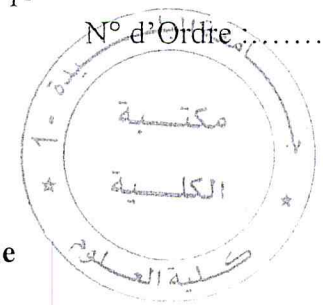
pp République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab - Blida 1-

N° d'Ordre :.....



Faculté des Sciences

Département d'Informatique

Mémoire présenté par :

Mme. LARBAOUI Nabila. Mlle. AOUAK Soumia.

En vue de l'obtention du diplôme de Master

Domaine : Mathématique et Informatique
Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet :

Le passage à l'échelle pour l'intégration des ontologies

Soutenue le : 20/09/2015

Devant le jury :

M. DERRAR Hacene	Président
M. BALA Mahfoud	Examinateur
Mme. MANCER Yasmine	Examinateur
Mme. FAREH Messaouda	Promotrice

Promotion
2014/2015

Sommaire

Liste des Tableaux

Liste des Figures

Remerciements

Nos remerciements les plus sincères à toutes les personnes qui ont contribué de prés ou de loin à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Nous remercions piètrement Allah le tous puissant de nous avoir donné le courage et la volonté de mener à terme ce présent travail.

Nous adressons nos vifs remerciements :

A notre promotrice Mme. FAREH M. pour son soutien sans failles et sa disponibilité. Ses conseils, ses suggestions de lecture, ses commentaires, ses corrections et ses qualités scientifiques ont été très précieux pour mener à bien ce travail.

Nous tenons également de remercier et exprimer nos respects aux membres de jury d'avoir accepté de juger ce travail.

« S'il n'y avait pas d'hiver, le printemps ne serait pas si agréable ; Si ne nous goûtions pas à l'adversité, la réussite ne serait pas tant appréciée »

Anne Bradstreet

Dédicace

A celui qui a toujours garni mes chemins avec force et lumière… Mon très cher père

A la plus belle perle du monde… Ma tendre mère

A mon frère Nor EL islam et ma sœur Lamia

Je leur souhaitant tout le succès… tout le bonheur

A mes beaux parents pour tout le soutien et la patience dont ils ont fait preuve.

A mon mari ***Rafik,***

Pour une sincérité si merveilleuse… jamais oubliable, en lui souhaitant tout le succès…tout le bonheur

A toute personne qui m'a aidé à franchir un horizon dans ma vie

LARBAOUI Nabila

Je dédie ce modeste travail A ma très belle et chère mère.

Au meilleur des pères.

A mon très chères frères Ali, Imad, Abd Elrahim, Ahmed et leur petits garçon.

A tous les membres de ma famille, petits et grands.

Aux personnes dont j'ai bien aimé la présence dans ce jour, je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour leurs conseils, aides, et encouragements.

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, et qui m'ont accompagnaient durant mon chemin d'études supérieures, ma famille, mes aimables amis, collègues d'étude.

A tous les gens qui m'aiment.

A tous ceux qui me sont chers.

AOUAK Soumia

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

Dans ce mémoire, nous proposons une méthode d'intégration qui aligne deux ontologies après les avoir partitionné et les divisés en blocs qui contiennent des entités ancrés. Dans ce dernier, nous verrons une implémentation et un jeu de test pour vérifier les résultats.

Mots clés : Ontologie, partitionnement, blocs, ancres, alignement.

Abstract: In real applications where ontologies are very large, the requirements in terms of execution time and memory space are the two significant factors influencing directly on the performance of an alignment algorithm. One of the solutions of the passage on the scale supposes the possibility of partitioner ontologies in blocks goshawks of the anchors before carrying out alignment, each block of the first ontology is aligned only with one block of the second ontology. The goal of partitioning is to decrease the number of entities in a block, because a system of alignment is not any more effective starting from a certain number of entities.

In this memory, we propose a method ofintegration which aligns two ontologi es after having partitioned them and divided into a block which contains entities anchored. In this last, we will see an implementation and a test set to check the results.

Keywords: Ontology, partitioning, blocks, anchors, alignment.

Introduction Générale

Le Processus d'intégration d'ontologie est un processus très délicat et contraignant. L'intégration automatique, en gardant la sémantique des informations d'ontologies a été au cœur de nombreuses recherches afin de réduire le coût qu'incombe l'intégration manuelle d'ontologies.

Au fur et à mesure des expérimentations, des méthodologies de construction d'ontologies et des outils de développement adéquat sont apparus. Emargeant des pratiques artisanales initiales, une véritable ingénierie se constitue autour des ontologies, ayant pour but leur construction et intégration mais plus largement leur gestion tout au long d'un cycle de vie.

Problématiques

Le mapping est une première étape de processus d'intégration d'ontologie, il permet de déterminer les correspondances entre les ontologies à intégrer. Les techniques d'alignement existant sont des outils dont on dispose pour trouver une solution à la problématique d'intégration.

Les techniques actuelles s'appuient en générale sur des mesures calculant la similarité de couple de concepts issus des deux ontologies. Ces mesures sont la plupart fondées sur les caractéristiques lexicales des labels des concepts et/ou des caractéristiques structurelles des ontologies, elles sont souvent testées sur des ontologies de petite taille.

Quand les ontologies sont de très grandes tailles, par exemples en agronomie et en médecine, des ontologies comportant plusieurs dizaines de milliers de concepts (AGROVOC : 28 439, NALT : 42 326, NCI : 25 652), l'efficacité des méthodes d'alignement automatiques diminue considérablement que ce soit en terme de temps d'exécution, de taille mémoire utilisée ou de précision des mappings obtenues du fait de l'augmentation de bruit.

Objectifs

L'objectif de ce projet consiste à concevoir et de réaliser un système d'intégration des ontologies de grande échelle, en effet, de plus en plus, de domaines représentent leurs connaissances à travers des ontologies réelles de grandes tailles. Elles contiennent

des milliers de concepts et de relations entre ces derniers. Par conséquent, le passage à l'échelle de processus d'intégration et d'une très grande importance, d'où il s'agit d'améliorer et optimiser le processus d'intégration pour qu'il prenne en compte de grandes ontologies.

L'organisation du mémoire

Ce mémoire est organisé en deux parties, la une première partie présente un survol des travaux les plus représentatifs pour le passage à l'échelle et l'intégration des ontologies, et la seconde partie concerne une étude conceptuelle sur l'intégration des ontologies.

La première partie :

Contient deux chapitres, elle a pour but d'avoir un aperçu global sur différentes approches de passage à l'échelle et l'intégration des ontologies de grandes volume.

Chapitre 1- Etat de l'art sur les méthodes d'Intégration des Ontologies: Ce chapitre propose une généralité sur des méthodes utilisées pour l'intégration des ontologies.

Chapitre 2- Etat de l'art sur les méthodes de passage à l'échelle des ontologies: Ce chapitre passe en revue les techniques de passage à l'échelle des ontologies volumineuses, et faire face par la suite à une étude comparative afin de pointer sur l'une d'elles.

La deuxième partie :

Après les généralités présentées dans la première partie. Cette partie est consacrée pour la conception et le développement du système.

Cette partie met l'accent sur le processus de développement de notre système d'intégration qui se présente comme une suite de phases enchaînées dans un déroulement linéaire.

Chapitre 3- La solution proposée pour l'intégration des ontologies : Ce chapitre contient la spécification et l'analyse des besoins qui présentent une compréhension des besoins et des exigences en utilisant des diagrammes UML. Il s'agit de donner des spécifications pour permettre de choisir la conception de notre système.

Pour la partie conception, elle élabore la conception générale qui présente l'architecture de notre système, et la conception détaillée qui présente les diagrammes de paquetage, de classes et de séquence ainsi que le calcul des mesures de similarité suivit des exemples simples. Ce chapitre constitue un point de départ à l'implémentation.

Chapitre 4: Implémentation : Ce chapitre est le résultat de la conception pour réaliser le système d'intégration des ontologies, il s'agit de transformer les éléments décrits lors de la conception en éléments du langage cible.

Chapitre 5: Test et Validation : cette phase est basée sur des expérimentations effectuées sur le système et des tests sur deux ontologies concrètes permettent de vérifier les résultats de l'implémentation en testant la construction du système.

La conclusion de ce mémoire synthétise les principales contributions de notre travail.

Chapitre I. Etat de l'art sur les méthodes d'Intégration des Ontologies

I.1. Introduction

Dans le but d'assurer l'interopérabilité des systèmes hétérogènes, l'utilisation d'ontologie est devenue une composante indispensable, en vue de l'émergence des termes et des ontologies utilisés dans l'informatique, la médecine, l'agronomie et la biologie ; Cependant l'utilisation d'une seule ontologie n'est pas suffisante et la construction d'ontologie à partir du zéro est un processus très long.

Dans ce contexte, nous allons présenter dans le premier chapitre les méthodes d'intégration des ontologies qui se fondent sur la réutilisation d'ontologies déjà existante tel que le mapping, la fusion, l'alignement et la transformation.

I.2. Notion d'ontologie

Le terme ontologie vient du mot grec (*ὄντος* : Onto) = L'être, (Λογια: Logia)= discours ou parler ; l'ontologie est une discipline philosophique qui peut être décrit comme la science de l'existence, ou l'étude de l'être [1] [2], on trouve plusieurs définitions tel que l'ontologie est l'étude des choses qui existent « Aristote » et la théorie des objets et de leurs relations [2], l'ontologie est une spécification explicite d'une conceptualisation. Une ontologie définit les termes et les relations de bases qui composent le vocabulaire d'un domaine, bien que les règles de combinaison des termes et les relations pour définir l'extension du vocabulaire [1].

I.3. Composants d'une ontologie

Les connaissances traduites par une ontologie sont à véhiculer à l'aide des éléments suivants [22] :

- **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.

I.4. Classification des ontologies

On trouve plusieurs travaux sur les classifications des ontologies selon les domaines de travails et de besoins [1], les auteurs Mizoguchi et ses camarades [1] ont proposés plusieurs types des ontologies tel que : les ontologies de représentation des connaissances, les méta-ontologies, les ontologies de domaine, les ontologies de tâche, les ontologies de domaine-tâche, les ontologies générales/communes, les ontologies d'application, les ontologies d'index, les ontologies interactives, etc [12].

Les ontologies de représentation des connaissances selon van Heijst et al. [3]: Regroupent les primitives de représentation utilisées afin de formaliser les connaissances selon des paradigmes de représentation des connaissances. L'exemple le plus représentatif de ce type d'ontologie est la Frame-Ontology, qui rassemble les primitives de représentation (classes, instances, cases, facettes, etc.) utilisées dans les langages à base de frames.

- ***Les méta-ontologies*** *(appelé encore les ontologies génériques ou noyau d'ontologie) [3]* : sont réutilisables dans différents domaines. L'exemple le plus représentatif serait une ontologie méréologique, qui inclurait le terme « *partie de* ».
- ***les ontologies de domaine*** selon Mizoguchi et al. et van Heijst et al. [3] : sont réutilisables dans un domaine donné. Elles fournissent le vocabulaire des concepts d'un domaine (par ex., scalpel, scanner dans un domaine médical) et les relations entre ces

derniers, les activités de ce domaine (par ex., anesthésié, accoucher) ainsi que les théories et les principes de base de ce domaine.

- ***les ontologies de tâche*** selon Mizoguchi et al. [3] : fournissent un vocabulaire systématisé des termes utilisés pour résoudre les problèmes associés à des tâches qui peuvent appartenir ou non à un même domaine. Ces ontologies fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles incluent des noms génériques (par ex., plan, objectif, contrainte), des verbes génériques (par ex., assigner, classer, sélectionner), des adjectifs génériques (par ex., assigné) et d'autres mots qui relèvent de l'établissement d'échéances.
- ***les ontologies de domaine-tâche*** [3]: organisation de métadonnée ou rôle des connaissances dans la réalisation d'une tache.
- ***les ontologies générales/communes*** [3] : incluent le vocabulaire lié aux objets, aux événements, au temps, à l'espace, à la causalité, au comportement, à la fonction, etc.

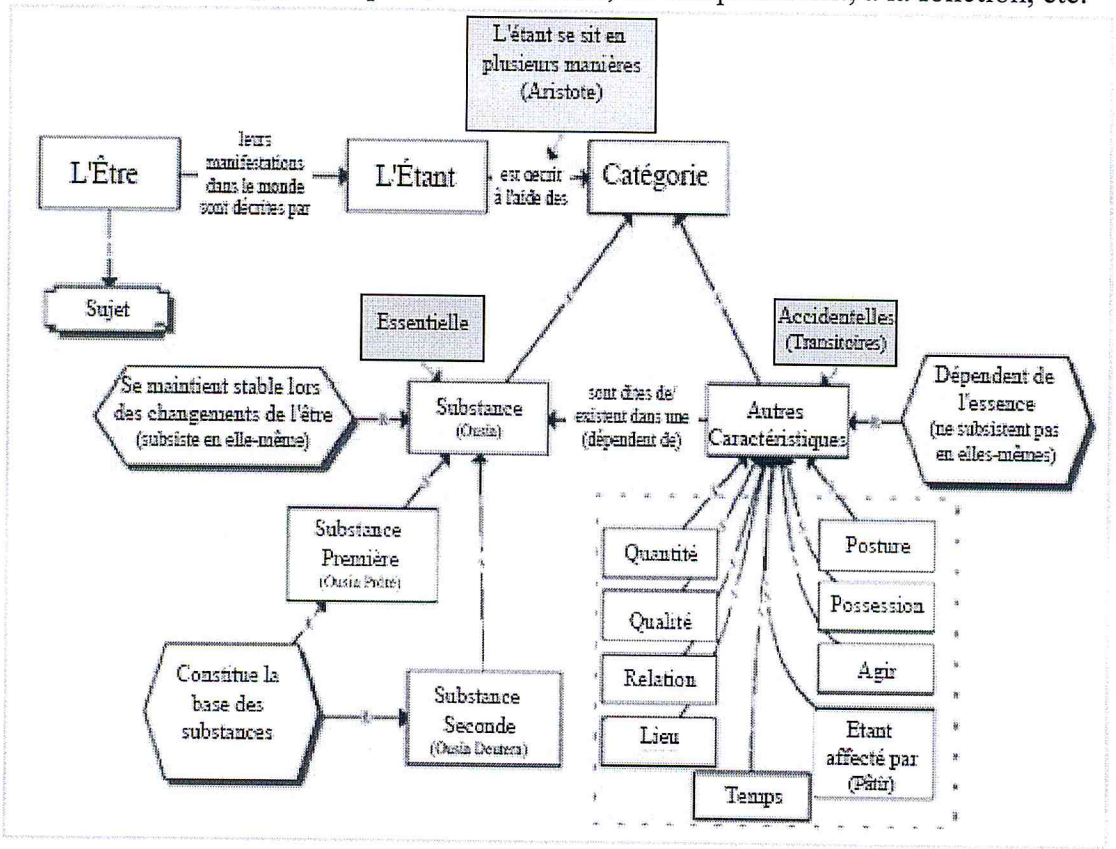


Fig.1- L'ontologie de l'être selon Aristote [6]

I.5. Langages 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 ayan 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.
- **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.
- **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.
- **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.

I.6. La Conception des ontologies

La conception d'ontologies est une tâche difficile nécessitant la mise en place de procédés élaborés afin d'extraire la connaissance d'un domaine, manipulable par les systèmes informatiques et interprétable par les êtres humains. Deux types de conception existent : la conception entièrement manuelle et la conception reposant sur des apprentissages [12]. Maedche et Staab [12] ont distingués différents types d'approches en fonction du support sur lequel elles se basent : à partir de textes, de dictionnaires, de bases des connaissances, de schémas semi-structurés et de schémas relationnels.

I.7. Les Outils de conception des ontologies

Différents outils de conception ont été développés, Chacun d'entre eux présente des fonctionnalités différentes et a permis l'élaboration de nombreuses ontologies

Text-To-Onto Développée à l'Institut AIFB de l'Université de Karlsruhe, est une application d'extraction d'ontologies à partir de corpus ou de documents Web qui permet également la réutilisation d'ontologies existantes. Elle est intégrée à la plate-forme logicielle KAON qui permet l'édition et la maintenance d'ontologies [12].

OntoBuilder Développée au Technion d'Haifa, permet de bâtir une ontologie à partir de ressources Web, elle autorise aussi la fusion d'ontologies extraites de différents sites Web.

TOVE est un projet de Gruninger et Fox, l'ontologie de domaine est construite (manuellement) à partir des scénarios d'entreprises pour lesquels elle sera utilisée. Elle est spécialisée sur la spécification d'ontologies pour les entreprises.

METHONTOLOGY de Fernandez et al. est conçue pour être appliquée dans des cadres plus généraux.

KACTUS (modelling Knowledge About Complex Technical systems for multiple USe) de Schreiber et al. est comme la METHONTOLOGY, elle conçue pour être appliquée dans des cadres plus généraux ; cette méthodologie vise à réutiliser des ontologies existantes et propose des mécanismes permettant cette réutilisation.

TERMINAE d'Aussenac et al. propose une approche pour sélectionner les concepts, leurs propriétés, les relations et leur regroupement. Cette méthodologie a été développée

dans le laboratoire et est une composante de la plate forme RFIEC[1] [12]. Elle repose sur l'utilisation d'outils de traitement automatique des langues analysant les termes de textes et les relations lexicales. Les termes sont regroupés suivant leur contexte et facilitent la création de concepts et de relations sémantiques. Les concepts et relations sont ensuite formalisés dans un modèle.

I.8. Hétérogénéités des ontologies

Les ontologies vise à constituer une représentation réelle qui puisse être acceptée par une communauté, et que les langages de représentation des ontologies et les méthodes de conceptualisation sont différents, ce qui engendre des hétérogénéités entre les ontologies ce qui limitent les possibilités d'interopérabilité entre eux, et gênent l'intégration aussi [1].

Selon Klein [1], l'hétérogénéité se manifeste sur deux niveaux : au niveau langage ou au niveau ontologique. Bouquet et al. [1] ont ajouté un autre niveau d'hétérogénéité à savoir l'hétérogénéité pragmatique.

a) Les hétérogénéités au niveau du langage

Se manifestent quand le langage de spécification diffère d'une ontologie à l'autre : les classes, les relations ne sont pas définies de la même manière. Les hétérogénéités de ce niveau sont situées sur le plan de la syntaxe, de la représentation logique et de l'expressivité.

• **Syntaxe** : la syntaxe définit la structure des représentations, les différences dans ce plan concernent le formalisme plutôt que le contenu. La solution de ce type est simple, un mécanisme de réécriture des ontologies dans le même langage.

• **Représentation logique** : les notions logiques sont exprimées de manière différente, par exemple dans un langage le fait que deux concepts A et B sont disjoints s'exprime par A *intersecte* (not B)= true, et que dans un autre langage cette même relation s'exprime par A∩B=0. Ce problème est facilement résolu par des règles de translation d'une représentation logique à l'autre.

• **Expressivité** : implique que certains langages sont capables d'exprimer des notions alors que les autres ne peuvent pas, par exemple les listes, la négation, etc.

[1] RFIEC est une plateforme regroupant un ensemble de résultats associé aux compétences locales d'équipes travaillant autour de l'analyse et la représentation de textes (outils, méthodologies, corpus, ressources linguistiques)

b) Les hétérogénéités au niveau ontologique

Visser et al. [6] Ont distingué dans ce niveau, les hétérogénéités conceptuelles et celles terminologiques.

• **Niveau terminologique** : ce type d'hétérogénéité concerne toutes les différences liées au processus de nomination des entités (classes, propriétés etc.). Il est représenté très souvent par la présence des synonymes, des homonymes, multilinguismes, les abréviations, etc. La difficulté major de ce niveau est la présence des homonymes qui n'est pas facile à résoudre.

• **Niveau Conceptuel** : est appelé aussi hétérogénéité sémantique, c'est la différence dans l'interprétation du domaine qui a comme conséquence différents concepts ou différents relations entre concepts, Benerecitti et al. [6] ont identifie trois sortes d'hétérogénéités dans ce niveau :

- ***Couverture*** : qui désigne le fait que les ontologies couvrent différents portions de l'univers du discours.
- ***Granularité*** : qui signifie que les ontologies peuvent décrire les objets de la réalité a des degrés de détails différents.
- ***Perspective*** : qui signifie que les ontologies couvrent des points de vue différents.

c) Les hétérogénéités au niveau pragmatique

Selon visser et al. [6] le niveau pragmatique est le niveau le plus complexe. Il concerne les hétérogénéités d'interprétation d'une ontologie qui peuvent survenir lorsque des individus ou des communautés différentes interprètent différemment l'ontologie selon différents contextes .

I.9. Méthodes d'intégration des ontologies

Il existe différentes approches pour intégrer des ontologies. Elles s'étendent des approches faiblement couplées jusqu'aux approches fortement couplées. Dans le cadre du projet européen INTEROP, on a retenu les principales approches d'intégration suivantes : le mapping, l'alignement, la transformation et la fusion d'ontologies. Chacune de ces approches est présentée ci-dessous [6].

9.1 Mapping d'ontologies

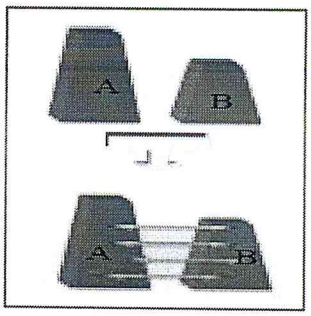


Fig.2- Principe du mapping d'ontologies [6].

Le mapping d'ontologies repose sur la définition de relations de correspondance entre les entités de deux ontologies qui présentent une similitude.

Habituellement la fonction est de type 1 à 1, mais on peut définir des relations fondées sur la notion de mesure de similarité. Une caractéristique importante de cette approche est qu'elle ne modifie pas les ontologies impliquées et qu'elle produit en sortie un ensemble de correspondances.

9.2 Alignement d'ontologies

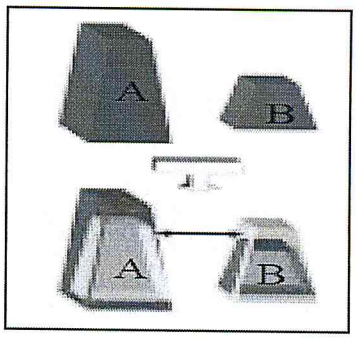


Fig.3- Principe d'allignement d'ontologies [6].

