الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire de Mila

Institut des sciences et de la technologie — Département de Mathématiques et Informatique

Mémoire préparé En vue de l'obtention du diplôme de Master
En:
Filière informatique générale

Spécialité sciences et technologies de l'information et de la communication STIC

Développement d'une application pour la découverte des services web à base de la distance sémantique

Préparé par :
-Slimani Soumia
-Laieb karima

Soutenue devant le jury..

Président : Boukhcheme Nadhir	M.A.A	C.U.Mila
Examinateur :Hadji Othman	M.A.B	C.U.Mila
Promoteur : Merabet Adil	M.A.B	C.U.Mila

Année universitaire :2013/2014

Remerciment

Après avoir terminé ce mémoire de fin d'étude, nous Réservons ces lignes
Pour exprimer nos remerciements les plus sincères à notre dieu
Tout puissant de nous avoir donné la santé et la patiente pour terminer
Ce travail Nous remercions tout d'abord
ALLAH

Un grand remerciement particulier à notre encadreur Mr Mérabet Adil
pour leur précieuse conseils, leur disponibilité, la confiance
qu'elle nous a toujours témoignée et la sollicitude
dont elle nous a entouré, et ce tout au long
de l'élaboration du présent travail.
Un grand remerciement particulier Mouhamed Amin Kheniwi.
Nos remerciements à tout le personnel de l'institut de l'informatique surtout
les enseignants qui nous ont enseigné durant toutes nos années d'étude.
Enfin nous remercions toutes personnes qui ont contribué de
prés ou loin à l'achèvement de ce travail.
Merci bien.

Dédicaces

Je dédie ce modeste travail aux êtres les plus

chers dans ma vie,

À mes chers parents, ma mère Rahima et mon père Mostefa pour leur amour, leurs encouragements et leurs sacrifices.

À mon cher frère Riyad, Farid, Houssam, Bader Eddine, et Youssef

À mes très chères sœurs Saliha, Siham, Amina.

À mon oncle nouer Eddine ainsi que toute sa famille

À ma famille, mes cousins et cousines

À mes très chères amies Imane, Amel, Ahlam.

À mes amies Radouane, Amine, Nassim, Djamel, mohammed,

Fateh, Farouk.

À Tous Mes Collègues d'étude.

Karima

Dédicaces

Je dédie ce modeste travail aux êtres les plus

chers dans ma vie,

À mes chers parents, ma mère Nadira et mon père Abd Elhamid pour leur amour, leurs encouragements et leurs sacrifices.

À mon cher frère Mouad et Ayoub

À mes très chères sœurs Bouchra.

À ma famille, mes cousins et cousines

À mes amies

À Tous Mes Collègues d'étude.

Soumia

Table des matières

Chapitre 02 : ontologie : définition, concept

Chapitre 03 :état de l'art sur les distances sémantique

Chapitre 04 : Modélisation conceptuelle

Table des Figure

Résumé

La plus part des systèmes de recherche d'information existants sont simplement des recherches textuelles qui ne sont pas capables de comprendre les besoins en information de l'utilisateur pour lui retourner les *services web* répondants à ses besoins.

L'apparition du paradigme du *Web Sémantique*, visant à augmenter l'intelligence de ces systèmes pour permettre de diriger ses recherches sur une base des connaissances associées à l'information demandée et aux données du Web.
Dans ce mémoire, nous proposons une modélisation à base d'ontologies d'un système de *découverte sémantique* d'information. Ce travail s'inscrit dans le cadre général d'ajouter une couche *sémantique*, visant à donner un sens aux résultats retournés selon un besoin en information d'un utilisateur lors d'une recherche sur les *services web*.
Notre travail s'articule sur les parties suivantes : une première partie est consacrée à l'étude des services web, le fonctionnement, les techniques et les différents composants. La seconde partie se base sur les *ontologies* pour ajouter la dimension sémantique au processus de *découverte*. Dans la troisième partie nous détaillons les mesures de similarité sémantique et passons en revue les différents paradigmes utilisés pour calculer les mesures de similarité *sémantique* à partir d'une *ontologie*, où ils utilisent également des mesures de *distance sémantique* pour vérifier le degré de correspondance entre la requête et les services web. Dans la quatrième et la cinquième partie nous détaillons la conception et l'implémentation de notre *système de découverte sémantique*.

Introduction générale

Le domaine de l'utilisation du Web a évolué à partir d'un répertoire pour stocker les textes et les images à l'intégration des processus d'affaires en donnant naissance à des services Web.

Les architectures orientées services et les systèmes à base de composants seront sans doute les techniques les plus utilisées dans le développement de futurs systèmes. La réutilisation de composants est une condition essentielle pour le processus de développement de logiciels à base de composants et orientés services.

La notion de *service Web* a reçu beaucoup d'attention ces dernières années comme un véhicule prometteur pour l'intégration transversale de l'organisation des applications basées sur le web. Mais la capacité de trouver facilement des services utiles (logiciels, composants logiciels, calculs scientifiques, …) devient de plus en plus critique dans plusieurs domaines.

En outre, le nombre croissant des*services Web* publiés par les différentes entreprises a rendu difficile à les gérer dans des environnements ouverts comme le Web. Le problème principal se pose lorsqu'on cherche un service entre des centaines de milliers de différents services publiés.

Un service est basé sur trois briques principales que sont SOAP (Simple Object Access Protocol), WSDL (Web Service Description Langage) et UDDI (Universal Description, Discovery and Integration). Il est considéré dans notre système comme une boîte noire qui réalise un ensemble d'opérations. Cette dernière est caractérisée par un ensemble d'entrés et un ensemble de sorites.

Donc, le but dans ce projet est de concevoir une application permet aux clients de chercher des *services web*. Afin de rendre le processus de découverte efficace et extensible nous allons utiliser essentiellement la mise en correspondance de leurs entrées-sorties avec la requête demandée par le client. Une éventuelle intervention d'un ensemble *d'ontologies* peut être envisagée afin d'améliorer les paramètres d'une certaine connaissance.

Le processus de *découverte* dans notre application est basé sur la *distance sémantique* entre les différents concepts. Nous allons utiliser la formule de Tamer A. Farrag, Ahmed I. Saleh, H.A. Ali [18] pour calculer cette distance.

Le mémoire est organisé en cinq chapitres :

Le premier chapitre est dédié à l'étude de la structure générale de l'architecture SOA, et les *services web*. Le fonctionnement, les techniques et les différents composants sont abordés dans ce chapitre.

Le deuxième chapitre s'intéresse aux *ontologies* qui seront utilisées dans le processus de la *découverte*.

Le troisième chapitre est consacré à un état de l'art sur les *distances sémantiques*. Plusieurs mesures de similarité sémantiques ont été proposées dans la littérature, mais la plupart ne respectent pas les qualités attendues d'une distance sur notre domaine.

Dans le quatrième et le cinquième chapitre, nous présentons la conception de notre application, en détaillant les différentes étapes utilisées, depuis la *distance sémantique*, les différents parser (analyseurs de fichiers), l'ensemble des services et *ontologies* utilisés, jusqu'à l'exécution en fournissant des résultats à un ensemble requêtes.

Enfin, en conclusion, nous effectuons une synthèse de ce travail et son apport sur la relation client-entreprise et nous terminerons par la proposition de quelques perspectives dont la réalisation permettra l'amélioration et l'aboutissement de ce travail.

Chapitre 1

Service Web et Web Sémantique : Définition, Concept et Technique

Introduction

Les méthodes dites Web services proposent une méthodologie et un éventail de protocoles standardisés et ouverts permettant à des composants logiciels distribués d'inter opérer via Internet. Les Web services, apparus avec la mouvance Internet et le développement rapide des applications client/serveur accessibles via un navigateur Web.

Les Web services sont des applications qui relient des programmes, des objets, des bases de données ou des processus d'affaires à l'aide de XML et de protocoles Internet standards.

Les Web services sont des compléments aux programmes et applications existantes, développées aux langages tel que Visuel basic, C, C++, C #, Java ou autres, et servent de ponts que ces programmes communiquent entre eux.

De ce fait, les Web services permettent aux entreprises et aux individus de publier des liens vers leurs applications de la même manière qu'ils publient des liens vers leurs pages Web. Conséquemment, les Web services peuvent faire en sorte que toutes les ressources informatiques dont une entreprise à besoin soient des ressources distribuées à la grandeur de l'Internet. Contrairement à la plupart des applications de type client/serveur, les Web services ne fournissent pas d'interface usager. Ils sont utilisés afin d'envoyer des données destinées à être lues par des machines. Cependant, les programmeurs peuvent tout de même développer une interface graphique pour l'utilisateur [1].

I. Partie 01 : Service Web (définition, concept et technique)

I.1. Définition des services web

Dans le cadre de la programmation orientée service, un service peut être défini comme une entité fonctionnelle auto-contenue, auto-décrite, indépendante des plateformes, et pouvant être décrite, publiée, *découverte*, invoquée, composée à l'aide de protocoles standards. La technologie des services Web constitue à une approche pour la concrétisation du paradigme de service sur le Web. Par le biais de cette technologie, il devient possible de délivrer et de consommer des services logiciels sur le Web.

Le W3C (World Wide Web Consortium) définit le *service web* de la manière suivante : Un service Web est un système logiciel identifié par un identificateur uniforme de ressources URI, dont les interfaces publiques et les liens sont définis et décrits en XML. Un *service web* peut être *découvert* et sélectionné dynamiquement par d'autres systèmes logiciels. Ces derniers, peuvent ensuite interagir avec le service sélectionné en utilisant des messages XML transportés par des protocoles Internet.

Les *services web* se basent sur un modèle de trois couches :

Un protocole de communication permettant de structurer les messages échangés entre les composants logiciels. Une spécification de description des interfaces des services et enfin une spécification de publication et de localisation de service.

Les *services web* sont le fondement de base des architectures orientées services [7].

I.2. Fonctionnement des Web services

Dans sa première génération, le fonctionnement des Web Services repose sur trois couches fondamentales présentées comme suit:

- **Invocation** : visant à établir la communication entre le client et le fournisseur en décrivant la structure des messages échangés.
- **Découverte :** permettant de localiser un Web service particulier dans un annuaire de services décrivant les fournisseurs ainsi les services fournis.
- **Description:** dont l'objectif est la description des interfaces des Web services (paramètres des fonctions, types de données) [4].

I.3. L'Architecture Orientée Service (SOA)

Actuellement, sous le vocable de SOA, se développe un style d'architecture orientée service permettant de construire des systèmes informatiques évolutifs et adaptables, en améliorant leur qualité et en simplifiant leur intégration dans l'infrastructure informatique de l'entreprise, par recours à des composants réutilisables appelés services.

Dans la suite, nous présentons les concepts sur lesquels repose l'architecture SOA [2].

I.3.1. Service

Un service est un programme autonome, réutilisable, indépendant des langages de programmation et qui peut s'exécuter sur n'importe quelle plateforme.

Cette composante clef du SOA expose un ensemble d'opérations (fonctionnalités) mises à disposition par un fournisseur à l'attention d'un client selon un ou des contrats prédéfinis.

L'interaction entre le client et le fournisseur est faite par le biais d'un bus de services (médiateur) qui peut être dédié ou être entièrement prise en charge par internet. Dans ce dernier cas, il s'agit des *services web* [2].

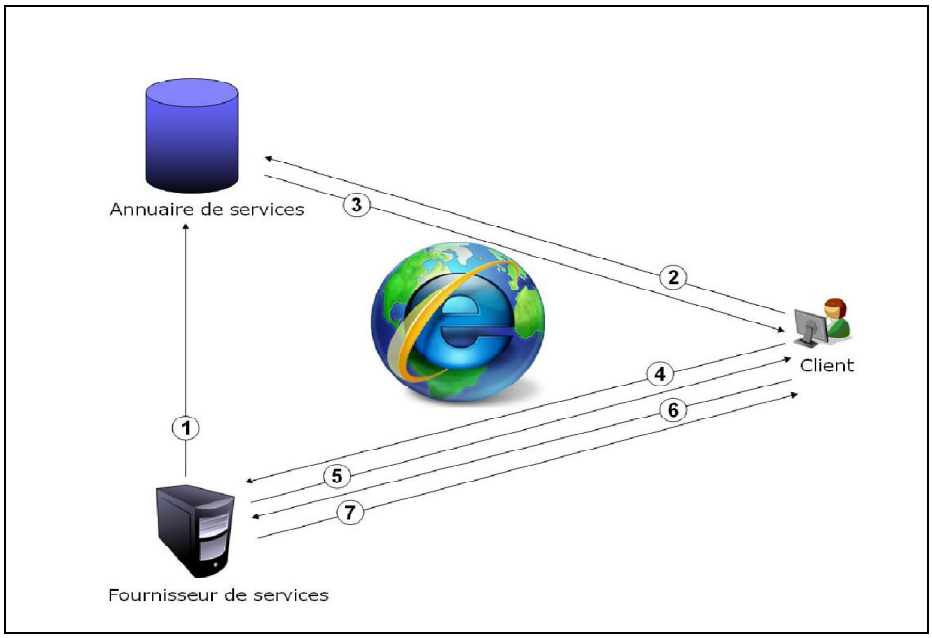


Figure 1. 1:Modèle fonctionnel de l'architecture SOA.

Le scénario se déroule dans la figure 1.1 comme suit. Un service est implémenté, décrit et publié par un fournisseur de services dans un annuaire de services (1).

Un client interroge l'annuaire en utilisant les outils de recherche et de découverte fournis par l'annuaire afin de trouver un service qui implémente les fonctionnalités recherchées (2,3). Une fois un service trouvé, le client se connecte au fournisseur, obtient la description du service (4,5), et invoque les opérations du service (6). Enfin, le service répond aux requêtes envoyées par le client (7) [3].

I.3.2. Les concepts de SOA

Les concepts de SOA sont à la mode dans le monde des systèmes d'information. Ces concepts énoncent diverses promesses:

➢ ***La réutilisation*** : qui signifie la possibilité de réutiliser des services par plusieurs consommateurs.

➢ ***La composition des services*** **:** qui est liée à la réutilisation. La composition est possible grâce à la modularité des services et permet d'assurer une fonctionnalité plus importante, en s'appuyant sur la composition de services

➢ ***L'autonomie des services***: peut être définie par la capacité des services à manipuler ses ressources.

➢ ***L'interopérabilité des services*** : c'est-à-dire leur interconnexion sans programmation spécifique [2].

I.4. Les technologies des services Web

La majorité des grands sites web (Amazon, eBay…) qui proposent des *services Web* aux développeurs, offrent simultanément deux catégories de *services Web*.

➢ ***Les services web REST :***exposent entièrement ces fonctionnalités comme un ensemble de ressources (URI) identifiables et accessibles par la syntaxe et la *sémantique* du protocole HTTP,

➢ ***Les services web SOAP :*** exposent ces mêmes fonctionnalités sous la forme de services exécutables à distance. Un *service web* est composé de quatre grandes parties comme la figure1.2: la description de l'interface du *service web* grâce au *langage WSDL* (Web Services Description Language), la sérialisation des messages transmis via le *protocole SOAP* (Simple Object Access Protocol), l'indexation des *services Web* dans des *registres UDDI* (Universal Description, Discovery Integration) et la sécurité des *services web*, obtenue essentiellement grâce à des protocoles d'authentification et de cryptage XML.

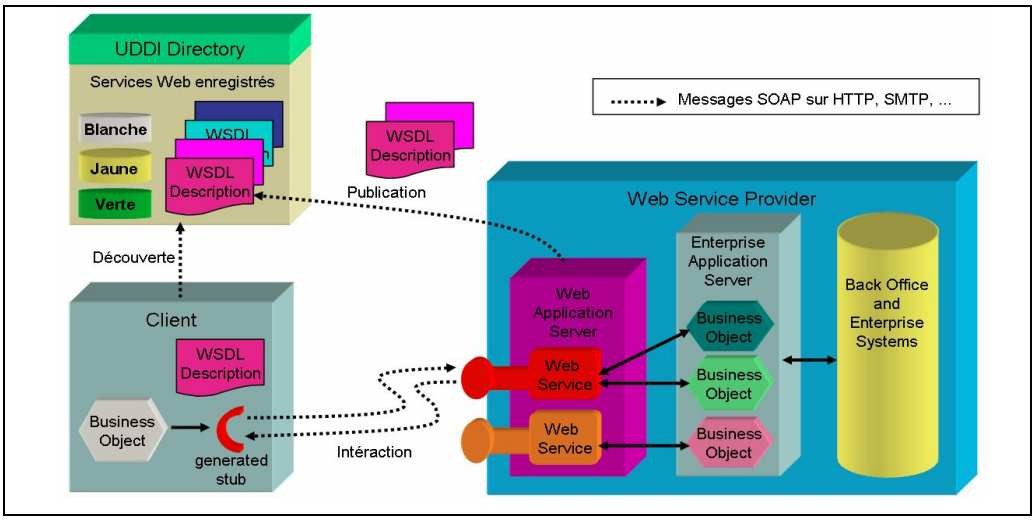


Figure 1. 2 : Les technologies des services Web.

I.4.1. Langage XML

(eXtensibleMarkupLanguage) est une famille de technologies développées au sein du W3C (Worle Wide Web Consortium).

XML est largement utilisé par les entreprises et supporté par les manufacturiers informatiques. Il est indépendant des plates-formes informatiques. Il est lisible par l'humain mais est destiné à être lu par la machine.

Les *services web* communiquent via XML dans le but de décrire leurs interfaces et encoder leurs messages. des documents sortant bruts de fonderie des traitements de texte ou des tableurs, le format XML a été conçu pour permettre le déchiffrement direct par l'utilisateur comme par les programmes.XML est un texte contenant des balises ouvrantes et fermantesqui décrivent la nature des données qu'elles encadrent. En plus, les protocoles de communication dans les Web services utilisent les interfaces et les messages d'XML, que n'importe quelle application peut interpréter, pour définir les interactions [1].

I.4.2. Langage WSDL

La description d'un service doit inclure la définition des composants nécessaires au protocole de communication (SOAP pour les *services web*) et à l'interaction avec un client ou un autre *service web*.

Les problématiques de réutilisation et d'interaction ont guidé le W3C afin de définir les catégories d'information à prendre en compte dans la description d'un service Web. Les différents éléments décrits dans WSDL sont les suivants :

- Les opérations proposées par le service Web ;
- Les données et messages échangés lors de l'appel d'une opération ;
- Le protocole de communication ;
- Les ports d'accès au service ;

Dans WSDL, il existe une séparation entre deux niveaux indépendants, respectivement nommés abstrait et concret. Le *niveau abstrait* regroupe les informations pouvant être réutilisées (non spécifique à un service), tandis que le *niveau concret* est constitué de la description des protocoles d'accès au service Web (information particulière à un service).

