République Algérienne Démocratique et Populaire.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.
Université A-Mira de Béjaïa Faculté des Sciences Exactes
Département Informatique

ÉCOLE DOCTORALE RÉSEAUX ET SYSTÈMES DISTRIBUÉS

Mémoire de Magistère

En Informatique

Option : Réseaux et Systèmes Distribués

Thème

Mapping d'ontologies dans un environnement distribué

Présenté par

Fouzia BOUDRIES

Devant le jury composé de

Président	DAHMANI A-Nacer	Professeur	U. A/Mira Béjaïa.
Rapporteur	TARI A-Kamel	M.C.A	U. A/Mira Béjaïa.
Rapporteur	JUIZ Carlos	Professeur	U.des iles Baléares.
Examinatrice	BOUFAIDA Zizette	Professeur	U. Constantine.
Examinatrice	NADER Fahima	M.C.A	E.S.I, Alger.
Invitée	TALANTIKITE Hassina	M.C.B	U. A/Mira Béjaïa.

Béjaïa, 2008/2009

Dédicaces

A mes très chers fils

A mon époux

A ma belle mère et mes parents

A toute la famille

A mes amis et collègues

Remerciements

Je tiens en premier lieu à exprimer ma profonde gratitude et mes vifs remerciements à mes encadrants, Pr Carlos JUIZ, Dr TARI A-Kamel et Isaac LERA qui m'ont accueilli, accompagné et conseillé tout au long de ce parcours.
Je remercie sincèrement les membres de jury qui m'ont fait l'honneur de juger ce travail.
Mes remerciements les plus chaleureux vont vers toute ma famille, ma belle mère, mon époux et mes parents qui m'ont supporté et soutenue depuis toujours.
Un merci très respectueux à mes amis dont la présence m'a permis de mener ce travail à bien.
Merci à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

Résumé

L'interopérabilité sémantique entre sources d'information hétérogènes est une problématique importante sur le web. L'utilisation des ontologies est une voie très prometteuse pour permettre l'interopérabilité, seulement les ontologies elles même peuvent être hétérogènes. Le mapping d'ontologies est le noyau de cette interopérabilité, cependant la génération automatique des correspondances entre deux ontologies est d'une extrême difficulté qui est dû aux divergences (conceptuelle, habitudes, etc.) entre communautés différentes de développement des ontologies. Dans ce travail, nous avons proposé un algorithme pour le mapping d'ontologies dans un environnement distribué. Le but de cet algorithme est de satisfaire les environnements distribués en termes de qualité des mapping produits et en termes de temps d'exécution. Notre algorithme considère seulement les éléments les plus significatifs des deux ontologies OWL en entrée et utilise une ressource externe Word net dans le but d'augmenter la possibilité de trouver des mapping entre deux entités. Notre algorithme donne des résultats très satisfaisants en terme de qualité, il obtient une moyenne de "Précision" et "Rappel" de 0,76 et 0,82 et une complexité du temps d'exécution de $O(n^2)$ trés proche de $O(n * log(n))$ se qui rend son applicabilité dans les environnements distribués.

Mots clés **:** Mapping d'ontologies, Ontologies, Intégration d'information, Web sémantique.

Abstract

Semantic interoperability between heterogeneous information sources is an important issue on the web. The use of ontologies is a very promising approach to enable interoperability, only ontologies themselves can be heterogeneous. The mapping of ontologies is the core of this interoperability, but the automatic generation of correspondence between two ontologies is extremely difficulty is due to differences (conceptual, habits, etc.) between different communities in the development of ontologies. In this work, we proposed a new algorithm for the mapping of ontologies in a distributed environment. The purpose of this algorithm is to deal with the distributed environments in terms of quality of mapping produced and in terms of time execution. The algorithm considers only the most significant elements of two OWL ontologies as input and uses an external resource Word Net in order to increase the possibility of finding mapping between two entities. The algorithm gives good results in terms of quality, it gets an average of "accuracy" and "Recall" of 0.76 and 0.82 and complexity of the time execution of $O(n^2)$ very near to $O(n * log(n))$ that make it applicability in distributed environments.

Keywords **:** Ontology mapping, Ontology, Information integration, Semantique web.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le mapping d'ontologies est un nouveau paradigme dans le web sémantique où les ontologies sont utilisées pour la représentation de l'information en utilisant un langage de description tel que OWL (Ontology Web Language)[12]. Ces ontologies représentent les connaissances d'un domaine, elles regroupent les concepts, les relations et les contraintes d'un domaine. Une définition de l'ontologie est proposée en [4] est que l'ontologie est une spécification formelle d'une conceptualisation partagée.

Le mapping d'ontologies apparait dans plusieurs domaines d'applications tel que l'intégration d'information, la composition des services web etc. Il est considéré comme étant le point clé pour la recherche d'interopérabilité entre agents et sevices basés sur les ontologies. L'interopérabilité vise à assurer la coopération, la communication et le partage d'informations et de services entre plusieurs applications indépendamment des plateformes matériels. Le mapping tente de découvrir les correspondances entre les entités de deux ontologies afin de les combiner et de produire des nouvelles informations.

Différentes techniques pour la découverte des correspondances ont été introduites dans la littérature. Parmi ces techniques, nous trouvons celles qui se basent sur la comparaison des labels (les termes attribués aux concepts), d'autres considèrent la relation hiérarchique entre les concepts pour tirer ces correspondances et en fin celles qui utilisent les instances.

Les ressemblances basées sur les labels sont essentielles puisque deux entités qui portent le même nom sont équivalentes. Il existe plusieurs méthodes, celles basées sur l'analyse des morphèmes (préfix et stem) et celles qui utilisent des formules pour trouver la distance entre les labels et d'autres considèrent des ressources linguistiques externes comme des dictionnaires et thésaurus. Généralement, ces méthodes nécessitent une phase de normalisation afin d'éliminer les mots vides (conjonction, propositions, chiffres ...) et de les mettre sous un format uniforme.

Les ressemblances structurales se basent soit sur la comparaison de la structure interne de l'entité(les attributs, leurs domaines...) ou externes (considération des voisins et les

relations). Généralement, la similarité entre deux entités est donnée en terme de nombre d'arcs séparant les deux entités. Enfin les techniques basées sur les instances, calculent l'intersection entre l'ensemble des instances de deux entités. A partir de cette intersection, elles estiment la relation entre les deux entités.

La nature distribuée des différentes applications où le mapping est considéré comme étant une phase essentielle, les multiples manières de représenter un domaine et l'hétérogénéité des sources d'information et des ontologies et leurs évolution en nombre et en taille rendent la résolution du mapping et de l'interopérabilité complexe.

Beaucoup d'efforts ont été conduits à l'automatisation de la découverte du mapping et différents algorithmes ont été proposés dans la littérature [6] et où chacun à une vision différente pour tirer ces correspondances.

Cependant, les algorithmes qui ont été proposés se focalisent et considèrent seulement la qualité des résultats produits et négligent d'autres paramètres liés à la nature distribuée des différentes applications comme le temps de réponse et la communication. L'interaction et la communication dans tels environnements nécessitent un temps de réponse raisonnable et une qualité élevée. Le problème dans ces algorithmes est qu'ils appliquent une recherche exhaustives et considèrent toutes les caractéristiques possibles entre les différentes entités des ontologies, ce qu'engendre un temps d'exécution considérable, en conséquence, ils perdent leurs efficacités dans les environnements distribués.

Dans ce mémoire, nous proposons un algorithme distribué "multicouche et multi agents" pour le mapping d'ontologies dans un environnement distribué qui se base sur la comparaison des entités significatives au lieu de comparer toutes les entités (concepts) et une exécution parallèle des différents agents qui se chargent de mapping dans le but de prouver l'efficacité (temps d'exécution) et la qualité des résultats. Afin d'augmenter la possibilité de trouver des mapping entre les entités, nous avons ajouté des termes supplémentaires comme les synonymes et les hyperonymes de chaque entité et cela par l'utilisation de la ressource externe Word net qui est un dictionnaire de langue Anglaise.

Pour l'évaluation de notre algorithme, nous avons utilisé le benchmark proposé par OAEI en 2010 ,qui est composé d'un ensemble d'ontologies OWL du domaine de la bibliographie avec leurs alignements de référence. Ce Benchmark, nous a permet le calcul les paramètres de performance (précision, rappel, temps d'exécution) et de faire une comparaison de nos résultats avec d'autre algorithmes validés avec le même jeu de données.

L'algorithme proposé obtient une moyenne de précision et rappel de 0,76 et 0,82. Avec ces valeurs, nous constatons que notre algorithme présente une qualité satisfaisante. Nous avons aussi obtenu une complexité quadratique c.-à-d polynomiale de $O(n^2)$ proche de la meilleur complexité obtenu par QOM.

Avec la valeur petite de n qui présente le nombre de classes de l'ontologie, les algorithmes polynomiaux sont considérés efficaces.
En résumant, l'approche proposée offre une qualité satisfaisante et efficace en terme du temps d'exécutions. En conséquence, ces caractéristiques rendent son applicabilité dans les environnements distribués.

Ce mémoire est organisé en 4 chapitres :
Dans le premier chapitre, nous avons abordé une étude sur le web sémantique et l'ontologie. Le second, une étude de la problématique "le mapping d'ontologies" qui est apparu dans le web sémantique, où nous avons abordé tous les points reliés à ce processus, sa définition, les problèmes rencontrés rendant ce processus difficile et les différentes techniques et méthodes proposées pour tirer les correspondances entre les entités de deux ontologies ainsi les domaines d'applications. Puis nous enchainons avec un état de l'art sur les algorithmes de mapping d'ontologies et une comparaison entre ces derniers.

Dans le chapitre3, nous avons présenté un algorithme " multicouche et multi agents " pour le mapping d'ontologies dans les environnements distribués, où nous avons détaillé son architecture et sa structure ainsi que l'environnement de développement. Le chapitre 4 est consacré pour l'évaluation de notre algorithme en terme de précision, rappel et complexité et une comparaison avec d'autres algorithmes. Enfin, nous conclurons ce travail en dégageant plusieurs pistes de recherche notamment en évoquant les hypothèses de travail qui restent à reconsidérer et les points forts et faibles de notre approche à améliorer dans l'avenir proche.

Chapitre 1

Ontologies

1.1 Introduction

Pour supporter l'interopérabilité, le partage et la réutilisation des connaissances formellement représentées parmi les différents systèmes, il est utile de définir un vocabulaire commun dans lequel les connaissances partagées seront représentées [2]. La spécification de ce vocabulaire de représentation est appelée Ontologie.

Le concept " ontologie " est un terme empreinte de la philosophie. Il est utilisé dans plusieurs domaines comme l'ingénierie des connaissances, les systèmes expert et en informatique. Parmi les damianes où l'ontologie joue un rôle fondamental est le web sémantique, qui est fortement distribué (réparti) et qui constitue une prolongation du web actuel où il a ajouté des ontologies pour préciser le sens et la sémantique des ressources du web afin que les machines puissent les comprendre.

Dans ce chapitre, nous allons commencer par la définition du web sémantique puis nous présentons le concept ontologie à savoir les différentes définitions proposées dans la littérature, ses constituants, ses rôles, les différents langages pour leur conception.

1.2 Web sémantique

L'expression Web sémantique, due à Tim Berners-Lee en 2001 au sein du W3C, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés. Le web sémantique est un espace virtuel et à la différence du Web que nous connaissons aujourd'hui, les utilisateurs sont déchargés d'une bonne partie de leurs tâches de recherche, de construction

et de combinaison des résultats, grâce aux capacités accrues des machines à accéder aux contenus des ressources et y effectuer des raisonnements sur ceux-ci. Le Web actuel est essentiellement syntaxique, dans le sens où la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus. La nouvelle génération de Web - Le Web sémantique - a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus par les ontologies. Concrètement, le Web sémantique est donc une infrastructure pour permettre l'utilisation de connaissances formalisées (ontologies) en plus du contenu informel actuel du Web.

1.3 Ontologie

Le terme " ontologie " est empreinte de la philosophie et signifie la représentation de l'être en tant qu'être, c'est-à-dire la théorie qui tente d'expliquer les concepts qui existent dans le monde et comment ces derniers s'imbriquent et s'organisent pour donner du sens. Les ontologies sont apparues au début des années 90 dans la communauté ingénierie des connaissances, dans le cadre des démarches d'acquisition de connaissances pour les systèmes à base de connaissances (SBC) [3]. Faisant suite aux systèmes experts qui séparaient une base de connaissances " déclarative " et un moteur d'inférence " procédurale ", les SBC proposaient alors de spécifier d'un côté les connaissances du domaine modélisé et de l'autre les connaissances du raisonnement en inférant par l'utilisation. L'idée de cette séparation modulaire était de construire mieux et plus rapidement des SBC en réutilisant le plus possible des composants génériques, que ce soit au niveau du raisonnement ou au niveau des connaissances. Les connaissances du domaine précisent tout ce qui a trait au domaine. Dans ce contexte, les chercheurs ont proposé de construire ces connaissances par une spécification sous forme d'une ontologie qu'est un ensemble structuré par différentes relations, principalement la relation de subsumption entre les objets du domaine. Par la suite, ce concept est adopté dans la science de l'informatique.

Plusieurs définitions de ce concept ont été proposées dans la littérature. Parmi ces définitions nous citons :

✍ **Définition 1 :** Une ontologie définie les termes basiques et les relations entre ces derniers. Elle regroupe le vocabulaire d'un domaine ainsi que les règles qui combinent ces termes et ces relations afin de définir une extension de ce vocabulaire [1].

A partir de cette définition, on peut comprendre qu'une ontologie est un ensemble de termes et de relations entre eux, plus toutes les connaissances que nous pouvons inférer à

partir d'elle (nouvelles informations).

✍ **Définition 2** Gruber [2] définit l'ontologie comme étant une spécification explicite d'une conceptualisation.

Le terme "Conceptualisation" se réfère à un modele abstrait d'un domaine, où nous relèvons tous les concepts de ce domaine et le terme "explicite" signifie que le type de concept et les contraintes sur ce dernier sont explicitement définis.

✍ **Définition 3**

Borst[4] a modifie légèrement la définition proposée par Gruber. Il stipule qu'une ontologie est une spécification formelle d'une conceptualisation partagée .

Le "conceptualisation" est une représentation abstraite d'un monde réel, le terme "formelle" signifie que l'ontologie est manipulée par la machine et le terme "partagé" reflète qu'une ontologie capture la connaissance consensuelle, c-à-d elle n'est pas privée d'un certain individu mais admise par un groupe (il faut qu'une communauté valide cette ontologie).

De multiple définitions d'ontologies existent dans la littérature, qui sont complémentaires et ont une vision commune de l'ontologie à savoir qu'elle constitue l'ensemble de concepts et de relations entre eux ainsi que les contraintes sur ces concepts.

1.4 Les éléments constitutifs d'une ontologie

Une ontologie est une représentation de connaissances au niveau conceptuel toujours liée à un domaine particulier de connaissances, autrement dit elle définit le vocabulaire et les axiomes qui restreignent l'interprétation de ce domaine. Ce vocabulaire est structuré en un ensemble de concepts et un ensemble de relations existantes entre eux.

1.4.1 Un concept

Un concept représente un objet, une notion ou une idée du domaine [6]. Il est caractérisé par un terme ou plusieurs termes (cas où le concept admis des synonymes), une extension et une intension. Il regroupe les entités de même type.

- Un terme : est un élément lexical qui permet d'exprimer le concept en langue naturelle et peut admettre des synonymes.
- Une extension : une extension est l'ensemble des objets manipulés à travers ce concept qui sont également appelés instances du concept.

– Une intension : appelée aussi notion. Elle spécifie la sémantique du concept à travers un ensemble de propriétés et des contraintes. Par exemple, le concept "personne" est caractérisé par un nom, un prénom et une adresse.

1.4.2 Les relations

Les relations représentent les liens conceptuels pouvant exister entre les concepts. Une relation est caractérisée par un terme, une extension qui est l'ensemble des tuples d'instances liés par cette relation et une intension qui spécifie d'une part les concepts pouvant être liés par cette relation et d'autre part la sémantique d'usage de la relation. A titre d'exemple, dans le domaine de la géométrie, les concepts "Point", "Droite" et "Plan" doivent être pris en compte ainsi que la relation (appartenir-à) entre un "Point" et une "Droite" ou un "Point" avec un "Plan".

1.4.3 Les propriétés

Les propriétés représentent les intensions des concepts et des relations du domaine. Parmi ces propriétés, nous citons :

✍ **Les propriétés algébriques d'une relation :** Comme la symétrie, la réflexivité et la transitivité etc.

✍ **La propriété de subsomption entre concepts ou entre relations :** Un exemple d'une relation de subsumption : un concept C1 subsume un concept C2 si toutes les propriétés sémantiques de C1 sont aussi des propriétés sémantiques de C2. Autrement dit C2 est plus spécifique que C1.

✍ **La généricité :** un concept est dit générique s'il n'admet pas d'extension (i.e. ce concept n'admet pas d'instances).

✍ **L'anti-rigidité :** un concept est anti-rigide si toute instance de ce concept est essentiellement définie par son appartenance à l'extension d'un autre concept. Exemple : un étudiant est un concept anti-rigide car l'étudiant est avant tout un humain

✍ **La cardinalité d'une relation :** La cardinalité précise le nombre d'instances d'un concept qu'une relation peut lire.

✍ **La disjonction (incompatibilité) :** deux concepts sont disjoints si leurs extensions sont disjointes.

✍ **L'exclusivité :** Deux relations sont exclusives si quand l'une lie des instances des concepts, l'autre ne lient pas ces instances, et vice-versa. L'exclusivité entraine l'incompatibilité. Exemple : l'appartenance et la non appartenance.

✍ **L'équivalence :** deux concepts sont équivalents s'ils ont la même extension.

✍ **L'inverse :** Deux relations binaires sont inverses l'une de l'autre si lorsque une lie deux instances I1 et I2, l'autre lie I2 et I1. Exemple : les relations " a pour père " et " a pour enfant " sont inverses l'une de l'autre.

1.5 Rôles d'ontologie

- **La classification :** c'est la représentation simple d'une ontologie, appelé aussi Taxonomie. Elle regroupe et organise les concepts dans une hiérarchie de classes afin de préciser leurs sens. La relation généralement utilisée est " is-a ".
- **La communication :** pour permettre une communication sémantique entres les systèmes, il faut qu'ils aient accès non seulement aux termes utilisés par l'être humain, mais également à la sémantique que ce dernier associe aux différents termes. Pour cela, les représentations symboliques utilisées dans les machines doivent avoir du sens aussi bien pour la machine que pour les utilisateurs. "Avoir du sens" signifie ici que l'on peut relier les informations représentées à d'autres informations.
  Supposons, par exemple, qu'un certain nombre de sites Web contiennent des informations médicales ou fournissent des services de e-commerce en médecine. Si ces sites partagent et publient tous la même ontologie, qui est à la base des termes qu'ils utilisent, alors les agents informatiques peuvent extraire et agréger l'information de ces différents sites. Les agents peuvent utiliser cette information agrégée pour pouvoir répondre aux interrogations des utilisateurs ou comme données d'entrée pour d'autres applications.
- **La recherche d'information :** Un des plus grands projets basés sur l'utilisation d'ontologies consiste à ajouter au Web une véritable couche de connaissances permettant dans un premier temps de préciser leurs sémantiques et de faire desrecherches d'informations au niveau sémantique et pas simplement syntaxique.
- **La réutilisation :** les systèmes à base de connaissances sont les systèmes qui doivent avoir la possibilité de manipuler les informations d'une manière formelle mais aussi en accord avec la sémantique du domaine de connaissance. A travers cette définition, nous pouvons affirmer que la représentation de connaissances du domaine est liée à la manière dont elle sera opérrationnalisée. Donc, la réutilisation de ces connaissances pour une autre utilisation est impossible.
  Pour augmenter la réutilisation de ces connaissances, il faut que leurs représentations se fasse au niveau conceptuel et non opérationnel. Pour cela, nous nous appuyons sur l'utilisation de spécifications des connaissances à travers des paradigmes conceptuels. Cette spécification est appellée ontologie.

Un autre exemple de la réutilisation est que plusieurs domaines ont besoin de représenter la notion de temps. Cette représentation comprend les notions d'intervalles de temps, de moments précis de temps, de mesures relatives de temps etc. Lorsqu'un groupe de chercheurs développe une telle ontologie, les autres groupes peuvent simplement la réutiliser pour leurs propres domaines. En plus, s'il a besoin de construire une ontologie plus large, il serait possible d'intégrer plusieurs ontologies existantes décrivant des portions d'un domaine. Nous pouvons également, réutiliser une ontologie générale(de haut nivea) et l'étendre afin de permettre de décrire un domaine d'intérêt plus spécifique.

