présenté quelques travaux sur les approches d'alignement existant dans la littérature en focalisant sur les approches appliquées dans le web sémantique et utilisant le format de représentation OWL recommandée par W3C. Ensuite nous avons présenté différentes approches qui s'attaquent au problème de l'hétérogénéité des connaissances et au problème d'interopérabilité entre agents qui en découle.

Le problème d'alignement d'ontologie dans web est un :

- Problème distribué : les ontologies de domaine développées séparément et dans des environnements différents et distribuées sur le web, la description des connaissances est exprimée pour chaque groupe de travaille.
- La solution de problème peut être distribué : comme on a déjà vu pour aligner des ontologies en se basant sur les calculs des similarités entre les entités des 2 ontologies ; le même algorithme appliquer pour tous les paires d'entités ; alors on peut appliquer l'algorithme sur plusieurs paires d'entités en parallèle pour réduire la durée d'exécution.
- La résolution coopérative du problème d'alignement : pour arriver à une qualité supérieure de l'alignement des approches combinées, plusieurs approches d'alignement sont utilisé comme des techniques des similarités hybrides (combinaison des techniques d'alignement d'ontologies lexicale, structurelle, sémantique) pour agréger un alignement entre 2 ontologies.

Les S.M.A sont des systèmes idéaux pour représenter des problèmes qui possèdent de multiples méthodes de résolution, de multiples perspectives. L'approche S.M.A est justifiée par les propriétés :

- La modularité.
- La vitesse, avec le parallélisme.
- Fiabilité, due à la redondance.
- Traitement symbolique au niveau de connaissances.
- La réutilisation et la portabilité.
- L'intervention des schémas d'interaction sophistiqués (coopération, coordination, négociation).

La technologie multi agents offre la possibilité à des agents spécialisés de travailler de façon parallèle et concurrente. De plus, les agents utilisent leurs capacités d'apprentissage pour s'adapter et interagir avec d'autres agents.

Dans le chapitre suivant nous présenterons notre contribution qui est une architecture basée agent pour l'alignement d'ontologie pour traiter le problème d'hétérogénéité d'ontologies de domaine et fournir un environnement qui assure le partage des ressources et des données entre les différents systèmes sur le web.

IV.1. Introduction

L'alignement d'ontologies dans le web sémantique représente un grand intérêt pour la gestion des connaissances partagées dans un environnement hétérogènes.

La littérature du domaine propose plusieurs méthodes d'alignement d'ontologies, Ces méthodes exploitent différents formats d'ontologies et repose sur le calcul des mesures de similarité. Ces méthodes sont souvent exploitées dans ces environnements de développement local pour répondre à des requêtes sémantiquement ou pour aligner des ontologies de domaine dans ces systèmes.

Le web sémantique est l'extension de web actuel, comme le web, le web sémantique sera: immense, ouvert, dynamique et hétérogène en plus il sera sémantique. Les ressources/la connaissance sont exprimées de différentes manières et doivent être réconciliées (aligner) avant d'être utilisées [3].

Dans ce chapitre nous allons décrire notre contribution qui est une « approche basée agent pour l'alignement d'ontologies : équivalence sémantique ».

Nous commençons la modélisation de notre architecture au quelle nous allons présenter les fonctions majeures de notre architecture en terme des modules et la spécification des différents agents logiciels exploiter, leurs rôles et leurs architectures internes. Ensuite nous présentons le fonctionnement du système ainsi que la communication entre les agents et l'interaction générale entre les agents pour gérer les alignements entre des ontologies de domaine distribuées dans le web et produire des nouveaux alignements entre les différentes ontologies distribuées. Enfin, la conclusion est présentée dans la dernière section.

IV.2. Modélisation du système

Un même domaine peut avoir plusieurs ontologies représentées de manière différente, alors si un utilisateur exprime une requête à partir d'une ontologie A, des systèmes à base de connaissances permet de répondre, mais avec des connaissances de ces ontologies de domaine B. De même, pour une question donnée nous pouvant avoir deux réponses syntaxiquement différentes, mais sémantiquement équivalentes.

L'architecture proposée dans ce travail est une approche basée agent pour l'alignement d'ontologies dans des systèmes à base de connaissance (**SBC**) répartis dans le web.

Elle définit l'architecture globale de notre système et l'interaction entre les différents agents de manière générale en utilisant une étude de cas pour voir le déroulement du système et expliquer les taches de chaque agent du système, ainsi que l'architecture interne de chaque

agent afin d'accomplir correctement l'ensemble des tâches du système d'alignement d'ontologies. Dans le contexte de notre étude, les composants fonctionnels correspondent aux différents agents constituant le système, et leurs modules internes.

La figure IV.1 suivante décrite l'architecture générale de notre approche :

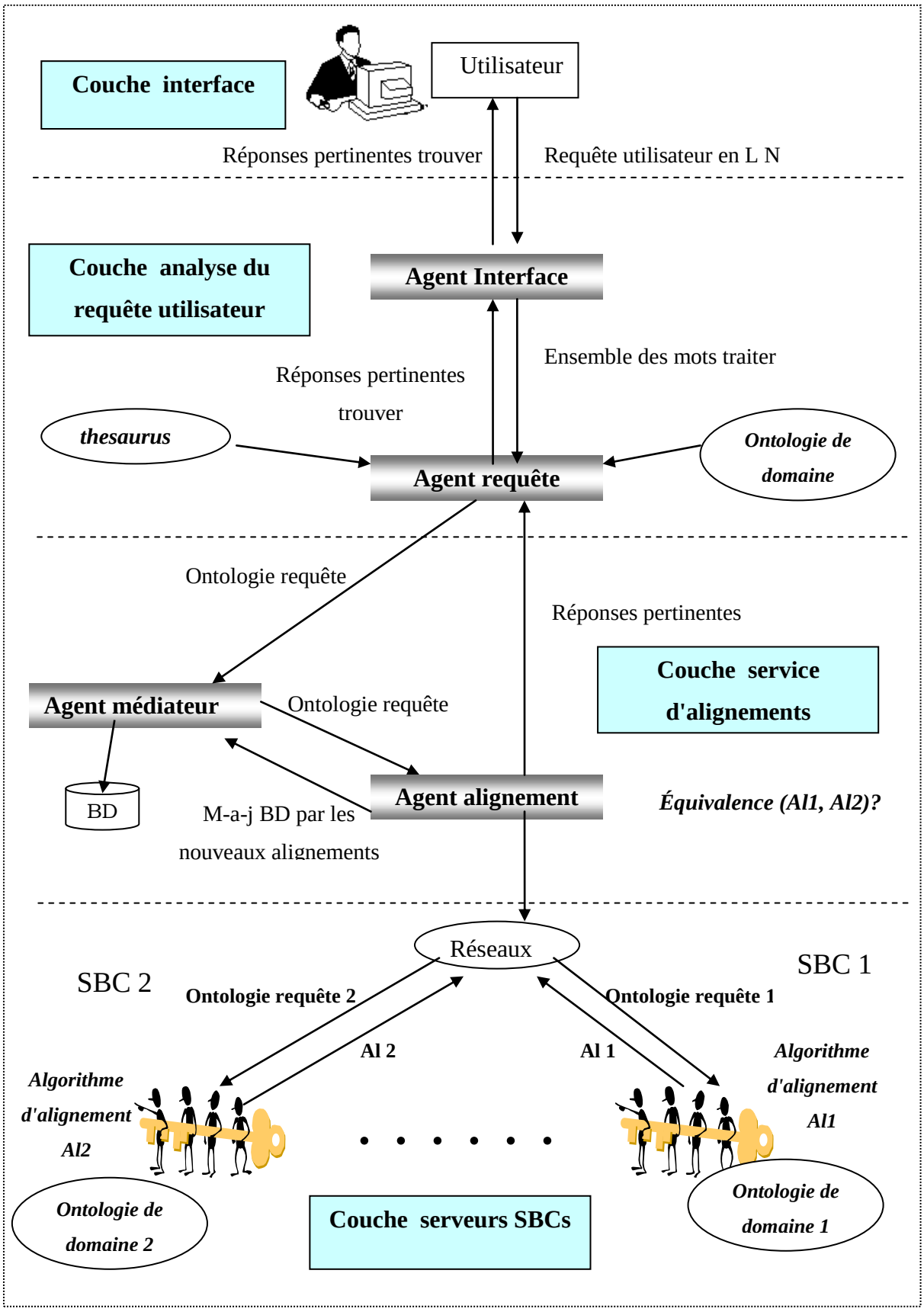


Figure IV.1 Conception globale d'une approche basée agent pour l'alignement d'ontologies

IV .2.1. La couche interface

Généralement l'interface est l'élément de référence qui permet à l'utilisateur de juger la qualité d'un système. Elle présente le seul moyen qui permet l'interaction directe entre le système et l'utilisateur.

Cette couche contient l'application qui permet au client d'interroger le système. Son rôle essentiel est de capturer le but de l'utilisateur afin de répondre le mieux possible à son besoin. Il faut noter qu'à ce niveau les besoins de client de recherche sont exprimés en terme des requêtes éditées en langage naturel comme une demande d'information dans un domaine visé et précis. Après l'exécution de processus de recherche, un ensemble d'information sera retourné à l'utilisateur comme une réponse à sa requête de recherche.

IV.2.2. La couche analyse du requête utilisateur

Cette couche contient les éléments nécessaires à l'extraction de la sémantique d'une requête écrite en termes de langage naturel et générer une nouvelle requête en utilisons les concepts d'ontologie de domaine local pour représenter la sémantique par une ontologie requête.

On trouve dans cette couche l'agent interface et l'agent requête

- **Agent interface:** le rôle de cet agent est de capturer le but de l'utilisateur afin de répondre le mieux possible à son besoin. À ce niveau les besoins de client de recherche sur un domaine visé sont exprimés en termes d'une requête générée en langage naturel. Dans cette étape les mots générés dans la requête doivent être normalisés (tokenisation, munisculation, et éliminer les mots vides (non sensé), …). . L'algorithme de normalisation est comme suit :

Pour chaque mot non vide (sensé) de la requête utilisateur

Algorithme Normalisation (nom)
résultat ← créer un ensemble vide
tokens ← Tokenisation(nom)
n ← nombre de tokens dans tokens
Pour i **de** 1 **à** n
ex ← Minusculisation(ex)
reculs ← Ajouter(ex, result)
Fin Pour

La requête normalisée devient une chaîne de concepts envoyés à l'agent requête.
Après l'exécution de processus de recherche, un ensemble des réponses pertinentes sera retourné à l'utilisateur.

- **Agent requête :** Le rôle de cet agent est la construction une ontologie requête qui représente la sémantique de la requête utilisateur interprétable par les agents logiciels d'autres systèmes.

Ce processus de construction commence par l'enrichissement sémantique des mots filtré par l'agent interface en ajoutant des synonymes fournis par un thésaurus de domaine par exemple : automobile→ voiture, car, bagnole.
Ensuite cet agent exploite une ontologie de domaine locale de notre système pour trouver des concepts similaires à celle de la requête entrée et générée des relations sémantiques entre ces concepts à fin d'arrivé à une **ontologie requête** ayant une expressivité sémantique c'est à d une requête utilisateur exprimé sémantiquement par les concepts de notre ontologie. L'algorithme de calculer la similarité est inspiré sur l'algorithme d'alignement d'ontologie **Asco 2** [60] est comme suit :

```
Algorithme  Similarité_des_Noms(nom1, nom2)
tokens1 ← Normalisation(nom1)
tokens2 ← Normalisation(nom2)
n1 ← nombre de tokens dans tokens1
n2 ← nombre de tokens dans tokens2
somme1 ← 0
Pour i de 1 à n1
  sim ← Sim_Max_Dans_Ensemble(tokens1[i], tokens2)
  somme1 ← somme1 + sim
Fin Pour
somme2 ← 0
Pour j de 1 à n2
  sim ← Sim_Max_Dans_Ensemble(tokens2[j], tokens1)
  somme2 ← somme2 + sim
Fin Pour
Si n1 = n2 = 0
  résultat ← 1
Sinon
résultat ← (somme1 + somme2) / (n1 + n2)
Fin Si
Retourner résultat
```

Cet agent envoi l'ontologie requête à la couche suivante et reçoit les informations retournées concernant le besoin utilisateur demandé.

IV.2.3. Couche service d'alignement

Cette couche regroupe tous les éléments nécessaires à l'exécution de processus de reformulation d'une ontologie requête à une requête interprétable par les différents SBCs de destination. Les concepts d'ontologie requête sont remplacés par son concepts équivalents dans les ontologies des SBCs de destination s'il existe dans notre base d'alignement.

À partir des réponses reçues de ces SBCs on déduit des nouveaux alignements entre les différentes ontologies proches à fin de fournir un médiateur sémantique qui permet d'interopérabilité sémantique entre des différents SBCs distribués dans le web.

Dans cette couche on trouve les agents suivants :

- Agent médiateur : l'ontologie requête exprime la sémantique des besoins utilisateur à partir de son ontologie locale de domaine. L'agent médiateur est un agent qui traite l'alignement d'ontologies des différentes SBCs de même domaine, il effectue des traitements sur l'ontologie requête utilisateur afin de la reformuler en une ontologie requête écrite avec des concepts interprétables pour chaque SBC. Il joue le rôle d'un médiateur entre les ontologies de domaine des différentes SBCs pour partager les connaissances de domaine et l'interopérabilité sémantique entre ces systèmes.

L'agent médiateur exploiter une base de données alignement qui contient des relations d'équivalence déduite entre les différentes ontologies de domaine des SBCs pour être déplacé vers le site serveur cible afin de trouver des informations adéquates à la requête utilisateur.

- Agent alignement : Cet agent joue le rôle d'interface avec l'environnement externe, c'est la partie la plus essentielle dans notre contribution pour gérer et déduire des nouveaux alignements et sélectionner les réponses proches et pertinentes des résultats reçus.

Les ontologies requête reformuler par l'agent médiateur sont envoyés par l'agent alignement vers chaque serveur SBC qu'il connaît, les réponses reçues sont générées au niveau de chaque système à partir de son ontologie locale Oi qui exploite un algorithme d'alignement ALi, alors des nouveaux concepts équivalents sont retournés à l'agent alignement comme une réponse ; l'agent alignement exploite une métrique proposée pour sélectionner les réponses pertinentes, si le nombre de concepts équivalents retournés divisés par le nombre de concepts envoyés dans l'ontologie requête est supérieur à un seuil prédéfini, la réponse est considérée pertinente sinon la réponse retournée considère non pertinente la métrique est comme suit :

Seuil s = (nombre Ci eq /nombre C source) .

Ensuite on déduit des nouveaux alignements qui seront ajoutés à la base d'alignement situé au niveau de l'agent médiateur pour reformuler les nouvelles requêtes à ces SBCs exploitant les concepts appropriés.