Le niveau abstrait est utilisé principalement lors du processus de sélection, tandis que le niveau concret est seulement utilisé lors de l'invocation des méthodes du service Web.

✓ **Le niveau abstrait :** décrit les informations propres aux méthodes proposées par le service, ainsi que les informations traitant des messages et des données échangés lors de l'invocation du service. Si deux services proposent les mêmes méthodes, le niveau abstrait de description WSDL peut être réutilisé. Ce niveau est composé des informations suivantes :

Les types de données : Précise les types de données complexes, pour lequel WSDL emplois XML schéma la figure 1.3.

- **Les messages**. l'abstraction décrivant les données échangées entre les services web.
- **Les opérations**l'abstraction décrivant une action implémentée par un Web service.

✓ **Le niveau concret :**décrit la manière dont le client accède à un service en particulier, et donc de ce fait, il est non réutilisable (propre à un service unique). Les informations décrites dans le niveau concret sont les suivantes :

Le protocole de communication (binding) un protocole concret d'accès à un port et un format de spécification des messages et des données pour ce port [10].

- **Les ports d'accès au service**. Dans un document WSDL, l'accès au service est défini par une collection de ports d'accès. Chaque port représente la localisation du service (*i.e*. son URL).Un même service peut être accessible sur plusieurs ports différents.

Les informations contenues dans un WSDL constituent la description du profil fonctionnel du service. Avec WSDL, le client peut invoquer le service par le biais de sa description abstraite (méthodes disponibles, paramètres d'entrée et sortie) et concrète (description des protocoles de communication et des points d'accès du service) [8].

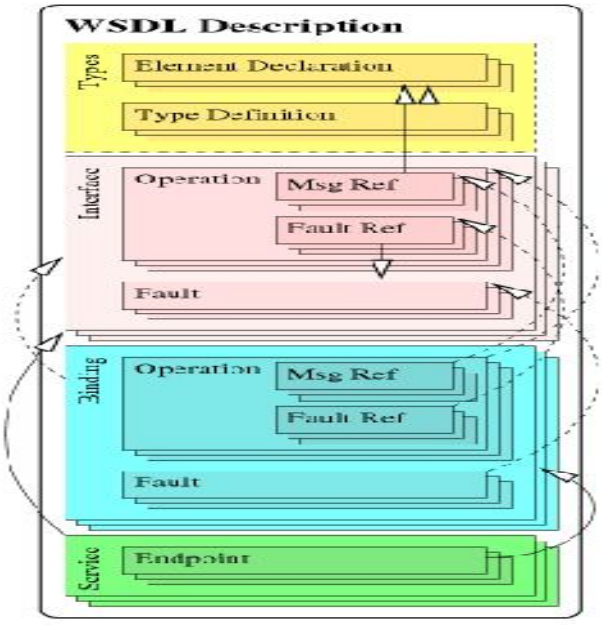


Figure 1. 3:Langage WSDL.

I.4.3. Protocole SOAP

Le protocole SOAP (Simple Object Access Protocol) est un protocole de communication base sur XML qui permet aux services Web d'échanger des informations dans un environnement décentralise et distribue tout en s'affranchissant des plate-formes et des langages de programmation utilises. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux *services web*. C'est le rôle du langage WSDL.

Un message SOAP est un document XML ordinaire qui contient les éléments suivants comme la figure 1.4.

- **L'élément Envelope :** qui identifie le document XML comme étant un message SOAP
- **L'élément Header :** qui est optionnel et qui contient des informations d'entête
- **L'élément Body :** qui contient l'appel ainsi que la réponse retournée
- **L'élément Fault** : qui est optionnel et qui fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message [4].

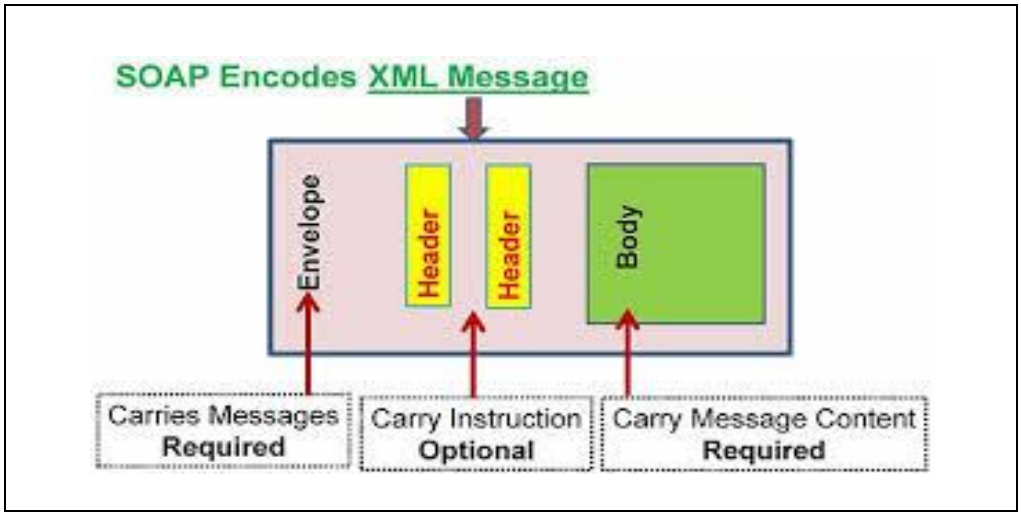


Figure 1. 4:Protocole SOAP.

I.4.4. UDDI (Universal Description, Discovery and Integration)

L'UDDI définit les mécanismes permettant de répertorier des *services web*. Ce standard gère donc, l'information relative à la publication, la découverte et l'utilisation d'un *service web*. En clair, l'UDDI définit un registre des *services web* sous un format XML. Les organisations publient les informations décrivant leurs *services web* dans l'annuaire, et

l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le *service web* qui fournit le service désiré ; ainsi l'UDDI a été créé pour faciliter la découverte de *services web* en plus de leurs publications.

L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP .L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte.

La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP. L'UDDI peut être vu comme un annuaire contenant :

- ***Les pages blanches*** **:** noms, adresses, identifiants et contacts des entreprises enregistrées.
- ***Les pages jaunes*** **:** donnent les détails sur le métier des entreprises et les services qu'elles proposent.
- ***Les pages vertes*****:** contiennent des informations techniques du service offert, la manière d'interagir avec le service, et aussi un pointeur vers le fichier WSDL et une clé unique identifiant le service [5].

I.5. Avantage des Web services

L'idée fondamentale derrière des *services web* est de morceler les applications et les processus en segments réutilisables appelés service de telle sorte que chacun de ces segments effectuent une tache distincte. Ces services peuvent alors servir à l'intérieur et à l'extérieur de l'entreprise, facilitant l'interopérabilité entre tous ces services.

Par leur nature, les Web services :

- Permettent à des portions de logiciels écrits dans différents langages, ou évoluant sur différents systèmes d'exploitation, de communiquer entre elles facilement et à peu de frais,
- Permettent à des applications supportant différents processus d'une organisation ou de différentes organisations, de communiquer entre elles et /ou d'échanger des données facilement et à peu de frais.

Plus spécifiquement, les Web services devraient permettre aux entreprises de :

- Donner aux clients un accès direct à l'information, aux données et aux fonctionnalités dont ils ont besoin pour interagir avec une entreprise;

- Donner aux partenaires d'une entreprise un accès direct à la fonctionnalité dont ils besoin pour mieux servir les clients qu'ils ont en commun avec cette entreprise;
- Donner aux fournisseurs d'une entreprise un accès direct à l'information et à la fonctionnalité dont ils ont besoin pour leur permettre d'ajuster les inventaires [1].

I.6. Inconvénients

La technologie des services web comporte plusieurs inconvénients dont:

- Problèmes de sécurité: Il est facile de contourner les mesures de sécurité mises en place par les pare-feu où l'utilisation du protocole HTTP.
- Confiance: Les relations de confiance entre différentes composantes d'un *service web* sont difficiles à bâtir, puisque parfois ces mêmes composantes ne se connaissent même pas.
- Syntaxe et sémantique: On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (la *sémantique*).
- Disponibilité: Les services web peuvent bien satisfaire un ou plusieurs besoins du client. Seront-ils pour autant toujours disponibles et utilisables? Ça reste un défi pour les *services web*.

I.7. La composition des services Web

La composition ou l'agrégation des services Web est une opération qui consiste à construire de nouveaux services appelés : services composites par un assemblage de services déjà existants nommés services basiques ou élémentaires.

La composition spécifie quels services doivent être invoqués, dans quel ordre et sous quelles préconditions.

Les services basiques peuvent être soient des services atomiques soient des services composites.

Exemple : Une agence de voyages *« e-TravelAgency »*fournit typiquement les services pour: consultation, réservation, paiement et annulation de billets d'avion, de chambres d'hôtel et de locations de voiture. Afin de fournir ces services à ses clients, l'agence de voyages doit établir des liens avec d'autres entreprises : compagnies aériennes, compagnies de location de voitures, réseaux hôteliers et une institution financière (une banque).

Un enseignant-chercheur, habite à « *Alger* », doit se rendre à « *Constantine* »*Samedi* pour assister à un colloque. Il décide d'organiser son voyage par Internet en faisant appel à différents *services web* de l'agence « *e-TravelAgency* ». Sa requête comprend la réservation de billet d'avion et la location d'un véhicule pour la durée du séjour, ainsi que le payement de ces derniers.

Afin de satisfaire la demande du client, on propose la composition des *services web* montrée dans la Figure 1.5.

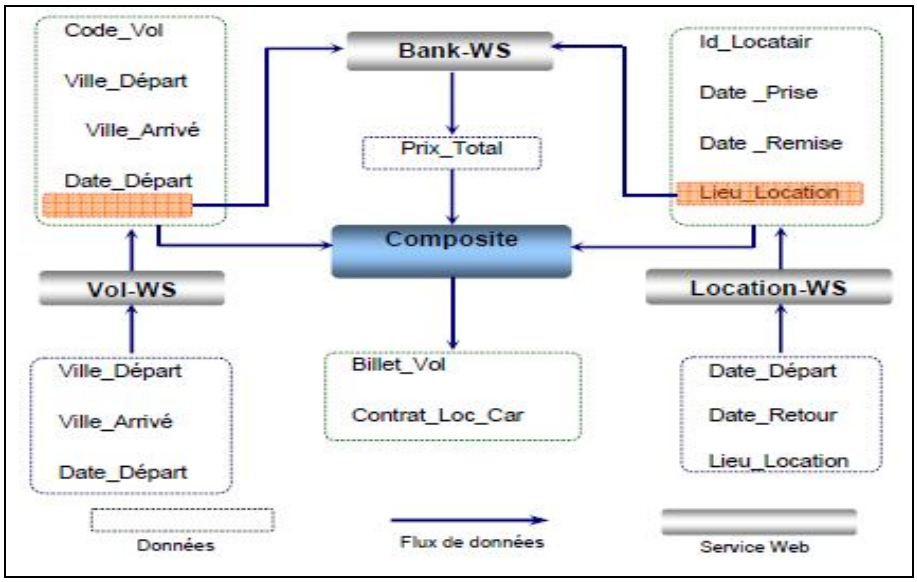


Figure 1.5: Composition de Services Web Vol-WS, Location -WS et Bank-.

La composition de services Web vise essentiellement quatre objectifs :

- Créer de nouvelles fonctionnalités en combinant des services déjà existants.
- Résoudre des problèmes complexes auxquels aucune solution n'a été trouvée.
- Faire collaborer plusieurs entreprises ensemble.
- Optimiser et améliorer une fonctionnalité existante [9].

I.7.1. Types de composition

La composition des *services web* peut être soit une composition statique soit une composition dynamique:

a- La composition statique

Est appelée aussi composition off-line. C'est une composition qui utilise des services basiques qui sont aux préalablement définis d'une façon figée et qui ne peuvent pas se changer en fonction du contexte du clientcomme la figure 1.6.

Il existe deux visions de la composition statique qui sont l'orchestration et la chorégraphie.

- **L'orchestration** : aborde le problème de façon centralisée, ou les collaborations de *Service web* statiques sont contrôlées par le service composé, tel un chef d'orchestre qui se charge d'ordonner les appels aux *services web* et de rattraper les erreurs.

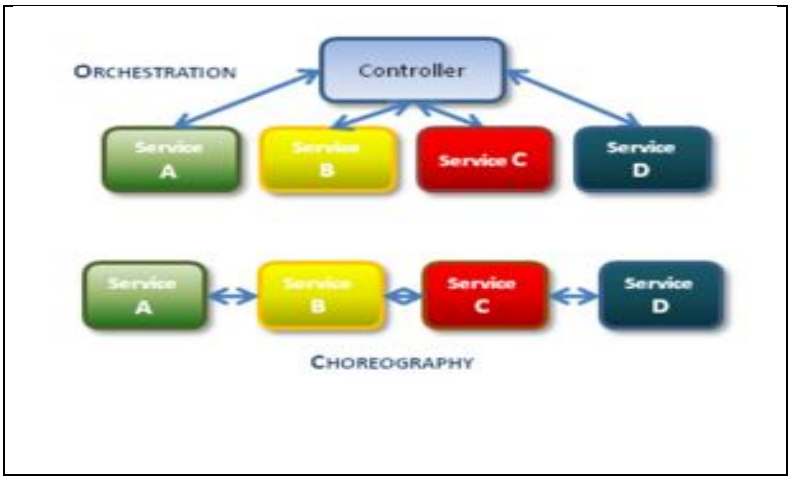


Figure 1.6: Orchestration et chorégraphie des services.

- **La chorégraphie** : aborde le problème de façon distribuée, chaque partenaire d'une composition, i.e. chaque fournisseur de *service web*, peut réaliser une ou plusieurs tâches, et chacun d'eux communique à l'aide des *services web*.

b- La composition dynamique

Appelée aussi composition on-line. la sélection des services basiques ne peut pas être prédéfinie à l'avance mais elle sera faite au moment de l'exécution en fonction des contraintes imposées par le client. Ceci permet d'élaborer des différents scénarios de composition qui offrent les mêmes fonctionnalités et qui tiennent compte de la dynamicité de la situation du client [9].

I. Partie 02:Web sémantique(définition, concept et technique)

I.1. Définitiondu Web sémantique

Le *Web Sémantique* est une extension du *Web* actuel, dans lequel l'information a un sens bien défini, et permet une meilleure coopération dans le travail entre les humains et les ordinateurs… « Le *Web* des données qui peuvent être traitées par les ordinateurs».

Le *Web sémantique* est donc une nouvelle approche pour l'organisation du contenu du *Web* instaurée dans le but d'améliorer l'interopérabilité, la *découverte* et la récupération des ressources. [10]

I.2. Définition du web services sémantiques

Les *services web sémantiques* sont la combinaison de deux technologies (Figure 1.7) : celle des *services web* et celle du *web sémantique*. Les *services web sémantiques* sont des *services web* dont la description est améliorée par des langages empruntés au *web sémantique*, tel que RDF et OWL. Cet emprunt au *web sémantique* permet à ces *services web* d'être découverts et sélectionnés automatiquement par des machines ou d'autres *services web* distants. Ceci permet aux *services web* sélectionnés de répondre au mieux à la requête du client [8].

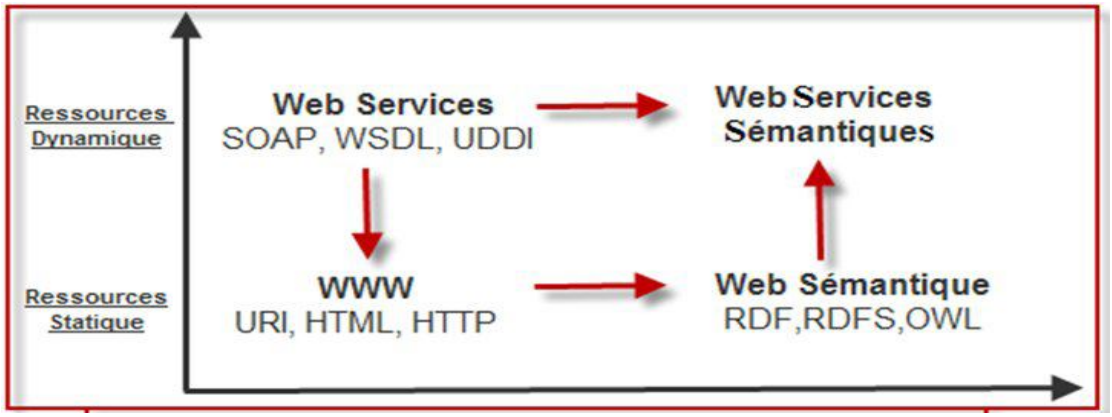


Figure 1.7: Origine des Web services sémantiques.

I.3. Approche de description à base d'annotations

I.3.1. Le standard WSDL

Le besoin d'une description claire de la communication entre *services web* a abouti à des normalisations aux niveaux des messages échangés et des protocoles. Une grammaire XML bien structurée était proposée pour décrire les services Web et paramétrer les échanges de messages. Un langage de description s'est basé sur cette grammaire et est devenu le standard des services Web, c'est le "Web Service Description Language" (WSDL) comme la figure 1.8.

Dans la suite, Nous allons présenter les principes du standard WSDL et ses deux spécifications largement utilisées par les chercheurs et les industriels.

En effet, afin qu'un client puisse communiquer avec un service Web décrit par une telle interface abstraite, il doit savoir comment et vers où les messages sont envoyés sur le réseau. C'est la partie concrète qui précise les détails sur le moyen d'échange de messages d'une ou plusieurs interfaces. Tout est spécifié par les éléments "binding".

Figure 1.8:Structure d'une description WSDL.

I.3.2. L'approche SAWSDL

Semantic annotation for WSDL (SAWSDL) est une approche sémantique de description de service Web. Il est évolutif et compatible avec les standards des services Web existants, et plus spécifiquement avec WSDL.

SAWSDL augmente l'expressivité du langage WSDL avec la sémantique en utilisant des concepts analogues à ceux utilisés dans OWL-S. D'une part SAWSDL, fournit un mécanisme permettant d'annoter sémantiquement les types de données, les opérations, les entrées et les

sorties de WSDL. Les aspects relatifs à la qualité et l'orchestration des services ne sont pas traités dans SAWSDL comme la figure 1.9[11].