1.6 Types d'ontologie

Les méthodes en Ingénierie des connaissances ont répertorié plusieurs types d'ontologie liés à l'ensemble des objets conceptualisés et manipulés au sein d'un SBC [21]. Parmi ces types d'ontologies, nous citons :

1. **Ontologie de domaine :** une ontologie de domaine est la représentation d'un domaine particulier de connaissances. Elle exprime des conceptualisations spécifiques à un domaine. Elle regroupe les concepts et les relations permettant de couvrir les connaissances d'un domaine. La plupart des ontologies disponibles sur le web sont des ontologies de domaines.A titre d'exemple d'ontologie de domaine médicale(MENELAS)qui regroupe les concepts deu domaine de la médecine, l'ontologie du domaine du tourisme et celles de domaine de la bibliographie et l'anatomie humaine ainsi celle de Food et Wine.
2. **Ontologie générique :** Un critère également utilisé pour caractériser une ontologie est sa granularité, c'est-à-dire le niveau du détail de sa conceptualisation. En fonction de l'objectif opérationnel, une connaissance plus ou moins fine du domaine est nécessaire, et des propriétés considérées comme accessoires dans certains contextes peuvent se révéler indispensables pour d'autres applications. Les ontologies génériques ou de haut niveau ont une granularité large, du fait que les notions sur lesquelles elles portent peuvent être raffinées par des notions plus spécifiques. elles regroupent les concepts les plus abstrait du domaine [10]. Elles sont conçues pour les réutiliser, les raffiner en cas de besoin et les intégrer dans différentes applications.
3. **Ontologie de tache :** c'est une ontologie spécifique pour résoudre un problème bien définit comme les problèmes de planification et de diagnostic. Elle offre un ensemble de termes au moyen desquel nous pouvons décrire généralement comment

résoudre un type de problèmes. Elle contient des noms, des verbes et des adjectifs pour la descriptions de tâches.

1.7 Construction des ontologies

La construction d'une ontologie est une tâche difficile et complexe en raison des différentes connaissances que nous pouvons trouver dans un domaine. L'ontologie est considérée également comme un composant logiciel et un cycle de vie pour sa construction est proposé.

1.7.1 Cycle de vie d'une ontologie

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes informatiques répondant à des objectifs opérationnels différents, leurs développements doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel [7]. En particulier, les ontologies doivent être considérées comme des objets techniques évolutifs possédants un cycle de vie qui nécessite une spécification. Les activités liées aux ontologies sont, d'une part, des activités de gestion de projet (planification, contrôle, assurance qualité), et d'autre part, des activités de développement (spécification, conceptualisation, formalisation) ; s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration [8]. Un cycle de vie inspiré du génie logiciel est proposé dans [7]pour l'ontologie.

La figure 1.1 suivante présente le cycle de vie d'une ontologie. Il comprend une étape initiale d'évaluation des besoins, une étape de construction, une étape de diffusion et une étape d'utilisation. Après chaque utilisation significative, l'ontologie et les besoins sont réévalués et peuvent être étendue et si nécessaire, en partie reconstruite..

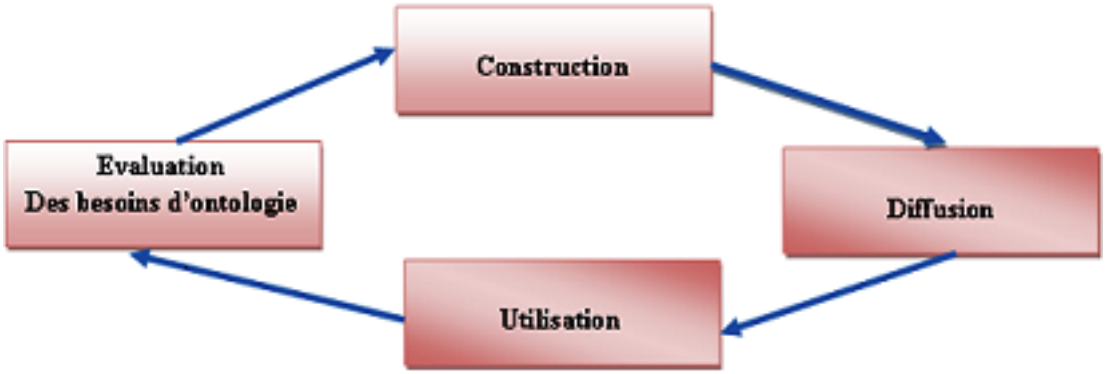

FIGURE 1.1 – Le cycle de vie d'une ontologie

1.7.2 Principes méthodologiques

La construction d'une ontologie passe par une suite d'étapes en commençant par sa conceptualisation jusqu'à son opérationnalisation. Dans chacune de ces étapes, un ensemble de règles et de conseils sont proposés pour guider le constructeur dans sa tâche.

1.7.2.1 Etape de conceptualisation

La conceptualisation d'une ontologie consiste à identifier dans un corpus les connaissances du domaine. Cette étape peut se dérouler en deux phases : La première étant de faire un tri entre les différentes connaissances identifiées en précisant celles qui sont spécifiques au domaine et celles qui ne participent qu'à l'expression du domaine. Dans le cas d'intégration d'autres ontologies, les connaissances spécifiées dans ces ontologies ne doivent pas être prises en compte. La découverte des connaissances d'un domaine peut s'appuyer à la fois sur l'analyse des documents (texte) et sur des interviews avec les experts du domaine. Ces activités doivent être raffinées au fur et à mesure que la conceptualisation émerge. Les interviews avec les experts permettent de préciser la sémantique différentielle d'un concept. Une fois les concepts et relations identifiés par leurs termes, il faut décrire la sémantique en indiquant, à priori en langage naturel, leurs instances connues, les liens qu'ils entretiennent entre eux et leurs propriétés. La description d'une primitive conceptuelle doit contenir des liens vers les parties du corpus qui mettent en évidence sa sémantique ; ce qui permet en cas où une ambiguïté sémantique demeure de revenir au corpus. La finalité de cette étape est un modèle conceptuel qui consiste en un ensemble de termes désignant les entités du domaine de connaissances (concepts, relations, propriétés des concepts et des relations, etc.).

La problématique de la conceptualisation est essentiellement de reconnaître, dans les

corpus brutes, les connaissances qui relèvent du domaine considéré et de les recenser de façon exhaustive.

Les étapes suivantes du processus conduisant progressivement à la formalisation de ces connaissances.

1.7.2.2 Etape d'Ontologisation

L'Ontologisation correspond à l'évolution de l'ontologie régionale (graphe de concepts) vers une ontologie formelle. D'après [7], le respect de la sémantique du domaine doit être assuré par un engagement ontologique. Cette notion est proposée initialement par T. GRUBER comme un critère pour utiliser une spécification partagée d'un vocabulaire. Le sens d'un concept est donné par son extension dans l'univers d'interprétation du langage. Le respect de l'engagement ontologique revient à donner à chaque concept son extension et sa manipulation conformément au sens donné par cette extension [6].

1.7.2.3 Etape d'Opérationnalisation

Une ontologie n'est qu'une représentation conceptuelle des connaissances d'un domaine, indépendamment des utilisations opérationnelles qui peuvent en être faites. La sémantique formelle dont elle est dotée sert à exprimer la sémantique de manipulation des primitives conceptuelles liée au domaine. Afin d'intégrer une ontologie au sein d'un SBC, il convient donc de transcrire celle-ci dans une forme adaptée à l'usage auquel est destiné le SBC, c'est-à-dire préciser la sémantique de manipulation des axiomes, qui est liée à l'application envisagée et non au domaine considéré.

La forme donnée à l'ontologie dans un SBC doit être opérationnelle dans le sens où la représentation des connaissances doit permettre leur utilisation conformément à l'objectif opérationnel du SBC dans le sens où le formalisme d'opérationnalisation doit offrir des mécanismes d'inférence permettant la manipulation de l'information prévu dans le SBC. Par exemple, pour effectuer des raisonnements automatiques, le formalisme opérationnel doit permettre la représentation de règles de dérivation et des mécanismes effectifs d'application de ces dernières. L'utilisation d'une ontologie dans un SBC suppose donc son opérationnalisation, qui est un processus qui consiste à plonger l'ontologie dans un formalisme opérationnel de représentation des connaissances, en accord avec le type d'application visé.

La figure 1.2 illustre les étapes de construction d'une ontologie en partant des données brutes (domaine de connaissances) jusqu'à sa formalisation.

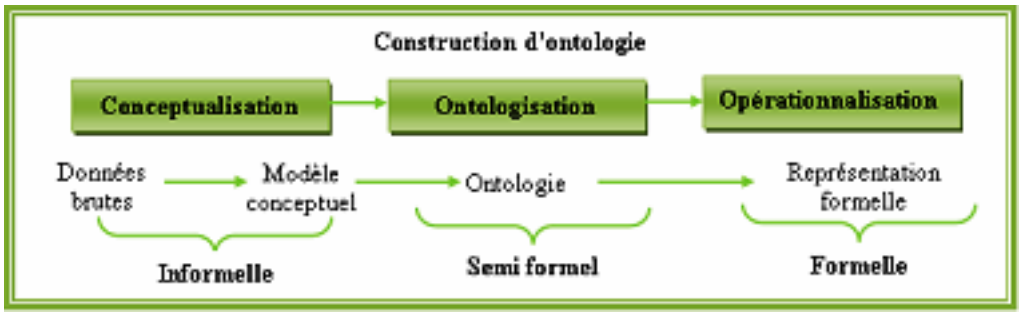

FIGURE 1.2 – Etapes de construction d'ontologie

1.8 Modèles de représentation des ontologies

Différents modèles de représentation des ontologies ont été proposés, comme les modèles qui s'appuient sur la logique de description (logique des prédicats du 1er ordre, logique de description) et ceux qui s'appuient sur les approches base de données (le modèle conceptuel).

Les modèles fondés sur la logique sont les plus utilisés car ils sont dotés de mécanismes de raisonnement sur les informations modélisés. Par contre, les modèles fondés sur les approches Bases De Données tirent profit de leur longue expérience dans la modélisation d'information.

Les modèles utilisés pour la représentation des connaissances d'ontologie sont les modèles de Frame, les logiques de descriptions et les graphes conceptuels.

1.8.1 Les langages à base de frames

Le "frame" est le fruit des travaux de Minsky et Charniak [13]. Un frame est une structure de données incluant des informations à la fois déclaratives et procédurales. Une ontologie dans le modèle Frame consiste en un ensemble de frames, facettes et slots où les frames représentent les concepts dans le domaine de discours, les slots décrivent les propriétés de ces concepts et les facettes englobent les contraintes que nous définissant sur ces propriétés.

Un exemple de frame est illustré dans la figure1.3 suivante.

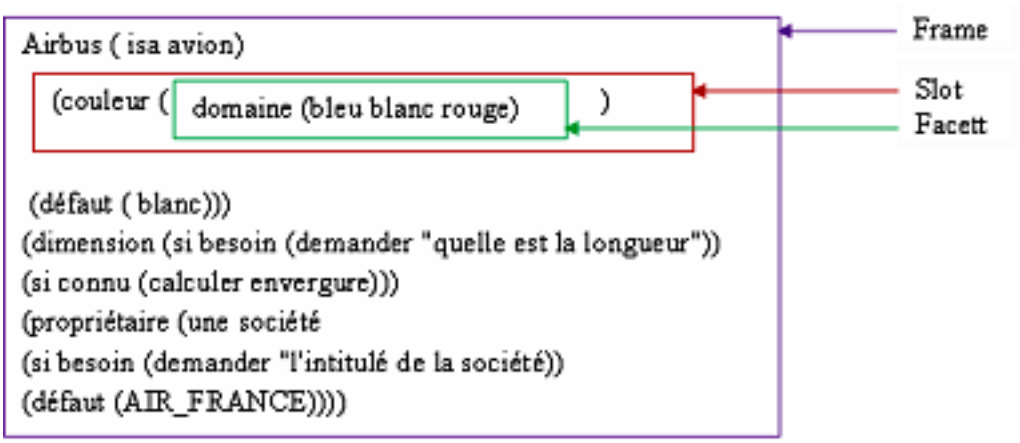

FIGURE 1.3 – Exemple de Frame

1.8.2 La logique de description

La logique de description forme une famille de langages de représentation de connaissances [9] d'un domaine d'application d'une façon structurée et formelle comme la logique de premier ordre. Une principale caractéristique de ces langages est qu'ils sont dotés d'une sémantique formelle [9]. La représentation de connaissances dans ces langages se fait à base de ses entités : les concepts sont des entités génériques d'un domaine d'application et représentent un ensemble d'individus, les rôles représentent des relations binaires entre ces individus et les instances des concepts.

Les connaissances sont prises en compte selon deux niveaux : la représentation des concepts et des rôles relève du niveau terminologique (T_box) et la description des individus relève du niveau factuel ou niveau des assertions (A_box). La figure 1.4 présente une ontologie ecrite en logique de description

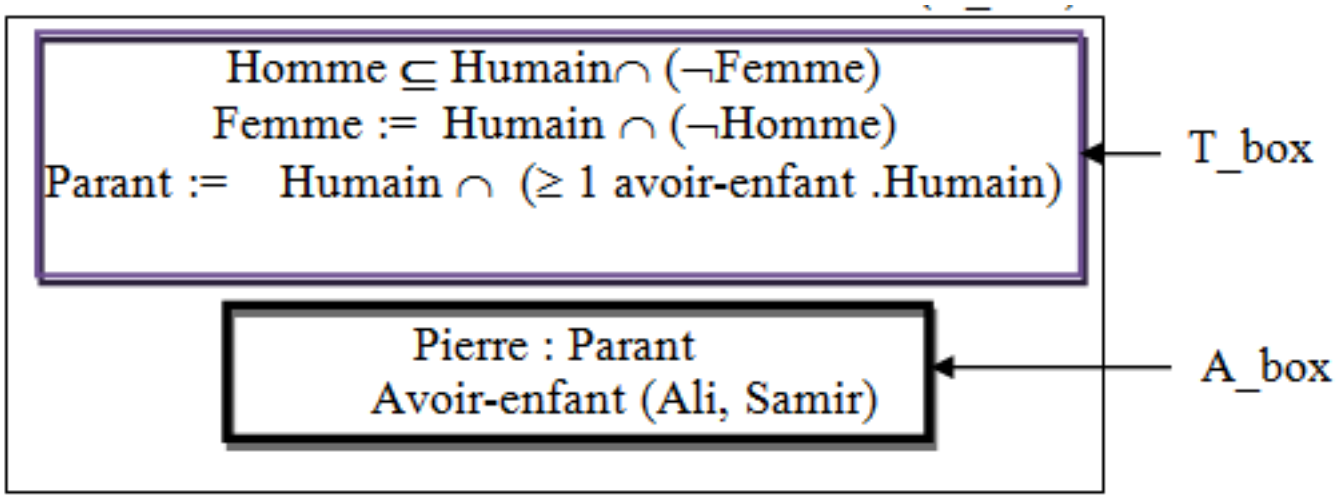

FIGURE 1.4 – Ontologie avec la logique de description

1.8.3 Les graphes conceptuels

Les graphes conceptuels sont des formalismes développés pour représenter facilement les connaissances en langage naturel.

Une ontologie sous forme du graphe conceptuel est un ensemble de concepts et les relations qui existent entre eux. Le mécanisme de base de raisonnement est la projection qui permet de dire si un graphe est plus spécifique (ou plus général) qu'un autre graphe.

La figure 1.5 illustre un exemple d'ontologie de domaine qui regroupe les concepts du domaine de la géométrie sous forme d'un graphe conceptuel.

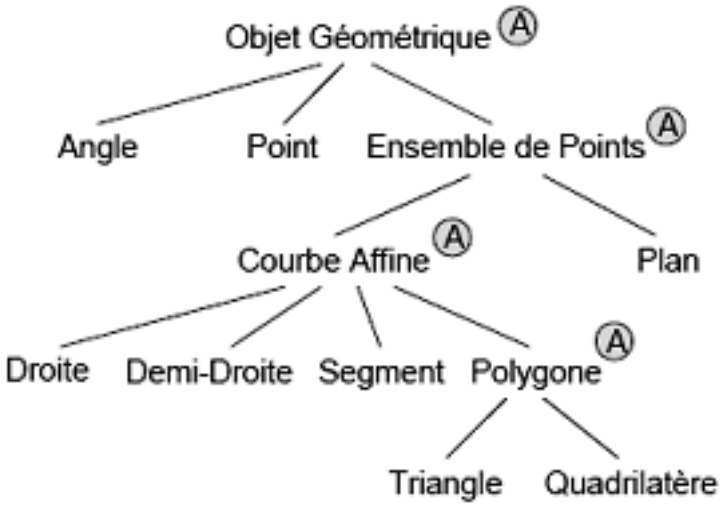

FIGURE 1.5 – Ontologie sous forme d'un graphe conceptuel

1.9 Langages d'ontologies

Différents langages ont étés conçues pour la construction des ontologies. Parmi ces langages,nous citons :

– **RDF** (**R**esource **D**escription **F**ramework) : RDF est un modèle de description basé sur la syntaxe de XML. Il est crée en 1999 et devient une recommandation en 2004 par W3C [11]. Il permet la description des ressources du web et les relations entre ces dernières afin de faciliter leurs traitements par les machines. La description des ressources avec ce langage est basée sur une structure fixe qui est une collection de triplets, chacun est composé d'un sujet (toute ressource avec un URI), d' un attribut et d'un objet qu'est la valeur de l'attribut. Un ensemble de tels triplets est appelé un graphe RDF. Par exemple si nous souhaitons que " l'auteur de http ://www.lacot.org/ est Xavier Lacot " en RDF cela se traduira comme le montre la figue1.6 suit :

```
<?xml version="1.0"?>
<rdf:rdf xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prefixe="http://source_de_schema/schema">
<rdf:Description about="http://www.lacot.org/">
<prefixe:auteur>Xavier Lacot</prefixe:auteur>
</rdf:Description>
</rdf:rdf>
```

FIGURE 1.6 – Exemple d'une déscription à base de RDF

– **RDFS** (**R**esource **D**escription **F**ramework **S**chema) : RDFS offre des primitives pour organiser les objets du web sous forme d'une hiérarchie. Il était introduit comme étant une couche au dessus de RDF [11](basé sur RDF).

– **OWL** (**O**ntology **W**eb **L**anguage) : OWL est une recommandation de W3C depuis février 2004[12]. Il s'agit d'une extension du langage RDF et RDFS. OWL est un langage basé sur la syntaxe XML, très expressif et avec un vocabulaire très riche. Il permet de définir des ontologies de domaines complexes, par exemple, en spécifiant les relations hiérarchiques entre les classes et les restrictions et en plus il permet de faire du raisonnement pour déduire des informations qui ne sont pas explicitement présents dans l'ontologie.
Avec OWL, nous pouvons définir :

1. La subsomption entres classes et relations.
2. Les domaines et les cardinalités des relations
3. Définition des concepts par énumération des instances
4. Définition des concepts par la combinaison des opérations : union, intersection et complément
5. Définition des propriétés et les cardinalités.
6. Définition des contraintes en logique du 1er ordre.

Les langages que nous avons présenté se basent sur la syntaxe XML. L'émergence de nouveaux langages est dûe aux nouveaux besoins et exigences pour définir la sémantique des ressources. Par exemple, avec le langage RDF et RDFS, nous pouvons seulement attribuer une description et une sémantique simple de généralisation-hiérarchies des propriétés et classes. Autrement dit, ces langages sont limités au niveau de l'expressivité. Par exemple :

- RDFS ne permet pas d'exprimer que deux classes sont disjointes. Par exemple, les classes "homme" et "femme" sont disjointes.
- RDFS ne permet pas de créer des classes par combinaison ensembliste d'autres classes (intersection, union, complément). Par exemple, on veut construire la classe "Personne" comme l'union disjointe des classes "des hommes" et "des femmes".
- RDFS ne permet pas de définir de restriction sur le nombre d'occurrences de valeurs que peut prendre une propriété. Par exemple, nous ne pouvons pas dire qu'une personne a exactement deux parents.
- RDFS ne permet pas de définir certaines caractéristiques des propriétés : transitivité (par exemple : estPlusGrandQue), unicité (par exemple : estLePèreDe), propriété inverse (par exemple : mange est la propriété inverse de estMangéPar).

1.10 Conclusion

Comme nous l'avons défini, une ontologie est une conceptualisation formelle d'un domaine particulier. Elle regroupe les connaissances d'un domaine, à savoir les concepts, les propriétés et les relations entre ces derniers.