L'alignement d'ontologies est considéré comme le processus qui permet d'amener deux ou plusieurs ontologies hétérogènes à un "accord mutuel" en les rendant ainsi consistantes et mutuellement cohérentes. L'alignement d'ontologies nécessite la transformation des ontologies impliquées en procédant à l'élimination des entités non pertinentes et au rajout des entités manquantes.

Ces structures peuvent être des ontologies (au sens large), des schémas XML ou des bases de données. Les liens sémantiques comprennent les relations : d'équivalence (=), de généralisation/spécialisation (⊇, ⊆), de chevauchement (Π) ou encore d'incompatibilité (≠). L'évaluation de la véracité de ces liens peut être booléenne ou par le biais d'autres mesures telles que : les probabilités, les mesures symboliques, les mesures de similarité.

Voici quelques précisions terminologiques :

9.2.1 Dimensions de l'alignement

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

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

Notre présent travail, se situe dans le cadre de l'alignement des ontologies « ontology matching » qui diffère à plusieurs égards, de l'alignement des schémas « schema matching », mais néanmoins reste assez proche, comme le montre le tableau ci-dessous:

Schema Matching	Ontology Matching
Différences	
Souvent, Les données des schémas ne sont pas porteuses de sémantique explicite.	Les ontologies sont des systèmes logiques porteurs de définitions (à travers les axiomes)
Les schémas (relationnels, par exemple) ne fournissent pas de généralisation	Les définitions des ontologies sont un ensemble d'axiomes logiques aptes à être généralisés par rapport à un contexte donné.
Les ontologies sont plus riches (le nombre de primitives est plus grand et elles sont plus complexes) que les schémas, par exemple OWL permet la définition de nouvelles classes à travers les unions et les intersections	
Points communs	
Les schémas et les ontologies comprennent des vocabulaires de termes qui décrivent le domaine d'intérêt	
Les schémas et les ontologies sont porteurs de définitions des termes du vocabulaire utilisé	
Les ontologies peuvent être considérées comme des schémas de bases de connaissances	

Tab.01- « Schema matching » vs. « Ontology matching » [13]

Shvaiko et Euzenat concluent ce tableau comparatif en disant ceci [5]:

« Techniques developed for both problems are of a mutual benefit »

« Les techniques développées pour les deux problèmes sont d'un bénéfice mutuel »

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

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

Ceci peut être représenté schématiquement de la manière suivante :

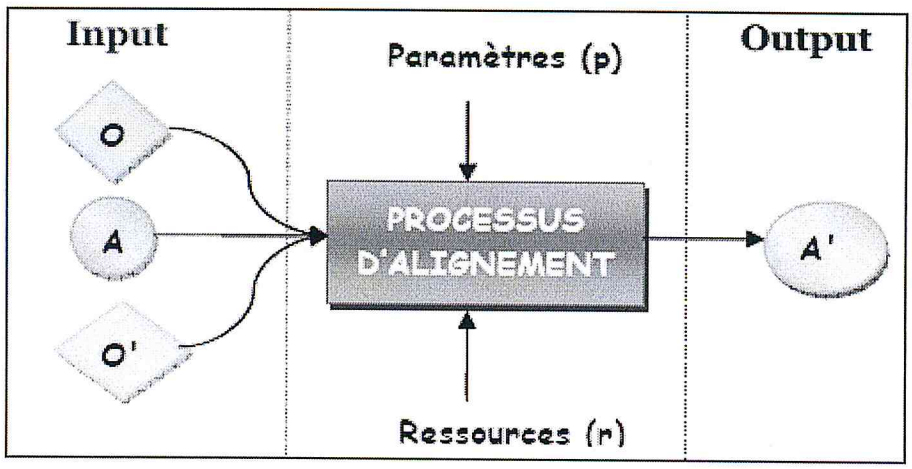


Fig.4- Les trois dimensions de l'alignement [13].

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

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

id : identifiant unique d'un mapping

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

e' : une entité, a aligner, appartenant a O'

r : la relation qui relie **e** a **e'** (=, ≠, ⊇, ⊓, ⊆)

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

L'output est caractérisé par :

a) La multiplicité

(Contraintes sur les relations entre les entités des deux ontologies).

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

1 : une et une seule relation

? : De 0 à 1 relation

+ : de 1 a plusieurs relations

***** : de 0 à plusieurs relations

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

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

Voici quelques exemples sur les configurations de multiplicité entre deux ontologies, constituée chacune de trois entités :

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

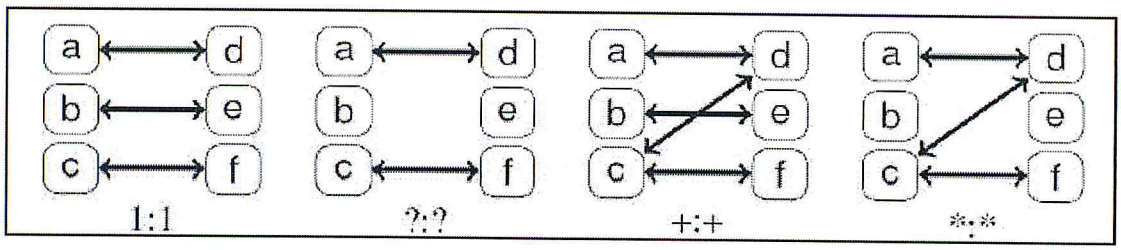


Fig.5- Exemples de configurations de multiplicité entre 2 ontologies [13].

b) Le niveau de l'alignement

- ***Le niveau (0) :*** Les alignements concernent des entités identifiées par des URIs.
- ***Le niveau (1) :*** Les alignements de paires d'entités sont remplacés par des paires d'ensembles d'entités.
- ***Le niveau (2) :*** des correspondances plus générales, que les niveaux précédents, peuvent être utilisés.

9.3 Transformation d'ontologies

La transformation d'ontologies est un processus qui permet de changer la structure des ontologies en conservant au maximum sa sémantique. Selon que la transformation se fait sans pertes (*lossless*) ou avec perte d'informations (*lossy*).

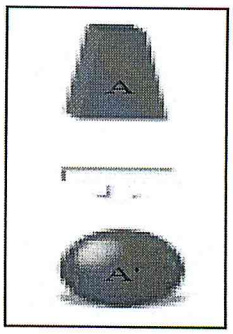


Fig.6- Principe de la transformation d'ontologies [6].

9.4 Fusion d'ontologies

La fusion d'ontologies est une approche qui permet de construire à partir de deux ou plusieurs ontologies existantes une nouvelle ontologie. La nouvelle ontologie intégrée s'obtient généralement en combinant les ontologies initiales et quelques concepts supplémentaires nécessaires pour la combinaison.

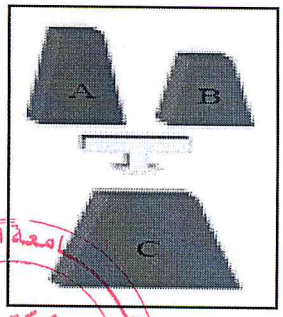


Fig.7- Principe de fusion d'ontologies [6].

Ce type d'approche est généralement utilisé, dans le cadre d'intégration de données, pour obtenir une ontologie globale qui sert d'interface pour un certain nombre d'ontologies locales. La principale limite de cette architecture tient au fait que l'on a besoin de développer, et encore plus important, on a besoin de s'entendre sur une ontologie globale, chose qui n'est toujours pas si évidente dans les environnements fortement évolutifs.

I.10. Quelques Outils de Calcul sémantique

De nombreux travaux basés sur des correspondances sémantiques entre les pairs ont été développés. Différents procédés sont utilisés pour générer ces correspondances de manière plus ou moins automatique [14].

Piazza Permet aussi bien l'échange de données relationnelles, XML que RDF. Il est basé sur une architecture Pair à Pair pure. En présence de différents schémas et de différentes représentations, les pairs intéressés par l'échange de données définissent des correspondances sémantiques entre eux, deux à deux ou entre petits groupes de pairs.

SomeWhere Dans SomeWhere, aucun utilisateur n'impose aux autres sa propre ontologie, car le système permet de créer des mises en correspondance entre différentes ontologies. Un pair se connectant un réseau construit les mappings entre sa propre ontologie et les ontologies des pairs servant de point d'entrée dans le réseau. Pour traiter une requête, l'utilisateur doit choisir le pair par lequel sa requête sera initiée dans le réseau. Le routage des requêtes est guidé vers les pairs dont les mappings sont pertinents. Cette pertinence est établie selon un algorithme distribué de raisonnement en logique des propositions (FOL).

GLUE Le système GLUE utilise une approche par machine learning pour classifier les concepts d'une ontologie afin de les mettre de manière semi automatique en correspondance avec les concepts définis dans les ontologies distantes. Cependant, bien que tout repose sur la phase d'apprentissage, cette approche suppose qu'un nombre important d'utilisateurs collabore pour définir les mappings sémantiques entre les ontologies.

PeerDB Permet le partage de données relationnelles distribuées sans partage de schéma. Il combine les propriétés des systèmes multi-agents avec celles des systèmes Pair-à-Pair. Chaque pair fournit une base de données relationnelle décrite grâce à des méta-données (mots-clés). La reformulation des requêtes est faite par des agents grâce à une mise en correspondance des méta-données associées aux schémas. L'approche PeerDB présente la faiblesse d'autoriser des correspondances entre mots clés pouvant aboutir à de fausses reformulations.

I.11. Calcul de Similarité

Rahm et Bernstein [17] proposent la classification suivante des différentes techniques d'alignement, la figure ci-dessous synthétise les mesures de similarité:

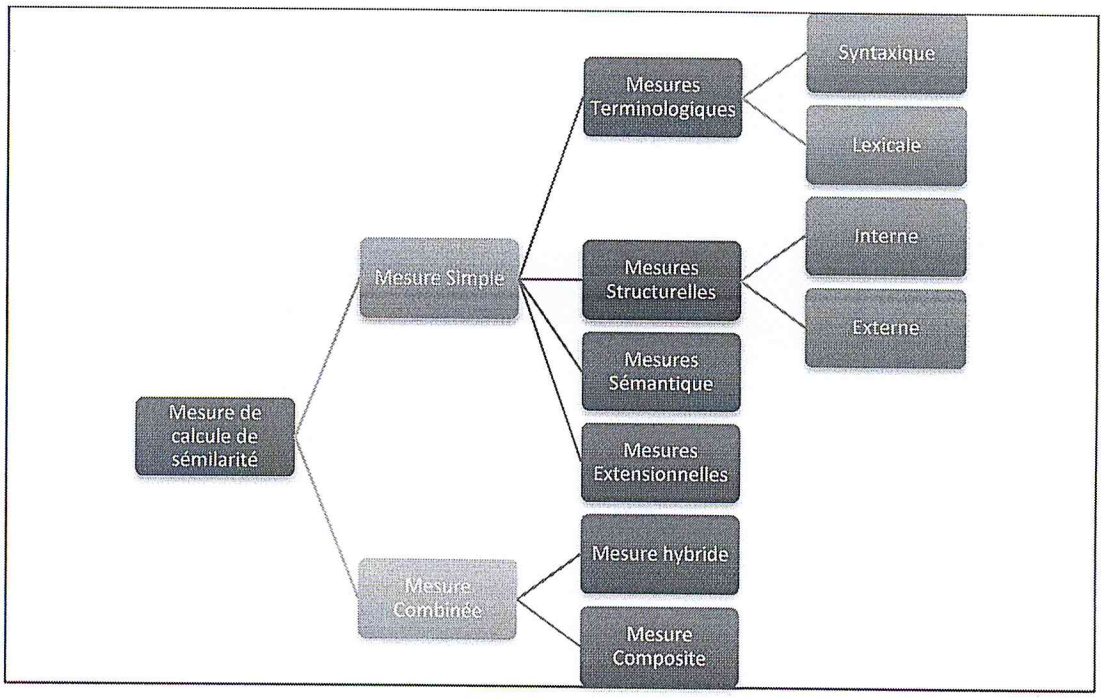


Fig. 8. Les Mesures de calcul de Similarités.

11.1 Les techniques terminologique

Ces techniques reposent principalement sur des comparaisons de textes, elles s'appliquent sur les noms, les commentaires et les propriétés des concepts afin de trouver ceux qui sont similaires; Le problème de ces comparaisons est l'existence des synonymes et des homonymes, il n'est pas possible d'assumer que deux entités sont similaires si elles portent le même nom, ou qu'elles sont différentes si elles possèdent des noms différents. De plus, l'utilisation des langages différents pour représenter la même entités, ainsi que la variation syntaxique du même terme qui est due lors de l'utilisation des abréviations, des préfixe et des suffixes,…

Il existe des méthodes de comparaison des termes selon leur considération :

- Méthodes basées sur les chaînes de caractères (texte) : comme leur nom l'indique, ces méthodes considèrent l'entité comme une séquence de lettres. Les résultats obtenus par ces méthodes sont utiles si les concepteurs utilisent des chaines de caractères similaires pour définir la même entité, en revanche, s'il y a des synonymes avec des structures différentes ces méthodes donnent une mauvaise estimation de la similarité.
- Méthodes basées sur les langages : les entités ici sont considérées comme un texte (des mots) dans un langage donnée, elles font appel généralement aux techniques du traitement automatique du langage (TAL) pour extraire les termes les plus représentatifs à partir d'un texte, elles utilisent généralement des ressources externe tel que les dictionnaires, les thésaurus,…

L'inconvénient de ces méthodes est que les ressources externes ne sont pas toujours disponibles pour certains langages.

11.2 Les techniques structurelles

Elles se focalisent sur la comparaison des structures des entités à mapper, en fonction de la nature interne ou externe des structures à comparer, on peut distinguer :

- Méthodes structurelles internes : qui utilise la comparaison de structure interne d'une entité (ex : la comparaison des attributs, des noms, des types des

attributs,..) indépendamment des autres entités, ces méthodes sont généralement combinées avec celles des techniques terminologiques.

Ce type de méthodes est très important pour le mapping du fait qu'il peut éliminer les correspondances incompatibles, cependant, il n'offre pas vraiment beaucoup d'information sur les entités à comparer, plusieurs entités très hétérogènes peuvent être décrites par le même structure interne et des entités très proches peuvent avoir des structures différentes.

- Méthodes structurelles externes : qui utilisent la comparaison externe en mettant par exemple en jeux la disposition des entités dans leur hiérarchie, le voisinage,…

Le mapping qui utilise ces méthodes est très performant parce qu'il prend en compte toutes les relations entre les entités ce qui nécessite l'utilisation d'autres méthodes terminologiques.

11.3 Les techniques extensionnelles

Elles se basent sur la comparaison des instances des entités des ontologies à mapper, en fonction de l'intersection des extensions de deux entités, on peut distinguer :

- Méthodes de comparaison d'extensions communs : elles sont utilisées lorsqu'il y a un ensemble des instances communes entre les ontologies, elles considèrent que deux entités sont similaires si leur intersection se réduit à l'une des entités(en terme d'instances).

Le problème de ces méthodes est la capacité d'engendrer des fautes, un petit ensemble de données incorrecte peut conduire à des résultats erronés, de plus si deux entités ne partagent pas les même instances, ces méthodes vont donner un 0 comme similarité sans tenir compte à la similarité entre les instances.

- Méthodes d'identification des instances : elles sont appliquées quand il n'existe pas un ensemble commun des instances, il est possible d'identifier qu'une instance d'un ensemble donné correspond à une instance d'un autre ensemble par le calcul de similarité entre les instances.

Ces méthodes sont utilisables dont le cas où l'on sait que les instances sont identiques (ex : lorsqu'on intègre deux ontologies de la même entreprise), mais quand il s'agit des entreprises différentes où des ontologies qui n'ont aucune relation entre eux, ces méthodes ne seront jamais applicables.

Généralement, le savoir de la partie extension des ontologies est inestimable pour le mapping car elle est indépendante de toute conceptualisation, cependant si l'ensemble des instances n'est pas disponible, ces méthodes ne pourraient pas s'appliquer, dans tel cas, les autres techniques sont plus adéquates.

11.4 Les techniques sémantiques

Elles sont très souvent des techniques qui se basent sur des modèles théoriques et se focalisent sur des techniques déductives exploitent très souvent la logique de description (test de subsumption) ou le calcul des prédicats.

- Méthodes basées sur des ontologies externes : si deux ontologies qui ne possèdent pas d'un terrain commun pour la comparaison vont être mappées, ces méthodes lui permettent d'utiliser une ontologie intermédiaire qui définit un contexte commun ou des connaissances de bases pour les deux ontologies.
- Méthodes déductives : ces méthodes ne sont pas très performantes seules pour une tâche inductive comme le mapping. En revanche, elles ont une grande valeur pour le mapping une fois que les correspondances sont générées, du fait qu'elles ont la capacité de détecter les inconsistances dans les mapping.

I.12. Conclusion

Dans ce chapitre nous avons vu quelques notions et définitions ; tels que la notion d'ontologie, ses concepts, et les outils utilisés pour la conception des ontologies. Puis nous avons vu quelques méthodes d'intégrations telles que le mapping, l'alignement, la transformation et la fusion d'ontologie ; et en fin nous avons entamé une partie importante dans l'intégration des ontologies et qui concerne le calcul de la sémantique ce qui va être bien expliqué dans le chapitre qui suit: Etat de l'Art sur les Méthodes de Passage à l'Echelle des Ontologies

Chapitre II. Etat de l'art sur les méthodes de passage à l'échelle des ontologies

II.1. Introduction :

Dans les domaines d'applications réelles, les ontologies devenant de plus en plus volumineuses, se sont intéressés au problème de leur partitionnement.

La décomposition d'une ontologie en sous-blocs (ou îlot) indépendants les uns des autres autour des ancres, de façon à faciliter en toute généralité différentes opérations sur les ontologies comme la maintenance, la visualisation, la validation ou le raisonnement ont pour objectif l'alignement d'ontologies.

La tâche d'alignement d'ontologies (recherche de mapping, appariements ou mises en correspondance entre concepts) est donc particulièrement importante dans les systèmes d'intégration puisqu'elle autorise la prise en compte conjointe de ressources décrites par des ontologies différentes.

Une solution possible pour résoudre ce problème est d'essayer de limiter la taille des ensembles de concepts en entrée des algorithmes d'alignement, et pour cela de partitionner les deux ontologies à aligner en plusieurs blocs, afin de n'avoir à traiter que des blocs de taille raisonnable.

Dans ce contexte, nous allons présenter dans le 2^{ieme} chapitre quelques méthodes de passage à l'échelle d'ontologie volumineuses, puis une étude comparative entre ces approches.

II.2. Définition de passage à l'échelle :

Le traitement de grands volumes d'informations est souvent désigné par l'expression « passage à l'échelle ». Plus précisément, le passage à l'échelle d'une technique ou d'un algorithme désigne sa capacité à traiter des volumes considérables d'informations tout en conservant une complexité du même ordre de grandeur réelle, c'est-à-dire chronométrée, que celle induite par le traitement des volumes antérieurs moins importants [24].

Nous avons vu qu'il est indispensable d'utiliser la technique de partitionnement dans le passage à l'échelle pour réduire la taille de l'ontologie. Par la suite nous présentons le partitionnement.

II.3. Les différentes étapes de partitionnement

Avant d'entrer dans le détail des méthodes de passage à l'échelle il est mieux de citer les différentes étapes de partitionnement qui sont issus de l'algorithme PBM, [16].

3.1 Déterminer les ancres

Dans l'étape de classification des concepts, l'algorithme tentera de regrouper les entités autour des ancres. S. kasri, F. Benchikha [16] ont dit que deux entités sont des ancres si et seulement si elles sont équivalentes.

3.2 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. 11). Puis il s'agit d'exécuter un algorithme de clustering inspiré de celui de Falcon-AO [16].

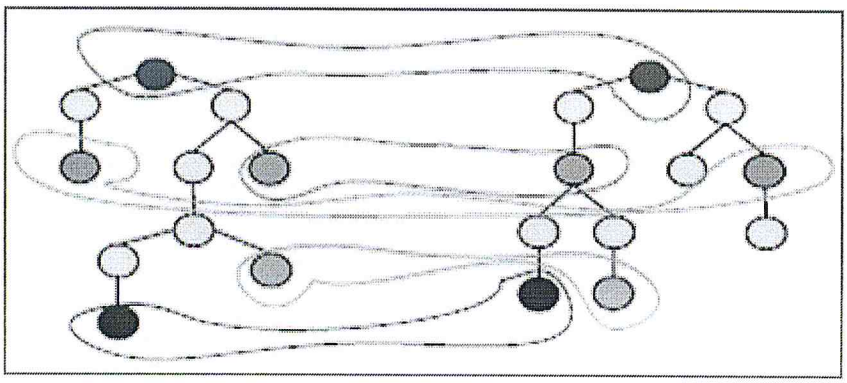


Fig. 09. L'entrée de l'algorithme (les couples d'ancres)[22]

3.3 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 (Fig. 10) en des clusters qui ne contiennent que les ancres d'une même ontologie.

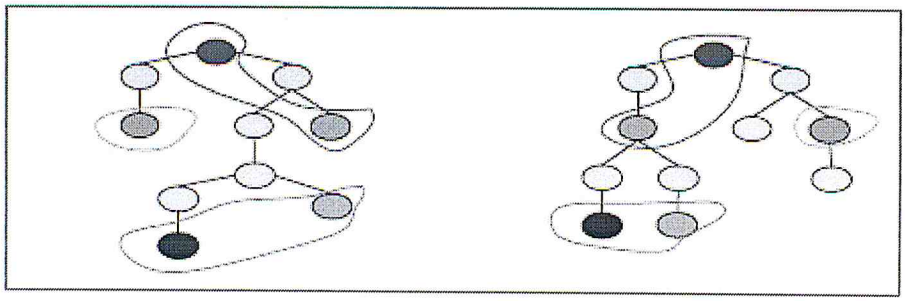


Fig. 10. Les clusters d'ancres de chaque ontologie. [22]

II.4. Partitionnement autour des ancres

Dans cette étape, les concepts sont classifiés progressivement autour des ancres.