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description
  targetNamespace="http://www.telecom-sudparis.eu/GeoLocApartBuyMapService.wsdl20"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:tns="http://www.telecom-sudparis.eu/GeoLocApartBuyMapService/"
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
  name="GeoLocApartBuyMapWebService">

  <wsdl:types>
    <xs:import    namespace="http://www.telecom-sudparis.eu/GeoLocApartBuyMapService.xsd"
          schemaLocation="http://www.telecom-sudparis.eu/GeoLocApartBuyMapService.xsd" />
  </wsdl:types>

  <wsdl:interface name="GeoLocApartBuyMapServiceInterface"
    sawsdl:modelReference="http://localhost/agensimo#terminalGPS
                           http://localhost/agensimo#buy
                           http://localhost/agensimo#apart">

    <wsdl:operation name="getLocationGPS"
      sawsdl:modelReference="http://localhost/agensimo#GPSactive
                             http://localhost/agensimo#GPSposition">
      <wsdl:input        element="tns:getSignalGPS"
        sawsdl:modelReference="http://localhost/agensimo#GPSsignal"/>
      <wsdl:output         element="tns:PositionGPS"
        sawsdl:modelReference="http://localhost/agensimo#GPScoordinates"/>
    </wsdl:operation>

    <wsdl:operation   name="getApartmentBuyList">...</wsdl:operation>
    <wsdl:operation   name="getMap">...</wsdl:operation>
  </wsdl:interface>

  <wsdl:binding   name="GeoLocApartBuyMapServiceSOAPBinding"
    interface="tns:GeoLocApartBuyMapServiceInterface" ...>    ...  </wsdl:binding>

  <wsdl:service   name="GeoLocApartBuyMapWebService"
    interface="tns:GeoLocApartBuyMapServiceInterface">
    <wsdl:endpoint   name="GeoLocApartBuyMapServiceEndpoint"
      binding="tns:GeoLocApartBuyMapServiceSOAPBinding"
      address="http://www.telecom-sudparis.eu/GeoLocApartBuyMapWebService/" />
  </wsdl:service>
</wsdl:description>
```

Figure 1.9:Extrait de la description SAWSDL du service Web de l'agence immobilière.

I.4. Langages de description de services Web sémantiques

Ils existent des travaux qui proposent une manière de représenter de façon *sémantique* des *services web* (autrement dit proposent de décrire des *services web sémantiques*). Malgré l'abondance de ce type de travaux, aucun ne s'est imposé comme une solution de description de *services web sémantiques*. OWL-S est l'un des travaux issus du W3C qui tentent d'apporter une solution standard en termes de description de *services web sémantiques*.

I.4.1. OWL-S

Le but initial du langage OWL-S est de mettre en œuvre des *services web sémantiques*. Cette mise en œuvre inclut un grand nombre d'objectifs, rendus possibles par le biais de l'expressivité héritée de OWL et de l'utilisation de la logique de description.
Ces objectifs sont :

- la description de services Web sémantiques,
- l'invocation automatique de ces services, par le biais de la détection et de l'interprétation automatique de la localisation et des paramètres d'entrée/sortie.
- la composition automatique de services (description et invocation) et la surveillance de l'exécution de la composition [9].

I.4.2. WSMO

L'architecture WSMO (Web Services ModelingOntology) est une architecture conceptuelle visant à expliciter la *sémantique* des *services web*.

Conclusion

Dans ce chapitre nous avons vu des éléments de définition des Web services et les technologies les plus importantes qui les constituent, en l'occurrence : XML, SOAP, WSDL et UDDI, en plus au protocole de communication universels tel que HTTP. Nous avons vu aussi les avantages des Web services par rapport aux entreprises. Technologies les plus importantes qui les constituent, en l'occurrence : XML, SOAP, WSDL et UDDI, en plus au protocole de communication universels tel que HTTP. Nous avons vu aussi les avantages des Web services par rapport aux entreprises.

Chapitre2

L'ONTOLOGIE

DÉFINITION, CONCEPT

Introduction

La modélisation et la représentation du monde réel constituent un sujet très riche à explorer. Le besoin de représenter les connaissances d'un domaine spécifique d'une manière à travers laquelle les machines puissent manipuler la *sémantique* des informations a donné naissance aux *ontologies* [7].

L'ontologie est un élément-clé de la résolution du problème de l'hétérogénéité *sémantique*, permettant ainsi l'interopérabilité *sémantique* entre les différentes applications Web et des services.

Récemment, les *ontologies* sont devenues un sujet de recherche populaire dans de nombreuses communautés, y compris l'ingénierie des connaissances, le commerce électronique, la gestion des connaissances et traitement du langage naturel [14].

II.1. Qu'est une ontologie ?

Une *ontologie* définit un vocabulaire commun pour les chercheurs qui ont besoin de partager l'information dans un domaine.

Pour quelles raisons développer une *ontologie* ? Voici quelques-unes:

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir sur un domaine.
- Expliciter ce qui est considéré comme implicite sur un domaine.
- Analyser le savoir sur un domaine [1].

En informatique, une *ontologie* est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être:

- des relations *sémantiques*;
- des relations de composition et d'héritage (au sens objet).

La structuration des concepts dans une ontologie permet de définir des termes les uns par rapport aux autres, chaque terme étant la représentation textuelle d'un concept. Par exemple, pour décrire les concepts entrant en jeu dans la conception de cartes électroniques (Figure2.1), on pourrait définir *l'ontologie* (simplifiée ici) suivante:

- une *carte électronique* est un ensemble de *composants* · un *composant* peut être soit un *condensateur*, soit une *résistance*, soit une *puce*.
- une *puce* peut être soit une *unité de mémoire*, soit une *unité de calcul* ;
- une *carte électronique* qui contient une *unité de calcul* contient aussi au moins une *unité de mémoire*

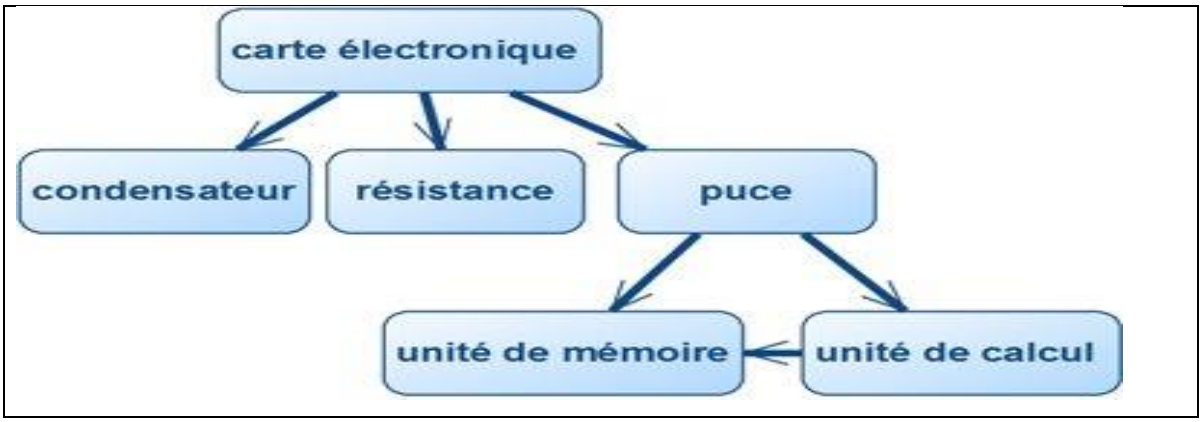


Figure 2. 1: exemple de la carte électronique.

Simplement, on peut construire une *ontologie* à partir d'un corpus de textes. On va parcourir le texte à la recherche de termes récurrents ou définis par l'utilisateur, puis analyser la manière dont ces termes sont mis en relation dans le texte.

Le résultat est une *ontologie* qui représente la connaissance globale que contient le corpus de texte sur le domaine d'application qu'il couvre.
« Une *ontologie* est une spécification formelle explicite d'une conceptualisation partagée »
Les attributs "formelle" et "explicite" signifient qu'une *ontologie* permet une interprétation automatisée de la conceptualisation par la machine.

Les *ontologies* peuvent être utilisées par des personnes, des bases de données et des applications qui ont besoin de partager des informations sur un domaine. Elles incluent des définitions, informations exploitables, des concepts élémentaires dans ce domaine et de leurs relations [13].

II.2. Composantes d'une ontologie

Comme nous l'avons abordé, les ontologies fournissent un vocabulaire commun d'un domaine et définissent la signification des termes et des relations entre elles. La connaissance dans les ontologies est principalement formalisée en utilisant les cinq types de composants : concepts (ou classes), relations (ou propriétés), fonctions, axiomes (ou règles) et instances (ou individus).

II.2.1. Les concepts

Les concepts sont appelés aussi classes de l'ontologie [1].Cette composante est représentée dans les graphes hiérarchiques. Le concept est représenté par une "super-classe" (Figure 2.2), représentant la classe supérieure ou soi-disant "classe parent", et une "sous-classe" qui représente le subordonné ou soi-disant "classe fils"[14].

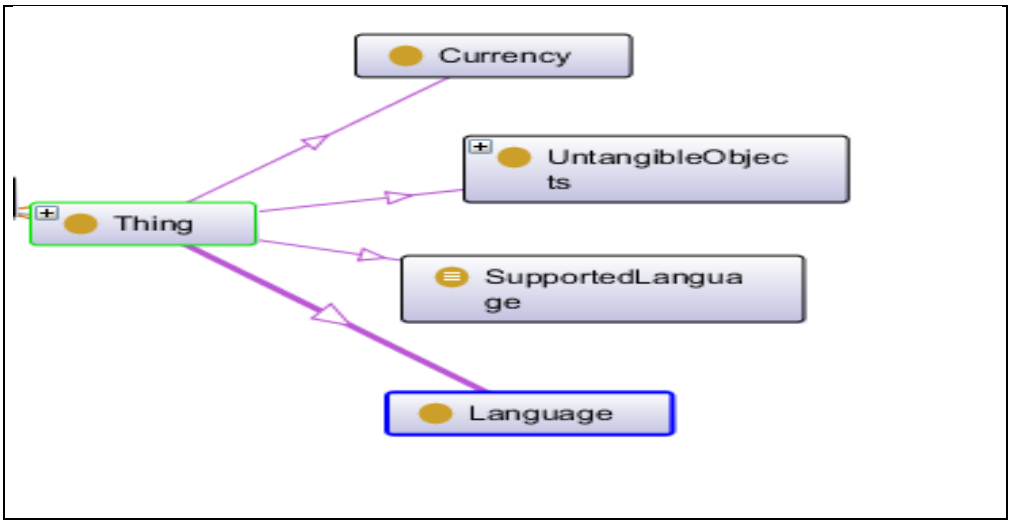


Figure 2. 2: exemple de concept : super-classe et sous-classe.

Un concept est composé de trois parties :

- **Une notion :** elle correspond à la sémantique du concept, elle est définie à travers ses propriétés et ses attributs. Elle est appelée **intention** du concept.
- **Un ensemble d'objets :** il correspond aux objets définis par le concept, il est appelé **extension** du concept. Les objets sont les **instances** du concept.
- **Un (ou plusieurs) terme(s) :** les termes permettent de désigner le concept. Ces termes sont aussi appelés **labels** de concept.

Exemple : le **terme** « lapin » renvoie à la **notion** « animal » possédant de longues oreilles et une queue et à **l'ensemble des objets** ayant cette description.

II.2.2. Les relations

Elles représentent des interactions entre concepts permettant de construire des représentations complexes de la connaissance du domaine. Elles établissent des liens sémantiques binaires, organisables hiérarchiquement (cf. Figure 2.3).

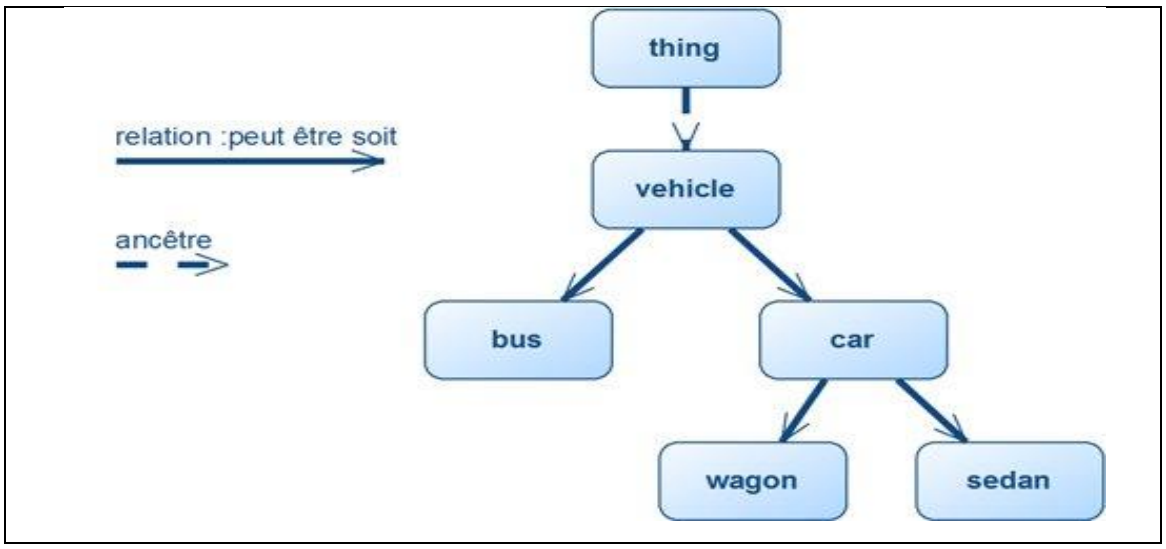


Figure 2. 3:les relations entre les concepts.

Exemple : les concepts « Personnalité » et « Film » sont reliés entre eux par la relation sémantique « réalise (Personnalité, Film) ».

II.2.3. Les axiomes (ou Règles)

Les axiomes sont des expressions qui sont toujours vraies. Ils ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n'ayant pas un caractère strictement terminologique.

Leur inclusion dans une ontologie peut avoir plusieurs objectifs :

- Définir la signification des composants.
- Définir les arguments d'une relation.

II.2.4. Les instances (ou individus)

Elles constituent la définition extensionnelle de l'ontologie ; elles sont utilisées pour représenter des éléments dans un domaine.

Exemple: les individus *Mohamed* et *Maha* sont des instances du concept «Personne» [12].

II.3. Structure de l'ontologie

D'une manière générale, la structure d'une ontologie est décrite comme un 5-tuples O :=(C, HC, R, RH, I), où:

- C : représente un ensemble de concepts (instances de"RDF: Classe") : Ces concepts sont disposés selon la hiérarchie de subsomption correspondante.
- R : représente un ensemble de relations qui relie des concepts à un autre (instances de"rdf: Property").
- HC : représente une hiérarchie de concepts sous la forme d'une relation (une relation binaire correspondant à «rdfs: subClassOf»). où HC (C1, C2) signifie que C1 est un sous-concept de C2.
- RH : représente une hiérarchie de relations sous la forme d'une relation : où RH (R1, R2) indique que R1 est une sous-relation de R2 ("rdfs: subPropertyOf»).
- I : est l'instanciation des concepts dans un domaine particulier ("Rdf: type").

II.4. La recherche d'information guidée par les ontologies

L'utilisation typique du Web actuel consiste en la recherche d'informations qui peut être d'ordre professionnel (veille stratégique/technologique, recherche d'articles…) ou d'ordre personnel (recherche de personnes ou de produits).

Pour faciliter ces tâches, plusieurs moteurs de recherche ont vu le jour (Google, Yahoo, Altavista…). Ces outils, bien qu'ils répondent à une bonne partie des besoins des utilisateurs, présentent quelques problèmes critiques :

- la masse énorme des documents retournés,
- la sensibilité au vocabulaire utilisé dans la requête,

- le résultat fractionné en pages Web, ce qui entraîne le besoin de faire plusieurs requêtes pour obtenir tous les documents pertinents et après en extraire manuellement la partie intéressante,
- la variabilité des langages utilisés sur le web et la non-structuration des documents, ce qui rend cette tâche de plus en plus laborieuse.

La réflexion sur le *web sémantique* a été essentiellement fondée sur ce problème de la recherche d'informations. En effet, les *ontologies* peuvent améliorer la pertinence d'une recherche et ce, en recherchant des documents faisant référence à un concept précis au lieu de se baser sur des mots-clés qui peuvent être ambigus.

Prenons l'exemple d'une personne anglophone qui cherche à trouver l'adresse d'un installateur de fenêtres ; en tapant la requête « Windows installation » dans n'importe quel moteur de recherche, elle obtiendra des milliers de pages traitant l'installation du système d'exploitation de Microsoft et les problèmes qui en résultent, mais elle aura beaucoup de mal à trouver l'information qu'elle recherchait.

Avec l'utilisation d'une *ontologie,* un moteur de recherche fera la différence entre un site sur lequel 'Windows' désigne un logiciel et un autre sur lequel il désigne une fenêtre.

Cette recherche basée sur les *ontologies* se présente comme une recherche intelligente qui repose sur la *sémantique* des ressources et sur les concepts contenus dans les documents qui leur sont associées. Ces *ontologies* peuvent ainsi, d'une part, guider la création d'annotations sous la forme de métadonnées sur les ressources, et, d'autre part, décrire leurs contenus de manière à la fois formelle et signifiante pour être exploitable aussi bien par les humains que par les machines.

II.5. Formalismes de représentation

Les modèles de représentation de connaissance utilisés en ingénierie ontologique peuvent être regroupés selon les paradigmes conceptuels qu'ils réifient. Sont ainsi distingués :

a) Les graphes

i) Les frames

ii) Les graphes conceptuels.

iii) Orienté-Web : RDF et RDF Schéma

b) Logique

i) Du premier ordre: KIF

ii) Descriptive: KL-One, OIL, DAML+OIL, OWL

II.5.1. Frames

Le principe de ce modèle est de décomposer les connaissances en *classes* (ou frames) qui représentent les concepts du domaine.

II.5.2. Graphes conceptuels

Les graphes conceptuels sont décomposés en deux niveaux: le niveau terminologique où sont décrits les concepts, les relations et les instances de concepts, ainsi que les liens de subsomption entre concepts et entre relations et le niveau assertionnelle où sont représentés les faits, les règles et les contraintes sous forme de graphes où les sommets sont des instances de concepts et les arcs des relations[12] .

II.5.3. Logiques de description

Une LD est composée de deux parties : un langage terminologique *TBox* et un langage assertionel *ABox*. Le langage assertionnel est dédié à la description de faits et le langage terminologique à la description de concepts et de rôles. La principale tâche de raisonnement au niveau terminologique est de calculer les relations de subsomption entre concepts.

Le *TBox* constitue la partie terminologique de la base de connaissances. Il contient la déclaration des termes et des rôles. Chaque nouveau concept est défini à base des concepts déjà définis, pour fournir à la fin un ensemble de concepts de plus en plus complexe. Cet ensemble de concepts est classifié en vérifiant la subsumption des concepts qui positionne chaque concept à la bonne place dans la hiérarchie.

II.6. Langages de spécification d'ontologies

Plusieurs langages de spécification d'ontologies ont été développés pendant ces dernières années. Parmi ces langages, nous citons :

II.6.1. KIF

KIF (Knowledge Interchange Format) : est un langage basé sur les prédicats du premier ordre avec des extensions pour représenter des définitions et des méta-connaissances.

II.6.2. KL-ONE

KL-ONE : est un langage basé sur la logique de description. Ce système maintient la définition des concepts par un simple nommage, et l'indication de la correspondance des concepts dans une hiérarchie de généralisation/spécialisation.

II.6.3. RDF(S) Resource Description Framework and RDF schema

RDF « *Resource Description Framework* » est un formalisme graphique pour représenter des méta-données. Il est basé sur la notion de triplet (sujet, prédicat, objet).

Le sujet et l'objet sont des ressources liées par le prédicat. RDF utilise la syntaxe XML, mais il ne donne aucune signification spécifique pour le vocabulaire comme sous classe de, ou le type. Les primitives de modélisation offertes par RDF sont très basiques.

RDF Schéma est un langage qui étend RDF avec un vocabulaire de termes et les relations entre ces termes, par exemple *Class*, *Property*, *type*, s*ubClassOf, subPropertyOf, range* et *domain*.

RDFS est reconnu comme un langage d'ontologie qui définit :

- Des classes et des propriétés.
- Les sous-classes, les super-classes, les sous-propriétés, et les super-propriétés.
- Le domaine de définition et le domaine image des propriétés.

II.6.4. OIL (Ontology Interchange Language)

Dans l'optique d'une utilisation d'ontologies sur le Web, le langage RDF(S) a été enrichi par l'apport du langage OIL (Ontology Interchange Language) qui permet d'exprimer une sémantique à travers le modèle des frames tout en utilisant la syntaxe de RDF(S).

II.6.5. DAML+OIL (DARPA Agent Markup Language +OIL)

DAML + OIL : est un langage construit sur des normes précédentes du W3C telles que RDF et RDF Schéma, et étend ces langages avec des primitives de modélisation plus riches.

II.6.6. OWL (Ontology Web Language)

OWL est considéré par W3C comme un langage d'ontologie standard (cf. Figure 2.4). Il a non seulement la capacité de décrire les concepts dans un domaine mais aussi d'un ensemble plus riche d'opérateurs, donc ses concepts sont bien définis et bien décrits. On peut construire des concepts complexes en basant sur les définitions des concepts plus simples. En outre, on peut vérifier si tous les relations et les définitions dans l'ontologie sont conformés, et identifier quels concepts s'adaptent sous quelles définitions. Donc, on peut maintenir la hiérarchie correctement entre les classes.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY books.owl "http://127.0.0.1/ontology/books.owl">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY simplified_sumo.owl "http://127.0.0.1/ontology/simplified_sumo.owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xml:base="&books.owl;" xmlns:owl="&owl;" xmlns:rdf="&rdf;" xmlns:rdfs="&rdfs;">
<!-- Ontology Information -->
<owl:Ontology rdf:about="" rdfs:label="Book Ontology" owl:versionInfo="1.0"> <rdfs:comment>An ontology
<owl:imports> <owl:Ontology rdf:about="&simplified_sumo.owl;"/> </owl:imports></owl:Ontology>
<!-- Classes -->
<owl:Class rdf:about="#A"><rdfs:subClassOf rdf:resource="#Grade"/></owl:Class>
<owl:Class rdf:about="#Article"><rdfs:subClassOf rdf:resource="#Text"/></owl:Class>
<.....>
<owl:Class rdf:about="#Genre"/>
<owl:Class rdf:about="#Grade"/>
<.....>
<owl:Class rdf:about="#Monograph">
<rdfs:subClassOf rdf:resource="#Publication"/>
<rdfs:subClassOf><owl:Restriction><owl:allValuesFrom rdf:resource="#Once"/><owl:onProperty rdf:resource
</owl:Class>
<!-- Object Properties -->
<owl:ObjectProperty rdf:about="#contains"/>
<owl:ObjectProperty rdf:about="#datePublished"/>
<owl:ObjectProperty rdf:about="#hasGenre"/>
```

