République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de Guelma

Faculté des Mathématiques, d'Informatique et des Sciences de la Matière

Mémoire de fin d'étude MASTER

Département de : Informatique

Spécialité : Ingénierie des Médias

Application des Colonies de Fourmis pour l'Optimisation d'Alignement des Ontologies

Présenté Par :

Kribes Manel

Rouibi Hassen

Sous la Direction de :

Mme Djakhdjakha Lynda

Juin 2011

Remerciements

Ce mémoire représente l'achèvement de plusieurs années d'un long travail qui n'aurait pu voir le jour sans la participation, l'aide, les conseils, ou encore la présence de nombreuses personnes

Nous tenons à remercier en premier lieu ALLAH le tout puissant de nous avoir prodigué du courage, de patience et de santé pour le parachèvement de ce modeste travail.

Et nous voudrions exprimer toute notre gratitude :

A notre encadreur Mme Djakhdjakha Lynda pour la confiance qu'elle nous a témoignée en acceptant de diriger ce travail et pour avoir mis à notre disposition ses compétences et ses conseils et surtout son encouragement afin de mener à une meilleure maîtrise du sujet

A tous les membres de jury pour l'honneur qu'ils nous font en acceptant de juger ce travail et en participant à l'évaluation de sa valeur scientifique.

A Mr M.Hadjris enseignant au sein du département d'informatique pour son aide et leur précieux conseils à fin d'enrichir cette expérience.

A tous les enseignants du département d'informatique, qui ont assisté à nos débuts en informatique, et à tous les enseignants qui ont corroboré à notre formation aussi bien primaire qu'universitaire.

A BOUDRA Meryem pour son soutien de lion.

Enfin, à tous ceux de prés ou de loin qui ont contribués à ce travail par leur remarques, leurs suggestions, et leurs soutiens

Manel & Hassen

Résumé

Résumé

L'alignement d'ontologies représente un grand intérêt dans le domaine de la gestion des connaissances hétérogènes. Son objectif est de permettre une utilisation conjointe de plusieurs ontologies. Il s'agit généralement de construire des *appariements* entre les éléments décrits dans différentes ontologies. Le résultat de la tâche d'alignement est un ensemble de paire d'entités, qui sont des correspondances entre les ontologies. Ce résultat peut être employé pour effectuer plusieurs autres tâches émergeant dans le web sémantique telles que la fusion, la gestion des versions d'une ontologie, le partage de la connaissance, l'intégration sémantique…

Dans ce mémoire nous proposons d'utiliser les colonies de fourmis pour optimiser le résultat d'alignement des ontologies. A l'origine, la plupart des méta-heuristiques et en particulier les colonies de fourmis ont été développées pour résoudre les problèmes d'optimisation combinatoire mais de nos jours, beaucoup de recherches ont pris comme objet l'adaptation de ces méthodes aux autres types de problèmes.

L'idée de notre travail, est de représenter les ontologies à comparer par des graphes, de mettre en correspondances les entités des deux graphes et de chercher par la suite le meilleur appariement parmi les appariements multivoques des nœuds des graphes, où chaque nœud du graphe de la première ontologie peut être apparié à zéro, à un ou à plusieurs nœuds du graphe de la deuxième ontologie.

Mots clés : Web sémantique**,** Ontologie, hétérogénéité des données, alignement, graphes, colonies de fourmis, optimisation.

Sommaire

Sommaire

Chapitre II : *Alignement des Ontologies*

Chapitre III : *Appariement des Graphes*

Chapitre IV : *Conception de L'application*

Chapitre V : *Implémentation de L'application*

Table des Figures

Table Des Figures

Liste des Tableaux

Liste Des Tableaux

Introduction Générale

Introduction générale

Contexte Général

Le Web sémantique sera-t-il la génération suivante du World Wide Web d'aujourd'hui ? Nous assistons ces dernières années à l'émergence de nouvelles applications ayant besoin de partager de l'information entre différents systèmes comme c'est le cas pour l'apprentissage enrichi par les technologies et d'autres applications de différents domaines. L'enjeu est de développer des techniques facilitant l'interopérabilité sémantique entre ces systèmes d'informations, qui constituent généralement des sources de données autonomes et hétérogènes.

Actuellement le Web contient plus de 4.2 milliards de pages, mais la grande majorité d'entre elles sont dans un format lisible et compréhensible uniquement pour l'homme. Par conséquent les machines ou plutôt les logiciels ne peuvent ni comprendre ni traiter cette information. De plus, une grande partie du Web reste jusqu'à présent inexploitée. Pour pallier cette insuffisance du Web, les chercheurs ont créé la vision du Web sémantique.

Le Web sémantique est une prolongation du Web actuel dans lequel l'information permet de donner une signification bien définie, afin que les ordinateurs et les personnes puissent travailler et coopérer entre eux efficacement. La vision du Web sémantique a été présentée la première fois par Tim Berners-Lee.

L'utilité du Web sémantique peut être vue au travers des exemples suivants : supposons que nous cherchions à comparer les prix des canapés qui se fabriquent dans notre région (c.-à-d. le même code postal que le notre), ou que nous cherchions les catalogues en ligne des différents fabricants de pièces de rechange d'une voiture de marque Peugeot 406. Les réponses à ces questions peuvent être disponibles sur le web, mais pas sous une forme exploitable par la machine. Nous avons toujours besoin d'une personne capable de discerner et de nous donner la signification de ces réponses et de leur importance par rapport à nos besoins.

L'idée du Web sémantique est de rendre le contenu des ressources diffusées sur le Web accessible aux programmes informatiques de façon à ce que ceux-ci soient mieux à même d'aider les utilisateurs humains. Il s'agit de décrire ces ressources selon une représentation formelle,

c'est-à-dire basée sur une sémantique clairement définie et qui soit interprétable par des programmes. À la base de l'infrastructure du Web sémantique, se trouvent *les ontologies*.

Une ontologie permet de décrire un domaine en définissant son vocabulaire et les axiomes qui le régissent. Les travaux actuels utilisent pour ce faire des formalismes proches des objets comme les logiques de descriptions *[Baader et al, 1991]*. Dans ce cadre, une ontologie définit un ensemble de concepts et les relations qu'ils entretiennent avec d'autres concepts par spécialisation ou au travers de propriétés. Ainsi, les diagrammes UML de la figure 1 peuvent être considérés comme des (fragments d') ontologies.

Dans de nombreux contextes applicatifs, tel que l'échange de documents sur le Web, plusieurs ontologies couvrant, totalement ou partiellement, un même domaine se trouvent impliquées. Pour permettre l'interopérabilité des applications et/ou agents qui s'appuient sur une des ces ontologies, l'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être résolue. À cette fin, les liens sémantiques entre entités appartenant à deux ontologies différentes doivent être établis, c'est le but de l'alignement d'ontologies *[Euzenat et Shvaiko, 2007]*.

Problématique

L'alignement de deux ou plusieurs ontologies est définie comme une tâche qui consiste à chercher des rapports entre entités des ontologies en question. Dans la littérature, il existe aujourd'hui de nombreuses méthodes permettant l'alignement des ontologies, ces méthodes sont basées sur des techniques très variées et obtiennent des performances très différentes en fonction des caractéristiques des ontologies à aligner. La plupart de ces méthodes se basent sur la stabilité de la mesure de similarité en utilisant un seuil défini par l'utilisateur.

Donc, ce n'est plus le manque des méthodes d'alignement qui pose problème mais, de pouvoir détecter l'ensemble d'alignement optimal et le plus similaire. Les méthodes proposées requièrent toutes des temps de calcul important. Il est nécessaire d'appliquer des algorithmes d'optimisation exploitant la structure de graphe particulière des ontologies pour accélérer la

détection des correspondances, optimiser le résultat d'alignement et améliorer les performances de ces méthodes.