➢ **L'algorithme**

La classification consiste à agréger progressivement dans chaque itération les concepts autour des ancres selon leur ressemblance en deux étapes :

1. *Etape d'affectation* : pour chaque concept, on détermine le cluster d'ancres auquel on doit l'affecter (le cluster le plus proche c.à.d. où le couplage est le plus grand).
2. *Etape de représentation* : pour chaque cluster défini, on recalcul les nouveaux couplages.

L'algorithme fournit en sortie un ensemble de blocs où chaque bloc contient une ou plusieurs ancres comme présenté en (Fig. 11).

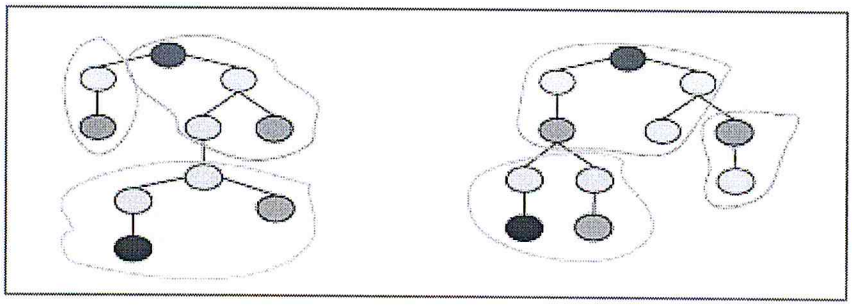


Fig. 11. Les blocs générés après la classification. [22]

Dans la phase d'alignement, chaque bloc de la première ontologie ne s'aligne qu'avec un seul bloc de la deuxième ontologie (Fig. 12).

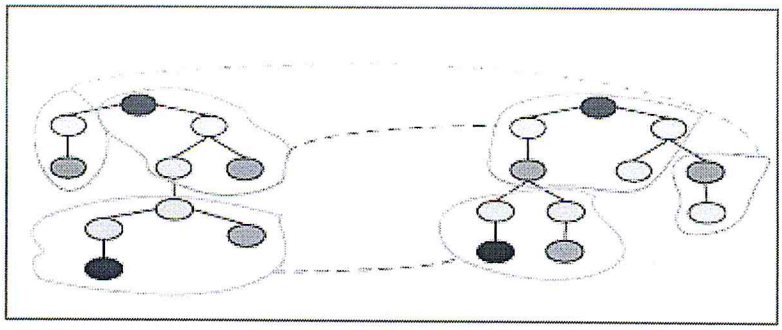


Fig. 12. Les blocs alignés après la classification. [22]

II.5. Quelque systèmes existent pour le passage à l'échelle

Pour prendre en compte au plus tôt l'objectif d'alignement, les méthodes choisis par [3] vont s'appuyer sur deux données : d'une part les couples de concepts issus des deux ontologies qui ont exactement le même label et peuvent être reliés par une relation d'équivalence et d'autre part, l'éventuelle dissymétrie structurelle des 2 ontologies à aligner [3].

Même sur des ontologies de grande taille, il est possible d'identifier avec une mesure de similarité stricte et peu coûteuse à calculer, les concepts dont un des labels est identique à l'un des labels d'un concept de l'autre ontologie. Comme dans **FALCON,** ils appelleront ces couples de concepts des **Ancres** mais ils les utiliseront dès la construction des partitions [3].

La dissymétrie structurelle des 2 ontologies est exploitée pour ordonner leur partitionnement : si l'une des deux ontologies est mieux structurée que l'autre, elle sera plus facile à décomposer en blocs ayant une forte cohésion interne et sa décomposition pourra servir de guide à celle de l'autre ontologie. Dans ce qui suit, l'ontologie la plus structurée sera appelée la cible, O_T et la moins structurée, la source, O_S [3].

5.1 Méthode FALCON

La méthode proposée dans Falcon [3] consiste à partitionner chaque ontologie en blocs en utilisant la méthode de Clustering Rock [3] , puis à mesurer la proximité de chacun des blocs d'une ontologie avec chaque bloc de l'autre ontologie de façon à n'effectuer l'alignement qu'entre les concepts des paires de blocs les plus proches.

Pour effectuer la partition, alors que Rock considère que les liens entre les concepts ont tous la même valeur, Falcon introduit la notion de liens pondérés qui s'appuie sur deux mesures de similarité entre concepts, une mesure linguistique et une mesure structurelle.

5.1.1 Mesures de similarité

a) Similarité lexicale

1. ***Sim$_l$ (ci, cj)*** : Soient di (resp. dj) la chaine de caractère correspondant à la description du concept ci (resp. cj) et *comm (di, dj)* (resp. *diff(di, dj))* le nombre de caractères communs (resp. différents) dans les chaines di et dj [3] .

   La similarité lexicale entre deux concepts ci et cj , *Sim$_L$(ci, cj),* se calcul comme suit :

$$Sim_l(ci, cj) = comm(di, dj) - diff(di, dj) + Winkler(di, dj)$$

   Où le terme *Winkler (di, dj)* est ajouté pour améliorer le résultat en utilisant la méthode de Winkler. Les concepts ci et cj sont jugés similaires si *Sim$_L$(ci, cj)* > 0.65.

2. ***Les Synsets***: La composante atomique sur laquelle repose le système entier est le synset (synonym set), un groupe de mots interchangeables, dénotant un sens ou un usage particulier. Nous le retrouvons dans ***WordNet*** qui est une base de données lexicale développée par des linguistes du laboratoire des sciences cognitives de l'université de *Princeton* depuis une vingtaine d'année. Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise [23].

b) Similarité structurelle

Soient *ci, cj* deux concepts d'une même ontologie *O*, c_{ij} leur plus petit ancêtre commun et *depthOf(c)* la distance en nombre d'arcs entre le concept c et la racine de O. Falcon mesure la similarité structurelle *aff$_s$(ci, cj)* en utilisant la mesure de W_U et Palmer (ref) qui se calcul comme suit :

$$aff_s(c_i, c_j) = \frac{2 * depthOf(c_{ij})}{depthOf(c_i) + depthOf(c_j)}$$

Le calcul de similarité structurelle entre les concepts d'une ontologie de grande taille peut prendre beaucoup de temps. En considérant que seuls les concepts de profondeurs adjacentes auront des similarités élevées, Falcon ne compare que les concepts qui satisfont la relation suivante [13].

$$|depthOf(c_i) - depthOf(c_j)| \leq 1$$

c) Les liens pondérés

Le calcul du lien pondéré entre deux concepts, link(ci, cj), s'effectue comme suit[13].

$$link(c_i, c_j) = \begin{cases} aff(c_i, c_j) & if\ aff(c_i, c_j) > \epsilon_1 \\ 0 & otherwise \end{cases}$$

$$aff(c_i, c_j) = \alpha \cdot affs(c_i, c_j) + (1 - \alpha) \cdot Sim_L(c_i, c_j)$$

Où Ɛ1 est un seuil donné tel que Ɛ1∈ [0, 1] et α ∈ [0, 1] permet à l'utilisateur de faire varier le poids relatif des mesures de similarité.

5.1.2 Algorithme de Partitionnement

L'algorithme permettant à FALCON de partitionner une ontologie en blocs s'appuie sur deux notions essentielles : *la cohésion* au sein d'un bloc et *le couplage* entre deux blocs distincts. La cohésion est une mesure du poids de l'ensemble des liens reliant les concepts appartenant à un même bloc et le couplage, du poids de l'ensemble des liens reliant les concepts de deux blocs différents. Les formules de calcul de la cohésion et du couplage s'expriment à partir d'une même mesure appelée *goodness* [13].

$$goodness(B_i, B_j) = \frac{\sum_{c_i \in B_i, c_j \in B_j} link(c_i, c_j)}{sizeOf(B_i) \cdot sizeOf(B_j)}$$

Cohésion(Bi) = goodness(Bi,Bi), Couplage(Bi,Bj) = goodness(Bi,Bj) où Bi ≠Bj .

Etant donnée une ontologie O, l'algorithme prend en entrée l'ensemble B des n blocs à partitionner, où chaque bloc est réduit au départ à un unique concept de O, et le nombre k de blocs souhaité en sortie. Il initialise tout d'abord la valeur de cohésion de chaque bloc ainsi que les valeurs de couplage. A chaque itération, l'algorithme choisit le

bloc qui a la cohésion maximale et le bloc qui a la valeur de couplage maximale avec ce premier bloc. Il remplace ces deux blocs par celui résultant de leur fusion et met à jour les valeurs de couplage de tous les blocs avec ce nouveau bloc. L'algorithme s'arrête quand il a atteint le nombre de blocs souhaité ou quand tous les blocs fusionnables ont atteint une taille limite [3].

5.1.3 Identification des paires de blocs à aligner

Une fois réalisé de façon séparée le partitionnement des deux ontologies, l'évaluation de la proximité des blocs s'effectue en s'appuyant sur des *ancres*, i.e. des appariements préalablement connus entre les termes des deux ontologies, définis par des techniques de comparaison de chaînes de caractères ou par un expert. Plus deux blocs contiennent d'ancres communes, plus ils sont jugés proches.

Soient k (resp. k') le nombre de blocs générés par le partitionnement d'une ontologie O (resp. O') et Bi (resp. B'j) un de ces blocs. Soit la fonction $anchors(B_u, B'_v)$ qui calcul le nombre d'ancres partagées par les deux blocs B_u et B'_v et soit $\sum^{k'}_{v=1}$ anchors(B_i,B'_v) le nombre d'ancres contenues par un bloc Bi. La relation de Proximité entre deux blocs Bi et B'j est définie comme suit :

$$Proximity(B_i, B'_j) = \frac{2.anchors(B_i, B'_j)}{\sum_{u=1}^{k} anchors(B_u, B'_j) + \sum_{v=1}^{k'} anchors(B_i, B'_v)}$$

Les paires de blocs alignées sont toutes les paires dont la proximité est supérieure à un seuil donné $e_2 \in [0, 1]$. Un bloc pourra donc être aligné avec plusieurs blocs de l'autre ontologie ou avec aucun suivant la valeur choisie pour ce seuil.

Cette méthode permet à FALCON de décomposer des ontologies volumineuses, mais la décomposition est faite a priori, sans prendre en compte l'objectif d'alignement, en s'appliquant sur chaque ontologie indépendamment l'une de l'autre. Le partitionnement étant fait à l'aveugle, certaines ancres pourront ne pas se trouver dans des blocs finalement alignés et l'alignement résultant ne comprendra pas forcément tous les appariements souhaités. Enfin, le calcul des blocs pertinents à aligner est coûteux (en temps de traitement) [3].

Malgré ces critiques, l'algorithme de décomposition de FALCON selon F. Hamdi et al. [3] est le plus adapté à la tâche d'alignement des algorithmes de partitionnement existants puisqu'il permet de contrôler la taille maximale des blocs construits.

Les deux méthodes que F. Hamdi et al. [3] ont proposées le réutilisent en modifiant sa façon de générer les blocs. L'idée est de prendre en considération au plus tôt lors du partitionnement, toutes les données relatives à l'alignement existant entre les concepts des deux ontologies et d'essayer de faire, au moins dans la deuxième méthode, du co-clustering.

5.2 Méthode PAP (Partition, Anchor, Partition)

La première méthode consiste à commencer par décomposer la cible O_T , 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. La première méthode comprend donc quatre étapes en plus du calcul des ancres :

Partitionner la cible O_T en plusieurs blocs B_{Ti} Le partitionnement est effectué conformément à l'algorithme FALCON [3].

Identifier les centres CB_{Si} des futurs blocs de O_S Les centres de O_S sont déterminés en se basant sur deux critères : les couples d'ancres identifiés entre O_S et O_T , et les blocs B_{Ti} construits à partir de l'ontologie cible O_T .

Soit la fonction *Ancre(E,E')*, dont les arguments E et E' peuvent être chacun, soit une ontologie, soit un bloc, et qui retourne l'ensemble des concepts de E qui ont le même label qu'un concept de E'. Pour chaque bloc B_{Ti} construit à l'étape précédente, les centres des futurs blocs correspondants de O_S sont calculés comme suit :

$$CB_{Si} = Ancre(O_S, B_{Ti}) \qquad [3]$$

Partitionner la source O_S autour des centres CB_{Si} Après l'identification des centres des futurs blocs de O_S ; l'algorithme FALCON doit être appliqué avec la différence suivante, Au lieu d'introduire en entrées l'ensemble des m concepts de l'ontologie comme m blocs réduits chacun à un unique concept, on introduit les n centres identifiés à l'étape précédente, comme autant de blocs distincts mais regroupant plusieurs concepts, puis les autres concepts de O_S qui n'ont pas d'équivalents dans O_T , chacun

dans un bloc individuel. La cohésion des blocs représentant les centres de O_S est initialisée avec la valeur maximale [3].

Identifier les paires de blocs à aligner Chacun des blocs B_{Si} construits à partir d'un centre n'est aligné qu'avec le bloc B_{Ti} correspondant. L'algorithme peut mener à la constitution de blocs B_{Sj} indépendants des centres, i.e. ne contenant pas d'ancres et qui, dans l'état courant de notre implémentation, ne sont pas pris en compte dans le processus d'appariement.
Le traitement de ces blocs sans ancres est une des perspectives de ce travail, encore à l'étude [3].

5.3 Méthode APP (Anchor, Partition, Partition)

L'idée de cette méthode est de partitionner les deux ontologies en même temps, c.à.d. de faire du co-clustering. D'après Hu, W et al. ces ontologies ne peuvent pas être traitées en parallèle du fait de leur grande taille. Pour simuler le parallélisme, ils ont proposé de partitionner l'ontologie cible en favorisant la fusion des blocs partageant des ancres avec la source, et de partitionner 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. Ainsi, et d'après les résultats qu'ont trouvé Hu, W et al. ce partitionnement, contrairement à celui de FALCON ou à celui implémenté dans la méthode 1, est orienté alignement pour les deux ontologies à partitionner [3].

Cette deuxième méthode comprend trois étapes :
Déterminer les blocs de O_T : Pour construire les blocs de la cible O_T , Hu, W et al. ont utiliser l'algorithme FALCON en modifiant la définition de la mesure de goodness pour prendre en compte les relations d'équivalence entre les deux ontologies. ils lui ajoutaient un coefficient qui représente la proportion d'ancres partagées qui sont présentés dans un bloc B_j de O_T . Plus un bloc contient d'ancres, plus ce coefficient augmente sa cohésion ou sa valeur de couplage à d'autres blocs. De ce fait, au cours de la génération des blocs, le choix du bloc qui a la valeur maximale de cohésion ou de

couplage ne dépend pas seulement des relations des concepts à l'intérieur ou à l'extérieur des blocs de O_T , mais aussi des ancres partagées avec O_S [3].

Soient $\alpha \in [0, 1]$, B_i et B_j 2 blocs de O_T , $|Ancre(Bj, O_S)|$ représentant le nombre d'ancres présentes dans B_j et $|Ancre(O_T, O_S)|$, le nombre d'ancres total, l'équation de goodness devient :

$$goodness(B_i, B_j) = \alpha \left(\frac{\sum_{c_i \in B_i, c_j \in B_j} link(c_i, c_j)}{sizeOf(B_i) \cdot sizeOf(B_j)} \right) + (1 - \alpha) \frac{|Ancre(B_j, O_S)|}{|Ancre(O_T, O_S)|}$$

Déterminer les blocs de O_S : Là aussi nous modifions la mesure de *goodness* pour qu'elle prenne en compte à la fois les valeurs des liens entre les concepts de O_S, les ancres partagées entre les deux ontologies et les blocs construits pour O_T . Soit le bloc Bi de O_S ayant la valeur de cohésion maximale, soit B_k le bloc de O_T qui partage le plus d'ancres avec B_i, le nouveau calcul de *goodness* favorisera la fusion de B_i avec le bloc B_j qui contient le plus d'ancre partagées avec B_k, de façon à regrouper dans un même bloc de la source, les ancres partagées avec un même bloc de la cible [3].

Soient $\alpha \in [0, 1]$, B_i et B_j , 2 blocs distincts de O_S, B_k le bloc de O_T qui partage le plus d'ancres avec B_i, l'équation de *goodness* devient :

$$goodness(B_i, B_j) = \alpha \left(\frac{\sum_{c_i \in B_i, c_j \in B_j} link(c_i, c_j)}{sizeOf(B_i) \cdot sizeOf(B_j)} \right) + (1 - \alpha) \frac{|Ancre(B_j, B_k)|}{|Ancre(O_T, O_S)|}$$

Identification des paires de blocs à aligner : 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 [3].

II.6. Analyse comparative des différentes approches de passage à l'échelle

Nous présentons dans cette partie une classification des approches et des techniques décrites dans les sections précédentes. Nous intéressons plus particulièrement à ces méthodes que nous présentons brièvement dans ce que suit.

Méthodes / Critères	Méthode PAP	Méthode APP	Falcon
Partitionnement Et alignement	Indépendant	Parallèle	Indépendant
Type	1.n	1.n	1.n
Les outils utilisés	TaxoPart (2008)	TaxoPart (2008)	Falcon-AO (2005)
Les techniques d'alignement utilisé	Conceptuelles, lexicales et structurelles (*Mesure de similarité stricte*)	Conceptuelles, lexicales et structurelles (*Mesure de goodness*)	Conceptuelles, lexicales et structurelles (*Mesure de Wu et Palmer, Mesure de goodness*)
Types de processus d'Alignement	Automatique	Automatique	Automatique
Types d'Opération	Fusion	- Co-Clustering - Fusion	- Clustering - Fusion
Ontologies de test	- CibleBDTopo - Source BDCarto - AGROVOC - NALT	- CibleBDTopo - Source BDCarto - AGROVOC - NALT	- CibleBDTopo - Source BDCarto - AGROVOC - NALT
Type de relation d'Alignement	Equivalence	Equivalence	_
Alignement	Orienté Alignement	Orienté Alignement	A priori/ A posteriori
Coût d'Ontologie volumineuse	Peu coûteuses	Peu coûteuses	coûteuses
Algorithme de partitionnement (similarité structurelle et linguistique)	LMO et GMO (la cohésion et le Couplage)	LMO et GMO (la cohésion et le Couplage)	LMO et GMO (la cohésion et le Couplage)

Tab.02- Tableau comparative des différentes approches.

Commentaire

Le tableau ci-dessus nous montre les différents critères de chaque méthode ; d'après les valeurs obtenues, nous remarquons que le partitionnement des ontologies se passe d'une manière particulière dans la méthode APP, où l'alignement des deux ontologies utilise la technique du parallélisme qui est difficile à illustré dans les deux autres méthodes, où chaque une d'elle traite les bloques des ontologies indépendamment l'une de l'autre.

Les méthodes PAP et APP utilisent l'algorithme de Falcon pour décomposer la cible de l'ontologie la plus structurée (O_T) mais dès la construction des ancres et pas avant. Nous remarquons aussi que la méthode Falcon décompose les Ontologies volumineuses soit à priori sinon à posteriori sans prendre en compte l'Alignement, c.à.d. le partitionnement peut se fait à l'aveugle contrairement aux autres méthodes qui sont orienté Alignement.

Dans les versions actuelles, Falcon-AO et TaxoPart intègrent des algorithmes de partitionnement adaptés au contexte d'alignement des ontologies volumineuses, Le système Falcon-AO est divisé en trois phases[16] : Calcul de proximité lexical, Calcul de proximité structurelle et le Partitionnement tout en basant sur les deux propriétés (la Cohésion et le Couplage : mesure de similarité entre les entités appartenant à deux clusters différents), qui sont également utilisées par les deux autres méthodes (PBM et APP) , sauf que le système TaxoPart prend en compte au plus tôt l'objectif de l'alignement.

II.7. Conclusion

Ce chapitre conclus les différentes méthodes sur le passage à l'échelle, nous avons également présenté un tableau de différences entre ces méthodes.

L'alignement d'ontologie est une tâche importante dans les systèmes d'intégration. Avec l'apparition de très grandes ontologies dans des domaines comme la Médecine ou l'Agronomie, les techniques d'alignement, qui mettent souvent en œuvre des calculs complexes, se trouvent face à un défi : passer à l'échelle.

Les techniques actuelles d'alignement s'appuient en général sur des mesures calculant la similarité de couples de concepts issus des deux ontologies.

2.2 La Méthode PAP

Critères	Avantage	Inconvénient
Partitionnement	Décomposer l'ontologie en une ontologie cible et une ontologie source afin de faciliter les calculs.	Cette décomposition n'aura pas lieu si les concepts n'assurent pas la relation d'équivalence ou on ne trouve pas la dissymétrie structurelle des deux ontologies. La décomposition se fait indépendamment.
Technique d'alignement	L'appariement se fait par paire de bloc ce qui assure l'exactitude des résultats.	Encore à l'étude.
Types d'opération	/	/
Alignement	/	/
Ontologie Volumineuses	Peu coûteuse.	/

Tab .04. Avantages et Inconvénients de la méthode PAP.

2.3 La Méthode APP

Critères	Avantage	Inconvénient
Partitionnement	Le partitionnement des deux ontologies s'effectuera en même temps.	Difficulté de traitement de deux ontologies de grande taille.
Technique d'alignement	Possibilité de modifier la mesure de *goodness pour assurer la relation d'équivalence.*	/
Types d'opération	Possibilité de fusionner les blocs d'ontologie avec d'autres d'une autre ontologie en même temps.	/
Alignement	L'alignement se fait entre blocs des ontologies en parallèle.	L'alignement se fait qu'avec un et un seul bloc.
Ontologie Volumineuses	Peu coûteuse.	/

Tab .05. Avantages et Inconvénients de la méthode APP.

2.4 Positionnement de notre travail par rapport à l'état de l'art

Les approches de partitionnement et d'alignement décrites précédemment ont pour objectif l'amélioration des résultats de l'alignement en choisissant les meilleurs critères et spécificités des ontologies, par exemple, les techniques d'alignement, les types d'opération, le coût de traitement des ontologies volumineuses, et les paramètres les plus appropriés tels que les coefficients utilisés dans les formules. Deux types d'approches ont été distinguées, celles qui prennent en compte les spécificités de partitionnement de l'ontologie au sein du processus d'alignement et celles qui prennent en compte les spécificités de l'ontologie via des traitements effectués sur des blocs.