Figure 2. 4:exemple d'un OWL.

Conclusion

Dans ce chapitre, l'ontologie est présentée comme un outil essentiel pour représenter le monde réel par la modélisation des concepts de ce monde dans un modèle représentatif, décrit par les classes existantes, et les propriétés qui définissent les relations établies entre ces concepts. Le standard OWL s'est montré capable de jouer un rôle du pivot, grâce à sa compatibilité avec d'autres langages DAML+OIL, RDF, et RDFS.

Chapitre 3

ETAT DE L'ART SUR LA DISTANCE SÉMANTIQUE

Introduction

La sémantique c'est le sens, c'est ce qu'on appelle aussi parfois le « fond » par opposition à la forme. Par exemple si je compare les mots « voiture » et « véhicule » d'un point de vue sémantique ce sont des mots relativement proches par leur sens. Je peux en général remplacer le mot voiture par le mot véhicule, cela ne changera pas le sens du discourt. La sémantique est naturelle et facile pour un humain, par contre un ordinateur est incapable de voir la proximité de sens entre ces deux mots.

III.1. Définition

La distance sémantique est une valeur attribuée à deux concepts qui dépend de la similarité entre ces derniers. Cette distance sémantique est calculée selon plusieurs moyens comme nous allons voire dans ce chapitre.

Notez que cela ne veux pas dire que l'ordinateur va comprendre réellement le sens du texte, mais il sera capable de les comparer et de mesurer leur similitudes ou différences sémantique.

III.2. Mesure de similarité sémantique basée sur l'ontologie

Ce chapitre passe en revue les différents paradigmes utilisés pour calculer les mesures de similarité sémantique à partir d'une ontologie (SSMs :semantic similarity measures).

SSMs visent à estimer la ressemblance entre deux concepts en se basant sur les connaissances taxonomiques modélisées dans les ontologies.

III.2.1. Des approches à base d'arêtes

Les mesures de la similitude estimée entre deux concepts par ces approches seront selon la force de leur interconnexion dans l'ontologie. L'approche la plus courante considère la similitude comme une fonction qui calcule la distance qui sépare les deux concepts dans l'ontologie.

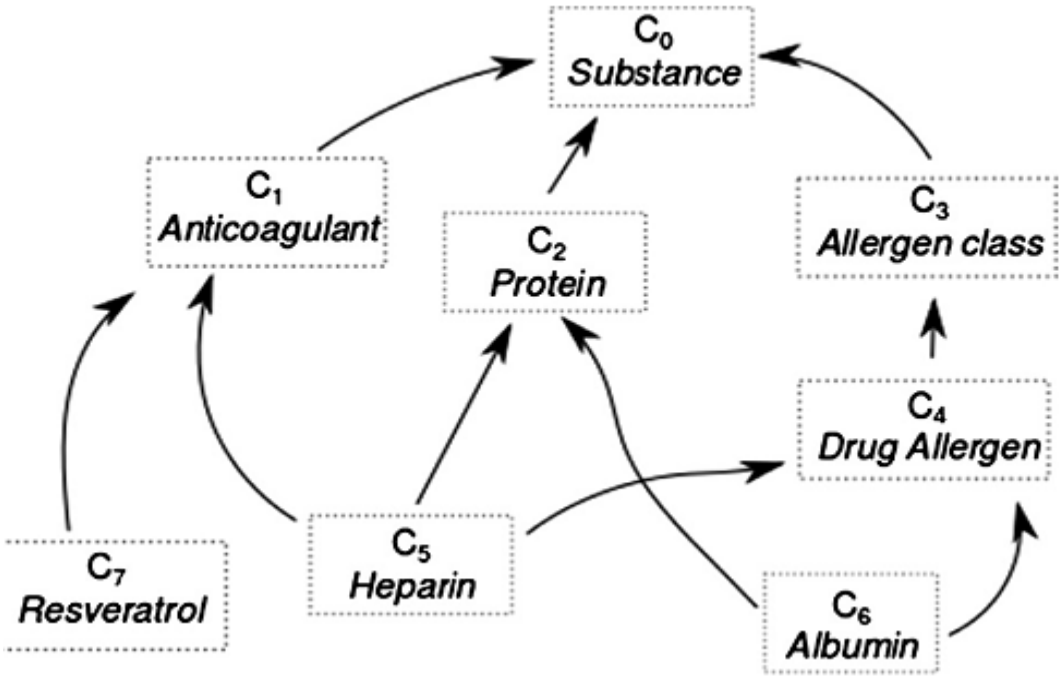


Figure 3. 1: Une Capture d'un ensemble des concepts définis dans l'ontologie SNOMED-CT.

Par exemple, *Rada et al*[21] estime la distance entre de deux concepts *u, v* comme le plus court chemin qui les relie *(sp (u, v)) :*

$$\mathbf{Dist}_{rada}(u,v)=sp(u,v)$$

Dans la figure 3.1, le plus court chemin entre les deux concepts c5 et c3 est C5→c4→c3. *Leacock et Chodorow* ont proposés une adaptation non-linéaire de la distance de *Rada* pour définir la mesure de similarité Sim_{LC}[33] :

$$\mathbf{Sim}_{LC}(u,v)=-\log\left(\frac{sp(u,v)}{2.MAX\_depth}\right)$$

- La distance de *Rada* est normalisée par la profondeur maximale de l'ontologie, Max_depth, c'est-à-dire le plus long des chemins les plus courts reliant un concept au concept qui englobe tous les autres.

La racine de l'ontologie est C_0 dans la Figure 3.1.

Certaines approches raffinent un peu plus en proposant de :

- Prendre en considération les variations du poids des liens entre les concepts.

- Et aussi plus les deux concepts sont profonds, plus leur relation sémantique est considérable.

Dans la plupart des cas, la similarité sémantique entre deux concepts est estimée en fonction de la profondeur de l'ancêtre commun le plus proche (LCA : Least Common Ancestor), c'est-à-dire le nœud (ancêtre)commun entre les deux concepts.

Dans la Figure 3.1,

- le *LCA* des concepts C_5 et C_3 est C_0.
- le *LCA* des concepts C_5 et C_7 est C_1.
- le *LCA* des concepts C_5 et C_6 est C_2.

Remarque :

- Plus le LCA est profond→ plus la spécification entre les concepts est considérable→ Donc, plus de similarité entre les concepts comparés.

Basant sur cette remarque, **Wu et Palmer** proposent $\mathbf{Sim}_{wp}$[17]et *Pekar et Staab* proposent de $\mathbf{SIM}_{PK}$:

$$\mathbf{Sim}_{wp}(u,v)=\frac{2.depth(\mathrm{LCAu,v})}{depth(u)+depth(v)}$$

$$\mathbf{SIMpk(u,v)} = \frac{sp(\mathrm{LCAu,v},root)}{sp(u,\mathrm{LCAu,v}) + sp(v,\mathrm{LCAu,v}) - sp(\mathrm{LCAu,v},root)}$$

La plupart de ces mesures satisfont l'identité de la propriété indiscernable (IOI : the Identity Of the Indiscernibles property), c'est-à-dire la similarité (respectivement la distance) d'un concept avec lui-même est maximale (minimale).

Exemple :

$$\mathbf{Dist}_{Rada}(u, u) = 0.$$

III.2.2. Les Approches à base de nœuds

Les approches à base de nœuds se concentrent sur l'évaluation des concepts définis dans l'ontologie.

Deux stratégies spécifiques peuvent être distinguées:

a- Stratégies à base de caractéristiques

Ces stratégies évaluent le concept comme un ensemble de caractéristiques. Ils se situent dans le modèle-à-caractéristiques proposé par Tversky [22].

Pour les mesures à base d'ontologies, les caractéristiques d'un concept sont généralement considérées comme l'ensemble des concepts qui l'englobent, c'est-à-dire, ancêtre $A(u)=\{v|u\leq v\}$.

Autrement dit, un concept se caractérise par les sémantiques qui sont hérités de ses ancêtres. À la (figure 1), le conceptC_6 sera donc représenté par l'ensemble des caractéristiques une

$A(C_6) = \{C_0, C2, C3, C4, C6\}$.

Par exemple, *Batet et coll*. évaluent la distance comme le rapport entre les caractéristiques distinctes et les caractéristiques partagés [23] :

$$\mathbf{Dist_{Batet}}(u,v)=\log_2\left(1+\frac{|A(u)\backslash A(v)|+|A(v)\backslash A(u)|}{|A(u)\backslash A(v)|+|A(v)\backslash A(u)|+|A(u)\cap A(V)|}\right)$$

Un autre exemple de mesures à base de caractéristiques est donné par *Rodríguez et Egenhofer*[24] :

$$\boldsymbol{Sim}\text{re}(u,v)=\frac{\boldsymbol{A(u)\cap A(v)}}{\boldsymbol{\gamma}.|\boldsymbol{A(u)\backslash A(v)}|+(\boldsymbol{1-\gamma}).|\boldsymbol{A(v)\backslash A(u)}|+|\boldsymbol{A(u)\cap A(v)}|}$$

Avec$\gamma\in$ [0, 1], un paramètre qui permet de régler la symétrie de la mesure.

Sim_{cmatch} est une mesure de similarité par correspondance entre concepts [25],est un autre exemple de mesures de similarités basées sur les caractéristiques :

$$\boldsymbol{Sim}cmatch(u,v)=\frac{\mathrm{A(u)}\cap\mathrm{A(v)}}{\mathrm{A(u)}\cup\mathrm{A(v)}}$$

b- Stratégies basées sur la théorie de l'information

Une approche basée sur la théorie de l'information c'est une approche qui évalue la similitude des concepts selon la quantité d'informations qu'ils fournissent, c'est-à-dire leurs contenu d'information (IC : Information Content).

À ce stade, *Resnik* calcule le CI d'un concept selon la théorie de l'information de Shannon comme une fonction de son utilisation dans un corpus [19]:

$$IC(u) = -\log\left(p(u)\right)$$

Avec ***p(u)*** la probabilité qu'un concept ***u*** se produit dans un document de corpus.

Plus un concept se produit, le moins informative sera considéré, en supposant qu'un concept se produit également lorsque les concepts qu'il subsume se produisent.

Par définition, l'IC diminue de façon monotone à partir du concept de terminal (feuilles) à la racine de l'ontologie. Cela garantit la cohérence taxonomique des résultats, i.e

$$u \leq v \Rightarrow IC(u) > IC(v)$$

Autrement dit, le IC d'un concept sera toujours supérieur à l'IC de l'un de ses subsumant.

La similitude de deux concepts est généralement estimée selon l'*IC* de leur ancêtre commun le plus informatif *(MICA : Most Informative Common Ancestor MICA)*, c'est-à-dire le concept qui subsume les deux concepts et il a le meilleur *IC*.

Dans la figure 3.1,

- Le concept C_1 est le *MICA* de la paire de concepts (C_5, C_7)

Resnik propose d'estimer la similarité sémantique comme suit [19]:

$$\text{SIMresnik}(u, v) = IC(MICA(u, v))$$

La mesure de *Resnik* ne capture pas explicitement la spécificité entre deux concepts, c'est à dire, l'IC des concepts évalués. En effet, les concepts avec le même *MICA* auront une même similarité, même si leur divergence vers leur MICA est différente.

Exemple : C'est le cas des paires *(C_2, C_3)* Et *(C_2, C_4),* Les deux ayant C_0 comme *MICA* dans la figure 3.1.

Par conséquent, la mesure de Resnik est raffinée par :*Lin* (***Sim***$_{Lin}$) [26], *Jiang et Conrath* (***Sim***$_{JC}$) [3], *Pirr et Euzenat* (***Sim***$_{Faith}$) [27] et *Mazandu et Mulder* (***Sim***$_{DIC}$) [28], en intégrant les différents IC des concepts comparés.

$$Simlin(u,v) = \frac{2.IC(MICAu,v)}{IC(u)+IC(v)}$$

$$Distjc(u,v) = IC(u)+IC(v)-2.IC(MICAu,v)$$

$$Simfaith(u,v) = \frac{IC(MICAu,v)}{IC(u)+IC(v)-IC(MICAu,v)}$$

$$SimDic(u,v) = \frac{2\sum_{c\in A(u)\cap A(v)} IC(c)}{\sum_{c\in A(u)} IC(c)+\sum_{c\in A(v)} IC(c)}$$

Notez que les mesures ci-dessus ne tiennent compte que le *MICA* pour estimer l'information partagée par deux concepts.

En effet, dans la plupart des cas, le *MICA* résume l'information contenue dans l'ensemble des ancêtres communs, comme il est subsumé par l'ensemble de cet ensemble. Toute fois, dans certains cas, dû à l'héritage multiple, la notion de *MICA* capture uniquement de façon partielle les informations partagées par les deux concepts.

Par exemple, dans la figure 3.1,

- considérant que *IC(C_4)>IC (C_2)*, le *MICA* de concepts C_5 ,C_6 est C_4.

Cependant, la quantité d'informations partagées par C_5 et C_6 est composée de la quantité d'informations transportées par {C_4, C_2}, leur *SLCAs*(Set of Least Common Ancestors).
Dans ce cas, les mesures basées sur *MICA* ne prendront en considération que le concept le plus informatif partagé par deux concepts.

L'élément clé des mesures ci-dessus est l'estimation exacte de l'IC des concepts. Afin d'éviter de baser les corpus annotés, dont la création prend du temps, et qui sont parfois difficiles à

obtenir (due à la sensibilité de données), différents modèles pour calculer l'*IC* ont été proposées.

Ils estiment l'IC du concept en considérant uniquement les informations structurales extraites de l'ontologie. Le calcul de l'IC peut être basé sur plusieurs caractéristiques topologiques telles que le nombre de descendants, le nombre d'ancêtres ou la profondeur de l'ontologie [29-31].

Seco et coll [30] proposent le calcul de l'IC d'un concept en fonction de son nombre de descendance :

$$ICseco = 1 - \frac{\log(D(u))}{\log(|c|)}$$

D(u)={v|v≤u} et *C* : l'ensemble des concepts définit dans l'ontologie.