IV.2.4. Couche serveurs SBCs

Cette couche représente l'environnement externe de notre système c'est un environnement ouvert ,dynamique et distribuer .

Cette couche capable de recevoir des requêtes, de les traiter et de retourner les résultats. Cette couche regroupe l'ensemble des SBCs associé au chaque site sur le web.

Les connaissances des SBCs (ontologies) sont caractérisées par :

- L'hétérogénéité : chaque ontologie choisie et conçue indépendamment des autres (structure hiérarchique, langage de représentation , langue utilisée, etc…)
- La distribution : les ontologies de domaine proviennent de plusieurs systèmes distribués.

Nous allons présenter dans les sections suivantes les spécifications des différents agents, ainsi que les concepts liés à leurs fonctionnements.

IV.3. Spécification des agents

Nous décrivons dans ce qui suit les rôles et les fonctionnalités des différents agents dans notre système tout en illustrant les messages échangés entre eux.

Ces types d'agents sont :

- ✓ Agent d'interface.
- ✓ Agent requête
- ✓ Agent médiateur.
- ✓ Agent d'alignement.

IV.5.1. Agent d'interface

Cet agent peut être vu comme un simplificateur permettant aux utilisateurs d'interagir avec le système. C'est un agent qui est responsable principalement d'acquérir toutes les requêtes des utilisateurs, envoyer ces requêtes aux agents requête et présenter les résultats aux utilisateurs.

L'agent d'interface doit assurer les services suivants :

- ➢ Assurer la bonne communication entre l'utilisateur et le système.

- Filtrer (normalisé, et éliminer les mots vides (non sensé)) les informations écrites dans l'interface utilisateur, les informations filtrer seront interprétées par l'agent requête comme une requête d'entrée.
- Renvoyer les résultats reçus à l'utilisateur comme une réponse à sa requête.

L'architecture de l'agent interface est comme suit :

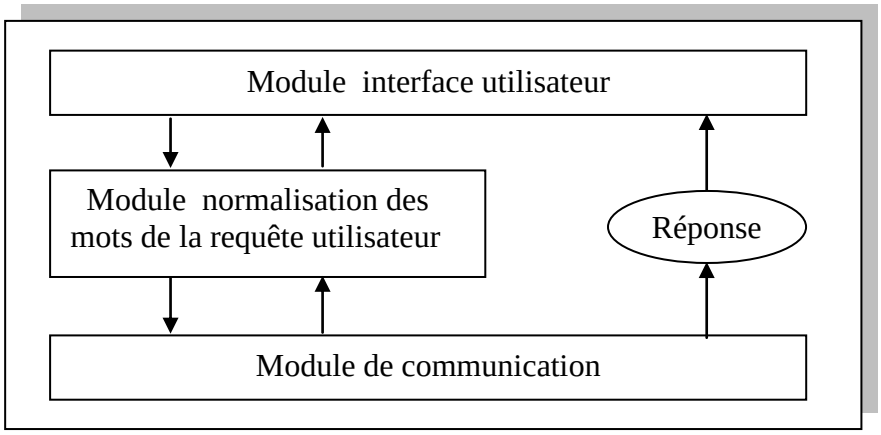


Figure IV.2 : Architecture de l'agent d'interface.

Les modules utilisés pour le développement d'agent interface sont les suivants :

Module interface utilisateur : ce module est responsable de la présentation de ce qui est disponible dans le système et ses différentes fonctionnalités, sous forme d'une interface utilisateur.

Module de communication : ce module permet aux agents d'échanger des messages entre eux et avec les éléments constituant le système. À travers ce module, l'agent interface communique avec l'agent requête.

Module normalisation des mots de la requête utilisateur : ce module permet de traiter les informations (le besoin de client) décrites dans la demande utilisateur à partir d'interface utilisateur. Il filtre (normaliser et éliminer les mots vides (non sensé)) les informations écrites dans l'interface utilisateur, et les informations filtrées seront interprétées par l'agent requête comme une requête d'entrée.

IV.5.2. Agent requête

Généralement la requête exprime le besoin de l'utilisateur. Afin de réaliser effectivement le but, la requête est toujours exprimée par ensemble de concepts de domaine.

L'agent requête enrichi la sémantique de la requête utilisateur en exploitant un thesaurus de domaine et construit l'ontologie requête qui représente la sémantique de besoin utilisateur interprétable par les autres agents logiciels en exploitant une ontologie de domaine locale de notre système. Il effectue une série de traitement sur la requête utilisateur écrite en terme d'ensemble des concepts filtrés afin de la reformuler en une requête écrite en terme d'une ontologie requête. Il joue le rôle de traducteur entre l'utilisateur qui pose une requête en langage naturel et les agents logiciels qui interprètent la sémantique des requêtes en forme des ontologies.

Pour chaque ensemble de concepts (requête utilisateur) l'agent requête va exploiter un thesaurus de domaine pour les enrichir par des concepts équivalents (des synonymes). Lorsque cette tache est terminée, cet agent génère des relations entre ces concepts pour réaliser une requête ontologie qui représente l'information demandée par l'utilisateur.

L'agent requête doit assurer les fonctionnalités suivantes :

- La réception de la requête de l'utilisateur depuis l'agent d'interface.
- L'enrichissement sémantique de la requête par des synonymes générés à partir d'un thesaurus.
- La construction d'une ontologie requête en exploitant une ontologie de domaine.
- L'envoi de l'ontologie requête à l'agent recherche.
- La réception de la réponse pertinente depuis l'agent recherche.
- Le renvoi de la réponse à l'agent d'interface.

L'architecture de cet agent est comme suit :

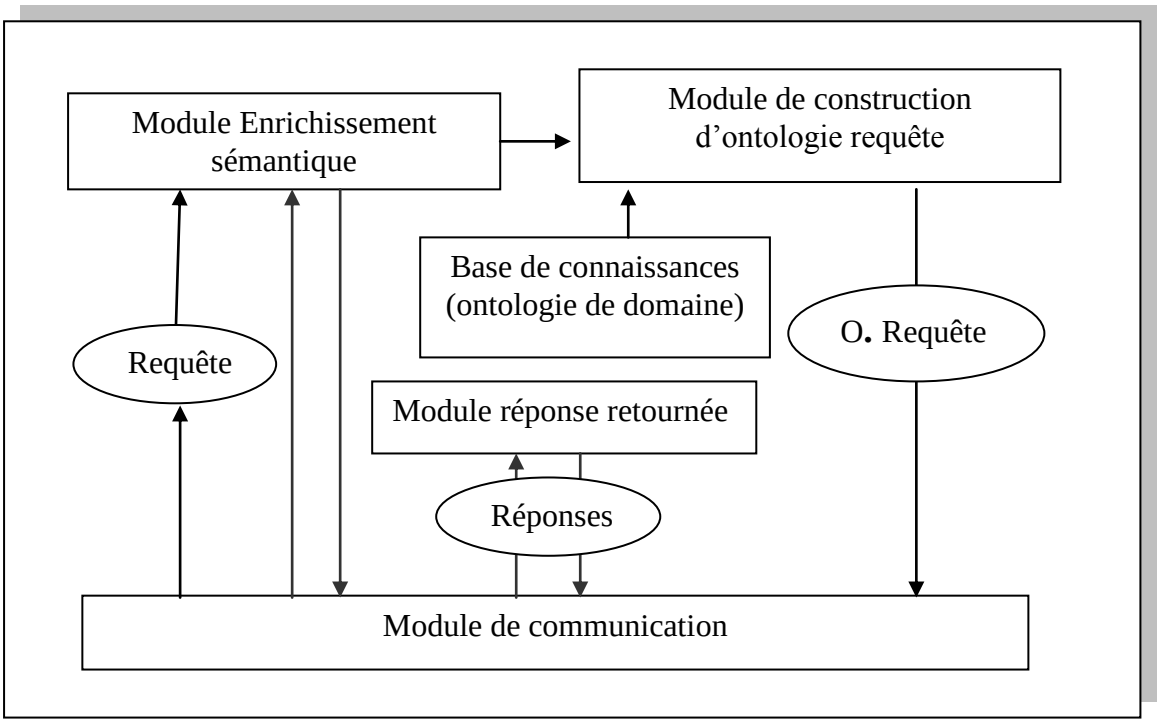


Figure IV.3 : Architecture de l'agent requête.

Les modules utilisés pour le développement d'agent interface sont les suivants :

❖ **Module Enrichissement sémantique :** Ce module permet d'enrichir la sémantique de la requête utilisateur par l'ajout des synonymes des concepts entrée dans la requête en exploitant un thesaurus de domaine.

❖ **Module de construction d'ontologie requête :** ce module est responsable du calcule de la similarité sémantique entre les entités de la requête utilisateur et d'ontologie de domaine de notre système. Il utilise les concepts équivalents de l'ontologie trouvé et les relations entre eux pour construire l'ontologie requête qui modélise la sémantique de la requête utilisateur exprimée par les concepts de notre ontologie de domaine interprétable par les autres systèmes.

❖ **Module réponse retournée :** ce module est responsable de la réception des résultats pertinents retourner et renvoyer à l'agent interface.

❖ **Module de communication :** ce module se charge de la construction des messages et leur envoi, et aussi la réception des messages. À travers ce module, l'agent requête communiqué avec les autres agents interface et l'agent médiateur et aussi ce module se charge de communiquer avec le thesaurus de domaine.

❖ **Base de connaissances (ontologie de domaine) :** la base de connaissance dans cet agent contient l'ontologie de domaine de notre système.

Les entités dans l'ontologie OWL DL sont décrites en employant des constructeurs (des primitives) du langage OWL. Ces descriptions sont interprétables sous forme des triplets RDF : (sujet, prédicat, objet), où les entités à décrire sont les sujets des triplets, les prédicats des triplets sont des primitives de OWL, et les objets sont les ressources (par exemple les entités, les littéraux…) , La description d'une classe ou d'une relation dans une ontologie OWL est donc réalisée par un ensemble de triplets RDF, dont le sujet correspond à la classe ou à la relation en question. Les prédicats dans les triplets de la description d'une entité sont des primitives de OWL, qui sont les propriétés du langage OWL et celles du langage RDF.

IV.5.3. Agent médiateur

L'agent médiateur traite l'alignement d'ontologies des différents SBCs de même domaine, il effectue des traitements sur l'ontologie requête utilisateur afin de la reformuler en une ontologie requête écrite avec des concepts interprétable pour chaque SBC.

L'agent médiateur doit assurer les fonctionnalités suivantes :

- Réception de l'ontologie requête depuis l'agent requête.
- Retourner les concepts équivalents existants pour chaque adresse SBC connue dans la base de données alignement.
- Reformuler l'ontologie requête en utilisant les concepts retournés pour chaque SBC.
- Envoyer l ontologie requête avec les adresse destinataire pour chaque SBC concerné et l'ontologie requête de l'agent requête avec une adresse de diffusion pour les autre SBCs.
- Ajouter des nouveaux concepts équivaut retourné par l'agent alignement, aussi des nouveaux SBC découverte.

L'architecture de cet agent est comme suit :

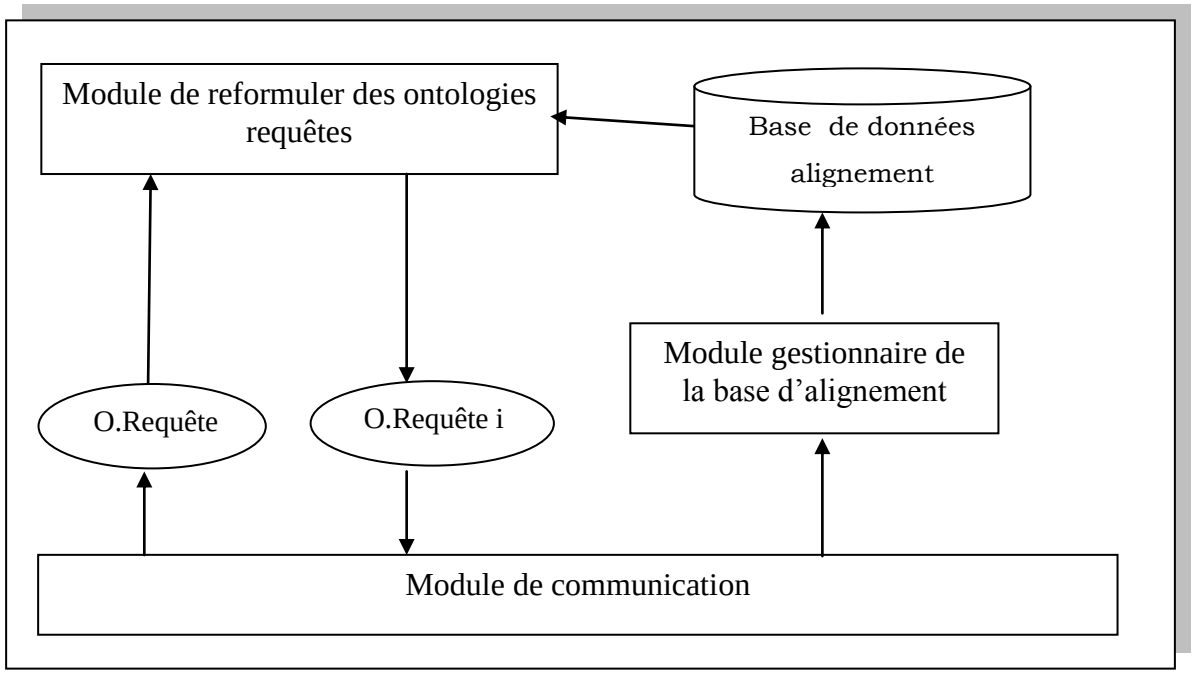


Figure IV.4 : Architecture de l'agent médiateur.

❖ **Module de reformuler des ontologies requête :** ce module est responsable de la réception de l'ontologie requête depuis l'agent requête, il exploite la base d'alignement pour trouver des concepts équivalents existants pour les SBCs connus dans la base de données alignement; il est aussi responsable de reformuler les ontologies requête et les envoyer à chaque SBC concerné et envoyer l'ontologie requête de l'agent requête vers une adresse de diffusion pour les autres SBCs.

❖ **Module gestionnaire de la base d'alignement :** Ce module doit ajouter des nouveaux concepts équivalents retournés par l'agent alignement, aussi des nouvelles SBCs découvertes à la base de données alignement aussi modifier ou supprimé d'autres concepts changés.

❖ **Module de communication :** ce module se charge de la construction des messages et leur envoi, et aussi la réception des messages. À travers ce module, l'agent médiateur communique avec les autres agents requête et l'agent alignement.