L'ontologie est un élément pivot pour l'avenir de web sémantique, en ajoutant des métas données sur les ressources de web afin de préciser leurs sémantique et leurs donner du sens.

La croissance continue du nombre d'ontologies dans le web et la façon dont elles sont conçues, à savoir, chacun développe son ontologie selon sa vision et ses besoins, ont ramené à avoir de multiple ontologies dans le même domaine.

L'hétérogénéité des sources d'informations dans le web a posé un vrai problème dans la communication et l'interopérabilité entre les systèmes existants.

Pour remédier à ce problème, surtout de l'hétérogénéité des sources d'informations qui empêchent les différents systèmes d'échanger, de communiquer et de collaborer, les ontologies et leurs mapping sont considérés comme solution.

Dans le chapitre qui suit, nous allons faire une étude détaillée sur le mapping d'ontologies et ses différentes applications, ainsi les problèmes rencontrés et les techniques utilisées pour le faire.

Chapitre 2

Le mapping d'ontologies

2.1 Introduction

L'interopérabilité sémantique entre les sources d'information est une problématique importante en raison du nombre croissant de sources d'information disponibles sur le web et leurs hétérogèniétés.

Le mapping d'ontologies est vue comme une solution prometteuse pour que les systèmes hétérogènes et les applications du web sémantique (où les informations sont présentées par des ontologies) soient interopérables. Il essaie de réduire l'hétérogénéité et permettre ainsi l'échange d'information d'une manière sémantique en créant une couche commune.

Les techniques existantes de mapping d'ontologies tentent de découvrir des relations implicites entre les éléments d'ontologie et de les combiner afin d'obtenir de nouvelles informations.

Avec la croissance continue du nombre d'ontologies disponibles dans le web et leur vaste utilisation dans différentes applications ainsi que la divercité de manière dont elles sont développées où, montre la multiplicité des ontologies pour un même domaine avec différentes terminologies, différents langages de représentations, différents niveaux de granularité et même hétérogènes [16].

Différents domaines métiers dépendent de la qualité et de l'efficacité du processus de mapping à savoir : l'intégration d'information, la composition de services web, les systèmes Peer to Peer et l'ingénierie des connaissances etc.

L'hétérogénéité de ces ontologies due aux différentes manières de représenter le même domaine métier et la nature distribuée, hétérogène et dynamique de ces systèmes ainsi que la non existence d'un système de médiation central et global [17] ont rendu le processus

de mapping difficile car il doit satisfaire des exigences telles que la qualité des résultats et le temps de réponse. Par conséquence, le besoin de développement d'outils automatiques pour le mapping d'ontologies est crucial pour le succès du web sémantique[16].

Dans ce qui suit, nous allons présenter une étude sur le mapping d'ontologies, les problèmes, les difficultés rencontrés dans ce processus ainsi les différents techniques existantes dans la littérature et un état de l'art des algorithmes existants .

2.2 Définition d'un mapping d'ontologies

Le mapping d'ontologies est un nouveau paradigme dans le web sémantique où les ontologies sont utilisées pour la représentation d'information en utilisant un langage de description tel que OWL (Ontology Web Language).

Le mapping d'ontologies peut être vu comme un processus qui relie sémantiquement deux vocabulaires différents [14]. Il tente de découvrir les correspondances sémantiques et les relations implicites qui peuvent exister entre deux ou plusieurs ontologies en appliquant des mesures de similarité (équivalence, disjonction, généralisation, spécialisation etc.) pour les combiner afin de dériver d'autres informations. En générale, les correspondances entre les entités de différentes ontologies peuvent s'établir aussi bien pour la relation d'équivalence que pour d'autres relations telles que la subsumption, la disjonction et la généralisation/spécialisation.

Les entités d'ontologie dénotent soit des classes, des propriétés ou des individus (instances). Cependant, ces entités peuvent être des expressions simples ou complexes comme les formules ou la définition du concept.

Le résultat du processus de mapping est appelé un alignement qui représente l'ensemble des fonctions qui relient les concepts de différentes ontologies.

Une définition de processus de mapping a été proposée par [15], il le considère comme un processus dans une seule direction.

Etant donné deux ontologies $O1$ et $O2$ où $O1$ est appelée l'ontologie source et $O2$ est appelée l'ontologie cible. Le processus de recherche de mapping est appelé la découverte de mapping. Formellement, la fonction de mapping *" map "* peut s'écrire de la façon suivante :

Notons par C les concepts d'une ontologie et par R les relations de cette dernière. Notons par ei_1 et ei_2 des entités avec $ei_1 \in (C(O1(\cup R(O1))$ et $ei_2 \in (C(O2) \cup R(O2))$ $\text{Map}(ei_1, O1, O2) = ei_2$ ou bien $ei_1 \overset{MAP}{\rightarrow} ei_2$.

La cible ei_2 peut contenir une, plusieurs ou aucune entité. Aucune signifie qu'il n'y a pas de mapping pour ei_1 dans $O2$.

Le processus de mapping d'ontologies peut être schématisé comme le montre la figure 2.1 suivante :

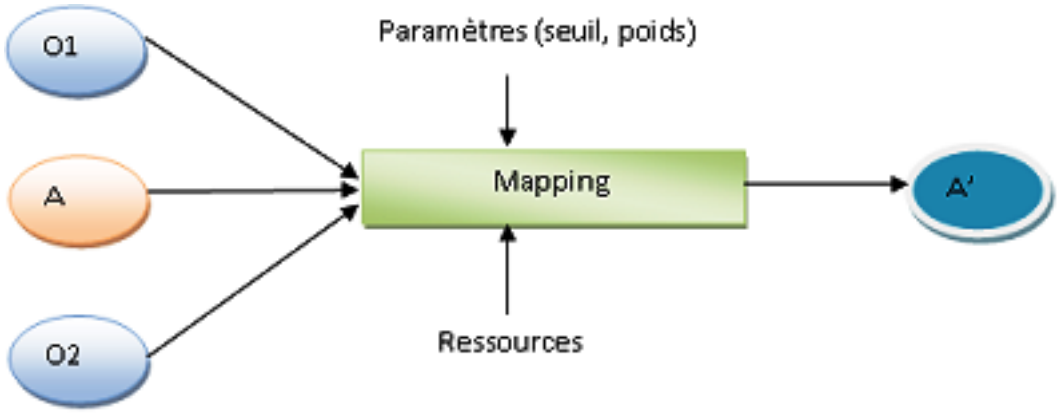

FIGURE 2.1 – Le processus de mapping

Comme illustré par la figure 2.1, le processus de mapping peut avoir un alignement en entrée (A) qui sera raffiné et utilisé pour déduire un autre (A'). Il peut aussi admettre des ressources externes comme les dictionnaires, les taxonomies etc. ainsi qu'un ensemble de paramètres, par exemple le seuil qui permet de prendre seulement les alignements les plus significatifs, les poids qui seront utilisés dans la phases d'aggrégation des résultats.

Il existe six types de cardinalités de mapping d'aprés [15](1 :1), (1 :n), (n :1), (1 :null), (null :1), et (n :m). La table suivante illustre des exemples de ces types de mapping. Par exemple l'entité "Name" correspond à deux entités "First name" et "Last name"

Type de mapping	**O1**	**O2**	**Expressions de mapping**
1 : 1	Faculty	Academic staff	$O1.Faculty = O2.Academicstaff$
1 : n	Name	First name, Last name	$O1.Name = O2.Firstname + O2.Lastname$
n : 1	Cost, Tax ratio	Price	$O1.Cost * (1 + O1.Taxratio) = O2.Price$
1 : null	AI		
null : 1		AI	
n : m	BookTitle, BookaNo,PublisherNo,PublisherName	Book, Publisher	$O1.BookTitle + O1.BookaNo + O1.PublisherNo + O1.PublisherName = O2.Book + O2.Publisher$

TABLE 2.1 – Types des mappings

Dans un mapping entre deux modèles, la correspondance des concepts d'un modèle vers les concepts d'un autre modèle peut être partielle [18]. Ainsi, pour faciliter le mapping, les ontologies doivent être normalisées sous une représentation uniforme afin d'éliminer les différences syntaxiques et de bien faire apparaitre les différences sémantiques entre la source et la cible car le mapping partiel est incomplet en raison de la perte d'information.

2.3 Etapes d'un processus de mapping

D'après l'étude faite par von Dipl and all en 2005 [19], le processus de mapping peut se définir d'une manière généralisée comme étant une suite d'étapes illustrées dans la figure 2.2 suivante :

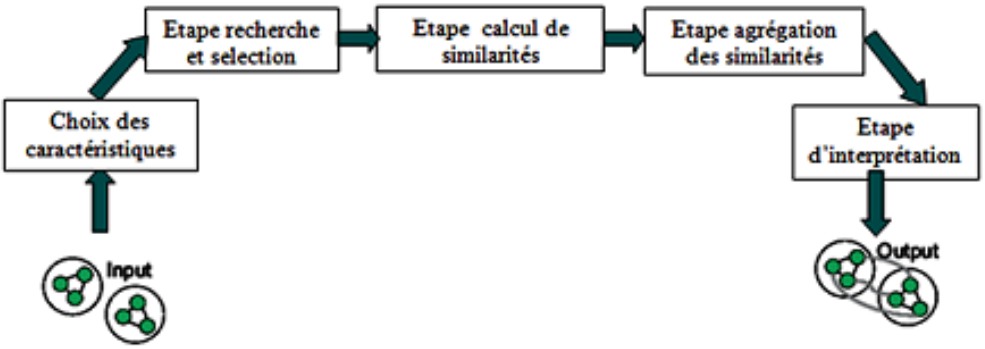

FIGURE 2.2 – Etapes de processus de mapping

- **Input (entrée) :** l'entrée du processus de mapping (matching) est une paire d'ontologies pour lesquelles nous souhaitons établir l'alignement. Il est possible d'avoir un alignement au départ qui aide à trouver d'autres alignements.
- **Choix des caractéristiques :** une partie d'une ontologie consiste à décrire des concepts et les relations qui les relient où ces derniers sont utilisés pour la comparaison. Dans le processus de mapping, nous pouvons considérer toutes les caractéristiques de l'ontologie (concepts, relations, propriétés, instances) qui seront extraites à partir de la définition intentionnelle ou extensionnelle de l'ontologie.
  A titre d'exemple, nous pouvons considérer les labels des concepts, les instances, ses relations avec d'autres concepts (spécialisation, généralisation ...).
  Le résultat de cette étape est l'ensemble des caractéristiques à comparer dans les phases recherche et sélection et calcul de la similarité.
- **Recherche et sélection :** Dans cette étape, le choix des entités qui seront prises en compte durant le processus de mapping est fixé à priori. Par exemple, nous pouvons comparer toutes les entités de l'ontologie sources avec celles de l'ontologie cible ou bien comparer seulement les entités similaires(les concepts avec les concepts , les propriétés avec les propriétés , les relations avec les relations ou les instances avec les instances).
  Le résultat de cette étape sont les entités candidats au calcul de similarité.
- **Calcul de la similarité :** c'est l'application des différentes mesures de similarité choisies sur les entités choisies. En général, ces mesures renvoient des valeurs qui appartiennent à l'intervalle [0,1]. L'application d'une seule mesure produit un matching individuel.
- **Agrégation des similarités :** Les résultats produits lors de la phase précédente (dans le cas d'un matching multiple) sont agrégées en utilisant des fonctions d'agrégation comme le Max, la Moyenne, la somme pondérée, le produit pondéré etc.
- **Interprétation :** l'interprétation utilise les résultats des matching individuels ou

agrégés pour dériver les alignements pertinents entre les entités.

– **Itération :** selon les résultats obtenus, nous pouvons décider de faire une autre itération afin de tirer ou de raffiner les résultats obtenus dans l'itération précédente.

2.4 Domaines d'application

Le mapping est considéré comme une étape primordiale et cruciale dans plusieurs domaines métiers. L'efficacité de ces systèmes dépend directement de l'efficacité du mapping à savoir la pertinence des résultats et sa rapidité dans l'exécution. Parmi ces systèmes, nous citons :

1. **Ingénierie d'ontologie :** Le contexte dont les utilisateurs sont confrontés face aux ontologies hétérogènes est l'ingénierie d'ontologies, plus précisément, la tâche de concevoir, d'implémenter et de maintenir des applications basées sur ces dernières.

   Cette activité nécessite un processus de mapping dû à la diversité des ontologies, leurs natures distribuées ainsi que leurs perpétuelles évolutions.

   Pour créer une ontologie d'un domaine métier, le concepteur doit intégrer différentes ontologies lors de la phase de conception afin de renforcer la réutilisabilité, d'éviter la redondance dans le même domaine et d'assurer la connectivité des ressources.

2. **Intégration d'information :** l'intégration d'information est la plus ancienne classe d'application qui considère le mapping comme une solution probable (souhaitable). Il existe deux sortes d'intégration : L'intégration des schémas et l'intégration de données.

   L'intégration des schémas a pour but de fusionner (d'intégrer) des schémas pour avoir un schéma global où la première étape consiste à trouver les correspondances sémantiques entre les entités des deux schémas avant qu'elles soient émergées. Cette étape est connue sous le nom de mapping.

   L'intégration de données est un processus de combinaison des données appartenant à différentes sources et fournissant à l'utilisateur une vue unique de ces données appelée schéma globale. L'intégration des données se rapporte à un problème combinatoire de données résidentes dans des sources autonomes et hétérogènes. Ce problème est rendue crucial avec la prolifération des sources de données sur Internet ou au sein des entreprises, le caractère hétérogène de ces données et le besoin de plus en plus pressant d'exploiter ces gisements de données pour des besoins décisionnels. Pour intégrer les données de diverses sources d'information, les systèmes d'intégration de données doivent faire face à plusieurs problèmes majeurs comme l'hétérogénéité, la

scalabilité et l'évolution. Comme étape nécessaire pour ce processus (le processus d'intégration de données) le mapping.

3. **Les systèmes Peer-to-Peer :** Le processus de mapping dans les systèmes P2P permet aux différents agents d'interagir et de communiquer d'une manière efficace en créant des mapping d'une manière dynamique entres les ontologies impliquées. Par exemple, le système PDMS (système P2P pour la gestion de données)[20] est un système P2P constitué de pairs autonomes qui communiquent pour répondre collectivement à une requête. La communication entre les pairs se fait grâce au mapping. Une spécificité des PDMS est que chaque pair ne connaît que ses propres connaissances et les mapping le connectant à d'autres pairs afin d'améliorer les réponses fournies globalement par le système tant en quantité et qu'en qualité.
4. **Composition des services Web :** Les services Web sont des processus qui exposent leurs interfaces sur le Web afin que les utilisateurs puissent les invoquer, et qui traite le problème majeur de l'interopérabilité par le protocole SOAP. Les services du web sémantique fournissent un moyen plus riche et plus précis pour décrire les services grâce à l'utilisation des ontologies.

   La découverte et l'intégration des services est le processus qui cherche un service capable d'offrir un service particulier dans le cas d'un matching direct ou de composer différents services dans le cas d'un matching indirect. Cette composition nécessite une phase de découverte de correspondances sémantiques entre les ontologies décrivant les services ; ceci est connu sous le nom de mapping.

2.5 Problèmes de mapping

Dans les systèmes distribués et ouverts, comme le Web sémantique et beaucoup d'autres applications, le problème de l'hétérogénéité des sources d'information est inévitable. Différents concepteurs ont différents intérêts et vues, ils utilisent différents outils et connaissances et avec différents niveaux de détails.

Le mapping d'ontologies permet de réduire l'hétérogénéité entre les ontologies. Cette hétérogénéité ne réside pas seulement sur l'objectif dont ces dernières sont conçues mais aussi sur le formalisme et la terminologie utilisée pour les encoder. La variation des raisons d'hétérogénéité à ramener différentes formes d'hétérogénéité. Plusieurs classifications de l'hétérogénéité ont été proposées, parmi lesquelles nous citons :

✍ **Hétérogénéité syntaxique :** Elle se produit lorsque nous comparons deux ontologies écrites dans deux langages ou modélisées par des formalismes de représentation de connaissances différents [24]. Pour palier ce problème, des outils sont développés pour

transformer une ontologie d'un formalisme vers un autre mais il existe toujours un risque de ne pas préserver le sens et le but de l'ontologie. Les différences que nous pouvons soulever au niveau du langage de représentation sont des différences dans les expressions et la sémantique des langages. D'après [18], il est facile de comparer des concepts s'ils sont exprimés dans le même langage. Un autre exemple d'hétérogénéité est la représentation des instances [15]. Une information peut être présentée sous différentes formes, par exemple la date peut être définie sous le format 12/10/2010 ou sous le format Déc.12. 2010. Un nom peut aussi être représenté sous la forme " Jackson Michael " ou bien sous forme " Michael Jackson"

✍ **Hétérogénéité terminologique :** le problème majeur dans le mapping est surtout celui de la terminologie car nous pouvons avoir une multitude de manières de représenter des données similaires et parfois des conflits sur les appellations. A titre d'exemple, l'utilisation de termes linguistiques identiques pour désigner différents concepts ou relation ou l'utilisation de différents termes pour désigner un même concept ou une même relation [21]. Il y a aussi le problème de polysémie où un concept peut avoir différents sens. Ce type d'hétérogénéité pose, par conséquence, une grande difficulté. Pour diminuer cette difficulté, des ressources externes sont utilisées comme les dictionnaires, les thésaurus etc.

✍ **Hétérogénéité conceptuelle :** Elle englobe les différences dans la modélisation du même domaine d'intérêt. D'après l'étude faite dans Jérôme Euzenat et Pavel Shvaiko en 2007[14], trois types d'hétérogénéité conceptuelle peuvent se produire selon le degré de couverture de domaine et le but pour lequel l'ontologie est conçue :

1. Différence dans l'assurance : Plusieurs ontologies peuvent décrire le même domaine mais avec des manières différentes et avec le même degré de granularité.
2. Différence dans la granularité : De multiple ontologies peuvent présenter le même domaine mais avec des niveaux de détails différents. Dans ce cas, nous pouvons trouver une ontologie O2 comme une spécialisation d'une ontologie O1.
3. Différence dans le but : Chaque ontologie a un objectif à atteindre et la différence se produit lorsque les ontologies décrivent le même domaine avec le même niveau de granularité mais pour des objectifs différents.

✍ **Hétérogénéité sémantique :** Elle concerne la façon dont les entités sont interprétées par les personnes. En effet, des entités qui ont exactement la même sémantique sont souvent interprétées différemment selon le contexte d'utilisation. Par exemple, la façon dont elles sont utilisées. Ce type d'hétérogénéité est très difficile à détecter par la machine et difficile à résoudre.

2.6 La recherche des correspondances

Avant de définir les différentes techniques utilisées pour la découverte des correspondances entre ontologies, nous allons définir les concepts de correspondance et de similarité entre concepts.

2.6.1 Définition d'une correspondance

Soient deux ontologies O et O' où E et E' sont les ensembles des entités de O et O' respectivement. Une correspondance doit prendre en considération les deux entités et la relation qui les relient. Une correspondance dans [14] est présenté par un cinq-uplet comme suit : <Id, E, E, R, N > où

Id : identificateur unique pour la correspondance
E : entité de l'ontologie O
E' : entité de l'ontologie O'
R : la relation entre les deux entités, elle peut être une relation d'équivalence, de subsumption, d'inverse ...etc.
N : valeur appartienne à l'intervalle de confiance généralement entre [0,1] qui indique le degré de semblance ou de différence entres les entités.

Le but principal des algorithmes de mapping est la découverte des relations sémantiques entre les entités de différentes ontologies. Généralement, c'est la relation d'équivalence qui est découverte par l'application des différentes mesures de similarité entre les entités. Différentes définitions de similarité sont proposées dans la littérature, on peut citer :

Définition 1 : cette définition est proposée en se référant au degré de similarité entre entités.

La similarité $\sigma : o \times o \longrightarrow R$ est une fonction entre deux entités qui renvoie un réel qui estime le degré de similarité entre les entités.

Définition 2 : un autre critère pour estimer la similarité entre entités est le de degré de dissimilitude, c'est une fonction $\sigma : o \times o \longrightarrow R$ qui admet deux entités en entrée et revoie un réel qui donne le degré de dissimilitude. La dissimilitude entre entités peut être estimée par la distance. En conséquence, la dissimilitude est proportionnelle à la distance. La plus part des mesures de (dés) similarité sont normalisées [24], leurs résultats appartiennent à l'intervalle [0,1]. Cette normalisation a faciliter la comparaison de similarité de différents

type d'entités. A chaque mesure de similarité normalisée x, nous avons une valeur de dissimilitude qui est (1-x).