Dans une autre approche, *Sánchez* et *coll.* [29] estiment l'IC d'un concept en fonction du rapport entre le nombre de concepts terminaux (feuilles) qu'il englobe et la quantité d'ancêtres qu'il a :

$$ICsanchez(u) = -\log\left(\frac{\frac{|leaves(u)|}{|A(u)|} + 1}{Max_{levaes} + 1}\right)$$

Avec *leaves(u)* comme étant le nombre de feuilles subsumé sous le concept u (exemple : *leaves* (C_2) = {C_5, C_6} dans figure 3.1) et *Max_Leaves* est le nombre de concepts terminaux dans l'ontologie.

III.2.3. Les approches Hybrides

Les approches hybrides combinent les notions des approches basées sur les arrêtes et les notions des approches basées sur les nœuds. Ils sont généralement définis comme des agrégations pondérées des ancêtres, des degrés de nœuds et des caractéristiques de concepts (par exemple IC) [32].

Conclusion

Dans ce chapitre, nous avons discuté sur la distance sémantique et les différents paradigmes utilisés pour calculer les mesures de similarité sémantique. Il y a plusieurs approches pour calculer la distance sémantique. Dans ce chapitre, nous avons parlé sur les approches à base d'arêtes, les approches à base de nœuds et en fin, nous avons parlé sur les approches Hybrides.

Chapitre 4

MODÉLISATION CONCEPTUELLE

Introduction

La découverte des *services web sémantique*(SWS_S) est le processus de trouver un service qui peut éventuellement satisfaire les besoins des utilisateurs, en choisissant entre plusieurs services. La compatibilité entre la demande (requête) de l'utilisateur et SWS_S est la tâche principale de tout mécanisme de découverte. Dans ce chapitre, un algorithme d'appariement basé sur la *distance sémantique* est introduit. L'idée derrière cet algorithme est d'utiliser la mesure de la *distance sémantique* entre la requête de l'utilisateur et le service testé comme un indicateur du degré de relevance entre eux. Un graphe de concepts est construit pour effectuer le processus de calcul de la *distance sémantiques* entre deux concepts.
La principale motivation pour l'utilisation d'une ontologie des concepts est le désir d'accélérer le processus de calcul de la *distance sémantique* entre deux concepts.

IV.1. Présentation de l'architecture

Afin d'assurer une modélisation cohérente de notre système, nous avons commencé par la création d'un certain nombre de modules où chacun d'eux assure des fonctionnalités distinctes, le regroupement de ces modules nous a permis par la suite de construire l'architecture générale du système. Ces modules sont en coopération et en collaboration entre eux où chaque module assure ses tâches et fournit ses sorties comme des entrées à un autre module.
La (figure 4.1) présente les quatre modules composants de notre architecture ainsi que la liaison entre eux. C'est l'architecture du système que nous proposons.
Le Premier module prend en charge l'extraction de l'information à partir d'un document *WSDL*, le deuxième module aussi prend en charge l'extraction de l'information à partir des fichiers généré par « *protège* ». Le troisième module s'intéresse essentiellement à la mesure de la distance entre deux concepts. Et finalement le quatrième module assure la gestion de l'affichage et la présentation des résultats.

Nous proposons par la suite l'architecture générale de notre système pour expliquer clairement le déroulement de notre système de découverte, à partir de l'entrée de la requête de l'utilisateur jusqu'à la récupération des services web qui satisfont les besoins.

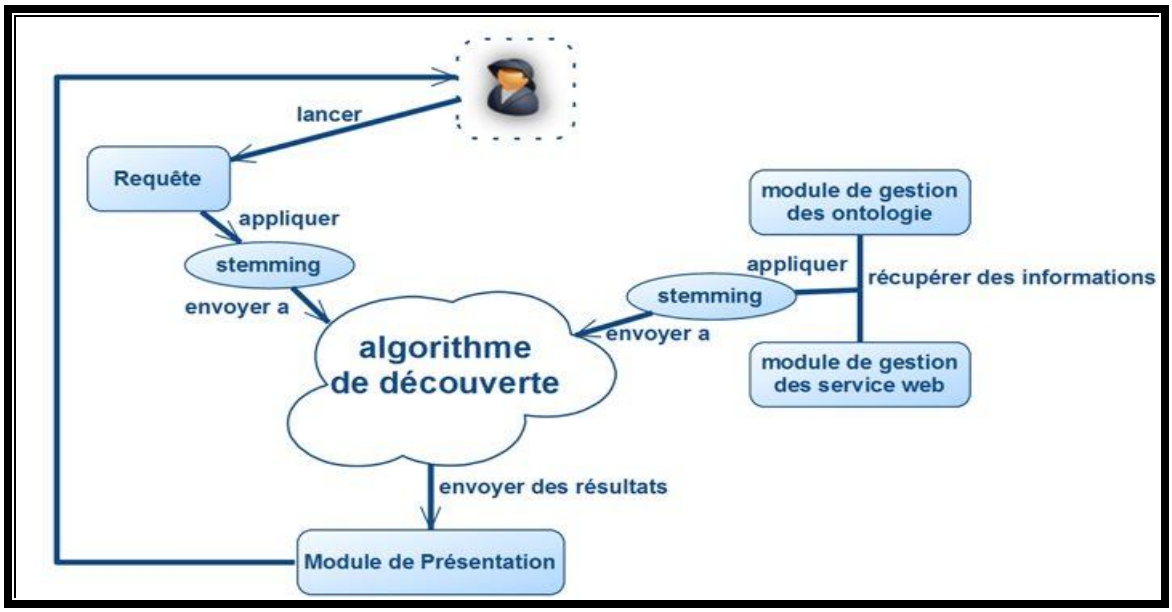


Figure 4.1: L'architecture du système.

VI.1.1. Stemming des mots

Dans cette étape, tous les mots de Ti sont réduits à leurs mots de base en utilisant l'algorithme de Portier. Les mots avec même origine auront généralement le même sens, par exemple «pay » « paid »« paying » et « payment » tous sont de même origine « pay ».

VI.1.2. Module de gestion des services web (parseur des SWSs)

Nous traitons le contenu du document WSDL (figure 4.2) pour extraire le vecteur de mots significatifs pour chaque Service Web. La construction de ce vecteur est composée à partir des trois étapes suivantes:

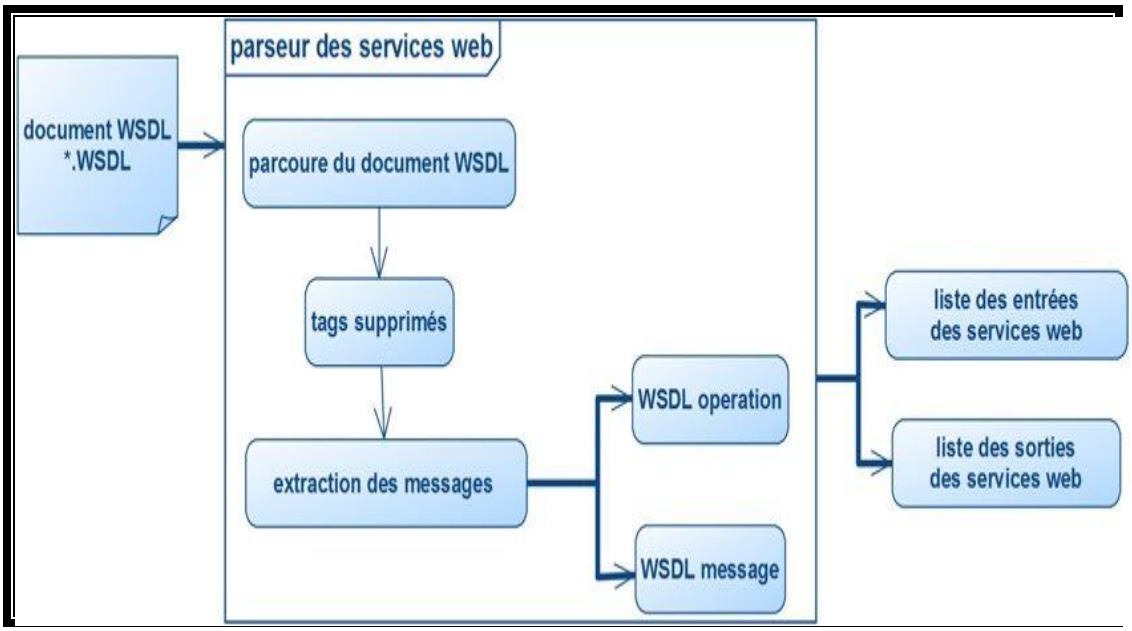


Figure 4.2: Module de gestion des services web.

a- Parcoure du document WSDL

C'est extraire les informations utiles : cette approche consiste, grâce à des règles, à identifier et extraire les termes pertinents à partir des descriptions textuelles disponibles pour un service web.

Le parseur segmente une suite de caractères en plus petits éléments pour produire un vecteur de mots Si. Il peut être utilisé pour reconnaître les mots clés et les informations pertinentes.

b- Suppression des Tag

C'est supprimer tous les symboles de Si qui font partie des balises XML. Les mots valides restent dans le vecteur Si.

c- Extraction des messages d'un document WSDL

Le nom de l'opération, les messages d'entrées et les messages de sorties sont les caractéristiques sélectionnées. Le nom de l'opération se trouve dans l'élément principal<port type> comme montré dans la (figure 4.3).

Exemple : le nom de l'opération get_PRICE

```
25    <wsdl:message name="get_PRICEResponse">
26        <wsdl:part name="_PRICE" type="tns:PriceType">
27    </wsdl:part>
28    </wsdl:message>
29    <wsdl:message name="get_PRICERequest">
30        <wsdl:part name="_USER" type="tns:UserType">
31    </wsdl:part>
32    </wsdl:message>
33    <wsdl:portType name="UserPriceSoap">
34        <wsdl:operation name="get_PRICE">
35            <wsdl:input message="tns:get_PRICERequest">
36    </wsdl:input>
37            <wsdl:output message="tns:get_PRICEResponse">
38    </wsdl:output>
39        </wsdl:operation>
40    </wsdl:portType>
```

Figure 4.3: documents WSDL.

i. WSDL <messages>

Cet élément dans les documents WSDL décrit les noms et les types des messages échangés par les services.

Exemple : le nom et le type de message :

nom="_PRICE" type="tns:PriceType"

nom="_USER" type="tns:UserType"

La définition du type se trouve aussi dans le même document WSDL en utilisant la balise type (il peut être entier, réel, … ou même des types complexes).

ii. WSDL <opération>

Le format de messages échangés est :

- **Format Request-response (Demande-réponse):** le service reçoit un message de demande (entrée) et renvoie un message de sortie. En WSDL, il existe une séquence d'entrée et une séquence de sortie.

Exemple: L'entrée du message="tns:get_PRICERequest"

La sortie du message="tns:get_PRICEResponse"

d- Algorithme de « *parseur WSDL* »

Algorithme:parseur de document WSDL
Entrée :
Liste des Fichiers WSDL
Sortie :
Liste des informations de chaque fichier (nom_opération,entrée_service,sortie_service)
Terminologie :
Fichier :les fichier WSDL
Ligne : tampon de chaque ligne(balise) de fichier WSDL
Ligne1 :ligne.contains("get_XXXXRequest")+1(la ligne1 est la ligne qui suit la ligne qui contient "get_XXXXRequest")
Ligne2 : ligne.contains("get_XXXXResponse")+1(la ligne2 est la ligne qui suit la ligne qui contient "get_XXXXResponse")
Ligne3 :ligne.contains("message")
Input :dans la « la ligne1 » après le mot « name »
output:dans la « la ligne2 » après le mot « name »
output:dans la « la ligne3 » après le mot « name »
servicelist_service = new service();
Debut

```
For(i ;i<Fichier.size() ;i++)
        si (ligne==ligne1) then
        ajouter(list_service,input) ;
        fin si
        si(ligne==ligne2)then
        ajouter(list_service,output);
        fin si
            si (ligne==ligne3)then
        ajouter(list_service,nom_operation) ;
        fin si
ajouter(SW,list_service);
End for
End
```

VI.1.3. Module de gestion des ontologies

Dans notre système l'ontologie a été utilisée pour la projection sémantique de la requête et les services web (les entrées et les sorties) afin d'extraire les concepts relatifs à chaque mot (figure 4.4).

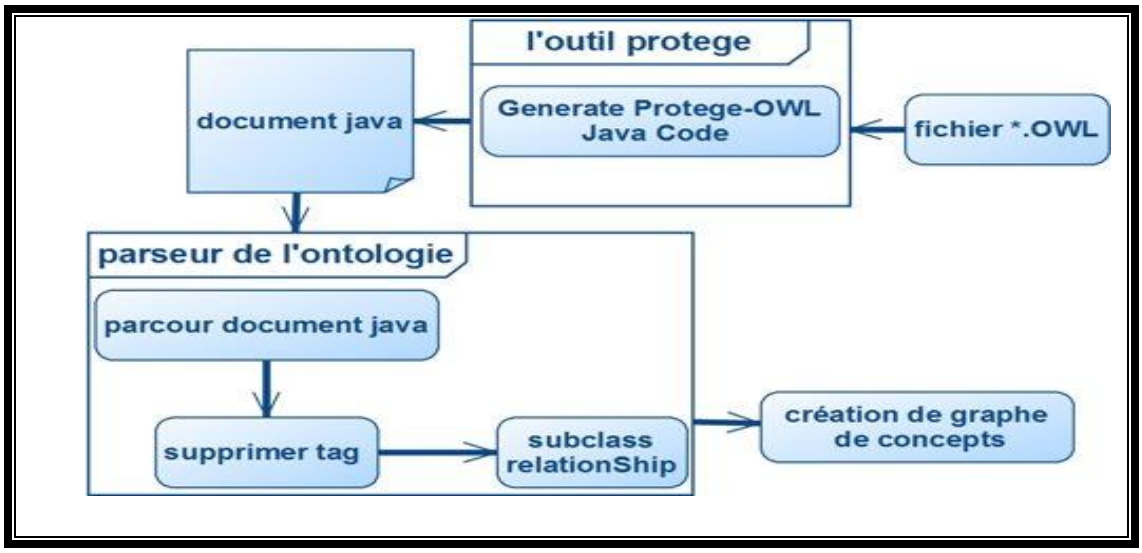


Figure 4.4: Module de gestion des ontologies

a- Generate Protege-OWL Java Code :

L'outil « Protège » est assez standard dans la création et l'édition des ontologies. Il peut être utilisé comme générateur de classes Java, selon le cas d'utilisation. Les résultats de cette étape sont des fichiers générés par « Protège » qui vont nous aider à extraire les informations de l'ontologie (c'est-à-dire au lieu de parser l'ontologie directement, ce qui est difficile, nous allons parser des fichiers générés par protège qui prend en entrée le fichier source de l'ontologie).

Après la génération, les fichiers résultats sont envoyés à un module d'extraction de méta données.

b- Parseur de fichiers générés par Protège

L'objectif de cette étape est d'extraire les informations utiles dans ces fichiers. cette approche consiste, grâce à des règles, à identifier, extraire les termes pertinents à partir des descriptions textuelles disponibles pour un fichier source.

Le parseur segmente une suite de caractères en plus petits éléments pour produire un vecteur des mots Ti. Il peut être utilisé pour reconnaître les concepts de l'ontologie.

c- Suppression des Tag

Cette étape supprime tous les symboles de Ti qui font partie d'un mot de programme en langage Java (les fichiers générés pas Protège sont des classes Java). Les mots valides restent dans le vecteur Ti.

d- Subclass Relationship (subsomption)

Les mots valides dans le vecteur Ti sont le concept père (classe) et le concept fils (sous classe).

Le concept C_i est une sous-classe de concept C_j si $C_i \in LSC(C_j)$, où $LSC(C_j)$ est l'ensemble des concepts moins spécifiques(fils directs) de C_j (figure 4.5) .les sous classes peuvent avoir plusieurs classes parentes.

Exemple : le concept père est *wrappedIndividual*,et le concept fils est *DefaultActor*

```
1  package ActorDefault;
2
3  import java.util.Collection;
4
5  import org.protege.owl.codegeneration.WrappedIndividual;
6
7  import org.semanticweb.owlapi.model.OWLNamedIndividual;
8  import org.semanticweb.owlapi.model.OWLOntology;
9
10 /**
11  *
12  * <p>
13  * Generated by Protege (http://protege.stanford.edu). <br>
14  * Source Class: DefaultActor <br>
15  * @version generated on Wed Apr 02 06:49:32 GMT+01:00 2014 by Lenovo
16  */
17
18 public interface DefaultActor extends WrappedIndividual {
19
20     /* ***************************************************
21      * Property http://www.daml.org/services/owl-s/1.1/ActorDefault.owl#email
22      */
23
24     /**
25      * Gets all property values for the email property.<p>
26      *
```

Figure 4.5 : représente fichier de code Java.

e- Graphe des concepts (l'ontologie)

L'idée de cette étape est de générer automatiquement des ontologies pour utiliser la sémantique des termes. Pour préparer le graphe de l'ontologie nous commençons par l'extraction des différents concepts à partir des fichiers générés par « protège».
L'aspect le plus important de la construction de l'ontologie est d'identifier les concepts sémantiquement significatifs ou il existe des relations entre eux.

L'architecture de l'ontologie permet de définir un sous-concept (sous-classe ou fils) avec n'importe quel nombre de concepts pères. En d'autres termes, les concepts de l'ontologie supportent le multi-héritage.

Pour simplifier la recherche dans l'ontologie, nous utilisons « *le graphe de concepts* ». La (figure 4.6) présente un exemple de ce graphe

Chaque nœud du graphe est appelé « *nœud concept* ». Le nœud concept, c'est un vecteur contient des informations sur le concept; tel que le nom du concept, l'id du concept,id du concept parent et deux listes : une des nœuds fils et une des nœuds parents. Il est important de mentionner que chaque concept nœud possède une propriété importante appelée niveau de réflexion qui fait référence au niveau d'un concept dans la hiérarchie de l'ontologie. Par exemple, le concept nœud « K » dans la (figure 4.6) contient les informations suivantes:

- nom du concept="k",
- l'id du concept ="8",
- l'id du parent ="6",
- niveau du concept="4",
- la liste des pères="C, A et ROOT",
- la liste des pères= "M, Net P".