L'objectif de notre travail est de relever le challenge du passage à l'échelle des méthodes d'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. Notre approche consiste à partitionner chaque ontologie en blocs autour des ancres en utilisant les algorithmes de clustering. Les blocs générés peuvent être utilisés dans un algorithme d'alignement car les blocs qui contiennent les entités similaires (ancrées) devraient l'être aussi.

Pour ce faire, notre méthode inspirée par le deuxième type d'approche (Méthode APP) qui parait plus intéressante dans notre domaine de recherche ; au lieu d'appliquer l'algorithme indépendamment sur chaque ontologie, nous prenons en compte dès que possible dans le processus de partitionnement le contexte de la tâche d'alignement.

III.3. Les démarche de la Méthode APP

3.1 Le partitionnement

Cette méthode comprend trois étapes :

1. Partitionner la première ontologie O_T en utilisant l'algorithme PBM mais en prenant en compte l'ensemble des Ancres lors de la génération des blocs.
2. Partitionner la deuxième ontologie O_S de la même manière mais en constituant des blocs contenant des ancres qui appartiennent à un même bloc de l'ontologie O_T.
3. Aligner les paires de blocs partageant le plus d'ancres.

3.1.1 Etape 1 : Déterminer les blocs de O_T

Pour déterminer les blocs de l'ontologie cible O_T, nous utilisons l'algorithme BPM en modifiant la mesure de *goodness* pour prendre en compte les liens entre les deux ontologies, Nous lui ajoutons un coefficient qui représente la proportion d'ancres présente dans un bloc de O_T relativement en nombre d'ancres total identifiées avec l'ontologie O_S.

De ce fait, au cours de la génération des blocs, le choix du bloc qui a la valeur maximale de cohésion ou de couplage ne dépend pas seulement des relations des concepts à l'intérieur ou à l'extérieur des blocs d'une même ontologie, mais aussi des relations d'équivalences identifiées avec l'autre ontologie.

3.1.2 Etape 2 : Déterminer les blocs de OS

Les blocs de l'ontologie source O_S sont générés en se basant sur les poids des liens entre les concepts de O_S, les relations d'équivalences entre les deux ontologies et les blocs de l'ontologie O_T .

Si nous considérons un bloc B_i dans O_S avec une valeur de cohésion maximale, le calcul de *goodness* pour trouver le bloc ayant la valeur de couplage maximale avec B_i se base non seulement sur les relations structurelles dans O_S, mais aussi sur les relations d'équivalences avec le bloc de O_T qui a le maximum d'équivalents avec B_i.

Lors des opérations de fusion, nous conservons pour chaque bloc de O_S l'ensemble des ancres appartenant au bloc, ce qui facilitera lors de la recherche des blocs à aligner, l'identification des blocs partageant le plus d'ancres.

3.1.3 Etape3 : Identification des blocs à aligner

La conservation des ancres introduites dans chaque bloc facilite la phase de calcul des paires de blocs partageant le plus d'ancres, un bloc de O_S ne s'alignant qu'avec un seul bloc de O_T. Une fois identifiés les blocs de O_S devant être alignés avec le même bloc de O_T nous pouvons éventuellement les fusionner si leur taille le permet, afin de diminuer le nombre de combinaisons à faire lors du processus d'alignement.

3.2 Les techniques d'alignement structurelle et syntaxique

Nous allons maintenant décrire les mesures de similarité utilisées dans notre algorithme de partitionnement.

3.2.1 Présentation des mesures de similarité utilisées dans l'algorithme de partitionnement

a) **Similarité lexicale** : Pour calculr la similarité lexicale entre les entités nous utilisant la mesure de Jaro-Winkler 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 calculr la similarité lexicale entre les entités, sans prétraitement, dans une mesure à moindre coût.

b) **Similarité structurelle** : 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.

3.2.2 Présentation de l'algorithme de partitionnement

Après avoir passé en revue les mesures de similarité, nous allons maintenant détailler notre algorithme de partitionnement [22].

Soient O_T et O_S deux ontologies à aligner, notre algorithme de partitionnement orienté alignement comprend les étapes suivantes :

- Déterminer les couples d'ancres entre O_T et O_S en utilisant une mesure de similarité lexicale.
- Classifier ces couples d'ancres en k clusters préliminaires.
- Identifier les clusters d'ancres de chaque ontologie (l'ensemble des ancres appartenant à chaque ontologie).

- Classifier les concepts de chaque ontologie autour des clusters d'ancres.

3.2.2.1 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 pour les déterminer.

3.2.2.2 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. 09. Puis il s'agit d'exécuter un algorithme de clustering inspiré de celui de Falcon[2].

L'algorithme. La classification consiste à agréger progressivement les clusters d'ancres selon leur ressemblance [22]. 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. La première différence entre notre algorithme et celui de Falcon 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.

[2] Voir l'annexe

III.4. Conclusion

Dans ce chapitre nous avons présenté notre solution pour le problème de passage à l'échelle ; nous avons préludé une petite étude comparative entre les méthodes utilisées tel que la méthode FLCON et celle de PAP puis nous nous somme positionné sur la méthode APP suite à des raisons citées ci-dessus. En revanche, nous avons entamés en second lieu de concevoir les étapes pour le partitionnement des ontologies en blocs en utilisant des mesures d'alignement à cet effet.

Chapitre IV. Conception du système d'Intégration d'Ontologie

IV.1. Définition des besoins

Les phases d'expression et d'analyse du besoin permettent de décrire les fonctionnalités du logiciel et les contraintes sous lesquelles celui-ci doit être réalisé ; c'est une définition et formulation des exigences bien définies.

1.1 Diagrammes utilisés

a) Diagramme de Cas d'Utilisation : Interactions entre le système et les utilisateurs (et autres systèmes externes). Il aide dans la visualisation des exigences / besoins ; Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service.

b) Diagramme de séquence : Interactions entre des objets pour lesquelles l'ordre des interactions est important; les connexions entre objets sont importantes aussi.

1.1.1 Diagrammes de cas d'utilisation

✓ *Premier niveau de décomposition (cas d'utilisation globale)*

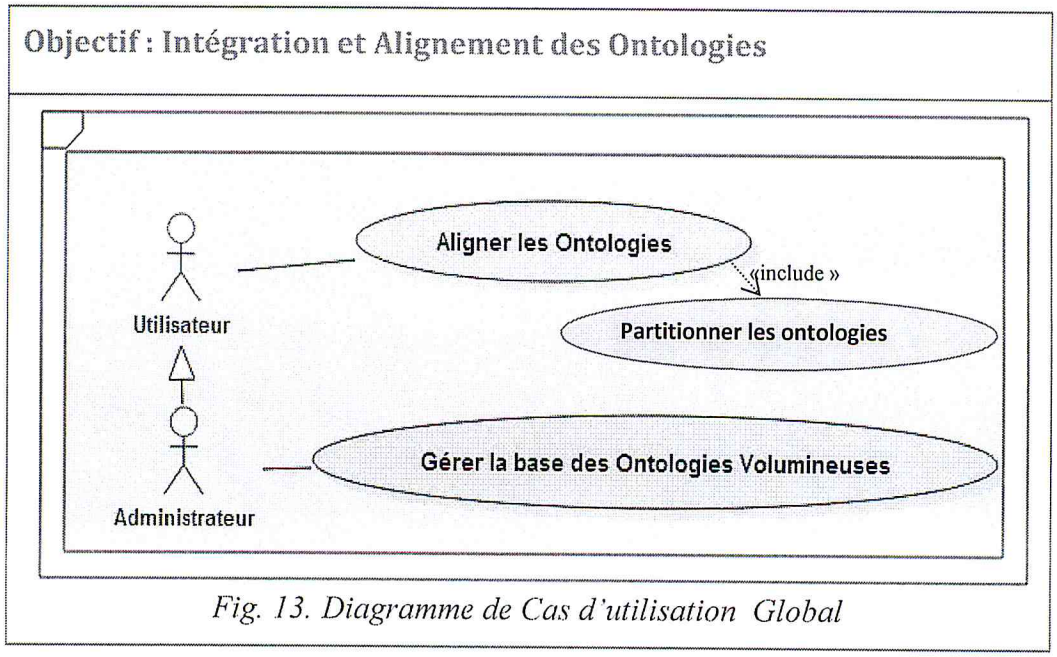


Fig. 13. Diagramme de Cas d'utilisation Global

✓ *Description textuelle (cas d'utilisation global « Intégrer et Aligner les Ontologies)*

Acteur	Utilisateur	Administrateur
Tâche	Visualiser les résultats de l'Alignement.	Gérer les ontologies et visualiser les résultats d'alignement.
Pré Condition	L'utilisateur doit sélectionner deux ontologies et choisir la mesure désirée.	Etre un utilisateur privilégié et possédant tous les droits d'accès.
Post-Condition	Visualiser les résultats d'Alignement	Avoir de nouvelles listes des Ontologies et des résultats d'Alignement.
Exception	Aucun traitement n'est effectué.	Aucun traitement n'est effectué.

Tab .06. Description textuelle de ca d'utilisation globale.

✓ *Deuxième niveau de décomposition*

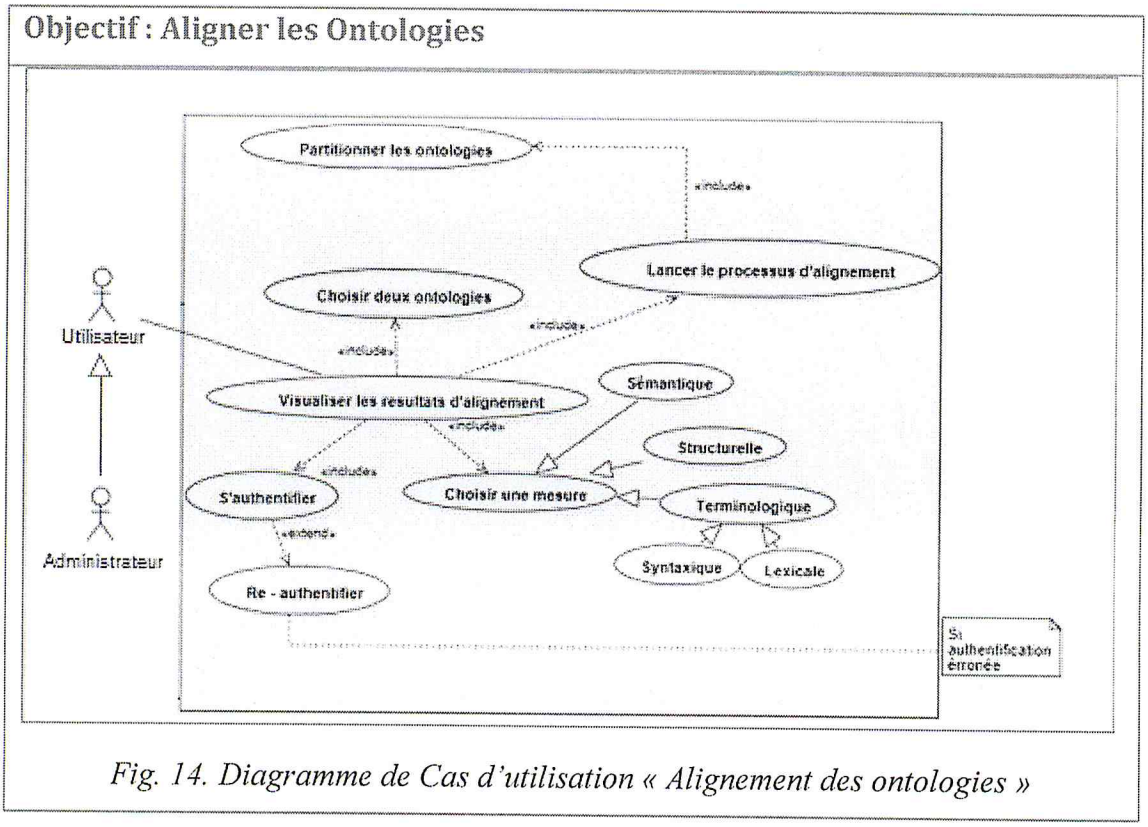


Fig. 14. Diagramme de Cas d'utilisation « Alignement des ontologies »

Acteur	Aligneur (un des deux acteurs)
Description	L'alignement s'effectue par l'administrateur et l'utilisateur aussi qui ont les droits d'accès adéquats.
Tâche	Visualiser les résultats de l'Alignement.
Pré - Condition	L'aligneur doit s'authentifié et figuré dans la base de données des droits d'accès. L'aligneur doit sélectionner deux ontologies . Il doit choisir une des mesures proposées pour le calcul des similarités. Il doit lancer le processus d'Alignement.
Post-Condition	Visualiser les résultats d'Alignement des Ontologies qui a choisit.
Exception	Un message d'erreur qui s'affiche si l'utilisateur saisissait son nom d'utilisateur ou le mot de passe erroné. Aucun traitement n'est effectué s'il n'a pas choisit deux Ontologies ou une des deux, ou encore il n'a pas précisé une mesure adéquate.

Tab .07. Description textuelle de ca d'utilisation « Alignement des ontologies ».

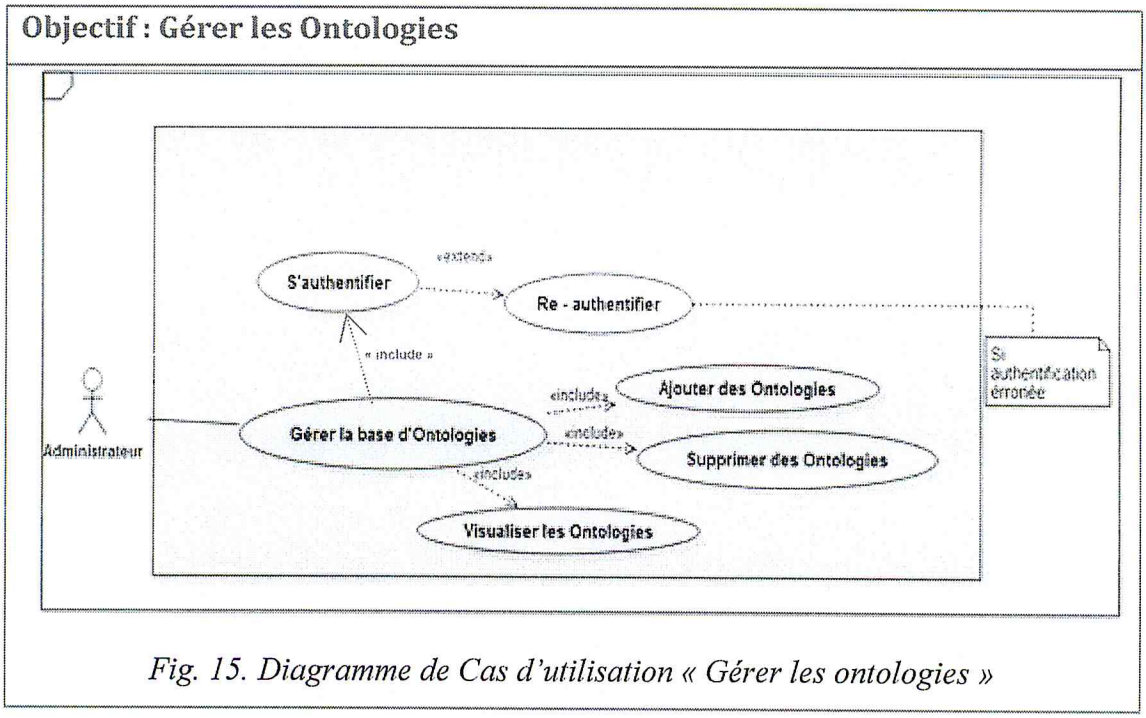


Fig. 15. Diagramme de Cas d'utilisation « Gérer les ontologies »

Acteur	Administrateur
Tâche	Gérer les Ontologies
Pré - Condition	L'Administrateur doit être privilégié par un droit d'accès adéquat. L'Administrateur peut ajouter, supprimer et visualiser les ontologies.
Post-Condition	Avoir des listes des Ontologies mises à jour.
Exception	Un message d'erreur qui s'affiche si l'administrateur saisissait son nom d'utilisateur ou le mot de passe erroné.

Tab .08. Description textuelle de cas d'utilisation « Gérer les Ontologies ».

1.1.2 Diagrammes de Séquence

Scénario	*Diagramme de séquence « **Authentification** »*
1	L'utilisateur se connecte au système.
2	Le Système demande le Nom d'utilisateur et le mot de passe.
3	L'Utilisateur saisit les informations nécessaires et valide.
4	Le Système vérifier l'identité de l'utilisateur dans la base de données.
5	Le système reçoit l'approbation de l'existence.
6	Le Système autorise l'utilisateur privilégié d'accéder à l'interface qui lui correspond.
7	Le système reçoit la contestation d'identité de l'utilisateur.
8	Le Système Affiche un message d'erreur.

Tab .09. Description textuelle de l'interaction « Authentification ».

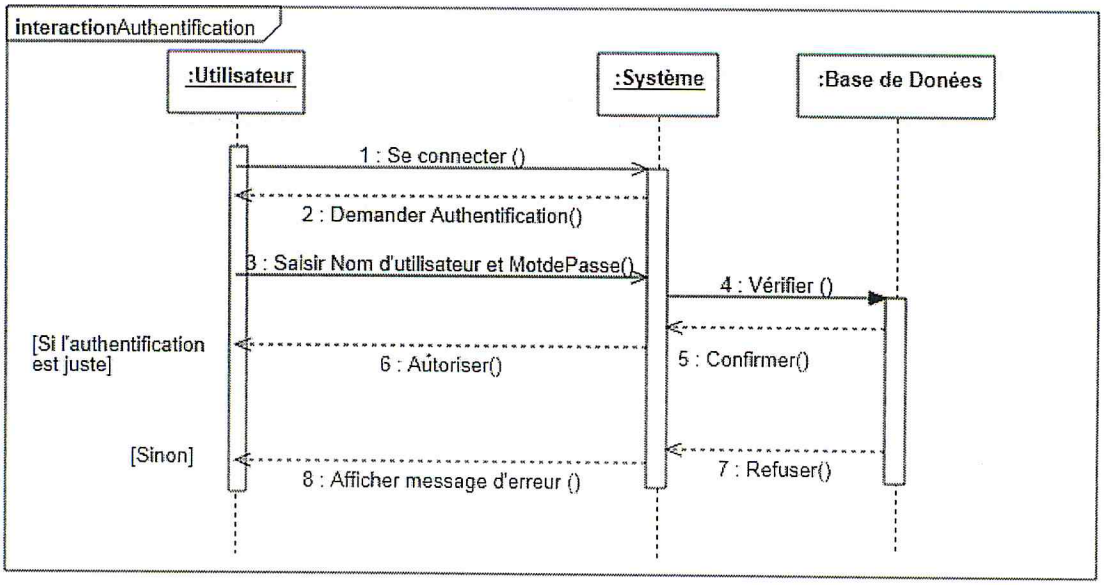


Fig. 16. Diagramme de Séquence « Authentification ».

Scénario	*Diagramme de séquence « **Partitionnement** »*
1	- L'utilisateur doit choisir et sélectionner deux ontologies.
2	- Le Système vérifier le choix d'utilisateur s'il existe dans la base des ontologies.
3	- Le système reçoit la réponse de la recherche.
4	- Le Système affiche la liste des ontologies à l'utilisateur.
5	- L'utilisateur sélectionne l'ontologie cible.
6	- Le Système affiche son choix.
7	- L'utilisateur sélectionne l'ontologie source.
8	- Le Système affiche son choix.
9	- L'utilisateur lancer le processus de partitionnement d'ontologie.
10	- Le système faire le partitionnement en plusieurs blocs (mesure de choix).
11	- Enregistrer le résultat de partitionnement
12	- Le système affiche le résultat de partitionnement à l'utilisateur

Tab .10. Description textuelle de l'interaction « Partitionnement ».

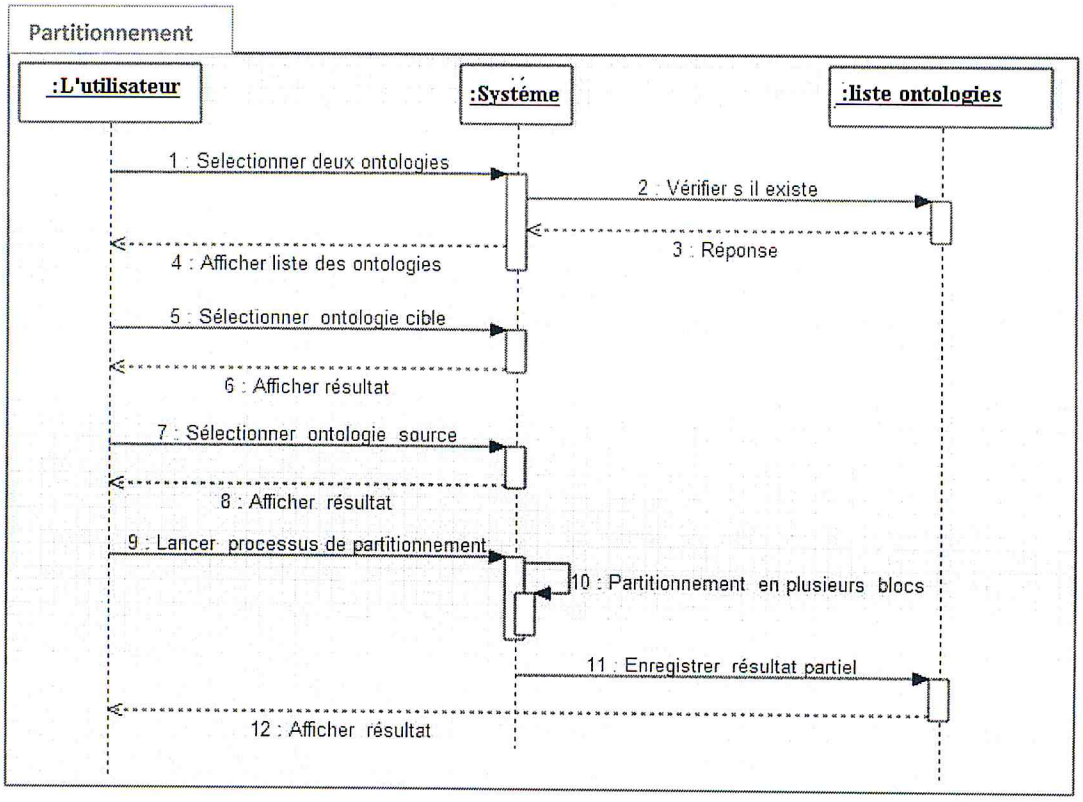


Fig. 17. Diagramme de Séquence « Partitionnement ».