Objectifs

Notre contribution dans ce mémoire, consiste en l'optimisation des résultats d'alignement d'ontologies. Le but de notre travail est d'adapter un algorithme des Colonies de Fourmis pour exploiter beaucoup plus les différentes structures de voisinage, et d'avoir un appariement optimal, et plus similaire.

Donc l'objectif est double, nous proposons dans un premier temps, de représenter les ontologies à comparer par des graphes et de mettre en correspondance les entités des deux graphes. Ensuite, nous essayons d'optimiser le résultat d'alignement et de chercher du meilleur appariement.

Plan du Mémoire

Nous expliquons dans ce qui suit comment les objectifs ci-dessus ont été traités par notre étude.

Chapitre1: Réaliser un état de l'art dans le domaine des ontologies (définitions, type d'ontologies, modèle de représentation,construction des ontologies).

Chapitre2: Réaliser un état de l'art sur le concept d'alignement les différents algorithmes d'alignement d'ontologies existants.

Chapitre3: Approfondir l'état de l'art sur l'appariements des graphes et les algorithmes utilisés pour résoudre le ce problème (colonies de fourmis,..)

Chapitre4: Présenter notre contribution à savoir l'optimisation du résultat d'alignement en utilsant un algorithme de colonie de fourmis.

Chapitre5: Réalisation logicielle de l'approche proposée avec une discussion et une comparaison des résultats obtenus.

Ce travail s'achève par une conclusion générale.


Chapitre
I
Ingénierie Ontologique

Introduction

Le web sémantique est la vision du web de demain, son but est de rendre explicite le contenu sémantique des ressources et des connaissances et de les rendre compréhensible et manipulable par les machines. Dans le noyau du Web sémantique, les *ontologies* permettent de capitaliser, de représenter, d'exploiter et de partager sémantiquement des connaissances et des informations.

Pour atteindre l'objectif de ce mémoire qui est l'application des colonies de fourmis pour l'alignement des ontologies, nous focalisons dans ce chapitre sur l'ontologie, qui est une composante de base dans notre contribution.

Dans ce chapitre, nous présentons un état de l'art sur les ontologies. D'abord nous donnons les différentes définitions qui sont attribuées à la notion d'ontologies, les différents éléments dont elle est composée, les différents niveaux de classification des ontologies. Ensuite, nous décrivons le processus et méthodologies servant leur construction, ainsi que les principaux formalismes utilisés pour servir leur représentation et les outils de développement d'ontologies y compris les langages de spécification. Finalement nous montrerons quelques domaines d'application des ontologies et notamment ces approches d'intégration.

1.1 Définition des ontologies

Une ***Ontologie*** est l'ensemble structuré des concepts et des relations représentant le sens d'un domaine donné. Elle est étudiée dans le domaine de l'intelligence artificielle, et permette la représentation des connaissances, et elle est aussi évoquée dans le Web sémantique. En effet**,** Il existe plusieurs définitions ou significations attribuées à ce concept. Les définitions les plus communes sont celles de (Neches et al 1991, Gruber 1993, Guarino1995) :

- La première définition de l'ontologie dans le domaine de l'informatique est proposée par Neches et al, ils définissent l'ontologie comme :« *les termes et les relations de base comportant le vocabulaire d'un domaine aussi bien que les règles pour combiner les termes et les relations afin de définir des extensions du vocabulaire* ».*[Neches et al,1991]*
- En 1993, Gruber proposait la définition la plus citée, il définit l'ontologie comme étant:« *une spécification explicite d'une conceptualisation* », *[Gruber, 1993]*.

La conceptualisation est le résultat d'une analyse du domaine étudiée et l'abstraction du monde de ce domaine.

- N. Guarino affine la définition de T. Gruber en considérant les ontologies comme des spécifications partielles et formelles d'une conceptualisation *[Guarino et Giaretta, 1995]*.Les ontologies sont ***formelles :*** car exprimées sous un formalisme doté d'une sémantique formelle, et ***partielles :*** car une conceptualisation ne peut pas toujours être entièrement formalisée dans un tel cadre, du fait d'ambiguïtés ou du fait qu'aucune représentation de leur sémantique n'existe dans le langage de représentation choisi.

1.2 Les Composants d'une Ontologie

Toutes les définitions citées dans *[Mihoubi, 2000]* et *[Actes, 1999]* s'accordent sur le fait qu'une ontologie est formée par des ***concepts*** et des ***relations*** entre ceux-ci et qu'elle veut permettre de spécifier les connaissances d'un domaine, de façon aussi indépendante que possible du type de manipulations qui vont être opérées sur ces connaissances.

Alors, une ontologie peut être vue comme un treillis de concepts et de relations entre ces concepts, ainsi que des propriétés et des ***axiomes***, destinés à représenter les objets du monde sous une forme compréhensible aussi bien par les hommes que par les machines. Ainsi, les ***instances*** sont utilisées pour représenter des exemples particuliers de concepts.

L'exemple ci-dessous (*Figure 1.1*) représente un petit extrait simplifié d'une ontologie des catastrophes naturelles, avec deux types de relations.

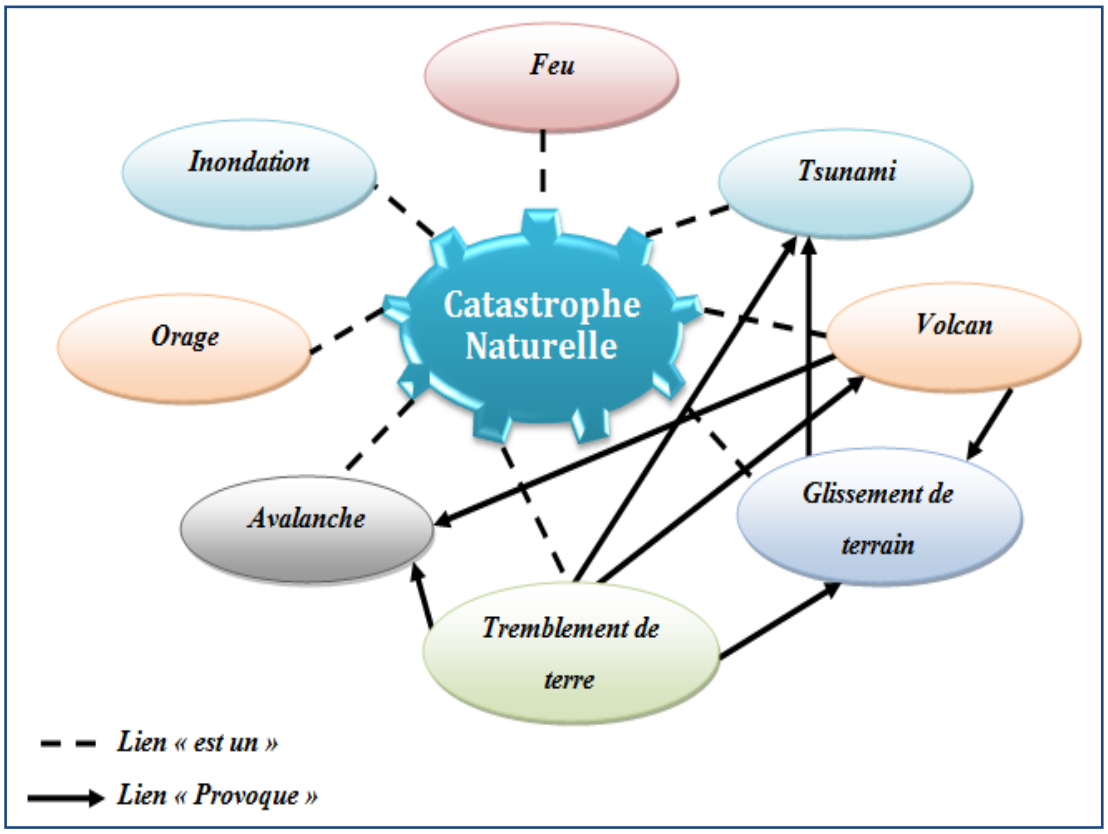


Figure 1.1 : *Exemple d'une Ontologie*

1.3 Classification des Ontologies

Les ontologies peuvent être classifiées selon plusieurs critères afin de déterminer une typologie d'ontologies:

1.3.1 Classification selon l'objet de Conceptualisation

Cette typologie dispose les types d'ontologies suivants :

- ✓ ***Les ontologies de représentation des connaissances*** regroupent les concepts impliqués dans la formalisation des connaissances. L'ontologie la plus citée dans ce contexte est *l'ontologie de* ***FRAME***. Elle intègre les primitives de représentation des langages à base de frames : classes, instances, facettes, propriétés/slots, relations, restrictions, valeurs permises, etc.
- ✓ ***Les ontologies supérieures ou de Haut niveau*** visent à étudier les catégories des choses qui existent dans le monde, comme les concepts de haut niveau d'abstraction tels que les

entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations et les propriétés.

- ✓ ***Les ontologies génériques*** sont appelées, également, des méta-ontologies ou *"Core ontologies*". Elles décrivent des concepts génériques moins abstraits que ceux décrits par des ontologies supérieures.
- ✓ ***Les ontologies de domaine*** décrivent le vocabulaire lié à des domaines particuliers comme la physique, la mécanique, la chimie, la médecine et la modélisation d'entreprise.
- ✓ ***Les ontologies d'applications*** dépendent, à la fois d'un domaine particulier et d'une tâche spécifique. Elles ont un domaine de validité restreint et correspondent à l'exécution d'un ensemble de tâches composant l'application. Généralement, les ontologies d'application ne sont pas réutilisables et possèdent donc un intérêt plus limité. *[Mellal, 2007]*

1.3.2 Typologie Selon le Niveau de Détail

Le deuxième type de classification est par rapport au niveau de détail utilisé lors de la conceptualisation de l'ontologie en fonction de l'objectif opérationnel visé par cette dernière. Dans cette typologie, deux types de granularité ont été distingués :

- ✓ Quand les ontologies sont très détaillées au niveau du vocabulaire utilisé, qui est plus riche, on parle de ***granularité fine***. Souvent, les ontologies de domaine, les ontologies de tâches et les ontologies d'applications représentent des ontologies à granularité fine.
- ✓ La ***granularité large*** concerne le cas où les ontologies sont moins détaillées. Un exemple est celui des ontologies de haut niveau.

1.3.3 Typologie Selon le Niveau de Complétude

Cette typologie a été introduite par l'identification de trois engagements correspondant aux étapes de la modélisation des connaissances : ***Un engagement sémantique*** qui fixe le sens linguistique des concepts, un ***engagement ontologique*** qui détermine le sens formel des concepts et enfin un ***engagement computationnel*** déterminant leur exploitation effective. *[Mellal, 2007]*

1.3.4 Typologie Selon le Niveau du Formalisme de Représentation

Dans la quatrième dimension de classification, *[Uschold et Gruninger, 1996]* ont distingué les quatre genres d'ontologies selon le niveau de formalisme de représentation de langage employé pour les mettre en application. Ce sont:

- ✓ ***Ontologies informelles*** exprimées dans un langage naturel.
- ✓ ***Ontologies semi-Informelles*** décrites à l'aide d'un langage naturel structuré et limité.

- ✓ ***Ontologies semi-formelles*** spécifiées dans un langage artificiel défini formellement.
- ✓ ***Ontologies formelles*** basées sur un langage artificiel contenant une sémantique formelle, ainsi que des théorèmes et des preuves de propriétés telles la robustesse et l'exhaustivité.

1.4 Construction des Ontologies

Le processus de construction d'ontologies, appelé *ingénierie ontologique*, peut être décrit selon les principes qui le gouvernent, et les méthodologies et les outils qui le soutiennent.

1.4.1 Principes de Construction d'une Ontologie

Le processus de construction d'une ontologie doit respecter certains principes de bases qui permettent d'aboutir à une ontologie capable de répondre aux objectifs visés par celle-ci. Le concepteur de l'ontologie, se doit donc de garder à l'esprit ces principaux critères tout au long du cycle de développement de son ontologie *[Gruber, 1993]* :

- ✓ ***La clarté et l'objectivité*** **:** l'ontologie doit fournir le sens des termes définis en offrant des définitions objectives ainsi que de la documentation associée en langage naturel.
- ✓ ***L'exhaustivité*** **:** une définition exprimée par une condition nécessaire et suffisante est préférable à une définition exprimée seulement par une condition nécessaire ou par une condition suffisante.
- ✓ **La cohérence :** (*consistance logique*) des axiomes, afin de pouvoir formuler des inférences cohérentes avec les définitions.
- ✓ **L'extensibilité monotone maximale :** les nouveaux termes, devraient être inclus dans l'ontologie sans entraîner de modifications dans les définitions existantes.
- ✓ **L'intervention ontologique minimale :** l'ontologie devrait spécifier le moins possible la signification de ses termes, donnant aux parties qui s'engagent dans cette ontologie la liberté de spécialiser et d'instancier l'ontologie comme elles le désirent.

1.4.2 Cycle de Développement d'une Ontologie

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, alors leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. En particulier, les ontologies doivent être considérées comme des objets techniques évolutifs et possédants un cycle de vie qui nécessite d'être spécifié.

Les activités liées aux ontologies sont, d'une part, des activités de gestion de projet (planification, contrôle, assurance qualité), et, d'autre part, des activités de développement (spécification, conceptualisation, formalisation), s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration *[Blazquez et al, 1998]*. Il en ressort que le cycle de développement général d'une ontologie se déroule en trois principales étapes qui sont : la ***conceptualisation***, la ***formalisation*** (appelée aussi ***ontologisation***) et ***l'opérationnalisation***. Ces étapes sont généralement précédées par une étape ***d'évaluation des besoins*** et de délimitation du domaine de connaissances à modéliser, et avant d'être livrée aux utilisateurs, l'ontologie doit être ***évaluée***.