Il existe un nœud concept spécifique dans chaque graphe des concepts appelé « nœud racine ». Le nœud racine est un nœud de concept sans parent, le niveau du concept est «1». Le « *graphNodesFullList »* est un vecteur des nœuds concept dont les clés sont les noms des concepts et une liste complète des nœuds parents comme représentés dans la (figure 4.6).

Il est interdit d'avoir la répétition dans cette liste. L'existence de « *graphNodesFullList »* nous aide à la recherche dans le graphe.

Pour surmonter le multi-héritage, nous proposons la liste père dans les nœuds concept. La liste père résoudre le problème de la présence de plusieurs pères pour un fils. L'exemple suivant explique comment la liste père fonctionne : si nous avons un concept "H" avec

Id=" 15"et il est un sous concept de deux autres concepts appelé "G et X" avec ID="7,12" respectivement; le concept" H" apparaissent dans le graphe des concepts comme deux nœuds séparément, chacun d'eux dispose le nom="H" et ID =" 15". Ces deux nœuds concept ont des parents différents, mais ils sont liés par la liste père.

Comme la (figure 4.6), chaque nœud des deux nœuds concept "H" a une liste père associée. Il est important de mentionner que le concept "H" apparaît une seule fois dans le « *graphNodesFullList* ».

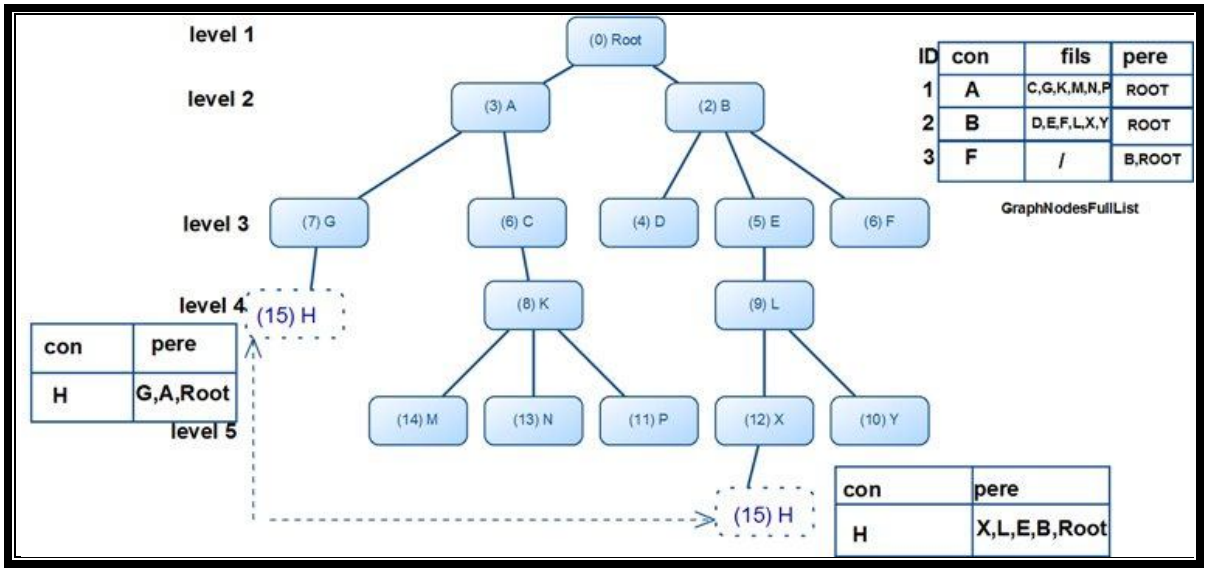


Figure 4.6:Exemple des concepts graphe

f- **Algorithme de parseur du l'ontologie**

Le processus de création est basé sur l'information et les données disponibles dans le vecteur Ti. Le processus commence par identifier le nom du nœud racine. Puis les nœuds sous-concepts sont créés en utilisant la liste des fils. Cette fonction est une fonction récursive, qui est utilisée pour créer le graphe des concepts en explorant tous les sous-concepts disponibles pour chaque concept dans le graphe et le processus s'arrête quand il atteint les concepts feuilles.

Algorithme :création de parseur de l'ontologie
Entrée :
Liste des fichiers générés par protège
Sortie :
graphe des Concepts(Graph_Onto) de chaque ontologie avec ses propriétés
Terminologie :
Dossier_fichier: les dossier qui sont résultats de l'extraction des ontologies par le logiciel protège
Fichier :sont les fichiers Java de chaque Dossier_fichier
Ligne : tampon de chaque ligne de fichier Java
Concept_père:dans la « ligne » après le mot "extends"
Concept_fils:dans la « ligne » après le mot "interface"
Liste_Pére : ajouter à la liste Liste_Pére les concept-père de chaque concept_fils
Liste_fils: ajouter à la liste Liste_fils les concept-fils de chaque concept_père
Level : Niveau de Chaque concept (Concept_père ou Concept_fils) dans l'ontologie
Profondeur: le dernier Niveau de l'ontologie
List_Onto: liste des concepts (Concept_père ou Concept_fils) d'une Ontologie
Initialiser List_Onto;
Graph_Onto: le Graphe de l'ontologie
Debut
pour(chaque Dossier de Dossier_fichier){
pour(chaque fichier de Fichier){
Graph_Onto.addNoeud(Concepte_pére) ;
Graph_Onto.addArc(Concepte_fils, Concepte_pére);
pour (chaque concept de la List_Onto)
si (List_Onto.get[i].name == Concepte_pére) then
ajouter(Liste_pére,List_Onto.get[i].Liste_Pére);
level = Liste_pére.length;
fin si
fin pour

Concept newConcepte = Concept();
newConcept.name = Concepte_pére;
newConcept.level = level ;
newConcept.liste_pére = Liste_pére;
newConcept.liste_fils = Liste_fils;
List_Onto.add(newConcept);
fin si
fin pour
fin pour
Fin

VI.1.4. Algorithme de découverte

La distance sémantique entre les termes (la similarité sémantique) est utilisée pour la première fois dans les études lexicales pour désigner le degré de liaison sémantique et de la similitude entre les mots.

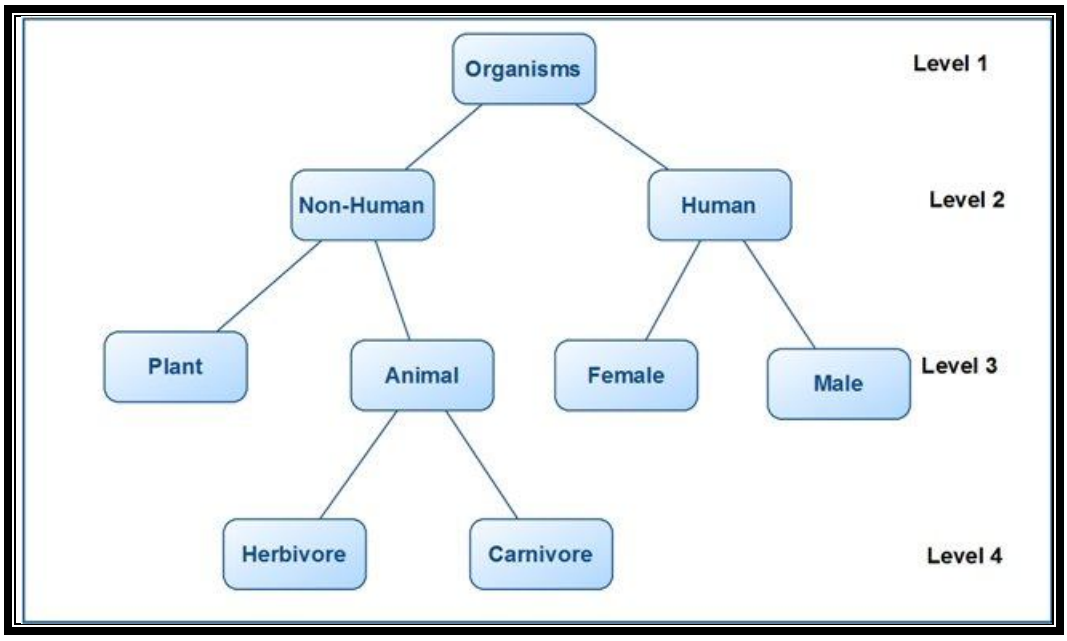


Figure 4.7:représente un exemple d'un graphe de concepts.

Il est très important de faire la distinction entre la distance sémantique et la longueur du chemin entre deux concepts. Pour illustrer la différence, La (figure 4.7) montre une partie d'un réseau sémantique, qui illustre la relation entre certain concepts.

La longueur du chemin«Carnivore » et « Plant » (deux niveaux haut et un niveau bas) est égale à la longueur du chemin entre « Carnivores » et « Organisms » (trois niveaux haut). D'autre part, la distance sémantique entre « Carnivores » et « Organisms »semble être beaucoup plus proche que la distance sémantique entre « Carnivores » et « Plant ».

La phrase « Carnivores est un Organisms » est correcte, contrairement à la phrase « Carnivores est une Plant ». Mais il ne fait aucun doute que l'information du chemin sera la base pour évaluer la distance sémantique entre deux concepts dans un réseau sémantique.

a- La formule de Tamer A. Farray,AhmedI.Salah et H.Ali pour le calcul de la distance sémantique

De nouveaux algorithmes sont proposés pour mesurer la distance sémantique entre la demande (requête) de l'utilisateur et le service testé. Dans cette étape on explique brièvement la formule que nous avons utilisée.

De nombreuses expériences sont conçues pour démontrer le grand impact d'utiliser les mesures de distance sémantique dans le domaine des SWS_S compatibles.

La formule suivante de [18] illustre comment être capable de mesurer la distance sémantique entre deux concepts C_1 et C_2.

$$SDM(C1, C2) = \begin{cases} \dfrac{MD - (LWc1 + LWc2) * PL - D}{MD} \% \text{ pour C1 ; C2 liés} \\ Zero \qquad \text{pour C1, C2 non liés} \end{cases}$$

$$LW = \frac{MD - Concept\ level + 1}{MD}$$

Avec :

$PL =$ Représente le nombre d'arcs comptés entre deux concepts C_1 et C_2 dans l'ontologie.

$MD =$ Représente la profondeur maximale des concepts dans l'ontologie, MD est utilisé pour éviter les valeurs négatives dans SDM.

$CL =$ Représente Une fonction qui nous donne le niveau du concept.

$D =$ Représente Le nombre d'arcs descendants entre deux concepts C_1 et C_2.

La valeur de SDM est égale à zéro s'il n'y a pas de relations(pas de chemin) entre les deux concepts(en d'autres termes, si ils n'appartiennent pas à la même ontologie).

Si C_1et C_2 représentent le même concept, la valeur de SDM est égale à 100% (où PL= 0 et

D =0)

b- La correspondance entre les SWSs

Pour faire une correspondance entre une demande de service (La requête R) et un service web disponible (S), il y a quatre listes de concepts comme montré dans la (figure 4.8) Ces quatre listes comprennent sont:

- liste des entrées de R concepts (R_I),
- la liste des entrées S concepts (S_I),
- liste des sorties de R concepts (R_O),
- et la liste des sorties S concepts (S_O).

Ensuite,une méthodologie générale pour faire une correspondance entre deux listes de concepts sera introduite. Dont l'objectif principal est d'obtenir la distance sémantique maximale.

Les valeurs de les distances sémantique des entrées et des sorties sera utilisé pour obtenir la mesure finale de la relation sémantique entre la requête R et le serviceS.

- On applique la formule suivant :

$$\sum_{x=0}^{m} \sum_{y=0}^{n} \frac{\max \_D(R\_I_x, S\_I_y)}{m + n}$$

- On choisie par une seuillage entre R_I et S_I les distances parfais (la meilleur distance entre les entrées de la requête R et les entrées de service S),
- Le même travail avec R_O et S_O (les sortis de la requête R et le sorties de service S).

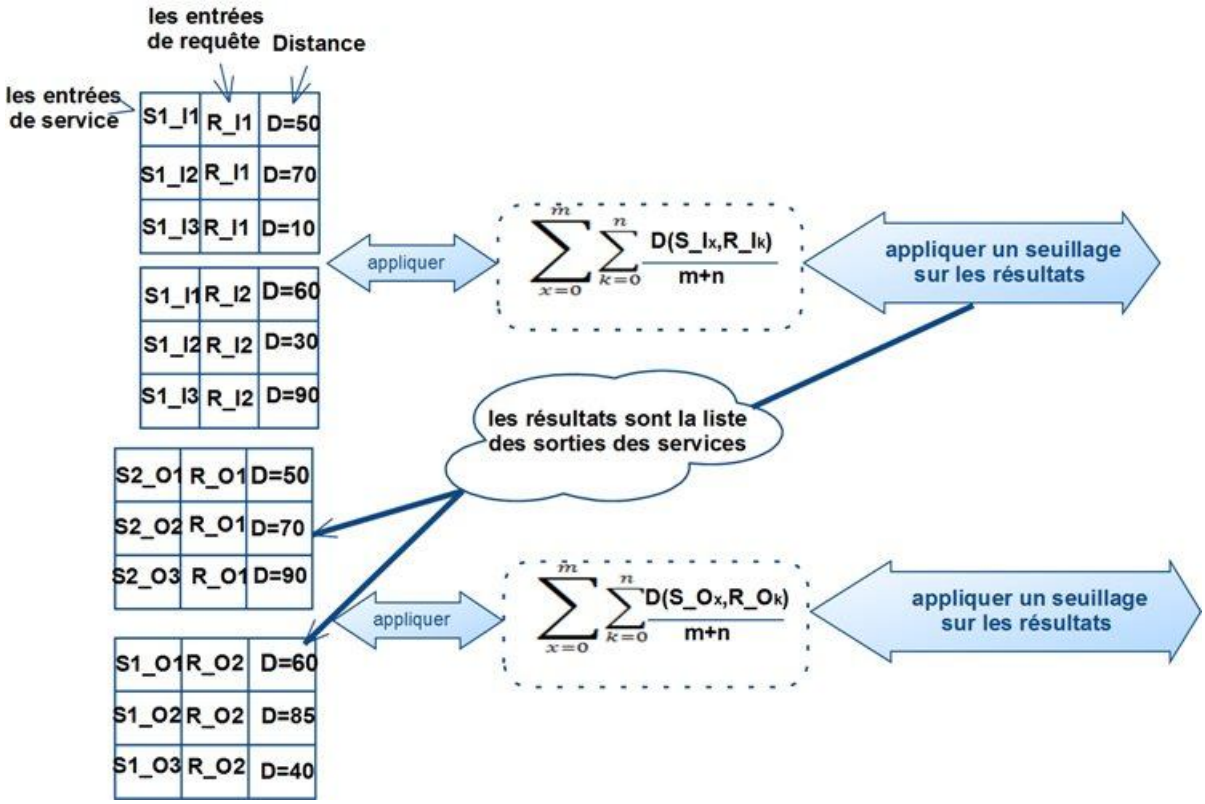


Figure 4.8: Exemple d'une Correspondance entre les SWS_S

c- Algorithme de découverte

Algorithme de découverte;
Entrée = les entrées et les sorties de la Requête R(R_input,R_output);
Sortie = la liste des services web satisfaisants;
Terminologie =
m=nombre de concept de la requête
n=nombre des entrées de chaque service
Seuil=x ;
Stemming:l'origine du mot
list_distance:liste contient les distance de chaque concept avec les autres concepts concernés
SDM:la distance sémantique
nbr_arcs:nombre d'arcs dans le chemin entre les deux concepts
nbr_arcs_descendant:nombre d'arcs entre le 1ier père commun jusqu'à le 2ieme concept
Liste 1>liste2>liste3
Liste1 : un ensemble de services
Somme=0
debut
saisir la Requête R;
Démarrer l'algorithme de parseur de fichiers WSDL
Démarrer l'algorithme de création de graphes de concepts;
Appliquer le stemming sur les concepts de la Requête et les concepts de service web et le concepts de l'ontologie;
Exécuter la fonction traitement des entrées
Exécuter la fonction traitement des sorties

```
Liste  fonction  traitement_des_entrées (liste_Input_R, liste1);
  Pour (chaque service S dans la liste1)
        SDM←Calculer distance entre deux listes (entrés de R, entrés de S)
        Si SDM > seuil alors
           Ajouter (S à liste2)
fin si
fin
Fin

entier fonction Calculer distance entre deux listes (liste_concepts1, liste_concepts2 )
       // la construction de la liste des distances
pour (chaque concept de liste_concepts1)
pour (chaque concept de liste_concepts2)
             d<-- calculer distance entre deux concepts(liste_concepts1[i], liste_concepts2[j
ajouter (d, liste de distances)
fin pour
fin pour
       // le calcule de la distance finale a partir de la liste des distances
Sdm<-- calculer distance finale (liste des distances)
retourner SDM
Fin
entier calculer distance entre deux concepts (R_input,S_input)
SDM(R_input,S_input)=(profondeur-(LW R_input+LWS_input)* nbr_arcs(R_input,S_input
 -nbr_arcs_descendant)/ profondeur ;
  LW= profondeur –level+1 / profondeur ;
Retourner d
Fin
Entier calculer distance finale (liste_dist)
//Calculer la distance moyenne de tous les distances de la liste
pour (chaque distance de liste_dist)
      somme=somme+(liste_dist[i])
fin pour
SDM_input=somme/m+n ;
Retourner SDM_input
Fin
```

```
Liste  fonction  traitement_des_sortie (liste_output_R, liste2);
  Pour (chaque service S dans la liste2)
        SDM←Calculer distance entre deux listes (sortie de R, sortie de S)
        Si SDM > seuil alors
           Ajouter (S à liste3)
fin si
fin
Fin

entier fonction Calculer distance entre deux listes (liste_concepts1, liste_concepts2 )
       // la construction de la liste des distances
pour (chaque concept de liste_concepts1)
pour (chaque concept de liste_concepts2)
              d<-- calculer distance entre deux concepts(liste_concepts1[i], liste_concepts2[j])
ajouter (d, liste de distances)
fin pour
fin pour
       // le calcule de la distance finale a partir de la liste des distances
Sdm<-- calculer distance finale (liste des distances)
retourner SDM
Fin
entier calculer distance entre deux concepts (R_input,S_input)
SDM(R_input,S_input)=(profondeur-(LW R_input+LWS_input)* nbr_arcs(R_input,S_input)
 -nbr_arcs_descendant)/ profondeur ;
  LW= profondeur –level+1 / profondeur ;
Retourner d
Fin
entiercalculer distance finale (liste_dist)
//Calculer la distance moyenne de tous les distances de la liste
pour (chaque distance de liste_dist)
      somme=somme+(liste_dist[i])
fin pour
SDM_output=somme/m+n ;
Retourner SDM_output
Fin
```

VI.1.5. Module de présentation

Le module de présentation prend en charge le coté affichage des résultats dans notre système, il présente à l'utilisateur les résultats avec un classement accompagnés par des informations sur le sens exprimé dans la page afin de lui permettre de cerner plus facilement le contenu des documents qui lui sont retournés.

