République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole doctorale de l'informatique

De l'Est

(Université **20 Août 1955 Skikda)**

Mémoire

En vue de l'obtention du diplôme de

Magister en Informatique

Option : Intelligence Artificielle **(IA)**

Etude des techniques d'alignement des ontologies: proposition d'un algorithme de passage à l'échelle

Présenté par : ***Soumaya Kasri***

Devant le jury :

Président :	Mahmoud Boufaida ;	Professeur à l'université de Constantine.
Rapporteur :	Fouzia Benchikha ;	Maître de conférences à l'université de Skikda.
Examinateurs :	Zizette Boufaida ;	Professeur à l'université de Constantine.
	Smail Maazouzi ;	Maître de conférences à l'université de Skikda.

07/07/2010

Remerciements

Je tiens vivement à remercier toutes les personnes qui ont participé de près ou de loin à l'élaboration de ce travail, en particulier Dr. Benchikha Fouzia.

Je suis profondément reconnaissant à madame Benchikha pour le temps qu'elle a investi pour suivre et analyser les résultats de ce travail ainsi que pour les nombreuses discussions enrichissantes.

Je remercie M. Mahmoud Boufaida, professeur à l'université de Constantine pour avoir accepté de présider le jury.

Je remercie sincèrement Mme. Zizette Boufaida, professeur à l'université de Constantine et M. Smail Maazouzi, Maître de conférences à l'université de Skikda pour leur participation au jury.

Je salue également les étudiants de l'école doctorale en informatique de l'est. Particulièrement, un grand remerciement à Samia et sa famille et les autres qui ont partagé les événements mémorables avec moi pendant mon séjour à Annaba.

Un énorme remerciement à mes parents et tous mes frères et mes sœurs pour leurs encouragements.

Résumé

De nombreuses méthodes d'alignement dédiées aux ontologies ont vu le jour cette dernière décennie. Cependant, ces méthodes sont conçues pour aligner des ontologies de petite taille. Dans la littérature actuelle, il existe très peu d'algorithmes d'alignement qui visent à traiter le problème du passage à l'échelle des méthodes d'alignement. L'une des solutions du passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser l'alignement. En effet, pour diminuer l'espace de recherche des correspondances, il faut limiter la taille des ensembles de concepts en entrée de l'outil d'alignement.

Pour contribuer à résoudre ce problème, nous proposons un algorithme de partitionnement d'ontologie autour des ancres nommé PAA (Partitioning ontology Around Anchors). Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la préservation des alignements. Chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie. Le but du partitionnement est de diminuer le nombre d'entités dans un bloc, car un système d'alignement n'est plus du tout efficace à partir d'un certain nombre d'entités.

Pour la phase d'alignement, nous proposons une méthode structurelle qui prend la syntaxe abstraite du langage OWL comme un pattern général pour déduire la similarité entre deux structures internes de deux entités en utilisant la mesure de Tversky. Notre algorithme a été testé sur des ontologies de test disponibles pour le partitionnement notamment les paires Russia12 et TourimAB. Des résultats satisfaisants ont été obtenus.

Mots clés : ontologie, hétérogénéité, alignement, partitionnement, clustering.

Abstract

Many ontology alignment methods have been emerged over the last decade . However, these methods are designed for small-scale ontologies. In the current literature, there are few alignment algorithms that solve the problem of large–scale ontologies. One way of scaling up alignment techniques implies partitioning ontologies into blocks before performing the alignment. In fact, to reduce the search space of the correspondences, we must limit the size of sets of concepts as input alignment tool.

To contribute to solve this problem, we propose an algorithm called PAA (Partitioning ontology Around Anchors). Our algorithm has the advantage of having no isolated block in ensures the preservation of alignments. Each block of the first ontology is aligned with one block of the second ontology. The goal of partitioning is to reduce the number of entities in a block because an alignment system is not much effective from a certain number of entities.

For alignment phase, we propose a structural method that takes the OWL abstract syntax as a general pattern to deduce the similarity between two internal structures of two entities using the measure of Tversky . Our algorithm was tested on ontologies available on the web for partitioning such pairs Russia12 and TourimAB. Satisfactory results were obtained.
Keywords: ontology, heterogeneity, alignment, partitioning, clustering.

Table des matières

3 Méthode d'alignement proposée

4 Implémentation et évaluation expérimentale

Liste des tableaux

2 Etude des techniques d'alignement

Liste des figures

1 Le web sémantique et l'hétérogénéité

2 Etude des techniques d'alignement

3 Méthode d'alignement proposée

4 Implémentation et évaluation expérimentale

Introduction générale

De nos jours, les ontologies sont devenues l'une des plus importantes orientations de recherche avec l'avènement du Web Sémantique. Elles jouent un rôle primordial dans l'annotation de pages ou de services web puisqu'elles modélisent les concepts, attributs et relations utilisées pour annoter le contenu des ressources.

Dans de nombreux contextes applicatifs plusieurs ontologies couvrant un même domaine ou des domaines connexes sont développées indépendamment les unes des autres. L'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être résolue. C'est la problématique de l'interopérabilité. Cette dernière a pour objectif de permettre à des systèmes hétérogènes qui s'appuient sur une des ontologies de pouvoir communiquer et coopérer dans le but d'atteindre leurs objectifs. À cette fin, les liens sémantiques entre entités appartenant à deux ontologies différentes doivent être établis d'où l'alignement des ontologies.

L'alignement consiste à déterminer l'ensemble de correspondances entre deux ontologies en utilisant ou en mettant en œuvre des solutions aux différents problèmes d'hétérogénéité. L'ensemble de correspondance peut par la suite être utilisé à des fins multiples : fusionner des ontologies, créer de nouvelles ontologies à partir d'ontologies existantes, exprimer formellement des relations sémantiques entre ontologies qu'on appelle le mapping d'ontologies, réconcilier des ontologies par l'harmonisation de leurs contenus, migrer des données entre ontologies, négocier le sens dans un protocole par lequel des agents humains ou logiciel s'accordent sur les changement nécessaires pour réconcilier leurs ontologies, coordonner des ontologies pour réaliser une tâche donnée ou traduire des requêtes formulées en fonction d'une ontologie vers une autre…etc.

Problématique

Plusieurs ontologies couvrant un même domaine ou des domaines connexes sont développées indépendamment les unes des autres. Elles sont très souvent source d'hétérogénéité. Cette hétérogénéité posera des problèmes pour l'échange, le traitement, l'intégration, et la recherche d'information. Afin de rendre ces ontologies interopérables, les liens sémantiques entre les entités leur appartenant doivent être établis d'où l'alignement d'ontologies.

Avec l'apparition des nouveaux standards, outils et langages très expressifs, de grandes ontologies ont été développées dans plusieurs domaines. Cependant, les méthodes d'alignement proposées sont confrontées au défi du « passage à l'échelle ». Elles sont conçues pour aligner des ontologies de petite taille. Dans la littérature actuelle, il existe très peu d'algorithmes d'alignement qui visent à traiter le problème du passage à l'échelle des méthodes d'alignement. L'une des solutions du passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser leur alignement.

Dans ce cadre, les techniques d'alignement avec partitionnement doivent soulever le défi de n'avoir aucun bloc isolé en assurant la non perte des alignements.

Contributions

Considérant la problématique cité ci-dessus, nous proposons une méthode d'alignement d'ontologies structurelle. Pour dépasser les limites dues, notamment à des ontologies volumineuses, nous proposons un algorithme de partitionnement. De ce fait, notre contributions se déclinent en deux points :

1. **Proposition d'un algorithme de partitionnement**
   Nous proposons un algorithme de partitionnement des ontologies autour des ancres. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la préservation des alignements et fournit en sortie des paires de blocs à aligner. Chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie.
2. **Proposition d'une méthode d'alignement structurelle**
   Nous proposons une méthode structurelle qui prend la syntaxe abstraite du langage OWL comme un pattern général pour décrire les classes et les propriétés. La similarité entre deux structures internes de deux entités est déduite en utilisant la mesure de Tversky.

Une évaluation des algorithmes proposés a été effectuée pour montrer leur efficacité. Elle a été faite sur deux jeux de test. Le premier se compose de deux paires d'ontologies Russia12 et TourismAB et le deuxième se compose d'une ontologie large nommée Anatomy.

Plan du Mémoire

Ce mémoire est organisé en quatre chapitres. Dans le premier chapitre 1, nous présentons les connaissances de base du web sémantique et ses défis. Nous nous intéressons à

l'alignement d'ontologie en mettant l'accent sur sa définition, ses notions de base et la classification des méthode d'alignement.

Le deuxième chapitre a pour premier objectif d'aborder les technique d'alignement dans le domaine du web sémantique en présentant quelques critères sur lesquels une comparaison doit s'appuyer pour les classifier. Dans un second temps, on s'intéresse à l'étude de deux méthodes d'alignement, Falcon-AO et TaxoMap ,qui s'attaquent au problème du passage à l'échelle des techniques d'alignement. Ces deux méthodes qui visent à partitionner les ontologies avant de les aligner et présentent une source d'inspiration pour notre travail.

Dans le troisième chapitre, nous proposons un algorithme de partitionnement des ontologies, inspiré de Falco-AO. Il consiste à partitionner chaque ontologie en blocs autour des ancres en utilisant les algorithmes de clustering. Ces derniers fonctionnent itérativement en affectant à chaque étape une entité au cluster d'ancre le plus proche. L'alignement s'effectuera ensuite entre les blocs les plus similaires car les blocs qui contiennent les entités similaires, c.-à-d. les ancres, devraient l'être aussi. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la non perte des alignements et fournit en sortie des paires de blocs à aligner. Afin de montrer l'intérêt de l'aspect de design pattern d'ontologie, nous proposons une méthode d'alignement d'ontologie structurelle qui prend la syntaxe OWL comme un pattern général.

Une évaluation des algorithmes d'alignement d'ontologies proposés (partitionnement et alignement) présentée dans le chapitre quatre. L'implémentation des algorithmes a été effectuée en Java sur eclipse dans l'application nommée L-O-S-A (Large Ontology Structural Alignemnt), pour évaluer leurs performances. Les évaluations ont été faites sur des ontologies de test disponibles pour le partitionnement notamment les paires Russia12 et TourimAB.

En conclusion, nous présentons un bilan de notre travail et nos principales contributions, ainsi que quelques perspectives de ce travail.

Le web sémantique et l'hétérogénéité 1

Sommaire

Introduction

Le web contient une énorme quantité d'information et ceci reste difficile à exploiter efficacement. Par exemple, Il exige de consulter d'interminables listes de vols, d'hôtels et d'offres de location de voiture pour trouver notre bonheur. Mais, demain avec le web sémantique, il sera doté d'une forme d'intelligence globale et collective, capable de répondre en faisant une proposition de voyage complète. Le web sémantique est vu comme la solution à l'exploitation pertinente et automatique des informations disséminées sur la toile. Ces informations contenues dans des ressources telles que des pages web, des bases de données, des services… seront compréhensibles non seulement pour les hommes mais aussi pour les machines, les programmes ou les agents informatiques.

En ce qui concerne les informations, et plus particulièrement leur représentation et leur sémantique, l'ontologie est apparue comme un moyen de structuration formelle, contribuant à faciliter leur compréhension. Grace à elle la construction d'un web sémantique est vue comme un rêve devenu réalité.

Dans ce chapitre, nous présentons les connaissances de base du web sémantique et ses défis. Nous nous intéressons à l'alignement d'ontologie en mettant l'accent sur sa définition, ses notions de base et la classification des méthode d'alignement.

1.1 Le web sémantique

La vision du Web Sémantique initiée en 1998 par Sir Tim Berners-Lee [6] a été définie par : *« le web sémantique est une extension du web actuel dans laquelle on a donné un sens précis à l'information, de manière à améliorer le travail coopératif des hommes, et aussi des machines »*.

D'après la définition de Tim Berners-Lee, le Web sémantique permettra de rendre le contenu sémantique des ressources Web interprétables non seulement par l'homme mais aussi par la machine avec une meilleure exploitation des informations disponibles. A cette fin l'ontologie joue un rôle primordial pour décrire les concepts et leurs relations. Comme le web actuel, ce web sera aussi immense, ouvert, dynamique, hétérogène, et sémantique.

Le web sémantique ne sera pas une application mais une infrastructure sur laquelle seront développées de nombreuses applications selon deux approches :

a. **Les applications du web sémantique destinées aux organismes :** telles que les applications de B2B dans le e-commerce où la problématique essentielle de l'interconnexion d'applications entre entreprises consiste à garantir que les acteurs en présence interprètent bien de la même manière la signification des données échangées.
b. **Les applications du web sémantique destinés au grand public :** tels que les assistants intelligent pour collecter et filtrer les informations pertinentes et les composer dans une image cohérente selon les préférences de l'utilisateur.

Cette infrastructure permettra l'utilisation de connaissances formalisées au plus du contenu du web, elle devra aussi permettre de localiser, d'identifier et de transformer des ressources. Pour assurer la meilleure exploitation de ces dernières, des langages plus expressifs sont nécessaires. Le W3C propose une pyramide de langages graduellement standardisés (voir la FIG .1.1).

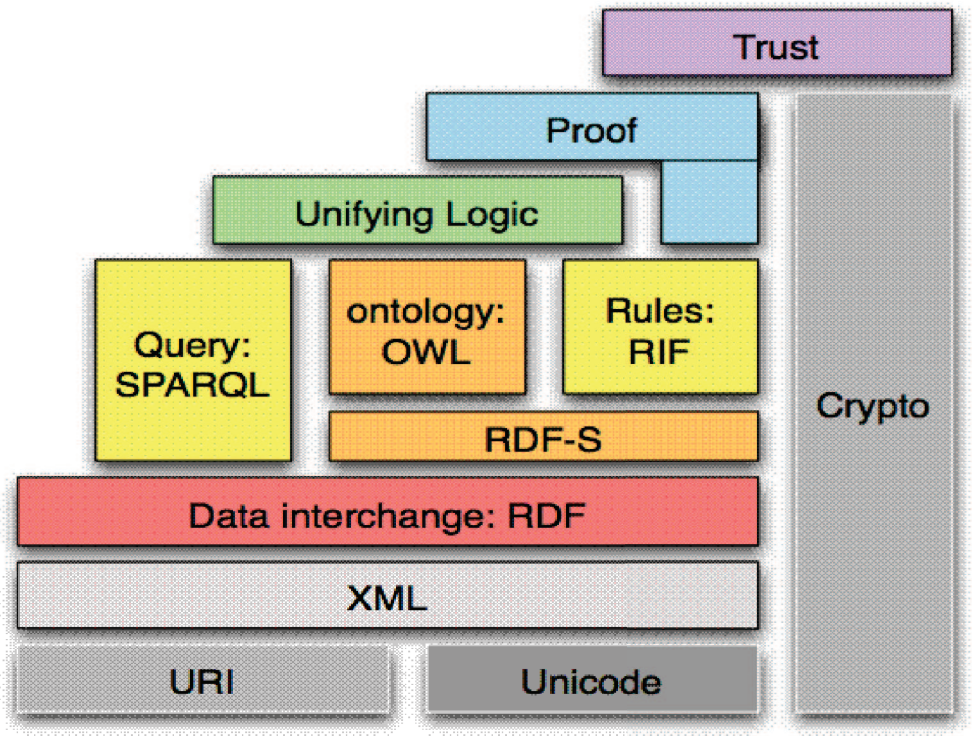

FIG.1.1 - *Pyramide du Web sémantique (Berners-Lee 2006)[7]*

Dans la base de la pyramide, on retrouve des technologies classiques. À savoir, pour l'essentiel, les URI (Universal Resource Identifier) permettant, d'identifier de façon unique des ressources. Ces ressources ne sont pas nécessairement disponibles sur le web. Le langage XML (eXtensible Markup Language) qui permet d'écrire des documents semi-structurés. La plupart des langages normalisés par le W3C dans le cadre du Web sémantique sont des dialectes XML.

La couche du milieu contient des technologies développées spécifiquement dans le cadre du web sémantique. Tout d'abord, le langage RDF (Resource Description Framework) est un format d'échange XML utilisé pour la représentation des connaissances sur les ressources du Web. C'est en quelque sorte le langage de base du Web sémantique. RDF permet de voir le Web comme un ensemble de ressources reliées par les liens étiquetés « sémantiquement ». Il permet de décrire des entités avec leurs propriétés, chacune associée à sa valeur, à partir d'un modèle simple de triplet (ressource, propriété, valeur). Au-dessus de RDF se trouve RDF Schema (RDFS) qui s'est ajouté rapidement pour offrir un niveau supérieur de structuration. La pyramide du W3C inclut aussi des couches supérieures où se trouvent des langages plus complets et plus complexes que RDFS. Parmi ces langages :

- OWL (Web Ontology Language) est une famille de langages de représentation des connaissances pour définir des ontologies. Basé sur les logiques de description, OWL enrichit les possibilités de description et de raisonnement offertes par RDF Schema.
- Le langage SPARQL (SPARQL Protocol and RDF Query Language) est un langage de requêtes pour RDF. Ce langage permet d'extraire des informations dans les documents RDF.
- Le langage RIF (Rule Interchange Format), en cours de normalisation par le W3C, pour faire des déductions plus complètes.

Les parties " Unifing Logic " et " Proof " forment la couche logique du Web sémantique, destinée au support des règles et à l'unification dans les ontologies. Les composants " Cryptography " et " Trust " (confiance) ont pour rôle la gestion de la sécurité des données sur le Web sémantique.

Dans la section suivante, nous allons présenter l'ontologie; la notion de base du web sémantique.

1.2 Les ontologies : vers le web sémantique

Le mot ontologie est à l'origine issu de la philosophie qui a pour objet l'étude des propriétés les plus générales de l'être, telles que l'existence, la possibilité, la durée, le devenir, ...etc. Il est emprunté du latin scientifique ontologia, lui-même composé à l'aide de onto-, tiré du grec ôn, ontos, « étant, ce qui est », et -logia, tiré du grec logos, « discours, traité » [15].

1.2.1 Définition d'une ontologie

La première définition de l'ontologie dans le domaine informatique a été présentée dans [44] comme : «une ontologie définit les termes et les relations de base comportant le vocabulaire d'un domaine aussi bien que les règles pour combiner des termes et les relations afin de définir des extensions du vocabulaire ». Dans la littérature, Il existe plusieurs définitions. Dans ce que suit on cite quelques unes.

Définition 1.1. «Une ontologie est une spécification explicite d'une conceptualisation ».

Dans cette définition d'ontologie de Gruber [21] « une spécification explicite » signifie que les concepts et les relations d'un modèle abstrait reçoivent des noms et des définitions explicites. Plusieurs critères sont à prendre en compte lors de la conception d'une ontologie:

1. **La clarté** : une ontologie doit fournir le sens attendu des termes définis, de manière indépendante du contexte. Lorsque c'est possible, la définition doit être complète et documentée en langage naturel.
2. **La cohérence** : l'ontologie ne doit autoriser que les inférences qui sont en accord avec les définitions afin de ne pas créer de contradictions.
3. **L'extensibilité** : une ontologie doit être conçue afin d'anticiper l'utilisation du vocabulaire partagé. C'est à dire qu'elle doit être capable de définir de nouveaux termes pour des usages spéciaux, basés sur le vocabulaire existant, et ne nécessitant pas la modification des définitions existantes.
4. **Une déformation d'encodage minimale** : une déformation d'encodage apparaît lorsque les choix d'une représentation sont faits uniquement pour une commodité d'implémentation. Ceci doit être évité car les agents partageant la connaissance peuvent être implémentés dans différents systèmes de représentation et styles de représentation.
5. **Un engagement ontologique minimal** : une ontologie doit faire aussi peu d'affirmations que possible sur ce qui a été modélisé afin de pouvoir instancier et spécialiser l'ontologie suivant les besoins.

Définition 1.2. « Une ontologie est une théorie logique qui donne une explication explicite et partielle d'une conceptualisation » [22]. Dans cette définition, une ontologie est exprimée sous un formalisme doté d'une sémantique formelle et partielle car une conceptualisation ne peut pas toujours être entièrement formalisée du fait d'ambiguïtés ou du fait qu'aucune représentation de sa sémantique n'existe dans le langage de représentation choisi.

Définition 1.3. « Une ontologie est une conceptualisation d'un domaine à laquelle sont associés un ou plusieurs vocabulaires de termes. Les concepts se structurent en un système et participent à la signification des termes. Une ontologie est définie pour un objectif donné et exprime un point de vue partagé par une communauté. Une ontologie s'exprime dans un langage (représentation) qui repose sur une théorie (sémantique) qui garantit des propriétés de l'ontologie en terme de consensus, cohérence, réutilisation et partage » [55].

1.2.2 Composants d'une ontologie

Les connaissances traduites par une ontologie sont à véhiculer à l'aide des éléments suivants [47]:

- **Les concepts :** les concepts doivent être compris dans un sens très large, ils peuvent être classifiés selon plusieurs dimensions :
  1. niveau d'abstraction : les concepts représentent les objets abstraits ou concrets du monde réel.
  2. Atomicité : les concepts représentent les objets élémentaires ou composites du monde réel.
  3. niveau de réalité : les concepts représentent les objets réels ou fictifs du monde réel.

  Un concept pourrait être aussi la description d'une tache, d'une fonction, d'une action, d'une stratégie, d'un processus de raisonnement…etc.
- **Les relations :** les relations traduisent les associations (pertinentes) existant entre les concepts du domaine. Ces relations incluent les associations suivantes:
  1. Sous-classe-de : les classes sont organisées par des relations taxonomiques selon un ordre de généralisation ou de spécialisation;
  2. Partie de : une classe peut définir une partie d'une autre classe par une relation d'agrégation ou de composition ;
  3. Instance-de : on peut relier une instance avec sa classe d'origine,…etc.
- **Les fonctions :** les fonctions sont un cas spécial de relations dans lesquelles le nième élément de la relation est unique pour les n-1 éléments précédents.
- **Les axiomes :** les axiomes constituent des assertions, acceptées comme vraies.
- **Les instances :** les instances représentent les éléments extensionnels spécifiques au domaine du problème modélisé.

1.2.3 Langage d'ontologie

Les langages d'ontologie sont des langages formels permettant de représenter une ontologie. Beaucoup de ces langages se bornent à permettre l'expression d'ensemble de concepts et leurs relations conceptuelles. Ils suffisent à construire des ontologies légères ayant des propriétés minimales nécessaires à la représentation et l'appréhension d'un domaine. Cependant, de plus en plus, il s'avère nécessaire de pouvoir ajouter aux ontologies légères des axiomes et des restrictions clarifiants le sens. pour construire des ontologies lourdes qui modélisent un domaine de façon plus profonde.

Il existe plusieurs langages informatiques spécialisés dans la création et la manipulation des ontologies. Parmi ces langages nous citons :

- **KIF :** KIF « Knowledge Interchange Format » est un langage destiné à faciliter des échanges de savoir. Il est basé sur les prédicats du premier ordre avec des extensions pour représenter des définitions et des méta-connaissances [43].
- **RDF, RDF(S) :** RDF « Resource Description Framework » est un formalisme graphique qui définit un modèle pour décrire la sémantique des données. Il est basé sur la notion de triplet (sujet, prédicat, objet) pour représenter les ressources (objets réels, concepts..) et pour que les informations présentes dans les graphes, sous la forme d'un réseau sémantique, soient utilisables par une machine. Cependant, RDF n'a pas de mécanismes pour définir des relations entre des propriétés et entre des ressources, c'est le rôle de RDF(S). Ce dernier définit des classes et des propriétés utilisables pour décrire des classes, des propriétés et d'autres ressources[51].
- **DAML+OIL :** DAML+OIL est un langage construit sur des normes précédentes du W3C telles que RDF et RDF Schéma, et étend ces langages avec des primitives de modélisation plus riches. DAML+OIL a été conçu à partir du langage d'ontologie DAML-ONT (DARPA Agent Modelling Language-Ontology, Octobre 2000) en vue de combiner plusieurs composants du langage OIL [14].
- **OWL :** Le langage OWL permet une interprétation du contenu Web par les machines supérieure, à celle offerte par les langages XML, RDF et le schéma RDF (RDF-S), en fournissant un vocabulaire supplémentaire avec une sémantique formelle. OWL se compose de trois sous-langages d'expressivité croissante : OWL Lite, OWL DL et OWL Full. Il est une évolution du langage d'ontologie Web DAML+OIL [46].

Dans ce qui suit, nous nous intéressons au problème d'hétérogénéité provenant de la création de multiples ontologies.

1.3 L'ontologie et l'hétérogénéité

La conceptualisation du monde et sa représentation nous amènent directement à la notion d'ontologie. Actuellement cette notion constitue une des voies les plus prometteuses quant à la modélisation et à la représentation formelle des systèmes informatiques. Cependant, beaucoup de projets opérationnels et de recherche ont abouti à la création de multiples ontologies, parfois pour un même domaine. Cet engouement sans précédent, en particulier en ingénierie des connaissances, a un risque inévitable de voir une hétérogénéité sémantique entre les ontologies car elles ne sont homogènes ni dans leur structure, ni dans leur sémantique. Dans ce qui suit, nous présentons brièvement deux recherches concernant l'analyse et la classification des disparités (mismatches) entre des ontologies.

1.3.1 Classification de Bouquet

Bouquet et al [9] présentent une classification des principales formes d'hétérogénéité selon quartes niveaux : syntaxique, terminologique, conceptuelle, sémiotique/pragmatique.

- **Hétérogénéité syntaxique** : Dans ce niveau, l'hétérogénéité est dépendante du choix du format de représentation. Celui-ci consiste à décrire et à coder les entités d'un domaine donné de manière à ce qu'une machine puisse les manipuler afin de raisonner ou de résoudre des problèmes. Dans la littérature, il existe de nombreux langages informatiques spécialisés dans la création et la manipulation des ontologies comme (XML, RDF, OWL..) et chacun d'entre eux est basé sur une syntaxe propre et parfois ils sont utilisés pour représenter la même ontologie.
- **Hétérogénéité terminologique :** Dans ce niveau, la disparité est liée au processus de nommage des entités (classes, propriétés, relations) d'une ontologie. Des exemples de disparité au niveau terminologique sont :

- Synonymie : des mots différents désignent la même entité.
- Polysémie : le même mot désigne plusieurs entités.
- Langage : des mots à partir des différents langages (Français, Anglais, Italien, …).
- Variation syntaxique : variation syntaxique du même mot (abréviations, préfixes, suffixes, …).

- **Hétérogénéité conceptuelle :** Chaque ontologiste, selon son domaine de prédilection va représenter un concept de manière propre et dans une hiérarchie spécifique. Benerecetti et al [3] présentent une analyse de cette disparité selon trois dimensions :

- Couverture : une ontologie ne couvre qu'un sous ensemble de connaissance d'un domaine donné.
- Granularité : des ontologies peuvent représenter les mêmes connaissances mais avec différents niveaux d'approximation, différents niveaux de granularité ou d'abstraction.
- Perspective : les représentations des connaissances, dans différentes perspectives, dépendent des éléments extérieurs tels que le lieu, le moment…etc. Deux ontologies peuvent être au même niveau de couverture et de granularité mais chacune représente le domaine à un moment donné.

- **Hétérogénéité sémiotique/ pragmatique :** Dans ce niveau, la disparité est liée au sens donné à l'ontologie où les individus et les communautés peuvent interpréter une même ontologie de différentes manières dans différents contextes.

1.3.2 Classification de Visser

Visser et al [65] présentent une analyse et une classification des disparités entre ontologies : disparité au niveau de la conceptualisation et au niveau de l'explication:

- **La disparité au niveau de la conceptualisation** : est la disparité entre deux (ou plus) conceptualisations d'un domaine au niveau des concepts, des relations…
- **La disparité au niveau de l'explication :** concerne les différences et les ressemblances dans la signification des concepts.

Dans les sections suivantes, nous allons présenter quelques défis du web sémantique. Puis nous allons intéresser à l'alignement d'ontologies.

1.4 Les défis du web sémantique

Il existe plusieurs défis à relever pour nous acheminer vers le web sémantique, parmi les travaux concernant les challenges qui vont conditionner la création du Web Sémantique on peut trouver ceux de Benjamins[4] et Chebotko[10] qui montrent ce que pourrait être un challenge lors de la construction du web sémantique en s'appuyant sur leurs expériences et leurs perspectives dans ce domaine.