1.4.3 Méthodes de Construction des Ontologies

A l'heure actuelle, Il n'y a pas encore un consensus sur une méthodologie fixe pour la construction des ontologies. Plusieurs méthodologies ont été proposées dont les plus connues sont :

A. La Méthode ENTREPRISE

Proposée et fondée sur l'expérience de la construction de *l'Entreprise Ontology* par Uschold et Grüninger *[Uschold et Grüninger, 96]*. Comme la méthode est générique, ses étapes sont considérées comme la base d'un processus standard de construction. Ce processus opère selon quatre étapes fondamentales (voir *Figure1.2*) *[Mellal, 2007]*

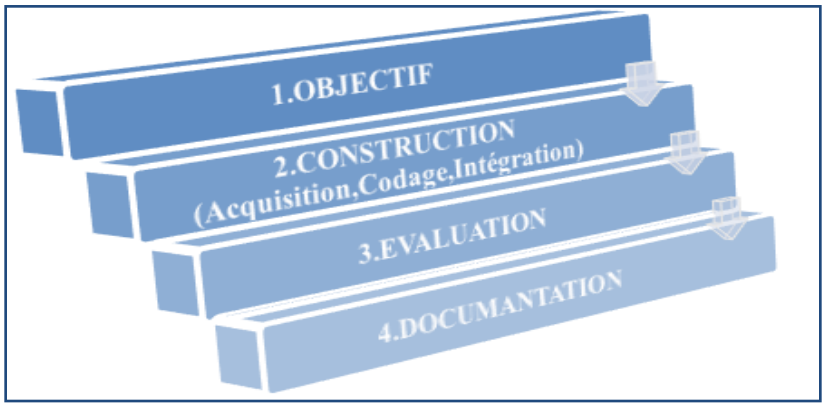


Figure 1.2 : *Etape de Constriction d'une ontologie*

1. ***L'identification de l'objectif*** **:** permet d'identifier, en termes généraux, l'objectif, la portée et les limitations de l'ontologie à construire.
2. ***La création de l'ontologie*** : étape la plus longue et la plus difficile, contient elle-même trois sous-étapes :

- ✓ *L'acquisition des connaissances* sert à définir les concepts dans un domaine donné et les relations entre eux, de manière à ne pas être ambiguës.
- ✓ *Le codage*, une fois les concepts et leurs relations acquises, permet de représenter l'ontologie dans un langage formel. La formalisation de l'ontologie peut être de différents degrés (Très informelle, Semi-informelle, Semi-formelle, Rigoureusement formelle).
- ✓ *L'intégration des ontologies existantes* est l'étape qui permet de réutiliser les concepts déjà définis dans des ontologies existantes.

3. ***L'évaluation de l'ontologie*** : l'évaluation d'une ontologie est définie à propos de quelques critères suivants : clarté, cohérence, …
4. ***La documentation*** permet de renseigner les ontologies, leurs concepts importants ainsi que leurs objectifs. *[Mellal, 2007]*

B. La Méthode METHONDOLOGY *[Fernandez et al, 1997]*

Elle a été développée au sein du groupe d'ontologie à l'université polytechnique de Madrid. Elle couvre tout le cycle de vie d'une ontologie :

- ✓ ***Cadrage***
- ✓ ***Conceptualisation***
- ✓ ***Implémentation***

La méthode METHONDOLOGY s'inspire d'une méthode de développement de Système à base de connaissances.

Il existe aussi plusieurs outils pour développer et maintenir une ontologie. Dans la suite, nous introduisons certains outils de développement qui sont largement utilisés par la communauté des onto-logistes.

OILEd[1] : Est un éditeur graphique développé par l'Université de Manchester, il permet à l'utilisateur de construire des ontologies en langage DAML+OIL.

OntoEdit[2] **:** Est un projet développé à l'Université de Karlsruhe qui fournit les méthodologies et processus d'*intégration* d'ontologie ainsi que l'éditeur d'ontologie *[OntoEdit, 2002]*.

[1] http://oiled.man.ac.uk/

[2] http://www.ontoknowledge.org/tools/ontoedit.shtml

Protégé[3]: Est un outil d'édition d'ontologie utilisé largement aujourd'hui pour élaborer des ontologies en RDF(S) et OWL, développé à l'Université de Stanford. Protégé fournit un environnement de développement graphique et interactif pour aider les ingénieurs et les experts du domaine à réaliser des tâches plus facilement. *[Noy et al, 2000]*

WebODE[4] : Est développé à l'Université technique de Madrid, est un *benchmark* supportant l'ingénierie de l'ontologie. Il est construit sur un serveur d'application, qui fournit une haute extensibilité et rentabilité en permettant d'ajouter facilement de nouveaux services et fonctionnalités.

1.5 Les Modèles de Représentation des Ontologies

Une ontologie, a besoin d'être représentée formellement. Plus encore, elle doit représenter l'aspect sémantique des relations liant les concepts. A cet effet, de nombreux formalismes ont été développés.

1.5.1 Les réseaux sémantiques

Un réseau sémantique est une structure de graphe qui encode les connaissances ainsi que leurs propriétés. Les nœuds du graphe représentent des objets (concepts, situations, événements, etc) et les arcs expriment des relations entre ces objets. Ces relations peuvent être des liens " ***sorte - de*** " exprimant la relation d'inclusion ou des liens " ***est-un*** " représentant la relation d'appartenance. Par exemple, on peut dire que *Volkswagen* est une marque de *voiture* (voir *Figure 1.3*), comme on peut dire que le *Busard* est un *rapace* qui est une sorte *d'Oiseau* (voir *Figure 1.4*). *[Mellal, 2007]*

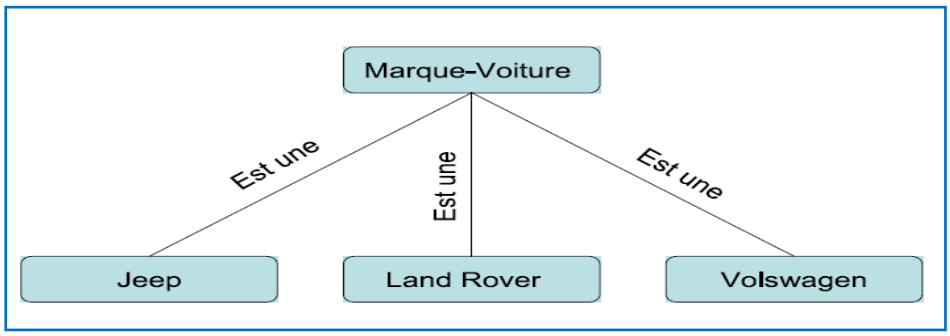


Figure 1.3 : *Exemple de réseau sémantique utilisant la relation " est-un "*

[3] http://protege.stanford.edu/

[4] http://kw.dia.fi.upm.es/wpbs/

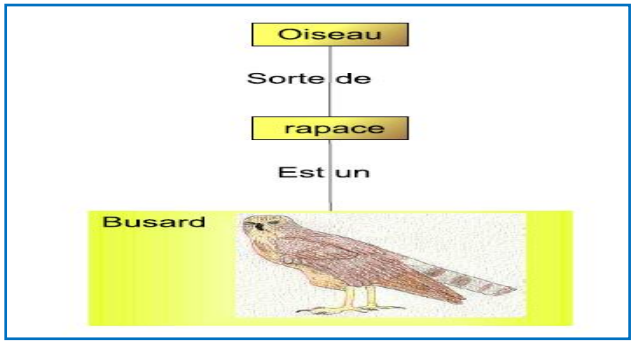


Figure 1.4 : *Exemple de réseau sémantique utilisant la relation " sorte-de "*

Parmi les réseaux sémantiques, très répandus pour la conceptualisation des ontologies, on trouve ***les graphes conceptuels*** dont le but fondamental est d'être " *un système de logique hautement expressif, permettant une correspondance directe avec la langue naturelle* ". Ce type de graphes constitue un formalisme général de représentation de connaissances fondé sur la logique.