Scénario	*Diagramme de séquence « **Alignement** »*
1	L'utilisateur doit choisir et sélectionner deux Ontologies.
2	Le Système cherche la disponibilité de choix de l'utilisateur dans la base d'ontologies.
3	Le Système reçoit le résultat de la recherche de disponibilité.
4	Le Système affiche la liste des ontologies mise à jour.
5	L'utilisateur doit choisir une Ontologie « Cible ».
6	Le Système Affiche son choix et le valide.
7	L'utilisateur doit choisir une Ontologie « Source ».
8	Le Système Affiche son choix et le valide.
9	L'utilisateur doit choisir une mesure de similarité.
10	L'utilisateur lance le processus d'Alignement.
11	Le système applique l'algorithme de partitionnement.
12	Le système calcul par la suite la correspondance en appliquant l'algorithme de la mesure choisit par l'utilisateur.

13	Le Système affiche les résultats obtenus.

Tab .11. Description textuelle de l'interaction « Alignement ».

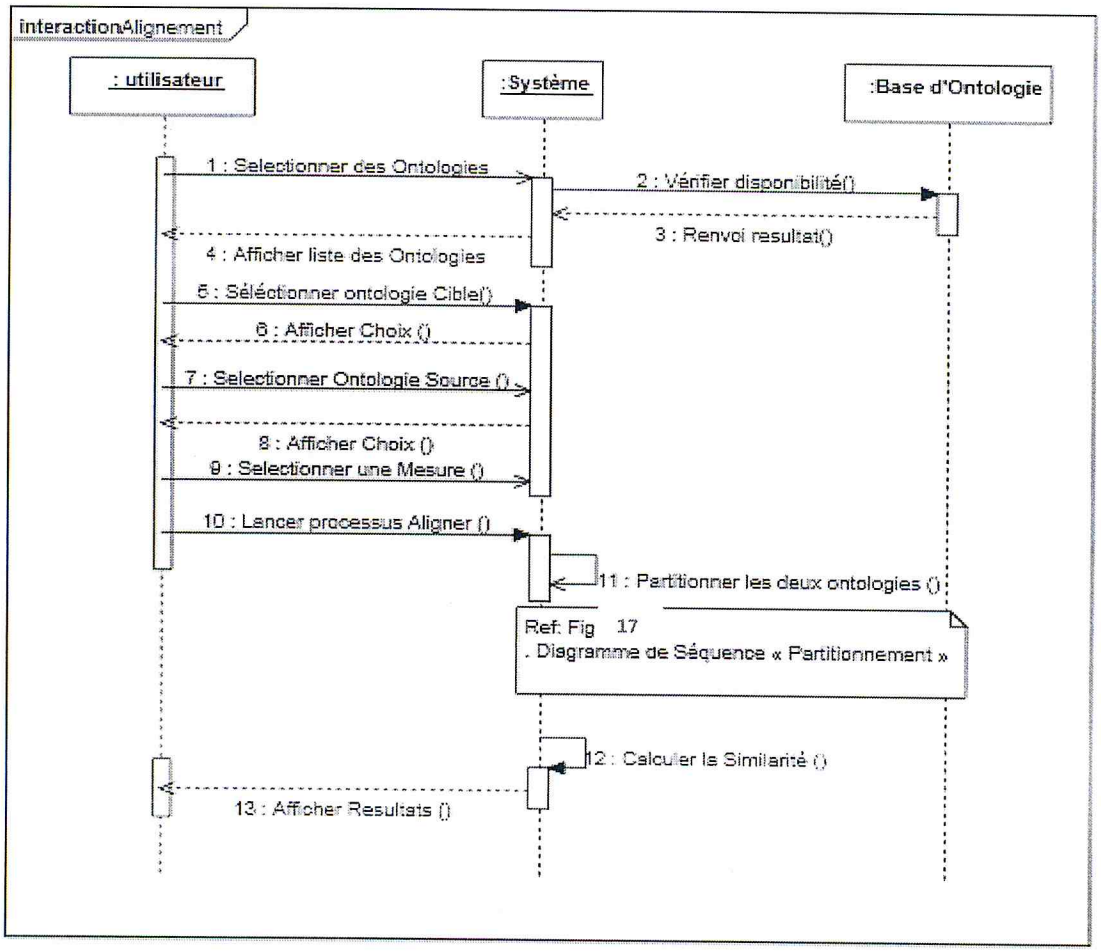


Fig. 18. Diagramme de Séquence « Alignement ».

1.1.3 Diagrammes de Paquetage (package)

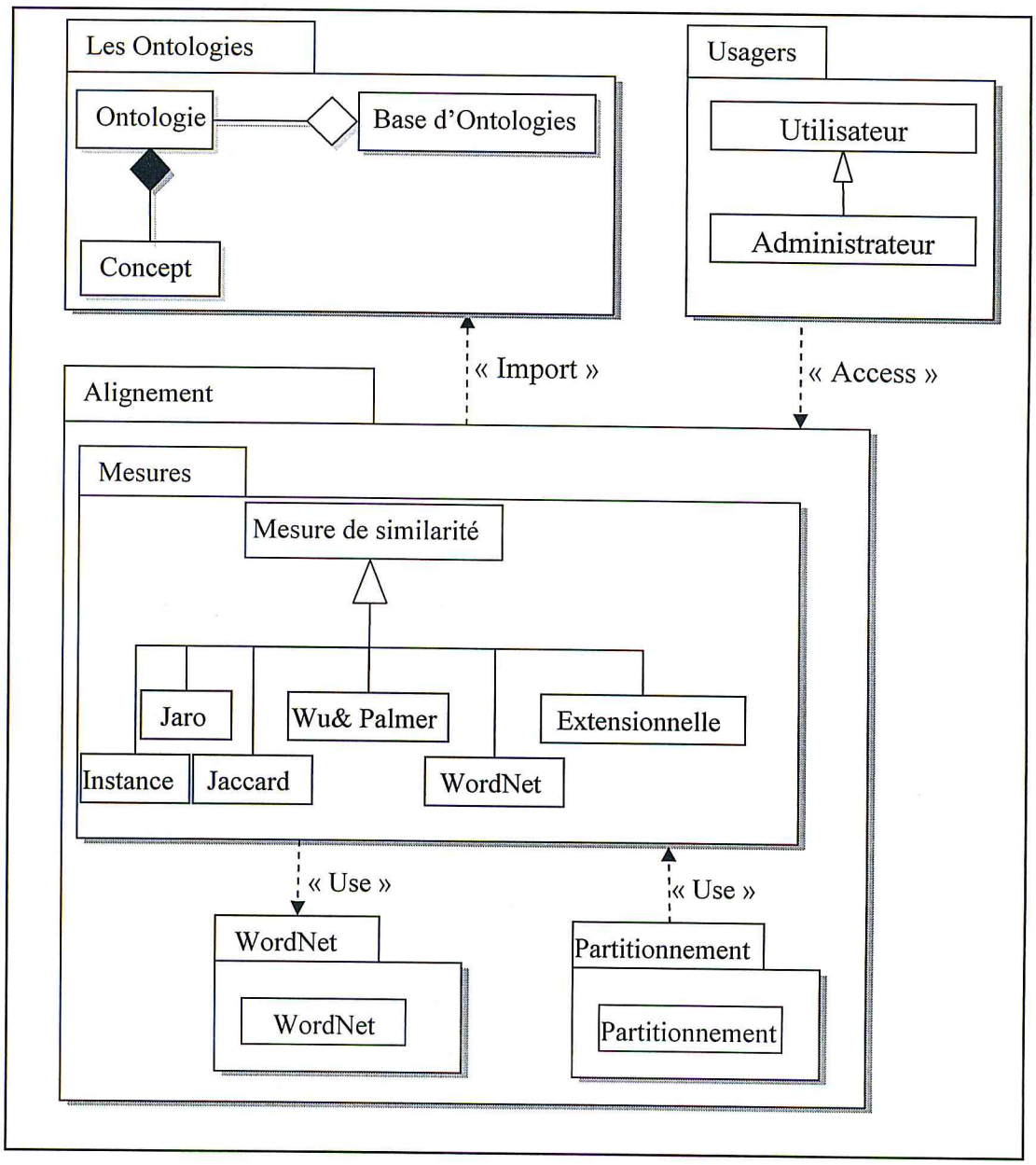


Fig. 19. Diagramme de Package « Processus d'Intégration ».

Un *diagramme de packages* est un diagramme UML qui fournit une représentation graphique de haut niveau de l'organisation de notre application, et nous aide à identifier les liens de généralisation et de dépendance entre les packages.

Ci-après le détail de paquetage de notre système :

- ***Les Usagers*** : C'est un paquetage qui représente les utilisateurs et l'administrateur ; il a accès aux autres paquetages tel que « Alignement » qui dépend du paquetage « Ontologies » à son tour.
- ***Les Ontologies*** : Ce paquetage comprend les classes qui ont lien entre eux :
  - ***Ontologie*** : Contient toutes les ontologies utilisées.
  - ***Concepts*** : C'est une classe élémentaire utilisée par d'autres unités pour effectuer des calculs sur les ontologies afin de les aligner.
  - ***Base d'ontologies*** : Classe regroupant toutes les ontologies utilisées dans notre système.
- ***Alignement :*** C'est un paquetage contenant toutes les fonctions utilisées pour le calcul de similarité, il est construit de trois autres paquetages qui se dépendent entres eux :
  - ***Mesures de similarité*** : C'est une classe qui effectue des calculs de similarités entre les concepts des ontologies. Ici nous trouvons une généralisation des classes filles tel que la classe Jaro, Jaccard, Wu&Palmer et d'autres.
  - ***WordNet*** : cette classe contient le dictionnaire lexical *WordNet* qui est importé par la classe *WordNet*.
  - ***Partitionnement*** : Ce paquetage contient la classe *Partitionnement*, cette dernière dépende de la classe *Mesures de similarité* afin de partitionner les ontologies avant d'effectuer l'alignement.

1.1.4 Diagrammes de Classes

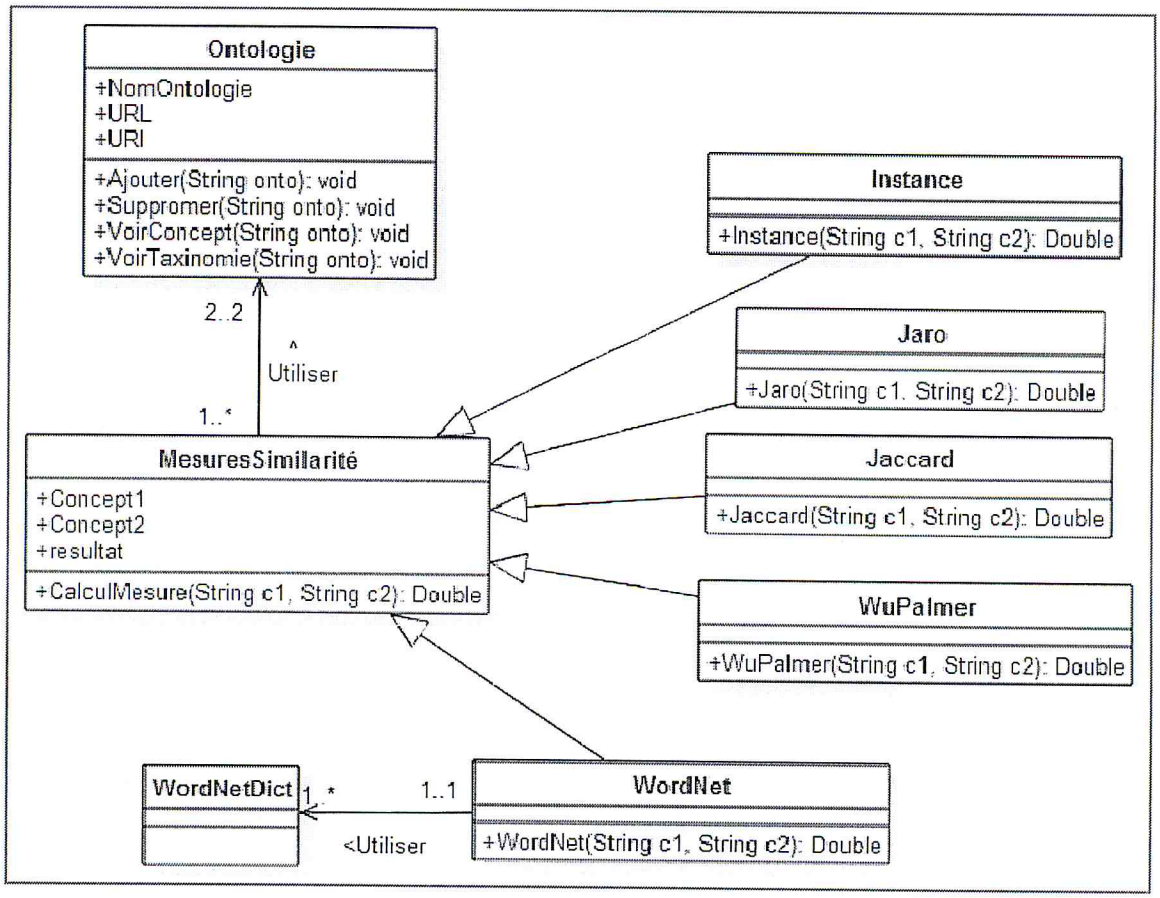


Fig. 20. Diagramme de Classes.

- **Description du diagramme de classe**

Classe	Attribut	Méthodes	Description
Ontologie	NomOntologie : String URL : String URI : String	-Ajouter (String onto) :void -Supprimer (String onto) : void -VoirConcept(String onto) : void -VoirTaxonomie(String onto) : void	Elle représente les ontologies, les manipulés et les visualisées aussi.

Mesure de similarité	Concept1 : String Concept2 : String résultat : double	-CalculMesure(String c1, String c2) : double	Classe de calcul globale en utilisant toutes les mesures vus dans ce document.

Tab .12. Dictionnaire de données.

IV.2. Conception

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrits à l'aide d'une notation commune. Elle constitue un point de départ à l'implémentation :

- elle décompose le travail d'implémentation en sous-système.
- elle crée une abstraction transparente de l'implémentation.

2.1 Solution proposée

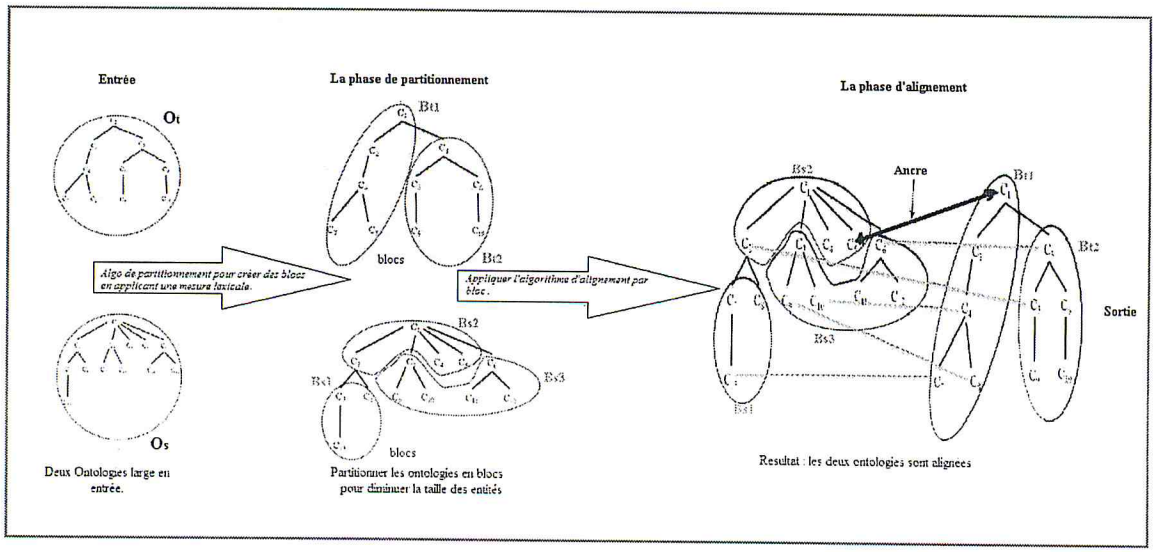


Fig.21. La solution de problème du passage à l'échelle des ontologies volumineuses.

2.2 L'Architecture du Système

Après avoir compris toutes les étapes précédentes et proposer notre solution; nous présenterons dans la suite une architecture du système "*OntInter*" qui visualise ces fonctionnalités afin de le bien concevoir. Nous avons deux acteurs, l'administrateur qui est le seul qui a l'habilité de faire toute mise à jour concernant l'Utilisateur (qui est le deuxième acteur de notre système) et de visualiser les opérations adéquates.

Nous avons besoin aussi d'une base de données des ontologies accessible par les deux acteurs pour qu'ils puissent visualiser les résultats des calculs.

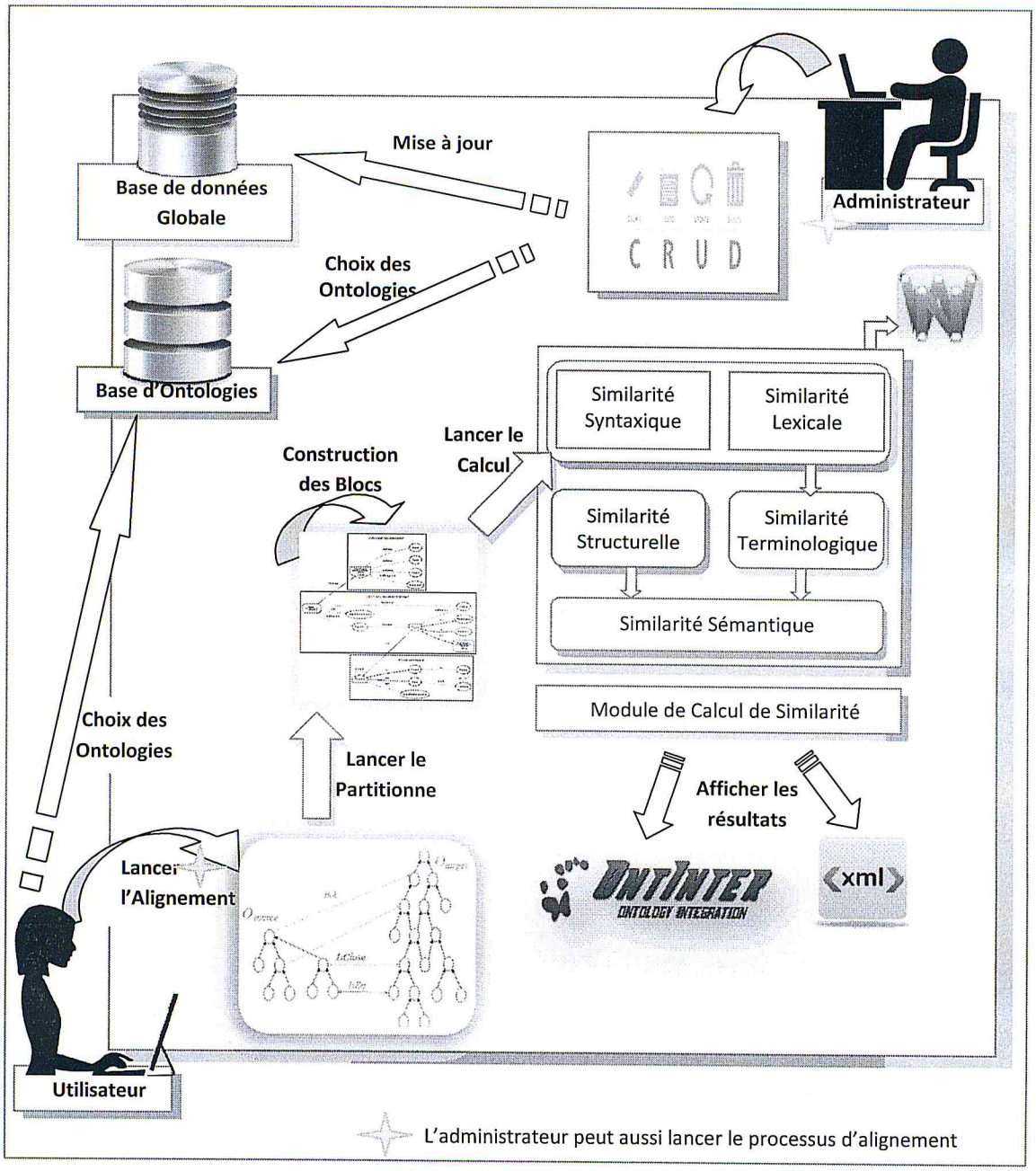


Fig. 22. L'Architecture du Système « OntInter ».

- ***Description du système***

Le système OntInter admet plusieurs représentations modulaires, il est constituer des modules suivants :

- **Module Base de données Globale**

  Ce module de base de données contient les informations des usagers du système et les résultats d'alignement pour la réutilisation, Ces derniers peuvent y accéder avec leurs natures; ils sont soit des administrateurs soit des utilisateurs.

- **Module Base d'ontologie**

  La base d'ontologie contient des ontologies utilisées pour le partitionnement et l'alignement.

- **Module WordNet**

  Dans ce module nous avons travaillé avec l'API Java pour WordNet Recherche (JAWS[3]) qui est une API qui permet à notre application la possibilité de récupérer des données à partir de la base de données WordNet. C'est une API simple et rapide, qui est compatible avec les versions 2.1 et 3.0 des fichiers de base de données de WordNet et peut être utilisé avec Java 1.4 et versions ultérieures.

- **Module XML *«eXtensible Markup Language»***

  XML est l'acronyme de «eXtensible Markup Language»,il permet d'échanger des données entres applications hétérogènes car il peut modéliser et stocker des données de façon portable.

- **Module mesures de similarités**

  Ce module admet toutes les techniques de calcul de similarité entre les entités des ontologies dans le but de les partitionner et les aligner.

Ces modules rendent notre application conviviale et simple à manipuler par l'utilisateur.