Benjamins[4] montre qu'il y a plusieurs problèmes à résoudre avant de pouvoir véritablement passer au web sémantique et il a résumé ces problèmes en six points :

1. **Disponibilité du contenu; faire migrer le web vers un contenu sémantique :** La difficulté de la recherche sur le web actuel émane du fait que la plupart des documents disponibles sur le web sont en HTML, donc sous une forme peu structurée. Toutes les balises d'une page HTML concernent la présentation finale du document et rien ne permet à un logiciel de connaître le sens de son contenu.
   Un premier pas vers le Web sémantique consiste à pousser les développeurs et les entreprises vers l'adoption des principes du Web sémantique, en expérimentant des formats facilitant la migration à partir de l'existant (comme le XHTML), vers la prise en compte des métadonnées.
2. **Disponibilité, développement et évolution des ontologies :** Les ontologies sont centrales pour le Web sémantique. Elles constituent une solution pour la représentation explicite des contenus des ressources du Web. Plusieurs efforts se concentrent sur le développement d'une infrastructure pour faciliter la création d'ontologies, la gestion du changement d'ontologie, le mapping d'ontologie, la gestion de l'évolution de l'ontologie, les annotations relatives à cette évolution…etc.

3. **Passage à l'échelle :** Le passage à l'échelle du web est un véritable défi qui demande des efforts aux chercheurs pour optimiser la gestion du contenu qui peut faire l'objet d'un enrichissement et évolution permanente. A cette fin, il est nécessaire d'améliorer la qualité de l'organisation, la structuration, la recherche, l'identification, l'accès, l'utilisation, la réutilisation des ressources, l'intégration, et les traitements automatisés de ce contenu.
4. **Multilinguisme :** Dès ses débuts, la principale langue du web a été l'anglais mais il y a d'autres langues qui sont maintenant présentes. En juin 1997, les résultats de la première étude sur la répartition réelle des langues sur le web sont publiés dans le Palmarès des langues de la Toile. Les pourcentages étaient de 82,3% pour l'anglais, 4% pour l'allemand, 1,6% pour le japonais, 1,5% pour le français, 1,1% pour l'espagnol, 1,1% pour le suédois et 1% pour l'italien. D'après les statistiques de Vilaweb.com en 2001, les pourcentages étaient de 68.4% pour l'anglais, 5.9% pour le japonais, 5.8% pour l'allemand, 3,9% pour le chinois, 3.0% pour le français, 2.4% pour l'espagnol, et 1.6% pour l'italien.

   Dans l'avenir, tôt ou tard, la répartition des langues sur le web correspondera à la répartition des langues dans le monde, et que des logiciels de traduction seront disponibles gratuitement sur le Web pour traduire instantanément une page dont on ne comprend pas la langue. Comme le web actuel, Le multilinguisme constitue un défi pour le web sémantique. Ceci dit, il reste beaucoup à faire pour que ce rêve devienne réalité.
5. **Visualisation :** Afin de réduire le travail d'un utilisateur du web pour la visualisation des informations et l'interaction face à une telle surcharge informationnelle croissante, il est nécessaire d'avoir des outils pour permettre la visualisation de contenus en termes graphiques, en opposition à une liste de données sans signification.
6. **Stabilité des langages du web sémantique :** Le Web sémantique est en cours de réalisation. Il permettra l'utilisation de connaissances formalisées. Le web sémantique devra aussi permettre de localiser, d'identifier et de transformer des ressources. Pour cela, des langages expressifs sont nécessaires. Il est important de noter que toutes les technologies doivent être normalisées afin de permettre le passage à l'échelle du Web sémantique.

A son tour Chebotko et al.[10] consacrent leur discussion sur les challenges et mettent les quatre défis suivants :

1. **Les défis de développement d'une ontologie du domaine :** De multiples ontologies ont été développées partout dans le monde. Cela implique que plusieurs experts, pour un même domaine, ont développé indépendamment leur propre ontologie chacun selon son point de vue. Cela exige de créer un environnement de développement collaboratif (Multiple points de vue sur le domaine) d'ontologies offrant un ensemble de services où les plus pertinents sont :

- Communication Synchrone / Asynchrone.
- Traçabilité.
- Gestion d'accès concurrents.
- Contrôle de Versions CVS.
- Hiérarchisation des utilisateurs
- Edition en ligne.

2. **Le défi de mapping, d'alignement et de fusion :** Plusieurs ontologies sont développées indépendamment les unes des autres. Pour cela, Le besoin de comparer les termes des ontologies, de passer de l'une a l'autre ou d'échanger les ressources et les instances entre des bases des ressources indexées par des ontologies devient donc nécessaire. Cependant, ils peuvent se révéler complexe à mettre en œuvre en raison de l'hétérogénéité des ontologies à traiter.
3. **Le défi de la gestion d'annotation :** Une annotation est un commentaire, une note, une explication ou toute autre remarque externe qui peut être attachée à un document web ou à une partie de celui-ci. Le premier défi pour la gestion d'annotation consiste à l'intégration des ontologies dans les outils d'annotation. Pour chaque domaine, l'outil d'annotation se base sur une ontologie donnée en tenant compte des connaissances de l'utilisateur.
4. **Le défi de la recherche d'information guidée par ontologie :** Une annotation désigne à la fois une métadonnée associée à un document ou une partie du document et le processus de génération de cette métadonnée. Une annotation est sémantique lorsqu'elle se réfère à une ontologie, qui décrit un domaine donné.

   Dans le Web sémantique, les pages Web sont annotées à partir de connaissances disponibles dans une ou plusieurs ontologies. Ces annotations sémantiques ayant un impact positif sur la pertinence des résultats d'une recherche d'information. Avec les annotations, les moteurs de recherche pourront être capables de répondre à des requêtes complexes tenant compte des connaissances de l'utilisateur. Mais ces

nouvelles possibilités et capacités nécessitent de nouvelles approches de raffinement des requêtes, de mapping d'ontologies…etc.

1.5 L'alignement; un défi dans le web sémantique

L'ontologie constitue une des voies les plus prometteuses quant à la modélisation et à la représentation formelle dans le web sémantique. Mais la création de multiples ontologies, parfois pour un même domaine conduit à une hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles qui doit être résolue c'est la problématique d'interopérabilité. Pour cette raison, un certain nombre de techniques ont été proposées dont l'une d'elles est l'alignement d'ontologie.

1.5.1 Définition de l'alignement d'ontologie

Dans la littérature, plusieurs définitions de l'alignement peuvent se rencontrer selon la nature de la structure alignée (schémas de base de données, ontologies…).

Définition 1.4. « Une fonction qui prend en entrée deux schémas de base de données S1 et S2 et qui retourne un alignement entre l'ensemble des éléments de S1 vers celui de S2. Chaque élément de l'alignement exprime une relation entre un élément de S1 et un élément de S2 » [50].

Définition 1.5. « Une opération de matching qui prend en entrée *O et O' (schéma/ontologie) et* qui retourne un ensemble de mapping *A'* ou chaque élément de *A'* est défini comme un 5-uplet : <*id, e, e', n, R*> où :

- id : est l'identifiant du mapping
- e et e' : sont les entités mises en relation (éléments XML, classes, etc.)
- n : est une mesure de confiance (valeur entre 0 et 1)
- R : est une relation : équivalence (=), plus général (⊇), disjonction (⊥)…..(voir FIG 1 .2)

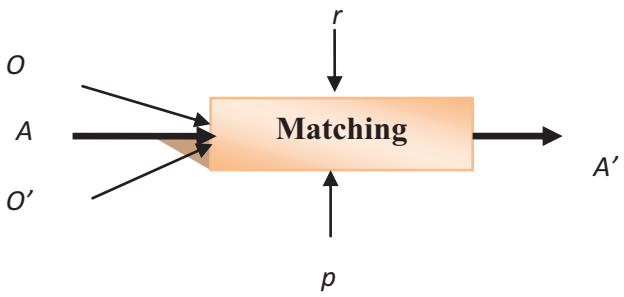

FIG 1.2 - *Processus de matching*

Cette opération peut prendre en compte des paramètres *p* (un alignement préliminaire *A*, un seuil, pondérations…) et des ressources externes *r* (thésaurus, dictionnaire….)[56].

1.5.2 Cas d'utilisation de l'alignement d'ontologie dans le cadre du web sémantique

Parmi les applications qui exigent l'alignement d'ontologie pour un bon fonctionnement on peut trouver :

- **La Communication entre agents :** L'alignement d'ontologies permet aux systèmes multi-agents de s'attaquer au problème d'hétérogénéité sémantique dans la communication où chaque agent a sa propre représentation du monde (voir FIG 1.3).

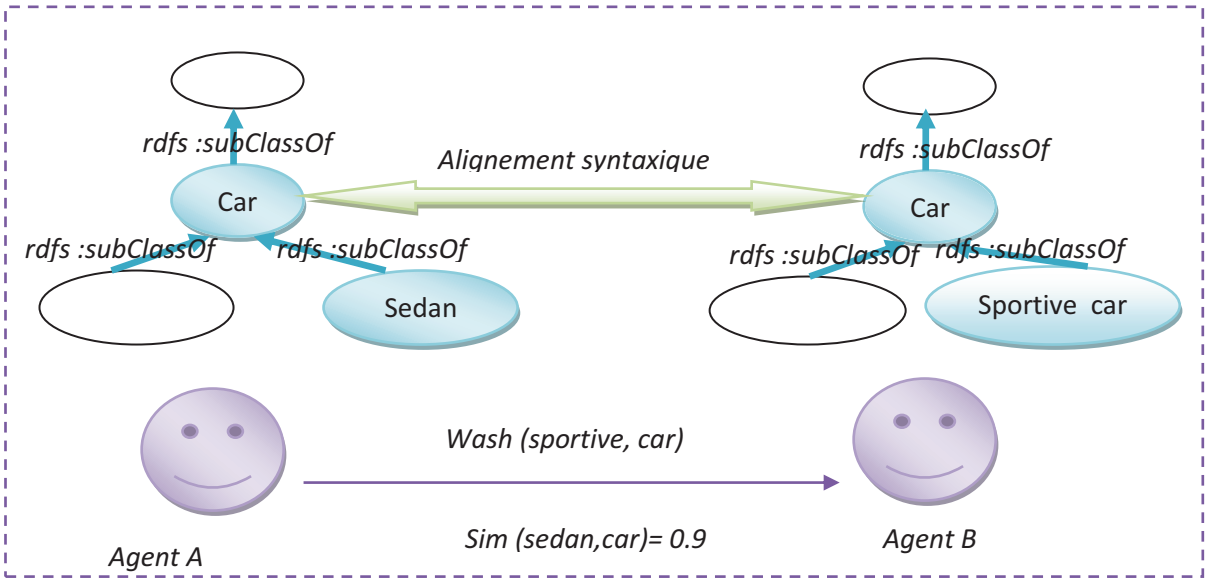

FIG.1.3 - *Communication entre deux agents ayant deux ontologies distinctes.*

- **L'intégration des web services :** De nos jours les entreprises ont adopté les Web Services [1] comme instrument nécessaire à l'intégration de leur métier dans l'e-commerce. Dans le cadre du web sémantique, les technologies et les outils développés dans ce contexte peuvent certainement compléter la technologie des web services en vue d'apporter des solutions au problème de l'automatisation du processus de découverte des services publiés sur le Web, et de les combiner pour répondre à une requête donnée . Dans ce domaine, les ontologies peuvent jouer un rôle primordial pour décrire explicitement la sémantique des services. Cependant, ces ontologies sont souvent différentes et hétérogènes. Une telle hétérogénéité entraîne souvent des ambiguïtés sémantiques. Cela a nécessité d'établir des mises en correspondance entre

[1] Un web service est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.

ontologies. La figure 1.4 présente un exemple d'utilisation des techniques d'alignement dans le domaine du web services. Dans cet exemple, la description des services web et les requêtes des clients sont écrites en logique de description. L'alignement consiste à trouver le service adéquat à une requête donnée.

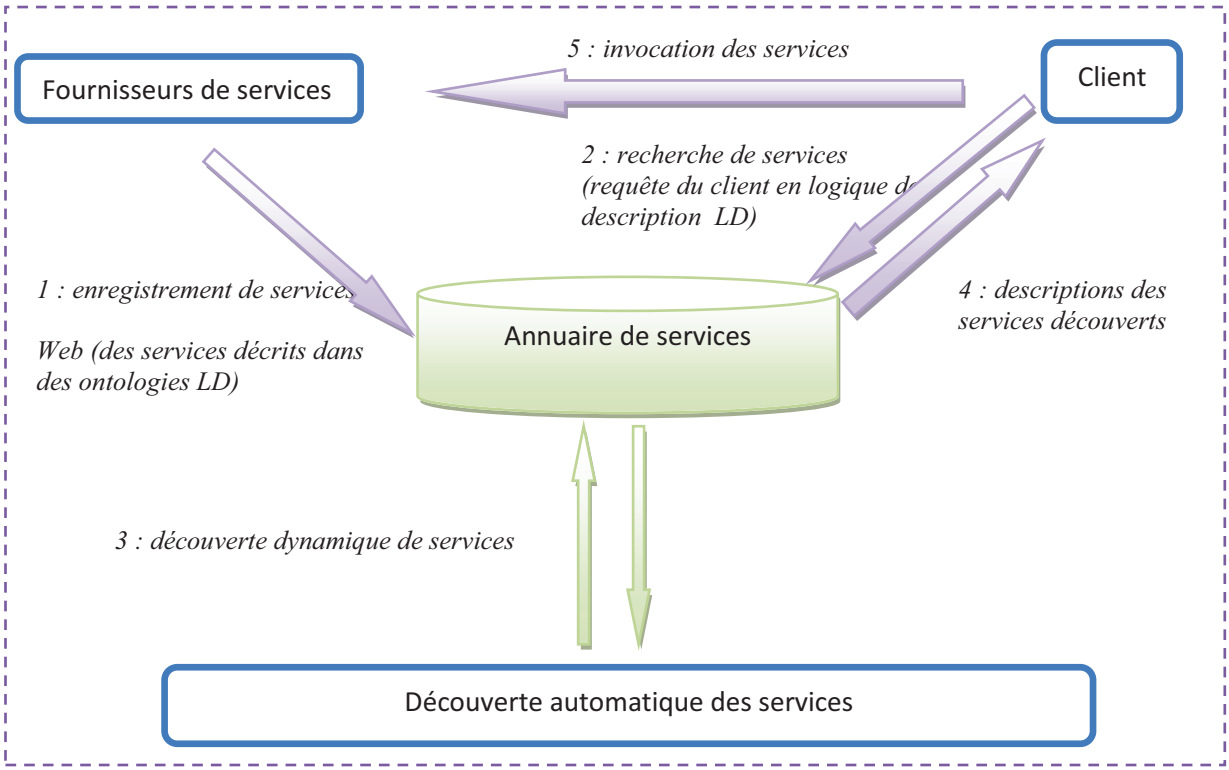

FIG 1.4 - *Etapes de découverte de services web sémantiques*

- **Le partage d'informations dans les réseaux P2P sémantiques :** Les technologies du web sémantique ne se limitent pas à la recherche des informations. Elles sont aussi utiles pour le partage des informations dans les systèmes pair à pair où chaque paire a sa propre ontologie afin de définir les concepts qu'elle manipule. Dans ce contexte, l'absence d'une ontologie globale et commune à tous nécessite d'identifier des mises en correspondance entre les concepts des différentes ontologies utilisées afin d'avoir une connaissance maximale des autres paires pour faciliter la recherche des informations précises.

1.5.3 Notions de base d'alignement

Aligner deux ontologies équivaut à mettre en correspondance les entités de ces dernières. Dans la littérature, ce problème a été traité par des méthodes d'alignement qui se basent sur

la recherche de la similarité ou dissimilarité où la relation entre deux concepts est une relation d'équivalence et parfois de subsomption. Elles reposent généralement sur le calcul des mesures de similarité qui quantifient les ressemblances sous la forme d'une valeur dans l'intervalle [0.1].

1.5.3.1 Notion de similarité sémantique

La similarité sémantique est une notion qui a été étudiée en psychologie. Elle est à la base de toute une partie du raisonnement humain. Plusieurs travaux éclairent cette notion mais la plupart ne respectent pas les qualités attendues d'une distance (minimalité, symétrie et inégalité triangulaire). Les résultats les plus intéressants sont ceux de Tversky [63], indiquant que la similarité sémantique n'est pas une distance, parce qu'elle ne satisfait pas la symétrie et l'inégalité triangulaire. Dans la comparaison de deux entités, par exemple un père et son fils, si nous disons facilement que le fils ressemble à son père, nous le faisons plus difficilement dans l'autre sens car la similarité entre les deux entités n'est pas symétrique.

Définition 1.6. « La similarité sémantique est une évaluation du lien sémantique afin d'estimer à quel point deux concepts sont proches dans leurs sens » [53].

Définition 1.7. « une fonction de similarité est définie dans un univers *U* qui peut être modélisé à l'aide d'un quadruplet : (L_d , L_s , *T, FS*) [8].

- Soit L_d le langage de représentation utilisé pour décrire les données.
- Soit L_s le langage de représentation des similarités.
- Soit T un ensemble de connaissances que l'on possède sur l'univers étudié.
- Soit FS la fonction binaire de similarité, telle que :

$$FS : L_d \times L_d \emptyset L_s$$

»

1.5.3.2 Similarité sémantique entre les concepts dans une ontologie

Les mesures de similarité prenant pour objet les concepts d'une ontologie ont été étudiées dans de nombreux travaux. Elles peuvent être classifiées selon la représentation sur laquelle elles s'appuient: arbre de la taxonomie de l'ontologie, représentations dans un espace vectoriel ou propriétés des concepts…etc.

- **Approches basées sur les arcs:** Parmi les solutions classiques pour les mesures de similarité, on peut trouver les approches basées sur les arcs qui s'appuient uniquement sur la structure de l'ontologie. Les mesures reposant sur les arcs considèrent que la similarité entre deux concepts peut être calculée à partir du nombre de liens qui séparent les deux concepts. Les deux mesures les plus utilisées sont :
  - **La mesure de Rada et al [49]** : La mesure du nombre d'arcs (edge counting) utilise une métrique *dist*(c_1 , c_2), qui indique le nombre d'arcs séparant les deux concepts c_1 et c_2 par le plus court chemin dans la hiérarchie. Plus deux concepts sont distants, moins ils sont similaires.

$$Sim\ (c_1, c_2\ ) = \frac{1}{1+\ dist(c_1\ ,c_2)} \tag{1.1}$$

  - **La mesure de Wu et Palmer [67] :** Une mesure prend en compte la profondeur des concepts dans la hiérarchie. Elle s'appuie sur le plus petit généralisant commun, c'est-à-dire le généralisant commun à c_1 et c_2 le plus éloigné de la racine. Cette mesure est définié comme suit :

$$Sim\ (c_1, c_2\ ) = \frac{2 * \ depth(c)}{depth_c(c_1)\ + depth_c(\ c_2)} \tag{1.2}$$

    Où c est le subsumant commun le plus spécifique , depth(c) est la longueur du chemin entre c et la racine de la hiérarchie, $depth_c(c_i)$ est le nombre d'arcs entre c_i et la racine en passant par c.

- **Approches basées sur le contenu informationnel :** L'approche suppose qu'on peut quantifier la similarité entre deux concepts et le moyen le plus naturel est de comparer leur contenu informatif partagé. Plus l'information est partagée par deux concepts, plus ils sont similaires. Le calcul du contenu informatif d'un concept *c* est basé sur la probabilité *P(c)* d'avoir ce concept ou un de ses descendants dans un corpus associé à l'ontologie. Il a été introduit la première fois par Resnik [52] comme suit :

$$IC(c) = -\log\ (P(c)) \tag{1.3}$$

Où *P(c)* est la probabilité de retrouver une instance du concept *c*. Elle est calculée par :

$$P(c) = \frac{freq(c)}{N} \qquad freq(c) = \sum_{n \in word(c)} count(n) \tag{1.4}$$

Où *word (c)* représente l'ensemble des termes représentant les concepts subsumés par c et *N* représente le nombre total de concepts retrouvés.

- **Mesure de Resnik [52] :** Resnik définit la similarité sémantique entre deux concepts par la quantité d'information qu'ils partagent c'est-à-dire par celle du plus petit généralisant *c* (le concept le plus spécifique subsumant c_1, c_2).

$$Sim(c_1, c_2) = \text{Max}_{c \in S(c_1,c_2)} (\log P(c)) \quad (1.5)$$

Où $S(c_1, c_2)$ est l'ensemble des concepts qui subsument c_1, c_2. Cette mesure ne prend en compte que le contenu informationnel du petit généralisant c (le concept le plus spécifique subsumant c_1, c_2 qui maximise la valeur de similarité). $Sim(c_1, c_2) \in [0\ .\ 1]$.

- **Mesure de Lin [39] :** La mesure de Lin ne tient compte non seulement du contenu informatif partagé entre deux concepts mais également de leur propre contenu informatif.

$$Sim(c_1, c_2) = \frac{2.\ \log P(c)}{\log P(c_1)\ .\ \log P(c_2)} \quad (1.6)$$

Il existe aussi des méthodes qui calculent le contenu informatif à partir de WordNet au lieu d'un corpus.

- **Mesure de Hirst & Onge [27] :** Cette mesure calcule la proximité sémantique en prenant tout type de relation dans Wordnet. Les directions des liens sont classées en lien haut (partie de), lien bas (sous classe) et lien horizontal (antonyme). La similarité est définie comme une relation qui est calculée entre termes par le poids du plus court chemin qui mène d'un synset d'un terme à un autre en fonction des changements de direction (haut, bas, horizontal).

$$Rel(c_1,\ c_2) = C - chemin - K * nd \quad (1.7)$$

Où *C, K* sont des constantes, chemin est la longueur du chemin le plus court en nombre d'arcs et *nd* est le nombre de changement de direction.

- **Mesure de Seco [59] :** Selon Seco et al , le contenu informationnel d'un concept est calculé en utilisant leur hyponymes,hypo(c), et le nombre de concepts dans la taxonomie max_{wn}.

$$\text{ic}_{\text{wn}}(c) = \frac{\log(\frac{\text{hypo}(c)+1}{\text{max}_{\text{wn}}})}{\log(\frac{1}{\text{max}_{\text{wn}}})} = 1 - \frac{\log(\text{hypo}(c)+1)}{\log(\text{max}_{\text{wn}})} \quad (1.8)$$

- **Approches basées sur une représentation vectorielle :** Ce modèle est surtout utilisé en recherche d'information pour représenter des documents où les n dimensions de l'espace vectoriel correspondent à l'ensemble des n termes (mots ou groupes de mots) qui constituent l'ensemble du corpus, et les documents sont représentés par des vecteurs V_D

dans cet espace, où chaque composante v_i correspond à la fréquence d'un terme dans le document. Parmi les mesures qui ont été dérivées de cette approche, nous citons :

- **L'indice de Jaccard :** L'indice de similarité est le nombre de mots communs divisé par le nombre total de mots moins le nombre de mots communs.

$$sim(D_1, D_2) = \frac{\sum_{i=1}^{n} v_{1i} v_{2i}}{\sum_{i=1}^{n} v_{1i} v_{1i} + \sum_{i=1}^{n} v_{2i} v_{2i} - \sum_{i=1}^{n} v_{1i} v_{2i}} \quad (1.9)$$

- **Mesure cosinus [57] :** Pour cette mesure, on utilise la représentation vectorielle complète, c'est-à-dire avec la fréquence des mots. Deux documents sont similaires si leurs vecteurs sont confondus c'est-à-dire l'angle entre les vecteurs est nul et le cosinus vaut 1 à l'opposé des documents entièrement différents qui sont représentés par des vecteurs orthogonaux donc leur similarité est nulle.

$$sim(D_1, D_2) = \frac{\sum_{i=1}^{n} v_{1i} v_{2i}}{\sqrt{\sum_{i=1}^{n} v_{1i}^2 \sum_{i=1}^{n} v_{2i}^2}} \quad (1.10)$$

- **Approches basées sur les propriétés :** Tversky [63] propose un modèle qui tient compte des parties communes et distinctives entre deux concepts : plus les concepts partagent de propriétés, et moins ils ont de propriétés distinctives, plus ils sont similaire.

$$S(c_1, c_2) = \frac{f(C_1 \cap C_2)}{f(C_1 \cap C_2) + \alpha f(C_1 - C_2) + \beta f(C_2 - C_1)} \quad (1.11)$$

Où C_1 est l'ensemble des propriétés de c_1 et C_2 est l'ensemble des propriétés de c_2, $\alpha \geq 0$, $\beta \geq 0$ sont des paramètres qui pondèrent les différences et f est une fonction qui quantifie l'information que porte chaque concept. Dans ce cas, $f(C_1 \cap C_2)$ mesure l'information commune portée par les deux concepts et, $f(C_1 - C_2)$ et $f(C_2 - C_1)$ mesurent l'information spécifique que porte chaque concept par rapport à l'autre.

- **Approches hybrides :** Ces approches sont basées sur un modèle qui combine les propriétés des modèles précédents pour produire un nouveau modèle qui considère plusieurs aspects (arcs, contenu informatif, ...etc.). Jiang et Conrath **[35]** proposent une mesure qui combine entre les approches basées sur les arcs et les approche basées sur le contenu informatif.

$$distance(c_1, c_2) = IC(c_1) + IC(c_2) - (2.IC(c)) \quad (1.12)$$

Où c est le plus petit généralisant. La mesure de similarité devient donc :

$$Sim(c_1, c_2) = \frac{1}{distance(c_1, c_2)} \quad (1.13)$$

1.5.4 Classification des méthodes d'alignement

Dans la littérature, la classification la plus citée dans les travaux qui traitent le problème de matching est celle proposée par *Rahm* et *Bernstien* [50]. Elle distingue deux approches de matching de schéma de base de données : l'approche individuelle, et l'approche combinée (Voir FIG 1.5).

a) **L'approche individuelle :** Le système d'alignement utilise un seul algorithme pour exécuter le matching. Les algorithmes sont classés selon les critères suivants :
   - Instances vs. Schéma : On distingue deux approches, la première est basée sur les instances de données, et la deuxième est basée sur les schémas de données qui incluent souvent les propriétés des éléments de schéma, telles que le nom, le type de données, etc.
   - Elément vs. Structure : On distingue deux niveaux de granularité de matching ; élément et structure. Dans le niveau élément le matching est réalisé sur les entités des schémas qui sont analysées d'une manière isolées. En revanche, dans le niveau structure le matching est réalisé sur les relations qui relient les entités entre elles.
   - Langage vs. Contrainte : on distingue les matchers utilisant des approches linguistiques et celles utilisant des contraintes comme les types de données, les clés des relations.
   - Cardinalité : La cardinalité de matching peut être classée en deux catégories. La première catégorie est le matching direct qui correspond à une cardinalité de matching (1:1). La seconde catégorie est le matching indirect qui couvre les cardinalités de type (1:n), (n:1) et (n:m) qui signifie qu'il peut y avoir n éléments d'un schéma et m éléments d'un autre schéma qui se correspondent.
   - Informations auxiliaires : plusieurs matchers dépendent non seulement des schémas d'entrée, S1 et S2, de matching mais aussi des informations auxiliaires comme les thésaurus et les dictionnaires (synonymes, hyperonymie, …..).

b) **L'approche combinée :** Le système d'alignement utilise plusieurs algorithmes pour exécuter le matching selon deux approches :

- L'approche hybride : consiste en une utilisation séquentielle et multicritère des algorithmes pour exécuter le matching.
- L'approche composite : les algorithmes sont utilisés de façon indépendante. Les résultats fournis par ces algorithmes sont combinés a fin d'obtenir le résultat final de matching.