❖ **Base de données alignement :** Elle contient les relations d'équivalence sémantique entre les concepts de différentes d'ontologies de domaine créées et gérée par le module mis à jour de la base d'alignement**.**

IV.5.4. Agent alignement

Cet agent joue le rôle d'interface avec l'environnement externe, c'est la partie la plus essentielle dans notre contribution pour gérer et déduire des nouveaux alignements et sélectionner les réponses proches et pertinentes des résultats reçus.

L'agent médiateur doit assurer les fonctionnalités suivantes :

- La réception des l'ontologie requête depuis l'agent médiateur.
- L'envoi de chaque requête à la SBC approprie.
- La réception des réponses des différents systèmes et les traités.
- Le calcul de la similarité entre l'ontologie source et destinataire pour sélectionner les réponses pertinentes.
- L'envoi des nouveaux concepts découvrir à l'agent médiateur pour générer des nouveaux alignements.

L'architecture de cet agent est comme suit :

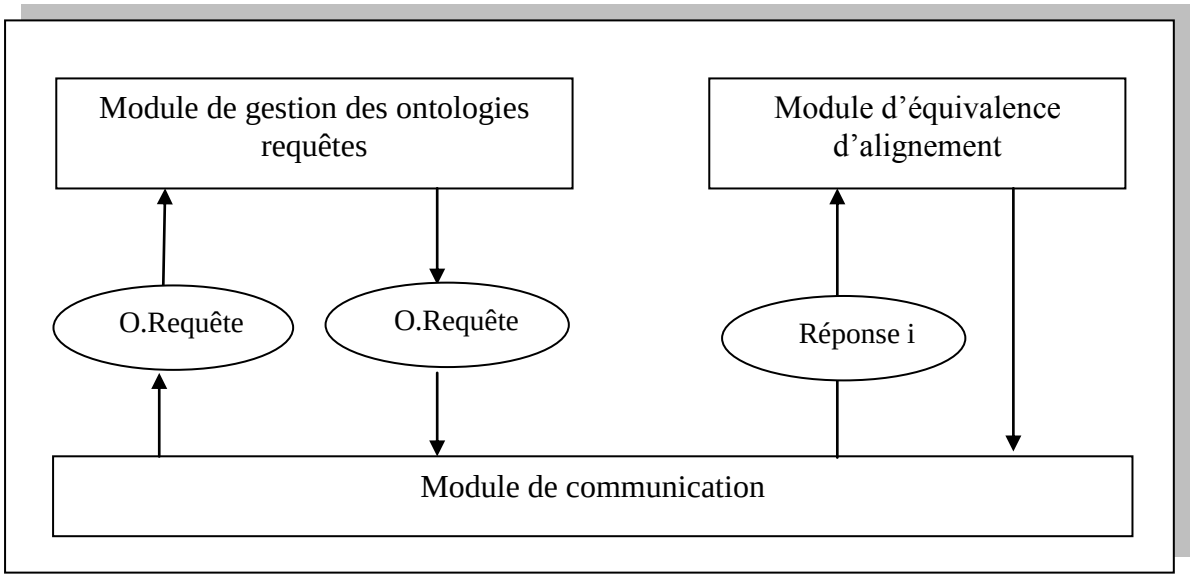


Figure IV.5 Architecture de l'agent alignement.

❖ **Module de gestion des ontologies requête :** Ce module se charge de la réception de l'ontologie requête depuis l'agent médiateur et envoie chaque requête à la SBC approprié.

❖ **Module d'équivalence d'alignement :** ce module se charge de la réception des réponses des différents systèmes et les traités ; Il calcule le résultat de nombre de concepts des réponses retournées divisées sur le nombre des concepts d'ontologie requête source pour sélectionner

les réponses pertinentes. Si le résultat calculé est supérieur à un seuil prédéfini, alors envoyer les nouveaux concepts découverts à l'agent médiateur pour générer des nouveaux alignements.

Module de communication : ce module se charge de la construction des messages et leur envoi, et aussi de la réception des messages. À travers ce module, l'agent alignement communique avec l'environnement externe.

IV.4. Fonctionnement du système

L'approche qu'on va présenter a pour but de gérer l'hétérogénéité des ontologies de domaine réparties en exploitant des réponses retournées (recherche d'informations) dans un système réparti à partir des requêtes de l'utilisateur, cela est possible en utilisant les ontologies de domaine comme un moyen de partage des connaissances et l'alignement d'ontologie pour gérer l'hétérogénéité de ces ontologies en exploitant les agents logiciel dans la réalisation de ce système.

Pour voir le fonctionnement de notre approche nous exploitant le domaine de recherche d'informations sémantique (web sémantique) comme un environnement de travail, notre système a en entrée une demande d'information sous forme d'une requête de l'utilisateur écrite librement (langage naturel). Cette requête est traduite alors en requête représentée par la sémantique de besoin utilisateur interprétable par la machine représentée par une ontologie requête. La réponse à la requête est construite à partir des résultats fournis par les différentes sources de données reparties.

L'agent interface capture le besoin rechercher sur le domaine visé en terme d'une requête généré en langage naturel, dans cette étape les mots générés dans la requête doivent être normalisé (tokenisation, munisculation, , et éliminer les mots vides (non sensé),…), la requête devient alors comme ensemble des mots traités, et envoyés à l'agent requête.

Ensuite l'agent requête construit une ontologie requête qui représente la sémantique de la requête utilisateur, ce processus commence par l'enrichissement sémantique des mots de requête par l'ajout des synonymes fournit par un thésaurus de domaine par exemple : automobile→ voiture, car, bagnole.

Ensuite cet agent exploite une ontologie de domaine locale de notre système pour trouver des concepts similaires à ceux de la requête entrée puis générée des relations sémantiques entre ces concepts à fin d'arriver à une **ontologie requête** ayant une expressivité sémantique c'est-

à-dire une requête utilisateur exprimé par les concepts de notre ontologie de domaine, l'ontologie requête est envoyée à l'agent médiateur.

L'agent médiateur effectue des traitements sur l'ontologie requête utilisateur afin de la reformuler en une ontologie requête écrite avec des concepts interprétables pour chaque SBC qui existe dans la base d'alignement. L'agent médiateur exploite une base de données alignement qui contient des relations d'équivalence entre des concepts d'ontologies des différents SBCs pour reformuler les ontologies requête et les envoyer à l'agent alignement.

Ensuite, pour chaque ontologie requête l'agent alignement va envoyer un message de recherche vers le SBC adéquat et une multi diffusion des ontologies requête vers les autres serveurs qui ne contiennent pas des relations d'équivalence.

Après avoir retourné les réponses des différents **SBCs**, pour chaque réponse retournée si le nombre des concepts retourné divisé sur le nombre des concepts envoyer (ontologie requête) supérieur à un seuil prédéfini alors la réponse est considérée pertinente et les nouveaux concepts trouvés sont envoyés à l'agent médiateur et les liens de ces réponses sont envoyés à l'agent requête comme réponse destiné à l'utilisateur.

L'agent médiateur reçoit les nouveaux concepts et fait des mises à jour sur la base d'alignement et déduit des nouvelles relations d'équivalences.

L'agent requête reçoit les réponses pertinentes et l'envoi à agent interface qui affiche à l'utilisateur les réponses pertinentes.

IV.5. La communication entre les agents

Un système à base d'agents fournit en général des primitives de communication permettant aux agents de communiquer entre eux, aussi bien qu'avec les services de plate-forme ou de l'environnement. Ces primitives de communication prennent la forme d'envois de messages, d'appels de procédures (méthodes), ou bien l'utilisation d'un blackboard. Par conséquent, les langages de communication entre les agents standardisés devront être fournis.

Le langage KQML a été proposé pour supporter la communication entre les agents. Ce langage définit un ensemble de messages (appelé performatif) et des règles qui définissent les comportements suggérés pour les agents qui reçoient ces messages.

Le langage ACL (Agent Communication Language), successeur de KQML, fournit une sémantique plus riche. Ce langage est fondé par la FIPA qui s'occupe de standardiser les

communications entre agents. ACL est basé également sur la théorie des langages et se rapproche au niveau des actes du langage avec KQML, mais pas au niveau de la sémantique, qui a subi une nette amélioration dans ACL.

FIPA fournit des performatifs basés sur l'acte du discours et une syntaxe standard pour les messages. Ces messages sont basés sur la théorie d'acte de discours, qui est le résultat de l'analyse linguistique de la communication humaine. La base de cette théorie est de produire une action à partir du langage. Dans le langage FIPA-ACL, aucun langage spécifique pour la description des contenus des messages n'est imposé. Plusieurs langages peuvent être utilisés pour la description du contenu des messages échangés tels que le langage KIF, le langage sémantique (SL), Prologue, le langage XML, RDFS ,OWL ….

Dans notre système l'interaction entre les différents agents sera faite par envoi de message et nous allons utilisant le langage OWL pour représenter l'ontologie qui modélise la sémantique du contenu de message.

IV.6. Interaction générale entre les agents

Pour bien comprendre le fonctionnement de notre architecture, nous présentons le protocole d'interaction dans une séquence d'étapes suivies :

1. Un client (**User**) se connecte à notre système, et lance une requête qui spécifie la tâche à réaliser.

2. L'agent interface (**AI**) reçoit la demande de l'utilisateur, la filtre et l'envoi à l'agent requête comme une requête d'entrée.

3. L'agent requête (**AR**) construit l'ontologie requête en exploitant un thesaurus et une ontologie de domaine pour enrichir la requête avec des synonymes et produire des relations sémantiques entre ces concepts enfin d'envoyer l'ontologie requête à l'agent médiateur.

4. Celle-ci doit être reformulée également par l'agent médiateur (**AM**) en un ensemble des ontologies requête interprétables par les différents SBC hétérogène et envoyer à l'agent alignement (**AL**).

5. Ensuite, pour chaque ontologie requête l'agent alignement (**AL**) va envoyer un message de recherche vers le SBC adéquat.

6. Une multi diffusion d'ontologie requête est envoyée par cet agent vers les autres serveurs qui ne contiennent pas des relations d'équivalence.

7. L'agent de ressource va extraire les informations requises de sa base de données locale. Ces données seront représentées par des concepts équivalents avec des liens vers l'information pertinente transmise ensuite à l'agent alignement (il faut signaler qu'il pourrait avoir un cas contraire, où l'agent de ressource répond négativement par un message indiquant que les informations demandées n'existent pas).

8. Les agents ressource (**AR**) retournent les résultats trouvés à l'agent alignement

9. La réception des résultats envoyés par tous les agents de ressource.

10. Le regroupement de tous les résultats reçus par l'agent alignement. Il fait une métrique pour la comparaison de ces réponses pour sélectionner les réponses pertinentes, qui sont finalement envoyées à l'agent interface.

11. Des nouveaux alignements déduits seront envoyés à l'agent alignement pour faire des mis à jours de sa base d'alignement.

12. L'agent requête envoyait les résultats à l'agent interface.

13. L'agent médiateur fait des nouveaux mis à jours sur sa base d'alignement.

14. Les réponses sélectionnées seront retournées à l'utilisateur comme une réponse à sa requête initialement posée.

15. La réception de la réponse par l'utilisateur.

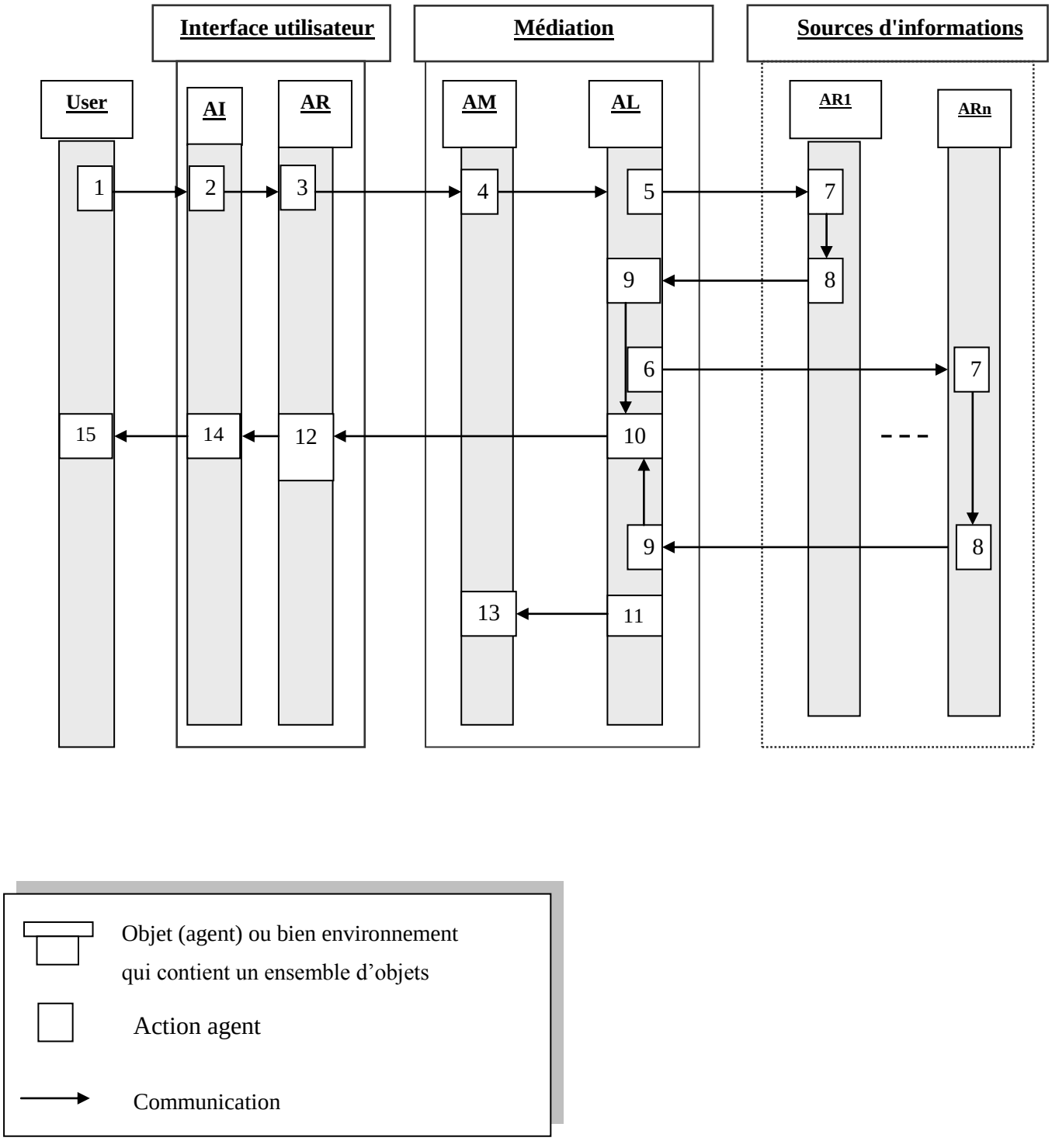


Figure IV.6 Diagramme d'interaction générale entre les agents.