1.5.2 Les schémas (Frame)

La notion de " schéma " (ou frame) est une structure de données complexe. Il est considéré comme un prototype décrivant une situation ou un objet standard. Il sert de référence pour comparer des objets que l'on désire reconnaître, analyser ou classer. Les prototypes doivent prendre en compte toutes les formes possibles d'expression de la connaissance. Un schéma, comme le montre la *Figure 1.5*, est caractérisé par des *attributs*, des *facettes* et des *relations*. *[Mellal, 2007]*

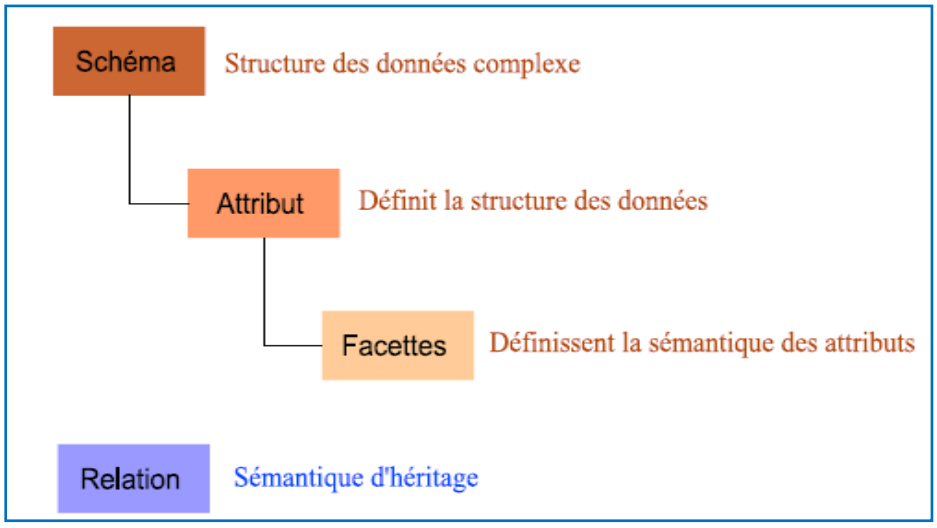


Figure 1.5 : *Eléments Caractérisant un Schéma*

1.5.3 Les Scripts

La notion de " script " (ou scénario) a été introduite par Schank et Abelson *[Schank et Abelson, 1988]*, sur le modèle des schémas pour le traitement du langage naturel.
Un script est donc une structure de données qui regroupe des connaissances relatives à une situation et qui permet de combiner des représentations. Il peut être vu comme un ensemble d'actions élémentaires ou de références à d'autres scénarios, ordonnées selon leur déroulement dans le temps.

1.5.4 Les Logiques de Description (LD)

Les logiques de description sont des formalismes logiques de représentation de connaissances. Elles sont issues d'un couplage entre la théorie des frames et les principes des réseaux sémantique, et offrent un modèle à mi-chemin entre ces deux théories.

Une ontologie formalisée à l'aide des logiques de description, permet de décrire les concepts d'un domaine à travers des *concepts* atomiques correspondant à des prédicats unaires spécifiant les objets du domaine, et des *rôles* atomiques, correspondant à des prédicats unaires décrivant les relations entre les concepts du domaine. Ces rôles sont spécifiés à l'aide de constructeurs fournis par le langage formel de description logique.

Les logiques de description combinent des représentations intentionnelles et extensionnelles des connaissances *[Baader et al, 1991]* :

- ✓ Une **TBox** (T pour terminologique) correspond au niveau descriptif, contient les déclarations des primitives conceptuelles organisées en concepts et rôles. Ces déclarations permettent de décrire les concepts en fonction d'autres concepts à partir des relations et des contraintes sur ces relations.
- ✓ Une **ABox** (A pour assertionnel) correspond au niveau assertions, contient les déclarations d'individus, instances des concepts définis dans la TBox.

La figure suivante représente un exemple d'une base de connaissances composée de TBox et ABox, représentée en logique de description

(Ax1) Tarte ⊑ PréparationCulinaire	(Ax7) Pomme ⊓ Noix ⊑⊥
(Ax2) Dessert ⊑ PréparationCulinaire	(Ax8) TarteSucrée ≡ Tarte⊓∃ingrédient.ProduitSucrée
(Ax3) ProduitSucré ⊑ Produit	(Ax9) TarteSalée ≡ Tarte ⊓ ¬TarteSucrée
(Ax4) Fruit ⊑ ProduitSucré	(Ax10) TarteSucrée ⊑ Dessert
(Ax5) Pomme ⊑ Fruit	
(Ax6) Noix ⊑ Fruit	
(a)*TBox*	
(A1) (Tarte⊓∃ingrédient.Pomme)(tarte1)	(A3) ingrédient(tarte1,noix1)
(A2) Noix(noix1)	
(b)*ABox*	

Figure 1.6: *Une partie d'une ontologie écrite en LD*

1.6 Vue D'ensemble des Langages D'ontologies

Il existe de nombreux langages informatiques, plus ou moins récents, spécialisés dans la création et la manipulation des ontologies (XML, Loom, RDF, OWL...Etc.). Nous en décrivons quelques-uns dans la suite.

1.6.1 RDF

Créé en 1999. RDF, fondé sur le langage XML, et permettant de décrire des métadonnées et facilitant leur traitement. Propose de représenter des données structurées non pas sous forme d'arbre, mais sous forme de graphe. (*Figure 1.7*)

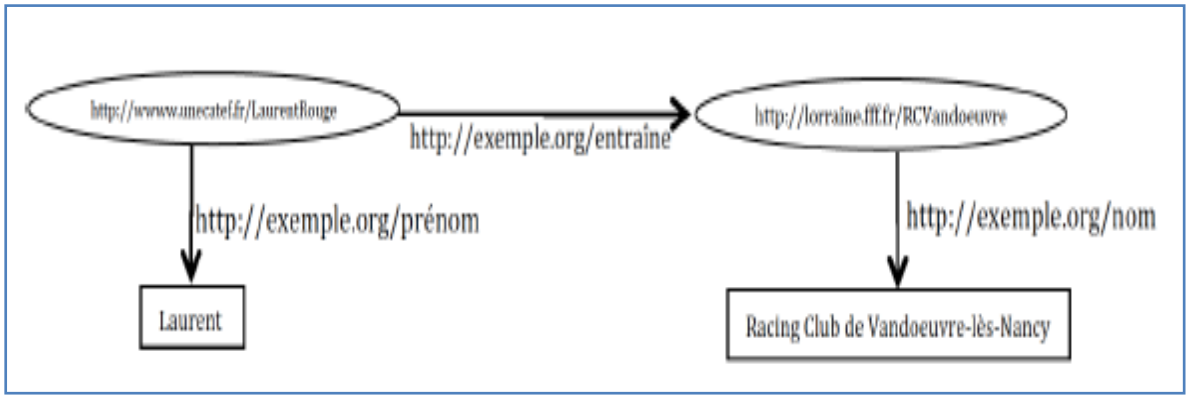


Figure 1.7 : *Exemple de graphe RDF*

En effet, la structure fondamentale de toute expression en RDF est une collection de triplets, chacun composé d'un sujet, un prédicat et un objet. Ceci peut être illustré par un diagramme composé de nœuds et d'arcs dirigés, dans lequel chaque triplet est représenté par un lien nœud-arc-nœud. *[Xavier Lacot, 2005]*

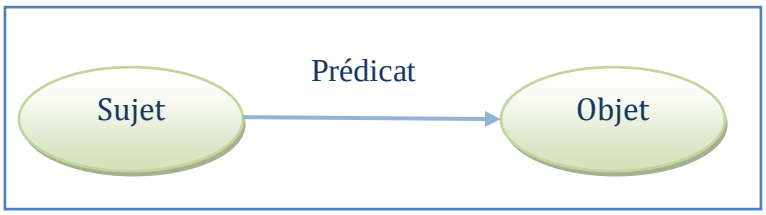


Figure 1.8 : *Exemple d'illustration Triple RDF*

1.6.2 RDF Schéma

L'information transmise par le triplet RDF est suffisamment triviale pour comprendre sa signification, Il est donc nécessaire, de donner un sens aux informations stockées sous forme de triplets RDF, de se donner un vocabulaire, de définir la signification de la propriété, ainsi que son type, son champ de valeurs, etc. C'est le rôle de *RDF Schéma*, qui permet de créer des vocabulaires de métadonnées. *[Xavier Lacot, 2005]*