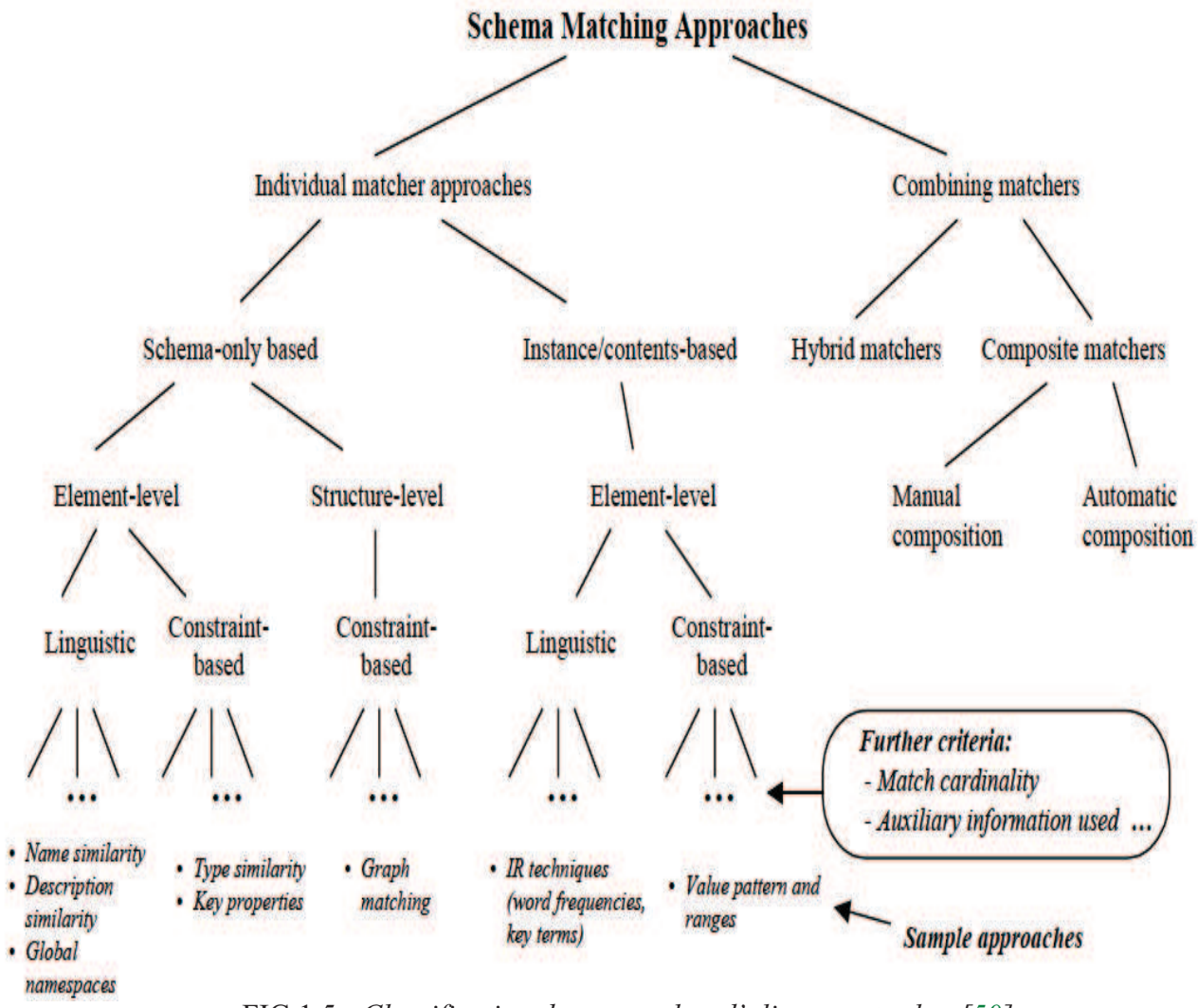

FIG 1.5 - *Classification des approches d'alignement selon [50]*

La classification la plus récente, présentée en FIG 1.6, est proposée par *Shvaiko* et *Euzenat* [50] où les auteurs se sont intéressés aux approches de matching de schémas/ontologies. La classification a été lue d'une façon descendante en se focalisant sur la granularité des informations d'entrées (niveau élément ou niveau structure). Dans ces niveaux on distingue les techniques syntaxiques, sémantiques et externes. La classification a été lue aussi d'une façon ascendante en se focalisant sur le type des informations d'entrées. Dans le premier niveau on distingue les approches terminologiques, structurelles, extensionnelles et

sémantiques. Celles-ci exigent un raisonnement sémantique. Dans le niveau suivant on distingue les techniques basées sur les chaines de caractères ou linguistiques dont les approches terminologiques, et les approches structurelles qui sont divisées en deux techniques : celles qui considèrent la structure interne des entités (nom, type..) et celles qui considèrent les relations entre les entités.

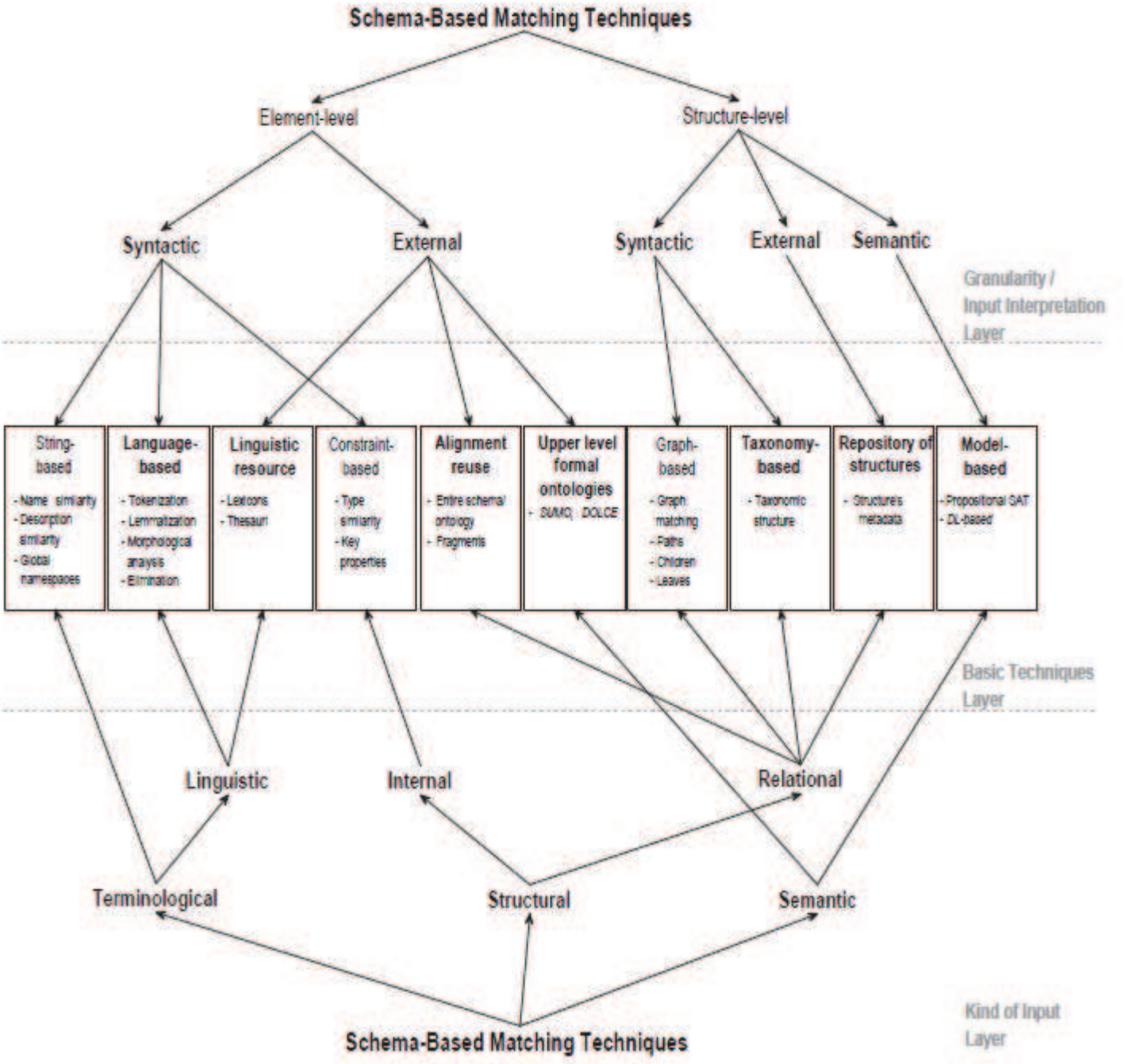

FIG 1.6 - *Classification des techniques d'alignement selon [60]*

Dans les systèmes d'alignement, on peut utiliser des mesures simples ou combinées pour calculer la similarité entre les entités. Ces mesures peuvent être classées selon la nature des entités alignées : des termes, des structures, des instances, ou des modèles théoriques.

Conclusion

Dans ce chapitre, nous avons tout d'abord présenté le cadre générale du web sémantique, en mettant l'accent sur la notion d'ontologie, ses composants, ainsi que les langages du web sémantique. Nous nous sommes ensuite intéressés aux problèmes de l'hétérogénéité des ontologies ainsi qu'aux défis confrontés pour l'acheminement vers le web sémantique. L'alignement d'ontologies s'avère au cœur de la résolution de ses problèmes. Une brève présentation de ses notions de base, et de la classification des méthodes d'alignement a été abordée. Une étude plus détaillée de ces techniques fera l'objet du chapitre suivant.

Etude des techniques d'alignement 2

Sommaire

Introduction

De nombreuses méthodes d'alignement dédiées aux ontologies ont vu le jour cette dernière décennie. Ces méthodes ont été étudiées dans ([50], [60]) qui les ont classées en fonction des techniques , des types d'informations, des critères qu'elles utilisent, et de la façon dont elles les exploitent.

Ce chapitre a pour premier objectif d'aborder les techniques d'alignement dans le domaine du web sémantique en présentant quelques critères sur lesquels une comparaison doit s'appuyer pour les classifier. Dans un second temps, on s'intéresse à l'étude de deux méthodes d'alignement, Falcon-AO et TaxoMap ,qui s'attaquent au problème du passage à

l'échelle des techniques d'alignement. Ces deux méthodes visent à partitionner les ontologies avant l'alignement et présentent une source d'inspiration pour nos travaux de recherche.

2.1 Les dimensions externes d'une méthode d'alignement

On peut distinguer différentes familles de méthodes d'alignement classées par rapport à leurs dimensions externes qui représentent les entrées, et les sorties utilisées et que nous détaillons dans ce qui suit :

2.1.1 Données d'entrées

Une méthode d'alignement est généralement conçue pour aligner certains types de structures qui peuvent être des schémas de base de données (de type relationnel, XML, objet), des ontologies (décrites en OWL, RDFS,...etc.) ou des instances d'une base de donnés ou d'une ontologie.

2.1.2 Données en sorties

On peut distinguer différentes familles de méthodes par rapport à leur alignement qu'elles produisent en sortie. L'alignement fournit un ensemble de correspondances qui décrit une relation sémantique entre les entités des différentes ontologies. Une correspondance identifie une entité e dans une première ontologie, une entité e' dans une seconde ontologie et une relation r se tenant entre e et e'. En effet, la grande majorité des méthodes associe une mesure de confiance aux éléments de correspondances trouvés et s'intéresse seulement à la relation d'équivalence (=). Cependant d'autres méthodes distinguent les relations de disjonction ($\perp$), plus général ($\supseteq$)…etc.

Une méthode d'alignement est caractérisée par la cardinalité d'alignement qu'elle produit 1-1, 1-*, …etc. La plupart des méthodes met en correspondance chaque élément d'une structure source à un élément de la structure cible.

Divers formats ont été proposés pour représenter des alignements dans le but de les stocker, les comparer, les combiner ou les utiliser dans les opérations d'intégration, de négociation des agents…etc.

- OWL : On peut utiliser le langage OWL pour exprimer les correspondances entre les ontologies. En fait, les primitives equivalentClass et equivalentProperty ont été introduites pour relier des éléments d'ontologies décrivant le même domaine de connaissance[46].

- C-OWL : C-OWL est une extension du langage OWL utilisée pour exprimer des relations entre les concepts ou les rôles d'ontologies décrites en OWL, mais hétérogènes du point de vue de leur contexte [9]. Les constructeurs de C-OWL sont appelés "passerelles" (bridge rules) et elles permettent d'exprimer les correspondances entre les contextes : une passerelle entre les contextes de deux ontologies O_i et O_j permet de déclarer une correspondance entre des éléments existants dans chacun de ces deux ontologies. Par exemple, une passerelle de la forme $i: C \xrightarrow{\sqsubseteq} j: D$ indique que la classe C telle qu'elle est représentée dans le contexte O_i est vue par le contexte O_j comme étant plus spécifique que la classe D.
- SWRL : SWRL (Semantic Web Rule Language) est basé sur une combinaison de OWL (Lite et DL) [42] et de Rule Markup Language (Datalog RuleML unaire/binaire) [28]. Il est prévu pour supporter les raisonnements reposant sur la logique de description et les règles de Horn. Une règle se compose d'un antécédent ou "corps" et d'un conséquent ou "tête". Une règle du type conséquent ← antécédent signifie que si l'on a pu démontrer la partie antécédent, alors la partie conséquent est vraie. Ces règles peuvent être vues comme des correspondances entre les ontologies lorsque les entités mises en jeu dans l'antécédent et le conséquent appartiennent à des ontologies différentes.

2.2 Les dimensions internes d'une méthode d'alignement

Si nous rentrons dans le détail du fonctionnement d'une méthode d'alignement, nous pouvons distinguer différentes méthodes selon leurs caractéristiques internes. Ces caractéristiques concernent principalement:

- La représentation interne de deux ontologies
- Les techniques utilisées
- La mesure de similarité choisie
- Le pattern d'exploitation des ontologies
- Le paramétrage et la composition des méthodes

Dans ce qui suit, nous présentons ces différentes caractéristiques :

2.2.1 La représentation interne de deux ontologies

Généralement les ontologies à aligner sont transformées sous forme d'un graphe afin d'améliorer les performances d'un algorithme. Le but de la représentation des ontologies en graphe est d'encoder les connaissances dans des nœuds et les relations sémantiques entre les nœuds dans des arcs. Cela réduit beaucoup le temps de calcul nécessaire chaque fois que l'algorithme devra chercher un concept dans les fichiers (OWL,RDF…) et rend l'algorithme plus rapide. Dans ce qui suit, nous présentons les représentations internes de quelques méthodes d'alignement:

- OLA [18] transforme les ontologies sous forme d'un graphe OL-graphs qui représente toutes les caractéristiques possibles des ontologies représentées en OWL-Lite. Les nœuds du graphe sont : classe (C), objet (O), relation (R), propriété (P), instance de propriété (A), type de donnée (D), valeur de type de donnée (V), les labels de restriction de propriété (L) . Les arcs entre les nœuds sont :
  - *rdfs:subClassOf entre deux classes ou deux propriétés (S);*
  - *rdf:type (I) entre les objets et les classes, instance de propriété et propriété, valeurs et types de données ;*
  - *A entre les classes et les propriétés, objets et les instance de propriétés ;*
  - *owl:Restriction (R) exprimant la restriction sur une propriété dans une classe ;*
  - *valuation (U) d'une propriété d'un individu.*
- ASCO3 [1] représente les ontologies sous forme d'un graphe étiqueté, orienté et cyclique, appelé O-Graphe. Les entités de l'ontologie (classes, relations, instances) sont des nœuds du graphe, deux nœuds sont liés par un arc orienté et étiqueté par la primitive de OWL. Un O-Graphe est un 4-uplet og = (V, E, a, b), où
  - V est l'ensemble fini de sommets (également appelés les nœuds)
  - $E \subseteq V \times V$ est l'ensemble d'arcs
  - $\alpha : V \rightarrow L$ est une fonction affectant une étiquette à chaque sommet
  - $\beta : E \rightarrow L$ est une fonction affectant une étiquette (type) à chaque arc

  arc(u,v) est un arc orienté, commençant au nœuds u et se terminant au nœuds v.

- EDOLA [69] transforme les ontologies sous forme d'un graphe OWL-Graphe. Les nœuds du graphe représentent les six types d'entités qui existent dans une ontologie OWL-Lite : les concepts, les instances des concepts, les types de données, les valeurs des types de données et les propriétés des classes. Les relations entre les entités au niveau de l'ontologies OWL-Lite sont les arcs entre les nœuds. Le graphe OWL-Graphe permet de représenter quatre catégories de liens de spécialisation, d'attribution, d'instanciation et d'équivalence.
- SODA [70] transforme les ontologies sous forme d'un graphe DL-Graphe. Les nœuds du graphe sont les entités de l'ontologie : les classes, les propriétés et les instances. Les arcs du graphe décrivent les relations qui existent entre ces différentes entités.
- FALCON [29] utilise les graphes RDF directs bipartis pour représenter les ontologies. Ces graphes sont dérivés des graphes RDF bipartis [26]. La figure 2.1 montre le graphe RDF et le graphe RDF biparti d'une ontologie O_A .

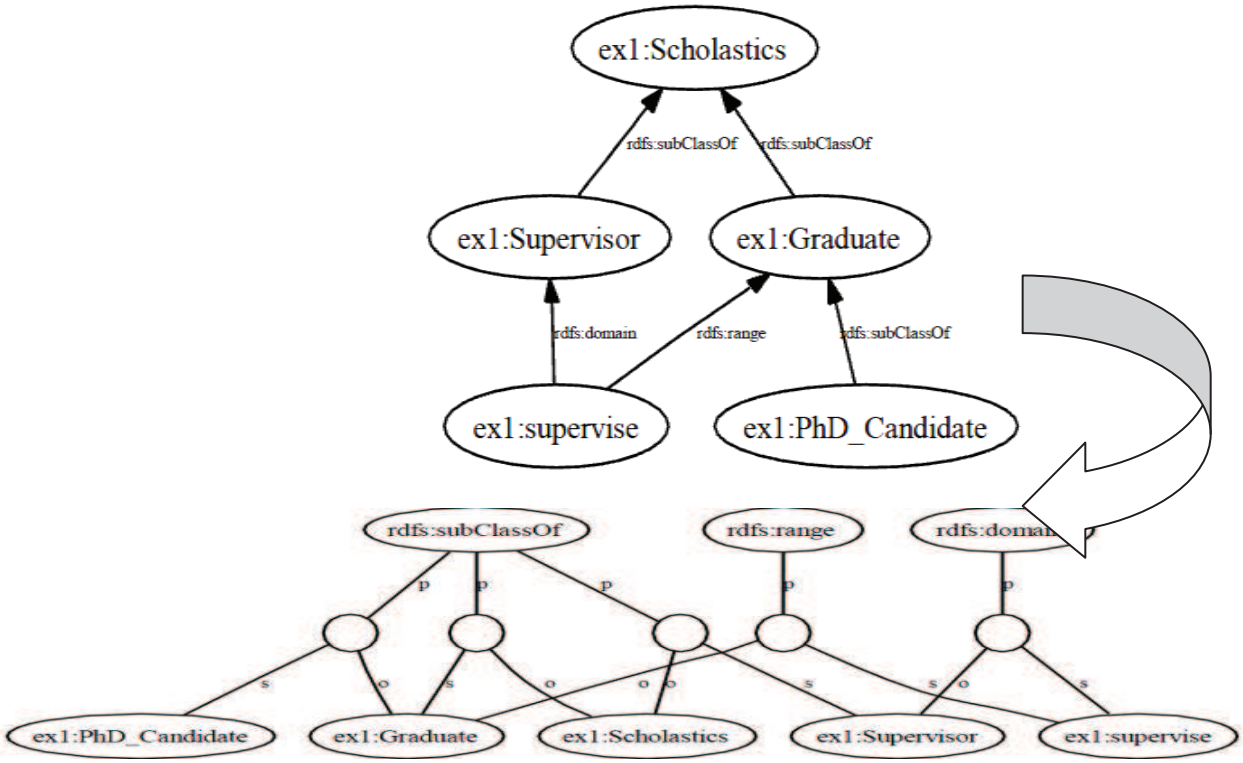

FIG 2.1 - *le graphe RDF et le graphe RDF biparti d'une ontologie* O_A

Dans la section 3.1, nous allons présenter notre représentation interne des ontologies qui nommée Graph-AO.

2.2.2 Les techniques utilisées

L'alignement d'ontologies utilise différents algorithmes basés sur des techniques qui exploitent différents types d'informations : les entités, leur position ou leur sémantique dans l'ontologie…etc. Nous détaillons ci-dessous les différentes techniques utilisées pour aligner les ontologies.

a) **Les méthodes terminologiques :** Les méthodes terminologiques se basent sur la comparaison des chaînes de caractères. Elles peuvent être employées pour mesurer la similarité entre les entités textuelle des ontologies comme les noms, les étiquettes et les commentaires. Ces méthodes peuvent être divisées en deux sous catégories : méthodes syntaxiques et linguistiques.

   - **Les méthodes syntaxiques :** Ces méthodes se basent sur la structure des chaines à comparer, plus elles partageront de caractères en commun, plus elles seront similaires. Généralement ces méthodes exigent un prétraitement qui consiste à normaliser les chaines à comparer avant de les fournir aux fonctions calculant la similarité.
   - **Les méthodes linguistiques :** Nous pouvons déduire la similarité entre les termes en s'appuyant sur des connaissances de la langue naturelle (les méthodes intrinsèques) et / ou sur des vocabulaires et des dictionnaires (les méthodes extrinsèques) .

b) **Les méthodes structurelles :** Ce sont les méthodes qui déduisent la similarité entre deux entités en exploitant leurs positions dans une hiérarchie. On peut distinguer entre deux méthodes structurelles. L'une qui n'exploite que des informations concernant des attributs d'entités (les méthodes structurelle interne) et l'autre qui considère des relations entre des entité (les méthodes structurelles externes). Dans le cadre de nos travaux, nous nous intéressons à cette technique pour déduire la similarité entre deux entités.

   - **Les méthodes internes :** La structure interne d'une méthode représente les attributs possède par l'entité (les attributs de types simple , cardinalité, restrictions). Les méthodes de cette classe exploitent ces attributs pour déduire les similarités elles couramment utilisé dans les cas où les entités ont des définitions intentionnelles précises. Les premiers systèmes qui se basent sur ce principe sont les systèmes d'intégration et l'alignement de schémas de bases de données puis reprise dans le contexte d'alignement d'ontologie.

- **Les méthodes externes :** Les méthodes structurelles externes ont la possibilité d'aligner deux entités en s'appuyant sur leurs voisines. Une entité d'une ontologie peut avoir trois type de voisinage : ses super-entités, ses sous-entités, ses sœurs. L'idée de base est que deux entités sont similaires, si leurs voisines sont similaires[18].

c) **Les méthodes extensionnelles :** Afin de déduire la similarité entre deux entités, les méthodes extensionnelles comparent leurs extensions c-à-d leurs ensembles des instances. Pour cela, des mesures de similarité entre les ensembles ont été adaptées pour construire des mesures extensionnelles telle que la mesure de Jaccard et la distance de Hamming. Ces mesures produisent la similarité de deux entités qui est en fait la similarité entre les deux ensembles de leurs instances en se basant sur la comparaison exacte des éléments dans deux ensembles[1].

d) **Les méthodes sémantiques:** Les méthodes sémantiques se basent sur des modèles logiques (la satisfiabilité propositionnelle (SAT), la SAT modale ou les logiques de descriptions) et sur des méthodes de déduction pour déduire la similarité entre deux entités. Parmi les méthodes sémantiques, [20] propose la méthode S-Match qui emploie des techniques de satisfiabilité propositionnelle où le problème d'alignement est vu comme un problème de satisfiabilité d'un ensemble de formules du calcul propositionnel.

2.2.3 La mesure de similarité

Comme nous l'avons déjà évoqué précédemment en section 1.5.4 ,de nombreuses mesures sémantiques ont été développées et qui prennent pour objet les concepts d'une ontologie. L'utilité de ces mesures de similarité est de pouvoir comparer les ressemblances et les différences entre deux concepts de deux ontologies. Cette comparaison sémantique est devenue une opération nécessaire dans de nombreux domaines (e.g. Le traitement du langage naturel, la gestion des documents, la bioinformatique, la recherche d'information, la gestion des compétences ([24], [2], [68], [40], [12]…etc.). Cependant, comment choisir entre ces mesures et laquelle fournira les résultats les plus pertinents dans un contexte donné(domaine) ?

1. Choisir en raison de sa simplicité d'implémentation [67] ;
2. Choisir en raison de la disponibilité des ressources supplémentaires (e.g. un corpus du domaine) [52];

3. Choisir en raison de la simplicité de l'hiérarchie des ontologies utilisées (e.g des liens simples de type is-a comme les thesaurus) [49], [67] ;
4. Choisir en raison de type d'information exploitées et attendues (e.g les concept comme des termes dans les ontologies où bien comme des structures et des définitions dans les ontologies) [52], [39], [35].

2.2.4 Pattern d'exploitation d'ontologie pour aligner

La similarité entre deux entités (classe ou propriété) est déduite des similarités partielles entre leurs composantes. Les méthodes d'alignement d'ontologie peuvent considérer plusieurs aspects d'exploitation des ontologies. Une entité est considérée selon deux modèles : soit elle est divisée en plusieurs morceaux atomiques(ensemble de triplets), ou elle est considérée comme une seule structure plus vaste de plusieurs composants. Les valeurs de la similarité partielle sont ensuite agrégées dans une somme pondérée pour obtenir une meilleure valeur de similarité finale de deux entités.

Le premier modèle, qui a été adopté dans plusieurs méthodes d'alignement telles que dans (ASCO2[1], Falcon-AO[33]), s'applique à des ontologies où les descriptions d'une classe et d'une relation de l'ontologie est un ensemble de triplets RDF. La comparaison de similarité entre deux entités revient à comparer les deux ensembles de triplets représentant les deux entités.

Le deuxième modèle utilisé dans (méthode de Rodriguez[56] , QOM[17]) s'applique à des ontologies où les classes sont considérées comme une seule structure qui se compose de plusieurs caractéristiques. La figure 2.2 présente le modèle de la méthode de Rodriguez.

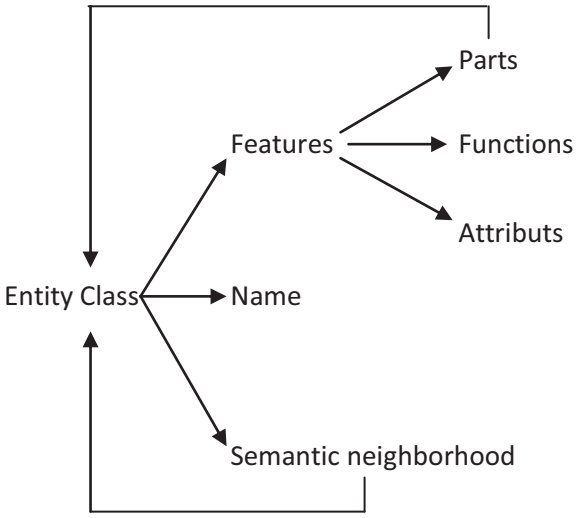

FIG 2.2 - *Les composants essentiels dans une représentation d'une entité classe [56]*

La similarité entre les entités de classes est évaluée par une somme pondérée des similarités calculées selon trois catégories : les propriétés, le nom et le voisinage sémantique des entités dans le graphe. Les propriétés peuvent être les attributs, qui caractérisent intrinsèquement les entités, les parties qui peuvent être des propriétés intrinsèques ou des éléments ayant une relation *part-of* avec les entités, et les fonctions qui caractérisent le comportement et le rôle des entités. Le voisinage d'un concept est l'ensemble des entités de classe qui se trouvent à un rayon r (défini selon le besoin) du concept dans le graphe.

2.2.5 Le paramétrage et la composition des méthodes

Les techniques d'alignement ont pour but de découvrir les correspondances entre les ontologies aussi proche que possible de celles faites par un expert. Ces technique peuvent être combinées afin d'obtenir des résultats convenables. Cette combinaison aboutit à des algorithmes très robustes et performants. Rahm et Bernstien [50] distinguent entre deux manières de combiner les différentes approches au sein de la méthode globale :

- L'hybridation qui permet de combiner, en un seul algorithme, plusieurs approches basées sur différents critères et types d'information. Les méthodes hybrides ont l'avantage d'obtenir de meilleures performances que l'exécution de plusieurs algorithmes individuels séparément ;
- La composition qui permet de combiner les résultats produits par plusieurs algorithmes exécutés de manière indépendante. Les méthodes composites ont l'avantage d'être modulaires et ainsi d'être adaptables plus facilement à différentes représentations de structures d'entrée.