La(figure 4.9) le principe de fonctionnement du module de présentation :

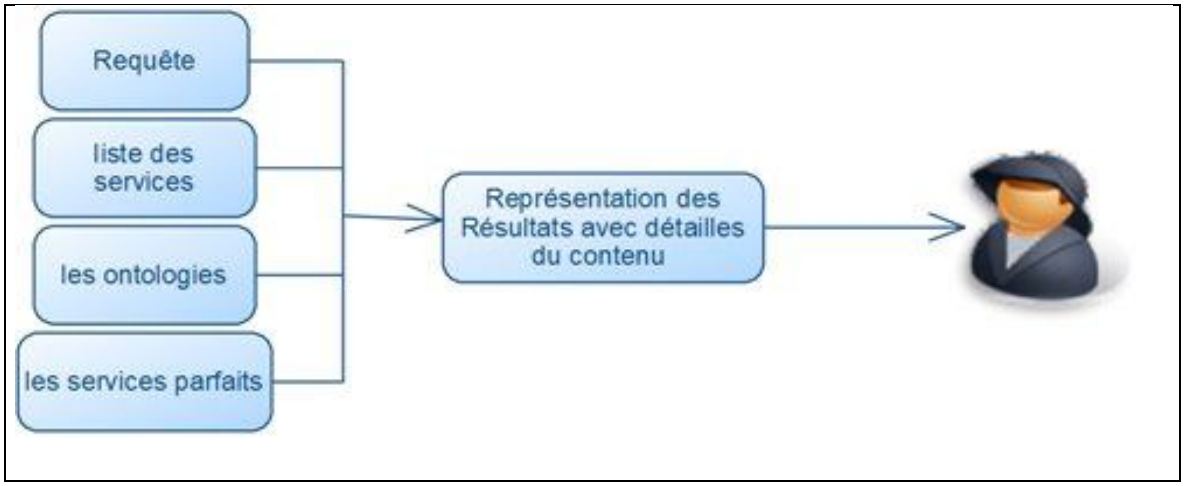


Figure 4.9 : Module de présentation.

IV.2. Architecture détaillée

Une fois les différents modules ont été présentés, leur regroupement nous a permet de dégager l'architecture détaillé, elle prend en entrée la requête utilisateur et elle produit à la fin un classement sémantique des résultats retournée par le découverte.

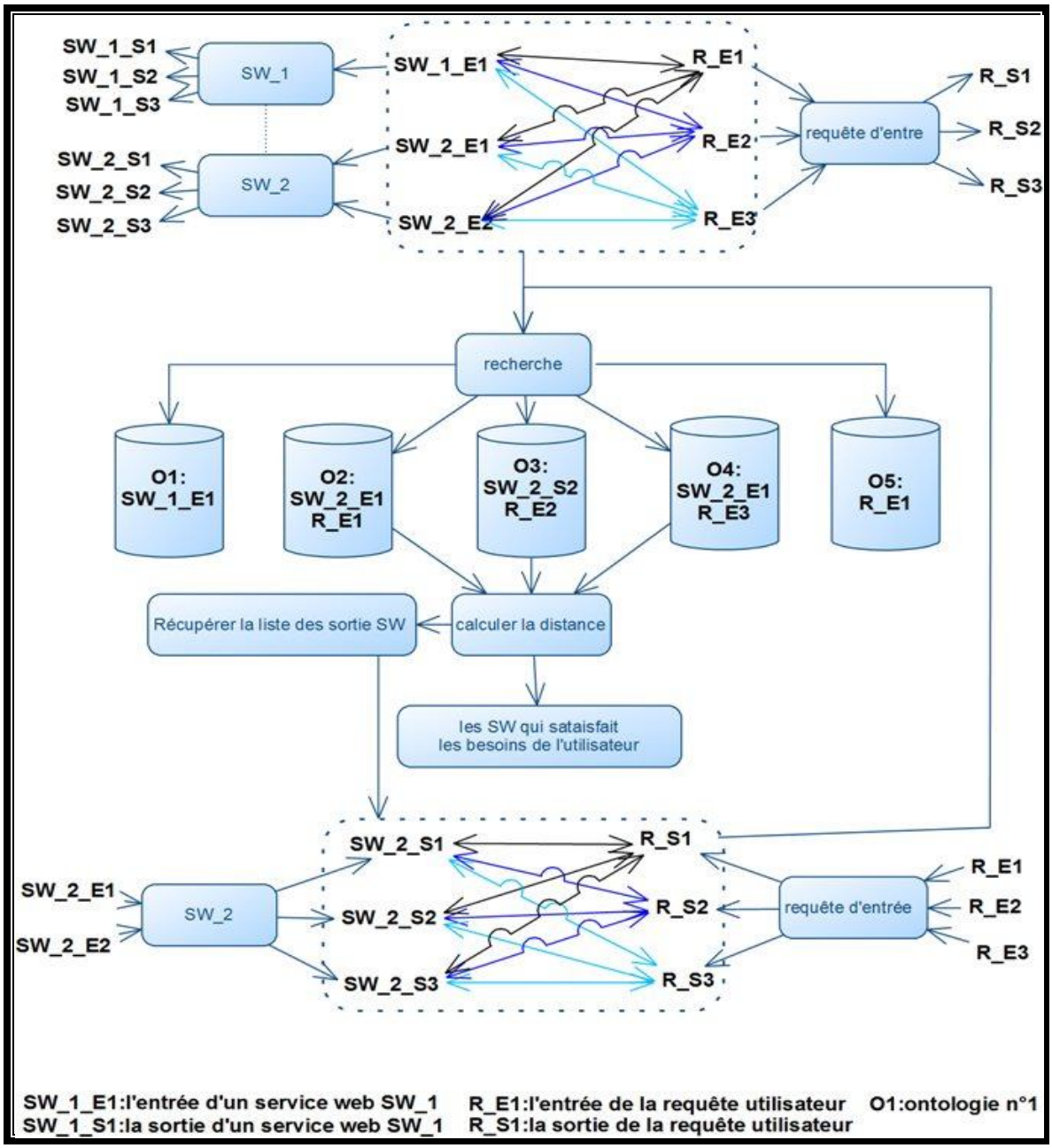


Figure 4.10: Architecture détaillée.

IV.3. Exemple

Etape1 : l'utilisateur saisit la requête (l'entrée :Hiking et la sortie :destination)

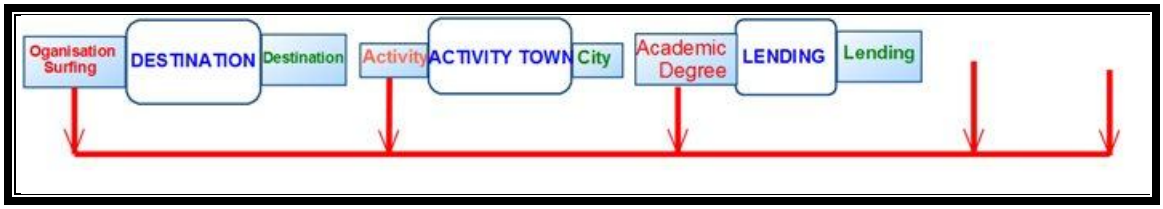
Appliquer la fonction stemming sur les entrées de la requête (le résultat est Hike).

Etape2 : les données de cette étape sont l'entrée de la requête « Hike » et les entrées des services web.

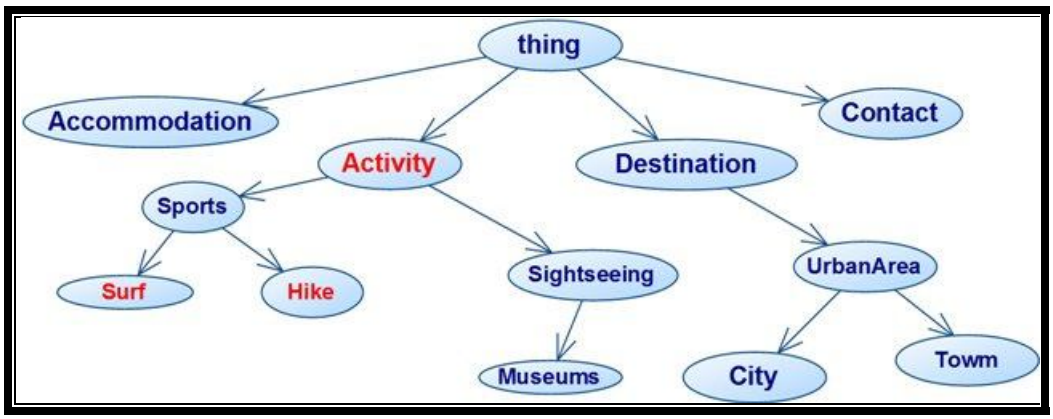
Soient les services suivants :

- les entrées du service1 « Destination » sont « organisation et surfing »
- l'entrée de service2 « ActivityTown » est « activity »
- les entrées de service3 « Lending » sont « Academic et Degree »

Appliquer la fonction stemming sur les entrées de tous les services web (le résultat est organisation et surf et activity et Academic et Degree) et sur tous les concepts des ontologies.



La recherche sur tous les ontologies à ces données résulte une seul ontologie appelé « travel » contient « Hike, surf et activity ».



Etape3 : l'ontologie contient les données (« Hike » et « surf et activity ») donc on va calculer la distance entre elles .

Pour calculer la distance, on utilise la formule expliquée précédemment suivante :

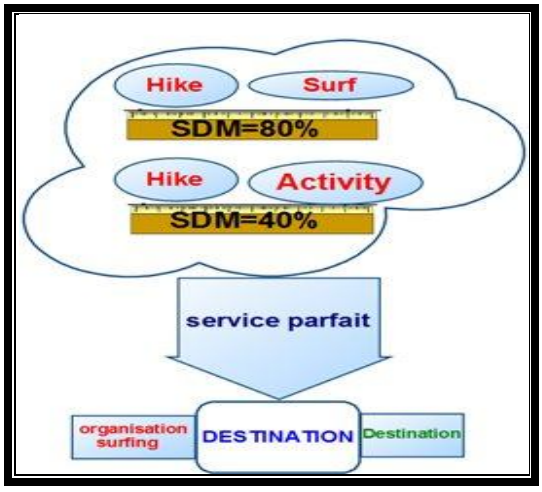
$$SDM(c1,c2) = \begin{cases} \dfrac{MD - (LWc1 + LWc2) * PL - D}{MD} \% \\ Zero \end{cases}$$

$$LW = \frac{MD - Concept\ level + 1}{MD}$$

Dans ce cas les distances calculées entre « Hike et surf »et entre « Hike et activity » sont :

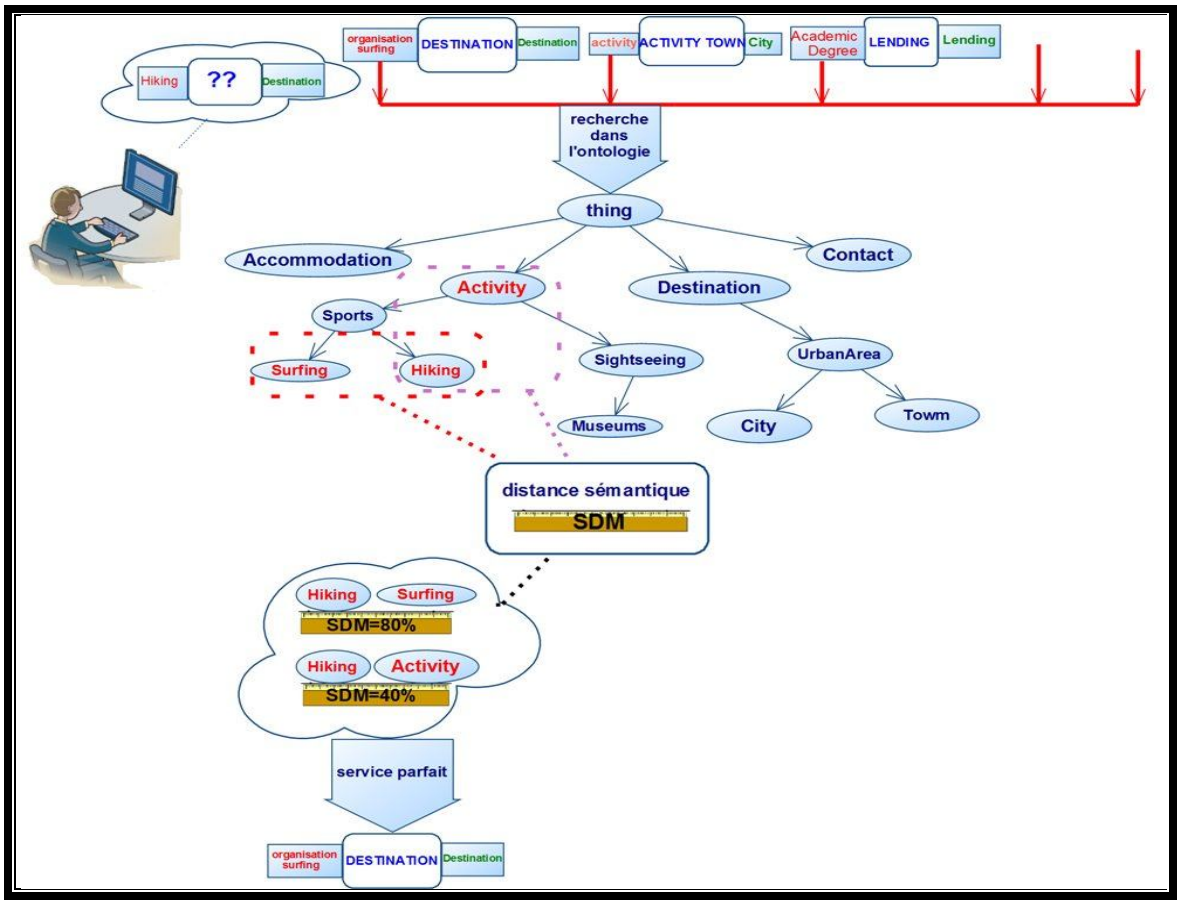
- Entre « Hike et Surf » est 80%
- Entre « Hike et Activity » est 40%

Donc la meilleure distance est 80% entre « Hike et Surf »



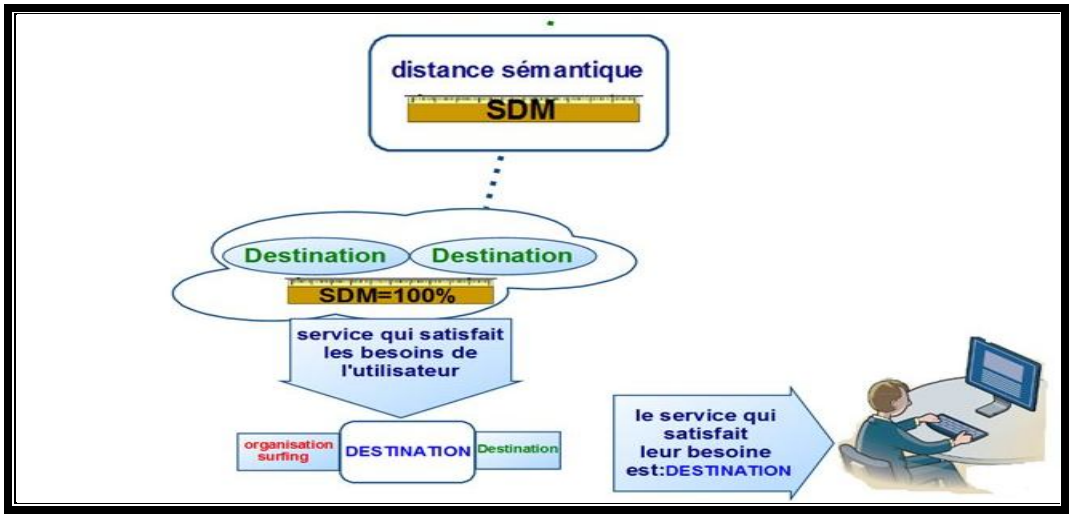
Puis récupérer les sorties du service DESTINATION qui ont la meilleure distance avec les entrées de cette requête « Destination » et lancer les étapes précédente une deuxième fois par

le service web « Destination » mais cette fois sur les sorties des services avec la sortie de la requête.



Etape4: Retourne à l'utilisateur le meilleur service qui satisfait ses besoins.

Dans ce cas le service qui satisfait les besoins de l'utilisateur est :DESTINATION



Conclusion

Dans ce chapitre, nous avons introduit les modèles supports de l'approche que nous avons utilisés pour une meilleure découverte des services web. Notre modèle est basé sur la *distance sémantique* et amélioré par l'invocation des concepts ontologiques.

En utilisant un module de *gestion des services web* qui repose sur le parsing et le traitement du contenu d'un document WSDL afin d'extraire le vecteur de mots significatifs pour chaque service Web. Et un Module de *gestion des ontologies*, où la construction de l'ontologie de concepts est nécessaire pour améliorer les résultats de la découverte en intégrant l'aspect sémantique. Et nous avons terminé par le module de la découverte des services basés sur le calcul de *la distance sémantique*.

Dans le chapitre suivant nous allons présenter l'implémentation de ce modèle et aussi expliquer les différentes interfaces réalisées.

Chapitre 5

IMPLÉMENTATION

Introduction

Après avoir achevé la conception de notre application, nous allons entamer la partie réalisation. Dans le système que nous avons implémenté la recherche sémantique des servies est guidée par des ontologies. L'utilisation de notre application permettra à l'utilisateur d'accéder à des informations sémantiques, qui sont les résultats de découverte des services web à base de la distance sémantique.

Nous concentrons notre effort dans ce chapitre sur la présentation de l'implémentation de notre application en se basant sur les quatre (4) modules présentés dans le chapitre précédent. Ce chapitre est organisé de la façon suivante : nous commençons tout d'abord par une présentation des environnements utilisés pour le développement de notre système, à savoir : le langage de programmation, les outils utilisés pour l'implémentation, et les bibliothèques utilisées. Nous présentons par la suite la description de l'application en décrivant sa structure.