IV.7. Conclusion

Dans ce chapitre, nous avons présenté une architecture basée agent pour l'alignement d'ontologies dans le web sémantique en exploitant la relation d'équivalence sémantique entre les concepts des différentes ontologies. Cette architecture comporte les concepts nécessaires pour assurer les exigences que nous avons pris en compte afin d'assurer le bon fonctionnement du système proposé. Les rôles et les structures internes des différents agents sont illustrés, les mécanismes d'interaction et de communication sont discutés.

Néanmoins, une étude de cas est nécessaire pour voir les différentes idées dans un environnement d'application. Cette étude de cas va permettre d'aborder la phase d'implémentation. Cette dernière phase fera l'objet du chapitre suivant pour la validation de notre approche.

V.1. Introduction

Dans le chapitre précédent de ce mémoire, nous avons proposé une approche basée agent pour l'alignement d'ontologies : « équivalence sémantique ». Afin d'illustrer les différentes idées et concepts inclus dans l'architecture proposée, nous allons utiliser cette architecture comme base pour une étude de cas dans le domaine de recherche d'informations sémantique sur le web.

Le but est de dérouler les principaux aspects de notre architecture sur une étude de cas sur le domaine sélectionné .

Notre système est composé de deux principaux modules qui sont :

- **Module d'analyse du besoin utilisateur**

Ce module contient une couche interface pour capturer la requête utilisateur et afficher les résultats, et contient aussi les éléments nécessaires à la représentation de la sémantique d'une requête de recherche écrite en langage naturel. Il génère une nouvelle requête qui représente la sémantique de besoin utilisateur interprétable par la machine (ontologie requête).

- **Module de médiation**

Il traite les ontologies requête construites par le module précédant et retourne les résultats de la recherche. Le processus se fait sur deux étapes : D'abord une couche médiatrice reformule l'ontologie requête pour chaque destinataire connue et les envoi, diffuse l'ontologie requête pour les autres dans le web, ensuite il sélectionne les résultats pertinents parmi ceux qui sont reçus et déduire des nouveaux alignements a fin de faire des mises à jour sur la base de données qui contient les alignements (relation d'équivalence) entre les concepts exploités par les différents systèmes connus ou découverts distribués sur le web.

Nous présenterons alors, dans la première partie de ce chapitre, l'ensemble des outils de développement utilisés. Dans la seconde partie, nous présenterons le processus d'implémentation en utilisant une étude de cas simple de déroulement de notre approche, ainsi que les principales interfaces qui le composent à travers des fenêtres de capture.

V.2. Environnement de développement

Avant de commencer l'implémentation de l'architecture conceptuelle de notre système, nous allons tout d'abord spécifier les outils utilisés.

Dans cette section, nous présentons les outils et les logiciels que nous avons utilisés : L'éditeur d'ontologie Protégé, le vocabulaire Wordnet 2.1 browser interface, la plateforme JADE (Java Agent DEvelopement framework) et l'environnement java NetBeans IDE 9.1.2.

V.2.1. Choix de l'éditeur d'ontologies

Pour l'implémentation des ontologies de domaine, nous avons opté pour l'éditeur d'ontologies Protégé. Plusieurs raisons ont motivé notre choix :

1. Protégé est un éditeur open source et gratuit.

2. Protégé permet d'importer et d'exporter des ontologies dans les différents langages d'implémentation d'ontologies (RDF-Schéma, OWL, DAML, OIL,...etc.)

3. Protégé possède une interface modulaire, ce qui permet son enrichissement par des modules additionnel (plugins).

4. Protégé permet l'édition et la visualisation d'ontologies.

5. Protégé permet le contrôle de la cohérence de l'ontologie par des vérifications de contraintes.

Pour la spécification de l'ontologie, nous utilisons Protégé qui générera un fichier en sortie sous le format OWL basé sur la syntaxe Xml compréhensible et traitable par la machine. L'installation de cet outil requiert la l'installation de la machine virtuelle java JVM. On lance le logiciel à partir du fichier JAR exécutable *org.eclipse.osgi*.

V.2.2. Le vocabulaire Wordnet

WordNet [74] est un système de référence lexicologique. Les noms, les verbes, les adjectifs et les adverbes en anglais sont organisés en des ensembles des synonymes. Ces ensembles des synonymes, appelés synsets, sont reliés par différentes relations sémantiques.
En utilisant WordNet, nous pouvons trouver les synonymes d'un mot ou d'un terme. Par exemple, l'un peut choisir le terme « Personne » pour le nom de la classe dénotant un individu, mais un autre peut décider d'employer le terme « Humain » à la place.

V.2.3. La plate-forme JADE (Java Agent DEvelopement framework)

À l'heure actuelle, il existe diverse plate forme pour le développement d'agents, ils peuvent être classifiés selon la nature des agents supportés, l'interaction, la communication entre agents et les standards utilisés.

V.2.3.1 Description générale de la plate-forme JADE

JADE (Java Agent DEvelopement framework) [70] est une plate-forme multi-agent créée par le laboratoire TILAB. JADE permet le développement de systèmes multi-agents et d'applications conformes aux normes FIPA [71]. Elle est implémentée en JAVA et fournit des classes qui implémentent « JESS » pour la définition du comportement des agents. Toute la communication entre agents est exécutée par messages FIPA ACL [71].

JADE fournit les facilités suivantes :

Une plate-forme agent distribuée : la plateforme peut être distribuée (partagée) entre plusieurs hôtes connectées via RMI de Java, de telle façon qu'une seule application Java, par conséquent une seule "Machine Virtuelle Java" est exécutée sur chaque hôte.

Une interface utilisateur graphique (GUI): l'interface GUI assure un traitement plus commode de la plate-forme, elle permet à l'utilisateur d'exécuter plusieurs ordres tels que créer un nouvel agent dans la même plate-forme, cloner l'agent, le déplacer, le suspendre, le tuer, etc.…

Un support d'exécution : pour les activités multiples, parallèles et concurrentes des agents via le modèle du comportement (behaviour).

Un transport efficace des messages ACL : à l'intérieur de la même plate-forme.

Une bibliothèque de protocoles : compatibles aux standards FIPA et prêts à être employés pour gérer l'interaction inter-agent.

Lorsqu'on lance la plate-forme, on remarque que JADE a trois modules qui sont actifs au démarrage de la plate-forme, ces modules sont (voir Figure 5.4) :

DF « Director Facilitor » fournit un service de « pages jaunes» à la plate-forme ;

ACC «Agent Communication Channel » gère la communication entre les agents ;

AMS « Agent Management System » supervise l'enregistrement des agents, leur authentification, leur accès et l'utilisation du système.

V.2.4. Choix du Langage de programmation

Pour le choix de programmation de notre système, nous avons opté pour le langage JAVA et cela pour de nombreuses raisons :

1. JAVA est un langage orienté objet simple, qui réduit le risque des erreurs d'incohérences.

2. Il est indépendant de toute plate-forme, il est possible d'exécuter des programmes JAVA sur tous les environnements qui possèdent une Java Virtual Machine (JVM).

3. Il est doté d'une riche bibliothèque de classes, comprenant la gestion des interfaces graphiques (fenêtres, menus, graphismes, boites de dialogue, contrôles), la programmation multi-threads (multitâches), la gestion des exceptions.

4. Il permet d'accéder d'une manière simple aux fichiers et aux réseaux (notamment Internet).

5. Il est caractérisé aussi par la réutilisation de son code ainsi que la simplicité de sa mise en oeuvre.

7. Il est compatible avec L'API JENA, ce qui nous permet la manipulation, le parcours et la modélisation des documents OWL.

L'environnement de développement utilisé pour programmation Java est NetBeans IDE 6.9.1

V.2.4.1 Description de NetBeans IDE

NetBeans [72] est un environnement de développement intégré (IDE) open source. Il est développé par Sun [73] et se trouve sous licence CDDL (Common Development and Distribution License). En plus de Java, il propose tous les outils nécessaires à la création d'applications professionnelles pour les particuliers, les entreprises, le web et les applications mobiles avec le langage C / C + +, et même les langages dynamiques tels que PHP, Javascript, Groovy, Ruby XML et HTML. NetBeans IDE est facile à installer et à utiliser. Il comprend toutes les caractéristiques d'un IDE moderne (coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web, etc).

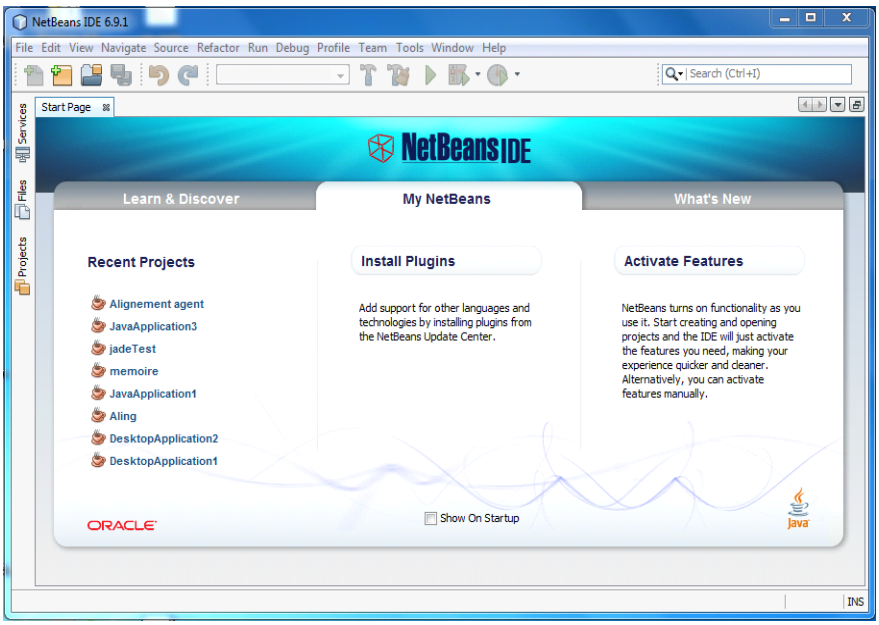


Figure V.1 : NetBeans IDE 6.9.1

Pour intégrer la plateforme JADE dans l'environnement NetBeans IDE il suffit d'ajouter les fichiers jar nécessaire correspond au plate forme Jade :

- **Les fichiers jar de JADE :**
- jade.jar
- http.jar
- iiop.jar
- jadeTools.jar
- CalendarBean.jar

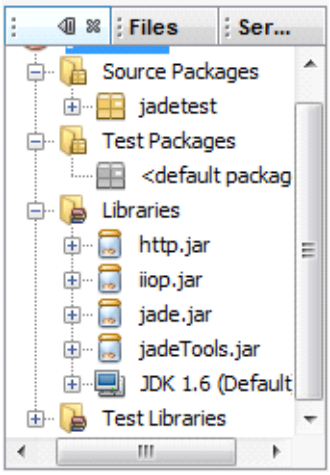


Figure V.2 : Intégrations des bibliothèques jade.

V.3. Description du système

L'interface homme/machine représente l'élément clé dans l'utilisation de tout système informatique. Les interfaces de notre système de recherche sont conçues de manière à être simples, naturelles, et de compréhension et d'utilisation faciles.

V.3.1. Interface principale

L'interface illustrée par la figure ci-dessous représente l'interface principale de notre système de recherche d'informations sémantique.

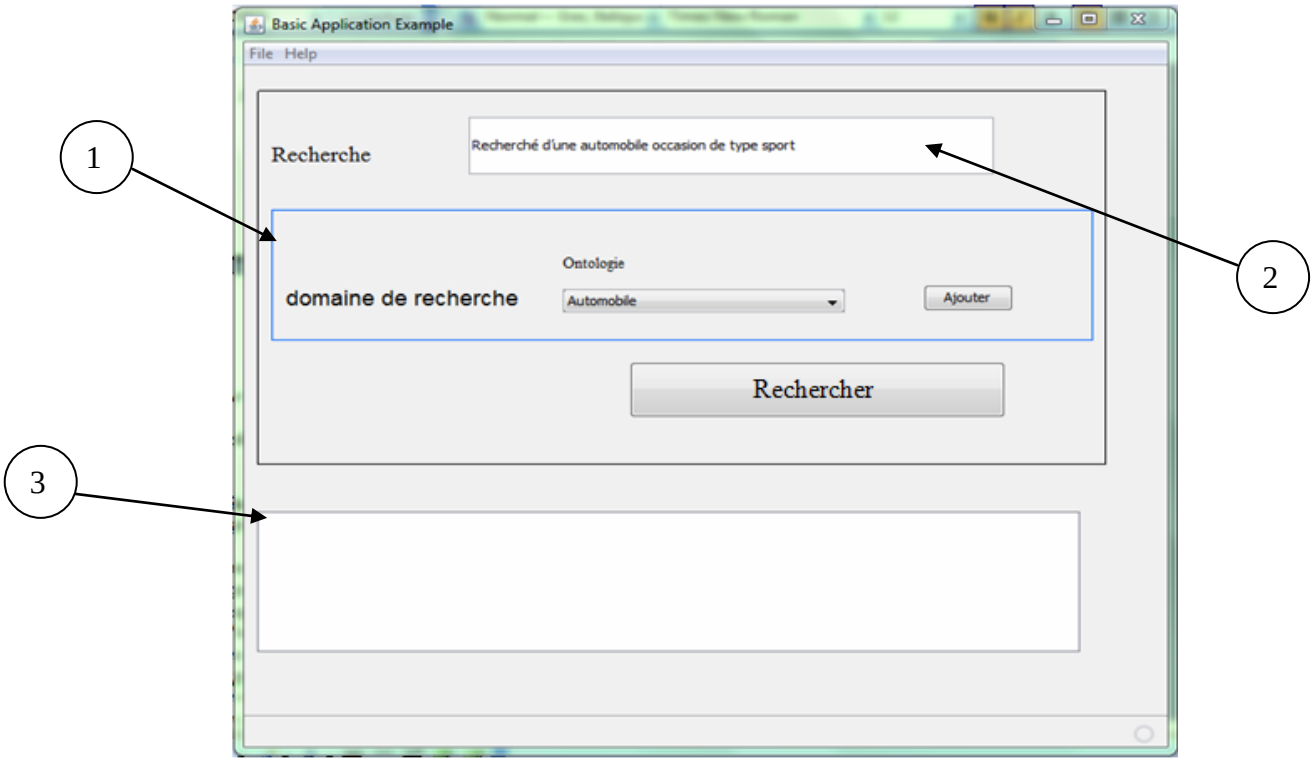


Figure V.3 Interface principale de système de recherche basé ontologie.

1) Choix du domaine de recherche : dans cette étape on sélectionne une ontologie de domaine correspondant au domaine de recherche depuis une liste fournit,

On peut ajouter des nouvelles ontologies de domaine à la liste par le bouton ajouté.