Le résultat final de la méthode globale combine les différents résultats issus des algorithmes d'alignement utilisés (différentes techniques). Souvent ces algorithmes possèdent de nombreux paramètres à ajuster manuellement par un expert du système. Ces paramètres peuvent porter sur l'alignement d'entrée, le seuil de similarité, le critère d'arrêt de l'algorithme itératif de propagation de similarité, le paramètre de *stemmer*[2] (la langue des ontologies : française, anglaise, allemande ..), les pondérations utilisées par les algorithmes, et également les ressources externes (Wordnet[3]) sur lesquelles s'appuie la méthode.

[2] Un stemmer est un programme ou un algorithme qui détermine la forme radicale à partir d'une forme infléchie ou dérivée d'un mot donné

[3] http://www.cogsci.princeton.edu/~wn/

2.3 Comparaison des méthodes d'alignement

Dans cette section nous comparons cinq méthodes d'alignement selon leurs dimensions externes puis selon leurs dimension internes.

2.3.1 Comparaison selon les dimensions externes d'une méthode d'alignement

Les méthodes que nous avons choisies dans notre comparaison , sont souvent citées dans la littérature comme des méthodes d'alignement d'ontologie dans le domaine du web sémantique. La plupart de ces méthodes ont été conçues pour aligner des ontologies RDFS/OWL et elles produisent en sortie des alignements au format RDF/XML. La table 2.1 présente leur comparaison selon les dimensions externes.

Au niveau des relations détectées, nous remarquons que la plupart des méthodes comparées se limite à l'équivalence. En effet, seule la méthode TaxoMap détecte la subsomption. Seule la méthode ASCO2 laisse un choix quant à la cardinalité des alignements qu'elle produit.

Méthode	Données d'entrées	Données sorties		
	Type de schémas	format	relation	cardinalité
Falcon-AO[33]	RDFS/OWL	RDF/XML	≡	0,1-0,1
TaxoMap[54]	OWL/RDF (taxonomies)	RDF/XML	≡,⊆	0,1-0,1
OLA[18]	RDFS/OWL	RDF/XML	≡	0,1-0,1
ASCO3[1]	OWL DL/Lite	RDF/XML	≡	0,n-0,n
QOM [17]	RDFS/OWL	RDF/XML	≡	0,1-0,1

TAB 2.1 - *Comparaison des méthodes par rapport à leurs dimensions externes*

2.3.2 Comparaison selon les dimensions internes d'une méthode d'alignement

La table 2.2. montre, pour chaque méthode d'alignement :

- sa représentation interne : généralement les méthodes transforment les ontologie à aligner en graphes ;
- sa technique utilisée : ces méthodes sont des méthodes structurelles et terminologiques ;
- sa mesure de similarité : distance d'édition, TF/IDF, Jaro-Winkler…etc. ;
- son pattern d'exploitation des ontologies : l'ontologie est un ensemble de triplet RDF comme ASCO2, où chaque entité est vue comme une seule structure de plusieurs composants comme QOM ;
- son paramétrage et sa composition : nous remarquons que toutes les méthodes présentées se composent de différentes techniques, et la plupart d'entre elles a une combinaison parallèle avec des moyennes pondérées. Quelques méthodes s'appuient également sur la fonction sigmoïde. La méthode TaxoMap utilise une combinaison séquentielle. Ces méthodes s'appuient, en générale , sur un seuil de similarité pour sélectionner les correspondances.

Méthodes	représentation interne	techniques utilisées	mesure de similarité	Pattern d' exploitation	paramétrage et composition
Falcon-AO [33]	Graphe direct bipartie	Terminologique / structurelle	Edition distance, TF.IDF	RDF triplets	Composition(intégration d'alignement), critère d'arrêt de l'algorithme itératif de propagation de similarité, seuil de similarité, la langue utilisé dans les ontologies(stemmer).
TaxoMap [38]	Format TaxoMap (seulement les étiquettes et les sous classes sont prends en considération)	Terminologique / structurelle	techniques basés sur la mesure de similarité de Lin (SimLinLike)	Caractéristiques des entités(la richesse des étiquettes des entités)	Composition(séquentielle), Les catégories de mots considérées par Treetagger , les différents seuils de similarité (Equiv.threshold, HiddenInc.thresholdSim,..), la langue utilisé dans les ontologies.
OLA [18]	OL-Graphe	Terminologique / structurelle	Hamming entre les synsets, système d'equation-s interdependentes	Caractéristiques des entités (descriptive knowledge about a couple of entities)	Composition(moyenne pondérée, parallèle),pondérations(poids), seuil de similarité, un alignement en entrée, WordNet.
ASCO2 [1]	Format RDF	Terminologique / structurelle	Jaro-Winkler(id,labels,synset) , TF.IDF	RDF triplets	Composition(la somme pondérée avec les poids variables, parallèle), pondération variable (poids),seuil de similarité, Wordnet.
QOM [17]	Format RDFS	Terminologique / structurelle	Distance d'édition	Caractéristiques des entités	Composition(moyenne pondérée, fonction sigmoïde), critère d'arrêt de l'algorithme itératif de propagation de similarité, seuil de similarité.

TAB 2.2 - *comparaison des méthodes par rapport à leurs dimensions internes*

2.4 Système d'Alignement : Ontologie simple Vs. Ontologie large

Dans les domaines d'applications réelles où les ontologies sont volumineuses et complexes, les exigences de l'exécution du temps et de l'espace de mémoire sont les deux facteurs significatifs qui influencent directement la performance d'un algorithme d'alignement. L'une des solution de passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser l'alignement.

Dans la littérature actuelle, il existe très peu d'algorithmes d'alignement qui visent à partitionner des ontologies en blocs avant de réaliser l'alignement. Nous avons répertorié deux travaux de découverte de correspondances qui ont été proposés dans le passé comme des méthodes d'alignement linguistique et structurelle mais actuellement ils intègrent des algorithmes de partitionnement adaptés au contexte d'alignement des ontologies volumineuses. Il s'agit des deux méthodes de Falcon-AO et TaxoMap que nous présentons d'abord leur utilisation dans le contexte d'alignement des ontologies simples puis dans le contexte d'alignement des ontologies larges.

2.4.1 Falcon-AO et TaxoMap pour aligner les ontologies simples

Dans cette section nous présentons les deux méthodes, Falcon-AO et TaxoMap, avant et après l'intégrations des modules de partitionnement.

2.4.1.1 Falcon-OA (2005)

Falcon est un outil d'alignement qui a été développé par Wei Hu et al ([29], [33]) afin de découvrir automatiquement les correspondances entre des ontologies en exploitant la structure et le langage de ces ontologies. Deux algorithmes sont intégrés dans Falcon : LMO (le matching linguistique) qui se base sur la similarité linguistique des nœuds, GMO (le matching structurel) qui se base sur la similarité structurelle des nœuds. La figure suivante illustre l'architecture du système Falcon-OA.

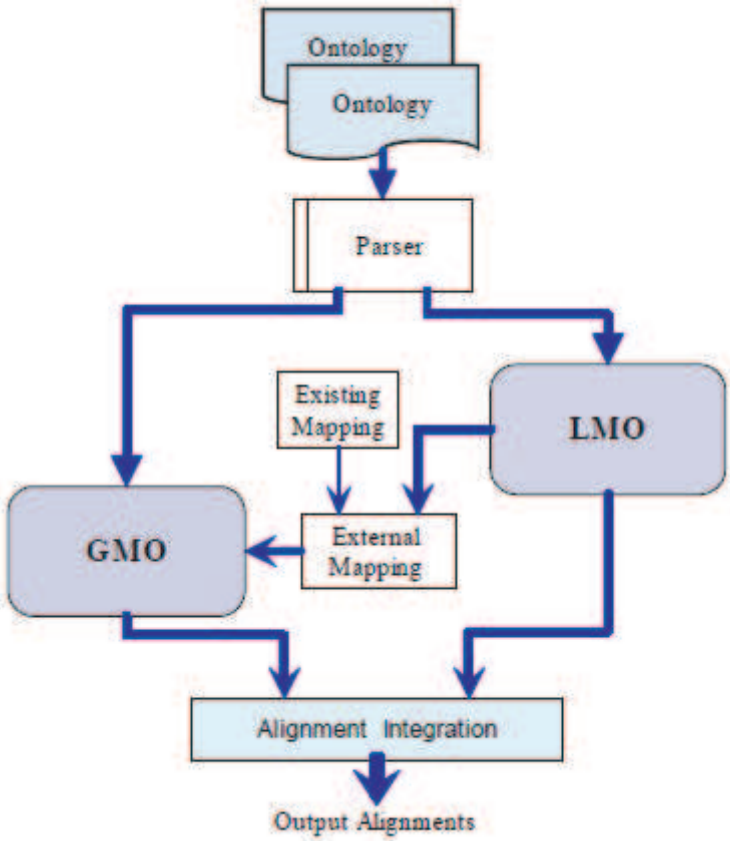

FIG 2.3 - *Architecture du système Falcon[33]*

- **LMO (le matching linguistique) :** LMO combine deux approches d'analyse, l'une lexicale et l'autre statistique. Dans l'approche lexicale, la distance d'édition entre les noms des entités a été calculée ensuite une fonction a été utilisée pour calculer la similarité des chaines de caractères. Dans l'approche statistique, le modèle d'espace vectoriel a été utilisé. Une collection des documents a été générée où chaque document correspond à une entité dans l'ontologie. Le document virtuel d'une entité consiste à extraire un « sac de termes » à partir d'une information textuelle des entités et l'information de leurs voisins. Chaque document est représenté comme un vecteur dans un espace de dimension n. Enfin, la comparaison lexicale et l'analyse statistique sont combinées
- **GMO (le matching du graphe) :** GMO utilise les graphes directs bipartis pour représenter les ontologies et les mesures de similarité structurelle entre les graphes[29]. La similarité entre deux entités est déterminée en revenant sur la similarité de leurs triplets où les entités prennent le même rôle (sujet, prédicat, objet). GMO prend un ensemble de couples d'entités alignées qui sont générés par LMO et tente de générer des nouveaux alignements par la comparaison de la similarité structurelle. Le GMO est calculé et si les

valeurs obtenues dépassent un seuil fixé au départ, ces valeurs sont ajoutées à l'alignement final sinon elles sont jetées.

2.4.1.2 TaxoMap (2007)

TaxoMap est un outil d'alignement qui a pour objectif de permettre un accès unifie via le Web aux documents d'un même domaine d'application. Il est adapté au traitement de taxonomies dont les structures sont hétérogènes et dissymétriques. L'objectif de TaxoMap est de mettre en correspondance les concepts de la taxonomie la moins structurée, la taxonomie source (T_{Source}), avec les concepts de la taxonomie la plus structurée, la taxonomie cible (T_{Cible}). Le processus d'alignement est donc orienté de T_{Source} vers T_{Cible} [54] (voir FIG 2.4).

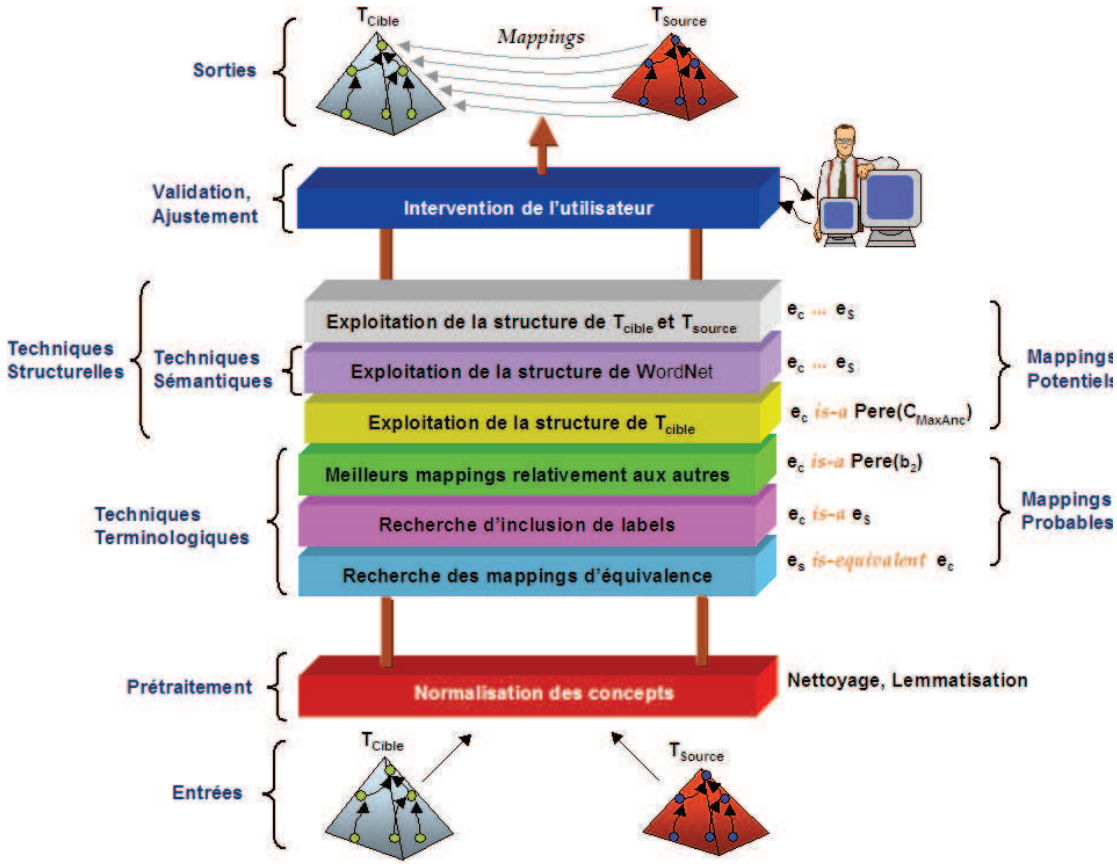

FIG 2.4 - *Le Système TaxoMap[38]*

Les schémas en entrée du processus de mise en correspondance sont des taxonomies, correspondant à des ontologies très sommaires avec des définitions de concepts très pauvres. Les concepts sont principalement définis par référence à leur terminologie. Ils n'ont pas d'attributs. Une taxonomie est généralement représentée par un graphe acyclique dont les

nœuds sont les concepts et les arcs correspondent aux liens de sous-classes. Dans TaxoMap, Plusieurs techniques sont utilisées : des techniques terminologiques puis structurelles.

- **La technique terminologique :** La technique terminologique est basée sur la mesure de similarité de Lin [39]. Le calcul est effectué entre chaque concept de la taxonomie source T_{Source} et tous les concepts de la taxonomie cible T_{Cible}, mais avec une adaptation pour prendre en compte l'importance des mots dans les expressions. Cette mesure permet de calculer *MC*, l'ensemble des concepts de T_{Cible} candidats au mapping.
- **La technique structurelle :** La technique structurelle exploite les concepts candidats à un mapping, (MC), avec un concept c_S de T_{Source}. Lorsqu'il n'a pas été possible de générer un mapping probable avec l'un de ces candidats, l'idée consiste à exploiter leur position dans T_{Cible}. Leur proximité dans le graphe est assimilée à une proximité sémantique.

2.4.2 Falcon-AO et TaxoMap pour aligner les ontologies larges

Dans les versions actuelles, Falcon-AO et TaxoMap intègrent des algorithmes de partitionnement adaptés au contexte d'alignement des ontologies volumineuses.

2.4.2.1 Falcon-AO (2007)

Le système proposé est divisé en trois phases comme le montre la figure (FIG 2.5) : la première phase consiste à effectuer un partitionnement. En effet, ce dernier a comme but de partitionner chaque ontologie d'entrée à un ensemble de petits clusters et cela en utilisant la proximité structurelle entre ses entités puis en construisant les blocs via les phrases RDF. La deuxième phase consiste à sélectionner les blocs à aligner et cela en utilisant les ancres pré-calculés. La troisième phase consiste à exécuter l'algorithme de Falcon (le matching linguistique : V_DOC, le matching structurel : GMO) entre les blocs sélectionnés[31].

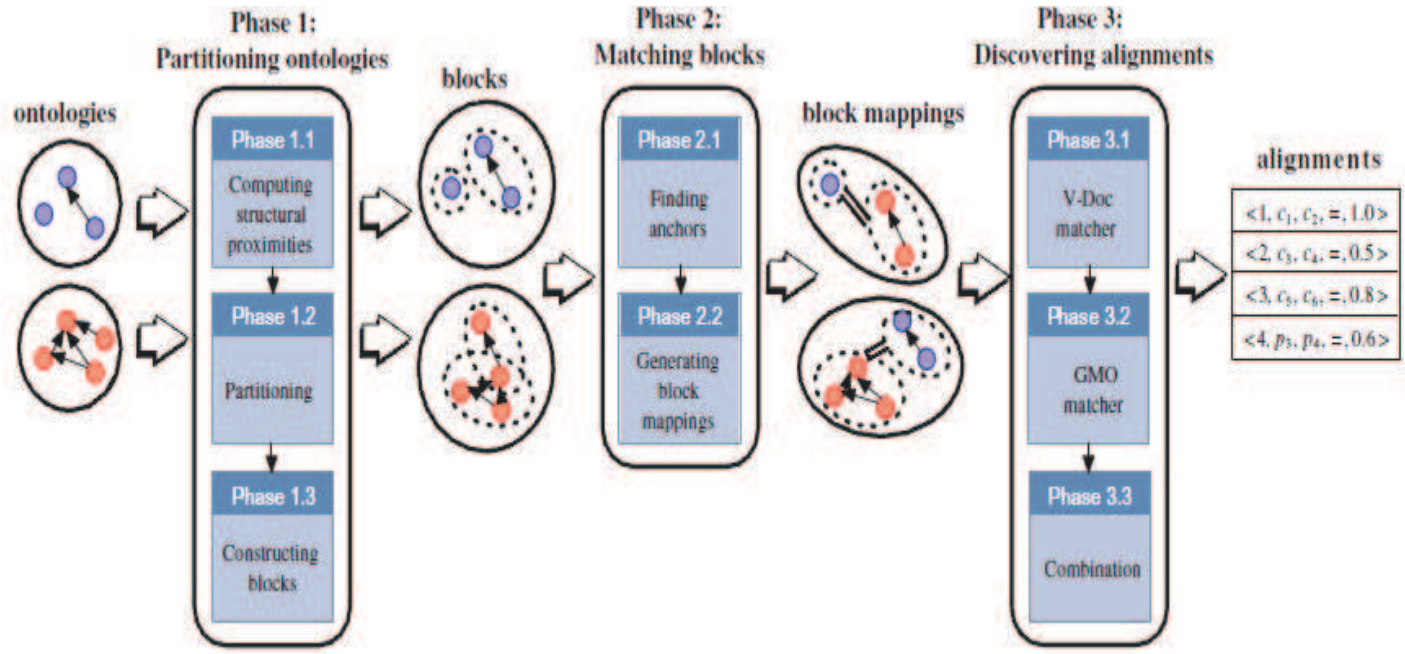

FIG 2.5 - *Architecture du système Falcon-AO[31]*

- **Calcul de la proximité lexicale :** Pour calculer la similarité lexicale entre les entités, Falcon utilise une approche basée sur les chaines de caractères (la distance d'edition et la méthode winkler) .
- **Calcul de la proximité structurelle :** La proximité structurelle entre les entées est calculée par la mesure de Wu & Palmer[67] qui prend en compte seulement la relation « est un». Pour réduire la complexité du calcul dans les ontologies larges, Falcon ne compare que les entités qui satisfont la relation suivante :

$$|depth\ (D_i) - \ depth\ (D_j)| \ \leq 1$$

- **Partitionnement :** L'objectif de cette phase est de partitionner une ontologie en clusters $g_1, g_2, \ldots \ldots \ldots g_n$. L'algorithme proposé est agglomératif et inspiré de l'algorithme de clustering de ROCK [23]. On part de petits clusters nombreux et on les regroupe progressivement en clusters plus conséquents. Ensuite les blocs sont construits en assignant des phrases RDF à ces clusters.

  L'algorithme de partitionnement s'appuie sur deux propriétés essentielles : la cohésion au sein d'un cluster et le couplage entre deux clusters distincts. La cohésion mesure la similarité entre les entités appartenant à un même cluster et le couplage mesure la similarité entre les entités appartenant à deux clusters différents. Les deux propriétés ont été calculées au sein d'une même fonction *cut* () qui mesure la distance

entre deux cluster en reposant sur un critère d'agrégation, déterminant la manière d'agglomérer deux clusters. La figure suivante montre l'algorithme de partitionnement utilisé.

Algorithm: partition(D, W, ϵ).

Input: a set of entities D, a matrix W indicating the structural proximities between the entities in D and a parameter ϵ limiting the maximum number of the entities in each cluster $(\epsilon < |D|)$.

Output: a set of clusters G.

```
01  for each entity d_i ∈ D  // Initialization
02      g_i = create(d_i);
03      G = G ⋃ {g_i};
04  for each cluster g_i ∈ G
05      cohesive(g_i) = depth(d_i);  // depth(root) = 1
06      for each cluster g_j ∈ G satisfying j ≠ i
07          coupling(g_i, g_j) = W(d_i, d_j);
08  while(true)  // Partitioning
09      g_s = arg max(cohesive(g_i));  // g_i ∈ G
10      g_t = arg max(coupling(g_s, g_j));  // g_j ∈ G, g_j ≠ g_s
11      if |g_s| + |g_t| > ε or cohesive(g_s) == 0  // Termination condition
12          return G;
13      else if coupling(g_s, g_t) == 0  // Isolated cluster
14          cohesive(g_s) = 0;
15      else  // Merging
16          g_p = g_s ⋃ g_t;
17          cohesive(g_p) = cohesive(g_s) + cohesive(g_t) + coupling(g_s, g_t);
18          for each cluster g_i ∈ G satisfying i ≠ p, s, t
19              coupling(g_p, g_i) = coupling(g_s, g_i) + coupling(g_t, g_i);
20              coupling(g_i, g_p) = coupling(g_p, g_i);
21          G = G ⋃ {g_p} \ {g_s, g_t};
```

FIG 2.6 *L'algorithme de partitionnement Falcon-AO[31]*

L'algorithme prend en entrée l'ensemble g des *n* clusters à partitionner, où chaque cluster est réduit au départ à une unique entité et le ε le nombre des entités dans un cluster que l'on souhaite obtenir. Dans chaque itération, l'algorithme choisit tout

d'abord le cluster qui a la cohésion maximale, puis le cluster qui a le couplage maimal avec ce premier cluster, et les fusionne.

- **Trouver les ancres :** Pour trouver les ancres qui représentent les entités equivalentes, Falcon utilise une approche lexicale basée sur les chaines de caractères (la distance d'edition) . Deux entités sont jugés similares si la similarité est superieure ou égale à 0.75.
- **Génération des blocs à aligner :** La similarité entre les blocs peut être calcuée en se basant sur la distribution des ancres acquis ci-dessus. Plus deux blocs contiennent d'ancres communes , plus ils sont jugés similares.

2.4.2.2 TaxoMap (2008)

Pour prendre en compte au plus tôt l'objectif d'alignement, deux méthodes ont été proposées et qui vont s'appuyer sur deux données : d'une part les couples de concepts issus des deux ontologies qui ont exactement le même label et qui peuvent être reliés par une relation d'équivalence et d'autre part, l'éventuelle dissymétrie structurelle des deux ontologies à aligner afin d'ordonner leur partitionnement. Dans ce qui suit, l'ontologie la plus structurée sera appelée la cible, O_T et la moins structurée, la source, O_S [25].

- **Méthode 1 :** La première méthode consiste à commencer par décomposer la cible O_T en utilisant l'algorithme Falcon, puis à forcer le partitionnement de O_S à suivre celui réalisé pour O_T . Pour cela, la méthode identifie pour chacun des blocs B_{Ti} construits à partir de O_T, l'ensemble des ancres lui appartenant. Chacun de ces ensembles constituera le noyau ou centre CB_{Si} d'un futur bloc B_{Si} à générer à partir de la source O_S. L'alignement des paires de blocs ainsi constituées permet de retrouver dans la phase d'alignement finale, toutes les relations d'équivalence entre les ancres.
- **Méthode 2 :** L'idée de cette méthode est de partitionner les deux ontologies en même temps, c.à.d. de faire du co-clustering. Le traitement réellement de ces ontologies en parallèle est difficile du fait de leur grande taille. Pour simuler le parallélisme, on partitionne l'ontologie cible en favorisant la fusion des blocs partageant des ancres avec la source, et on partitionne la source en favorisant la fusion des blocs partageant des ancres avec un même bloc généré pour la cible. Prendre en compte les relations d'équivalence identifiées entre les ontologies dès le partitionnement de O_T, devrait permettre par la suite de faciliter la recherche des paires de blocs les plus proches et d'améliorer les résultats de l'alignement. On peut ainsi dire que ce partitionnement,

contrairement à celui de FALCON ou à celui implémenté dans la méthode 1, est orienté alignement pour les deux ontologies à partitionner.

Dans ces méthodes, l'alignement se fait entre les blocs partageant le plus d'ancres, un bloc de O_S ne s'alignant qu'avec un seul bloc de O_T .

2.4.3 Evaluation comparative entre Falcon-AO et TaxoMap

L'evaluation comparative entre Falcon-AO et TaxoMap est basée sur les trois mesures de Précision, Rappel, et F-Mesure qui sont largement exploitées pour évaluer les méthodes d'alignement. La table suivante récapitule les resultats obtenus par les deux méthodes d'alignement lors de leurs évaluations dans la compagnie OAEI[4].

Système	Rappel				Précision			F-Mesure		
année	2005	2006	2007	2008	2006	2007	2008	2006	2007	2008
Falcon	0.31	0.45	0.61		0.41	0.55		0.43	0.58	
TaxoMap				0.34			0.59			0.43

TAB 2.3 - *Résumé d'évaluation des méthodes Falcon-AO et TaxoMap par année[45]*

2.5 Défis d'alignement d'ontologies

Nous montrons dans cette section ce que pourrait être un challenge lors du développement d'une nouvelle méthode d'alignement d'ontologies.

1. **Design patterns d'ontologies :** Chaque ontologiste, selon son domaine de prédilection va représenter un concept de manière propre et dans une hiérarchie spécifique. La création des ontologies de manière ad hoc a un risque inévitable d'avoir une hétérogénéité entre elles. Dans l'ingénierie des connaissances, et plus précisément celle des ontologies, il est nécessaire d'avoir des patterns sur lesquels les ontologies sont crées afin de réduire l'hétérogénéité.

[4] Ontology Alignment Evaluation Initiative. http://oaei.ontologymatching.org

Le design patterns d'ontologie (Ontology Design Patterns ODPs) est défini comme une solution aux problèmes récurrents de conception d'ontologies. Cette notion constitue une des voies les plus prometteuses quant à la conception et la création des ontologies. Dans [62], Svab propose d'utiliser les patterns pour obtenir des meilleurs algorithmes de matching.

Les patterns d'ontologies sont très importants pour les raisons suivants :

- Ils aident à développer des méthodes d'alignement robustes et plus rapides parce qu'ils donnent les meilleures informations sur la conventions des noms et la structure d'hiérarchie;
- Si les deux ontologies utilisent le même pattern, l'hétérogénéité va être réduite ce qui donnent des résultats d'alignement plus satisfaisants;
- Si les deux ontologie sont conçues par différents patterns, les patterns fournissent au développeurs le contexte de la nouvelle méthode d'alignement à créer, la raison pour laquelle certaines correspondances sont effectuées d'une certaine manière;
- Grâce aux patterns, on peut développer des méthodes sans avoir recours aux hiérarchies ontologiques.

2. **Ontologie Large :** l'alignement d'ontologies est par définition un processus de découverte de correspondance. Plusieurs travaux d'alignement ont été proposés comme solution à la problématique d'hétérogénéité de connaissances. La plupart de ces travaux se sont focalisés sur l'alignement des ontologies simples. Cependant, Dans les domaines d'applications réelles, les ontologies sont volumineuses et complexes. Elles entraînent des problèmes d'efficacité en temps d'exécution des algorithmes. De plus, l'alignement des ontologies larges, en utilisant les algorithmes d'alignement classiques, devient presque impossible. L'une des solutions du passage à l'échelle suppose la possibilité de partitionner les ontologies en blocs avant de réaliser l'alignement pour limiter la taille des ensembles de concepts en entrée de l'outil d'alignement afin de diminuer l'espace de recherche des correspondances[25],[31].