2.6.2 Techniques de mapping d'ontologies

Le but de processus de mapping est de trouver les entités correspondantes entre deux ontologies. Plusieurs méthodes et techniques ont été proposées dans la littérature pour l'alignement, basées sur des critères différents. Deux classifications de ces techniques sont proposées [14] :

- **Classification basée sur la granularité et l'interprétation des entrées :** elle se base sur la granularité c-à-d il s'agit de savoir si les matchings considèrent le niveau élément ou structurelle et comment elles interprètent généralement les informations en entrée.
- **Classification sur les types d'entrées :** elle se base sur les types d'entrées que ces techniques utilisent.

La classification des matchers selon la granularité et l'interprétation des entrées peut se faire en se basant sur ces niveaux d'interprétations suivants :

✍ **Niveau élément et niveau structurelle :** les techniques de matching au niveau élément calculent les correspondances entre entités ou instances en isolation, en ignorant ses relations avec d'autres entités ou instances. Le niveau structurel, les calcule en comparant les relations des entités avec d'autres (considération des relations hiérarchiques des entités).

✍ **Syntaxique, externe et sémantique :** la caractéristique principale des techniques syntaxiques est qu'elles interprètent les entrées par rapport à leurs structures. Les techniques externes sont basées sur des ressources auxiliaires de domaine et de connaissance commune pour interpréter les entrées ; ces ressources peuvent êtres humaines ou des thésaurus qui expriment les relations entre les termes. Les techniques sémantique utilisent des modèles sémantiques par exemple model-theoretic semantics, pour interpréter les entrées et justifier les résultats.

La classification qui est basée sur le types des entrées, elle concerne la manière dont les techniques considèrent ces entrées, à savoir des algorithmes opérants sur des Strings (terminologique), sur des structures, sur des modèles sémantiques ou des instances (extensionnel)

Les techniques de mapping (matching) sont des techniques qui se basent sur différents niveaux d'interprétations et de types des entités des ontologies en entrée. Elles consi-

dèrent le niveau terminologique et structurel des données. Ces deux types de données sont présents déjà dans la définition de l'ontologie. Le niveau sémantique requiert une interprétation sémantique, utilise généralement des raisonneurs pour déduire la correspondance. Le niveau instance représente l'actualité des algorithmes de mapping où deux concepts avec les mêmes instances sont considérés similaires.

2.6.2.1 Méthodes terminologiques

Elles considèrent les entités comme des Strings et en isolation. Elles se basent généralement sur les labels assignés aux entités de l'ontologie car les ressemblances basées sur les labels sont essentielles puisque deux entités qui portent le même nom sont équivalentes [26]. Il existe plusieurs méthodes, celles qui se basent sur l'analyse des morphèmes (préfix et stem) et celles qui se basent sur des formules pour trouver la distance entre les labels et d'autres utilisent des ressources linguistiques externes comme des dictionnaires et thésaurus.

Ces méthodes généralement nécessitent une phase de normalisation afin d'éliminer les mots vides (conjonction, propositions, chiffres ...) et de les mettre sous un format uniforme. Parmi les méthodes existantes dans la littérature considérant le niveau terminologique, nous citons :

- **Egalité des String :** c'est une mesure de similarité stricte où tous les caractères char(x) à la position x doivent être identiques entre les deux termes c et d. sa formule est la suivante :

$$Sim_{streq}(c,d) = \begin{cases} 1 & \text{si } \forall i \in [0,|c|] \quad c.char[i] = d.char[i] avec |c| = |d| \\ 0 & \text{sinon} \end{cases}$$

- **La distance de Hamming :** C'est une distance calculant le nombre de position où les deux Strings sont différents. Elle est donnée par la formule 2.1 suivante

$$\delta(s,t) = (\sum_{i=1}^{min(|s|,|t|)} s[i] \neq t[i] + (||s| - |t||)/max(|s|,|t|) \tag{2.1}$$

- **Edit Distance :** elle est également une dissemblance qui calcule le cout minimum du nombre d'opérations (insertion, suppression, substitution) que nous pouvons appliquer sur un String pour obtenir un autre. Différentes mesures basées sur la distance, telle que la distance de Levenshtein où le coût de chaque opération est 1, de Needleman-Wunch qui donne le coût le plus élevé pour l'opération d'insertion et de suppression.

– **La similarité N-gram :** une méthode est basée sur les sous séquences pour déterminer la similarité entre deux Strings. Elle détermine le nombre de sous séquences communes de longueur N entre deux Strings. La fonction normalisée de cette méthode est donnée en 2.2 où ngram (s,n) et ngram (t,n) est l'ensemble des sous séquences des Strings s et t respectivement.

$$\sigma(s,t) = \frac{|ngram(s,n) \cap ngram(t,n)|}{\min(|s|,|t|) - n + 1} \tag{2.2}$$

– **La similarité Synonymy :** C'est une mesure utilisant une ressource linguistique externe comme un dictionnaire (Word net). Elle prend deux termes s et t en entrée et une ressource de synonymie et renvoie une valeur : 1 cas où les deux termes sont équivalents sinon un 0

2.6.2.2 Méthodes basées sur la structure

La structure existante entre Les entités del' ontologie peut être utilisée pour la comparaison. Cette comparaison peut être faite sur la structures internes et externes des entités. Ces méthodes sont similaires à celles utilisées dans le matching de schémas de base de données et des documents XML. Généralement, elles s'appuyaient sur des heuristiques et non pas des mesures exactes.

Méthodes internes : Les méthodes basées sur la structure interne peuvent prendre comme critères les attributs des propriétés et leurs domaines, les cardinalités, la transitivité et la symétrie des propriétés etc pour calculer la similarité entre elles.

Par exemple, nous pouvons comparer le type des propriétés en comparant la manière dont elles sont stockées en mémoire (entier, float,...) ou leurs domaines qui caractérisent un ensemble de type bien particulier ([10,20), des énumérations...). L'approximation entre deux types de données doit être maximale dans le cas où les types sont identiques ou lorsque l'un est inclut dans l'autre (exemple d'entier et réel sont deux types compatibles), minimale dans le cas où les deux types sont incompatibles.

La comparaison des domaines dépend du type d'entités considérées. Par exemple, nous utilisons les domaines des classes ou les valeurs des instances. Le domaine peut être structuré comme étant un ensemble ou une séquence. A titre d'exemple, nous pouvons comparer les domaines de la propriété âge appartenant à des classes différentes comme "Ecolier", "Adolescent" et "Adulte".

Méthodes externes : Différentes méthodes basées sur la hiérarchie ont été proposées dans la littérature pour calculer la similarité entre deux entités de deux ontologies. Si deux entités sont similaires cela entraine que leurs voisins(les parents, les fils...etc.) sont

forcement semblables. Cette remarque peut être exploitée de différentes manières et cela selon le type de la relation considérée.

Différents critères peuvent décider sur la similarité de deux entités à savoir :

- Leurs directes superclasses ou toutes les superclasses de ces deux entités.
- Leurs directes sous classes où toutes les sous classes de ces deux entités.
- Tous les descendants de ces deux entités dans le sous arbre où ces dernières sont des racines.
- Toutes les feuilles des sous arbres dont leurs racines sont les deux entités.
- Toutes les entités situées sur les chemins menant de la racine à ces deux entités en question.
- Une combinaison de ces différents critères.

Beaucoup de méthodes ont été proposées basées sur la structure taxonomique des deux ontologies. La plupart de ces méthodes se basent sur le calcul du nombre d'arcs séparant les deux entités dans une taxonomie. Voici quelques méthodes :

- **Structural topological dissimilarity on hierarchies :** Cette approche calcule une dissimilitude sur une hiérarchie $H = (O, <)$ où O est une ontologie et $<$ est la relation hiérarchique (is-a) entre les classes de O. Cette méthode renvoie la longueur du chemin le plus court entre deux entités e et é. Sa formule est donnée en 2.3 comme suit :

$$\forall e, e' \in O, \delta(e, e') = \min[\delta(e, c) + \delta(e', c)] avec c \in O \tag{2.3}$$

  Notons que $\delta(e, c)$ est le nombre d'arcs intermédiaires entre l'entité e et une entre entité c dans la même hiérarchie H.
- **Wu-Palmer similarity :** Cette méthode utilise la structure hiérarchique de l'ontologie pour déterminer la similarité sémantique entre les concepts. Soit une ontologie O contenant un ensemble de nœuds et une racine r comme illustré par la figure 2.3. Le principe de calcul de similarité entre deux nœuds X et Y avec cette méthode se base sur les distances N1 et N2 qui séparent les nœuds X et Y du nœud racine R et la distance (N) qui sépare le concept subsumant (CS) de X et de Y du nœud r.
  La similarité de Wu et Palmer est définie par la formule suivante et la figure2.3 illustre comment se fait le calcul de cette similarité entre deux concepts X et Y :

$$Sim(X, Y) = \frac{2 \times N}{N1 + N2} \tag{2.4}$$

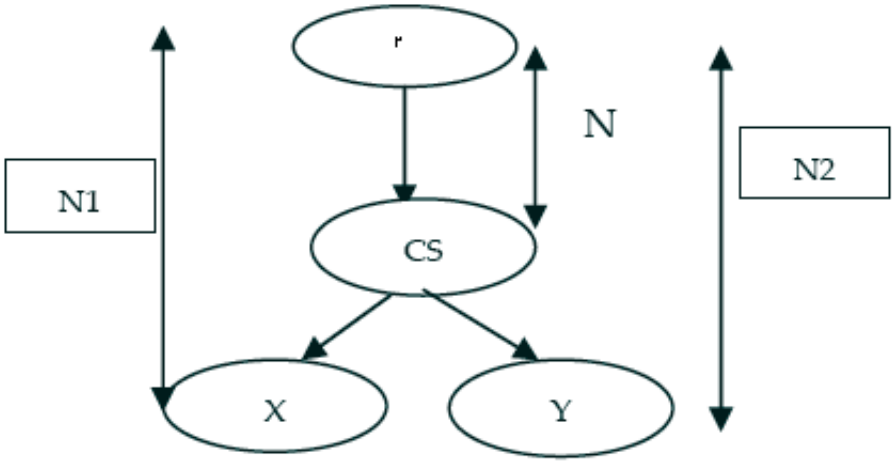

FIGURE 2.3 – La mesure de Wu et Palmer

– **Upward cotopic similarity :** Cette similarité s'opère également sur une hiérarchie H=(O,<) et utilise la similarité de Jaccard. La similarité entre les entités C et C' dans O est calculée par la formule 2.5 Où UC(C,H) et UC(C',H) sont l'ensemble des superclasses de l'entité C et C' respectivement.

$$\sigma(C, C') = \frac{|UC(C, H) \cap UC(C', H)|}{|UC(C, H) \cup UC(C', H)|} \tag{2.5}$$

Ces différentes mesures ne peuvent pas être directement appliquées dans le matching d'ontologies car les ontologies ne sont pas censées partager la même hiérarchie mais une alternative est d'utiliser une ressource externe comme Word Net ou une troisième ontologie.

2.6.2.3 Méthodes basées sur les instances

Dans le cas où les instances des clases sont présentes dans les ontologies, la similarité entre classes peuvent être calculée en se basant sur ces dernières. Différentes méthodes ont été introduites pour le calcul de similarité entre les classes en se basant sur les instances. La comparaison entre deux classes dans ce cas est facile et cela en calculant l'intersection entre les deux ensembles d'instances A et B qui correspondent aux classes $c1 \in O1$ et $c2 \in O2$ respectivement (O1 et O2 sont deux ontologies à aligner).

A partir de cette intersection, nous pouvons conclure que si les deux classes C1 et C2 sont similaires ($A \cap B = A = B$), plus générale alors ($A \cap B = BouA \cap B = A$) et dissimilaire dans le cas où aucun de ses cas ne figure. Différentes méthodes sont applicables dans ce cas, à savoir :

– **Hamming distance entre deux ensembles :** Cette méthode prend deux ensemble d'instances en entrée A et B et renvoie un nombre appartient à l'intervalle

[0,1]. Sa formule est donnée en 2.6 comme suit :

$$\sigma(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \tag{2.6}$$

– **La similarité de Jaccard :** cette méthode est basée sur la probabilité pour tirer la similarité. Soit deux ensemble A et B et soit P(x) la probabilité qu'une instance appartient à l'ensemble A. le calcul de la similarité avec cette méthode donne 1 cas où A= B et 0 cas où $(A \cup B) = \emptyset$. Sa formule est donnée en 3.2 comme suit :

$$\sigma(A, B) = \frac{P(A \cup B)}{P(A \cap B)} \tag{2.7}$$

2.6.2.4 Méthodes sémantiques

La caractéristique principale des méthodes sémantiques est qu'elles admettent un modèle-théorétique qui est employé pour justifier leurs résultats. Autrement, se sont des méthodes déductives, par exemple des propositions de satisfiabilité ou des techniques basant sur la logique de description. Ces méthodes généralement nécessitent une phase où les entités déclarées similaires sont connus.

2.7 Etat de l'art sur les algorithmes de mapping

L'interopérabilité des systèmes hétérogènes dans le web peut s'achever par un agrément entre ontologies. Différentes travaux ont été effectués afin de résoudre le problèmede communications et d'interactions entres les différentes entités des systèmes. Chaque système a une manière particulière pour faire le mapping. Une classification de ces systèmes peut se faire en considérant différents facteurs comme les entrées considérées par les systèmes à savoir le type des ontologies, les alignement, le type de mapping produit (cardinalités), le but pour lequel ce système a été conçue, l'utilisation des ressources externes comme les dictionnaires et les thésaurus et enfin une classification peut se faire selon les caractéristiques prises pour le calcul de similarité : les concepts, la structure, les instances ou une combinaison ou bien selon le degré d'automatisation (automatique ou semi automatique).

1. **QOMFDE :** L'algorithme proposé par Isaac Lera et all [26] admet deux ontologies en entrée écrites en OWL. Les deux ontologies en entrées sont enrichit par une structure. Cette structure contient seulement les éléments les plus significatifs de chaque ontologie. Ces éléments préservent le niveau de granularité et l'objectif pour lequel l'ontologie est conçue. Le choix de ces éléments se base sur les propriétés et les relations hiérarchiques des concepts. Deux formules sont utilisées.

$$W_i^H = \frac{H_i * C_i}{D_i} \tag{2.8}$$

$$W_i^R = R_i^o ut * \frac{R_T otal}{\sum R_i n} + \frac{R_i^i n}{R_T otal} \tag{2.9}$$

La formule 2.8 affecte un poids (W^H) pour chaque concept en considérant trois variables : la position du concept dans l'hiérarchie(H_i), le nombre des fils (C_i) et sa profondeur(D_i). La formule 2.9 affecte un poids (W^R) pour les relations entre les concepts en considérant les cardinalités et les domaines des propriétés.

Une fois les éléments les plus significatifs sont pris (la structure est définie) , chaque élément de cette structure est enrichi par des ensembles de termes comme les synonymes, les hyperonymes et instances.

L'enrichissement se fait à partir d'une ressource externe qui est Word net dans le but d'augmenter la probabilité de coïncidence dans le processus de mapping. Notons bien que la construction de cette nouvelle structure est fait durant le développement de l'ontologie (phase de conception).

QOMFDE ressoud deux problèmes majeurs dans les environnements distribués et mobiles. Le premier problème est la multiplicité de la représentation du domaine par la création d'une structure qui synthétise l'objectif et le degré de granularité des ontologies à comparer. Le deuxième problème est la quantité de données à comparer par la réduction du nombre de comparaison à effectuer.

Cette approche change le processus de développement d'ontologies. La création de la nouvelle structure consomme du temps dans la phase de conception et de l'espace mémoire, mais elle offre un gain dans la rapidité et la simplicité du processus de mapping ; ce qui facilite l'échange de données dans tels systèmes.

La phase d'agrégation des similarités est ignorée car elle donne une mauvaise qualité.

2. **GLUE :** GLUE [29]est un système basé sur une approche probabiliste et semi automatique dans la découverte des mapping sémantique entre deux ontologies (plus précisément deux taxonomies) en utilisant différentes techniques d'apprentissage (machine Learning techniques). L'entrée du processus de mapping est deux ontologies O1 et O2 et un ensemble d'instances. GLUE considère les instances d'un concept comme des instances de son prédécesseur.

   Il découvre des correspondances de cardinalité (1,1) entre les deux taxonomies. C'est-à-dire, pour chaque concept d'une taxonomie, il tente de trouver le concept le plus similaire dans l'autre. GLUE utilise la notion de la distribution de probabilité conjointe (joint probability distribution) d'un concept pour avoir la possibilité

d'appliquer n'importe quelle mesure de similarité qui peut s'exprimer en fonction de cette notion.

La distribution de probabilité conjointe entre deux concepts A et B consiste en P(A,B), P($\neg A$, B), P(A, $\neg B$) et P($\neg A$,$\neg B$).

P(A,B) est la probabilité qu'une instance de domaine appartienne au concept A et au concept B.

GLUE applique différentes mesures de similarité comme : le coefficient de Jaccard $P(A \cap B)/P(A \cup B)$, la similarité du parent le plus spécifique $MSP(A, B) = P(A|B)$.

GLUE s'exécute en trois étapes. La première étape est le calcul de la distribution de probabilité conjointe de chaque paire de concepts (A, B) sachant que $A \in O1$ et $B \in O2$ par l'application de différentes techniques d'apprentissage.

Dans la deuxième étape, GLUE utilise les résultats de l'étape précédente pour calculer la similarité entre les différents pairs de concepts en appliquant différentes mesures de similarité. Le résultat de cette étape est une matrice de similarité entre les concepts de deux taxonomies. Dans la dernière étape, il utilise la matrice de similarité et les contraintes spécifiques au domaine ainsi un ensemble d'heuristiques pour tirer les mapping pertinents, par exemple deux concepts se correspondent si leurs voisinages se correspondent aussi(les voisinages sont les fils, les parents ou les deux à la fois).

Donc GLUE est extensible en termes de mesures de similarité qui peuvent se définir en termes de probabilité. Il utilise le contenue informationnel (instances) des concepts afin de découvrir les correspondances mais il est très limité dans son application par la considération d'un seul type d'ontologies qu'est la taxonomie.

3. **FalconAO(Aligning Ontologies with Falcon)[25] :** Falcon AO est un système automatique pour l'alignement d'ontologies écrites en OWL dans le but de rendre les différentes applications du web sémantique interopérables. Il est constitué de 5 modules qui sont :
   - **Model Pool :** Les deux ontologies en entrée sont transformées sous forme de modèles en utilisant l'API Jena et un ensemble de règles afin d'éliminer les axiomes inutiles et de réduire l'hétérogénéité structurelle.
   - **Matcher Library manages :** Il est composé de 4 matchers : V-Doc et I-Sub qui sont deux matchers linguistiques, GMO matcher se basant sur la structure pour tirer les mapping et PBM adopte une stratégie de division pour trouver les blocs de mapping entre des ontologies à grandes échelles.
   - **Alignment Se :** est un module pour générer et évaluer les alignements écrits sous format RDF/XML en utilisant des métriques conventionnelles : la précision et le rappel.

- **Central Controller :** il permet une configuration manuelle des stratégies de mathings et exécute les matchers puis combine les similarités produites à partir des comparaisons linguistiques et structurelles.
- **Repository :** stocke les données réutilisables au cours du processus de mapping.

4. **QOM [27] :** QOM est un système semi automatique pour la génération des mapping entre deux ontologies. Il était conçue pour achever une interopérabilité entre les agents ou les services utilisant différentes ontologies.

   Il supporte en entrée de légères ontologies écrites en OWL. Il transforme les deux ontologies en entrée en un modèle RDF (ensemble de statements).

   Le but de QOM est de proposé une approche efficace en terme du temps d'exécution. Il considère que le facteur qui influence le plus sur la complexité du temps d'exécution est le nombre de comparaisons à effectuer pour avoir les meilleurs mappings. Afin de réduire ce nombre, QOM utilise des heuristiques et prend en compte deux structures de données : la première est les mappings candidats à étudier et la deuxième est un agenda qui ordonne les mappings candidats et écarte certains d'eux dans le but de gagner de l'efficacité. QOM compare toutes les entités(concepts, propriétés, relations et instances) et applique une variété de mesures de similarités en considère différentes caractéristiques des entités à aligner comme illuster par le tableau 2.2 suivant :

Entités	**caractéristiques**
Concepts	les labels ,les URIs, les relations , les super concepts, les sous concepts, propriétés directes, les properties des directs super-concepts, les instances des sous concepts
Relations	les labels, les URIs, les domaines des relations , super propriétés directes, sous properties directes , les instances des propriétés
Instances	les labels , les URIs ,les relations, les super concepts, les instances des propriétés
Les instances des propriétés	les domaines , les ranges , les propiiétés des super concepts

TABLE 2.2 – Les caractéristiques des entités comparées

Les mappings produits par QOM est de cardinalité (1,1). Il obtient une complexité de $O(n*(log(n))$ au lieu de $O(n^2)$ par la réduction du nombre des mapping candidats pour trouver les mappings pertinents. QOM suppose que la perte de la qualité des résultats est marginale. Cependant, l'amélioration dans l'efficacité peut être énorme par la réduction du nombre de comparaison.