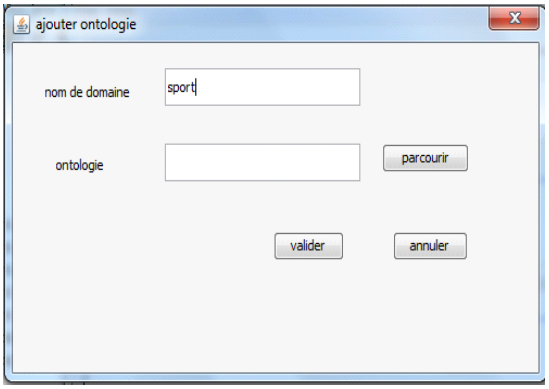


Figure V.4 : sélection une ontologie de domaine.

Dans le champ nom de domaine en doit saisir le nom de domaine comme « sport » dans notre capture, puis dans le champ ontologie en spécifiant l'adresse de l'ontologie de domaine à ajouter, on peut spécifier l'emplacement de l'ontologie par le bouton parcourir, l'ontologie sélectionner est ajouter au liste en clickant sur le bouton valider.

2) Zone de saisie des demandes utilisateur dans le champ rechercher sur le domaine sélectionné.

3) Zone d'affichage des résultats de recherche.

V.3.2. Fonctionnement

Lancer la requête de recherche par le bouton rechercher.

Dans cette étape les mots générés dans la requête doit normaliser (munisculation, éliminer les mots vides (non sensé))) par **l'agent interface** puis il envoie la requête normalisée (chaîne des mots normalisés) à l'agent requête. La classe java de normalisation est suivante :

```
package aling;

import ...


 class normalisation {


    LinkedList normalisation(String s) {
     try{
             LinkedList token=new LinkedList();
     String r=new String();
     tokinaze t=new tokinaze();
     t.tokenizes(s,token);
     for(int i=0;i<token.size();i++)
     { r=(String)token.get(i);
       r=r.toLowerCase();
       token.set(i,r);
     }
      return token;
     }catch (NullPointerException e){
             System.out.println("erreur de normalisation");
             return null ;
     }
    }

}
```

L'**agent requête** exploite une ontologie de domaine sélectionnée dans l'étape précédente de notre application pour interpréter la sémantique de la requête utilisateur, et la reformuler en ontologie requête représentée en langage OWL interprétable par les autres systèmes.

Au niveau de cet agent, l'ontologie de domaine est représentée comme un ensemble des concepts.

Chaque concept dans l'ontologie est représenté par une triplette contient trois éléments :

- sujet(class entité)
- prédicat(Calss predicat)
- objet (LinkedList)

L'ontologie est un ensemble des triplets.

L'agent requête exploite aussi le vocabulaire WordNet (ontologie de haut niveau) pour enrichir les concepts de la requête utilisateur par des synonymes .

Un algorithme d'alignement (calcul de similarité) est appliqué pour chaque concept (entité) de la requête utilisateur pour trouver des concepts équivalent sémantiquement (synonymes..) dans l'ontologie de domaine de notre système et les remplacé s'il existe .le code java de la classe qui calcule la similarité entre deux concepts est comme suit :

```
double sim_nom(String nom1,String nom2) {
 try{
    double result=1, som1=0, sim;
    LinkedList token1=new LinkedList(),token2=new LinkedList();
    normalisation nor=new normalisation();
    token1=nor.normalisation(nom1);
    token2=nor.normalisation(nom2);
    int n1=token1.size(),n2=token2.size() ;
    sim_max_d_ens s=new sim_max_d_ens();

    for(int i=0;i<n1;i++)
    {sim=s.sim((String)token1.get(i),token2);
     som1=som1+sim;
    }
    double som2=0;
    for(int j=0;j<n2;j++)
    {sim=s.sim((String)token2.get(j),token1);
     som2=som2+sim;
    }
    if(n1==0&&n2==0)
       result=1;
    else
       result=((som1+som2)/(n1+n2));
    return result;
 }catch (NullPointerException e){
         System.out.println("erreur");
         return (0);     }
 }
}
```

Après cette étape la requête utilisateur est représentée par les concepts similaires trouvés dans notre ontologie de domaine. Une nouvelle requête engendrée représente la sémantique de la requête utilisateur et exploite les concepts de notre ontologie de domaine et ses relations entre

eux , en fin cet agent crée un fichier à extension OWL représentant la sémantique de requête utilisateur interprétable par la machine en respectant le langage OWL (langage de représentation d'ontologie) .

Le fichier créer représente la sémantique de la demande utilisateur ,on l'appelle ontologie requête.

L'agent requête envoie l'ontologie requête à l'agent médiateur qui a une base de données (base d'alignement) qui lui permet de reformuler cette requête pour chaque serveur (Système à base de connaissance) par des concepts équivalents existant dans notre base d'alignement, et l'ontologie requête pour les autres serveurs non connus. Ces requêtes sont envoyées à l'agent alignement qui lance la recherche distribuée sur le web.

Jusqu'à cette étape on est arrivé à modéliser la sémantique de la demande utilisateur de manière interprétable par les systèmes à base de connaissance à fin de pouvoir d'interopérabilité sémantique entre les différents systèmes à base de connaissances hétérogènes distribuées sur le web.

Alors le contenue de la demande ou de réponse dans un message est représentée par une ontologie requête.

L'agent alignement sélectionne les résultats pertinents grâce à une métrique proposée qui compare le nombre de concepts similaires retournés au concept envoyé, et déduit des nouveaux alignements qu'ils envoi à l'agent médiateur pour faire des mises à jour sur la base d'alignement, et envoyer les liens pertinents à l'agent requête qui renvoi à l'agent interface pour les afficher à l'utilisateur final.

La communication est l'une des plus importantes opportunités offertes par la plateforme JADE. Le paradigme adopté est le passage des messages asynchrones "asynchrone message passing". Chaque agent détient une sorte de boite aux lettres "the agent queue" destinée à recevoir les messages provenant des autres agents. Tout message arrivant sur cette boite est notifié pour qu'il puisse être traité par l'agent. Les messages échangés dans JADE ont le format spécifié par le langage ACL défini par FIPA.