3. **Paramétrage du système :** Le paramétrage ou bien le tunnig est primordial dans les systèmes d'alignement d'ontologies puisqu'il est susceptible de modifier de manière significative les valeurs de similarité. Le tunnig peut être effectué manuellement par un expert humain familier au contexte. Cette tâche impose une maîtrise parfaite du système d'alignement. Cependant un tel choix du paramètre n'est pas toujours souhaitable ni possible.

   Dans la littérature actuelle, plusieurs travaux s'intéressent à automatiser le tunnig dans les sytèmes de matching. Berkovsky [5] utilise les algorithmes génétiques pour le tuning automatique. L'approche de Sayyadian [58] propose un système de tuning automatique nommé eTuner.

4. **Exploitation des ressources externes :** De nombreux travaux portent actuellement sur l'utilisation des ressources externes, dites de "background". Ces travaux montrent l'intérêt d'utiliser ces connaissances externes pour la découverte des correspondances. Les ressources peuvent être des résultats d'un autre outil d'alignement, un corpus du domaine spécifique, une ontologie du domaine spécifique, une ontologie disponible dans le web sémantique (obtenir par Swoogle[5]).

   La similarité est déduite en exploitant les liens sémantiques existant dans ces ressources externes. Les méthodes utilisant une ressource lexicale suivent, en général, le processus ci-dessous [32] :

   1. Pour chacun des termes t_1 et t_2 à comparer, trouver les ensembles d'entités N_{t1} et N_{t2} de la ressource auxquels ils correspondent;
   2. Pour chaque couple $(c_i, c_j) \in N_{t1} \times N_{t2}$, estimer la relation qu'entretiennent c_i et c_j;
   3. Déduire la relation entretenue par t_1 et t_2 à partir des relations estimées des couples (c_i, c_j).

   Cependant, le problème à identifier au travers ce processus concerne le traitement des relations présentées dans ces ressources externes :

   - Comment tirer l'ensemble d'entités N_{t1} et N_{t2} de toute la richesse représentée dans ces ressources ?
   - Comment estimer la relation qu'entretiennent c_i et c_j ?
   - Comment interpréter les relations estimées des couples (c_i, c_j) pour déduire la relation entretenue par t_1 et t_2 ?

[5] Moteur de recherche des ontologies annotées.http//www.swoogle.com

5. **Visualisation :** Afin de réduire le travail d'un expert dans un système d'alignement d'ontologies pour la visualisation des correspondances et l'interaction face aux informations cachées derrière les noms des concepts, il est nécessaire d'avoir des outils qui permettent la visualisation de correspondances en termes graphiques, en opposition à une liste des correspondances moins significative.

6. **Performance du système :** Le temps d'exécution et les consommations de l'espace mémoire influencent directement la performance d'un algorithme d'alignement, surtout lorsqu'on aligne des ontologies volumineuses. Les tâches pour lesquelles ces facteurs sont fondamentaux concernent : le calcul de similarité, le partitionnement, la visualisation, l'accès, l'exploitation des ressources externes…etc. La table 2.4 présente le temps d'exécution de quelques systèmes d'alignement d'ontologie (PBM, DSSIM, RIMOM, PRIOR+ et AOAS) sur les deux paires de test (Anatomy et Food) de la compagnie OAEI.

	PBM	DSSIM	RIMOM	PRIOR+	AOAS
Anatomy	12 mn	4 h	75 mn	23 mn	2 h
Food	6 h	1 semaine	4 h	1.5 h	-

TAB 2.4 - *Temps d'exécution de PBM, DSSIM, RIMOM, PRIOR+ et AOAS [31].*

Il est important de noter que l'algorithme le plus performant est celui qui prend mois du temps et qui donne les meilleurs résultats.

Conclusion

Dans ce chapitre, nous avons présenté les dimensions internes et externes des méthodes d'alignement dans le domaine du web sémantique sur lesquelles une comparaison doit s'appuyer pour les classifier. Puis, nous avons détaillé deux méthodes d'alignement, Falcon-AO et TaxoMap. Ces méthodes sont conçues pour aligner des ontologies larges. Finalement, nous avons présenté plusieurs défis qui confrontent la création d'une nouvelle méthode d'alignement. Pour contribuer à relever le défi des ontologies larges, nous allons présenter notre méthode d'alignement dans le chapitre suivant.

Méthode d'alignement proposée 3

Sommaire

Introduction

Dans la littérature actuelle, il existe très peu d'algorithmes d'alignement qui visent à traiter le problème du passage à l'échelle des méthodes d'alignement. Pour relever le challenge du passage à l'échelle, nous proposons une méthode d'alignement d'ontologies avec partitionnement. L'algorithme de partitionnement proposé, inspiré de Falco-AO, consiste à partitionner chaque ontologie en blocs autour des ancres en utilisant les algorithmes de clustering qui fonctionnent itérativement en affectant à chaque étape une entité au cluster d'ancre le plus proche puis l'alignement s'effectue entre les blocs les plus similaires car les blocs qui contiennent les entités similaires, c.-à-d. les ancres, devraient l'être aussi. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la non perte des alignements et fournit en sortie des paires de blocs à aligner.

Cet algorithme est intégré dans notre méthode d'alignement structurelle qui exploite l'intérêt de design pattern d'ontologie. La syntaxe OWL est ainsi utilisée à cet effet. Le présent chapitre présente en détail notre proposition.

3.1 Processus général d'alignement

Avec l'apparition des nouveaux standards, outils et langages très expressifs, de grandes ontologies ont été développées dans plusieurs domaines (e.g biologie, géographie, médecine …etc.). Ces ontologies comportent plusieurs dizaines de milliers de concepts par exemples, l'ontologie Gene Ontology GO[4] et l'ontologie AGROVOC[5] contiennent 26 057 et 28 439 concepts respectivement. Dans la littérature, on rencontre plusieurs algorithmes ([25], [31], [48], [61]) qui visent le partitionnement des ontologies de façon à faciliter et à rendre plus performantes les opérations de maintenance, de visualisation, de raisonnement ou d'alignement.

Ainsi, les travaux de [48] proposent deux méthodes de décomposition d'ontologie en plusieurs sous-ontologies exprimées en logique de description. Cette décomposition est appliquée telle qu'elle préserve toujours la sémantique et les services d'inférence de l'ontologie originale. La première méthode est basée sur le séparateur minimal en utilisant l'algorithme récursive de Even [19] et la deuxième est basée sur la segmentation d'images en utilisant les vecteurs et les valeurs propres [34]. Les travaux dans [61] partitionnent une ontologie en blocs indépendants et cohérents à partir d'un graphe de dépendance. Le but de cette méthode est de faciliter les différentes opérations sur les ontologies (maintenance, validation et raisonnement).

Dans le cadre de notre travail, nous nous intéressons aux méthodes qui ont pour objectif l'alignement d'ontologies larges. Plus particulièrement, Falcon-AO et TaxoMap, que nous avons présenté dans le chapitre précédent.

Dans l'étape d'alignement, Falcon-AO aligne toutes les paires ayant une proximité supérieure à un seuil. Cette proximité s'effectue en s'appuyant sur des ancres. Soit b et b′ deux blocs, la proximité entre b et b′ est calculée comme suit :

$$prox(b, b') = \frac{2 \,.\, nombre\ d'ancres\ partagées\ par\ b\ et\ b'}{nombre\ d'ancres\ contenues\ par\ b + nombre\ d'ancres\ contenues\ par\ b'} . \qquad (3.1)$$

La proximité entre deux blocs est liée au nombre d'ancres partagées. Si le nombre d'ancres partagées est petit, la proximité sera inférieure au seuil et la paire n'est pas alignée c.-à-d. l'alignement de deux ancres partagées n'est pas retrouvé. D'autre part, plusieurs blocs pourront ne contenir aucune ancre. Dans ce cas ces blocs sont isolés et on a risque de perdre

[4] The Gene Ontology, http://www.geneontology.org

[5] Une ontologie construite par le FAO (Food and Agriculture Organization). Agricultural Information Management Standards, http://www.fao.org/aims/

plusieurs alignements. Pour TaxoMap, le problème de l'alignement des ancres partagées est résolu mais le problème de perdu des alignements provenant des blocs isolés existe toujours.

Pour contribuer à résoudre ces deux problèmes, nous proposons une méthode d'alignement avec partitionnement autour des ancres [37]. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la non perte des alignements et fournit en sortie des paires de blocs à ligner. Chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie. Ainsi notre processus d'alignement comporte trois phases comme montré en FIG 3.1.

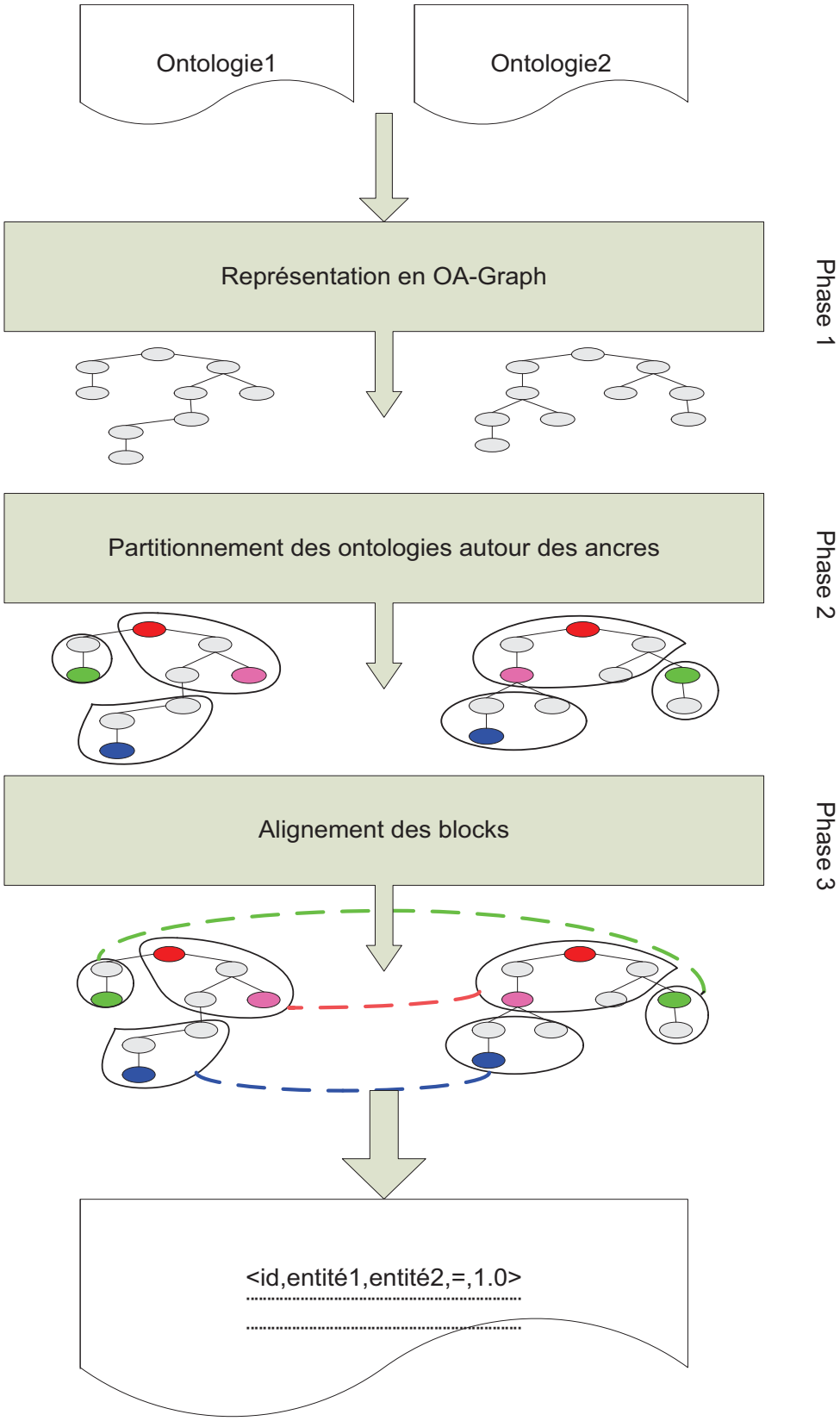

FIG. 3.1 - Processus d'alignement [37]

- **Représentation en OA-Graph** : les deux ontologies à aligner sont représentées en OA-Graph. Cela réduit beaucoup le temps de calcul nécessaire chaque fois que l'algorithme devra chercher un concept dans les fichiers (OWL,RDF).
- **Partitionnement des ontologies** : Dans la deuxième phase, les deux ontologies sont partitionnées autour des ancres car les blocs qui contiennent les entités similaires devraient l'être aussi.
- **Alignement des ontologies** : Dans la phase d'alignement, chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie.

Dans la suite, nous présentons en détail chacune de ces trois étapes.

3.2 Représentation des ontologies en graphe

Une ontologie OWL peut être vue comme un réseau sémantique de classes, de propriétés et d'individus qui sont liés entre eux par des liens sémantiques grâce aux constructeurs (primitives) comme *rdfs:subClassOf, rdfs:subPropertyOf, owl:equivalentClass, owl:equivalentProperty rdfs:domain, rdfs:range.*

Dans cette phase, les ontologies OWL à aligner sont représentées sous forme d'un graphe étiqueté OA-Graph ou les entités (classe nommée, propriété) sont les nœuds dans le graphe et les liens sémantiques (les constructeurs de OWL) sont les arcs (voir la FIG 3.2).

Dans OA-Graph nous nous intéressons à représenter les informations qui facilitent le partitionnement de l'ontologie et la tâche d'alignement. Celle–ci se base sur les descriptions des concepts et les structures des propriétés pour mesurer la similarité entre les entités.

- **Les nœuds :** Pour déduire la similarité entre deux entité, notre méthode est basée sur leurs structures internes en utilisant la mesure de Tversky. Pour cela, Nous nous intéressons plus précisément à la représentation des classes nommées et celle des propriétés.
- **Les arcs :** Dans la phase de partitionnement, nous allons chercher un calcul de proximité à moindre coût. Pour cela, nous utilisons la similarité de Wu & Palmer [67] qui se base sur les arcs et prend en compte la structure de la hiérarchie. Cependant cette mesure n'admet que la distance sémantique entre deux entités reliées par la relation " est un". Celle-ci est présenté par les constructeurs *rdfs:subClassOf, rdfs:subPropertyOf* de OWL. Donc le premier type d'arcs dans OA-Graph est le type "est un".

Les constructeurs *rdfs:domain* et *rdfs:range* sont représentés par des arcs de type "domaine ou rang " qui lient la propriété en question avec son domaine et son co-domaine. Cependant, la plupart des informations à propos des propriétés s'expriment plus naturellement dans des

restrictions. Le langage OWL distingue deux types de restrictions qui permettent de définir localement l'image et la cardinalité de la propriété :

1. celles contraignant sa valeur ;
2. celles contraignant sa cardinalité.

Dans OA-Graph**,** Nous représentons seulement le premier type de restriction par des arcs de type "rang".

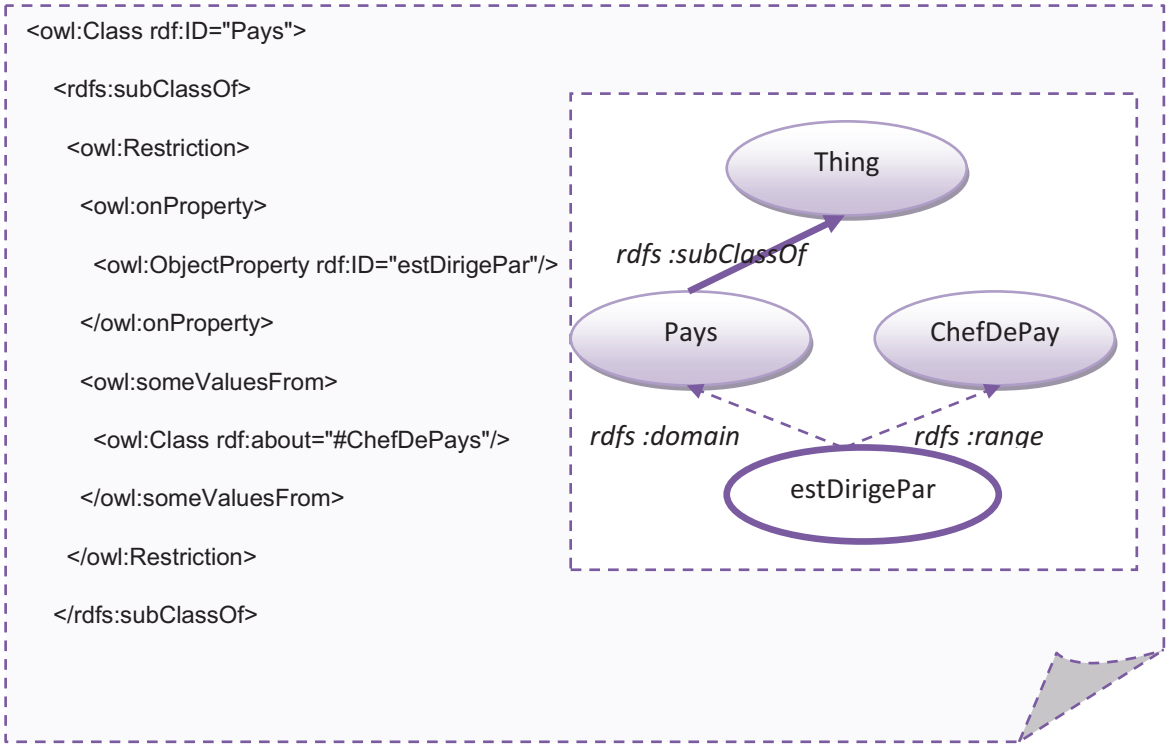

FIG. 3.2 - *Représentation en OA-Graph*

3.3 Algorithme de partitionnement proposé

Dans une ontologie, les entités sont décrites en utilisant les noms, les étiquètes, et les commentaires. Les ancres sont des entités jugées comme équivalentes en se basant sur la mesure de jaro-winkler[66]. L'algorithme que nous proposons consiste à partitionner chaque ontologie en blocs autour des ancres, puis l'alignement s'effectue entre les paires des blocs fournies en sortie.

Dans ce qui suit, nous présentons les définitions des mesures de similarité utilisées, puis nous détaillons notre algorithme de partitionnement.

3.3.1 Présentation des mesures de similarité utilisées

nous allons maintenant décrire les mesures de similarité utilisés dans notre algorithme de partitionnement.

1. **La similarité lexicale :** Pour calculer la similarité lexicale entre les entités nous utilisons la mesure de Jaro-Winkler [66] qui prend simultanément en compte le nombre et la position des sous chaînes communes avec l'utilisation de la taille p du plus grand préfixe commun de deux chaines comparées. En OWL, les entités sont décrites en utilisant les constructeurs (rdf :id, rdfs : label, rdfs : comment) qui traduisent le triplet (nom, étiquète, commentaire) respectivement. Dans notre cas nous utilisons seulement les noms des entités (classe, propriété) pour calculer la similarité lexicale entre les entités, sans prétraitement, dans une mesure à moindre coût.

$$Sim_L(e_1, e_2) = 1\text{-}DS_{Jaro-Winkler}. \quad (3.2)$$

2. **La similarité structurelle :** Wu & Palmer [67] ont proposé de calculer la similarité entre deux concepts par la formule suivante :

$$Sim_S(e_1, e_2) = \frac{2*depth(e)}{depth(e_1)+depth(e_2)}. \quad (3.3)$$

e: le concept le plus spécifique qui subsume les deux concepts e_1, e_2 dans l'ontologie.

$depth(e)$: le nombre d'arcs qui séparent le concept e de la racine.

$depth(e_i)$: le nombre d'arcs qui séparent le concept e_i de la racine en passant par e .

La mesure de Wu & Palmer est intéressante, simple à implémenter et performante dans les évaluations. Cependant, pour une ontologie de grande taille, le calcul de similarité entre toutes les entités peut prendre beaucoup de temps. Pour cela, nous proposons d'effectuer le calcul de similarité seulement entre les entités qui se situent à un rayon r (r étant le nombre d'arcs entre les deux entités) dans un procédé itératif où r est défini selon la taille et la structure de l'ontologie.

3.3.2 Présentation de l'algorithme de partitionnement

Après avoir passé en revue les mesures de similarité, nous allons maintenant détailler notre algorithme de partitionnement.

Soient O_t et O_s deux ontologies à aligner, notre algorithme de partitionnement orienté alignement comprend les étapes suivantes [36] :

1. Déterminer les couples d'ancres entre O_t et O_s en utilisant une mesure de similarité lexicale.
2. Classifier ces couples d'ancres en k clusters préliminaires.
3. Identifier les clusters d'ancres de chaque ontologie (l'ensemble des ancres appartenant à chaque ontologie) .
4. Classifier les concepts de chaque ontologie autour des clusters d'ancres.

Dans ce que suit, nous détaillons les différentes étapes de partitionnement.

3.3.2.1 Etape 1 : Déterminer les ancres

Dans l'étape de classification, l'algorithme tentera de regrouper les entités autour des ancres. Nous disons que deux entités sont des ancres si et seulement si elles sont équivalentes. A cet effet, nous utilisons la similarité lexicale présentée ci-dessus (la formule 3.2) pour les déterminer.

3.3.2.2 Etape 2 : Partitionnement préliminaire

Le but de ce partitionnement est de regrouper les ancres dans K clusters. Après la détermination des ancres par la similarité lexicale, on regroupe chaque couple d'ancres dans un cluster comme montré en Fig. 1. Puis il s'agit d'exécuter un algorithme de clustering inspiré de celui de Falcon-AO[31].

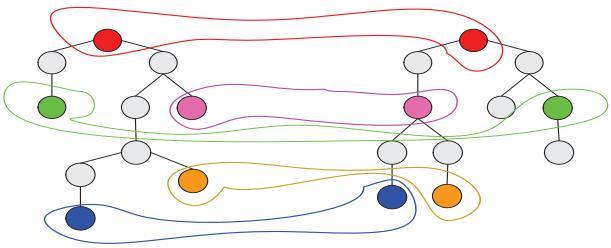

FIG. 3.3 - *L'entrée de l'algorithme (les couples d'ancres)*

L'algorithme. La classification consiste à agréger progressivement les clusters d'ancres selon leur ressemblance. Cette agrégation nécessite un critère de classification. L'algorithme est réalisé grâce à l'utilisation d'un algorithme de partitionnement agglomératif hiérarchique inspiré de l'algorithme de Falcon-AO. La première différence entre notre algorithme et celui de Falcon-AO est que ce dernier prend en entrée toutes les entités de l'ontologie à aligner et le nombre k des blocs que l'on souhaite obtenir en sortie par contre notre algorithme prend en entrée les clusters initialisés par les paires d'ancres, le nombre de cluster k et le nombre maximum d'entités dans un cluster fusionnable m. Le but du partitionnement est de diminuer

le nombre d'entités dans un bloc, car un système d'alignement n'est plus du tout efficace à partir d'un certain nombre d'entités.

La deuxième différence est que la méthode Falcon-AO utilise la fonction cut() comme un critère de classification. Celle-ci est définie comme suit :

$$cut(g_i, g_j) = \frac{\sum_{d_i \in g_i} \sum_{d_j \in g_j} W(d_i, d_j)}{|g_i| \cdot |g_j|}. \quad (3.4)$$

$$\text{avec} \quad \begin{cases} cut\,(g_i, g_i) = cohésion(g_i) \\ cut\,(g_i, g_j) = couplage\,(g_i, g_j) \quad avec\ \ g_i \neq g_j \end{cases}.$$

Où g_i, g_jsont deux clusters et *W* est la matrice de la proximité (lexicale/structurelle) entre les entités. Pour calculer l'élément (i, j) dans W, Falcon-AO mesure le lien pondéré entre deux entités, $link(e_i, e_j)$ comme suit :

$$link(e_i, e_j) = \begin{cases} prox(e_i, e_j)\ \ si\ \ prox(e_i, e_j) > \varepsilon \\ 0, sinon \end{cases} . \quad (3.5)$$

$$avec\ \ prox(e_i, e_j) = \alpha\ prox_S(e_i, e_j) + (1 - \alpha)\ sim_L(e_i, e_j). \quad (3.6)$$

Où ε est un seuil donné tel que ε ϵ [0, 1] et α ϵ [0, 1]. Il permet à l'utilisateur de faire varier le poids relatif des mesures de similarité.

Dans notre cas, le calcul de similarité lexicale ne s'effectue que pour obtenir la cohésion au sein d'un cluster pour les raisons suivantes [36] :

- le rapprochement de deux clusters se fait sur la base de similarité de leurs entités. Généralement, ces entités sont décrites différemment au sein d'une même ontologie. Pour cela, la similarité structurelle sera suffisante ;
- réduire la complexité du calcul ;
- obtenir un meilleur temps d'exécution.

Donc si $cut\,(g_i, g_j) = couplage\,(g_i, g_j)$, le calcul de lien pondéré s'effectue comme suit :

$$prox(e_i, e_j) = Sim_S(e_i, e_j). \quad (3.7)$$

Et si $cut\,(g_i, g_j) = cohésion(g_i)$, le calcul de lien pondéré s'effectue comme suit :

$$prox(e_i, e_j) = \alpha\ Sim_S(e_1, e_2) + \beta\ Sim_L(e_1, e_2). \quad (3.8)$$

L'algorithme prend en entrée l'ensemble S de n clusters à partitionner, ou chaque cluster est réduit au départ à une paire d'ancres, le nombre k de clusters que l'on souhaite obtenir en sortie, et le nombre maximum m d'entités au sein d'un cluster fusionnable.

Comme Falcon-AO, dans chaque itération, l'algorithme choisit tout d'abord le cluster qui a la cohésion maximale (ordonner les clusters par leurs cohésions), puis le cluster qui a la valeur de couplage maximale. Si le nombre d'entités dans un cluster est égale au nombre maximum m, sa cohésion est réinitialisée par la valeur zéro, et les clusters sont réordonnés. Le pseudo-code de cet algorithme est présenté ci-dessous :

```
Algorithme(S,k,m)
Initialisercohesion(S);//initialiser chaque Si  dans S  par sa valeur de cohésion.
Ordonner(S);//ordonner les Si  selon leurs cohésions .
Initialisercouplage(S);//dans une structure de type matrice
Tant que le nombre courant de cluster l > k  faire
  Prendre Si ;//le premier cluster dans S
  Choisir Sj ;//le cluster  qui a  la valeur de couplage maximale avec Si et |Si| + |Sj| <=m*2
  Fusionner les deux cluster Si , Sj dans St;
  Supprimer Si et Sj ;
  Si |St| < m*2 faire// cluster des paires d'ancres
   Mettre à jour la cohésion de St ,
  Sinon
   Mettre la cohésion de St à zéro ,
  Finsi
  Ordonner(S),
  Mettre à jour la matrice de couplage,
  l:=l-1,
Fin tant que
```

L'algorithme fournit en sortie un ensemble de k clusters d'ancres comme le montre la FIG. 3.4.

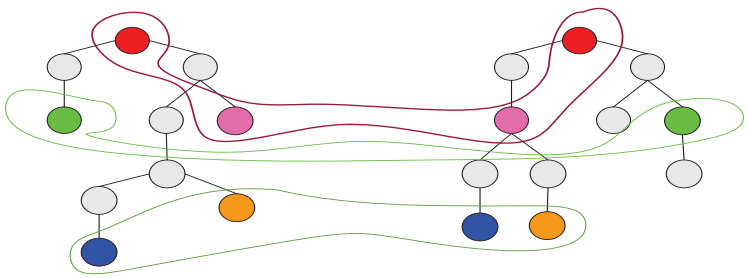

FIG. 3.4 - *La sortie de l'algorithme de partitionnement préliminaire (k clusters d'ancres)*

3.3.2.3 Etape 3 : Identifier les clusters d'ancres de chaque ontologie

Après la classification des ancres en K clusters d'ancres, nous partons de ces clusters et nous les divisons (voir FIG. 3.5) en des clusters qui ne contiennent que les ancres d'une même ontologie.