La qualité des mapping produits en présence des axiomes et des propriétés sémantiques est très faible.

5. **MAFRA [22] :** est un système de mapping d'ontologie exécuté pour la transformation de données dans le web sémantique. Il était défini d'une manière à couvrir toutes les étapes de processus de mapping. Il est composé de 5 modules horizontaux et 4 verticaux. Les modules horizontaux résument les étapes fondamentales de processus de mapping et les modules verticaux interagissent avec les horizontaux afin d'offrir des services.

   La première étape exécutée par MAFRA est l'étape la normalisation qui consiste à mettre les deux ontologies en entrée sous une représentation uniforme. Le but de cette étape est de minimiser les différences syntaxiques et de faire apparaitre les différences sémantiques entre les deux ontologies par exemple éliminer les mots vides, l'extension des acronymes etc. L'étape qui se suit est la découverte des similarités. MAFRA applique de multiple stratégies. Il commence par le calcul des similarités lexicales entre toutes les entités de l'ontologie sources avec toutes les entités de l'ontologie cible avec algorithme Resnik [23] et Word net puis il les calcule en se basant sur les propriétés (attributs et relations) des concepts.

   Une fois le calcul de similarité est achevé, vient la phase de Semantic Bridging qui consiste à établir les correspondances entre les entités de l'ontologie source et cible et

définir les transformations requises pour transformer une instance de source vers une instance la plus similaire dans la cible. Donc, le " Semantic Bridging " est le module le plus important dans MAFRA, il décrit les entités qui seront reliés sémantiquement entre les deux ontologies et comment les instances de la source sont transformées en instances de la cible ainsi les conditions à tenir en compte lors de son exécution. Pour relier les entités, Semantic Bridging utilise un ensemble d'heuristiques. Par exemple si les concepts de l'ontologie cible sont des hyperonymes d'un concept de l'ontologie source, cela signifie que une instance du concept source peut se translater à l'une des instances de l'ontologie cible.

Enfin la dernière étape est l'exécution de Semantic Bridging qui consiste à la transformation des instances de l'ontologie source vers les instances de l'ontologie cible ainsi le Post-processing qui prend les résultats d'exécution et prouvent leurs qualités c.-à-d dire que deux instances ont le même sens.

Parmi les services offerts par les modules verticuax, nous avons le module Evolution qui se focalise sur la maintenance du " Semantic Briding " en fonction des changements dans l'ontologie source et cible (évolution des ontologies), le module Cooperative Consensus Building responsable d'établir un consensus entre les deux communautés participantes dans le processus de mapping en choisissant un mapping parmi ceux obtenus, ainsi que les contraintes du domaines et les connaissances supplémentaire comme les thésaurus et les dictionnaires afin augmenter la chance de trouver des mapping.

Une interface graphique de MAFRA permet à l'utilisateur de comprendre les deux ontologies.

6. **OLA (OWL Lite Aligner) [28] :** est un système de mapping d'ontologies OWL. Il transforme les deux ontologies en un graphe étiqueté. Le graphe contient les différentes entités de l'ontologie : classes, objets, relations, propriétés, instances des propriétés etc. Les relations existantes dans ce graphe sont : la spécialisation (entres classes ou relations), l'instanciations (entres une classe et un objet, entre une propriété et propriété instance, entre data types et valeur), l'attribution (entre une classe et propriétés et entre objet et les instances des propriétés) et la restriction.

   Pour déterminer la similarité entres les entités, OLA considère toutes les caractéristiques possibles des entités (Labels, Noms, les voisins (subsomption, généralisation), instances et les cardinalités) et applique un ensemble de fonctions de similarités pour chaque catégorie d'entités. Enfin, il extrait l'alignement à partir des matrices des similarités de chacune des catégories.

7. **MapPSO(Ontology Mapping by Particle Swarm Optimisation)** : MapPSO est algorithme de mapping d'ontologies. Il considére le probleme de l'alignement

comme étant un problème d'optimisation et applique un algorithe de *discrete particle swarm optimisation*(PSOD) pour le calculer[30].
Son objectif est de trouver un alignement raisonnable et de maximiser le nombre des correspondances.

Le développement de Taxomap est motivé par les observations suivantes :
- Les ontologies accroîtrent en nombre et en taille
- Les ontologies évoluent progressivement
- Les ontologies représentent différentes caractéristiques qui peuvent être utilisés pour le calcul de l'alignement

DPSO consiste en un ensemble de particules (soit N le nombre de particules). Chaque particule est initialisée par un alignement initial appelé configuration. Puisque MapPSO produit un alignement de cardinalité (1,1), il estime que le nombre des correspondances qu'il peut obtenir est égale à $n = Min(|C1|, |C2|)$ où $C1$ et $C2$ sont le nombre d'entités dans chacune d'ontologies en entrée.

MapPSO s'opèrent sur plusieurs itérations pour avoir l'alignement optimal et pour chaque itération, il tire un nombre aléatoire $nb \in [1, n]$. Puis, chaque particule sélectionne aléatoirement des entités à partir des deux ontologies qui ne sont pas déjà sélectionnées et ajoute cette correspondance à l'alignement initial, puis il applique une fonction de similarité pour calculer la similarité entre ces deux entités. Noter que cette opération est exécuté nb fois, autrement dit, il ajoute nb correspondances à l'alignement initial. Après, une fonction d'évaluation de l'alignement produit F(la somme des différentes valeurs calculer pour chaque correspondances) est appliquée pour chaque particule afin de prendre le meilleur alignement local. Dans ce cas, il applique la distance Levenshtein pour chaque correspondance et la plus grande valeur de F est désigné comme étant le meilleur alignement local.

Une mise à jour de l'alignement global est faite par rapport aux meilleurs alignements locaux des particules durant chaque itération.
La mise à jour d'une particule se fait comme suit : Pour assurer une convergence guidée vers un alignement optimal pendant les itérations, il a limité l'influence de la réinitialisation aléatoire de chaque particule. En effet, il augmente la probabilité de préserver les correspondances des particules les mieux évaluées et celles présentent dans le meilleur alignement local et global par l'ajout des paramètres comme suit : Pour chaque particule un vecteur V est associe. Initialement, pour chaque correspondance dans ce particule une valeur 1 lui correspond dans le vecteur V et sa même pour les nouvelles correspondances qui joindront la particule durant son évolution. La mise à jour des particules est faite en deux étapes : premièrement si la correspondance est présente dans le meilleur alignement local, il ajoute un paramètre β à

sa correspondance dans le vecteur et s'elle est présente dans le meilleur alignement global, il ajoute un paramètre γ. Après cela il multiplie le vecteur V par un nombre $\phi \in [0, 1]$. Afin que la particule ne soit pas remplacer par une réinitialisation aléatoire, il a introduit deux ensembles F,V : le premier contient les correspondances ordonnées selon leurs similarités et le deuxième, les correspondances sont ordonnées selon le vecteur V. L'intersection K entre ces deux ensembles (F, V) contient les meilleures correspondances. Pour une convergence plus rigoureuse vers un alignement optimal, il a introduit un autre ensemble S qui contient les correspondances qui ne seront pas remplacés durant les itérations suivantes. Le choix de ces correspondances est pris par rapport à un paramètre σ. Toutes les correspondances dont leurs évaluations est inférieur à σ sont pris. Cet algorithme est implémenté sous java et l'API alignapi. Il est été évalué par l'organisme OAEI en 2010 avec les différents jeux de données proposés en termes de Précision et le Rappel. Il obtient une moyenne de 0,60 et 0,68 respectivement. L' avantage que présente cet algorithme est qu'il a formulé le problème de l'alignement comme étant un problème d'optimisation à deux objectifs qui sont : le premier est de trouver l'alignement le plus raisonnable et le second est de maximiser le nombre des correspondances. Un autre avantage est qu'il ne compare pas directement toutes les entités des ontologies mais il procède d'une manière itérative en choisissant un ensemble aléatoire des entités à comparées, ce que lui permet d'éviter de calculer de grande matrices. Il est aussi fortement parallélisable ce qui règle les problèmes d'exécution ou de mémoire. Les limites de cet algorithme est qu'il considère seulement les labels des entités et applique seulement des méthodes opérant sur les Labels comme String distance, il ne considère pas la structure et les axiomes présent dans les ontologies. Ainsi que le nombre d'itérations et les particules qui vont participer à trouvé l'alignement optimal sont pas optimal mais se sont des données a introduire ainsi que d'autres paramètres qui sont introduit d'une manière non justifié qui pourront etre prit avec plus de précision.

8. **Taxomap [31] :**Taxomap est un système pour l'alignement des ontologies OWL, plus précisément des taxonomies (ensemble de concepts structuré sous forme d'hiérarchie(la relation is-a). Il a été conçu pour l'intégration d'information. Le processus de mapping est un processus orienté qui essaye de relier les concepts de l'ontologie source aux concepts de l'ontologie cible. Les relations trouvées dans ce système sont : la relation d'équivalence (is-eq), la relation de subsomption (is-a) et la relation d'approximation (is-close).
   Taxomap tente de trouver pour un concept de l'ontologie source, le concept le plus approprié dans l'ontologie cible.

Afin d'identifier les correspondances, Taxomap implémente des techniques qui exploitent les labels attribués aux concepts et la relation de subsomption entre ces derniers dans la hiérarchie. Premièrement, il applique un analyseur syntaxique pour classifier les mots des labels en deux types " full words " et " complementary words" et sa par rapport à leurs catégories et leurs position dans le label.
Par la suite, une mesure de similarité est appliquée qui compare les trigrammes des labels des concepts en donnant plus de poids pour les "full words". Deuxièmement, les résultats de l'application de cette méthode est utilisé par une technique appelé "LabelInclusion" pour trouver la relation "is-a" entre les concepts. Par exemple : X is-a Y si seulement si :

(a) Les concepts Y et X ont une valeur de similarité très élevée

(b) Un des labels de Y est inclut dans l'un des labels de X

(c) Tous les labels de Y inclus dans X sont classifié comme "full words".

Parmi les techniques utilisées par Taxomap pour tirer les mapping entre deux concepts C_s de l'ontologie source et C_t de l'ontologie cible sont :

– Relation d'équivalence : La relation d'équivalence est génèrée entre les concepts C_s et C_t si la valeur de similarité entre eux est supérieur à un seuil donné.

– Relation d'inclusion : La relation is-a entre C_s et C_t est déduite si l'un des labels de C_t est inclut dans C_s et que tous les labels qui sont inclus sont classés comme étant "full words". Inversement si l'un des labels de C_c est inclut dans l'un des labels de C_t alors une relation "plus générale" entre C_s et $C_t$est déduite.

– Relation is-close : La relation isclose est déduite entre C_s et C_t si C_t a une valeur de similarité très elevé (superieur a un seuil) et si un de ses labels partage au moins deux " full word " de C_s.

– Deux classes C_s et C_t sont susceptibles d'être alignées si elles partagent les mêmes propriétés.

Comme OAEI considère seulement la relation d'équivalence dans les alignements de références alors les mapping produits par TAxomap avec les autres relations (la subsomption et l'approximation) sont transformés en relation d'équivalence. Cette transformation a engendré des mappings incorrects. Afin d'éliminer ces mappings, Taxomap introduit un module appelé "raffinement" qui tente de supprimer ces mappings. Parmi les étapes appliquées pour le raffinement :

– s'il existe un x, s'il existe un y où x est équivalent à y et qu'il existe un z tel que il existe un mapping entre y et z et que z est différent de x alors supprimer ce

mapping.

– Si un mapping entre x et y existe alors supprimer tous les mapping où il existe un concept z appartienne à l'ontologie source dont il existe un mapping avec y.

Taxomap a été évalué par OAEI en 2010 avec les différents jeux de données proposés où il a obtenu une moyenne de précision et rappel de 0,86 et 0,29 respectivement. Il a obtenu une très bonne valeur de précision vue les différentes caractéristiques qu'il a pris en compte pour tirer les mapping(les labels, structure). La valeur de rappel est très mauvaise parce que la plupart des mappings produit avec les autres relations sont incorrects après leurs transformations en relation d'équivalence et que dans l'alignement de référence seulement la relation d'équivalence est considérée.

2.8 Comparaison de différents algorithmes

Différents algorithmes pour le mapping d'ontologies ont été développés, dont chacun traite le problème différemment et considèrent différents caractéristiques des entités pour tirer les mapping. Une comparaison entre ces algorithmes peut se faire selon différents critères, à savoir : le types des ontologies considérés, les caractéristiques prisent pour tirer les mapping (niveau élément ou niveau structurelle, instances), les mesures de similarités appliquées, le domaine pour lequel ces algorithmes ont été conçus ainsi que la cardinalité de l'alignement produits et en fin nous pouvons aussi les comparer en regardant le degré d'automatisation. le tableau 2.4présente les différents comparaison :

Critères / Algorithme	Entrée	Les caractéristiques	Cardinalité	Degré d'automatisation	Domaine d'application
QOMFDE	OWL/ RDF	Labels, propriétés, synonymes, hyperonymes, instances	(1, n)	Automatique	service et agent basant sur les ontologies
QOM	OWL/RDF	Label, propriétés, superclasses, sous-classes, URI	(1,1)	Semi automatique	service et agent basant sur les ontologies
GLUE	OWL (Taxonomies)	Instances	(1,1)	Semi automatique	Web sémantique
Falcon-AO	OWL	Labels, superclasses, sous-classes	(1, n)	Automatique	Web sémantique
MAFRA	OWL	Labels, propriétés, relations	(1, n)	automatique	Intégration d'information dans Web sémantique
OLA	OWL sous forme de graphes	Labels, instances, superclasses, sous-classes	(1, n)	Automatique	Web sémantique
MapPSO	OWL	Labels	(1,1)	Semi automatique	Web sémantique
TaxoMap	OWL (taxonomies)	Labels, superclasses et sous-classes	(1, n)	automatique	l'intégration d'information

FIGURE 2.4 – Comapaison entre les différents algorithmes

D'après le tableau de comparaison, nous pouvons constater que les algorithmes de mapping proposés se différent dans quelques critères et partagent d'autres. Par exemple, presque tous les algorithmes comparés admettent en entrée deux ontologies OWL et pour certains (GLUE et Taxomap) un type particulier des ontologies OWL qui sont des taxonomies. Les algorithmes QEFDE, QOM, Falcon-AO considèrent différentes caractéristiques des entités à aligner soit au niveau élément ou structure ainsi que les instances. Seulement GLUE et MapPso se base sur une seule caractéristique pour faire le mapping, à savoir, les instances pour GLUE et les labels pour MapPso. Si nous regardant le domaine d'application pour lequel ces algorithmes sont conçus, nous pouvons remarquer que ces algorithmes sont tous conçus pour le web sémantique et certains pour des taches plus précisent. A titre d'exemple, MAFRA et MapPso pour l'intégration d'information, QOMFDE et QOM pour l'interopérabilité entre agents et services basés sur les ontologies. Un autre critère sur lequel nous pouvons baser pour faire la comparaison est le degré d'automatisation dont

certains nécessitent une intervention humain pour accomplir la tâche de mapping comme QOMFDE, QOM, GLUE et enfin MapPso.

2.9 Conclusion

Le mapping d'ontologies est une tache difficile en raison des problèmes posés par l'hétérogénéité des ontologies disponibles dans le web, des ontologies qui exposent des hétérogénéités terminologiques, conceptuelles et syntaxiques etc. Le mapping joue un rôle dans l'interopérabilité et la communication entre les applications basées sur les ontologies. Différents algorithmes pour le mapping d'ontologies ont été proposés dans la littérature mais la majorité tente de produire des mapping de qualité et seulement QOM et QOMFE se concentrent sur la proposition d'un algorithme qui couvre à la fois la qualité des résultats et l'efficacité en temps d'exécution pour qu'ils soient applicables dans les environnements distribués. Les algorithmes proposés considèrent différentes caractéristiques et comparent toutes les entités trouvant dans l'ontologie. En conséquence, le nombre important des entités et de comparaisons à effectuer consomment énormément du temps ce qu'est n'est pas souhaitable dans les environnements distribués qui exigent un temps de réponse raisonnable. QOM et QOMFDE ont proposés des algorithmes réduisant le nombre de comparaison et le nombre d'entités à comparer pour gagner de l'efficacité.
Dans ce chapitre, nous avons présentés une variété d'algorithmes, chacun procède d'une manière afin d'accomplir la tâche de mapping.

Chapitre 3

Algorithme distribué pour le mapping d'ontologie dans un environnement distribué

3.1 Introduction

La quantité croissante et l'hétérogénéité des sources d'information (fichiers, bases de données relationnelles et objet, fichiers XML, fichiers multimédia...) dans le web posent un vrai problème dans la communication et d'interopérabilité entre les systèmes existants.

L'ontologie considérée comme étant une spécification explicite et formelle d'une conceptualisation partagée, représente les concepts et les relations entres ces derniers d'un domaine joue un rôle dans la description des contenues des sources de données et la précision de leurs sens et leurs sémantiques.

Le mapping d'ontologies est vue comme une solution prometteuse pour que ces systèmes puissent interagir et échanger des données de manière sémantique. Le mapping d'ontologies tente de trouver les entités qui se correspondent entre deux ontologies.

La croissance continue du nombre d'ontologies dans le web et la façon dont elles sont développées, ont ramené à la multiplication d'ontologies pour un même domaine. Ces ontologies dépendent de la manière dont les développeurs interprètent le domaine, en conséquence, elles peuvent avoir différents niveaux de granularité (niveau de détail), différentes représentations ce qui expose le problème de l'hétérogénéité terminologique (exemple différentes appellation pour un même concept) et syntaxique (langage utilisé) et même hétérogènes. Les hétérogénéités exposées par les ontologies rendent l'accomplissement de mapping complexe et difficile.

Différents domaines d'applications dépendent de l'efficacité et de la qualité des mappings produit par l'application d'un algorithme de mapping comme : l'intégration des schémas de données, la composition des services web et le partage d'information dans les réseaux Peer To Peer etc.

Différents algorithmes pour le mapping d'ontologies ont étés proposés dans la littérature [18], où chacun propose une manière particulière pour le faire mais le problème est que la plupart de ces travaux se focalisent sur la qualité des résultats et néglige d'autre paramètres liés à la nature des applications (applications distribuées) où ces algorithmes sont conçues (web sémantique). Ces applications s'exécutent sur des architectures distribuées et hétérogènes et sur la représentation de service. L'interaction entre ces éléments (différents agents) nécessite un temps de réponse minimum et un niveau élevé de qualité.

Différentes techniques pour le mapping ont été proposés allant des algorithmes basés sur des méthodes lexicales à ceux qui tirent profit de la hiérarchie des concepts et ceux qui exploitent les instances afin de tirer les correspondances.

La plupart de ces approches appliquent leurs méthodes sur toutes les entités des ontologies à aligner à savoir les concepts, les propriétés, les instances etc ; ce qui consomme beaucoup du temps dans l'exécution.

Dans cet axe, l'approche que nous proposons tente de satisfaire ces besoins et cela en réduisant le nombre d'entités d' ontologies à comparer (les entités les plus importantes) en suivant le principe des approches proposées en [26, 27] et d'appliquer des méthodes qui renvoie des résultats satisfaisants et en un temps réduit par l'exécution parallèle des différents agents.

3.2 Approche proposée

Pour découvrir les correspondances (relation d'équivalence) entre les entités de deux ontologies O et O', nous considérons les concepts (équivalent aux classes en OWL) des deux ontologies. Ces concepts présentent les objets de domaine et ils regroupent les entités qui ont les mêmes caractéristiques. Le choix de concept est motivé par le fait qu'il est considéré comme étant l'entité la plus importante par rapport à d'autres entités présentent dans l'ontologie car dans un domaine particulier, les entités qui nous donnent plus de sens à un domaine sont les concepts et que les autres entités (propriétés et instances) servent à la définition de concepts et que les similarités entre ces dernières (propriétés et instances) peuvent être déduite à partir des équivalences trouvées entres les concepts. A titre d'exemple, si nous citons les concepts suivant Etudiant, Enseignant, TD etc. Nous pouvons déduire qu'il s'agit du domaine de l'enseignement.

Nous considérerons dans notre approche multicouche et multi agents les ontologies écrites en OWL, qui est un standard recommandé par W3C et qui constitue une extension du langage RDF qui est lui même dérivé de DAML et OIL langages pour les ontologies.