V.4. Étude de cas

considérons que

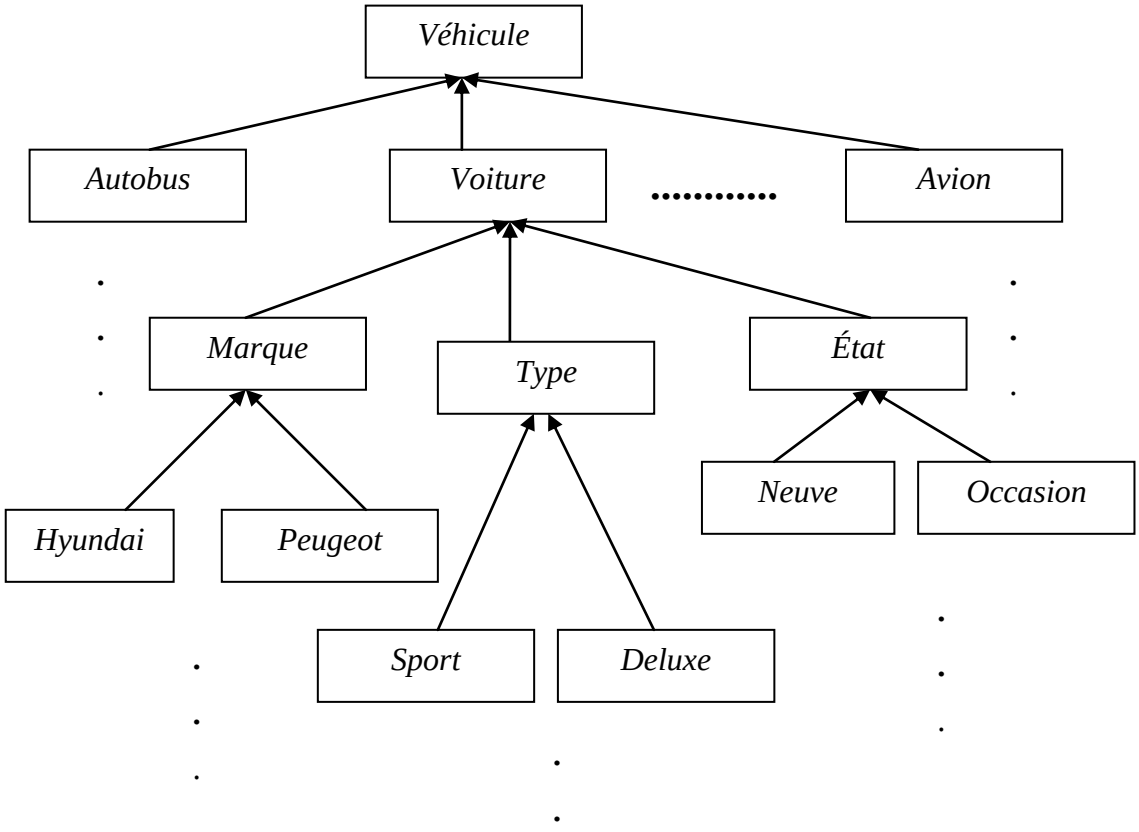


Figure V.5 : ontologie de domaine de notre système

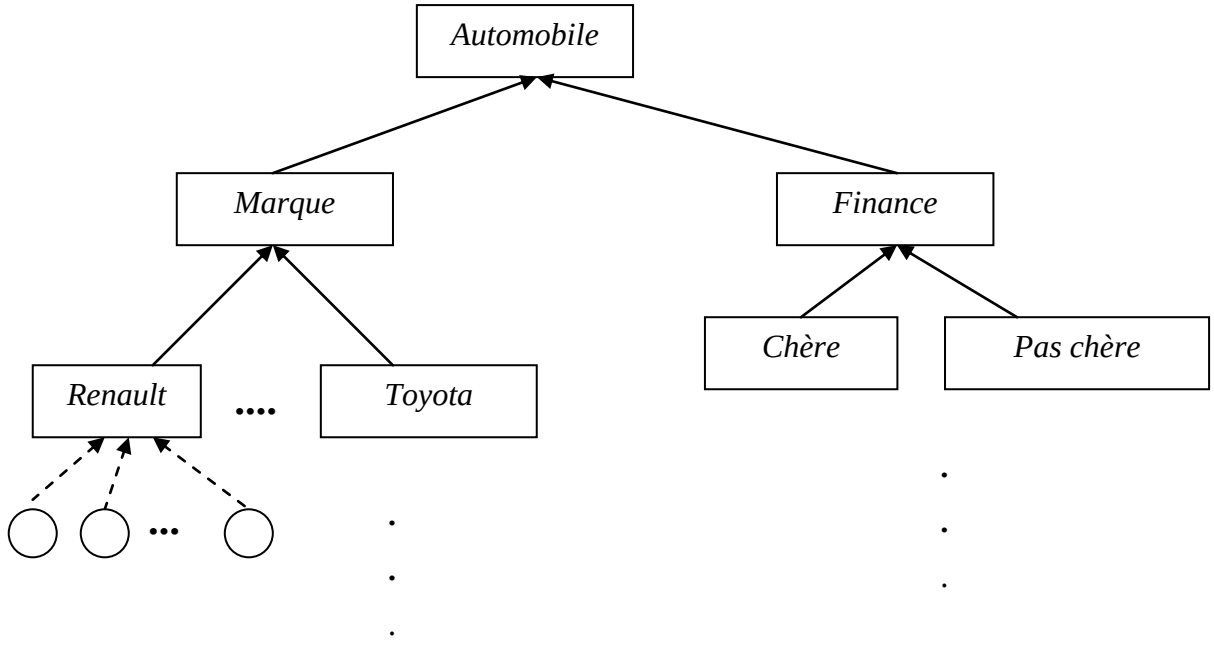


Figure V.6 : Ontologie de domaine SBC 1

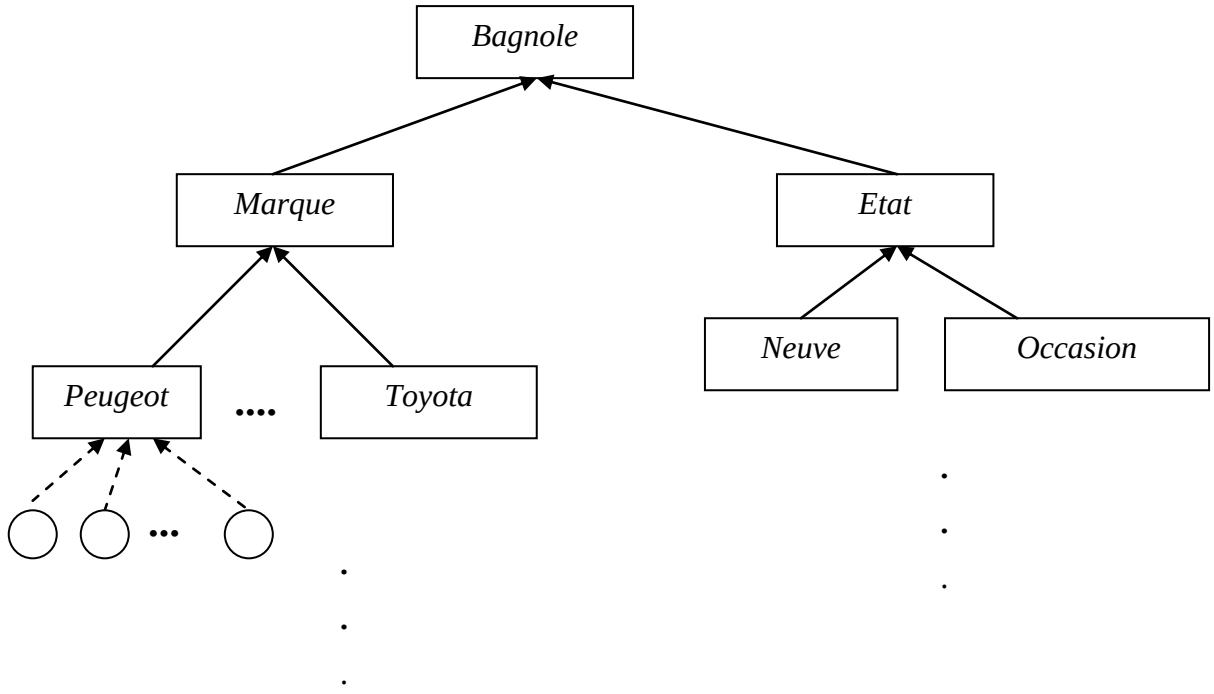


Figure V.7 : Ontologie de domaine SBC 2

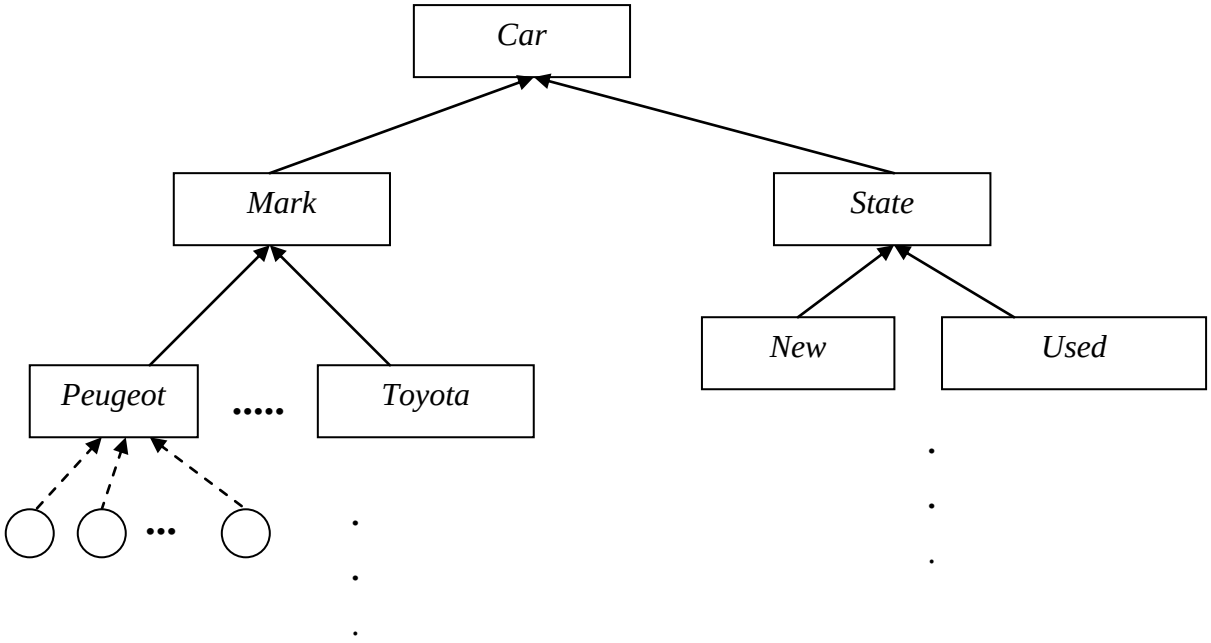
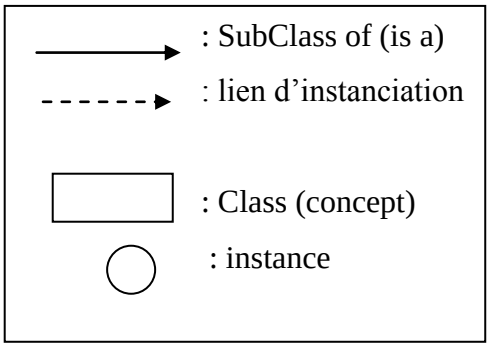


Figure V.8 : Ontologie de domaine SBC 3



En utilisant l'éditeur d'ontologie protégé 2000 pour l'implémentation les ontologies proposées

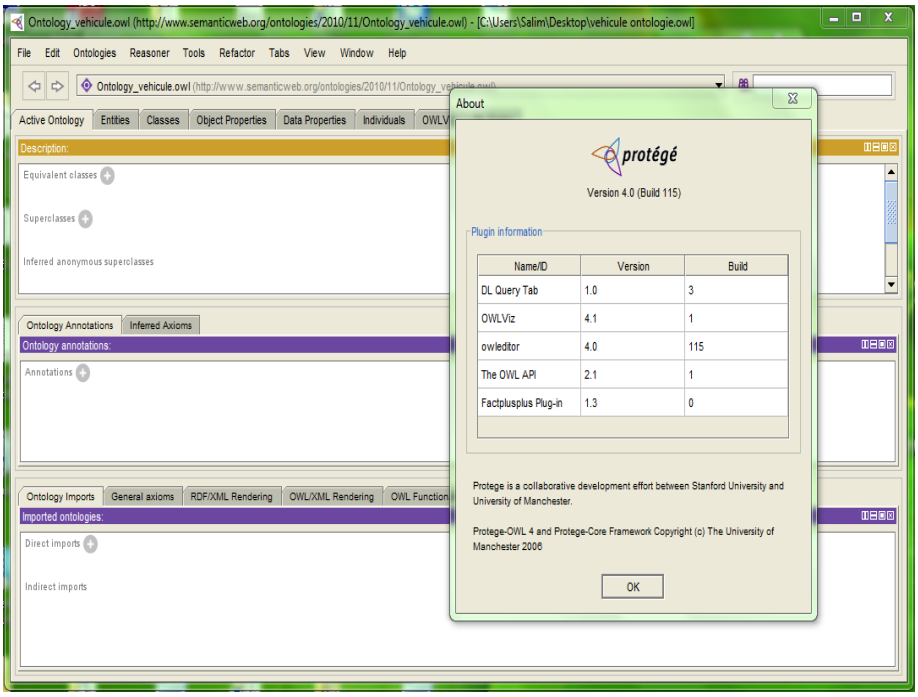


Figure 5.9 : Interface graphique de Protégé 4.0

La figure suivante illustre l'ontologie de domaine de notre système construite par l'éditeur protégé.

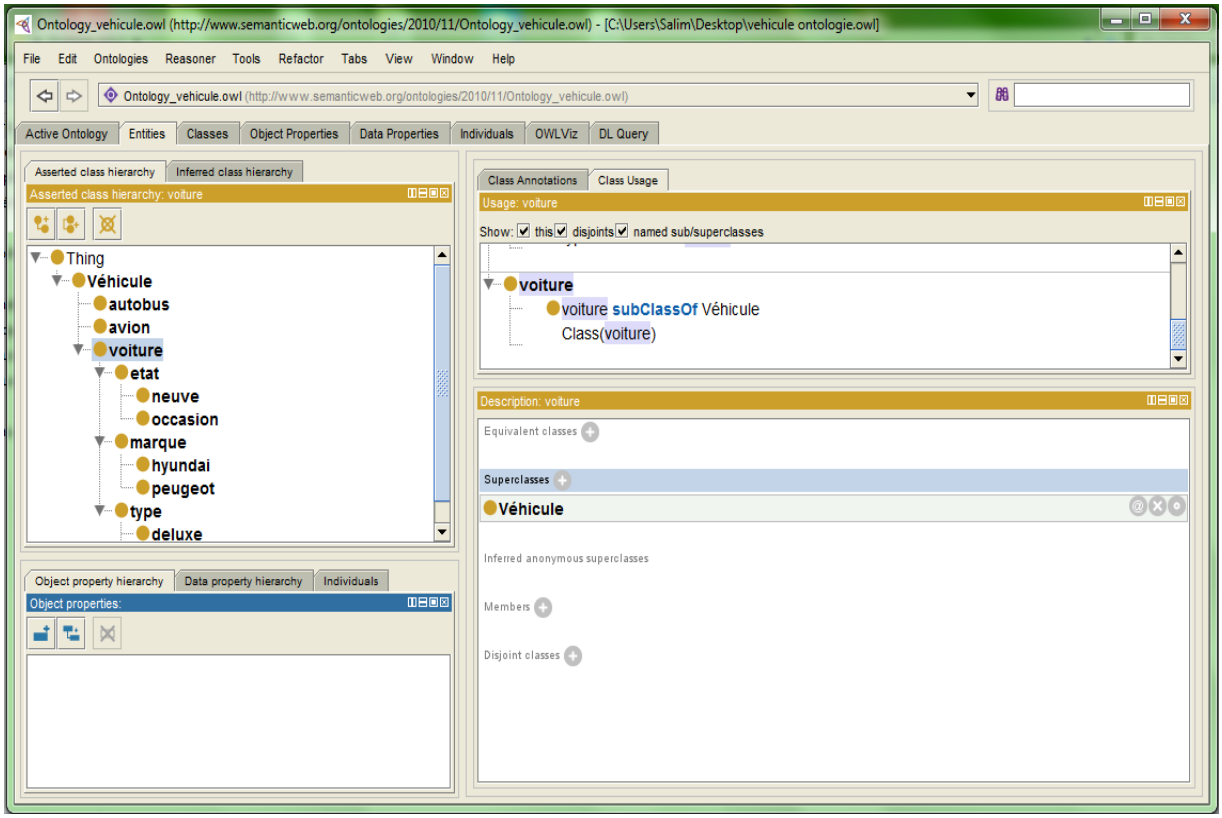


Figure V.10 : Création d'une ontologie de domaine Automobile par protégé.

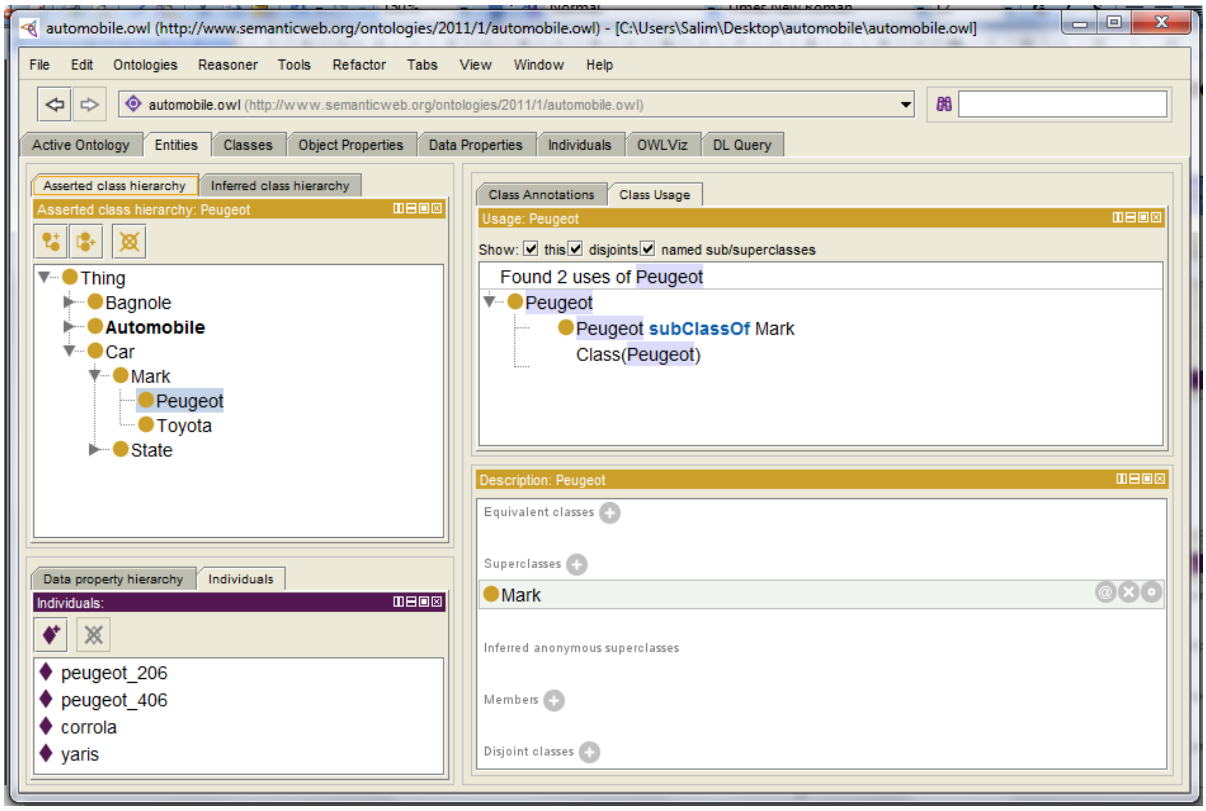


Figure V.11 : Création d'une ontologie de domaine Automobile pour SBC1.

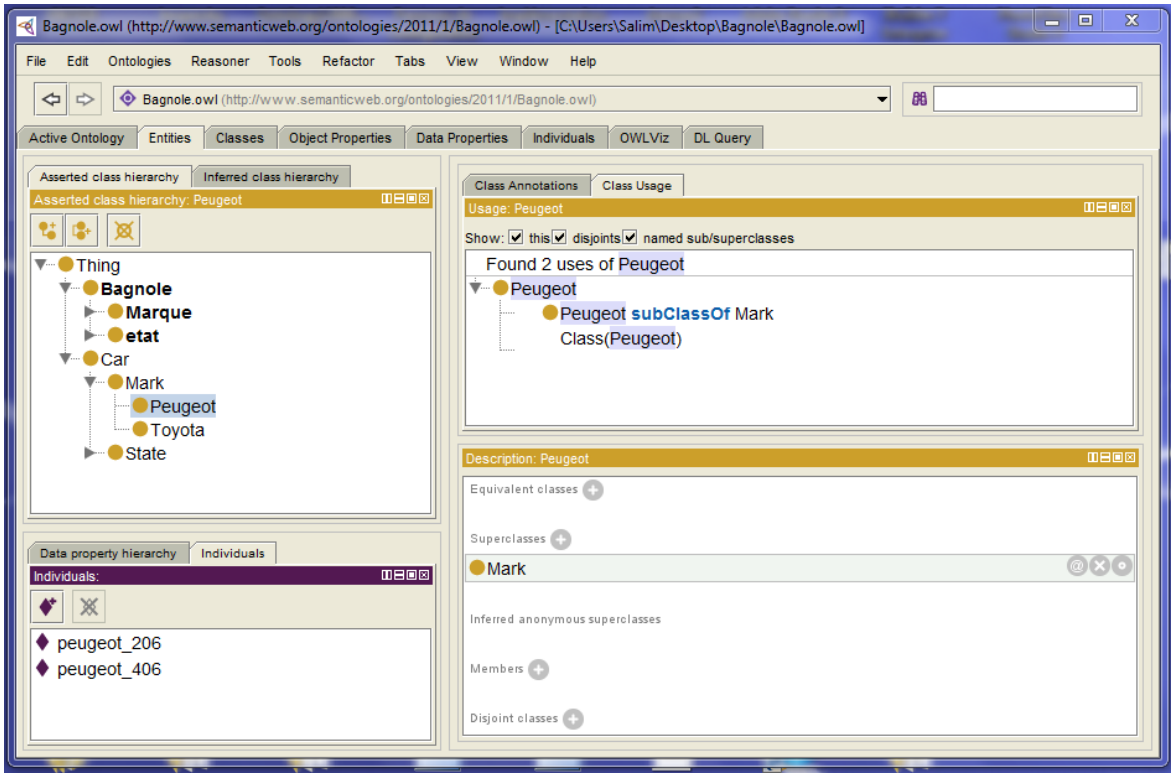


Figure V.12 : Création d'une ontologie de domaine Bagnole pour SBC2.