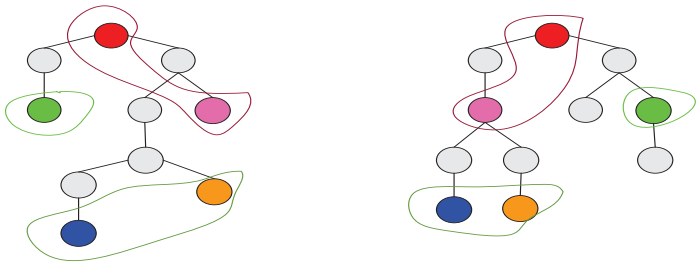

FIG. 3.5 - *Les clusters d'ancres de chaque ontologie.*

3.3.2.4 Etape 3 : Partitionnement autour des ancres

Dans cette étape, les concepts sont classifiés progressivement autour des ancres selon l'algorithme suivant :

```
Algorithme(S,m)
Initialisercouplage(S);// initialiser chaque cluster Si  dans S  par sa valeur de couplage avec
                       //toutes les  entités d'ontologie  dans une structure de type matrice.
Pour chaque entité e ∈ O faire// O est l'ensemble des entités d'une ontologie
  Choisir Si ;//le cluster  qui a  la valeur de couplage maximale avec e  et | Si| + 1<=m
  Affecter e au  cluster Si;
  Mettre à jour la le couplage de Si ,
Fin pour
```

L'algorithme fournit en sortie un ensemble de blocs où chaque bloc contient une ou plusieurs ancres comme présenté en FIG. 3.6.

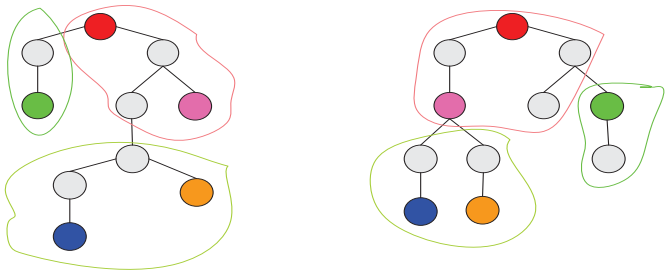

FIG. 3.6 - *Les blocs générés après la classification*

Dans ce qui suit, nous présentons notre méthode d'alignement.

3.4 Algorithme d'alignement proposé

Dans notre méthode, nous avons adopté le deuxième modèle présenté dans la section 2.2.4 pour déduire la similarité entre les entités de deux ontologies [37]. Pour cela, nous utilisons la syntaxe abstraite de OWL[6] comme un pattern général pour décrire les classes et les propriétés. Il s'agit d'une notation BNF. Les symboles terminaux sont représentés en italique, alors que les non-terminaux sont représentés en caractère gras. Les alternatives sont séparées par des barres (|) . Les éléments qui peuvent apparaître un nombre illimité de fois (incluant zéro) sont englobés par des accolades ({: : :}). Pour des raisons de clarté, nous notons que dans la présentation qui suit, plusieurs simplifications ont été apportées à la définition du langage. Le pattern OWL utilisé est :

[6] http://www.w3.org/TR/owl-semantics/syntax.html.

Entity_Class ::= ' *Class* (' ***classID { descriptrion } { axiom} { annotation }') '***

description ::= ***classID***

|***restriction***

| '*unionOf*(' { ***description*** } ')'

| '*intersectionOf*(' { ***description*** } ')'

| '*complementOf*(' ***description*** ')'

| '*oneOf*(' { individualID } ')'

axiome::= '*EquivalentClasses*(' ***description*** { ***description*** } ')'

| '*SubClassOf*(' ***description description*** ')'

restriction ::= '*restriction*(' ***dataPropertyID dataRestriction*** {***dataRestriction*** } ')'

| '*restriction*(' ***individualPropertyID individualRestriction*** { ***individualRestriction)'***

dataRestriction ::= '*allValuesFrom*(' ***dataRange*** ')'

| '*someValuesFrom*(' ***dataRange*** ')'

| '*value*(' ***dataLiteral*** ')'

| ***cardinality***

dataRange ::=datatypeID | *'rdf :Literal ' | oneOf(' { **dataLiteral** } ')'*

cardinality ::= ' *minCardinality**(' non-negative-integer ')'***

|*maxCardinality**(' non-negative-integer ')'***

|*cardinality**(' non-negative-integer ')'***

individualRestriction ::= '*allValuesFrom*(' ***description*** ')'

| '*someValuesFrom*(' ***description*** ')'

| '*value*(' ***individualID*** ')'

| ***cardinality***

Entity_Property *::= 'DatatypeProperty ('* ***PropertyID*** *{* ***annotation*** *}*

*{ 'domain(' **description** ')' } { 'range(' **dataRange** ')' } {**axiome** } ')'*

*| 'ObjectProperty(' **PropertyID { annotation** }*

*{ 'domain(' **description** ')' } { 'range(' **description** ')' } {**axiome** } ')'*

axiome ::= '*EquivalentProperties(*' ***PropertyID PropertyID*** { ***PropertyID*** } ')'

| '*SubPropertyOf(*' ***PropertyID PropertyID*** ')'

La similarité entre les entités est déduite en comparant les différentes descriptions d'ontologies [37]. Nous utilisons la mesure de Tversky (voir la formule 1.11) pour calculer la similarité entre les entités et leurs composants, en définissant chaque fois les fonctions qui présentent les parties communes et distinctives : plus les entités partagent des parties, moins

elles ont des parties distinctives, et plus elles sont similaires. Nous commençons tout d'abord par la similarité des classes.

3.4.1 La similarité entre les classes

Dans une entité classe, nous distinguons quatre catégories de composants: *classID, description, axiome, annotation.* Soit $C_1 \cap C_2 = \{$ *classID,* $description^*$, $axiome^*$, $annotation^*$ $\}$.

La fonction de l'intersection peut être formalisée de la manière suivante :

$$f(C_1 \cap C_2) = w_{classID}\, Sim_{classID}(c_1, c_2) + w_{descriptions}\, Sim_{descriptions}(c_1, c_2) + w_{axiomes}\, Sim_{axiomes}(c_1, c_2) + w_{annotations}\, Sim_{annotations}(c_1, c_2). \quad (3.9)$$

la différence entre deux ensembles E et F est définie par :

$$E - F = E - (E \cap F) \text{ et que } E \cap F = F \cap E$$

Et nous aurons :

$$f(C_1 - C_2) = |C_1| - f(C_1 \cap C_2). \quad (3.10)$$

$$f(C_2 - C_1) = |C_2| - f(C_1 \cap C_2).$$

Les valeurs des poids *w* sont choisies de manière que la somme des poids est égale $|C_1|$ ou $|C_2|$ selon le cas.

Nous détaillons maintenant la similarité de chaque composant.

3.4.1.1 La similarité des identificateurs

En OWL, le classID décrit une entité classe au moyen d'un nom de classe qui prend la forme syntaxique d'un appel d'adresse *URI*. Ce dernier se compose de deux parties : l'espace de nom et le nom local. Deux entités classe sont jugées similaires si elles sont référencées par une même *URI* . Cependant, les ontologies à aligner ont généralement différents espaces de noms, il est intéressant donc de ne comparer que les noms locaux des entités classe.

Le classID peut être un terme ou une combinaison de termes(des expressions) et il est unique dans une ontologie pour identifier une entité classe. Afin de préparer les classID à la similarité choisie, une étape de normalisation est effectuée. La normalisation consiste à segmenter les expressions à un ensemble de termes, à éliminer les mots d'outils (articles, prépositions, mots grammaticaux ….etc.).

Soit $C_{1_{classID}}$={ $term_{1classID_1}$, $term_{2classID_1}$, $term_{3classID_1}$ …., $term_{nclassID_1}$ } l'ensemble des termes de classID de l'entité classe C_1.

Soit $C_{2_{classID}}$={ $term_{1classID_2}$, $term_{2classID_2}$, $term_{3classID_2}$ …., $term_{mclassID_2}$ } l'ensemble des termes de classID de l'entité classe C_2.

La fonction de l'intersection peut être formalisée de la manière suivante :

$$f(C_{1_{classID}} \cap C_{2_{classID}}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} Sim(term_{iclassID_1}, term_{jclassID_2})}{|C_{1_{classID}}| \times |C_{2_{classID}}|} \quad (3.11)$$

Où $Sim(term_{iclassID_1}, term_{jclassID_2}) = 1\text{-}DS_{Jaro-Winkler}$.

Les différences peuvent être formalisées de la manière suivante :

$$f(C_{2_{classID}} - C_{1_{classID}}) = \begin{cases} 0 \ si \ Sim(term_{iclassID_1}, term_{jclassID_2}) = 1 \ et \ i = j \ et \ n = m \\ 1 - f(C_{1_{classID}} \cap C_{2_{classID}}) \ sinon \end{cases} \quad (3.12)$$

$$f(C_{2_{classID}} - C_{1_{classID}}) = f(C_{1_{classID}} - C_{2_{classID}})$$

Si $C_{1_{classID}}$ et $C_{2_{classID}}$ ne contiennent qu'un seul terme, la similarité $Sim_{classID}(c_1, c_2)$ se calcule directement en utilisant la distance de Jaro-Winkler.

Remarque 3.1 : Le calcul de la similarité de deux chaines de caractères est effectué en deux étapes : la normalisation, et le calcul de similarité. Dans la première étape, nous convertissons la chaine de caractères en un ensemble de tokens par un algorithme de tokenization et ensuite nous utilisons un algorithme de *stemmer* configuré à la langue des deux ontologies à aligner pour trouver les lemmes. Dans la deuxième étape nous utilisons une mesure de similarité pour comparer entre les deux ensembles de lemmes.

3.4.1.2 La similarité des descriptions

Dans une définition de classe, OWL-DL permet d'utiliser des descriptions complexes contenant des identificateurs et des restrictions de classes. Elles peuvent aussi être des combinaisons booléennes des autres restrictions, et des ensembles d'individus. Les descriptions de classe peuvent apparaître un nombre illimité de fois dans une définition de classe.

Soit $C_{description}$= *classID / restriction / unionO / intesectionOf / complementOf / oneOf* .

La similarité de chaque type de description est calculée comme suit :

1. **la restriction de propriété** : Soit $C_{restriction}$=dataProperty / individualProperty . La similarité entre deux restrictions de même type est calculée comme suit :

$$sim(restiction_{descriptions_1}, restriction_{descriptions_2}) = w_{PropertID}\ sim(propertyID_1, propertyID_2) + w_{dataRestriction}\ \ sim(propertyRestriction_1, propertyRestriction_2). \quad (3.13)$$

Le langage OWL distingue deux types de restrictions de propriété, celles contraignant sa valeur et celles contraignant sa cardinalité. La similarité est calculée entre les restrictions ayant le même prédicat.

(a) *propertyRestriction*∈ { *allValuesFrom*, *someValuesFrom*, *value* } : si la propriété est de type *individualProperty ,* la similarité de deux *propertyRestriction* est déduite en comparant leurs descriptions. Dans le cas de *dataProperty,* la similarité est déduite en comparants leurs *dataRanges*, considérées comme des chaines de caractères. Si les deux chaines sont exactement égales l'une à l'autre, la similarité est égale à 1 sinon elle vaut 0.

(b) *propertyRestriction*∈ { *maxCardinality*, *minCardinality*, *cardinality* } : la similarité de deux *propertyRestriction* est déduite en comparant leurs valeurs de cardinalité. La similarité est égale à 1 si les deux valeurs sont exactement égales l'une à l'autre. Sinon 0. Si les restrictions sont modélisées différemment, la similarité est :

$$\begin{cases} 1 \text{ Si } \text{Valeur}_{\text{minCradinality}} < \text{Valeur}_{\text{cradinality}} < \text{Valeur}_{\text{maxCradinality}} \\ 0 \text{ sinon} \end{cases} . \quad (3.14)$$

2. **l'intersection ou l'union de deux descriptions de classe ou plus** : Les deux types de description peuvent contenir plusieurs descriptions. Ainsi la similarité est calculée par extension de la similarité des descriptions.
3. **le complément d'une description de classe** : Une classe peut être reliée à une seule description de classe au moyen de la propriété complementOf . La similarité est calculée selon le type de description. Si les deux types de descriptions sont différents , la similarité est égale à zéro.
4. **l'énumération :** Pour calculer la similarité de deux descriptions de ce type, nous commençons par l'extraction des classes d'origine de la liste des individus, ensuite la similarité est calculée par l'utilisation de la mesure de Tversky.

   Soit $C_{classID1}$ ={ $classID_1$, $classID_2$, $classID_3$,….. $classID_n$ } et $C_{classID2}$ ={ $classID_1$, $classID_2$, $classID_3$,….. $classID_m$ } les deux ensemble des classes d'origine de la liste des individus de deux descriptions à comparer.

$$f(C_{classID1} \cap C_{classID2}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} Sim(classID_1, classID_2)}{|C_{classID1}| \times |C_{classID2}|} \quad (3.15)$$

$$f(\mathrm{C}_{\mathrm{classID1}} - \mathrm{C}_{\mathrm{classID2}}) = \begin{cases} 0 \ si \ Sim(classID_1, classID_2) = 1 \ et \ i = j \ et \ n = m \\ 1 - f(\mathrm{C}_{\mathrm{classID1}} \cap \mathrm{C}_{\mathrm{classID2}}) \ sinon \end{cases} \quad (3.16)$$

$$f(\mathrm{C}_{\mathrm{classID1}} - \mathrm{C}_{\mathrm{classID2}}) = f(\mathrm{C}_{\mathrm{classID2}} - \mathrm{C}_{\mathrm{classID1}})$$

la fonction d'intersection correspond à la somme pondérée des similarités calculées selon les types de descriptions.

$$f(C_{1_{descriptions}} \cap C_{2_{descriptions}}) = \sum_i^6 w_i sim(description_{i_1}, description_{i_2}) \quad (3.17)$$

Où $description_i$ est une description de type i et w_i est le poids associé à la valeur de similarité de même type (i), si l'un des types est absent dans l'équation, son poids est remis à zéro.

Les différences peuvent être formalisées de la manière suivante :

$$f(C_{1_{descriptions}} - C_{2_{descriptions}}) = |C_{1_{descriptions}}| - f(C_{1_{descriptions}} \cap C_{2_{descriptions}}) \quad (3.18)$$

$$f(C_{2_{descriptions}} - C_{1_{descriptions}}) = |C_{2_{descriptions}}| - f(C_{1_{descriptions}} \cap C_{2_{descriptions}})$$

Remarque 3.2 : Les descriptions de classe de type (restriction, unionOf, intesectionOf) conduisent à des descriptions de classe imbriquées qui peuvent donc théoriquement produire des descriptions de classe de complexité arbitraire. En pratique, le niveau d'imbrication est habituellement limité.

Remarque 3.3 : Le langage OWL n'impose pas de contraintes sur l'utilisations de composants de même type. Si C_x est un ensemble des composants de même type (x∈ {$descriptions_i$, $axiomes_i$, $annotations_i$} la similarité entre C_{x1}, C_{x2} est la valeur de similarité la plus élevée de l'une des paires de composants trouvées.

3.4.1.3 La similarité des axiomes

Une classe peut contenir plusieurs axiomes d'équivalence ou de spécification. Un axiome peut contenir une description et peut devenir plus complexe lorsque les descriptions de classes s'imbriquent. Ainsi la similarité d'un axiome est calculée par extension de la similarité des descriptions.

La similarité des axiomes de deux classes, est calculée par la mesure de Tversky. Dans ce cas, la fonction d'intersection correspond à la somme pondérées des similarités calculées selon le type d'axiome.

$$f(C_{1_{axiomes}} \cap C_{2_{axiomes}}) = \sum_i^2 w_i sim(axiome_{i_1}, axiome_{i_2}) \quad (3.19)$$

Où $axiome_i$ est un axiome de type i et $\boldsymbol{w_i}$ est le poids associé à la valeur de similarité de même type (i), si l'un des types est absent dans l'équation son poids est remis à zéro.

Les différences peuvent être formalisée de la manière suivante :

$$f(C_{1_{axiomes}} - C_{2_{axiomes}}) = |C_{1_{axiomes}}| - f(C_{1_{axiomes}} \cap C_{2_{axiomes}}) \quad (3.20)$$

$$f(C_{2_{axiomes}} - C_{1_{axiomes}}) = |C_{2_{axiomes}}| - f(C_{1_{axiomes}} \cap C_{2_{axiomes}})$$

3.4.1.4 La similarité des annotations

Une classe peut avoir un nombre quelconque d'annotations de type label ou commentaire. Les labels sont agrégés (de même langue) avant de les comparer en utilisant la similarité classID. Comme les labels, la similarité entre les commentaires est calculés en se basant sur le modèle vectoriel [57]. Dans ce dernier un commentaire est représenté par un vecteur de poids. Chaque poids dans le vecteur désigne l'importance d'un terme correspondant dans ce commentaire.

On considère l'espace vectoriel : $U = < t_1, t_2, t_3, \dots t_i >$ où les t_i sont les termes contenus dans les commentaires de toutes les classes dans les deux ontologies à aligner (ou les blocs à aligner). Un commentaire co_j est représenté par : $co_j = [\ w_{1,j}, \dots\dots\dots w_{T,j}\ ]$ où $w_{T,j}$ correspondent aux poids des termes t_i dans co_j.

La similarité entre deux commentaires est définie par le cosinus de l'angle entre deux vecteurs.

$$sim(co_i, co_j) = \frac{\sum_{k=1}^{T} w_{ki} w_{kj}}{\sqrt{\sum_{k=1}^{T} {w_{ki}}^2 * \sum_{k=1}^{T} {w_{kj}}^2}} \quad (3.21)$$

Pour définir les poids associés aux termes des commentaires, nous utilisons la technique de TF/IDF comme suit :

$$w_i = tf_i * idf_i \quad (3.22)$$

$$idf_i = \log_2 \frac{N}{n_i}$$

Où tf_i (fréquence de terme) est le nombre de fois où le i-ème terme dans l'espace U apparaît dans le commentaire de la classe c, idf_i (fréquence inverse de document), N est le nombre de classes dans les deux ontologies à aligner (ou les blocs à aligner), n_i est le nombre de classes qui contiennent le terme w_i au moins une fois.

La similarité des annotations de deux classes, est calculée par la mesure de Tversky. Dans ce cas, la fonctions intersection correspond à la somme pondérées des similarités calculées selon le type d'annotation comme suit :

$$f(C_{1_{annotations}} \cap C_{2_{annotations}}) = \sum_i^2 w_i sim(annotation_{i_1}, annotation_{i_2}) \quad (3.23)$$

Où $annotation_i$ est une annotation de type i et w_i est le poids associé à la valeur de similarité de même type (i). Si l'un des types est absent dans l'équation, son poids est remis à zéro.

Les différences peuvent être formalisée de la manière suivante :

$$f(C_{1_{annotations}} - C_{2_{annotations}}) = |C_{1_{annotations}}| - f(C_{1_{annotations}} \cap C_{2_{annotations}}) \quad (3.24)$$

$$f(C_{2_{annotations}} - C_{1_{annotations}}) = |C_{2_{annotations}}| - f(C_{1_{annotations}} \cap C_{2_{annotations}})$$

Remarque : Le langage OWL prédéfinit cinq propriétés d'annotations, à savoir : versionInfo, seeAlso, isDefinedBy, label, comment. Notre algorithme ne traite que les deux dernières.

Après avoir présenté la similarité entre les entités de type classe, nous allons maintenant présenter similarité de type propriété, et ensuite nous allons présenter la similarité adjacente.

3.4.2 La similarité entre les propriétés

Nous distinguons quatre catégories de composants dans une entité propriété : *propertyID*, *domain*, *range*, *axiome*.

Soit $P_1 \cap P_2 = \{$ *propertyID, domain*, range*, axiome* ,annotation** $\}$

Les fonctions de l'intersection et de la différence peuvent être formalisées de la manière suivante :

$$f(P_1 \cap P_2) = w_{propertyID}\, Sim_{propertyID}(p_1, p_2) + w_{domains}\, Sim_{domains}(p_1, p_2) \quad (3.25)$$
$$+ w_{ranges}\, Sim_{ranges}(p_1, p_2) + w_{axiomes}\, Sim_{axiomes}(p_1, p_2) + w_{annotations}$$
$$Sim_{annotations}(p_1, p_2).$$

Et

$$f(P_1 - P_2) = |P_1| - f(P_1 \cap P_2) \quad (3.26)$$
$$f(P_2 - P_1) = |P_2| - f(P_1 \cap P_2).$$

Dans ce qui suit, nous détaillons la similarité de chaque composant.

3.4.2.1 La similarité de propertyID

Le propertyID est également une chaine de caractère. Tandis que le classID sert à identifier de manière unique la classe dans l'ontologie, le propertyID est employé pour identifier de manière unique la propriété dans l'ontologie. Ainsi, le calcule de la similarité entre des propertyID$_s$ est obtenu par extension du calcul de la similarité des classID$_s$.

3.4.2.2 La similarité des domaines et des rangs

Dans le langage OWL DL, on peut définir plusieurs domaines ou rangs pour une propriété et on devrait les interpréter comme une conjonction. Dans ce cas, la similarité est calculée comme la similarité de description sauf lorsque les rangs sont des dataRanges .

3.4.2.3 La similarité des axiomes

une propriété peut contenir plusieurs axiomes d'équivalence ou de spécification. Dans ce cas la similarité d'un axiome est calculée comme la similarité d'énumération en remplaçant la liste des classID par la liste des propertyID.

3.4.3 La similarité structurelle adjacente

Si l'entité à aligner est définie uniquement par son label et les relations de sous-classes qui le relient à d'autres entités, la similarité est calculée de la manière suivante :

$$Sim_{entity}(e_1, e_2) = w_{classID}\ Sim_{classID}(e_1, e_2) + w_{label}\ Sim_{label}(e_1, e_2) + w_{voisinage}\ Sim_{voisinage}(e_1, e_2) \quad (3.27)$$

Dans notre algorithme, les voisines d'une entité sont les entités ayant un lien de subsomption direct avec l'entité en question. Nous considérons trois types de voisines :

1. Les sous entités directes ;
2. Les super entités directes ;
3. Les entités sœurs.

Soit $V_{sous-entités}$, $V_{super-entités}$, $V_{\text{sœurs}}$ l'ensemble de sous entités, l'ensemble de super entités, l'ensemble d'entités sœurs respectivement. La similarité voisinage entre deux entités est calculée comme suit :

$$Sim_{voisinage}(e_1, e_2) = w_{V_{sous-entités}}\ Sim_{V_{sous-entités}}(e_1, e_2) + w_{V_{super-entités}}\ Sim_{V_{super-entités}}(e_1, e_2) + w_{V_{\text{sœurs}}}\ Sim_{V_{\text{sœurs}}}(e_1, e_2) \quad (3.28)$$

Pour calculer la similarité Sim_V, nous utilisons la similarité de tversky . La fonction d'intersection correspond à la cardinalité des deux ensembles de voisins.

$$f(V_1 \cap V_2) = |V_1 \cap V_2| \quad (3.25)$$

Les différences sont calculées comme suit :

$$f(V_1 - V_2) = |V_1| - f(V_1 \cap V_2) \quad (3.26)$$

$$f(V_2 - V_1) = |V_2| - f(V_1 \cap V_2)$$

Conclusion

Ce chapitre présente la méthode d'alignement proposée. Cette méthode intègre une étape de partitionnement pour pendre en compte le problème du passage à l'échelle des méthodes d'alignement. L'algorithme proposé consiste à partitionner chaque ontologie en blocs autour des ancres en utilisant les algorithmes de clustering, puis l'alignement s'effectue entre les blocs fournis en sortie. Dans la phase d'alignement, chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie en utilisant la mesure de Tversky pour comparer les structures internes des entités. Une évaluation de notre méthode sur deux jeux de test fera l'objet du chapitre suivant.

4

Implémentation et évaluation expérimentale.

Sommaire

Introduction

Ce chapitre présente les aspects implémentatoires de notre méthodes d'alignement ainsi que l'évaluation des algorithmes de partitionnement et d'alignement que nous avons proposés et présentés dans le chapitre 3. L'implémentation des algorithmes a été effectuée en Java sur eclipse dans l'application nommée L-O-S-A (Large Ontology Structural Alignment). Les évaluations ont été menées dans le but d'étudier leurs performances. Elles ont été faites sur deux jeux de test : le premier se compose de deux paires d'ontologies disponibles pour le partitionnement notamment Russia12 et TourimAB, et le deuxième se compose d'une ontologie large de la compagnie OAEI'09 nommé anatomy. Des résultats satisfaisants ont été obtenus.

4.1 Outils de développement

Nous avons réalisé l'application LOSA dans le langage java sur l'environnement de développement Eclipse la figure 4.1 montre son interface.

- **Langage java :**Nous avons utilisé comme langage de programmation ; le langage objet « java », le choix de langage java présente les avantages suivants :
  1. C'est un langage bien connu et largement répandu. Il existe de nombreuses librairies qui facilitent le développement des applications notamment des parser pour XML et OWL.
  2. Les applications java s'exécutent en utilisant une machine virtuelle, ce qui les rend indépendantes du système d'exploitation.
  3. Des machines virtuelles java ont été développées pour la plupart des systèmes actuels, ce qui facilite la portabilité des applications java,
  4. Les compilateurs java sont gratuits,
  5. Java permet de définir facilement des interfaces graphiques agréables à utiliser.

- **Environnement Eclipse** : Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate−forme modulaire pour permettre de réaliser des développements informatiques. La figure suivante montre son interface.

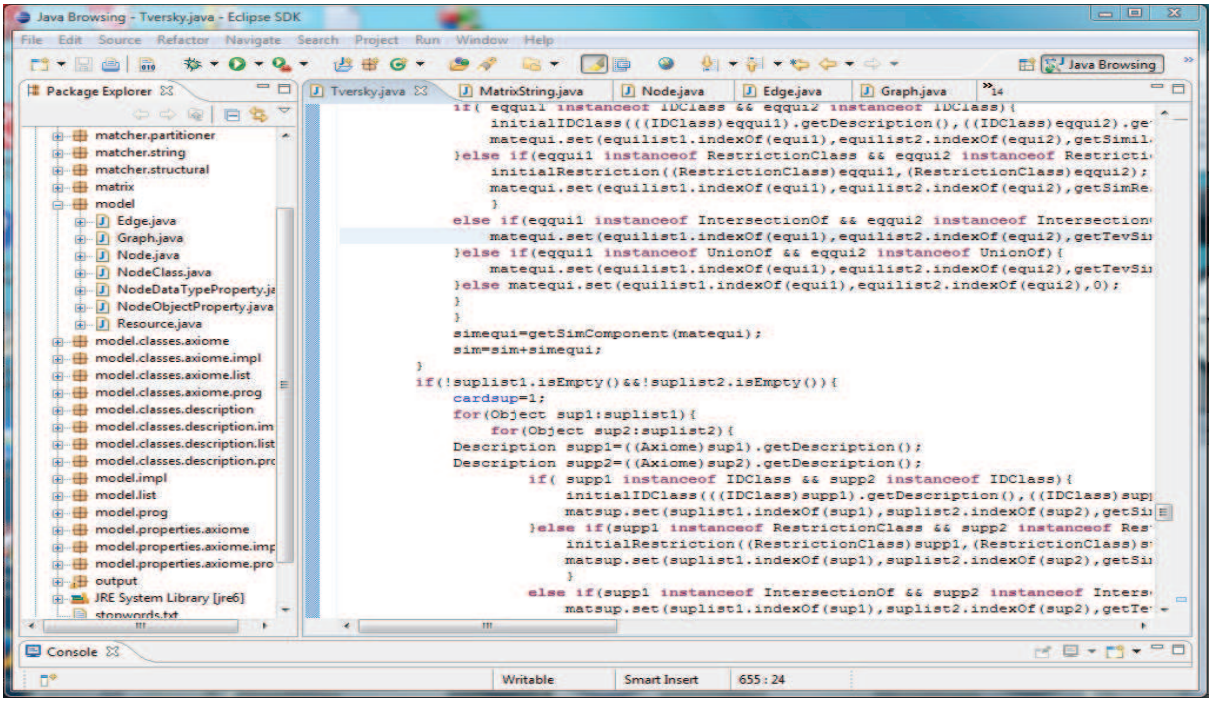

FIG4.1- *Environnement de développement (Eclipse 3.2)*

4.2 Développement de la méthode LOSA

l'application LOSA comprend cinq modules. Ces modules permettent le chargement, la représentation, le partitionnement, l'alignement des ontologies OWL et l'évaluation des résultats obtenus. La figure suivante présente le schéma d'application de la méthode LOSA.