V.1. Environnements utilisés pour le développement

Dans cette sectionnous présentons les principes d'implémentation de notre application, pour cela nous définissons les différents outils de programmation que nous avons utilisés, afin d'atteindre le but de la recherche sémantique des services web.L'environnement de développement est constitué de :

- Le langage de programmation java.
- L'environnement de programmation NeatBeans.
- Les *ontologies*.
- Les *services web.*

V.1.1. Langage de programmation

Nous avons choisis JAVA comme langage de programmation pour l'implémentation de notre application. JAVA a été mise au point en 1991 par les laboratoires Sun Microsystems.

De plus nous avons choisi ce langage pour le bon fonctionnement ainsi que sa qualité connue, à savoir :

- C'est un langage orienté objet simple

- Java est indépendante de toute plate-forme : grâce à sa machine virtuelle ***« Java Virtuelle Machine JVM ».***
- Il fournit aux programmeurs de nombreuses bibliothèques pour la gestion des interfaces et des programmes et il permet la gestion des exceptions.
- Java est un langage "multi threade" : Le **"thread"** (processus léger) est unepartie de code, un "flot d'instructions" s'exécutant en concurrence avec d'autres threads dans un même processus, cela permet à un seul programme d'effectuer plusieurs activités simultanément.
- Le JDK (*Java Development Kit*), regroupe l'ensemble des éléments permettant le développement, la mise au point et l'exécution des programmes Java. Le JDK et la documentation sont librement téléchargeables sur le site web de Sun

V.1.2. Environment de programmation « NetBeans »

NetBeans est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web).

NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS.

V.1.3. les ontologies

L'objectif principal de notre travail consiste à prendre en compte la sémantique en se basant sur l'ontologie dans un système de recherche d'information. Le système utilise un ensemble des ontologies.

V.1.4. les services web (WSDL)

C'est un ensemble de documents WSDL où chaque document est un service web, ces services sont utilisés pour le test du bon déroulement de notre système.

V.2.Implémentation

Le système que nous avons développé est un système de recherche sémantique des services web basé sur le calcul des distances sémantiques en s'appuyant sur les ontologies, nous l'avons appelé «découverte». Découvertefait une correspondance entre ce qui est retournés par l'extraction des documents WSDL par rapport à la requête utilisateur. Ces résultats sont obtenus par une projection sémantique de la requête sur l'ontologie. Par la suite le système procède à un classement de ces réponses par ordre de pertinence. Dans ce qui suit nous présentons et nous décrivons les fonctionnalités offertes par découverte.

V.2.1. Presentation de l'application découverte

Au lancement de l'application découverte, l'interface de l'utilisateur s'affiche, il s'agit d'une interface simple d'utilisation pour permettre à l'utilisateur d'effectuer des découvertes sémantique. La figure 5.1 présente la fenêtre de l'utilisateur :

Figure 5. 1 :l'interface utilisateur

L'interface utilisateur se compose d'une zone :

- ***Zone de recherche*** : elle permet à l'utilisateur d'exprimer son besoin en services sous forme d'une requête et de faire passer la requête à « découverte ».

L'interface principale et l'interface ontologie/services_web s'affiche en cliquant sur le bouton « Découverte ».

L'interface principale se compose d'une zone comme la figure 5.2:

- ***Zone de résultats :*** c'est le volet dans lequel le système affiche les résultats rendus par le « découverte ».

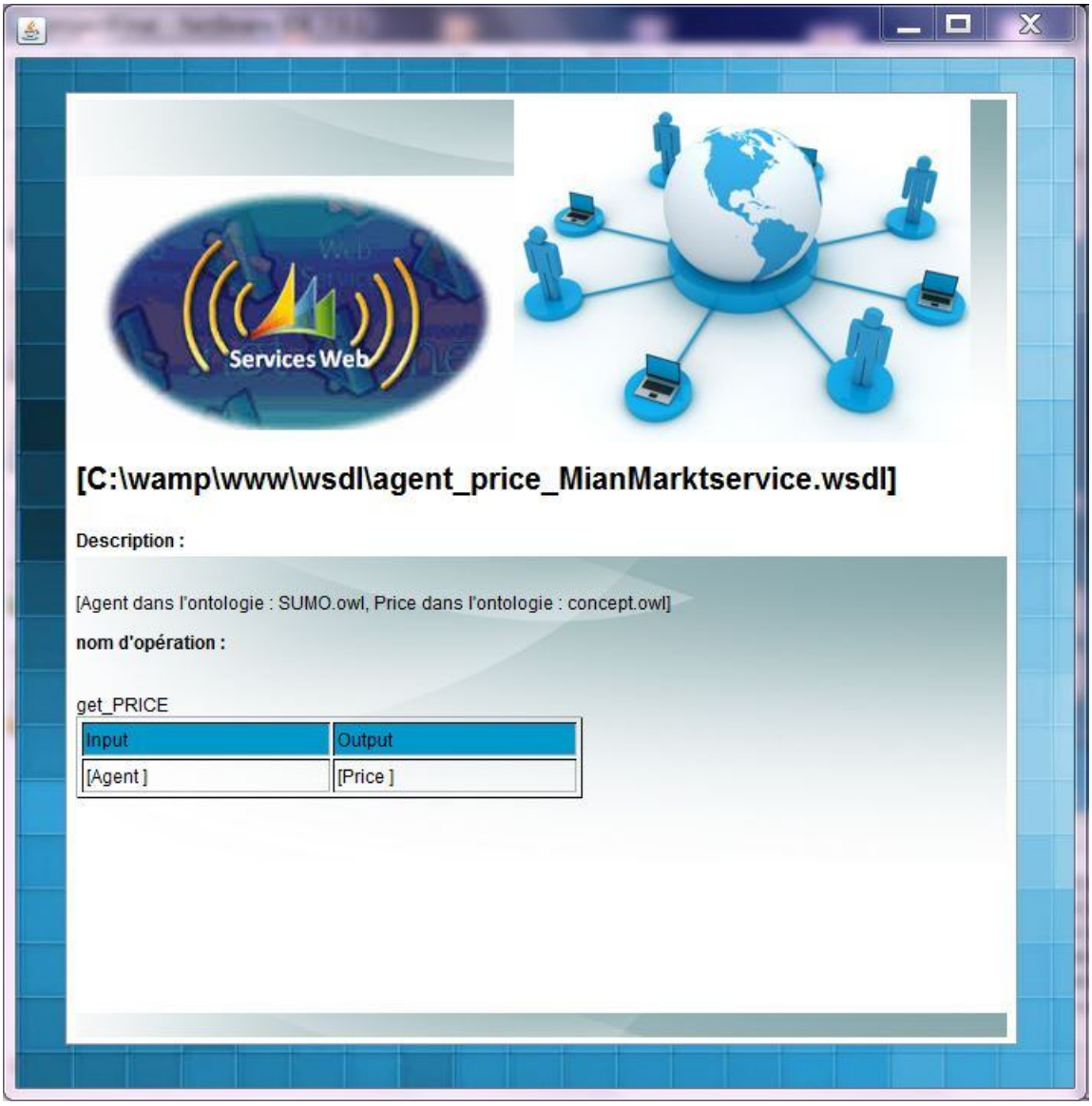


Figure 5. 2 :L'interface résultat

L'interface ontologie/service_web se compose de 2 zones comme montré dans la figure 5.3 :

- ***Zone d'ontologie*** : elle représente la partie où le système affiche les concepts obtenus après la projection sémantique de la requête utilisateur sur les ontologies, l'affichage est fait sous forme d'arborescence.
- ***Zone de WSDL:*** elle représente la partie ou le système affiche tous les services web et les entrées et les sorties de chaque service, l'affichage est fait sous forme d'un table.

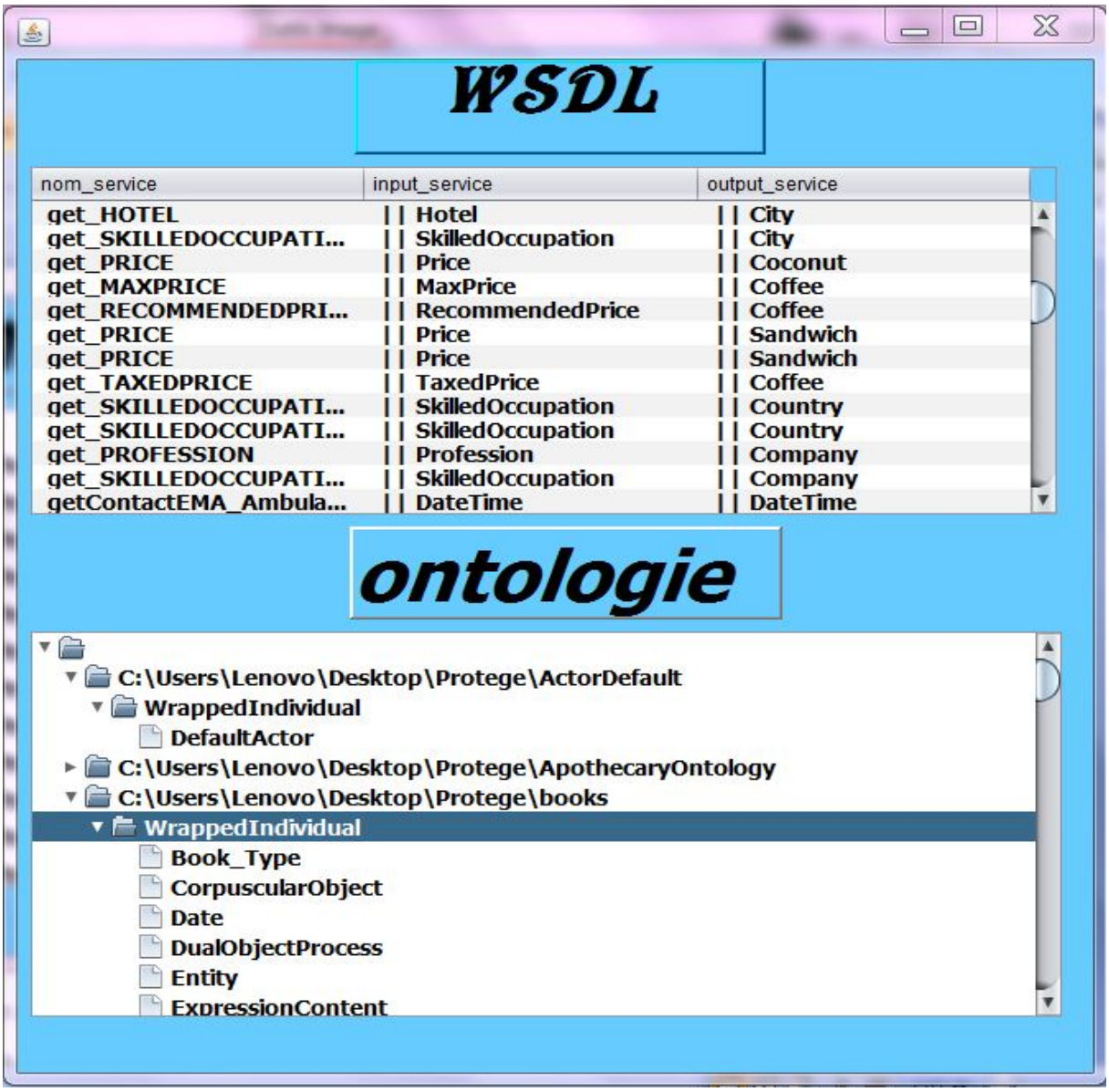


Figure 5. 3:L'interface ontologie_service web

V.2.2. Mécanisme et Principe de fonctionnement

L'objectif de l'utilisation de l'application découverte est d'accéder à des informations pertinentes par une recherche sémantique, guidée par les ontologies. Lorsque l'utilisateur soumit son besoin de services sous forme d'une requête au découverte, un clique sur le bouton « découverte», donnera lieu au déclenchement de quatre processus qui coopèrent et collaborent entre eux afin de répondre au besoin de l'utilisateur. Ces processus sont relatifs aux modules décrits dans le chapitre précédent. Il se déroule de la manière suivante :

a- l'extraction d'information

Le processus de recherche et d'extraction d'information est une suite d'étapes permettent l'extraction des informations rendus par les fichiers WSDL. Il s'agit d'une recherche qui utilise la requête utilisateur et (entrées ou sortie) des services web envoyé au processus de projection (recherche).

b- Projection sémantique (recherche)

Une fois le processus de projection récupère la requête utilisateur et (entrées ou sorties) des services web, il lance la recherche de ces termes dans tous les ontologies. Les résultats de ce processus sont des données envoyés au processus de calculde la distance sémantique. Ces données sont la requête de l'utilisateur et les (entrées ou sorties) de chaque service web.

c- Calcule la distance sémantique

Une fois le processus de calcule de la distance sémantique récupère les résultats de la projection, nous avons appliqué un mécanisme permettant le calcul de la distance entre la requête et les entrée ou sortie d'un service web. La distance utilise des formules pour calculer la mesure de similarité entre la requête utilisateur et les services web.

Conclusion

Nous avons présentée dans ce chapitre une implémentation de notre application nommée « découverte ». Le travail réalisé est un système de découverte des services web guidé par des ontologies. Ces ontologies permettent d'ajouter une couche sémantique à notre système et d'effectuer une recherche par la sémantique des mots.

L'implémentation a été réalisée en utilisant le langage de programmation Java et l'environnement NeatBeans.

Conclusion générale

Dans ce mémoire, nous avons présenté une conception et une réalisation d'un système de découverte d'information sur les services web qui prend en compte la sémantique. Le processus de découverte est guidé par un graphe de concepts construit à partir d'une ontologie. Ce système permet de vérifier la pertinence des services Web par rapport à la requête utilisateur. À cet effet, nous avons proposé une architecture générale pour extraire les services web qui ont une correspondance sémantique avec la requête utilisateur.

Donc, le système permet la récupération des résultats d'une découverte sur les services web, il s'appuie sur plusieurs éléments, à savoir, les ontologies pour la prise en compte de la sémantique,le parsing est le traitement du contenu d'un document WSDL à fin d'extraire les données significatifs pour chaque service Web, et il utilise également un mécanisme d'évaluation basé sur certaines relations ontologiques et des mesures de distance sémantique.

Bibliographies

[1] : <Razika DRIOUCHE> Proposition d'une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'EbXML et la mobilité des agents [2007]

[2] : < Issam RABHI>Testabilité des services Web[09 janvier 2012]

[3] : Jiang J, Conrath D. Semantic similarity based on corpus statistics and lexical taxonomy. In: Int Conf Res Comput Linguist (ROCLING X) [1997].

[4] :Julien Guitton. Planification multi-agent pour la composition dynamique de services Web [12 Juin 2006]

[5] :Amina BEKKOUCHE. Composition des Services Web Sémantiques a base d'Algorithmes Génétiques [2011-2012]

[6] : Sylvain RAMPACEK.Sémantique ,interactions et langages de description des services web complexes[10 novembre 2006]

[7] :*Mme. Kaouthar FAKHFAKH.*Approche sémantique basée sur les intentions pour la modélisation, la négociation et la surveillance des contrats de qualité de service[*21 Mai 2011*]

[8] :Céline LOPEZ-VELASCO. Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation[18 novembre 2008]

[9] : Conception et Réalisation pour la médecine de travail basé sur : Les Services Web Sémantique

[10] :Mohamed Gharzouli. Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer[25/09/2011]

[11] : Mr. Yassin CHABEB. Contributions à la Description et la Découverte de Services Web Sémantiques[novembre 2011]

[12] : Mohamed GHARZOULI.Intégration d'ontologies dans le cadre du web sémantique : une détection des relations sémantiques basée sur le RÀPC.[25/09/2011]

[13] :PHAN Quang Trung Tien>.Ontologies et Web Services[juillet 2005]

[14] : Mohammad Mustafa Taye. Understanding Semantic Web and Ontologies: Theory and Applications [6, June 2010]

[15]: Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha,Tim Finin, Anupam Joshi. Using Ontologies in the Semantic Web: A Survey [July, 2005]

[16]:Mohamed Khaled KHELIF. Web sémantique et mémoire d'expériences pour l'analyse du transcriptome[Le 4 avril 2006]

[17] : Wu Z, Palmer M. Verb semantics and lexical selection. In: 32nd. Annu Meet Assoc Comput Linguist [1994].

[18] : Semantic web services matchmaking: Semantic distance-based approach

[19]: Resnik P. Using Information content to evaluate semantic similarity in a taxonomy. In: Proc 14th Int Jt Conf Artif Intell IJCAI [1995].

[20]: Sébastien Harispe , David Sánchez , Sylvie Ranwez , Stefan Janaqi , Jacky Montmain **.**A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain[2013]

[21]:] Rada R, Mili H, Bicknell E, Blettner M. Development and application of a metric on semantic nets. IEEE Trans Syst Man Cybern [1989];

[22]: Tversky A. Features of similarity. Psychol Rev [1977];

[23]: Sánchez D, Batet M, Isern D, Valls A. Ontology-based semantic similarity: a new feature-based approach. Expert Syst Appl [2012];

[24]: Rodríguez A, Egenhofer MJ. Determining semantic similarity among entity classes from different ontologies. IEEE Trans Knowl Data Eng[2003];

[25]: Maedche A, Staab S. Comparing ontologies – similarity measures and a comparison study (internal report). Karlsruhe; [2001].

[26]: Lin D. An information-theoretic definition of similarity. In: 15th Int Conf Mach Learn, Madison, WI; [1998].

[27]: Pirró G, Euzenat J. A feature and information theoretic framework for semantic similarity and relatedness. In: Proc 9th Int Semant Web Conf ISWC 2010, Springer; [2010].

[28]: Mazandu Gaston K, Mulder Nicola J. IT-GOM: an integrative tool for IC-based GO semantic similarity measures; [2011].

[29]: Sánchez D, Batet M, Isern D. Ontology-based information content computation. Knowledge-Based Syst [2011];

[30]: Seco N, Veale T, Hayes J. An intrinsic information content metric for semantic similarity in WordNet. In: 16th Eur Conf Artif Intell. IOS Press; [2004].

[31]: Zhou Z, Wang Y, Gu J. A new model of information content for semantic similarity in WordNet. In: FGCNS ’08 Proc 2008 Second Int Conf Futur Gener Commun Netw Symp – vol. 03, IEEE Computer Society; 2008.

[32]: Couto FM, Silva M, Coutinho PM. Implementation of a functional semantic similarity measure between gene-products. Department of Informatics, University of Lisbon; 2003.

[33]:Leacock C, Chodorow M. Combining local context and WordNet similarity for word sense identification. In: Fellbaum C, editor. WordNet an electron, Lex database. MIT Press; 1998.