[3] http://lyle.smu.edu/~tspell/jaws/

2.3 Partitionnement

Nous avons abordé dans le chapitre précédent les différentes étapes de partitionnement et les mesures utilisées à cet effet, dans ce chapitre nous allons détailler en commençant par la détermination des blocs puis les mesures de similarités.

a) Déterminer les blocs de O_T

Nous avons prit connaissance que la valeur de *goodness* à pour objectif de prendre en compte les liens entre les deux ontologies $O_{T\,et}\,O_S$.

L'équation de *goodness* est comme suit :

$$goodness(B_i;B_j) = \alpha\left(\frac{\sum_{ci\in Bi;cj\in Bj} link(ci\,;\,cj)}{sizeOf(B_i)\,.\,sizeOf(B_j)}\right) + (1-\alpha)\left(\frac{\sum_{cj\in Bj\,;ck\in OS} simlight(cj\,;\,ck)}{\sum_{cn\in OT\,;cm\in OS} simlight(cn\,;\,cm)}\right)$$

Où α ϵ [0, 1], B_i et B_j 2 blocs de O_T , $\sum_{cj\in Bj\,;ck\in OS} simlight(cj\,;\,ck)$ représente le nombre d'ancres présentes dans Bj et $\sum_{cn\in OT\,;cm\in OS} simlight(cn\,;\,cm)$, nombre d'ancres total.

b) Déterminer les blocs de OS

En considérant que, un bloc B_i dans O_S avec une valeur de cohésion maximale, le calcul de *goodness* pour trouver le bloc ayant la valeur de couplage maximale avec B_i se base non seulement sur les relations structurelles dans O_S, mais aussi sur les relations d'équivalences avec le bloc de O_T qui a le maximum d'équivalents avec B_i.

L'équation de *goodness* devient :

$$goodness(B_i;B_j) = \alpha\left(\frac{\sum_{ci\in Bi;cj\in Bj} link(cj\,,cj)}{sizeOf(B_i)\,.\,sizeOf(B_j)}\right) + (1-\alpha)\left(\frac{\sum_{cj\in Bj\,;ck\in Bk} simlight(Bj\,;Bk)}{\sum_{cn\in 0t\,;cm\in 0s} simlight(cn\,;\,cm)}\right)$$

Où α ϵ [0, 1], B_i et B_j 2 blocs distincts de O_S et où B_k est le bloc de O_T qui partage le plus d'ancres avec B_i.

Lors des opérations de fusion, nous conservons pour chaque bloc de O_S l'ensemble des ancres appartenant au bloc, ce qui facilitera lors de la recherche des blocs à aligner, l'identification des blocs partageant le plus d'ancres.

c) *Identification des blocs à aligner*

La conservation des ancres introduites dans chaque bloc facilite la phase de calcul des paires de blocs partageant le plus d'ancres, un bloc de O_S ne s'alignant qu'avec un seul bloc de O_T (Voir l'Algorithme). Une fois identifiés les blocs de O_S devant être alignés avec le même bloc de O_T nous pouvons éventuellement les fusionner si leur taille le permet, afin de diminuer le nombre de combinaisons à faire lors du processus d'alignement.

Algorithm APP_Align(S, T, P)

Require: $C = \{B_{T1}, B_{T2}, ...B_{Tn}\}$ l'ensemble des blocs de l'ontologie cible O_T.
Require: $S = \{B_{S1}, B_{S2}, ...B_{Sn}\}$ l'ensemble des blocs de l'ontologie source O_S.
1: **for** each bloc B_{Ci} dans T **do**
2: initialiser $P_i = \emptyset$
3: **for** each bloc B_{Sj} dans S **do**
4: **if** B_{Ti} est le bloc avec lequel B_{Sj} partage le plus d'ancres **then**
5: $P_i = P_i \cup B_{Sj}$
6: Supprimer B_{Sj} de S
7: **end if**
8: **end for**
9: **end for**
10: **return** $P = \{(P_1, B_{T1}), (P_2, B_{T2})...(P_m), B_{Tm}\}$ où les P_i représentent l'ensemble des blocs B_{Sj} devant être alignés avec un bloc B_{Ti}.

2.4 Le calcul de similarité

2.4.1 Les Méthodes terminologique

a.1) Similarité syntaxique

- ***Distance de Jaro*** : Soit s et t deux chaînes de caractères. Soit N_c le nombre des caractères communs apparaissant dans les deux chaînes dans une distance de moitié de la longueur de la chaîne la plus courte. Soit N_t le nombre des caractères transposés, qui sont des caractères communs apparaissant dans des positions différentes. La distance de *Jaro* est une fonction de la dissimilarité

DS_{Jaro} : $S \times S \in [0, 1]$ telle que :

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

✓ $Exemple: jaro(Marie, Maires) = \frac{1}{3}\left(\frac{5}{|5|} + \frac{5}{|6|} + \frac{(5-2)/2}{5}\right) = \boxed{0.71}$

- ***La métrique de jaccard*** Soit s et t deux chaînes de caractères. Soit S et T les ensembles des caractères de s et t respectivement. La similarité de Jaccard est une fonction de la similarité $Sim_{Jaccard} : S \times S \in [0, 1]$ telle que :

$$\overline{Sim_{jaccard}}(s,t) = \frac{|s \cap t|}{|s \cup t|}$$

✓ $Exemple: jaccard(Marie, Maires) = \frac{5}{11} = \boxed{0.45}$

a.2) Similarité lexicale

La similarité lexicale entre deux concepts ci et cj , $Sim_L(ci, cj)$, se calcul comme suit :

$$Sim_L\ (ci,\ cj) = 1 - \overline{DS_{jaro-winkler}}\ (s,\ t)$$

Où s (resp. t) la chaine de caractère correspondant à la description du concept ci (resp. cj), et le terme *Winkler (s, t)* est ajouté pour améliorer le résultat en utilisant la méthode de jaro-Winkler [19].

Les concepts ci et cj sont jugés similaires si $Sim_L(ci, cj) > 0.65$.

✓ $Exemple:\ Lex(Marie, Maires) = 1 -\ Winkler\ (Marie, Maires) = 1 - 0.85 = \boxed{0.15} < 0.65$

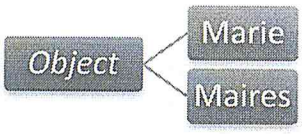
a) Les méthodes structurelles

- ***la mesure de Wu et Palmer*** qui se calcul comme suit :

$$Sim_s\ (c_i,\ c_j) = \frac{2 * depthOf\ (C)}{depthOf\ (Ci) + depthOf\ (Cj)}$$

$$|depthOf(ci) - depthOf(cj)| \leq 1$$

✓ *Exemple*:



depthOf(C)= racine =1 ;

DepthOf(Ci)=1 ; depthOf(Cj)=1

$$Sim_s = \frac{2 * 1}{1 + 1} = \boxed{1}$$

2.4.2 Les méthodes Sémantiques (dites aussi globales)

Elles sont fondées sur le modèle des entités utilisé pour valider et justifier les résultats d'alignement. Ce sont des méthodes déductives.

Nous allons appliquer des méthodes terminologiques (*lexicales et syntaxiques*) et structurelles afin de chercher les correspondances entre les entités des deux ontologies. Nous avons proposés la méthode suivante :

$$Sim_{sem}(c_i, c_j) = \begin{cases} \dfrac{(Sim_s(ci,cj) \times CoefStr) + (Sim_L(ci,cj) \times CoefLex) + \left(\overline{DS_{jaro}}(s,t) + \overline{Sim_{jaccard}}(s,t)\right) \times CoefSyn}{CoefStr + CoefLex + CoefSyn} & \text{si } CoefStr, CoefLex, CoefSyn \neq 0 \\ \dfrac{(Sim_s(ci,cj) \times CoefStr) + (Sim_L(ci,cj) \times CoefLex)}{CoefStr + CoefLex} & \text{si } CoefSyn = 0 \\ 0 & Sinon \end{cases}$$

Tel que :

***Sim*$_s$ (c_i, c_j) :** est la similarité structurelle entre les concepts c_i et c_j.

***Sim*$_L$ (c_i, c_j) :** est la similarité lexicale entre les concepts c_i et c_j.

***Sim*$_{jaccard}$ (c_i, c_j) :** est la métrique de jaccard entre le concepts c_i et c_j.

$\overline{DS}_{jaro}$ **(s, t) :** est la distance de jaro entre les ensembles $s \in S$ et $t \in T$.

CoefStr, CoefLex et CoefSyn : sont respectivement coefficient structurelle, coefficient lexicale et coefficient syntaxique. Ces coefficients sont calculés comme suit :

$$\begin{cases} CoefStr = e^{Sim_s} \\ CoefLex = e^{Sim_L} \\ CoefSyn = e^{Sim_{jaccard}} \end{cases}$$

2.4.3 Les méthodes Extensionnelles

- ***Distance de Jaccard*** (*version adaptée pour les ensembles des instances*).

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

$$\overline{DS_{jaccard}}(s,t) = 1 - \frac{P(S \cap T)}{P(S \cup T)}$$

IV.3. Conclusion

Dans ce chapitre vous avons présenté quelques diagrammes UML afin d'interpréter notre système, par la suite nous nous trouvons devant une architecture globale qui montre les différents modules utilisés, puis, nous avons rentré dans les détails de partitionnement et de calcul de similarité. Dans ce qui suit, nous allons aborder la phase implémentation de notre système avec des figures exprimables.

Chapitre V. Implémentation

V.1. Introduction

Implémenter c'est réaliser la phase finale d'élaboration d'un système qui permet au matériel, aux logiciels et aux procédures d'entrer en fonction. [20]

Dans ce chapitre nous allons présenter quelques captures d'écrans de différentes interfaces afin de bien visualiser notre système.

V.2. Outils et Langages utilisés

2.1 NetBeans IDE 7.2.1

NetBeans est un environnement de développement intégré (EDI), placé en *open source* par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, JavaScript, XML, Groovy, PHP et HTML de façon native ainsi que bien d'autres (comme Python ou Ruby) par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme. [21]

2.2 API Jena

Jena est un API java open source développé par le laboratoire de HewlettPackard permettant la lecture et la manipulation des ontologies décrites en RDFS ou en OWL. La page d'accueil de Jena, illustrée ci-dessous, est http://jena.sourceforge.net/

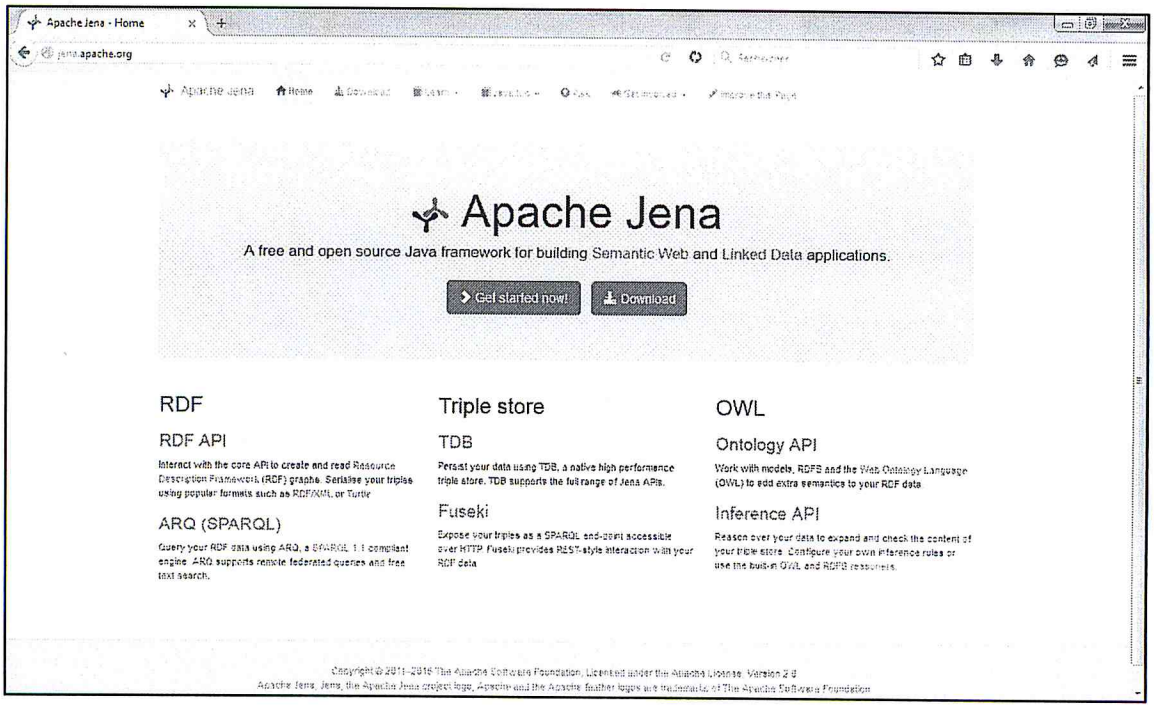


Fig. 23. Page d'accueil Jena.

2.3 Java

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy Il existe plusieurs IDE (Integrated Development Environement) pour le langage JAVA par exemple Eclips, JBuilder et NetBeans que nous avons utilisé.

2.4 SGBD « SQLite »

SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. SQLite implémente en grande partie le standard SQL-92 et des propriétés ACID.

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma

habituel client-serveur mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

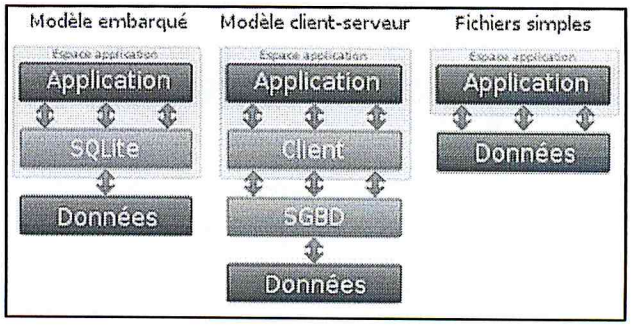


Fig. 24. Base de données embarquée.

SQLite est le moteur de base de données le plus distribué au monde, grâce à son utilisation dans de nombreux logiciels grand public comme *Firefox*, *Skype*, *Google* , dans certains produits *d'Apple*, *d'Adobe* et de *McAfeeet* dans les bibliothèques standards de nombreux langages comme PHP ou Python. De par son extrême légèreté (moins de 300 Kio), il est également très populaire sur les systèmes embarqués, notamment sur la plupart *dessmartphones* modernes : *l'iPhone* ainsi que les systèmes d'exploitation mobiles *Symbian* et *Android* l'utilisent comme base de données embarquée. Au total, on peut dénombrer plus d'un milliard de copies connues et déclarées de la bibliothèque.

2.5 OWL2Perfuse

OWL2Prefuse est un package Java qui crée des graphes et des arbres Prefuse à partir de fichiers OWL. Il prend soin de convertir la structure de données OWL en une structure de données Prefuse.

2.6 JBDC

JDBC (*Java DataBase Connectivity*) est une interface de programmation pour les programmes utilisant la plateforme Java. Elle permet à notre système d'accéder par le biais d'une interface commune, qui est SQLite dans notre cas, à des sources de données relationnelle.

2.7 JOWS

JAWS (*Java API for WordNet Searching*) comme son nom l'indique c'est un API4 Java qui permet aux applications Java de récupérer des données à partir de la base de données WordNet.

2.8 JDOM

JDOM est une *API* open source dont le but est de représenter et manipuler un document XML en Java.

2.9 WordNet 2.1

WordNet est une base de données lexicale. Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise.

2.10 SAX API Java

SAX est l'acronyme de Simple API for XML. Cette API a été développée par *David Megginson*[4]. Ce type de parseur utilise des événements pour piloter le traitement d'un fichier XML. Un objet (nommé handler en anglais) doit implémenter des méthodes particulières définies dans une interface de l'API pour fournir les traitements à réaliser (selon les événements, le parseur appelle ces méthodes[5]).

V.3. Mise en œuvre du Système

- *SplashScreen*

Fig. 25. Capture de « SplashScreen » ou « Écran de Démarrage ».

Un *splashscreen* (traduction littérale : *écran d'éclaboussure* ; en français, on devrait dire page de garde ou fenêtre d'attente) est la toute première fenêtre affichée par notre application.

[4] http://www.jmdoudoux.fr/java/dej/chap-sax.htm
[5] www.megginson.com/SAX/index.html

- ***Authentification***

Fig.26. Capture de « Authentification ».

Cette fenêtre permet à l'utilisateur de s'identifier, sinon il n'y aura pas accès. Si l'usager est un Administrateur il se retourne à la page d'Accueil, Si ce n'est qu'un simple Utilisateur, il passe à la fenêtre AlignementFrame.

- ***Page d'Accueil***

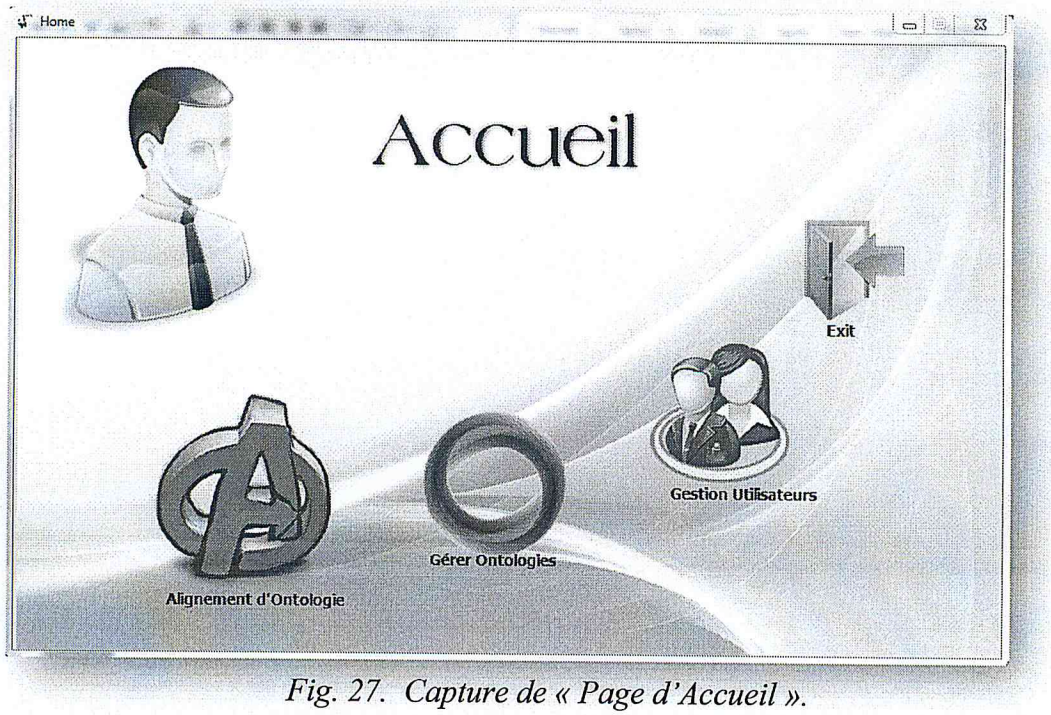


Fig. 27. Capture de « Page d'Accueil ».

C'est la deuxième fenêtre visitée par l'Administrateur dont il peut gérer les utilisateurs, gérer les ontologies, effectuer l'alignement ou sortir.

- ***Fiche d'Utilisateur***

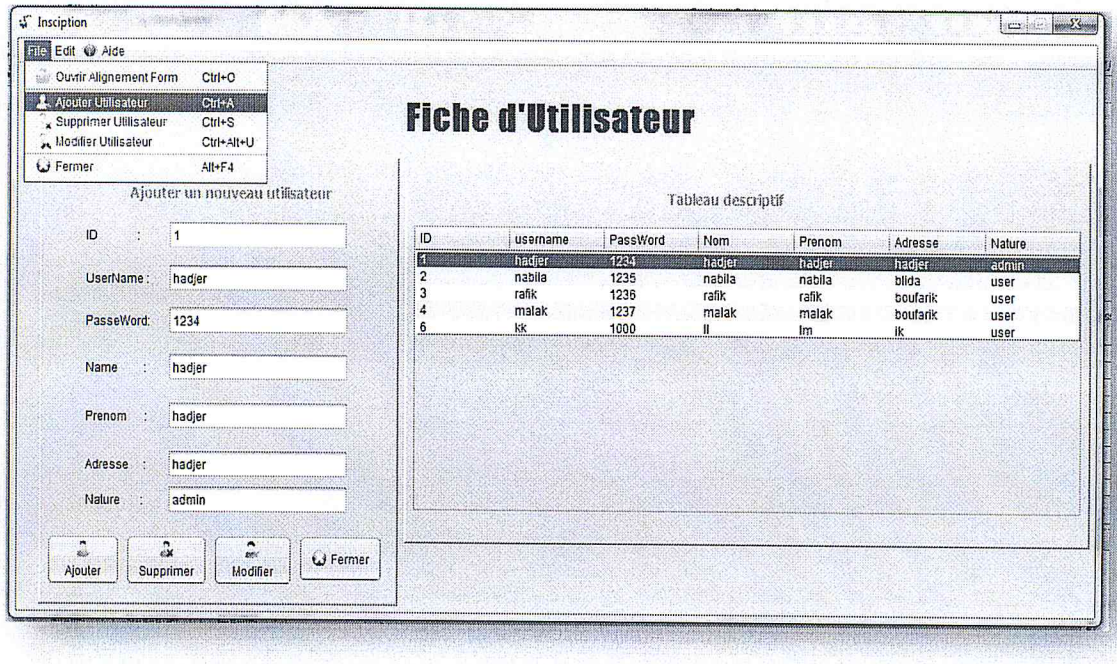


Fig. 28. Capture de « Fiche d'utilisateur ».

C'est la fenêtre dont l'Administrateur Met à jour les usagers de l'application.

- ***Gérer Ontologie***

Fig. 29. Capture de « Gérer Ontologie ».

Cette interface permet de voir la liste des ontologies utilisées dans l'application, et de supprimer un fichier donné.