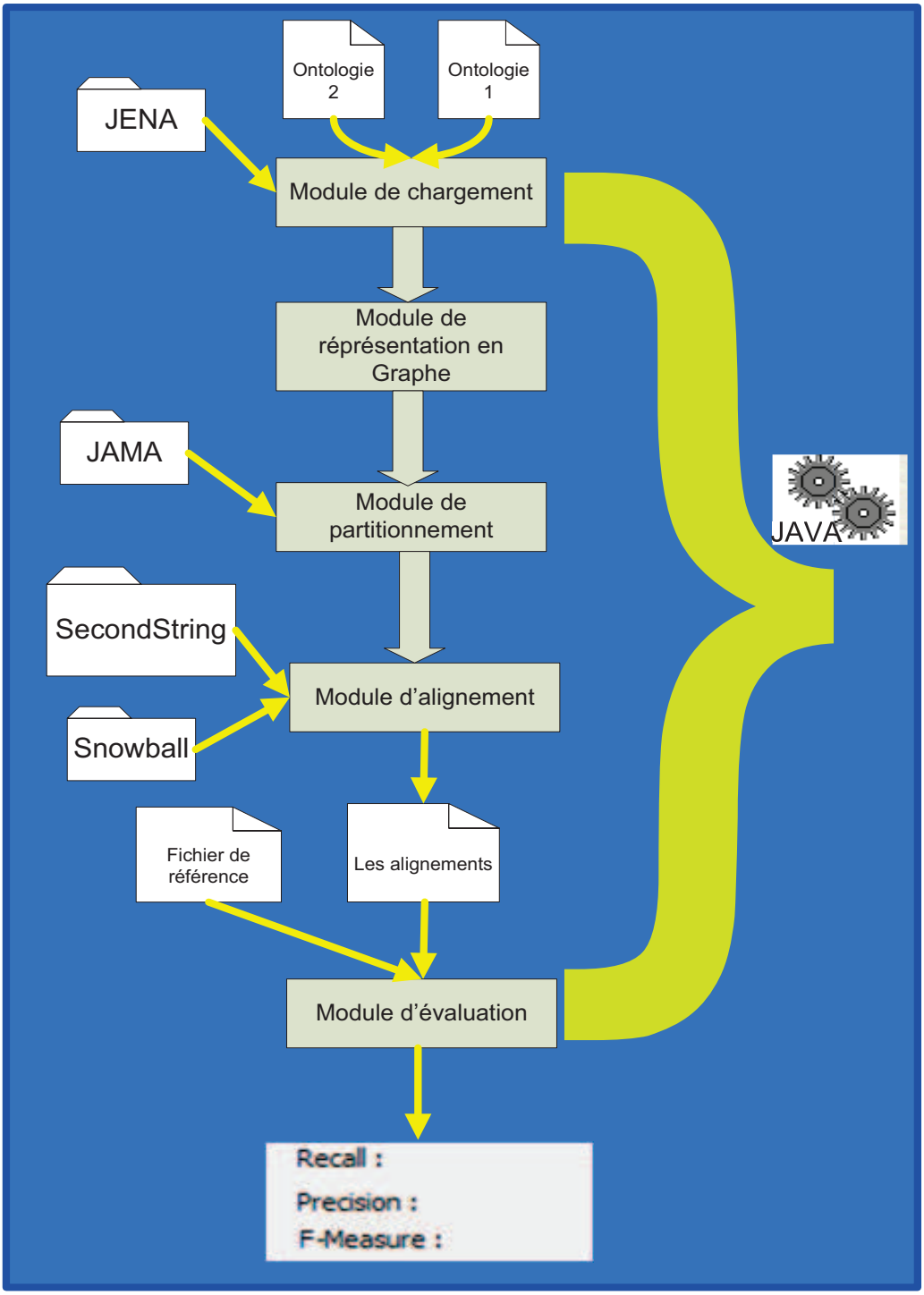

FIG 4.2 Schéma d'application

1. **Le module de chargement d' ontologies :** Pour charger les ontologies, nous avons utilisé la bibliothèque *Jena*[7] proposée par les laboratoires *HP* [52]. *Jena* est une bibliothèque de classes Java qui facilite le développement des applications pour le web sémantique. Elle permet :
   - La manipulation de déclarations RDF.
   - La lecture et écriture RDF/XML, Notation 3.
   - Le stockage en mémoire ou sur disque de connaissances RDF.
   - Le langage d'interrogation d'une base RDF.
   - La gestion d'ontologies : RDF-Schema, DAML+OIL, OWL.
2. **Le module de représentation en graphe :** Après le chargement des ontologies, elles sont représentées sous forme d'un OA-Graph.
3. **Le module de partitionnement en blocs :** concernant le module de partitionnement, nous avons utilisé la bibliothèque *JAMA*[8] (JAva MAtrix package) afin d'implémenter les différentes mesures de similarité dans une structure de type matrice. Celle ci présente la similarité entre les entités. *JAMA* est une bibliothèque de calcul numérique en développement qui contient plusieurs opérations sur les matrices.

La figure 4.3 montre l'interface de partitionnement. L'application prend en entrée les deux ontologies à partitionner, le nombre de blocs que l'on souhaite obtenir en sortie, le nombre d'entités maximum dans un bloc préliminaire et le nombre d'entités maximum dans un bloc final.

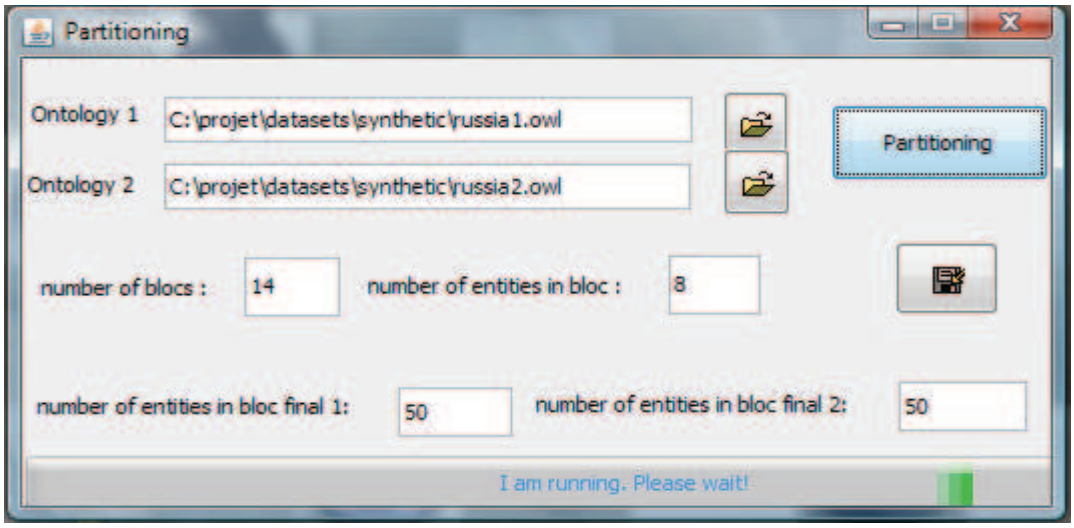

FIG4.3 – *L'interface de partitionnement*

[7] http://jena.sourceforge.net
[8] http://math.nist.gov/javanumerics/jama/

Les résultats fournis en sortie sont enregistrés sous format de fichier XML . Cet format est utilisé dans l'application Falcon-AO pour enregistrer le résultat de partitionnement. Les exemples suivants présentent deux blocs du résultat de partitionnement de la paire Russia12.

L'exemple de bloc de Russia1:

```
<Cluster>
  <name>9</name>
  <cut>0.0</cut>
     <uri rdf:resource="http://lonely.org/russia#person"/>
     <uri rdf:resource="http://lonely.org/russia#traveller"/>
     <uri rdf:resource="http://lonely.org/russia#president"/>
     <uri rdf:resource="http://lonely.org/russia#inhabitant"/>
     <uri rdf:resource="http://lonely.org/russia#animal"/>
     <uri rdf:resource="http://lonely.org/russia#living_being"/>
     <uri rdf:resource="http://lonely.org/russia#human_being"/>
     <uri rdf:resource="http://lonely.org/russia#plant"/>
</Cluster>
```

L'exemple de bloc de Russia2:

```
<Cluster>
 <name>9</name>
 <cut>0.0</cut>
    <uri rdf:resource="http://travel.org/russia#composer"/>
    <uri rdf:resource="http://travel.org/russia#human_being"/>
    <uri rdf:resource="http://travel.org/russia#robber"/>
    <uri rdf:resource="http://travel.org/russia#revolutionary"/>
    <uri rdf:resource="http://travel.org/russia#author"/>
    <uri rdf:resource="http://travel.org/russia#czar"/>
    <uri rdf:resource="http://travel.org/russia#dancer"/>
    <uri rdf:resource="http://travel.org/russia#normal_traveler"/>
    <uri rdf:resource="http://travel.org/russia#physician"/>
    <uri rdf:resource="http://travel.org/russia#living_being"/>
    <uri rdf:resource="http://travel.org/russia#president"/>
    <uri rdf:resource="http://travel.org/russia#animal"/>
    <uri rdf:resource="http://travel.org/russia#person"/>
    <uri rdf:resource="http://travel.org/russia#plant"/>
    <uri rdf:resource="http://travel.org/russia#foreign_business_person"/>
    <uri rdf:resource="http://travel.org/russia#student"/>
    <uri rdf:resource="http://travel.org/russia#musician"/>
    <uri rdf:resource="http://travel.org/russia#inspector"/>
</Cluster>
```

4. **Le module d'alignement d'ontologies :** Pour l'algorithme d'alignement, nous avons utilisé les mesures implémentées dans la bibliothèque *SecondString*[9] [13] développée à l'université de Carnegie Mellon. La lemmatisation est faite par *Snowball*[10], un *framework* qui implémente les algorithmes de stemmers[11]. Le résultat de l'alignement est enregistré sous format d'OAEI. L'exemple suivant présente un extrait du résultat d'alignement de la paire Russia12.

Un extrait du résultat de l'alignement de la paire Russia12 :

```
<Cell>
  <entity1 rdf:resource="http://lonely.org/russia#person"/>
  <entity2 rdf:resource="http://travel.org/russia#person"/>
  <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
  <relation>=</relation>
</Cell>
<Cell>
  <entity1 rdf:resource="http://lonely.org/russia#president"/>
  <entity2 rdf:resource="http://travel.org/russia#president"/>
  <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
  <relation>=</relation>
</Cell>
<Cell>
  <entity1 rdf:resource="http://lonely.org/russia#animal"/>
  <entity2 rdf:resource="http://travel.org/russia#animal"/>
  <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
  <relation>=</relation>
</Cell>
```

Deux entités

Relation d'équivalence entre les deux entités trouvées avec une mesure de similarité égale à 1

[9] http://secondstring.sourceforge.net/
[10] http://snowball.tartarus.org/
[11] Un stemmer est un programme ou un algorithme qui détermine la forme radicale à partir d'une forme infléchie ou dérivée d'un mot donné.

```
<Cell>
<entity1 rdf:resource="http://lonely.org/russia#living_being"/>
<entity2 rdf:resource="http://travel.org/russia#living_being"/>
<measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
<relation>=</relation>
</Cell>
<Cell>
<entity1 rdf:resource="http://lonely.org/russia#human_being"/>
<entity2 rdf:resource="http://travel.org/russia#human_being"/>
<measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
<relation>=</relation>
</Cell>
<Cell>
<entity1 rdf:resource="http://lonely.org/russia#plant"/>
<entity2 rdf:resource="http://travel.org/russia#plant"/>
<measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">1.0</measure>
<relation>=</relation>
</Cell>
```

5. **Le module d'évaluations de la qualité d'alignement :** Ce module est implémenté les mesures classiquement utilisées pour comparer les méthodes d'alignement (la précision, rappel et F-mesure).

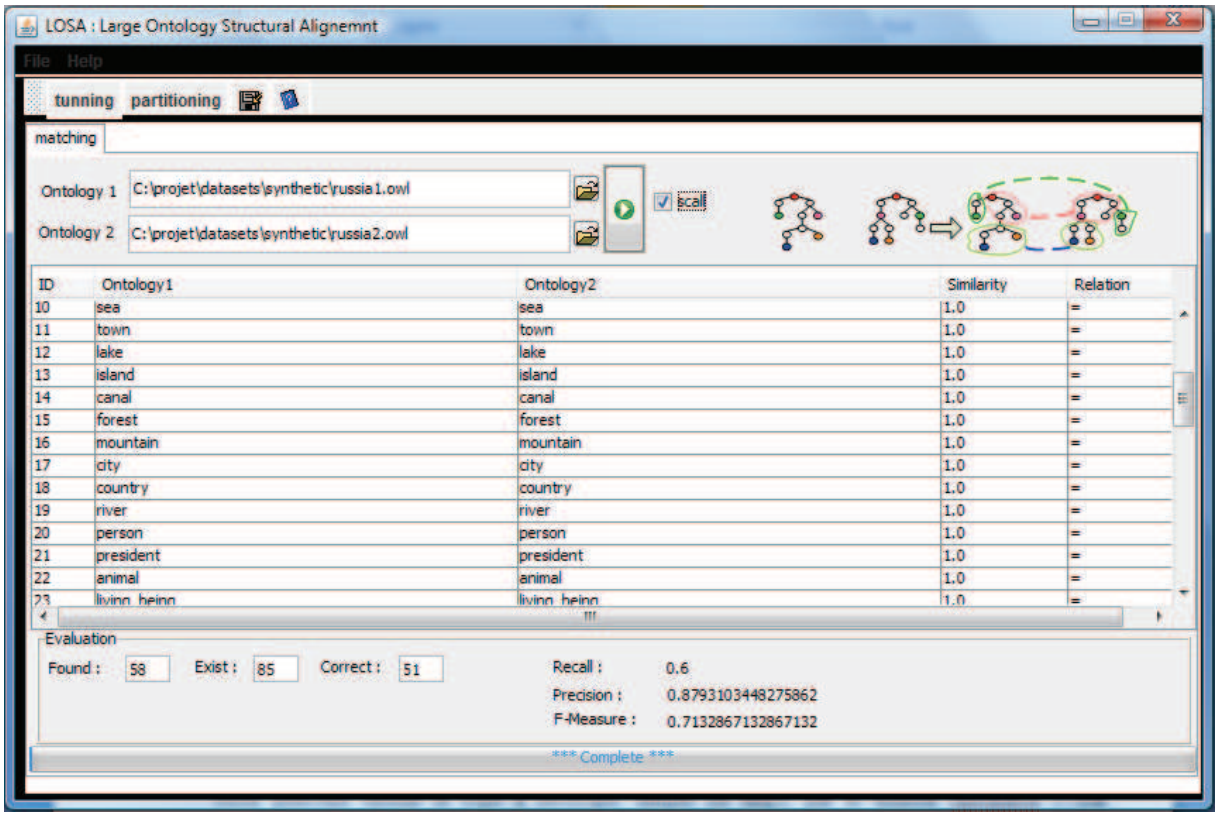

FIG 4.4 - *L'interface de l'application LOSA*

Nous pouvons choisir le type d'ontologie, simple ou large, par le bouton checkBox scall et lancer grâce au bouton " play " .

Les champs Found : Exist : Correct : présentent le nombre des alignement trouvés par la méthode , le nombre des alignement qui existe dans le fichier de référence et le nombre des alignements corrects trouvés par la méthode respectivement.

Les champs Recall : Precision : F-Measure : présentent les résultats d'évaluation de la méthode.

4.3 Evaluation des algorithmes sur les deux paires d'ontologies Rusia12 et TourismAB

Dans cette première évaluation, nous avons choisi deux paires d'ontologies : Russia12 et TourismAB. Ce choix se justifie par :

- la taille des deux ontologies est modérée ;
- les fichiers de références, qui contiennent les entités classifiées en blocs, sont disponibles ;

- possibilité de comparer des résultats de notre algorithme de partitionnement avec ceux des autres systèmes testés sur le même jeux de test.

Dans ce que suit, nous décrivons brièvement les deux paires de jeu de test.

Russia12. Russia1 et Russia2 sont deux ontologies qui ont été créées séparément par différentes personnes de deux contenus de sites web de voyages de la Russie. Russia1 contient 151 classes, 76 propriétés et 22 blocs dans son fichier de référence. Russia2 contient 162 classes, 81 propriétés et 22 blocs dans son fichier de référence.

TourismAB. Les deux ontologies sont créées séparément par différentes communautés décrivant le domaine du tourisme de MecKlenburg-Vorpommern (Allemagne). TourismA contient 340 classes, 97 propriétés et 18 blocs dans son fichier de référence. TourismB contient 447 classes, 100 propriétés et 23 blocs dans son fichier de référence.

Notre algorithme est testé sur une machine de technologie Intel Core 2 Duo CPU 2 GHz avec une mémoire de 3 GB DDR2, sur un système d'exploitation de Windows Vista, et un compilateur java 1.6.

4.3.1 Evaluation de l'algorithme de partitionnement

Nous utilisons la métrique d'entropie pour comparer les blocs générés automatiquement avec les blocs de référence [40].

Soit B l'ensemble de blocs générés automatiquement (|B|=n) et R l'ensemble de blocs de référence (|R|=m). b_i est un bloc dans B, tandis que r_j est un bloc dans R. $|b_i|$ est le nombre d'entités dans b_i , et $|r_j|$ est le nombre d'entités dans r_j .$b_i \cap r_j$ présente les entités communes dans les deux blocs b_i et r_j. Nous décrivons l'opération de base appelée « prec» qui mesure la précision des blocs générés par rapport aux blocs de référence.

$$\text{prec}(b_i, r_j)=\frac{|b_i \cap r_j|}{b_i} \quad . \tag{5.1}$$

L'entropie mesure la distribution des entités dans les blocs et reflète la qualité de partitionnement. Un score faible de l'entropie indique une meilleure qualité de partitionnement. Le meilleur score est donc, égale à 0 et le mauvais est égal à 1. L'entropie de l'ensemble B est définie de la manière suivante :

$$\text{entropy (B)}=\frac{1}{\sum_{i=1}^{n}|b_i|}\sum_{i=1}^{n}\text{entropy}(b_i).|b_i|. \tag{5.2}$$

$$\text{entropy}(b_i) = \frac{1}{\log m} \sum_{j=1}^{n} \text{prec}(b_i, r_j).\log(\text{prec}(b_i, r_j)). \quad (5.2)$$

L'histogramme de la figure 4.5 présente les résultats d'évaluation de quelques systèmes (PBM, BMO,COMA) en utilisant cette métrique [31,30,16]. Le critère d'arrêt du partitionnement est l'obtention du nombre de blocs de référence. L'histogramme présenté ci-dessous indique que les résultats d'expérimentation de PMB est dominant par rapport aux autres systèmes dans tous les tests.

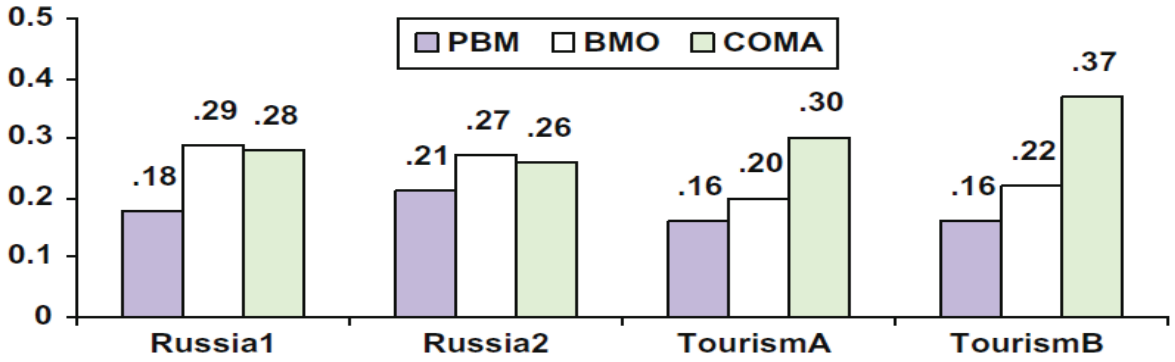

FIG. 4.5 - *L'entropie de PBM, BMO et COMA [31].*

Notre algorithme de partitionnement est orienté alignement. La phase de partitionnement préliminaire des ancres est la plus importante car les blocs sont générés autour des ancres. Notre objectif est de partitionner les ontologies de façon à n'avoir aucun bloc isolé et diminuer le nombre de combinaisons à faire lors du processus d'alignement (un bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième). En sortie, le nombre de blocs générés est exactement égal au nombre de bloc d'ancres généré lors de la phase de partitionnement préliminaire. Cependant, les blocs de référence disponibles sont construits manuellement sans prendre en compte l'objectif de l'alignement. Pour cela, nous adaptons les fichiers de référence aux fichiers de référence d'ancres pour évaluer notre algorithme de partitionnement préliminaire des ancres.

Dans notre expérimentation, le nombre de blocs de référence après l'adaptation est comme suit : 13 blocs pour Russia1, 13 pour Russia2, 11 pour TourismA, et 14 pour TourismB. On a 55 ancres dans Russia12 et 144 dans TourismAB.

❖ **Démarche expérimentale :** Les expérimentations que nous avons menées ont eu pour objectifs de calculer l'entropie de notre algorithme de partitionnement. Pour chaque paire d'ontologies, l'algorithme de partitionnement est exécuté plusieurs fois avec plusieurs valeurs présentant le nombre de blocs que l'on souhaite obtenir en sortie.

Les histogrammes de la figure 4.6 et 4.7 présentent les résultats d'évaluation de notre algorithme.

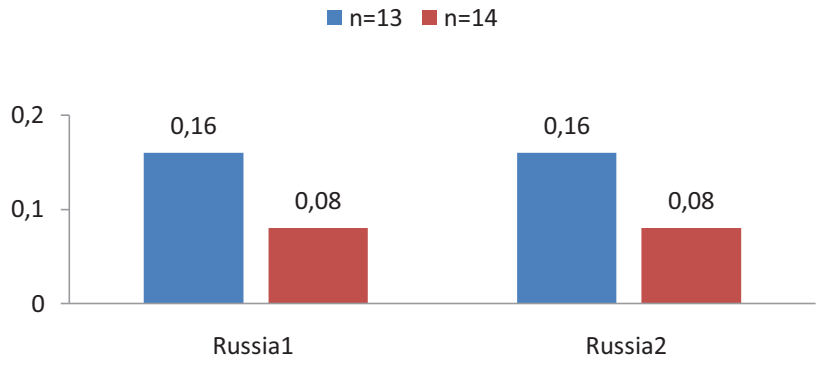

FIG. 4.6- *L'entropie de l'algorithme de partitionnement d'ancres (ontologie Russia12)*

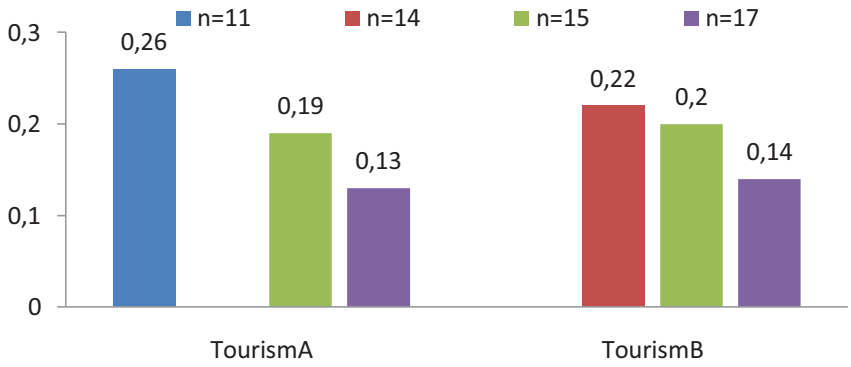

FIG. 4.7- L'entropie de l'algorithme de partitionnement d'ancres (ontologies TourismAB)

Les résultats renvoyés par l'algorithme sont bons. Dans Russia12, si le nombre de blocs générés est égal au nombre de blocs de référence (n=m) l'entropie est égale à 0.16. Si nous augmentons le nombre de blocs générés, la valeur de l'entropie diminue (0.08). Pour l'ontologie TourismAB, nous avons appliqué plusieurs valeurs du nombre de blocs générés notamment celui de référence (11 et 14) ainsi que les valeurs 15 et 17 (voir Fig. 7) et ceci afin de montrer l'influence de la valeur du nombre de blocs générés sur la qualité de

partitionnement. Nous observons que l'entropie est changée en fonction du nombre de blocs générés choisi. Pour cela, le nombre de blocs a été choisi de façon à n'avoir aucun impact négatif sur l'alignement (i.e. la complexité de l'algorithme et le temps d'exécution). Les valeurs de l'entropie (0.26, 0.22) s'expliquent par le fait que le calcul de similarité structurelle se base seulement sur la taxonomie décrite par l'ontologiste et ignore les liens prédéfinis entre les classes et les propriétés OWL. Par exemple, les deux propriétés suivantes sont des ancres dans l'ontologie TourismA.

```
<rdf:Description rdf:about='http://meh/tourism1#DEFAULT_ROOT_RELATION'>
   <rdf:type rdf:resource='http://www.w3.org/2002/07/owl#ObjectProperty'/>
 </rdf:Description>
 <rdf:Description
rdf:about='http://meh/tourism1#DEFAULT_ROOT_DATATYPE_RELATION'>
   <rdf:type rdf:resource='http://www.w3.org/2002/07/owl#DatatypeProperty'/>
 </rdf:Description>
```

Elles sont classifiées dans deux clusters distincts car notre algorithme ne découvre pas le lien structurel à partir de leurs types (les deux propriétés DEFAULT_ROOT_RELATION et DEFAULT_ROOT_DATATYPE_RELATION' sont sous-propriétés de Property).

4.3.2 L'efficacité de l'algorithme de partitionnement pour l'alignement

Pour évaluer la qualité d'une méthode d'alignement, on compare les résultats produits par cette méthode par rapport à un alignement de référence existe (manuellement). Cette comparaison est largement utilisée dans les évaluations typiques de la recherche d'information basée sur Les mesures classiques de rappel et de précision. Cette mesure modélise le problème de manière ensembliste,(voir FIG 4.8) :

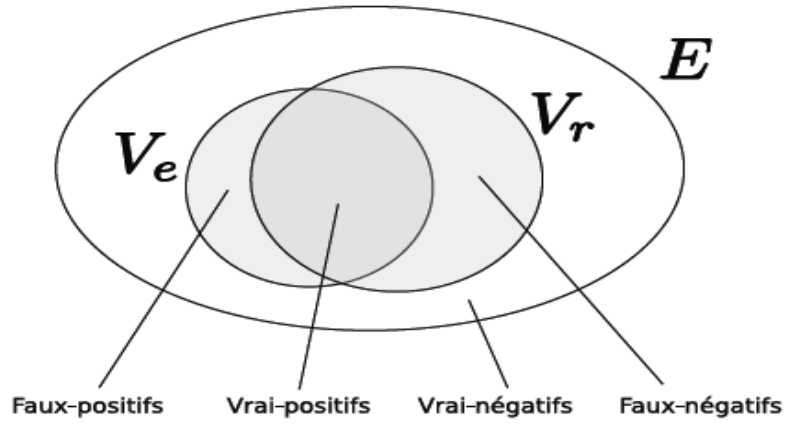

FIG 4.8 - *Diagramme de Venn du modèle d'évaluation classique [32]*

Et considère les ensembles suivants :

- vrai-positifs : les éléments de correspondance trouvés par la méthode qui sont dans l'alignement de référence.
- faux-positifs : les éléments de correspondance trouvés par la méthode qui ne sont pas dans l'alignement de référence.
- faux-négatifs : les éléments de correspondance de l'alignement de référence qui n'ont pas été trouvés par la méthode.

Les cardinalités des ensembles des vrai-positifs, des faux-positifs et des faux-négatifs sont exprimés à partir de V_e et V_r . Où

- V_e : représente l'ensemble des éléments de correspondance d'un alignement à évaluer produit par la méthode.
- V_r : représente l'ensemble des éléments de correspondance d'un alignement de référence.