La considération des ontologies écrites dans le même langage(OWL) nous a permis d'éviter les problèmes liés aux différences syntaxiques et de bien faire apparaitre les différences sémantiques entre les ontologies considérés. De plus, ce type d'ontologies permet d'intégrer les ontologies RDF car OWL est une extension de ce langage.

Le choix de OWL est motivé aussi par le fait que 82% [32] d'ontologies parmi les 3959 ontologies disponibles sur le web sont implémentées sous OWL ou RDF. Dans [32], une approche a été proposée pour l'alignement des ontologies larges car les autres approches sont applicables seulement sur des petites ontologies et perdent leur efficacité avec les ontologies à grande échelle.

Notre approche s'appuie sur une ressource externe Word net qui un dictionnaire de langue Anglaise dont l'unité de base est le concept[33]. Dans Word net, les concepts sont présentés sous forme d'un string (chaine de caractères) et leurs sens est défini par l'ensemble de ses synonymes (noms qui ont le même sens) qui sont présentés sous une structure appelé synset et les différentes relations sémantiques entre les synsets. Word net respecte la catégorie syntaxique : noms, verbes, adjectifs et adverbes.
Différentes relations sémantiques sont définies entre les synsets pour définir le sens des concepts. Parmi ces relations, nous citons :

- **Synonymie :** c'est la relation de base dans Word net car elle utilise un ensemble de synonymes (synsets) pour définir le sens des mots et que cette relation est une relation symétrique.
- **Antonymie :** c'est une relation sémantique symétrique, en particulier importante dans l'organisation de la signification des adjectifs et des adverbes.
- **Hyponymie (sous nom)et hyperonymie (super nom) :** c'est une relation transitive entre les synsets. Elle est utilisée dans le but d'organiser les significations des noms dans une structure hiérarchique.

L'utilisation de cette ressource permet d'enrichir chaque concept par leurs synonymes, de récupérer leurs superclasses et ainsi augmenter la probabilité de coïncidence entre les concepts dans le processus de mapping. Word net a un apport considérable pour le mapping de point du vue de son utilisation par un nombre important d'algorithmes de mapping tel que [19, 27, 28] .

L'utilisation des synonymes (synsets) des concepts permet de déduire l'équivalence entre les concepts dont les labels (nom attribué à un concept) ne sont pas syntaxiquement similaires et d'appliquer les deux méthodes choisies (Jaccard similarité et Upward cotopic

similarity) car sont des méthodes qui s'appliquent dans la même hiérarchie.

Une fois les concepts des deux ontologies sont extraits, notre approche multicouche et multi agents s'exécute en trois étapes pour établir le mapping à savoir :

Etape 1 (Normalisation) : c'est l'étape de normalisation linguistique où les méthodes de NLP (Naturel Language Processing) sont appliquées pour ramener chaque forme d'un terme à une forme normalisée afin de réduire l'hétérogénéité lexicale et de faciliter l'étape de découverte de similarités basée sur les Labels (les noms attribués aux concepts). Parmi les étapes de la normalisation, nous avons appliqué dans notre approche :

- **Suppression des numéros :** c'est le processus qui permet de détecter les numéros et de les supprimer par exemple si nous avons un concept "Book24545-18" après la normalisation il devient "book".
- **Elimination des mots vides :** l'élimination des mots vides tel que les propositions, les conjonctions car ils sont considérées comme des mots non significatifs pour le mapping par exemple (of, and, for...). Une fois les conjonctions, les propositions et les numéros sont supprimés, nous mettons tous les termes en minuscule.

Etape 2 (Mapping) : une fois la phase de normalisation est achevée, nous procédons à la création de la matrice des concepts Mat (n, m) où n et m sont le nombre de concepts des ontologies O et O' respectivement, ensuite vient la phase de diffusion de cette matrice aux agents en charge du matching. Il existe trois types d'agent à savoir :

Agent 1 : c'est l'agent qui est en charge de découvrir les similarités lexicales basées sur les termes attribués aux concepts. La méthode appliquée au niveau de cet agent est N-gram similarité. N-gram similarité [34] est mesure normalisée (renvoie des valeurs appartient à l'intervalle [0,1] et fait apparaitre clairement la similarité lexicale entre les termes. Dans le cas où les deux termes sont identiques ou l'un est une sous chaine de l'autre elle renvoie une valeur de 1.0 ce qui signifie que les deux termes sont équivalents. Généralement dans les langues comme la langue anglaise, l'ajout d'un mot à un autre réduit sa largeur par exemple *reviewed article* et *article* sont des mots qui sont similaires et l'application de cette méthode renvoie des très bon résultats.

N-gram est une mesure qui détermine si deux chaines sont équivalentes en calculant le nombre de séquences communes de longueur n entre elles.

Dans notre cas la valeur de n est fixée à 3 si nous concéderons que la longueur du mot le plus court est de langueur 3. Dans le cas où les mots sont de longueurs courtes, nous pouvons ajouter des caractères sans influencer sur les résultats cette méthode.

Agent 2 : cet agent détecte la similarité entre concepts en exploitant la relation d'hiérarchie is-A entre concepts dans Word net. Plusieurs mesures ont été proposées dans

la littérature et généralement la similarité est déduite à partir de la distance qui sépare les concepts en termes d'arcs. La méthode appliquée au niveau de cet agent est Upward cotopic similarity (UCS) qui une mesure normalisée.

Cette méthode s'appuié sur la similarité de Jaccard et les parents des concepts dans la même hiérarchie[35]. Pour l'adapter au mapping d'ontologies, nous avons utilisé la ressource Word net pour avoir les classes générales de concepts à comparer. La similarité avec cette méthode est donnée par le contenu informationnel et pas en terme d'arcs séparant les deux concepts. Avant d'appliquer cette méthode, pour chaque concept des deux ontologies O et O', nous extrairons à partir de Word net leurs parents. La similarité entre deux concepts c et c'est donnée par la relation 3.1 suivante :

$$\sigma(c, c') = \frac{|UC(c, H) \cap UC(c', H)|}{|UC(c, H) \cup UC(c', H)|} \tag{3.1}$$

Où UC(c, H) est l'ensemble des superclasses de concepts c dans H est Word net.

Le choix de cette méthode est motivée par le faite qu'elle considère et compare tous les parents des concepts jusqu'à la racine et que la relation d'hyperonymie est une relation sémantique transitive très utilisée pour définir le sens des concepts et n'est pas seulement les parents directs comme le font les autres méthodes. Donc, la similarité retournée par cette méthode est une similarité sémantique. Une étude est faite par [35] où il a comparé les méthodes basées sur la hiérarchie des concepts pour tirer la similarité. Il a montré que UCS donne des très bons résultats par rapport à d'autre.

Agent 3 : la découverte de similarité au niveau de cet agent se base sur les synsets de Word ne. Chaque concept est renforcé avec le synset (ensemble de ses synonymes) qui lui correspond dans ce dictionnaire puis la méthode de Jaccard est appliquée pour calculer la similarité entre ces derniers.

La similarité de Jaccard est donnée par la formule 3.2 suivante où elle renvoie la similarité entre deux concepts A et B :

$$\sigma(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.2}$$

Nous obtenons avec cette mesure un 1.0 lorsque les deux concepts à comparer appartiennent au même synset et 0 autrement. Par exemple, la similarité entre " person " et " individual" ou entre " water " et "H2O" est 1.0.

Etape 3(Extraction des mappings pertinents) : Dans cette dernière phase, les résultats obtenus par l'application des différents agents sont utilisés pour établir les correspondances entre les entités de l'ontologie source et l'ontologie cible.

Pour extraire ces mapping, la technique que nous avons utilisée est le seuil.

D'abord, nous commençons à analyser les résultats obtenus par la méthode n-gram (agent 1) le fait que deux concepts sont similaires s'ils sont écrits de la même manière. Dans ce cas, le seuil sur lequel nous nous basons est "0.5" car dans le cas ou un concept est une sous chaine de l'autre la valeur minimal est 0.5. Dans le cas où la similarité entre deux concepts est superieur ou égale à 0.5, directement une correspondance est établie.

Dans le cas contraire où nous n'avons pas trouvé une correspondance entre les deux concepts au niveau des résultats de cet agent (agent1), nous passons à l'analyse des résultats produit par l'agent (3). Si la valeur renvoyé est égale à 1 une correspondance est établit directement entre les deux concepts sinon nous procédons à l'analyse des résultats produit par l'agent 2. Dans ce cas si la valeur dépasse un seuil de 0.5 nous allons établir la correspondance sinon ces deux concepts ne sont pas équivalents (aucune correspondance ni établit). Une fois la phase d'analyse est terminé, nous allons écrire les différentes correspondances trouvées.

3.3 L'architecture de notre approche

La figure 3.1suivante illustre l'architecture de l'approche que nous avons proposée. Elle consiste en deux ontologies en entrée pour lesquelles nous voulons calculer leur mapping et les différents agents chargés de calcul de similarité entre les concepts des deux ontologies et leurs interaction avec la ressource externe Word net ainsi que les phases d'extraction et d'écriture des correspondances trouver entre concepts.

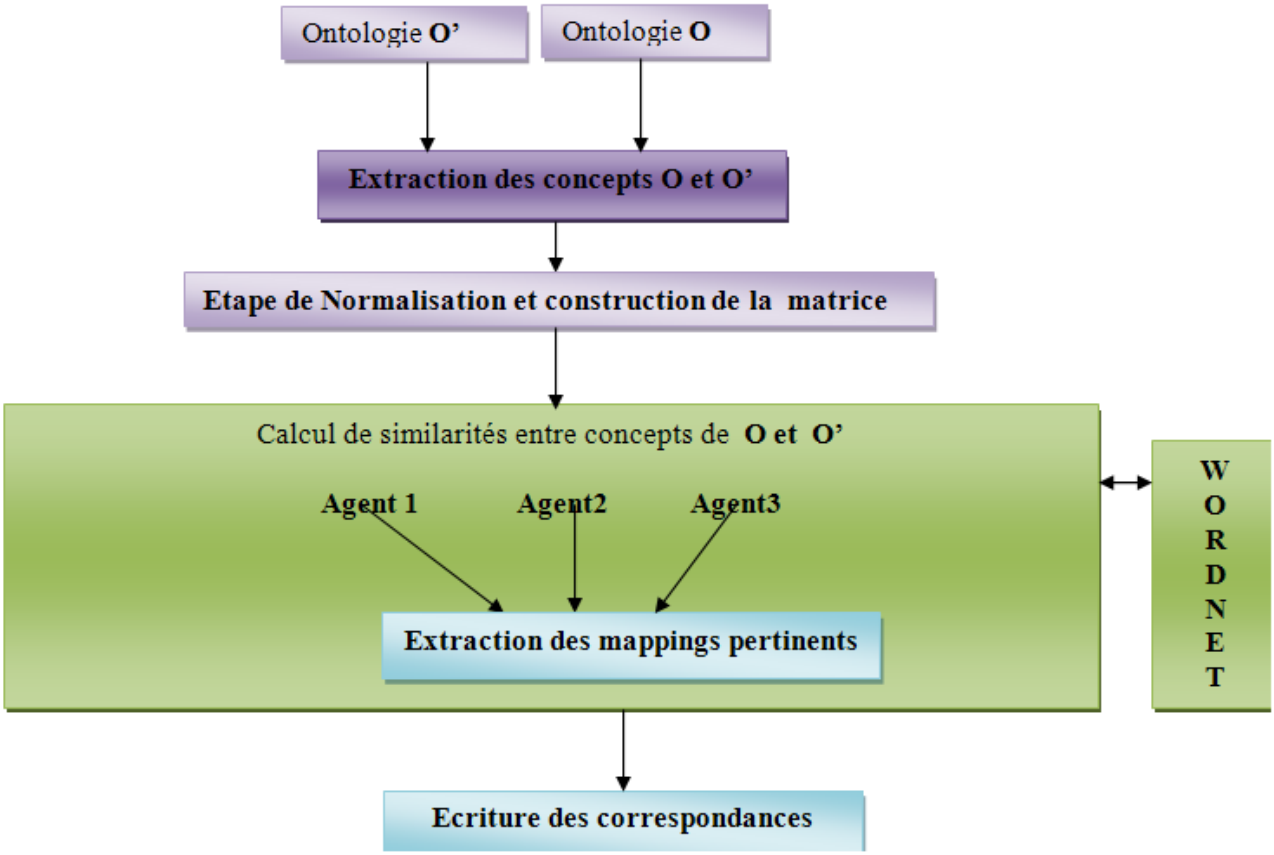

FIGURE 3.1 – L'architecture de notre approche

La première étape exécutée dans l'architecture proposée est l'extraction de concepts des deux ontologies en entrée O et O'. L'étape suivante est la normalisation. L'objective de cette étape est de mettre tous les concepts sous une représentation uniforme. Elle consiste à supprimer les mots vides, les numéros et de mettre tous les concepts en minuscule.
Une fois l'étape de normalisation est achevée, nous allons diffuser la matrice des concepts aux différents agents chargés de mapping. Nous avons trois types agent : l'agent1 calcule la similarité lexicale entre les différents concepts des deux ontologies avec la méthode N-gram similarité. L'agent2 est l'agent3 sont en interaction avec la ressource externe Word net pour enrichir les concepts avec des informations supplémentaires. L'agent2 applique la méthode "Upward cotopic similarity" qui est une méthode qui consédére tous les super concepts des concepts dans le dictionnaire Word net. L'agent2 enrichi chaque concept avec ses synonymes, puis applique la méthode Jaccard pour estimer la similarité entre les différents concepts. La dernière étape est l'extraction des mapping pertinents. La méthode adoptée est le seuil car toutes les méthodes appliquées aux niveaux des agents sont des méthodes normalisées qui renvoient des valeurs appartenant à l'intervalle [0,1].

3.4 Algorithme Proposé :

Notre algorithme admet comme données en entrée deux ontologies OWL O et O' et leurs URLs et le dictionnaire Word net.
Avec l'URL de chaque ontologie, nous créons un modèle avec l'Api Jena et à travers ce modèle, nous pouvons extraire les concepts de chacune des ces ontologies. Une fois que tous les concepts des ontologies sont extrait, nous allons les sauvegarder dans des vecteurs (vect1 et vect2) vect1 contient les concepts de l'ontologie O et vect2 contient ceux de l'ontologie O'.

Puis pour chaque élément des deux vecteurs, nous allons appliquer une fonction de normalisation " Normaliser () " qui prend comme paramètre un string qui est le terme attribué au concept. Une fois la normalisation est terminée, nous allons lancer en parallèle les tois agents (agent1, agent2, agent3) qui se chargent de mapping qui admet les deux vecteurs vect1, vect2 comme données en entrée. A chaque fois un agent termine, nous allons récupérer ses résultats. Une fois tous les résultats des différents agents sont récupérés, nous les examinons pour établir les correspondances entre les différents concepts.
Tout d'abord, nous commençons par l'analyse des résultats obtenus par l'agent1, si une correspondance est déduite, nous allons passé à un autre couple de concepts sinon, nous vérifiant pour l'agent2 est ce que nous pouvons établir une correspondance. Dans le cas échéant, nous passons aux résultats de l'agent3. Nous allons procéder de cette façon pour tous les couples de concepts. Voici l'algorithme proposé :

Les entrées :
O, O' : deux ontologies
URL1= chemin vers O
URL2=chemin vers O'
Word net : Dictionnaire ;
i=0 ;
Tant que (Trouver (classe, O)=vrai) faire
Vect1 [i]=classe de O ;
i++ ;
fin Tant que
i=0 ;
Tant que (Trouver (classe, O')=vrai) faire
Vect2 [i]=classe de O ;
i++ ;
fin Tant que

Pour i=0 allant de 1 à taille de vect1 faire
Normaliser (Vect1 [i]) ;
fin Pour
Pour i=0 allant de 1 à taille de vect2 faire
Normaliser (Vect2 [i]) ;
fin Pour
Créer la matrice des concepts Mat (n, m)
Diffusion de la matrice Mat(n,m) aux différents agents.
Agent1 :
Pour i allant de 1 à taille de vect1 faire
Pour j allant de 1 à taille de vect2 faire
Sim-Ngram[i][j]=Ngram(vect1[i],vect[j]) ;
fin Pour
fin Pour
Agent2 :
Pour i allant de 1 à taille de vect1 faire
Pour j allant de 1 à taille de vect2 faire
Sim-Jaccard[i][j]=Jaccard((recuperer(synset(vect1[i]),Wordnet),recuperer(synset(vect2[i]), Word net)) ;
fin Pour
fin Pour
Agent3 :
Pour i allant de 1 à taille de vect1
Pour j allant de 1 à taille de vect2
$Sim_cotopic[i][j] = Jaccard((recuperer(super-classes(vect1[i]), Wordnet), recuperer(super-classes(vect2[j]), Wordnet))$;
fin Pour
fin Pour
Récupérations des résultats des différents agents
Pour i allant de 1 à taille de vect1
Pour j allant de 1 à taille de vect2
Si $(Sim - Ngram[i][j] >= 0.5)$ Alors
Ecriture la correspondance entre Vect1 [i] et Vect2[j]
Sinon Si (Sim-Jaccard[i][j]=1.0)
Ecriture la correspondance entre Vect1 [i] et Vect2 [j]
Sinon Si $(Sim - cotopic[i][j] >= 0.5)$
Ecriture la correspondance entre Vect1[i] et Vect2[j] ;

fin Pour
fin Pour

3.5 la complexité des algorithmes de mapping :

Les différents algorithmes de mapping d'ontologies qui ont été proposés suivent les étapes du processus de mapping. Donc le calcul de la complexité du temps d'exécution du tel système revient à calculer le coût de chacunes de ces étapes, à savoir, le coût de l'étape de choix des caractéristiques *feat* , le coût de l'étape Recherche et sélection *sele*, le coût de l'étape de calcul de similarité. Pour chaque paire d'entités sélectionnée *comp*, un nombre K de mesures de similarité va être appliquée sim_k et leurs agrégation *agg*.

Le nombre d'entités impliquées et la complexité des mesures de similarités affectent la performance du temps d'exécutons. En conséquence, l'interprétation des valeurs des mesures de similarités requiert une complexité *inter* et en fin le nombre d'itération que l'algorithme peut effectuer réclame une complexité de *iter*.

D'après [27], la complexité d'exécution du plus mauvais cas est définie pour toutes les approches comme suit :

$$complexite_{tempsd'execution} = (feat + sel + comp * (\sum_k sim_k * agg) + inter) * iter$$

Calcul de la complexité de notre algorithme : L'algorithme " Multicouche et multi agents " que nous avons proposé couvre les étapes du processus de mapping. Donc sa complexité est donnée par le calcul de coût de chacune de ces étapes. Notons par n le nombre des entités à comparer. Dans notre approche, nous comparons seulement les concepts (classes)des ontologies au lieu de comparer toutes les entités des ontologies. Cela revient à dire que la valeur de n est petite par rapport au n considéré dans d'autres approches où ils compareent toutes les entités (concepts, propriétés, instances).
Le coût de l'étape de choix des caractéristiques *feat* ne nécessite pas des transformations ce qui engendre feat=0. L'extraction des concepts à partir des deux ontologies requiert une complexité $O(1)$ le fait que l'extraction d'une entité ou un ensemble d'entités est indépendant de la taille de l'ontologie. Pour l'étape de calcul de similarité, puisque les différents agents chargés de mapping s'exécutent en parallèle, le coût de cette étape revient à calculer le temps consommer par l'un des agents. Par exemple, l'agent 1 requiert un coût de $O(n^2)$ car il compare chaque concept de l'ontologie source avec tous les concepts de l'ontologie cible. L'étape d'agrégation des résultats ne s'applique pas dans notre cas et une seule itération est suffisante pour tirer les mapping pour cela le cout de ces étapes est

égale à 0. Pour l'étape d'interprétation, son coût est de $n * (3 * O(1)) = O(1)$ car au pire des cas nous devons parcourir tous les résultats des différents agents (3 agents) pour un couple de concepts.
par conséquence :
$Complexit = 0 + O(1) + O(n^2) + O(1) = O(n^2)$
Donc, la complexité de l'algorithme proposé est quadratique c-à-d polynomial en $O(n^2)$ et pour n pas top grand, les algorithmes polynomiaux sont encore efficaces. En conséquence, la complexité $O(n^2)$ de notre algorithme est très proche de la meilleure complexité qu'est d'ordre de $O(n * log(n))$ obtenu par QOM[27]en raison que la valeur de n considéré dans notre algoithme présente seulement le nombre de concepts et dans QOM, la valeur de n présente toutes les entités(concepts, propriétés, instances et relatons). Et si nous regardant l'ordre de grandeur des différentes complexités, nous pouvons dire que les deux complexités $O(n^2)$ et $O(n * log(n))$ sont très proche.