- *Alignement Frame*
  - ***Onglet Alignement***

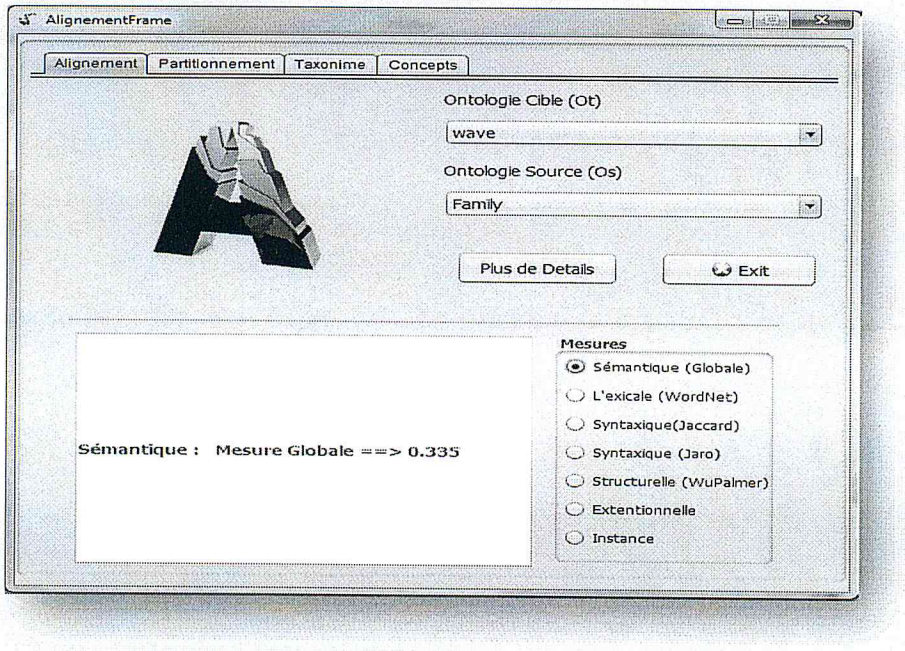


Fig. 30. Capture de « Onglet Alignement ».

Nous aurrons dans cet onglet la valeur de correspondance des ontologies choisits tout en selectionnant une mesure.

  - ***Detail***

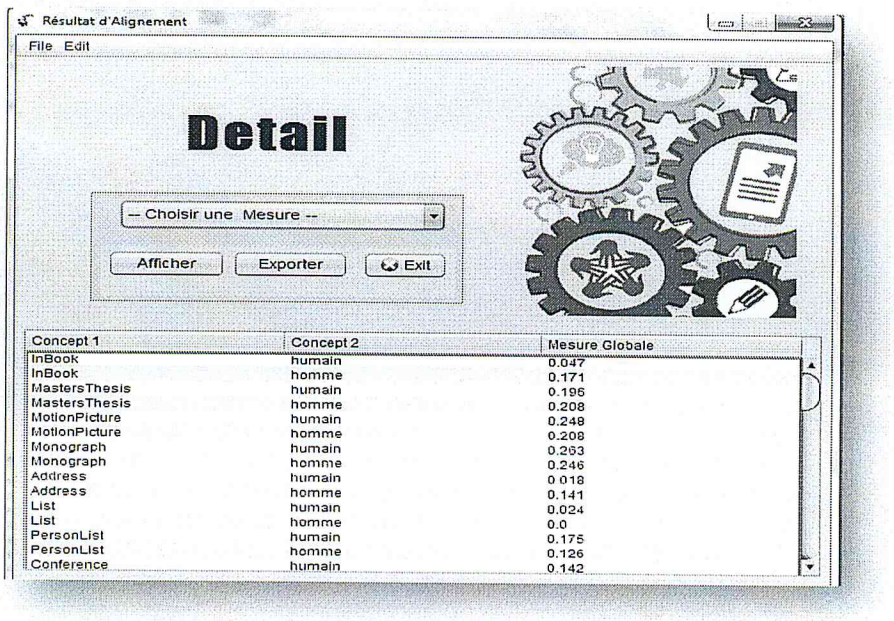
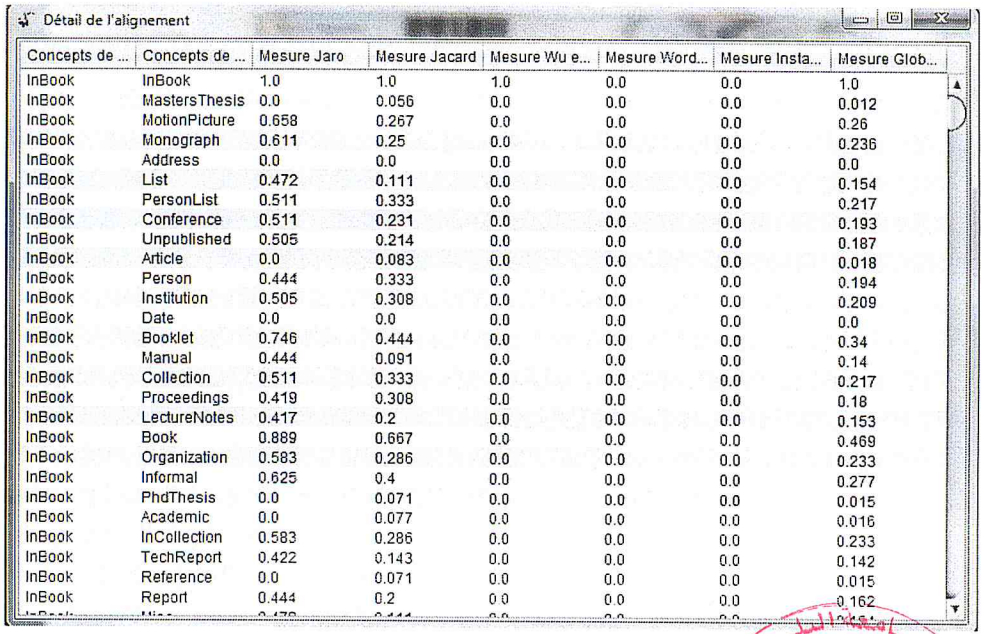


Fig. 31. Capture de « Détail ».

Cette fenêtre affiche le résultat d'alignement en détail et par concept, et lorsque nous choisissons une autre mesure ou une récapitulation de calcul un autre tableau nous montra ça « voir fig. 32 ».



Concepts de ...	Concepts de ...	Mesure Jaro	Mesure Jacard	Mesure Wu e...	Mesure Word...	Mesure Insta...	Mesure Glob...
InBook	InBook	1.0	1.0	1.0	0.0	0.0	1.0
InBook	MastersThesis	0.0	0.056	0.0	0.0	0.0	0.012
InBook	MotionPicture	0.658	0.267	0.0	0.0	0.0	0.26
InBook	Monograph	0.611	0.25	0.0	0.0	0.0	0.236
InBook	Address	0.0	0.0	0.0	0.0	0.0	0.0
InBook	List	0.472	0.111	0.0	0.0	0.0	0.154
InBook	PersonList	0.511	0.333	0.0	0.0	0.0	0.217
InBook	Conference	0.511	0.231	0.0	0.0	0.0	0.193
InBook	Unpublished	0.505	0.214	0.0	0.0	0.0	0.187
InBook	Article	0.0	0.083	0.0	0.0	0.0	0.018
InBook	Person	0.444	0.333	0.0	0.0	0.0	0.194
InBook	Institution	0.505	0.308	0.0	0.0	0.0	0.209
InBook	Date	0.0	0.0	0.0	0.0	0.0	0.0
InBook	Booklet	0.746	0.444	0.0	0.0	0.0	0.34
InBook	Manual	0.444	0.091	0.0	0.0	0.0	0.14
InBook	Collection	0.511	0.333	0.0	0.0	0.0	0.217
InBook	Proceedings	0.419	0.308	0.0	0.0	0.0	0.18
InBook	LectureNotes	0.417	0.2	0.0	0.0	0.0	0.153
InBook	Book	0.889	0.667	0.0	0.0	0.0	0.469
InBook	Organization	0.583	0.286	0.0	0.0	0.0	0.233
InBook	Informal	0.625	0.4	0.0	0.0	0.0	0.277
InBook	PhdThesis	0.0	0.071	0.0	0.0	0.0	0.015
InBook	Academic	0.0	0.077	0.0	0.0	0.0	0.016
InBook	InCollection	0.583	0.286	0.0	0.0	0.0	0.233
InBook	TechReport	0.422	0.143	0.0	0.0	0.0	0.142
InBook	Reference	0.0	0.071	0.0	0.0	0.0	0.015
InBook	Report	0.444	0.2	0.0	0.0	0.0	0.162

Fig. 32. Capture de « Tableau des Détails ».

- ***Onglet Partitionnement***

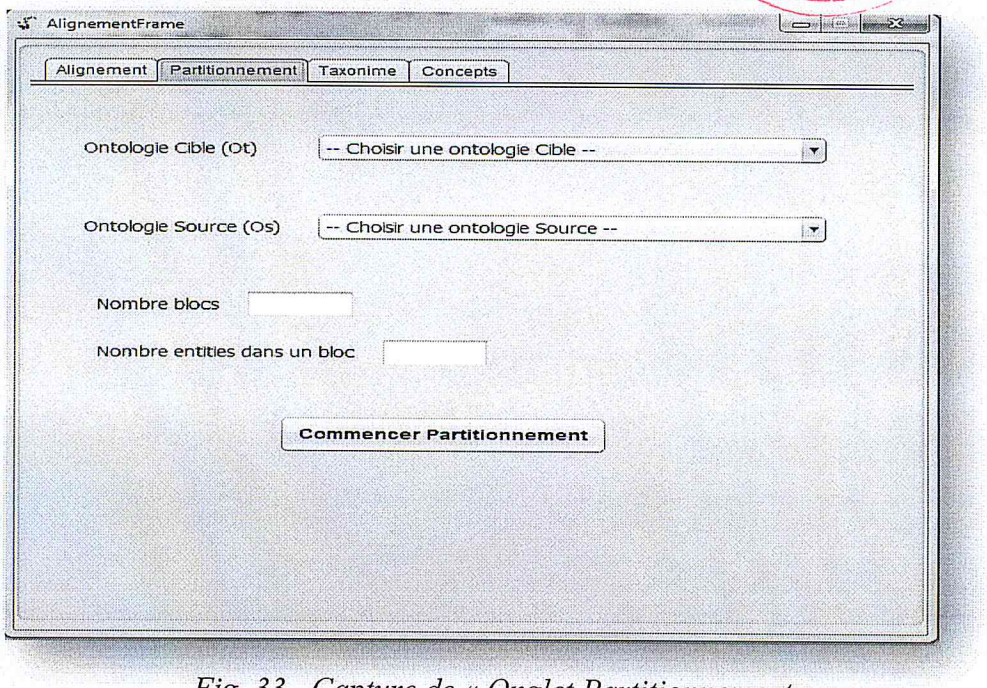


Fig. 33. Capture de « Onglet Partitionnement ».

Dans cet onglet, nous obtiendrons comme résultat un ensemble de blocs contenant un nombre maximum d'entités précisé par l'usager sous forme d'un fichier XML.

- ***Onglet Taxonomie***

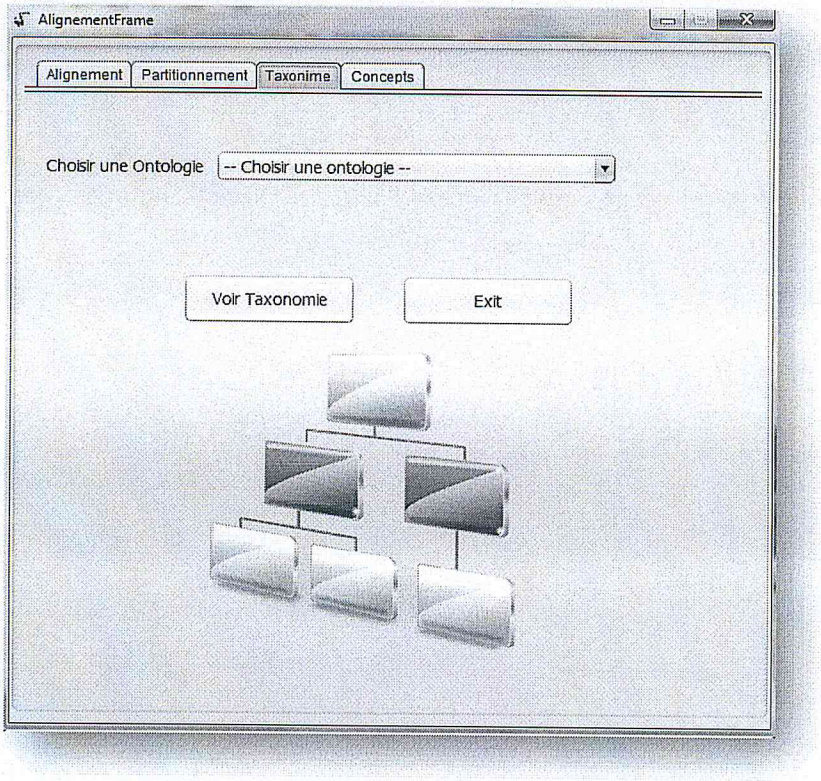


Fig. 34. Capture de « Onglet Taxonomie ».

- ***Taxonomie***

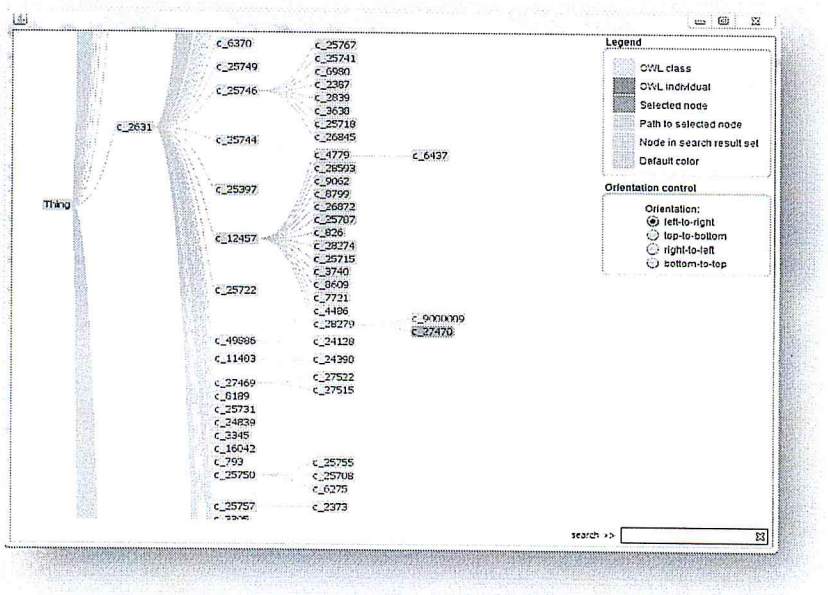


Fig. 35. Capture de « Taxonomie ''une partie d'AGROVOC'' ».

Cette capture montre la taxonomie de l'ontologie Agrovoc.

- ***Onglet Concepts***

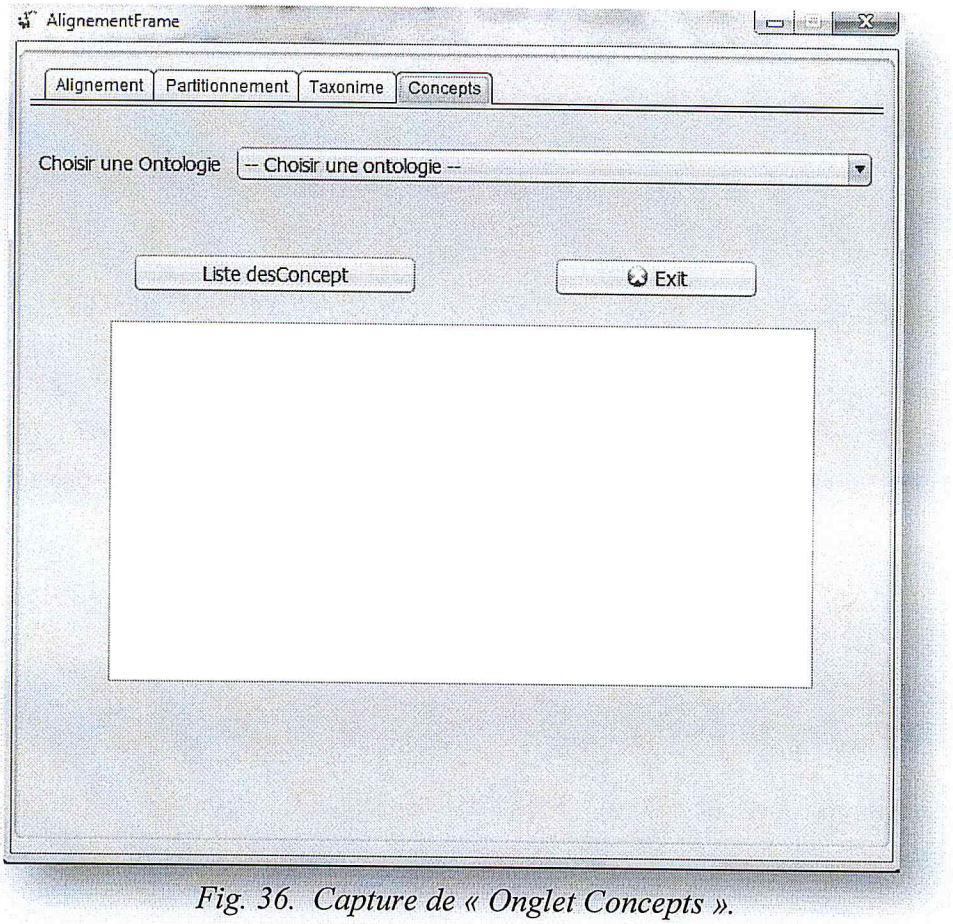


Fig. 36. Capture de « Onglet Concepts ».

En cliquant sur le bouton « Liste des Concepts » nous obtienderons les concepts d'une ontologie choisie.

V.4. Conclusion

Ce chapitre inclut l'étape finale de notre étude, dans lequel nous avons eu une idée sur les outils et les APIs utilisés pour réaliser cette partie, et quelques captures d'écrans des différentes fenêtres avec des petites descriptions. Le chapitre suivant aborde nos résultats avec les résultats d'autres expérimentations.

Chapitre VI. Test et Validation

VI.1. Introduction

Pour évaluer les performances de notre algorithme, nous avons réalisé notre outil qui supporte en entrée deux ontologies décrites en OWL, puis à comparer les correspondances obtenu par notre système et celles trouvées par le système TaxoPart (de Hamdi et al.).

Nous avons procédé à une série de tests en utilisant les tests fournis dans la base *Benchmark* de test à la disposition de la communauté internationale par la compétition OAEI (Ontology Alignment Evaluation Initiative).

VI.2. Les Benchmarks

Le nombre croissant de méthodes disponibles pour le mapping d'ontologie nécessite d'établir un consensus pour l'évaluation de ces méthodes. Depuis 2004, Les chercheurs en alignement des ontologies, avec le nombre croissant des systèmes automatiques d'alignement, organisent des campagnes annuelles d'évaluation appelées OAEI [6].

L'objectif du cas de test des benchmark est de fournir une image stable et détaillée de chaque algorithme de mapping. A cet effet, les algorithmes sont exécutés sur des cas de test générés de façon systématique.

OAEI est une campagne qui vise à comparer les systèmes de mapping des ontologies sur des cas de test définis précisément. Les cas de test peuvent utiliser des ontologies de nature différente.

Les objectifs de l'initiative d'évaluation d'alignement d'ontologies sont :

- Mettre en évidence les forces et les faiblesses de ces systèmes.
- Comparer les performances des différentes techniques.
- Favoriser la communication entre les développeurs de ces systèmes.
- Développer l'évaluation des techniques d'alignement.

[6] oaei.ontologymatching.org

- D'une manière générale, faire progresser la recherche dans le domaine de l'alignement des ontologies.

VI.3. Les données de tests

Dans le cadre des expérimentations menées, les tests fournis dans la base benchmark mise à la disposition de la communauté par la compétition OAEI sont utilisés. Le domaine de ce test est le domaine des références bibliographiques.

L'ontologie de base est constituée par un ensemble de références bibliographiques. Elle représente une version plus allégée en nombre d'entités ontologiques comparativement avec des ontologies réelles. Chaque cas de test de la base benchmark met en exergue une caractéristique de la deuxième ontologie à aligner avec la base de test. L'objectif de cette base de tests est de prendre en charge tous les aspects qui existent dans une ontologie OWL-DL et qui pourraient avoir un impact considérable sur les métriques d'évaluation du résultat de l'alignement.

VI.4. Test des ontologies 101, 103 et Nalt

Les résultats obtenus sont des parties des résultats d'alignement des ontologies 101 et 103 en les partitionnant, puis avec l'ontologie Nalt.owl.

```
<?xml version="1.0" encoding="utf-8"?>
<Alignement>
  <map >
    <Cell>
      <entity1>InBook  </entity1>
      <entity2>InBook  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
  </map >
  <map >
    <Cell>
      <entity1>MastersThesis  </entity1>
      <entity2>MastersThesis  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
  </map >
```

Fig. 37. Capture de « résultats alignement 101 et 103 ».

```
<map >
   <Cell>
      <entity1>MotionPicture  </entity1>
      <entity2>MotionPicture  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
   </Cell>
</map >
<map >
   <Cell>
      <entity1>Monograph  </entity1>
      <entity2>Monograph  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
   </Cell>
</map >
<map >
   <Cell>
      <entity1>Address  </entity1>
      <entity2>Address  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
<map >
   <Cell>
      <entity1>Person  </entity1>
      <entity2>Person  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
   </Cell>
</map >
<map >
   <Cell>
      <entity1>Institution  </entity1>
      <entity2>Institution  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
   </Cell>
</map >
<map >
   <Cell>
      <entity1>Date  </entity1>
      <entity2>Date  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
   </Cell>
</map >
```

Fig. 38. Capture de « résultats alignement 101 et 103 ».