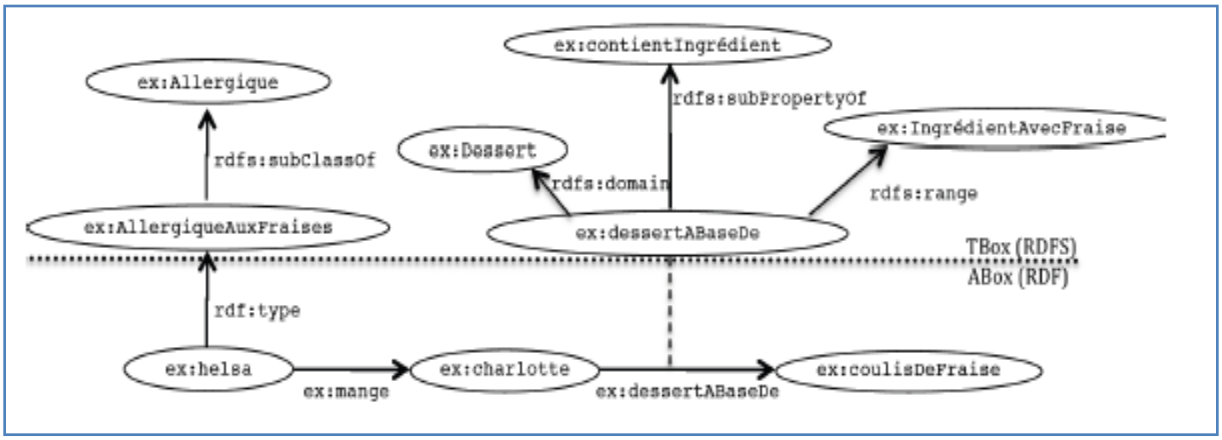


Figure 1.9 : Exemple de graphe RDFs

1.6.3 OWL : Web Ontology Language

Une autre famille de langages de représentation de connaissances est construite sur XML et les logiques de descriptions (LD). Ces langages sont ainsi compatibles avec des normes standard existantes du Web et en même temps offrent la sémantique des logiques de description. Le principal représentant de cette famille de langages est OWL. OWL est devenu une recommandation de W3C en février 2004.

OWL se différencie du couple RDF/RDFS en ceci que, contrairement à RDF, il est justement un langage d'ontologies. Si RDF et RDFS apportent à l'utilisateur la capacité de décrire des classes et des propriétés, OWL intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc.

Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grâce à un vocabulaire plus large et à une vraie sémantique formelle. *[Xavier Lacot, 2005].*

Le langage OWL se compose de trois sous langages qui proposent une expressivité croissante, chacun conçu pour des communautés de développeurs et des utilisateurs spécifiques : OWL Lite, OWL DL et OWL Full. Chacun est une extension par rapport à son prédécesseur plus simple.

• **OWL Lite** : supporte les utilisateurs ayant besoin principalement d'une hiérarchie de classification et des contraintes simples de cardinalités 0 ou 1. Cette cardinalité correspond à des relations fonctionnelles, par exemple une personne a une adresse. Toutefois, cette personne peut avoir un ou plusieurs prénoms. OWL Lite ne suffit donc pas pour cette situation.

• **OWL DL** : D'après son nom OWL DL utilise la LD. Il supporte les utilisateurs qui réclament l'expressivité maximale tout en retenant la complétude informatique (toutes les conclusions sont garanties d'être calculables), et la possibilité de décision (les calculs finiront en un temps fini). Il inclut toutes les constructions du langage OWL, qui ne peuvent être utilisées que sous certaines restrictions.

• **OWL Full** : a été défini pour les utilisateurs qui veulent une expressivité maximale et une liberté syntaxique de RDF sans des garanties informatiques. OWL Full permet à une ontologie d'augmenter la signification du vocabulaire prédéfini (RDF ou OWL). L'inconvénient majeur de OWL Full est d'avoir un haut niveau de capacité de description, quitte a ne pas pouvoir garantir la complétude et la décidabilité des calculs lies a l'ontologie.

1.7 L'utilisation des Ontologies

Les ontologies peuvent servir à améliorer des applications existantes basées sur le Web et permettre de nouvelles utilisations du Web. Dans cette section, on décrit des cas d'utilisation représentatifs d'ontologies. Remarquez que cette liste n'est pas exhaustive et il s'agit plutôt d'une vue transversale de d'autres cas d'utilisation intéressants.

1.7.1 Les Portails Web

Un portail web est un site Web qui offre un contenu d'informations concernant un thème commun, par exemple, une ville ou un domaine d'intérêt particulier. Un portail Web permet aux

personnes intéressées par ce thème de recevoir des nouvelles, de trouver et parler à quelqu'un, de bâtir une communauté et de trouver des liens vers d'autres ressources Web d'intérêt commun.
Un exemple de portail basé sur une ontologie est ***OntoWeb***. Ce portail sert la communauté universitaire et industrielle intéressée par la recherche sur les ontologies. Un autre exemple de portail utilisant les technologies du Web sémantique et pouvant tirer avantage d'un langage d'ontologie est le projet « The Open Directory Project », un annuaire complet et gigantesque du Web.

1.7.2 Les Collections Multimédias

Les ontologies peuvent servir pour l'annotation sémantique des collections d'images, de sons ou d'autres objets non textuels. Il est encore plus difficile pour les machines d'extraire une sémantique significative d'objets multimédias que d'un texte en langue naturelle. Ainsi, ces types de ressources sont habituellement identifiés par des légendes ou des balises de métadonnées. Par contre, puisque les personnes décriront peut-être différemment ces objets non textuels, il importe que les installations de recherche offrent plus qu'une simple correspondance de mots-clés. Idéalement, les ontologies devraient saisir une connaissance supplémentaire à propos du domaine pouvant servir à améliorer la récupération des images.

1.7.3 La Documentation d'un Concept

Ce cas d'utilisation correspond à un corps volumineux de documents techniques, comme ceux trouvés dans l'industrie aérospatiale. Cette documentation peut être de plusieurs types différents, comprenant la documentation de conception, la documentation de fabrication et la documentation d'essai. Ces ensembles de documents ont chacun une structure hiérarchique différente d'un ensemble à l'autre.
Les ontologies peuvent servir à bâtir un modèle informationnel permettant l'exploration de l'espace d'informations selon les éléments représentés, les associations entre éléments, les propriétés des éléments et les liens vers la documentation qui les décrit et définit.

1.7.4 L'intégration de Données

L'intégration de données est une problématique cruciale en informatique. Dans le domaine des systèmes d'information, les ontologies sont utilisées pour représenter, et manipuler les connaissances, et dans le cadre du Web sémantique, les ontologies sont évoquées pour collaborer entre les utilisateurs et les machines. De plus en plus, nous pouvons trouver dans la littérature plusieurs ontologies qui modélisent le même domaine. Cette multitude d'ontologies évoque un

problème d'hétérogénéité. Face à cette hétérogénéité, il est nécessaire de trouver des moyens de réconciliation des ontologies pour assurer leur interopérabilité et pouvoir les faire collaborer et permettre aux composants intelligents de les utiliser. Pour ce faire, Noy et Musen *[Noy et al, 1999]* distinguent deux approches pour unifier les terminologies des ontologies alignement et fusion d'ontologies. Les méthodes *d'alignement d'ontologies* établissent les différents types de correspondances entre les ontologies, par conséquent cette option préserve les ontologies en question (Cf. § chapitre 2). Cependant, les méthodes de fusion consistent à intégrer deux ontologies pour produire une nouvelle ontologie, regroupant les concepts, les relations et les instances des ontologies originales. La fusion des ontologies est nécessaire lorsque les modèles et les formalismes de représentation des ontologies sont uniformes. Préalablement à la fusion, il convient de déterminer quelle est l'ontologie la plus générale, ou celle qui est la plus étendue, c'est-à-dire celle qui ne sera pas modifiée. Les autres devront être alignées sémantiquement et syntaxiquement sur l'ontologie la plus générale. Le problème se ramène alors à l'intégration d'une ontologie dans une autre. La fusion de deux ontologies suppose la présence dans ces deux ontologies d'entités conceptuelles (concepts ou relations) communes. Une fois les deux ontologies exprimées dans le même formalisme et à travers le même modèle cognitif, ces entités communes aux deux ontologies doivent être identifiées.

Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les ontologies. D'abord nous avons donné les définitions importantes des ontologies. Ensuite nous avons décrit les différents composants des ontologies, et catégorisé les différents types d'ontologie. Ensuite, nous avons décrit les approches, méthodes, méthodologies et outils existants de construction d'ontologies. Puis nous nous sommes intéressés aux formalismes de représentation des ontologies, les modes de raisonnement sur ces formalismes et l'ensemble de langages d'ontologies. Finalement, nous avons vu les cas d'utilisation d'ontologies, surtout dans le domaine d'intégration des données et nous avons met le point sur les notions de fusion et d'alignement d'ontologie.

Après ce panorama des techniques qui constituent le paysage technique de cette thèse, le chapitre suivant se focalise sur les travaux de recherche autour de l'alignement d'ontologies, qui constitue une notion clé dans notre contribution.

Alignement des Ontologies

Introduction

En ce qui concerne l'intelligence artificielle, et plus particulièrement la représentation des connaissances et le raisonnement, l'interopérabilité apparaît comme une étape cruciale vers une unification de la sémantique des connaissances distribuées. Les ontologies sont précisément un des moyens contribuant à faciliter la compréhension des informations échangées entre les systèmes interopérables en essayant de standardiser la représentation des concepts et de leurs relations. La notion d'interopérabilité est également présente dans les ontologies. Elle consiste à pouvoir acquérir et intégrer des informations, des données et des services issus des ontologies hétérogènes. Cette hétérogénéité survient par exemple lorsque les ontologies sont implantées par différents formalismes de représentation, différents langages, etc.

Plusieurs approches existent pour mettre en œuvre cette interopérabilité. Parmi ces approches, on distingue l'alignement d'ontologies. Brièvement cette technique consiste à identifier des relations entre éléments de différentes ontologies.

Nous nous intéressons dans ce chapitre, a s'adressé au début au problème de l'hétérogénéité des ontologies, et au formalisme de la correspondance de ses dernier. Par la suite, on passe à définir le concept d'alignement, à identifier les différents techniques d'alignement d'ontologies, tout en citant quelques systèmes d'alignement. À la fin on termine par une comparaison entre ses méthodes.

2.1 Notion d'hétérogénéité

« ***Les ontologies ont été créées pour résoudre le problème de l'hétérogénéité des données sur le web, cependant elles sont devenues elles-mêmes source d'hétérogénéité*** » *[Euzenat et al, 2004]*.

Selon *[Euzenat et al, 2007]*, l'hétérogénéité est résumée sous quatre niveaux :

2.1.1 Le niveau syntaxique

Il s'agit de toutes les formes d'hétérogénéité relatives au choix du formalisme de représentation plutôt que le contenu. En effet il existe différentes façons de représenter les ontologies (OWL, KIF …) et chaque langage est basé sur une syntaxe différente. La solution de ce type d'hétérogénéité est simple, un mécanisme de réécriture des ontologies dans le même langage de représentation.

2.1.2 Le niveau terminologique

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

- Différents noms utilisés pour designer une même entité (synonymie)
- Le même nom utilisé pour designer deux entités distinctes (polysémie)
- Des mots provenant de différentes langues (Français, Anglais, Italien …) utilisés pour designer une même entité.
- Des variations syntaxiques du même mot (différentes prononciations, abréviations, utilisation des préfixes et des suffixes …)

2.1.3 Le niveau conceptuel

Appelé aussi hétérogénéité sémantique, c'est la différence dans l'interprétation du domaine qui a comme conséquences différents concepts ou différentes relations entre concepts. Les divergences à ce niveau peuvent être résumées en trois aspects :

- *La couverture* : la différence entre deux ontologies peut être au niveau de la portée de la couverture du domaine décrit. Elles peuvent couvrir des parties différentes (du monde réel ou d'un domaine) ou alors des parties qui se chevauchent.
- *La granularité* : deux ontologies peuvent décrire les mêmes entités avec des niveaux de détail différents.
- *La perspective* : deux ontologies peuvent décrire un domaine de deux points de vue différents. Par exemple : le concept de la chaleur chez un Norvégien sera forcement différent du même concept chez un Sénégalais.

2.1.4 Le niveau sémiotique ou pragmatique

Le niveau pragmatique est le niveau le plus complexe. Ce type d'hétérogénéité intervient lorsqu'il y a différence d'interprétation de la même ontologie par différents individus ou différentes communautés selon différents contextes. Ces différences d'interprétations sont souvent liées au choix du formalisme de représentation des connaissances.

2.2 Mise en Correspondances D'ontologies

Pour assurer l'interopérabilité, un certain nombre de techniques ont été proposées dans la littérature. Elles sont souvent utilisées pour permettre le partage des données entre des bases

de connaissances hétérogènes et pour la réutilisation des informations de ces bases. Dans la section suivante, nous distinguons les trois catégories principales qui sont :

- ✓ ***Correspondances ou Mappings*** : selon *[Klein, 2001]*, Les mappings sont des relations entre les éléments de deux représentations (ontologies, schémas de bases de données, etc.), indiquent une similarité relative selon une mesure donnée.
- ✓ ***Appariement ou Matching*** : Le matching d'ontologies est le processus de définition d'un ensemble de fonctions permettant de spécifier des ≪ correspondances » entre termes *[Xu et Embley, 2003]*, *[He et al, 2003]*.
- ✓ ***Alignement d'ontologies***: selon *[Kalfoglou et Schorlemmer 2003]*, l'alignement est l'établissement de *correspondances* binaires entre les concepts des deux ontologies afin de parvenir à un agrément. Il semble que *[Klein, 2001]* et *[Noy et Musen, 2000]* acceptent que les ontologies soient légèrement modifiées, en ce qu'ils demandent que l'alignement aboutisse à un ensemble « *pertinent et cohérent* ».
- ✓ ***Fusion d'ontologies*** : La fusion d'ontologies est le processus de création d'une seule ontologie rassemblant les connaissances de deux ou plusieurs ontologies existantes et différentes qui décrivent le même sujet ou appartiennent au même domaine d'application. L'ontologie générée inclut les informations de toutes les ontologies sources, mais est plus ou moins inchangée.
- ✓ ***Intégration d'ontologies*** : L'intégration d'ontologies est un processus de construction d'une nouvelle ontologie qui n'est pas forcement destinée à remplacer les autres (ces dernières peuvent continuer à être utilisées par ailleurs, a être mises a jour, a évoluer, etc.). Ces différentes ontologies peuvent être connexes.

2.3 Définition D'alignement

Dans un contexte d'alignement (de schémas de base de données, d'ontologies, etc.), on trouve différentes définitions pour l'alignement.