3.6 Implémentation de l'algorithme "multicritères et multi agents"

Pour l'implémentation de notre algorithme, nous avons choisi le langage java et les apis qui nous permettent d'interroger les ontologies et le dictionnaire Word net.
Java : Jave est un langage de programmation développé par Sun Microsoft, et qui présente l'avantage d'être multi plateforme, open source et gratuit et extensible. Il est assai proche du C++ dans sa structure. Il est utilisé pour développer différents types d'applications (applications web, base de données,...).

L'API Jena : c'est une API écrite sous java pour le développement des applications web. Elle offre un environnement de programmation pour les langages RDF, RDFS et OWL et inclut des règles d'inférence et un langage de requêtes appelé RDQL. Elle est open source et supporte différents types d'ontologies comme DAML, RDF et OWL. Avec l'URI d'une ontologie OWL, Jena crée un modèle sur lequel nous avons la possibilité de manipuler l'ontologie soit par l'ajout de nouvelles classes, de propriétés et d'instances ou par l'extraction de ces entités. Jena aussi offre des mécanismes pour stocker et charger les ontologies à partir du web. En résumé, l'utilisation de cette API dans notre implémentation, nous a facilité l'extraction des entités des deux ontologies après la création de leurs modèles ainsi que sa simplicité.
Voici un extrait de code pour l'extraction des classes des ontologies en entrée : les premières instructions servent à créer les modèles des ontologies en mémoire et cela par l'utilisation de la classe "ModelFactory" et l'appel à la méthode "createOntologyModel()". Une

fois les modèles des ontologies à aligner sont crées, nous allons les chargées avec leurs URLs par l'appel à la méthode "read". Puis, nous allons extraire les classes en utilisant la classe "Ontclass" qui sert à la description des classes OWL. La méthode "getLocalName" permet de récupérer le terme attribué au concept.

```
    // ce programma permet d'extraire les classe(concept) des ontologies
OntModel model1 = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
OntModel model2 = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);

model1.read("file:/C:/directories/source.owl");
model2.read("file:/C:/directories/target.owl");
System.out.print("\n la liste des classe de onto Source");
String l =null ;
for(ExtendedIterator i=model1.listClasses(); i.hasNext();)
{
OntClass Class=(OntClass)i.next();
vect1[i]=Class.getLocalName();
i++;
}
System.out.print("\n la liste des classe de onto Source");
for(ExtendedIterator i=model2.listClasses(); i.hasNext();)
{
OntClass Class=(OntClass)i.next();
vect2[i]=Class.getLocalName();
i++;
}
```

FIGURE 3.2 – Code pour l'extraction des classes des ontologies

API JWI (Java Word net Interface) : C'est une API développée sous java, extensible. Elle nous offre la possibilité d'accéder aux données de dictionnaire, faire la recherche des synsets et de naviguer à travers ces derniers par les différentes relations définies dans ce dictionnaire. La portion du code suivant nous permet de récupérer le synset du concept "person".

Dans ce code, nous commençons d'abord par la création d'une instance du dictionnaire avec l'interface "IDictionary". Une fois l'instance du dictionnaire est crée, nous appelons la méthode "open" pour l'ouvrir. En suite, nous procédons à la recherche de synset de concept "person".

```
public void getSynset() throws IOException
{
 // construct the URL to the Wordnet dictionary directory
 String wnhome = System.getenv("WNHOME");
 String path = wnhome + File.separator + "dict";
 URL url = new URL("file", null, path);

 // construct the dictionary object and open it
 IDictionary dict = new Dictionary(url);
 dict.open();
 // look up first sense of the word "person"
 IIndexWord idxWord = dict.getIndexWord("person", POS.NOUN);
 IWordID wordID = idxWord.getWordIDs().get(0);
 IWord word = dict.getWord(wordID);
 System.out.println("Id = " + wordID);
 System.out.println("Lemma = " + word.getLemma());
 System.out.println("Gloss = " + word.getSynset().getGloss());
}
```

FIGURE 3.3 – Code pour l'extraction des synsets d'un concept

Le résultat d'exécution de ce code est : [person, mortal, individual, someone, soul, somebody]

3.7 Conclusion

L'approche multicouche et multi agents que nous avons présentée dans ce chapitre est une approche qui tente de satisfaire les besoins des environnements distribués en terme d'efficacité et de fiabilité. L'approche proposée se base sur une ressource externe Word net dans le but d'associer des informations supplémentaires aux différents concepts d'ontologies afin de préciser leurs sens et d'augmenter la possibilité de trouver des correspondances. En conséquence, obtenir une qualité satisfaisante des résultats. La considération des concepts d'ontologies et l'exécution parallèles des différents agents chargés de mapping nous a permet dans un premier temps de réduire le nombre de comparaison et ainsi d'optimiser le temps d'exécution. l'algorithme proposé obtient une complexité quadratique c.-à-d polynomial $O(n^2)$. Cette complexité est très proche de la meilleur complexité qui est $O(n * log(n))$. Avec la valeur petite de n (nombre de classes), l'algorithme proposé est considéré comme efficace, ce qui rend son applicabilité dans les environnements distribués. Le chapitre qui se suit est consacré à l'évaluation de notre approche en termes de qualité des résultats (précision et rappel).

Chapitre 4

Evaluation de l'algorithme "Multicouche et Multi agents"

4.1 Introduction

Avec le nombre important des algorithmes et de techniques de mapping qui ont été développés ces dernières années, né le besoin d'établissement d'un consensus pour l'évaluation et la comparaison de tels systèmes.
L'évaluation des systèmes de mapping est une tache difficile car il y'a pas une méthode bien définit pour le faire et le choix des jeux de données (ensemble d'ontologies) est très important car il doit refléter le but de mapping et en plus, il est difficile de trouver deux systèmes qui sont évalués avec le même ensemble de données(DataSet).
Une initiative pour l'évaluation de ces systèmes est OAEI(Ontology Alignment Evaluation Initiative)[36]. OAEI est désormais une initiative internationale coordonnée qui a été mise en place depuis 2004 pour organiser l'évaluation des algorithmes de mapping. Les objectifs visés par OAEI sont :

- Evaluer la force et la faiblesse de tels systèmes.
- Comparer les performances de différentes techniques.
- Mettre les développeurs en relation et accroitre la communication entre eux.
- Proposer des jeux de données stables qui reflètent des cas réels (des benchmarks) pour pouvoir comparer les différents algorithmes et de faire apparaitre leurs points forts et faibles.

4.2 Les jeux de données proposés

Une variété d'ontologies a été proposée par OAEI 2010 [36] dans le but d'évaluer les systèmes de mapping. Les benchmarks proposés par cet organisme sont des ontologies qui couvrent un certain nombre de domaines, à savoir :

- **Edition :** Le benchmark du domaine Edition contient un ensemble de tests contenant une ontologie de référence du domaine d'edition et un nombre d'ontologies alternatives du même domaine écrites en OWL avec l'alignement de référence pour chaque test. Le but de ce benchmark est de pouvoir comparer les différents systèmes et d'évaluer le système lui-même en termes d'efficacité et de qualité. l'ontologie de référence est composée de 33 classes, 24 propriétés, 44 instances de propriétes et de 56 instances.
- **Anatomie :** Ce benchmark couvre le domaine de la médecine, son but est de faire l'alignement entre " Adult Mouse Anatomy "(ontologie OWL décrivant l'anatomie des souris adultes) et une partie de thésaurus NCI qui décrit l'anatomie humaine. Il est considéré comme étant un cas réel pour la validation de tell systèmes vue la richesse des ontologies proposées (plus de 2744 classes) et l'utilisation des annotations spécifiques au domaine.
- **Répertoire :** L'objectif de cet ensemble est de faire le mapping entre les arberescence des repertoires de sites web comme Yahoo et Google. Cet ensemble contient deux types de tests : le premier contient 400 tests pour le mapping du simple arberescence et le second contient deux arberescence complexes contenant respectivement 2854 et 6555 nœuds chacune.

4.3 Evaluation des systèmes de mapping

L'évaluation qualitative et quantitative des systèmes de matching peut se faire en calculant différentes mesures de performances. Le calcul de ces mesures se base sur des ensembles produit par l'intersection entre deux alignements. Les relations qui peuvent existés entre deux alignements [37] sont illustrées par la figure 4.1 suivante où :

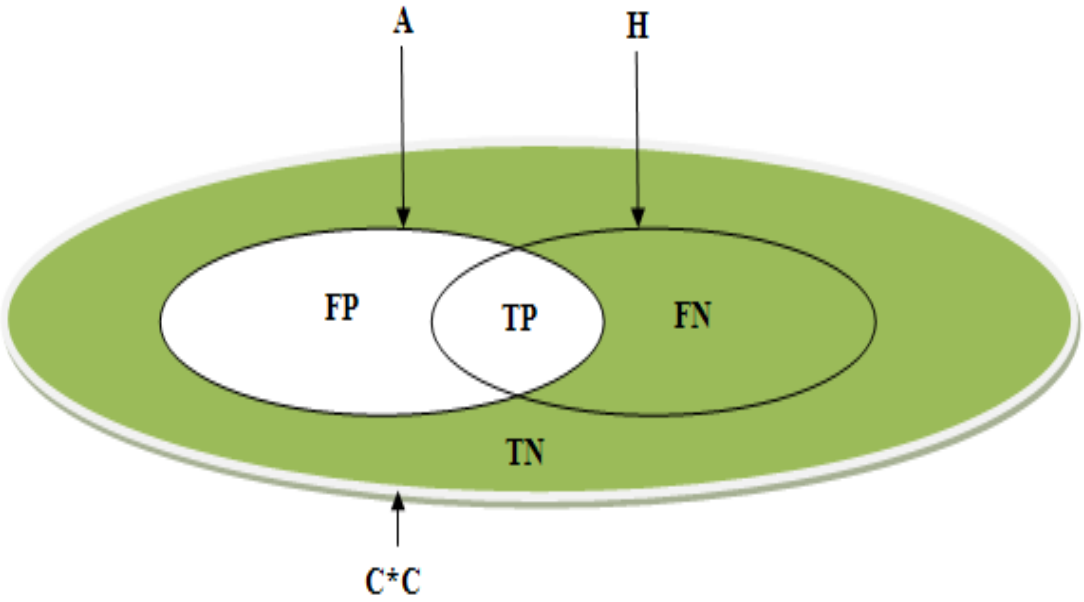

FIGURE 4.1 – Relation entre deux mapping

TP : ensemble des mappings trouvés correctes par le système par rapport à la l'alignement de référence $TP = H \cap A$
FP : ensemble des mapping incorrectes trouvés par le système $FP = A - A \cap H$
TN : ensemble des mappings incorrectes qui ne sont pas trouvés par le système $TN = M - A \cap H$
FN : ensemble des mappings correctes que le système n'a pas trouvés $FN = H - A \cap H$
$C * C'$: Le produit cartésien entre les entités de deux ontologies en entrée.

Parmi les mesures que nous trouvons pour l'évaluation de tels systèmes :

1. **La précision :** La précision est une mesure qualitative d'un système de mapping qui se calcul par rapport un alignement de référence (ensemble des correspondances considérées correctes généralement par un humain)[14].
   Soit H l'alignement de référence. La précision d'un alignement A produit par un système S est une fonction qui renvoie une valeur appartenant à l'intervalle [0,1].
   La précision est le rapport entre les correspondances correctes (True positives) sur la totalité des correspondances retournées par le système S (True positives et False positves). Sa formule est donnée en **??** comme suit :

$$Precision = \frac{|A \cap H|}{|A|} \tag{4.1}$$

2. **Le rappel :** Le Rappel est le rapport des correspondances correctes produites

par le système sur le nombre total de correspondances considérées correctes (Alignements de référence H)[37] .Il est consédéré comme étant un parametre de perfermance. une valeur elevé de Recall signifie que un nombre important des mappings ont été trouvés, mais aucune d'information sur les mappings incorects trouvés est donnée. le calcul de Rappel est donné par la formule 4.2 suivante :

$$Rappel = \frac{H \cap A}{H} \tag{4.2}$$

3. **F-mesure :** F-mesure est une mesure évaluant la qualité des systèmes de mapping. L'objectif de cette méthode est de remédier à l'insuffisance des deux mesures précédentes (précision et Recall) dans l'évaluation car l'une peut être déduite à partir de l'autre. F-measure est une mesure globale qui agrège la précision et recall. Elle est donnée par la formule 4.3 suivante :

$$F - measure = \frac{2 * Prcision * Recall}{Prcision + Recall} \tag{4.3}$$

   La difficulté rencontrée lors de calcul de la Précision et le Rappel est la connaissance précoce du H (alignement de référence) ainsi que la difficulté de sa construction cas où les ontologies considérées est larges (un nombre important de mappings à évalués).

4. **Les mesures de performances :** Les mesures de performances sont des mesures évaluant les systèmes en fonction des ressources consommées. Ces dernières dépondent de l'environnement d'exécution. Parmi ces mesures, nous avons la vitesse d'exécution qui est donnée par la quantité du temps pris par l'algorithme pour accomplir la tache de mapping.
   Pour avoir la possibilité de comparer différents systèmes par cette mesure, ils doivent être exécutés dans les mêmes conditions (vitésse du processeur, l'espace mémoire...). Une autre mesure de performance est la scalabilité qui est donnée par la possibilité du système à faire le mapping de différents types d'ontologies (légères ou larges). Une dernière est le degré d'automatisation, est ce que le processus nécessite une intervention humaine ou non.

4.4 Résultats d'execution de notre algorithme

Pour l'évaluation de notre algorithme, nous avons utilisé le benchmark de domaine "Edition" qui est un ensemble d'ontologies du domaine de bibliographie. Nous avons utilisé ce benchmark vue la disponibilité de l'alignement de référence qui nous permettra de calculer la précision et le rappel des résultats obtenus ainsi la simulation du comportement de notre système avec le changement que

les ontologies alternatives de ce domaine présentent par rapport à l'ontologie de référence. l'ontologie de référence est noter par (101) et les autres ontologies admettent comme noms des numéros de (102) jusqu'a (304)
Dans ce qui suit, nous allons présenter les résultats d'exécutions de l'algorithme proposé avec ce benchmark. Dans chaque test, le mapping est calculé entre l'ontologie de référence (101) et une ontologie alternative de même domaine. Nous avons exécuté l'algorithme sur 9 tests. Pour chaque test, l'ontologie alternative a subi des modifications remarquables par rapport à l'ontologie de référence afin de simuler son comportement et les paramètres "la précision" et "le rappel" et "F-mesure" de l'alignement produit sont calculés ainsi que le temps consommé pour accomplir cette tâche.

– **Mapping entre l'ontologie de référence (101) et l'ontologie (103) :** Dans ce test, l'ontologie de référence est comparée à sa généralisation en OWL lite. La généralisation supprime toutes les classes définies par les constructeurs " owl :unionOf ", " owl :oneOf " et les types des propriétés " owl :TransitiveProperty ".

le mapping produits par notre approche est le suivant

1 O1.book est equivalent à O2.book
2 O1.academic est equivalent à O2.academic
3 O1.techreport est equivalent à O2.report
4 O1.report est equivalent à O2.report
5 O1.report est equivalent à O2.techreport
6 O1.organization est equivalent à O2.organization
7 O1.motionpicture est equivalent à O2.motionpicture
8 O1.deliverable est equivalent à O2.deliverable
9 O1.lecturenotes est equivalent à O2.lecturenotes
10 O1.booklet est equivalent à O2.booklet
11 O1.chapter est equivalent à O2.chapter
12 O1.journal est equivalent à O2.journal
13 O1.date est equivalent à O2.date
14 O1.informal est equivalent à O2.informal
15 O1.mastersthesis est equivalent à O2.mastersthesis
16 O1.list est equivalent à O2.list
17 O1.part est equivalent à O2.part
18 O1.conference est equivalent à O2.conference
19 O1.onference est equivalent à O2.reference

20 O1.school est equivalent à O2.school
21 O1.monograph est equivalent à O2.monograph
22 O1.collection est equivalent à O2.collection
23 O1.collection est equivalent à O2.incollection
24 O1.manual est equivalent à O2.manual
25 O1.incollection est equivalent à O2.collection
26 O1.incollection est equivalent à O2.incollection
27 O1.institution est equivalent à O2.institution
28 O1.article est equivalent à O2.article
29 O1.publisher est equivalent à O2.publisher
30 O1.publisher est equivalent à O2.unpublished
31 O1.personlist est equivalent à O2.person
32 O1.address est equivalent à O2.address
33 O1.inproceedings est equivalent à O2.proceedings
34 O1.inproceedings est equivalent à O2.inproceedings
35 O1.unpublished est equivalent à O2.publisher
36 O1.unpublished est equivalent à O2.unpublished
37 O1.reference est equivalent à O2.conference
38 O1.reference est equivalent à O2.reference
39 O1.proceedings est equivalent à O2.proceedings
40 O1.proceedings est equivalent à O2.inproceedings
41 O1.person est equivalent à O2.person
42 O1.person est equivalent à O2.personlist

Valeurs des différents paramètres obtenus :
Précision= 0,81
Rappel= 0,85
F-mesure =0,82
temps d'éxecution = 18 secondes
A partir des valeurs des paramètres obtenues, nous pouvons constater que tous les paramètres sont satisfaisonts, c-à-d que la majorité des mapping trouvés sont correctes où 81% par rapport à l'alignement produit et 85% par rapport à l'alignement de référence. Le temps consommé pour accomplir le mapping est très raisonnable.

– **Mapping entre l'ontologie de référence (101) et l'ontologie (201) :** Dans ce test, différentes conventions de nommage dans l'ontologie 201 sont appliqués (majuscule, les tirets, souligner etc).

les valeurs des différents paramètres obtenues sont :
Précision= 0,77
Rappel= 0,62
F-mesure=0,68
temps d'éxecution = 20 secondes
Dans ce test, les valeurs des différents paramètres sont moyennes et satisfaisants et sa grâce à l'étape de normalisation où nous procédons à éliminer tous les mots considérés comme vides pour le mapping et mettre tous les concepts sous une représentation uniforme. Pour le temps d'exécution, il est toujours raisonnable.

- **Mapping entre l'ontologie de référence (101) et l'ontologie (205) :**
  Les labels de différents concepts de l'ontologie (101) sont remplacés par leurs synonymes dans l'ontologie (205) et les commentaires sont supprimés.
  Les valeurs des différents paramètres sont données comme suit :
  Précision=0,55
  Rappel =0,28
  F-mesure=0,35
  temps d'éxecution = 20 secondes
  Les résultats obtenus sont pas vraiment satisfaisants où la valeur de la précision est moyenne 55% mais le rappel est mauvais. Nous pouvons justifier ces résultats par le fait que les synonymes utilisés dans l'ontologie 205 sont pas les mêmes avec ceux de ce dictionnaire. En conséquence, nous suggérons lors de la conception d'une ontologie de se basé sur les concepts de Word net afin que son apport dans le mapping soit considérable.

- **Mapping entre l'ontologie de référence (101) et l'ontologie (221) :**
  Toutes les sous classes des classes définies dans l'ontologie 101 sont supprimés dans l'ontologie 221.
  Les valeurs des différents paramètres sont :
  Précision =0,82
  Rappel=0,86
  F-mesure=0,84
  temps d'éxecution = 20 secondes
  Puisque nous n'avons pas considéré les sous classes des classes dans notre approche. Cette modification n'influence pas sur nos résultats. En effet, la précision, le rappel et F-mesure sont tous satisfaisants .

- **Mapping entre l'ontologie de référence (101) et l'ontologie (224) :** La différence que présente l'ontologie 224 par rapport à 101 est que toutes les instances sont supprimées.
  dans ce test, nous avons obtenus :
  Précision 0,82
  Recall=0,86
  F-mesure=0,84
  temps d'éxecution = 20 secondes
  L'approche proposée ne se base pas sur les instances. Pour cela, leurs suppression n'a pas d'effet sur les résultats. Les résultats obtenus par ce test sont trés satisfaisons.