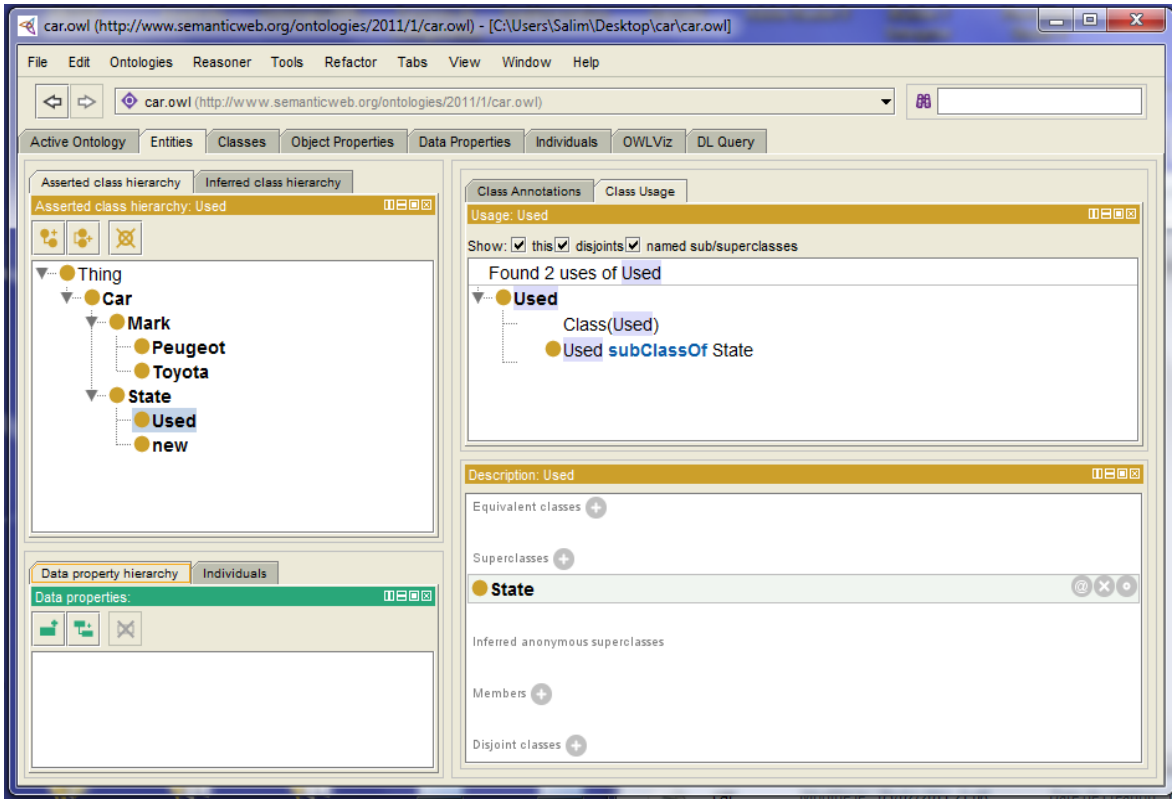


Figure V.13 : Création d'une ontologie de domaine Car pour SBC3.

Si un utilisateur lance la requête de recherche suivante :

" **Recherche d'une automobile occasion de type sport** "

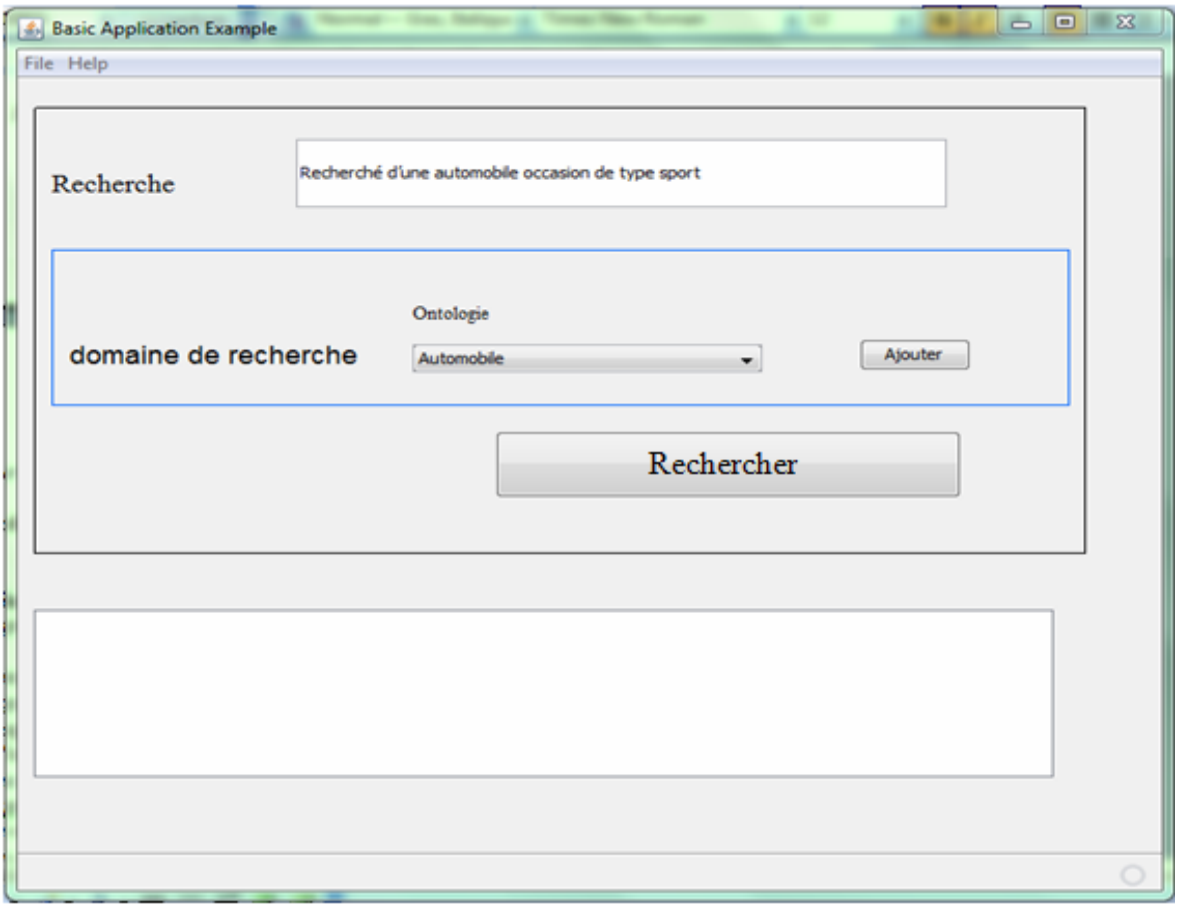


Figure V.14 Lancer une requête de recherche dans le domaine automobile.

L'agent interface :

Génère l'ensemble suivant :{chercher, automobile, type, sport, occasion} après la normalisation

Il envoie la requête à l'agent requête .

L'agent requête

Pour chaque mot ajouté des synonymes à partir d'un thesaurus

Exp.: automobile→ auto, car,voiture, engin ... en exploitant le vocabulaire WordNet pour

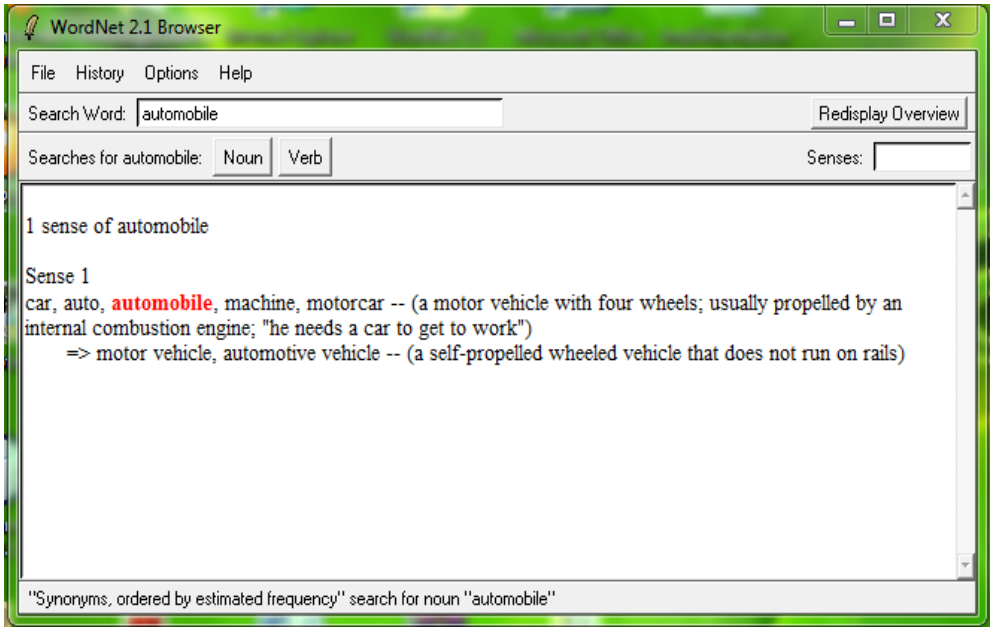


Figure V.15 L'interface graphique WordNet 2.1 .

nous construisons enfin l'ontologie requête qui suit :

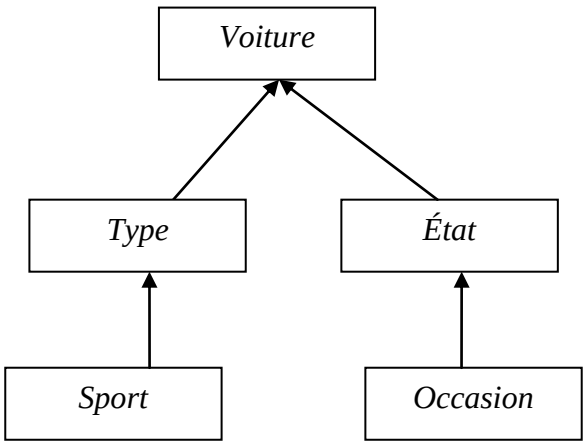


Figure V.16 : Ontologie requête

Cette ontologie est envoyée à l'agent requête

Agent médiateur

Reçoit l'ontologie requête, initialement la base de données d'alignement est vide donc la même ontologie requête qui diffusée à tous les serveurs SBCs.

Le SBC1 reçoit ontologie requête par son algorithme d'alignement et seulement le concept voiture est trouvé dans son ontologie de domaine qui est équivalent au concept **automobile** ; une réponse de la requête ontologie contient le concept automobile.

Le SBC2 reçoit ontologie requête et par son algorithme d'alignement il répond par : bagnole→état→occasion ...

Le SBC3 reçoit ontologie requête et par son algorithme d'alignement répond par : car → state → used ...

L'agent alignement reçoit ces réponses et sélectionne les réponses pertinentes (SBC 2, SBC 3) (concepts trouvé/concepts ontologie requête)=3/5 >= seuil prédéfinie(0.5).

Les réponses pertinentes sont retournées a l'agent requête qui les renvoie à l'agent interface qui les affiche à l'utilisateur.

Il sélectionne les réponses les plus pertinentes des requêtes utilisateur et il reformuler les requêtes ontologie avec les concepts appropriés pour chaque SBC exp. :

voiture →bagnole pour SBC2 et, car pour SBC3.

En outre cet agent déduit des nouveaux alignements entre des ontologies proches (les réponses pertinentes équivalentes sémantiquement pour une requête ontologie) on considère deux réponses équivalentes Si (nombre de concept équivalent trouvé commun)/nombre de concepts trouvés) >= seuil (0.5) , dans notre cas d'étude

Les réponses pertinentes sont réponse de SBC2 et réponse de SBC3 car les deux réponses retournent 3 concepts /5 concepts de l'ontologie requête =0.6 >= 0.5 . et les ontologies proches sont ces deux puisque les

(Concepts communs trouvés/concepts trouvés =3/3= 1 >=0.7).

La base d'alignement devient après cette recherche comme suit :

SBC	SBC 1	SBC 2	SBC 3
Voiture		Bagnole	Car
Etat		État	State
Occasion		Occasion	Used
Type			
Sport			

On peut ajouter dans la table des nouvelles colonnes pour les nouveaux SBC de domaine explorés dans le web et des nouvelles lignes pour les nouveaux concepts ajoutés dans l'ontologie de domaine de notre SBC.

L'agent alignement est généralement peut fournir à tous les SBCs (1,2,3) un médiateur qui fait reformuler la requête ontologies pour le partage des connaissances entre eux.
Les liens URI des concepts pertinents retournés sont affichés à l'utilisateur comme lien des réponses pertinentes dans notre étude de cas les liens retournés sont les liens des concepts occasion et used :

www.commerce/bagnol/occasion.com

www.bisness/car/used.com

V.5. Conclusion

Dans ce chapitre, nous avons présenté l'implémentation des principaux modules de notre architecture. Nous avons tout d'abord présenté l'environnement de développement ainsi que les différents outils utilisés, puis nous avons donné une description de l'application et les outils utilisés à travers des fenêtres de capture . Nous avons proposé une étude de cas comme application de notre architecture pour essayer de mettre en œuvre l'ensemble des idées qui caractérise l'architecture proposée .

Conclusion générale et perspectives

Tout au long de ce mémoire, nous avons présenté les différentes technologies nécessaires pour proposer une approche basée agent pour l'alignement d'ontologies dans le web sémantique. Nous nous sommes intéressés à la résoudre le problème d'hétérogénéité d'ontologies de domaine.

Dans le noyau du Web sémantique, les ontologies permettent de capitaliser, de représenter, d'exploiter et de partager sémantiquement des connaissances et des informations. En tant qu'entités du Web, un environnement très large, très complexe et divers, les entités du Web sémantique possèdent aussi des caractéristiques de cet environnement si hétérogène. Les ontologies du Web sémantique peuvent donc être variées et hétérogènes même si elles sont créées dans un même domaine.
Une des clés importantes pour le succès du Web sémantique est de maîtriser cette hétérogénéité et de cohabiter avec elle.

Pour atteindre l'objectif de ce mémoire qui est de permettre de l'exploitation d'un Web sémantique dans une organisation hétérogène, nous envisageons la problématique de l'hétérogénéité de l'ontologie, qui est une composante de base importante du Web sémantique.

La littérature du domaine propose plusieurs méthodes d'alignement d'ontologies, ces méthodes exploitent différents formats d'ontologies et repose sur le calcul des mesures de similarité, ces méthodes sont souvent exploitées dans ces environnements de développement locaux.

Notre architecture est consacrée à fournir un environnement pour sélectionner les ontologies de domaine proches dans web sémantique et déduire des alignements en exploitant les réponses sémantiques reçues et faire des mises à jour à la base d'alignement de notre système.