La précision, notée *P*, représente la proportion de vrai-positifs parmi l'ensemble des éléments de correspondance trouvés par la méthode. Cette mesure permet de qualifier la pertinence de la méthode d'alignement. Plus la valeur de précision se rapproche de 1, moins la méthode produit de bruit (La proportion de mauvais dans les retournés : Bruit = 1 – $P(\mathrm{V_e}, \mathrm{V_r})$).

$$P(V_e, V_r) = \frac{|V_e \cap V_r|}{|V_e|} \tag{4.1}$$

Le rappel, noté *R*, représente la proportion de vrai-positifs parmi l'ensemble des éléments de correspondance contenus dans l'alignement de référence. Cette mesure permet de quantifier la couverture de la méthode d'alignement. Plus le rappel se rapproche de 1, moins la méthode est silencieuse (La Proportion de manqués dans les pertinents : Silence = 1 – $R(V_e, V_r)$).

$$R(V_e, V_r) = \frac{|V_e \cap V_r|}{|V_r|} \tag{4.2}$$

Enfin, La mesure La F-mesure, notée F, représente la moyenne harmonique entre la précision et le rappel. Elle permet de comparer les performances des méthodes par une seule mesure.

L'histogramme de la figure 4.9 présente les résultats d'évaluation de l'algorithme de Falcon-AO sans et avec partitionnement en utilisant les mesures de rappel et du précision [40] sur les deux paires Russia12 et TourismAB. Nous observons que certains alignements sont perdu à cause de partitionnement. Par exemple, dans Russia12, le rappel de Falcon-AO sans partitionnement est 0.65 et .59 pour PBM, ce qui signifie que 6% des alignements ne sont pas trouvés par PBM.

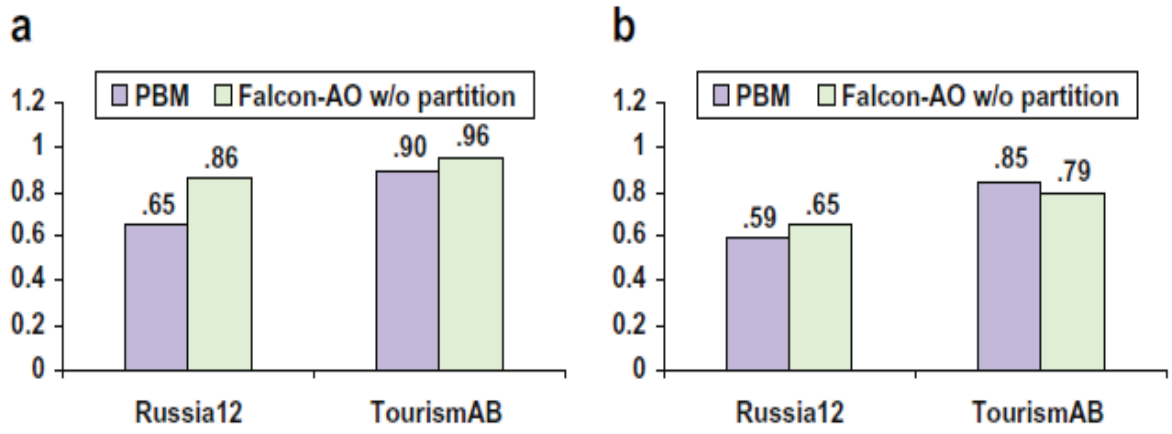

FIG. 4.9 - *La précision et Le rappel de la méthode Falcon sans et avec partitionnement : (a) précision, and (b) rappel [31]* .

- **Démarche expérimentale :** Pour évaluer l'efficacité de notre algorithme de partitionnement pour la méthode d'alignement, nous avons appliqué notre méthode d'alignement sans et avec partitionnement. Le seuil utilisé pour la paire Russia12 est égale à 0.56 et 0.68 pour la paire TourismAB.

Les histogrammes de la figure 4.10 et 4.11 présentent les résultats obtenus. Pour la méthode d'alignement, les résultats renvoyés sont satisfaisants. Cependant, plusieurs alignements ne peuvent pas être trouvés parce que dans les fichiers de référence, nous avons plusieurs entités ayant plusieurs entités correspondantes. Par exemple, pour la paire TourismAB nous avons les correspondances : `Flugzeug-Flugzeug`, `Flugzeug-Luftverkehrsmittel`, `Kathedrale- Kathedrale`, `Kathedrale-Dom`, `Mensch-Mensch`, `Mensch-Person`, tandis que notre méthode ne cherche qu'une seule correspondante pour chaque entité. Certaines correspondances ne peuvent pas être détectées par la version d'implémentation actuelle de notre algorithme car dans cette version, nous n'employons pas encore la similarité de voisinage. Par esxemple, pour la paire Russia12, certaines correspondances telles que `area` et `region`, `political_area` et `political_region` ne peuvent pas être détectées .

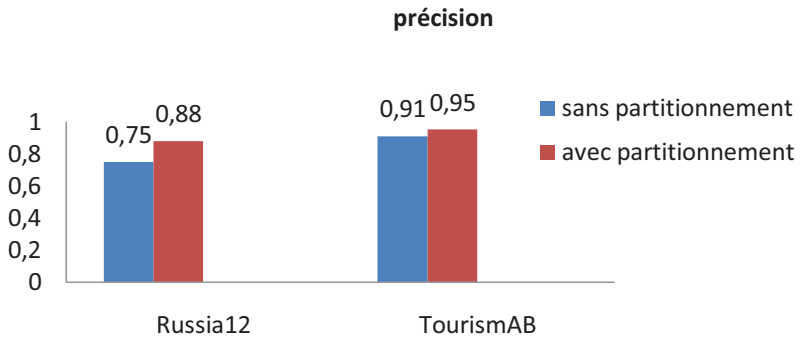

FIG. 4.10 – *La précision de la méthode sans et avec partitionnement*

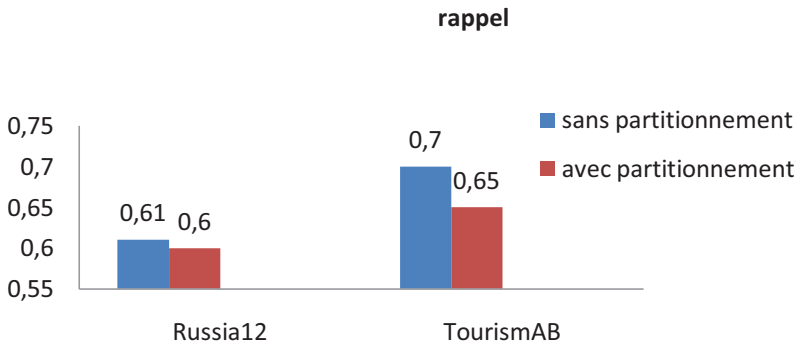

FIG. 4.11 – *Le rappel de la méthode sans et avec partitionnement*

La qualité d'alignement par partitionnement dépend de la qualité d'alignement et la qualité de partitionnement. Notre objectif est de préserver les alignements après le partitionnement. Nous avons observé que les résultats sont efficaces. Par exemple, dans la paire Russia12, on a

69 alignements trouvés avec 52 corrects et 85 alignements de référence. Après le partitionnement, le nombre d'alignements trouvés est 58 avec 51 corrects. Pour la paire TourismAB, on a 174 alignements trouvés avec 158 corrects et 226 alignements de référence. Après le partitionnement, le nombre d'alignements trouvés est 155 avec 147 corrects. Le résultat est bon mais n'est pas parfait vue par la qualité de partitionnement de cette paire.

4.4 L'évaluation de l'algorithme d'alignement par partitionnement sur les ontologies larges de la compagnie OAEI 2009

Plusieurs jeux de test sont utilisés dans le cadre de la compagnie d'évaluation OAEI'09 (benchmark, anatomy, conference, fishery gears…etc.). Notre évaluation a été faite sur le jeu de test d'anatomy. Ce jeu de test contient une ontologie de référence en format RDF/XML qui contient 988 alignements et deux ontologies décrites comme suit :
Un thésaurus décrivant l'anatomie humaine publié par l'institut nationale de cancer (NCI) qui contient 3304 entités, et le dictionnaire de l'anatomie de la souris. Ce dernier a été créé dans le cadre du projet de base de données d'expression de gène souris qui contient 2744 entités.

- **Démarche expérimentale :** Pour évaluer notre algorithme d'alignement par partitionnement sur des ontologies larges, nous avons appliqué notre méthode d'alignement sur des ontologies qui comportent plus de mille de concepts (le jeu de test d'anatomy). Les ontologies de ce jeu de test (i.e. l'ontologie de l'anatomie humaine et celle de la souris) utilisent des noms non significatifs pour nommer leurs concepts, par exemple, ″ MA_0000590 ″ dans l'ontologie souris et ″ NCI_C40185 ″ dans l'ontologie NCI. Pour cela, nous avons utilisés les étiquettes pour déterminer les ancres. Deux étiquettes sont équivalentes si et seulement si leur similarité supérieure ou égale à 0.75. Pour ce test, nous avons appliqué plusieurs valeurs du seuil (0.75, 0.78, 0.80, 0.85, 0.90, 0.95, 1.0) . L'histogramme de la figure 4.12 présente Les valeurs de f-mesure, de précision et de rappel. Les résultats renvoyés sont bons.

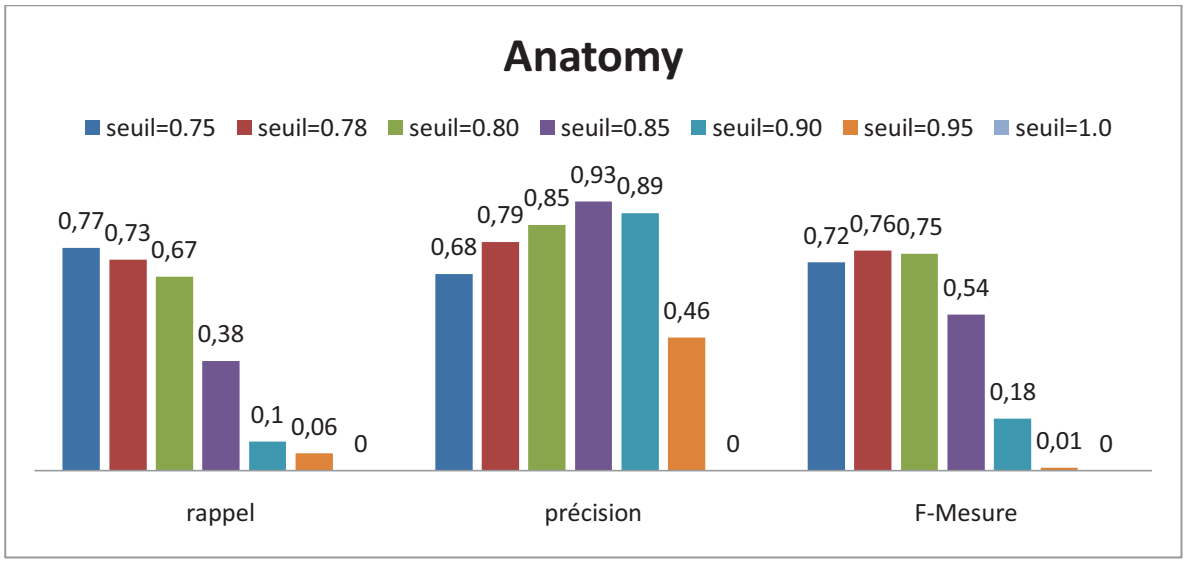

FIG. 4.12 – *Le résultat d'évaluation de la méthode d'alignement sur l'ontologie Anatomy*

Conclusion

Dans ce chapitre, nous avons tout d'abord présenté l'implémentation de notre outil d'alignement d'ontologies avec partitionnement. Ensuite, nous avons abordé les expérimentations réalisées dans le but d'étudier les performances de nos algorithmes de partitionnement et d'alignement. Ces expérimentations sont effectuées sur deux jeux de test : deux paires d'ontologies disponible pour le partitionnement Russia12 et TourismAB, et une ontologie large de la compagnie OAEI'09 nommée Anatomy. Nous poursuivons actuellement les expérimentations pour tester l'efficacité de notre méthode d'alignement structurelle sur d'autres ontologies à grande échelle comme Food, Library…etc. Cet algorithme sera appliqué sur les ontologies réelles, complexes, et de forte hétérogénéité.

Conclusion générale et Perspectives

Le web sémantique est vu comme la solution à l'exploitation pertinente et automatique des informations disséminées sur la toile. Ces informations contenues dans des ressources telles que des pages web, des bases des données, des services… seront compréhensibles non seulement pour les hommes mais aussi pour les machines, les programmes ou les agents informatiques. En ce qui concerne les informations, et plus particulièrement leur représentation et leur sémantique, l'ontologie est apparue comme un moyen de structuration formelle, contribuant à faciliter leur compréhension. Cependant ces ontologies sont très souvent source d'hétérogénéité.

Les travaux mènes dans cette thèse se situent dans le domaine du Web sémantique. Notre objectif est de tirer profit des travaux mènes notamment dans le domaine de l'interopérabilité sémantique des connaissances, pour aligner des ontologies, plus particulièrement des ontologies larges.

Pour atteindre cet objectif d'alignement des ontologies larges, nous proposons une méthode d'alignement structurelle avec partitionnement tout en préservant les alignements.

Contributions

De façon générale, les travaux effectués ont pour objectif de relever le challenge du passage à l'échelle des méthodes d'alignement d'ontologies. Pour cela, nous proposons une méthode d'alignement par partitionnement. Notre contribution se décline en deux points :

1. **Proposition d'un algorithme de partitionnement**

   Nous proposons un algorithme de partitionnement d'ontologies autour des ancres. Notre algorithme a l'avantage de n'avoir aucun bloc isolé en assurant la préservation des alignements et fournit en sortie des paires de blocs à aligner. Chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie.

2. **Proposition d'une méthode d'alignement structurelle**

   Nous proposons une méthode structurelle qui prend la syntaxe abstraite du langage OWL comme un pattern général pour décrire les classes et les propriétés. La similarité entre deux structures internes de deux entités est déduite en utilisant la mesure de Tversky.

Limites et Perspectives

Cependant, les algorithmes que nous proposons sont implémentés en Java pour évaluer la qualité de partitionnement et d'alignement des ontologies. Le programme réalisé n'est pas encore optimisé au niveau de la performance.

En plus, plusieurs ontologies sont multilingue. Cependant, notre méthode d'alignement n'utilise pas encore une ressource externe comme Wordnet pour enrichir les ontologies avant de les aligner.

Différentes perspectives sont envisagées :

- Nous allons utiliser la ressource WordNet pour améliorer le résultat d'alignement.
- Nous poursuivons actuellement les expérimentations pour tester l'efficacité de notre méthode d'alignement structurelle sur d'autres ontologies à grande échelle comme Food, Library…etc. Cet algorithme sera appliqué sur des ontologies réelles, complexes, et de forte hétérogénéité;
- Une autre approche intéressante est d'étudier les relations entre les patterns d'ontologies et les différentes méthodes d'alignement. Il est probablement intéressant de pouvoir créer des méthodes d'alignement plus rapides, plus robustes et sans avoir recours aux hiérarchies ontologiques.

Bibliographie

1. Bach, T.L.: *Construction d'un Web Sémantique Multi-Points de Vue*. These de doctorat Informatique, Ecole des Mines de Paris a Sophia Antipolis, (2006).
2. Baziz, M., Boughanem, M., Pasi, G., Prade, H.: *An Information Retrieval Driven by Ontology from Query to Document Expansion*. Proceedings of the 8th Conference on Large-Scale Semantic Access to Content (Text, Image, Video and Sound), RIAO, (2007).
3. Benerecetti, M., Bouquet, P., Ghidini, C.: *Contextual Reasoning Distilled*. Journal of Theoretical and Experimental Artificial Intelligence, 12(3):279–305, (2000).
4. Benjamins, V. R., Contreras J., Corcho O., Gomez-Perez, A.: *Six Challenges for the Semantic Web*. In KR2002 : Semantic Web workshop, (2002).
5. Berkovsky, S., Eytani, Y., Gal, A.: *Measuring the Relative Performance of Schema Matchers*. In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence. WI'05.366-371, (2005).
6. Berners-Lee, T.: *Weaving the Web.* Harper Eds, San Francisco, 226 p, (1998).
7. Berners-Lee, T.: *Artificial intelligence and the semantic Web*. AAAI Keynote, Boston, USA, (2006).
   http:// www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html.
8. Bisson G.: *La similarité: Une Notion Symbolique/Num*érique. Apprentissage symbolique-numérique (tome 2). Eds Moulet, Brito. Editions CEPADUES. pp. 169-201, (2000).
9. Bouquet P., Giunchiglia F., Van Harmelen F., Serafini L. and Stuckenschmidt H.,: *Contextualizing Ontologies*. Journal of Web Semantics, 1(1):325–343, (2004).
10. Chebotko, A., Lu, S., Fotouhi, F.: *Challenges for Information Systems Towards the Semantic Web*. AIS SIGSEMIS Semantic Web and Information Systems Newsletter, Volume 1, Number 1, pp. 26-28, (2004).
11. Cohen, W., Ravikumar P., Fienberg, S.: *A Comparison of String Metrics for Matching Names and Records* . Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb 03), p. 73-78,(2003)

12. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: *Querying the Semantic Web with the Corese Search Engine*. Dans Proc. 15th ECAI/PAIS, Valencia (ES), August IOS Press,(2004).

13. Cox, T. et Cox, M.: *Multidimensional Scaling*. Chapman and Hall, (1994).

14. Dan, C., Van, H., F., Horrocks, I., Deborah L., McGuinness, Patel-Schneider, P., Stein, L., A.: *DAML+OIL Reference Description*. W3C Note 18 December (2001). http://www.w3.org/TR/daml+oil-reference.

15. Dictionnaire de l'Académie française, neuvième édition, Version informatisée. http://atilf.atilf.fr/academie9.htm

16. Do, H.H. , Rahm, E.: *Matching Large Schemas: Approaches and evaluation, Information Systems*, vol.32 n.6, pp.857--885. (2007)

17. Ehrig, M., Staab, S.: *QOM - Quick Ontology Mapping*. In International Semantic Web Conference pp. 683-697, (2004).

18. Euzenat, J., Valtchev, P.: Similarity-based Ontology Alignment in OWL-lite. In Proc. 15th ECAI, pages 333–337, Valencia (ES), (2004).

19. Eyal, A., Sheila, M.: P*artition-based Logical Reasoning for First-Order and Propositional theories*. Artificial Intelligence, Vol. 162, pp. 49--88. (2005)

20. Giunchiglia, F., Shvaiko, P. et Yatskevich, M.: *S-Match: an Algorithm and an Implementation of Semantic Matching*. Dans Proceedings of ESWS 2004, Heraklion (GR), pages 61–75, (2004).

21. Gruber, T.R.: *A Translation Approach to Portable Ontologies*. Knowledge Acquisition, 5(2):199-220, (1993).

22. Guarino, N., Giaretta, P.: *Ontologies and Knowledge Bases-Towards a Terminalogical Clarification*. Dans N.J. Mars, editor, Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995, pages 25--32. IOS Press, Amsterdam, (1995).

23. Guha, S., Rastogi, R., Shim, K.: *ROCK: a Robust Clustering Algorithm for Categorical Attributes*. In Proceedings of the 15th International Conference on Data Engineering. 512-521, (1999)..

24. Gurevych, I., Strube, M.: *Semantic Similarity Applied to Spoken Dialogue Summarization*. In Proceedings of the 20th International Conference on omputational Linguistics, Geneva, Switzerland, 23 -27, pp. 764-770, (2004).

25. Hamdi, F., Zargayouna, H., Safar, B., Reynaud, C.: *TaxoMap in the OAEI alignment contest In Ontology Alignment Evaluation Initiative (OAEI) 2008 Campaign*, 8p, Workshop ISWC'08, Karlsruhe, Germany, 26-30 October (2008).

26. Hayes, J., Gutiérrez, C.: *Bipartite Graphs as Intermediate Model for RDF*. International Semantic Web Conference, 47-61, (2004).

27. Hirst, G., St-Onge, D.: *Lexical Chains as Representation of Context for the Detection and Correction Malapropisms*. In C. Fellbaum (Ed.), WordNet : An electronic lexical database, Chapter 13, pp. 305–332. The MIT Press, (1989).

28. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL : *A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission 21 May 2004, (2004). http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

29. Hu, W., Jian, N., Qu, Y., Wang, Y.: *GMO: A Graph Matching for Ontologies*. in: Proceedings of K-CAP Workshop on Integrating Ontologies, pp. 41-48, (2005).

30. Hu, W., Qu, Y.: *Block Matching for Ontologies*. In: LNCS, vol. 4273. Springer. pp. 300--313. (2006)

31. Hu W., Qu Y., Cheng G.: *Matching Large Ontologies: A Divide-and-Conquer Approach*. Journal on Data and Knowledge Engineering, vol.67, n°1, p.140-160, (2008).

32. Jérôme, D.: *AROMA : Une Méthode pour la Découverte d'Alignements Orientés entre Ontologies à Partir de Règles d'Association*. Thèse doctorat de à l'Ecole Polytechnique de l'Université de Nantes, (2007).

33. Jian, N., Hu, W., Cheng, G., Qu, Y.: *Falcon-AO: Aligning Ontologies with Falcon*. in: Proceedings of K-CAP Workshop on Integrating Ontologies, pp. 85-91, (2005).

34. Jianbo, S., Jitendra, M.: *Normalized Cuts and Image Segmentation*. IEEE Transactions on PAMI, Vol. 22, No. 8, pp. 888--905. (2000).

35. Jay, J., Jiang, David, W., Conrath.: *Semantic Similarity Based on Corpus Statistics and Lexical Laxonomy*. Proceedings of Internat. Conf. on Research in Computational Linguistics, Taiwan,(1997).

36. Kasri, S., Benchikha, F.: *Un Algorithme de Partitionnement d'Ontologies Orienté Alignement* Procceding of COSI'10 (le Colloque sur l'Optimisation et les Systèmes d'Information). Ouargla, Algérie, (2010).

37. Kasri, S., Benchikha, F.: *Large-scale Ontologies: Alignment-based Partitioning.* Procceding of ICWIT'10 (International Conference on Web and Information Technologies). Marrakech, Marocco. In RNTI, Revue des Nouvelles Technologies de l'Information, (à paraitre).

38. Kéfi, H.: *Ontologies et Aide à l'Utilisateur pour l'Interrogation de Sources Multiples et Hétérogène*s. Thèse de doctorat de l'Université Paris-Sud, (2006).

39. Lin D.: *An Information-Theoretic Definition of Similarity*. In Proc. of the 15th Int. Conf. on Machine Learning: Morgan Kaufmann, 296–304,(1998).

40. Lord, P.W., Stevens, R.D., Brass, A. Goble, C.A.: *Semantic Similarity Measures as Tools for Exploring the Gene Ontology*. Pacific Symposium on Biocomputing 8, pp.601-612, (2003).

41. McBride, B.: *Jena : A Semantic Web Toolkit*. IEEE Internet Computing 06, no. 6, p. 55-59, (2002).

42. Mike, D., Guus, S.: *OWL Web Ontology Language Reference*. W3C recommendation, W3C, February (2004).

43. National Committee for Information Technology Standards, Technical Committee T2(Information Interchange and Interpretation). Draft proposed American national standard for Knowledge Interchange Format,(1998). http://logic.stanford.edu/kif/dpans.html.

44. Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., Swartout, W.R.: Enabling Technology for Knowledge Sharing. Dans AI Magazine, pp. 36-56, (1991).

45. OAEI-2008.: *Ontology Alignment Evaluation Initiative.* in cooperation with the ISWC Ontology Matching workshop, Karlsruhe, Germany, (2008).

46. Deborah L. McGuinness, Van, H., F.: OWL Web Ontology Language Overview, W3C Recommendation, (2004). http://www.w3.org/TR/owl-features/.

47. Gomez-Perez, A.: *Ontological Engineering: A state of the art*. Expert Update, 2(3), 33-43, (1999).

48. Pham, T.A.L., Thanh, N. L., Sander, P.: *Some Approaches of Ontology Decomposition in Description Logics*. 14th ISPE International Conference on Concurrent Engineering, CE2007, São José dos Campos, SP, Brazil, (2007)

49. Rada, R., Mili, H., Bicknell, E., Blettner, M.: *Development and Application of a Metric on Semantic Nets.* IEEE Transaction on Systems, Man, and Cybernetics 19(1), 17–30, (1989).

50. Rahm, E., Bernstein, P.: *A Survey of Approaches to Automatic Schema Matching*. VLDB Journal, 10(4):334–350, (2001).

51. Brickley, D., R. V. Guha.: *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Recommendation, (2004). http://www.w3.org/TR/rdf-schema/

52. Resnik, P.: *Using Information Content to Evaluate Semantic Similarity in Taxonomy*. In Proceedings of 14th International Joint Conference on Artificial Intelligence, Montreal, (1995).

53. Resnik, P.: *Semantic Similarity in A Taxonomy: An Information-Based Measure and Its Application to Problems of Ambiguity in Natural Language*. Journal of Artificial Intelligence Research (JAIR), 11, 95–130, (1999).

54. Reynaud, C., Safar, B.: *Techniques Structurelles d'Alignement pour Portails Web*. In RNTI, Revue des Nouvelles Technologies de l'Information, (2007).

55. Roche, C.: *Terminologie et Ontologie*. Revue Langages, numéro 157,Mars (2005).

56. Rodriguez, M., Egenhofer, M.: *Determining Semantic Similarity Among Entity Classes from Different Ontologies*. IEEE Trans Knowl. Data Eng;15(2):442–56, (2003).

57. Salton, G., Buckley, C.: *Improving Retrieval Performance by Relevance Feedback*. Journal of the ASIS, 41:4,288-297, (1990).

58. Sayyadian, M., Lee, Y., Doan, A., Rosenthal, A.: *Tuning Schema Matching Software using Synthetic Scenarios*. In Proceedings of VLDB, (2005).

59. Seco, N., Veale, T., Hayes, J.: *An Intrinsic Information Content Metric for Semantic Similarity in Wordnet*. In Proceedings of ECAI'2004, the 16th European Conference on Artificial Intelligence, (2004).

60. Shvaiko, P., Euzenat J.: *A Survey of Schema-Based Matching Aapproaches*. Journal on Data Semantics, IV:146–171, (2005).

61. Stuckenschmidt, H., Klein, M.: *Structured-Based Partitioning of Large Concept Hierarchies*. In International Semantic Web Conference- ISWC, pp. 289--303. (2004).

62. Svab, O.: *Exploiting patterns in ontology mapping*. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea. In:LNCS, Berlin, Heidelberg, Springer Verlag. 4825:950-954, (2007).

63. Tversky, A.: *Features of Similarity*. Psychological Review 84(4), 327–352, (1977).

64. Valtchev, P.: *Construction Automatique de Taxonomies pour l'Aide à la Représentation de Connaissances par Objets*. Thèse d'informatique, Université Grenoble 1, (1999).

65. Visser, P.R.S., Jones, D.M., Bench-Capon, T.J.M., Shave, M.J.R.: *Analysis of Ontology Mismatches*. AAAI. Spring Symposium on Ontological Engineering, (1997).

66. Winkler W. E.: *The State of Record Linkage and Current Research Problems*. Statistics of Income Division, Internal Revenue Service Publication R99/04, (1999).

67. Wu, Z., Palmer, M:. *Verb Semantics and Lexical Selection.* Proceedings of the 32nd Annual Meetings of the Associations for Computational Linguistics, pages 133-138, (1994).

68. Zargayouna, H., Salotti, S.: *Mesure de Similarité Sémantique pour l'Indexation de Documents Semi-Structurés*. dans 12ème Atelier de Raisonnement à Partir de Cas, Mars (2004).

69. Zghal, S., Kamoun, K., Ben Yahia, S., Mephu, N.E., Slimani, Y. : *EDOLA : Une Nouvelle Méthode d'Alignement d'Ontologies OWL-Lite*. CORIA : 351-367, (2007).

70. Zghal, S., Ben Yahia, S., Mephu, N.E., Slimani, Y. : *SODA: an OWL-DL Based Ontology Matching System*. OM (2007).