- **Mapping entre l'ontologie de référence (101) et l'ontologie (301) :**
  C'est une ontologie réelle appelée BibTeX. Elle est très similaire aux ontologies précédentes où différentes conventions de nommage est utilisées.
  Précision =0,56
  Recall=0,74
  F-mesure=0,64
  temps d'éxecution = 14 secondes
  C'est une ontologie réelle appelée BibTeX. Elle est très similaire aux ontologies précédentes où différentes conventions de nommage sont utilisées. La valeur de la précision est moyenne vue le nombre de mappings incorrects trouvés. La valeur de rappel est satisfaisante 74% et le temps d'exécution est très raisonnable.
- **Mapping entre l'ontologie de référence (101) et l'ontologie (304) :**
  c'est un cas réel de l'ontologie de domaine de bibliographie. Sa particularité est que quelque classes de cette ontologie sont des sous classes de l'ontologie de référence. Précision = 0,75
  Recall=0,76
  F-mesure=0,75
  temps d'éxecution = 20 secondes
  les résultats de ce tests sont tous satisfaisants.
- **Mapping entre l'ontologie de référence (101) et l'ontologie (228) :** Les propriétés et les relations entre les objets dans l'ontologie 208 sont complètement supprimées.
  Précision = 0,69
  Recall= 1.0

F-mesure=0,73

Dans l'approche proposée, nous avons considéré seulement les concepts. Pour cela, la suppression des propriétés et des relations n'ont pas d'effet sur le mapping produit. La valeur de précision est moyenne car elle est influencée par le nombre de mappings incorrects trouvés. Par contre, pour le rappel est meilleur (1.0) car les mappings corrects trouvés par l'algorithme sont ceux qui se trouvent dans le mapping de référence.

– **Mapping entre l'ontologie de référence (101) et l'ontologie (232) :** pas d'hiérarchie entre classes et absence d'instances dans l'ontologie 232
les valeurs des différents paramètres sont :
Précision =0,84
Recall=0,88
F-mesure=0,86
Les superclasses que nous avons utilisées proviennent du dictionnaire Word net. Donc, l'absence de la hiérarchie ne pose pas de problème dans le mapping. Pour les instances, nous n'avons pas basé sur ces dernières pour mapping. Pour cela les valeurs des paramètres calculés sont satisfaisantes.

4.5 Discussion des résultats

L'algorithme que nous avons proposé obtient une qualité satisfaisante dans tous les différents tests en raison que les caractéristiques sur lesquelles nous avons basé provient d'une ressource externe qui est Word net. Pour cela les modifications apportées aux différentes ontologies n'affectent pas les résultats obtenus. Sauf que dans le test (101-205), nous avons obtenus des valeurs peu satisfaisantes (0,55 et 0,28) et sa peut se justifier par le fait que les synonymes des concepts utilisés dans l'ontologie 205 sont différents de ceux existants dans le dictionnaire Word net. Pour cette raison, nous proposons aux développeurs d'ontologies de se basé sur la terminologie utilisée dans Word net afin que son apport dans le processus de mapping soit considérable. Dans le test dont nous avons comparé l'ontologie 101 et l'ontologie 201 où les différentes conventions de nommages sont appliquées, l'étape de normalisation a joué un grand rôle pour obtenir des bons résultats. En résumant, l'algorithme proposé peut s'en passé des hétérogénéités que les ontologies de Benchmark présentent vue les résultats obtenus.

4.6 Comparaison avec d'autres approches

Pour pouvoir comparer notre algorithme avec d'autres algorithmes qui addressent le même problème "le mapping d'ontologies", nous avons utilisé un ensemble de tests issus de benchmark proposé en OAEI en 2010. Ce benchmark contient un nombre de test, dans chacun, nous avons une ontologie de référence de domaine d'Edition avec une ontologie alternative du même domaine plus leur alignement de référence. Nous avons testé notre algorithme sur 9 tests qui sont (test103, test 201, test205, test221, test 224, test228, test232, test 301, test303) qui présentent des changements remarquables et qui influence le processus de mapping.
Pour chaque test, nous avons calculé la précision et le rappel et sa par rapport à l'alignement de référence. Puis nous avons calculé la moyenne de chacune (précision et rappel) afin de les comparer aux moyennes obtenus par d'autres algorithmes. Les algorithmes que nous avons choisis pour la comparaison sont des algorithmes qui ont été déjà évalué par l'organisation OAEI en 2010 avec le même jeu de données et qu'ils présentent des manières différentes pour trouver l'alignement. Ces algorithmes sont : Falcon qui est un système automatique pour le mapping d'ontologies OWL dont le but visé est de rendre les applications du web sémantique interopérables. Le principe de son fonctionnement est qu'il exploite les caractéristiques d'ontologies (label des entités, structure, propriétés) et combines différentes mesures de similarité soit par rapport au données en entrées ou bien par rapport aux résultats déjà obtenus par l'une des mesures. Taxomap est un système considérant en entrée des ontologies OWL, plus précisément des taxonomies. Il découvre différentes relations entre les entités des deux ontologies (relation d'équivalence, de subsomption et d'approximation). Il exploite les labels attribues aux entités et leurs structure dans la hiérarchie. Il applique différentes mesures de similarité et différentes heuristiques pour chercher les correspondances et il applique un module de raffinement dans le but de supprimer les mapping incorrects. MapPso est un algorithme qui modélise le problème de mapping comme étant un problème d'optimisation à deux objectifs, il cherche à trouver des mapping raisonnables et à maximiser leurs nombre. Il considère deux taxonomies en entrée et applique un algorithme de "discrete particle swarm optimisation(PSOD) " qui consiste à un nombre N de particules qui sont initialisées avec un alignement initial. Puis à chaque itération, il applique la Distance de string pour calculer la similarité entre les entités et mis à jour les alignements locaux et l'alignement global à chaque fois qu'une particule présente un meilleur alignement par rapport à celui qui existe en tenant toujours compte que l'algorithme converge vers un alignement optimal. Nous avons aussi l'alignement de référence refAlign qui

est un alignement conçue par l'être humain et qui présente le meilleur précision et rappel (1.0) La figure 4.2 présente une comparaison entre ces différents algorithmes en terme de précision et de rappel, où l'axe des des ordonnées présente les valeurs des différentes précisions et rappels obtenues par les différents algorithmes,qui sont des valeurs appartenant à l'intervalle [0,1] et l'axe des abscisses qui présente les différents algorithmes à comparés. Les résultats des différents algorithmes ont été obtenus à partir de [38]. A partir de cette figure, nous pouvons dire que notre algorithme obtient le meilleur rappel par rapport aux algorithmes comparés est sa car nous avons obtenu 82% de correspondances correctes par rapport à l'alignement de référence vue les différentes caractéristiques considères dans notre approche (les labels des concepts, leurs superclasses dans Word net et leurs synonymes) et la manière dont nous avons interprété nous résultats afin de tirer les correspondances. Premièrement, nous avons donné plus d'importance à la méthode lexical N-gram car deux entités qui s'écrivent de la même manière sont équivalents. Si cette fonction, nous offre des valeurs satisfaisantes, nous établirons directement une correspondance sinon nous passons à la méthode Jaccard qui s'opère sur l'ensemble des synonyme de chacun des concepts pour remédier au problème des synonymes. Enfin, la dernière méthode UCS qui considère toutes les superclasses des concepts dans Word net.

Pour la précision, notre algorithme obtienne une valeur satisfaisante mais pas trop loin des précisions obtenus par les autres algorithmes, le fait que ces algorithmes considèrent plus de caractéristiques (labels, commentaires, super classes, sous classes, propriétés...) et appliquent une variétés de mesures de similarité.

La précision que nous avons obtenu est plus grande que celle obtenu par MapPso , le fait qu'il a appliqué une seul méthode sur les labels attribués aux entités qu'est "String distance".

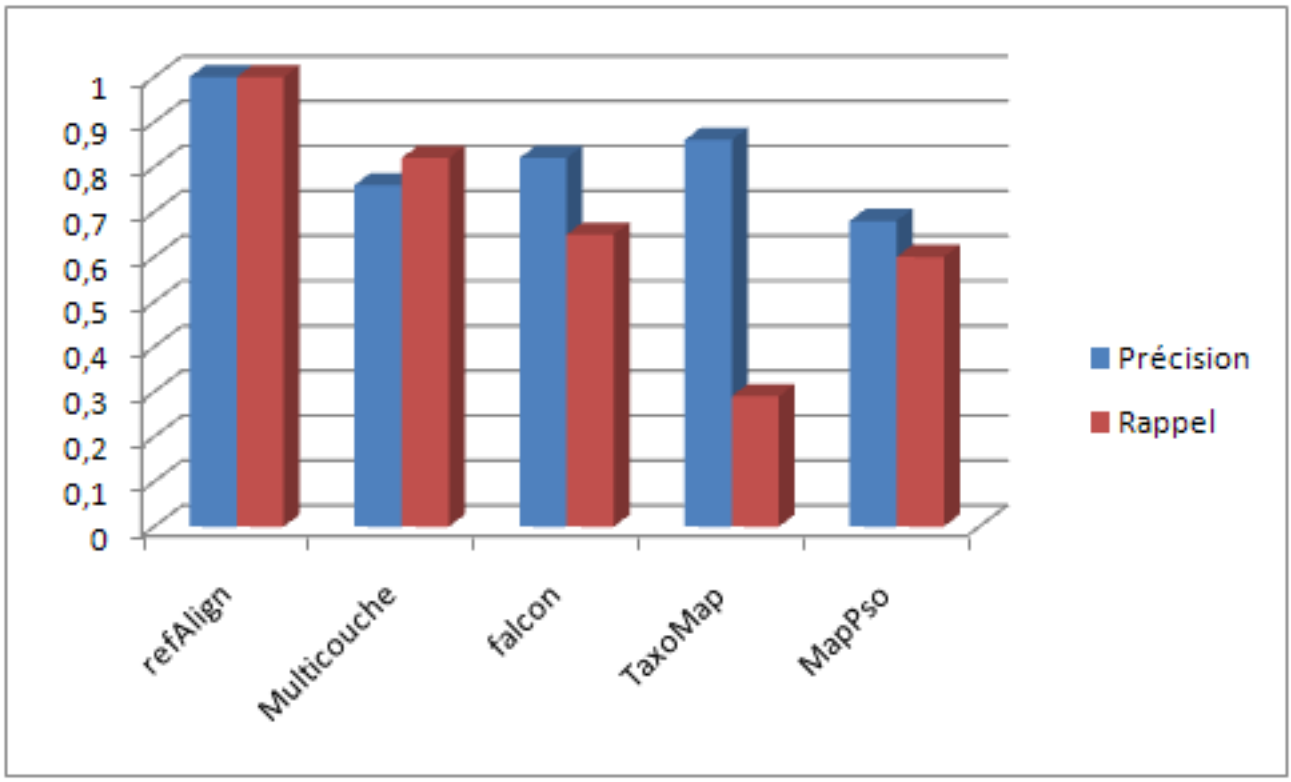

FIGURE 4.2 – Comparaison entre les différents algorithmes

4.7 Conclusion

Dans ce chapitre, nous avons évalué notre algorithme avec le benchmark proposé par OAEI en 2010 qui est composé d'un ensemble d'ontologies de domaine de bibliographie. Cette évaluation, nous a permet en premier lieu de calculer les paramètres qualitatifs (le rappel et la précision) et un autre quantitatif qu'est la précision afin de le comparer avec d'autres. Nous avons constaté que ces derniers sont les paramètres les plus déterminants pour l'évaluation de tels systèmes. L'algorithme " multicouches " proposé obtient une moyenne de précision et rappel de 0,76 et 0,82 respectivement qui présentent des valeurs très satisfaisantes. Notre algorithme présente une amélioration de rappel et une perte marginale de la précision par rapport aux algorithmes comparés et évalués avec les mêmes jeux de données. Le rappel est amélioré et cela est reflété par le nombre important de mappings corrects trouvés par rapport à l'alignement de référence et la considération des caractéristiques pertinentes qui aident le plus à trouver les mappings. La perte marginale de la précision peut se justifier par le fait que les autres algorithmes exploitent toutes les caractéristiques des entités trouvés dans l'ontologie. en effet, l'algorithme proposé présente une qualité et une efficacité très satisfaisantes ce

qui le rend applicable dans les environnements distribués.

Conclusion et Perspectives

Dans ce travail, nous nous sommes intéressés au problème de mapping d'ontologies dans les environnements distribués, qui est une conséquence de problème de l'interopérabilité entre les ontologies et la nature des applications où le mapping est une phase primordiale. Les environnements distribués s'exécutent sur des architectures et des données hétérogènes ce qui cause des diffécultés dans la communications et de coopérations.

Le mapping d'ontologies pose des vraies difficultés qui peuvent se résumer par :

- L'hétérogénéité des ontologies vues la manière dont elles sont développées, à savoir l'hétérogénéité conceptuel, terminologique, syntaxique et sémantique.
- Leur nombre et leur taille qui ne caisse de croitre vue l'augmentation continue des sources d'informations sur le web.
- Les environnements distribués qui s'exécutent sur des architectures et des données hétérogènes, leurs besoins de communication et de partage ainsi que leurs exigences en qualité des mapping produit et leurs efficacités en termes de temps d'exécution.

La plupart des algorithmes qui ont été proposés pour le mapping d'ontologies se concentrent sur la qualité des mapping produits et consomment un temps considérable à cause du nombre de comparaisons qu'ils effectuent et le nombre d'entités qu'ils considèrent. A savoir, il compares toutes les entités (concepts, propriétés, instances et relations).

L'approche que nous avons proposé tente de couvrir à la fois l'efficacité en terme de qualité et de temps d'exécution.

Pour cela, notre approche compare seulement les éléments significatifs des deux ontologies qui sont les concepts et il se base sur une ressource externe qui est le dictionnaire Word net afin d'enrichir les concepts avec des informations supplémentaires (les synonymes et les hyperonymes des concepts) dans le but d'augmenter la possibilité de trouver des mappings. L'exécution parallèle des différents

agents chargés de mapping qui couvre tous les aspects pour tirer les mappings, à savoir l'aspect lexical et structurel, ainsi que le nombre des entités (concepts) à comparées permis un gain du temps.

Dans notre approche, nous avons considéré en entrée des ontologies OWL, ce qui a permis d'inclure les ontologies RDF. Cette considération rend son applicabilité sur un nombre très important d'ontologies ainsi que d'éviter les problèmes de l'hétérogénéité syntaxique qui se produit lorsque nous comparons deux ontologies écrites dans deux langages ou modélisées par des formalismes de représentation de connaissances différents.

L'enrichissement des concepts avec les synonymes permet de résoudre le problème de l'hétérogénéité terminologique c.-à-d lorsque un même concept est noté par deux termes différents.

L'évaluation de notre approche avec le benchmark proposé par OAEI en 2010 qu'est composé d'un ensemble d'ontologies du domaine de bibliographie, nous a permis dans un premier temps de calculer les paramètres de performances et de dégager les points faibles et forts de notre approche. Par exemple, dans le cas où les ontologies posent des hétérogénéités terminologiques. L'évaluation a montrée que l'algorithme proposé est applicable dans les environnements distribués vue les résultats obtenus. A savoir, il a obtenu des moyennes satisfaisantes de précision et rappel de "0,76" et "0,82" respectivement et efficace en terme du temps d'exécution de $O(n^2)$.

Notre approche présente l'avantage d'être extensible en termes d'agents chargés de mapping où nous avons la possibilité d'ajouter d'autres agents qui vont exploiter d'autres caractéristiques comme les URIs, les instances etc des concepts dans le but d'améliorer plus la qualité des résultats.

Il applique des méthodes qui reflètent clairement la notion de similarité. Par exemple deux entités qui s'écrivent de la même manière sont équivalentes ou si deux concepts sont des synonymes alors ils sont équivalents. L'avantage que ces méthodes présentent est que sont toutes des méthodes normalisées ce qui nous a permis de faire des comparaisons entre elles. Un autre avantage que l'approche présente est son efficacité par l'exécution parallèle des différents agents et la réduction du nombre de comparaison à effectuer. L'utilisation du dictionnaire Word net, nous a permis d'augmenter la chance de trouver des similarités entre concepts en exploitants les synsets(ensemble des synonyme) et la relation d'hyperonymie entre les synsets.

Comme perspectives de notre travail, nous voudrons améliorer l'algorithme de manière pour qu'il soit applicable pour n'importe qu'elle type d'ontologies et

d'améliorer encore sa complexité en réduisant le nombre de comparaison et le nombre d'interprétations en préservant toujours sa qualité par l'application des algorithmes d'optimisation comme les algorithmes génétiques. Nous voudrons aussi réimplémenter notre approche avec l'API apiAlign qui est une API pour l'écriture des mapping sous format RDF afin qu'il puisse être publié, exploité et partagé sur le web et de l'évaluer par OAEI avec des ontologies de différents domaines et avec différentes caractéristiques pour bien simuler et apparaitre ses points faibles et forts ainsi que de l'appliquer sur un cas réel "environnement distribué" pour simuler son comportement.

Bibliographie

[1] R.Neches et al. Enabling technology for knowledge sharing, Article, 1991.

[2] Thomas R et al. A Translation Approach to Portable Ontology specifications Knowledge System Laboratory Stanford. Article.1993.

[3] Jean Charlet et al. Web sémantique.livre . 2003.

[4] W.N.Borst . construction of engineering ontologies. PHD thesis. centre for comunica and information.1997.

[5] Jean Charlet et al. Ontologies pour le Web sémantique.Rapport de recherche .2005

[6] M.Uschold et al. Towards a methodolgy for building ontologies. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada . 1995.

[7] Frédéric Fürst. L'ingénierie ontologique. Rapport de recherche. Octobre 2002.

[8] M. Blázquez et al . Building Ontologies at the Knowledge Level using the Ontology Design Environment.Laboratorio de Inteligencia Artificial Facultad de Informática, Universidad Politécnica de Madrid.

[9] Mohamed Chaabaniet al . formalisation de la logique de description ALL dans l'assistant de preuve Coq. journée francophones sur les ontologies.2009. page139.

[10] Jean Charlet et al. Ontologies pour le Web sémantique. Rapport de recherche. Institut National de l'Audiovisuel, Université Technologique de Compiègne.

[11] http ://www.w3schools.com/RDF/

[12] http ://www.w3.org/2004/OWL/

[13] Gaston Bachelard . Représentation des connaissances. livre. 1990

[14] Jérôme Euzenat · Pavel Shvaiko, Ontology Matching, Springer, livre, 2007

[15] Jie Tang, Juanzi Li. Using Bayesian Decision for Ontology Mapping. Department of Computer Science and Technology.Article.2006

[16] AnHai Doan et al. Learning to Map between Ontologies on the Semantic Web. Springer.2003

[17] Namyoun Choi et al. A Survey on Ontology Mapping . College of Information Science and Technology Drexel University. 2006.

[18] Yannis kalfoglou, Marco Schorlemmer. ontology Mapping : The State of the art. Advanced Knowledge Technologies.Article.2003

[19] von Dipl et al. Ontology Alignment :Bridging the Semantic Gap.Article. 2005

[20] François-Élie Calvier et al. Une aide à la découverte de mappings dans SomeRDFS.Article.2010

[21] Stephen L et al . Mapping Ontologies into Cyc. Article. 2002

[22] Nuno Silva, João Rocha. MAFRA - An Ontology MApping FRAmework for the Semantic Web. Proceedings of the 6th International Conference on Business Information Systems.2003

[23] Philip Resnik. Semantic Similarity in a Taxonomy : An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence Research .1999

[24] Jérôme Euzenat. Quelques pistes pour une distance entre ontologies .Article. 8èmes Journées Francophones pour Extraction et Gestion des Connaissances .2008.

[25] Wei Hu, Yuzhong Qu. Falcon-AO : A practical ontology system.Article. 2008

[26] Isaac Lera et .Quick Ontology Mapping Algorithm for distributed environments.Article.2008

[27] Marc Ehrig et al . QOM - Quick Ontology Mapping . Article.2004

[28] Jerôme Euzenat et al . OLA in the OAEI 2005 alignment contest . Integrating Ontologies Workshop Proceedings KCap Conference. Canada .2005.

[29] AnHai Doan et al . Ontology Matching : A Machine Learning Approach(GLUE).Article. 2004

[30] Jurgen Bock et al .Ontology Alignment using Discrete ParticleSwarm Optimisation. Article . 2010

[31] F.Hamdi et al . A Framework for Mapping Refinement Specification.Article.2010

[32] Wei Hu et al. Matching large ontologies : A divide-and-conquer approach.Data and Knowledge Engineering.2008

[33] George.A. Miller . WordNet : A Lexical Database for English. Communications of the ACM .November 1995

[34] Grzegorz Kondrak. N-gram similarity and distance . String Processing and Information Retrieval.Springer . 2005

[35] B. Bagheri Hariri et al. A new Structural Similarity Measure for OntologyAlignment. Article .2006

[36] http ://oaei.ontologymatching.org/

[37] Mikalai Yatskevich1 et al. A Large Scale Dataset for the Evaluation of Matching Systems. Article . 2009

[38] :Pavel Shvaiko, Jérôme Euzenat. Ontology Matching OM-2010. Proceedings of the ISWC Workshop.2010