La réalisation de cette étude nous à conduits à suivre les étapes suivantes :

1. En premier lieu, nous avons fait un état de l'art sur les ontologies.

2. Nous avons effectué une étude générale sur l'alignement d'ontologies.

3. Après, nous avons discuté quelques travaux et projets d'alignement d'ontologies et celle qui exploite le paradigme agent logiciel.

4. Ensuite, nous avons présenté notre approche base agent pour l'alignement d'ontologies : « Équivalence sémantique" dans un système réparti et hétérogène, où l'on donne un intérêt de notre architecture et le rôle, les fonctions et les différentes interactions entre les agents.

5. Finalement, dans le but de mieux comprendre le fonctionnement de notre approche, nous avons choisi comme cas à étudier la recherche d'informations basée sur l'ontologie dans le web. De plus, nous avons implémenté les aspects les plus importants de notre architecture.

Ce mémoire constitue une base de travail à partir du quelle de nouvelles activités de recherche peuvent être lancées afin d'améliorer le travail présenté.

Perspectives

Les perspectives que nous proposons peuvent donc s'orienter vers les directions suivantes :

- Faire une validation de notre approche dans le cadre d'une application concrète. Par exemple, dans le domaine médical.
- La réalisation d'un système complet permettant de gérer des alignements ontologies par la modification, la suppression, l'ajout des nouveaux résultats.
- Prendre en compte l'aspect des agents mobiles et l'adaptabilité des agents.

Référence :

[1] **L.Ghomari** « alignement d'ontologies de domaines : « étude comparative syntaxique versus sémantique cas d'application : COMA++ VS. OWL-CTXMatch » . Mémoire de Magister Ecole nationale Supérieure d'Informatique (E.S.I) Alger, 2008-2009 .

[2] **L.Mazuel et J.Charlet** « Alignement entre des ontologies de domaine et la Snomed: trois études de cas » Article IC 2009.

[3] **J.Euzenat** « Dynamique et sémantique: l'alignement d'ontologies dans le web sémantique est bien plus difficile que vous n'osiez même le penser » INRIA Grenoble, France 19 octobre 2008.

[4] « État de l'art Ontologies et Intégration/Fusion d'ontologies » rapport de centre de Recherche et Développement de France Télécom (FTR&D) à Lannion du 15 avril au 13 septembre 2002.

[5] J.Charlet, B.Bachimont et R.Troncy « Revue Ontologie pour web sémantique » Mission de recherche STIM, AP-HP & INSERM ERM 0202 , 2004.

[6] **A.Valentina et M.Tamma** « An Ontology Model supporting Multiple Ontologies for Knowledge sharing » **thèse** octobre 2001.

[7] **L.Bougchiche** « vers une ontologie pour le dispositif d'interaction ». mémoire magister Ecole Nationale Supérieure d'Informatique E.S.I Oued-Smar, Alger *Ecole Doctorale STIC 2007*.

[8] **Benjamin et Perez** « **o**ntology and problem solving methods » Knowledge system technology 1999.

[9] **P. Bouillon, Fr. Vandooren, L. Da Sylva, L. Jacqmin, S. Lehmann, G.Russell et E.Viegas**, « *Traitement automatique des langues naturelles* » Duculot 1998.

[10] **Jasper et Uschold** « *Framework for understanding and classifying ontology » application* 1999

[11] http://nkos.slis.kent.edu/KOS_taxonomy.htm

[12] **N.F.Noy et C.Hafner** « The State of the Art in Ontology Design: A Survey and Comparative» Review In AI Magazine, 18(3), 53-74 (1997). http://www.aaai.org/Library/ Magazine/Vol18/18-03/vol18-03.html.

[13] **T.GRUBER** « A translation approach to portable ontology specifications » *Knowledge Acquisition* 5(2),pages 199-220, 1993.

[14] **M. Uschold et M. Grüninger**, « ONTOLOGIES: Principles, Methods and applications» Knowledge engineering review, vol.11, N.2, 1996.

[15] **N. Guarino** « Formal ontologies and information systems » In N. Guarino, editor, *Proceedings of FOIS'98*, IOS Press, Amsterdam, 1998.

[16] **A.GÓMEZ-PÉREZ., M.FERNANDEZ-LOPEZ et O.CORCHO** « Ontological Engineering ». Advanced Information and Knowledge Processing. Madrid, Spain : Springer, 2 edition. 2004.

[17] **F.FÜRST**, « Opérationalisation des ontologies : une méthode et un outil ». In Ingénierie des Connaissances (IC), p. 199–210, Lyon, France. 2004.

[18] **N.GUARINO, M.CARRARA et P.GIARETTA** « Formalizing ontological commitments». In National Conference of the American Association on Artificial Intelligence (AAAI), p. 560–567. 1994.

[19] **G. van Heijst, A. Th. Schreiber et B. J. Wielinga**. « *Using explicit ontologies for KBS development*» *International Journal of Human-Computer Studies* 1997.

[20] **O. Lassila et D. McGuinness**, « The role of frame-based representation on the semantic web ». Electronic Transactions on Artificial Intelligence (ETAI) Journal: area The Semantic Web, volume To appear, 2001.

[21] **H.Sofia Pinto, J.P.Martins**, « *Reusing Ontologies*, aifbhermes.aifb.unikarlsruhe. de/AAAI2000/ CameraReady/HPinto00.pdf ».

[22] B. BACHIMONT, « Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances ». In Ingénierie des connaissances: évolutions récentes et nouveaux défis, pages 305–323. Eyrolles, 2000.

[23] **J. Sowa**. « *Conceptual structures: information processing in mind and machine* ». Addison-Wesley, 1984.

[24] **T.Berners-Lee, J.Hendler, et O.Lassila**. The semantic web. Scientific American, 284(5) :34–43, 2001.

[25] Recommendation **W3C** XML, 2004. http://www.w3.org/TR/2004/REC-xml-20040204/.

[26] Recommendation **W3C** XML Schema, 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[27] Recommendation **W3C** RDF, 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[28] Recommendation **W3C** RDF Schema, 2004. http://www.w3.org/TR/rdf-schema/.

[29] Recommendation **W3C** OWL, 2004. http://www.w3.org/TR/owl-ref/.

[30] **F.Baader, D.Calvanese, D.McGuinness, D.Nardi, et P.PatelSchneider** « The Description Logic Handbook : Theory, Implementation and Applications». Cambridge University Press, 2003.

[31] **M.Minsky**. « A framework for representing knowledge». In D. Metzing, editor, Frame Conceptions and Text Understanding, pages 99–125. de Gruyter, Berlin, 1980.

[32] **P.Fournier-Viger** « Un modèle de représentation des connaissances `a trois niveaux de sémantique pour les systèmes tutoriels intelligents ». mémoire de magister , Université de Sherbrooke, Sherbrooke, Canada., 2005.

[33] **U.Sattler et C.Lutz**. « Description logics », esslli, 2002.

[34] **V. Haarslev et R. Moller**. « Racer user's guide and reference manual version 1.6 » . Technical report, University of Hamburg, Computer Science Department,2001.

[35] **E.Sirin, B.Parsia, B.Grau, A.Kalyanpur, et Y.Pellet** : « A practical owl-dl reasoner ». Submitted for publication to Journal of Web Semantics, 2006.

[36] A.Valentina M. Tamma, « An Ontology Model supporting Multiple Ontologies for Knowledge sharing », thèse octobre 2001.

[37] **M.Blazquez, M.Fernandez, J. M.Garcia-Pinar et A.Gomez-Perez** « Building Ontologies at the Knowledge Level using the Ontology Design environment ». in Proceedings of the Banff Workshop on Knwoledge Acquisition for Knwoledge-based Systems, 1998.

[38] **R.Dieng, O.Corby, F.Gandon, A.Giboin, J.Goleblowska, N.Atta et M.Ribiere** « Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge management » (2ième édition), Dunod Edition Informatiques Séries Systèmes d'Information, 2001.

[39] **M.Fernandez, A.Gomez-Perez et N.Juristo**, « METHONTOLOGY : from ontological art towards ontological engineering », in Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97), AAAI Press , 1997.

[40] **F.Gandon**, « Ontology Engeneering : a survey and a return on experience », rapport de recherche 4396, INRIA, 2002.

[41] **M.GRUNINGER et M. S.FOX**, « Methodology for the design and evaluation of ontologies » , in Proceedings of the Workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI'95, 1995.

[42] **A.Gomez-Perez, M.Fernandez et A. J. DEVICENTE**, « Towards a Method to Conceptualize Domain Ontologies » , in *Proceedings of the European Conference on Artificial Intelligence, ECAI'96*, pages 41-52, 1996.

[43] **J.Bouaud, B.Bachimont, J.Charlet et P.Zweigenbaum,** « Acquisition and Structuring of an Ontology within Conceptual Graphs » , in *Proceedings of the International Conference on Conceptual Structures,ICCS'94*, Springer LNCS 835, pages 1-25, 1994.

[44] **C.Welty et N.Guarino** « Supporting ontological analysis of taxonomic relationships » , *Data et Knowledge Engineering* (39), pages 51-74, 2001.

[45] **J.Ferber**. « Les systèmes multi-agents. Vers une intelligence collective », Editions InterEditions, 1995.

[46] **A.Viollet** « Un protocole entre agents pour l'alignement d'ontologies » mémoire de Master 2 "Intelligence, Interaction, Information" – UJF 2003 / 2004 université de Joseph Fourier INRIA france .

[47] **B. Chaib-draa, I. Jarras et B. Moulin** *« Systèmes multiagents : Principes généraux et applications »* *Article à paraître dans : J. P. Briot et Y. Demazeau « Agent et systèmes multiagents » chez Hermès en 2001.*

[48] J.Euzenat M.Ehrig, A.Jentzsch, M.Mochol et P.Shvaiko, « *Heterogeneity in the semantic web* », *deliverable 2.2,* Knowledge Web, December 2007.

[49] J.Euzenat, et P.Shvaiko, *« Ontology matching* » , Springer, Heidelberg (DE), 2007.

[50] I.Zaihrayeu, « Towards Peer-to-Peer Information Management Systems », thèse de PHD, International Doctorate School in Information and Communication Technology, Université de Trento, Italy, March 2006.

[51] D.Fensel , H.Lausen, A.Polleres, J.DeBruijn, M.Stollberg, D.Roman, et J.Domingue.« Enabling semantic web services: the web service modeling ontology », Springer, Heidelberg (DE), 2007.

[52] M.Klein, « *Combining and relating ontologies: an analysis of problemsand solutions* », in Proc. IJC AI Workshop on Ontologies and InformationSharing, Seattle (WA US), 2001.

[53] I.Dagan, , L.Lee, et F.Pereira, « Similarity-based models of word cooccurrence probabilities ». Machine Learning 34(1-3).

[54] I.P.Fellegi, et A.B.Sunter, « A theory for record linkage ». Journal of the American Statistical Society 64:1183–1210.

[55] W.Cohen, , P.Ravikumar, , et S.Fienberg, « A comparison of string metrics for matching names and records ». Dans Proc. KDD-2003 Workshop on Data Cleaning and Object Consolidation, 2003.

[56] D.G.Maynard, et S.Ananiadou « Term extraction using a similarity-based approach. In Recent Advances in Computational Terminology ». John Benjamins, 1999.

[57] **A.Doan, , J.Madhavan, , P.Domingos, , et A.Halevy** « Learning to Map between Ontologies on the Semantic Web » . le 11eme International World Wide Web Conference (WWW'2002), Hawaii, USA..

[58] **F.Giunchiglia, , P.Shvaiko, et M.Yatskevich**, « S-Match: an algorithm and an implementation of semantic matching » . Dans Proceedings of ESWS 2004, Heraklion (GR), pages 61–75, 2004.

[59] **F. Wiesman, N. Roos et P. Vogt**. « Automatic ontology mapping for agent communication ». MERIT-Infonomics Research Memorandum series, 2001.

[60] T.L.Bach , « *Construction d'un Web sémantique multi-points de vue* », Thèse de doctorat Informatique, Ecole des Mines de Paris a Sophia Antipolis, 23 octobre 2006.

[61] *T.Cox, et M.Cox*, « *Multidimensional Scaling* ». *Chapman and Hall, 1994.*

[62] **J.Charlet, B.Bachimont, R.Troncy** « Ontologies pour leWeb Sémantique », *Revue I3, numéro Hors Série «Web sémantique»*p. 43-63, 2004.

[63] **A.Hameed, A.Preece and D.Sleeman** « *Ontology reconciliation* », in Steffen Staab and Rudi Studer, editors, Handbook on ontologies, chapter12, pages 231–250.Springer Verlag, Berlin (DE), 2004.

[64] **J.Euzenat, and P.Shvaiko**, "*Ontology matching* », Springer, Heidelberg (DE), 2007.

[65] **J.Euzenat, T.L. Bach., J.Barrasa, P.Bouquet, J.D.Bo, R.Dieng-kuntz, M.Ehrig, M.Hauswirth, M.Jarrar, R.Lara, D.Maynard, A.Napoli, G.Stamou, H.Stuckenschmidt, P.Shvaiko, S.Tessaris, S.V.Acker et Zaihrayeu**, « State of the art on ontology alignment », deliverable 2.2.3", IST Knowledge web NoE, 80p., June 2004.

[66] **F.Giunchiglia, P.Shvaiko et M.Yatskevich,** « Discovering missing background knowledge in ontology matching » , in Proc. 16th European Conference on Artificial Intelligence (ECAI), pages 382–386, Riva del Garda (IT), 2006.

[67] **F.Giunchiglia et P.Shvaiko**, « *Semantic matching* », The Knowledge Engineering Review, 18(3):265–280, 2003.

[68] **J.Euzenat and P.Valtchev**, « Similarity-based ontology alignment in OWLlite », in Proceedingq 15th ECAI, Valencia (ES), 2004.

[69] **L. Steels**. « Emergent adaptive lexicons ». In P. Maes, editor, From Animals to Animats 4: Proceedings of the Fourth International Conference On Simulating Behavior, Cambridge Ma., 1996. The MIT Press.

[70] **JADE** (Java Agent DEvelopement framework) : http://jade.tilab.com.

[71] **FIPA**, IEEE Foundation for Intelligent Physical Agents, http://www.fipa.org/ ,2000.

[72] **NetBeans** IDE : www.netbeans.org/.

[73] **Sun** Microsystems : http://fr.sun.com/.

[74] A.G.Miller, « Wordnet: A lexical database for English ». Communications of the ACM, 38(11):39–41, 1995.

[75] **S. Zghal, K.Kamoun, S.Ben_Yahia, E.M.Nguifo, Y.Slimani** « EDOLA : Une nouvelle méthode d'alignement d'ontologies OWL-Lite » *CRIL CNRS FRE 2499, Université d'Artois, IUT de Lens Rue de l'Université - S.P. 16, 62307 Lens Cedex, France*