Après avoir tester les ontologies 101 et 103 en utilisant l'algorithme de partitionnement et appliquer des méthodes vues dans le chapitre précédant en les comparant avec les deux systèmes experts OAEI et TaxoPart, cette figure nous affiche l'état finale après le partitionnement, nous remarquons que l'algorithme n'a prit en considération que les concepts en commun où la valeur de similarité est égale à 1, c.à.d. ils ont la relation « égale » (isEqual).

```
<?xml version='1.0' encoding='utf-8'?>

<rdf:RDF xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment'
         xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>11</type>
  <onto1>http://oaei.ontologymatching.org/tests/101/onto.rdf</onto1>
  <onto2>http://oaei.ontologymatching.org/tests/103/onto.rdf</onto2>
  <uri1>http://oaei.ontologymatching.org/tests/101/onto.rdf</uri1>
  <uri2>http://oaei.ontologymatching.org/tests/103/onto.rdf</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource='http://oaei.ontologymatching.org/tests/101/onto.rdf#type'/>
      <entity2 rdf:resource='http://oaei.ontologymatching.org/tests/103/onto.rdf#type'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://oaei.ontologymatching.org/tests/101/onto.rdf#howPublished'/>
      <entity2 rdf:resource='http://oaei.ontologymatching.org/tests/103/onto.rdf#howPublished'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://oaei.ontologymatching.org/tests/101/onto.rdf#periodicity'/>
      <entity2 rdf:resource='http://oaei.ontologymatching.org/tests/103/onto.rdf#periodicity'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
  </map>
  <map>
    <Cell>
      <entity1 rdf:resource='http://oaei.ontologymatching.org/tests/101/onto.rdf#proceedings'/>
```

Fig. 39. Capture de « résultats alignement de 101 et 103 avec l'OAEI».

Ces résultats sont pris de la compagne OAEI, les concepts ici sont précédés par un dièse ; en comparant avec notre système nous trouvons que les résultats sont pertinents.

(level 0) Alignment

Source: http://oaei.ontologymatching.org/tests/101/onto.rdf

Target: http://oaei.ontologymatching.org/tests/103/onto.rdf

Correspondences

type = type
1.0
howPublished = howPublished
1.0
periodicity = periodicity
1.0
proceedings = proceedings
1.0
volume = volume
1.0
annote = annote
1.0
PersonList = PersonList
1.0
month = month
1.0
copyright = copyright
1.0
Unpublished = Unpublished
1.0
address = address
1.0
Address = Address
1.0
chapter = chapter

Chapter = Chapter
1.0
editor = editor
1.0
InBook = InBook
1.0
Date = Date
1.0
series = series
1.0
PageRange = PageRange
1.0
date = date
1.0
title = title
1.0
Booklet = Booklet
1.0
numberOrVolume = numberOrVolume
1.0
LectureNotes = LectureNotes
1.0
url = url
1.0
MastersThesis = MastersThesis

Fig. 40. Capture de « résultats alignement de 101 et 103 avec l'OAEI».

Cette figure nous montre les résultats d'une manière statique, compréhensible par les utilisateurs.

```
<?xml version='1.0' encoding='iso-8859-1'?>

<rdf:RDF xmlns='http://xobjects.seu.edu.cn/falcon'

 xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'

 xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>

<Match>
  <Anchor>

    <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Informal"/>

        <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Informal"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>

  </Anchor>

<Anchor>

    <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Manual"/>

    <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Manual"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>


</Anchor>

<Anchor>

     <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Proceedings"/>

        <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Proceedings"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>

</Anchor>

<Anchor>

    <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Article"/>

    <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Article"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>

</Anchor>

<Anchor>

    <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Unpublished"/>

    <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Unpublished"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>

</Anchor>
<Anchor>

    <uri1 rdf:resource="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Monograph"/>

    <uri2 rdf:resource="http://oaei.ontologymatching.org/2011/benchmarks/103/onto.rdf#Monograph"/>

    <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>

</Anchor>
<Anchor>

```

Fig. 41. Capture de « résultats alignement de 101 et 103 avec TaxoPart».

La figure ci-dessus. Est une partie de résultats de TaxoPart, nous remarquons que les dénouements sont les mêmes.

```
<?xml version="1.0" encoding="utf-8"?>
<Alignement>
  <map >
    <Cell>
      <entity1>CHEMINF_000065  </entity1>
      <entity2>CHEMINF_000065  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
      <entity1>CHEMINF_000297  </entity1>
      <entity2>CHEMINF_000297  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
      <entity1>CHEMINF_000215  </entity1>
      <entity2>CHEMINF_000215  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
      <entity1>CHEMINF_000094  </entity1>
      <entity2>CHEMINF_000094  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
      <entity1>CHEMINF_000174  </entity1>
      <entity2>CHEMINF_000174  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
      <entity1>CHEMINF_000421  </entity1>
      <entity2>CHEMINF_000421  </entity2>
      <mesure>1.0 </mesure>
      <relation> = </relation>
    </Cell>
    <Cell>
```

Fig. 42. Capture de « résultats alignement de Nalt avec notre système».

```
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000111"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000111"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000078"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000078"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000110"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000110"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000077"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000077"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000076"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000076"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000075"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000075"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000074"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000074"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
</Anchor>
<Anchor>
  <uri1 rdf:resource="http://semanticscience.org/resource/CHEMINF_000073"/>
  <uri2 rdf:resource="http://semanticscience.org/resource/CHEMINF_000073"/>
  <similarity rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</similarity>
```

Fig. 43. Capture de « résultats alignement de Nalt avec TaxoPart».

Les figures 42 et 43 décrivent les dénouements des expériences avec TaxoPart, nous constatons que nous avons arrivé à approuver que notre système est bon et que les résultats sont pertinents.

VI.5. Conclusion

Dans ce chapitre nous avons testé l'efficacité de notre système sur quelques ontologies afin de pouvoir visualiser et assurer l'exactitude des résultats obtenus par rapport à la campagne OAEI et TaxoPart, et à la fin nous avons trouvé que notre système a répondu aux contraintes de notre solution proposée dans le chapitre 3. Ces résultats sont testés sous Windows 7 Édition Intégrale de type 64bits, avec un processeur Intel® Pentium ® CPU B960 @ 2.20 GHz, et une mémoire libre de 2.00 Go.

Conclusion générale

L'intégration est un processus de combinaison d'éléments qui au premier abord semblent incompatibles ou même conflictuels, mais qui après un peu d'analyse et de re-synthèse (menant à la reformulation ou la réorientation), s'avèrent être plutôt complémentaire. Le mapping est une première étape de processus d'intégration d'ontologie, il permet de déterminer les correspondances entre les ontologies à intégrer. Les techniques d'alignement existant (les mesures) sont des outils dont on dispose pour trouver une solution à la problématique d'intégration. L'alignement des ontologies de petites tailles ne fait pas des problèmes, mais avec l'apparition des nouveaux standards, outils et langages très expressifs, de grandes ontologies ont été développées dans plusieurs domaines ; l'alignement des ontologies volumineuses propose un problème appelé «passage à l'échelle» des méthodes d'alignement qui est l'une des solutions pour résoudre et traiter ce problème. En suppose un algorithme de partitionnement autour des ancres ; avant réaliser l'alignement, il faut utiliser des mesures lexicales pour améliorer le résultat ; l'avantage est de n'avoir aucun bloc isolé en préservation des alignements (c.à.d. la non perte des alignements) dans le but de fournir en sortie des paires de blocs à aligné et qui ne s'aligne que avec un seul bloc (les techniques d'alignement avec partitionnement doivent soulever le défi de n'avoir aucun bloc seul). En revanche, le résultat d'alignement avec le partitionnement est plus rassurer, suite à la limite et la diminution du nombre des concepts.

La construction des blocs permet de diminuer l'espace de recherche des correspondances, et avant d'y procéder, il faut passer par l'alignement des ontologies, à condition ces derniers sont interopérable (c.à.d. utiliser la solution de problème d'hétérogénéité).

Dans ce mémoire, Nous avons parlé des différentes étapes de partitionnement, et de différentes mesures qui peuvent être utilisées pour l'alignement, en fin nous avons proposé une architecture d'intégration d'ontologie, qui partitionne deux ontologies volumineuse en un ensemble des blocs autour des ancres afin de limiter la taille des ensembles des concepts.

REFERENCE

[1] N.CHERGUI, « *Une approche de mapping pour l'intégration des ontologies* », Mémoire Présenté en vue de l'obtention du diplôme de Magister en informatique, Université Mentouri de Constantine, 2008, p8.

[2] R. LAURINI, « *Ontologies pour les applications géographiques* », cours de Master Informatique de Lyon, 2010-2011, p 7.

[3] F. HAMDI et al. « *Partitionnement d'ontologies pour le passage à l'échelle des techniques d'alignement* », LRI, Université Paris-Sud, LIPN, Université Paris 13 - CNRS UMR 7030, 2009, p1-12.

[4] S. ABDUL GHAFOUR, « *Méthodes et outils Pour l'intégration des ontologies* », Mémoire de stage de DEA, Laboratoire d'Informatique en Images et Systèmes d'information LIRIS FRE2672 CNRS, INSA Lyon, UCB Lyon 1, EC Lyon, 2003, p 5-35.

[5] J. EUZENAT et al. « *Ontology Matching* », Springer Berlin Heidelberg New York, p9-29-36-44, 2007.

[6] S.IZZA, « *Intégration des Systèmes d'information industriels Une approche flexible basée sur les services sémantiques* », thèse Pour obtenir le grade de Docteur de l'Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006, p3-32-33

[7] A. GAL et al. « *Advances in Ontology Matching* », Technion – Israel Institute of Technology et University of Trento, Povo, Trento, Italy, p3-18, 2009.

[8] P. ALBERT et al. *Actes de la conférence IC 2007,* 18es Journées Francophones d'Ingénierie des Connaissances, 4 – 6 juillet 2007, vol(4), p 97

[9] A. ELBYED, « *ROMIE, une approche d'alignement d'ontologies à base d'instances* », Thèse présentée pour l'obtention du grade de Docteur de l'institut national des télécommunications, Université d'Evry-Val d'Essonne, 2009, p 24-66.

[10] S. BENHARZALLAH, « *Coopération multi agents pour le traitement des requêtes sur des sources de données hétérogènes et distribuées* », Mémoire de Magister en informatique, 2005, p 27-29

[11] S.THABET et al. « *Une extension de mesure de similarité entre les concepts d'une ontologie* », 4rth International Conference: Sciences of Electronic, Technologies of Information and Telecommunications March 25-29, 2007 – TUNISIA, 2007, p2-9.

[12] N. HERNANDEZ et al. « *TtoO: une méthodologie de construction d'ontologie de domaine à partir d'un thésaurus et d'un corpus de référence* »,

[13] L.GHOMARI, « *alignement d'ontologie de domaine : Etude comparative Syntaxique versus Sémantique Cas d'application : COMA++ vs. OWL-CTXMatch* », Mémoire de Magister en Informatique, Ecole nationale Supérieure d'Informatique (E.S.I), 2009, p13-15.

[14] A. AZZAZ et al. « *Une Approche Ontologique d'Intégration de Sources de Données dans un Environnement de Pair à Pair* », *p 3-4.*

[15] S. GHARBI, « *Appariement d'ontologies hétérogènes Application aux turbines à vapeur* » ; Mémoire de Magistère, 2011, p 18.

[16] S. KASRI, F. BENCHIKHA, « *Un Algorithme de Partitionnement d'Ontologies Orienté Alignement* », Université de Skikda, département d'informatique Algérie, Université de Skikda, laboratoire LIRE Constantine Algérie, p. 346—352, 2011.

[17] A. ESSAID, B. BEN YAGHLANE, A. MARTIN, « *Gestion du conflit dans l'appariement des ontologies* », publié dans "*Extraction et Gestion des Connaissances, Brest* : France ,2011.

[18] T. L. BACH, « Construction d'un Web sémantique multi-points de vue », mémoire pour l'obtention d'un titre de docteur en science option informatique, L'École des Mines de Paris à Sophia Antipolis, 2006, p 22.

[19] F. SAÄIS, « *Intégration Sémantique de Données guidée par une Ontologie* », thèse de doctorat de l'université de Paris- Sud, Université de Paris- Sud, 2007.

[20] http://www.futura-sciences.com/magazines/high-tech/infos/dico/d/informatique-implementer-549/

[21] https://fr.wikipedia.org/wiki/NetBeans.

[22] S.KASRI, « *Etude des techniques d'alignement des ontologies: proposition d'un algorithme de passage à l'échelle* », thèse de Magister en Informatique de l'université de 20 Aout 1955 Skikda, Université de Skikda, 2010.

[23] https://fr.wikipedia.org/wiki/WordNet.

[24] ftp://www.irit.fr/IRIT/SIG/Article_ISI_Passage_echelle.pdf.

ANNEXE

ANNEXE

- ***Algorithme de Partitionnement FALCON***

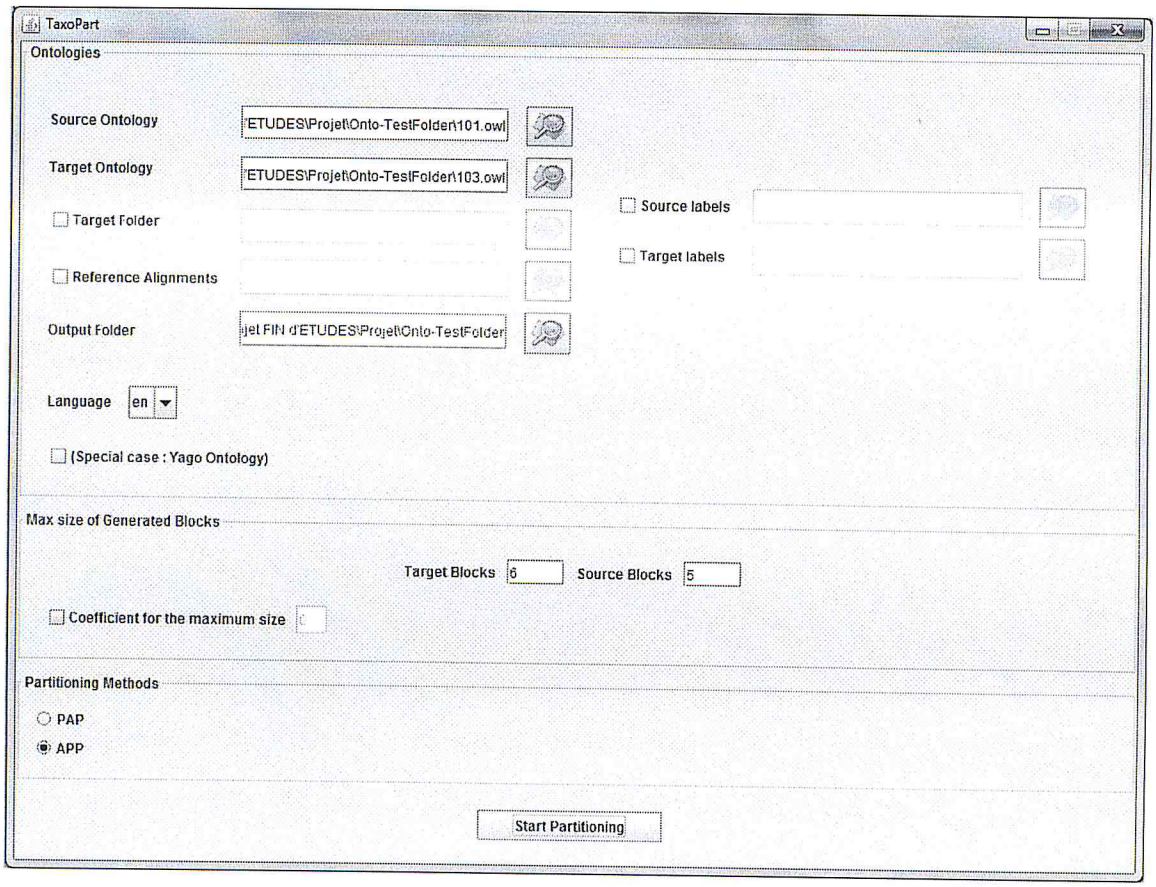
Algorithm: partition(D, W, ϵ).

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};
```

- ***Application TaxoPart***



- ***Une partie de la Syntaxe de l'ontologie 101.owl***

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1313157142234 "http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#"
     xml:base="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:Ontology1313157142234="http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->



    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Academic -->

    <owl:Class rdf:about="#Academic">
        <rdfs:subClassOf rdf:resource="#Reference"/>
    <owl:Class rdf:about="#Article">
        <rdfs:subClassOf rdf:resource="#Part"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Book -->

    <owl:Class rdf:about="#Book">
        <rdfs:subClassOf rdf:resource="#Reference"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Booklet -->

    <owl:Class rdf:about="#Booklet">
        <rdfs:subClassOf rdf:resource="#Informal"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Chapter -->

    <owl:Class rdf:about="#Chapter">
        <rdfs:subClassOf rdf:resource="#Part"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Collection -->

    <owl:Class rdf:about="#Collection">
        <rdfs:subClassOf rdf:resource="#Book"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Conference -->

    <owl:Class rdf:about="#Conference"/>
    !-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Date -->

    <owl:Class rdf:about="#Date"/>
    <!-- http://www.semanticweb.org/ontologies/Ontology1313157142234.owl#Delivrable -->

```

- ***Une partie de la Syntaxe de l'ontologie 103.owl***

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF xmlns="http://oaei.ontologymatching.org/2007/benchmarks/103/onto.rdf#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/" xmlns:ical="http://www.w3.org/2002/12/cal/ical#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xml:base="http://oaei.ontologymatching.org/2007/benchmarks/103/onto.rdf#">

<!-- This ontology describes bibliographic references. -->

 <owl:Ontology rdf:about="">
   <dc:creator>Nick Knouf &lt;nknouf@mit.edu&gt;</dc:creator>
   <dc:contributor>Antoine Zimmermann &lt;antoine.zimmermann@inrialpes.fr&gt;</dc:contributor>
   <dc:contributor>Jérôme Euzenat</dc:contributor>
   <dc:description>Bibliographic references in OWL</dc:description>
   <dc:date>08/06/2005</dc:date>
   <rdfs:label>Bibliographic references</rdfs:label>
   <rdfs:comment>Possible ontology to describe bibTeX entries.</rdfs:comment>
   <owl:versionInfo>$Id: onto.rdf,v 1.27 2006/08/01 10:03:10 euzenat Exp $</owl:versionInfo>
 </owl:Ontology>

 <!-- Every entity (even external) must be typed in OWL-DL. -->

 <!-- this is for satisfying the OWL Species validator
      (which is not satisfied anyway) -->
 <owl:Class rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
 <rdf:List rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
 <owl:ObjectProperty rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#first">
   <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#rest">
   <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
   <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
 </owl:ObjectProperty>

 <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/creator"/>
 <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/contributor"/>
 <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/description"/>
 <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/date"/>

 <!-- foaf extensions -->
 <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Person"/>
 <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Organization"/>
 <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/firstName"/>
 <owl:DatatypeProperty rdf:ID="lastName"/>
 <owl:DatatypeProperty rdf:about="http://xmlns.com/foaf/0.1/name"/>

 <!--
   ********  ENTRIES  ********

   Entries form the basis of a bibTeX database and are categorized by their type, such as a book, journal article,
   Note: all rdfg:comment values for the entries come from http://newton.ex.ac.uk/tex/pack/bibtex/btxdoc/node6.htm
 -->

 <owl:Class rdf:ID="Reference">
   <rdfs:label xml:lang="en">Reference</rdfs:label>
   <rdfs:comment xml:lang="en">Base class for all entries</rdfs:comment>
   <rdfs:subClassOf>
     <owl:Restriction>
   <owl:onProperty rdf:resource="#date"/>
   <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
   <owl:onProperty rdf:resource="#title"/>
   <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
```

- ***Une partie de la Syntaxe de l'ontologie Nalt.owl***


```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY obo "http://purl.obolibrary.org/obo/" >
    <!ENTITY dc "http://purl.org/dc/elements/1.1/" >
    <!ENTITY OBO_REL "http://purl.org/obo/owl/OBO_REL#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY skos "http://www.w3.org/2004/02/skos/core#" >
    <!ENTITY ro "http://www.obofoundry.org/ro/ro.owl#" >
    <!ENTITY resource "http://semanticscience.org/resource/" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY oboInOwl "http://www.geneontology.org/formats/oboInOwl#" >
]>


<rdf:RDF xmlns="http://semanticscience.org/ontology/cheminf-core.owl#"
     xml:base="http://semanticscience.org/ontology/cheminf-core.owl"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:ro="http://www.obofoundry.org/ro/ro.owl#"
     xmlns:resource="http://semanticscience.org/resource/"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:obo="http://purl.obolibrary.org/obo/"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:OBO_REL="http://purl.org/obo/owl/OBO_REL#"
     xmlns:oboInOwl="http://www.geneontology.org/formats/oboInOwl#"
     xmlns:skos="http://www.w3.org/2004/02/skos/core#">
    <owl:Ontology rdf:about="http://semanticscience.org/ontology/cheminf-core.owl">
        <owl:versionInfo rdf:datatype="&xsd;string">1.0.1</owl:versionInfo>
        <dc:contributor rdf:datatype="&xsd;string">Colin Batchelor</dc:contributor>
        <dc:contributor rdf:datatype="&xsd;string">Egon Willighagen</dc:contributor>
        <dc:contributor rdf:datatype="&xsd;string">Janna Hastings</dc:contributor>
        <dc:contributor rdf:datatype="&xsd;string">Michel Dumontier</dc:contributor>
```

```
    <owl:AnnotationProperty rdf:about="&dc;language"/>
<!-- http://purl.org/dc/elements/1.1/rights -->

    <owl:AnnotationProperty rdf:about="&dc;rights"/>
    <!-- http://purl.org/dc/elements/1.1/title -->

    <owl:AnnotationProperty rdf:about="&dc;title"/>
    <!-- http://semanticscience.org/ontology/cheminf-core.owl#short_name -->

    <owl:AnnotationProperty rdf:about="http://semanticscience.org/ontology/cheminf-core.owl#short_name"/>
    <!-- http://www.geneontology.org/formats/oboInOwl#hasAlternativeId -->

    <owl:AnnotationProperty rdf:about="&oboInOwl;hasAlternativeId"/>
    <!-- http://www.w3.org/2000/01/rdf-schema#comment -->

    <owl:AnnotationProperty rdf:about="&rdfs;comment"/>
    <!-- http://www.w3.org/2000/01/rdf-schema#label -->

    <owl:AnnotationProperty rdf:about="&rdfs;label"/>
    <!-- http://www.w3.org/2002/07/owl#versionInfo -->

    <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>
    <!-- http://www.w3.org/2004/02/skos/core#prefLabel -->

    <owl:AnnotationProperty rdf:about="&skos;prefLabel"/>
```