Définition1. Dans *[Rahm et Berstein, 2001]*, une méthode d'alignement est définie comme une fonction qui prend en entrée deux schémas *S1* et *S2* (eux-mêmes définis comme un ensemble d'éléments connectés par une structure quelconque) et qui retourne un appariement entre l'ensemble des éléments de *S1* vers celui de *S2*. Chaque élément de l'appariement exprime une relation entre un élément de *S1* et un élément de *S2*.

Définition 2. Dans *[Kalfoglou et Schorlemmer 2003]*, l'alignement est l'établissement de *correspondances* binaires entre les concepts des deux ontologies afin de parvenir à un agrément.

Définition 3. Une définition plus détaillée est donnée dans *[Euzenat et al, 2007]*, où l'alignement de structures est défini comme un processus de mise en correspondance sémantique des entités qui les composent. Le processus est exécuté selon une stratégie ou une combinaison de techniques de calcul de mesures de similarité et utilise un ensemble de paramètres (ex : paramètres de pondération, seuils …) et un ensemble de ressources externes (ex : thésaurus, lexique…). Au final, nous obtenons un ensemble de liens sémantiques reliant les entités qui composent les ontologies. Ces derniers comprennent des relations d'équivalence, de généralisation/spécialisation, de chevauchement ou encore d'incompatibilité.

On conclure que l'alignement d'ontologies est un processus de découverte des correspondances entre deux ou plusieurs ontologies. Il décrit comme une application, dont l'entrée est constituée d'un ensemble d'ontologies et la sortie, formée des correspondances entre ces ontologies (voir la *Figure 2.1*).

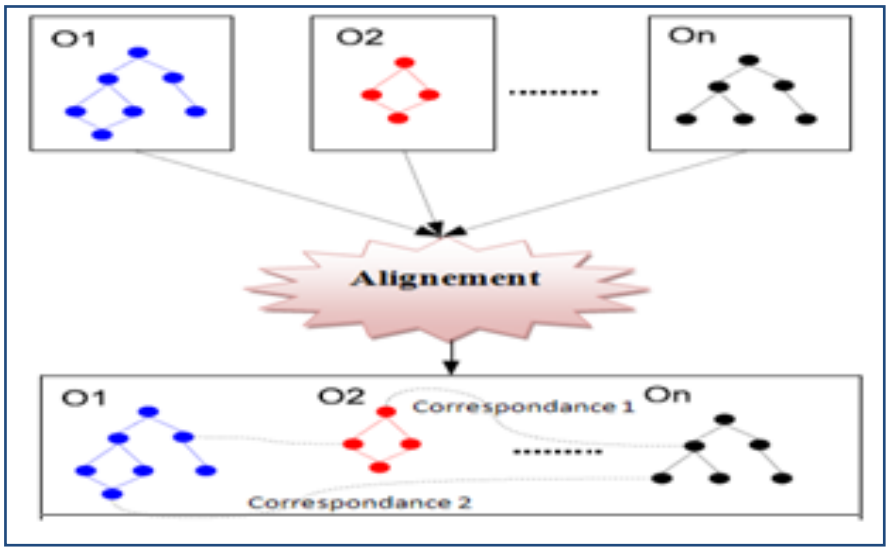


Figure 2.1*: Alignement d'ontologies*

2.4 Caractéristiques d'une méthode d'alignement

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

2.4.1 L'input

Est constitue essentiellement les structures destinées à être alignées et qui peuvent être des schémas XML, des schémas de base de données ou des ontologies.

Notre présent travail, se situe dans le cadre d'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	

Tableau 2.1: *"Schema matching" vs "Ontology matching"*

2.4.2 Le processus d'alignement

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

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

Ceci peut être représente schématiquement de la manière suivante :

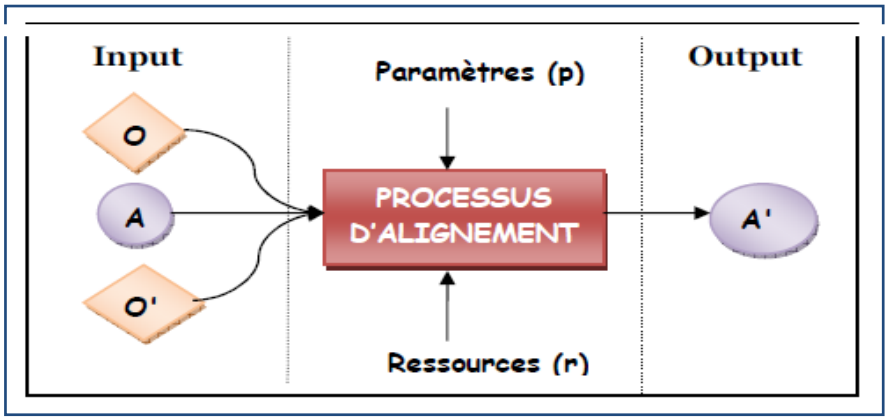


Figure2.2 *: Les trois dimensions de l'alignement*

Le processus d'alignement est exécuté selon *une stratégie* ou une combinaison de techniques d'alignement de base que on va les décrites dans les parties suivantes.

2.4.3 L'output

Le format et la structure du résultat de l'alignement sont précisés pour chaque méthode. Il faut préciser si l'alignement s'effectue entre les structures entières ou entre couples d'entités des deux ontologies. Le résultat pour la majorité des méthodes existantes est un fichier d'alignement (généralement en format XML), indiquant quelle sont les couples entités ontologiques qui correspondent. Toutes les méthodes d'alignement déterminent des correspondances entre les entités ontologiques en utilisant de mesures de similarité.

Il existe deux approches selon le résultat que la méthode d'alignement se produise :

A. Les approches basées sur la similarité

Vont quantifier la confiance que l'on doit accorder aux éléments de correspondance qu'elles retournent. Un alignement est décrit comme un ensemble de cinq éléments :

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

id : identifiant unique d'un mapping

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

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

r : la relation qui relie **e** à **e'**

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.

B. Les approches symboliques *[David, 2007]*

Elles vont permettre de déterminer si deux entités sont en correspondance ou non.

La figure suivante montre une représentation graphique d'un alignement basé sur la similarité.

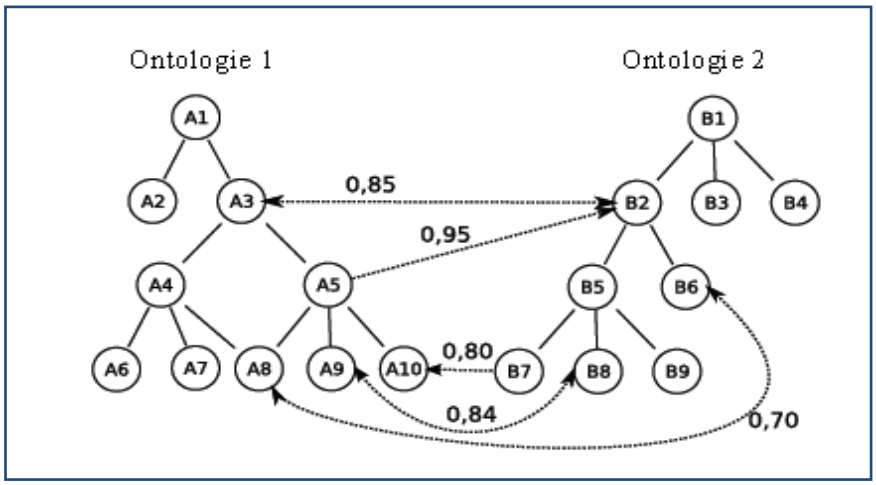


Figure 2.3 *: Représentation graphique d'un alignement basé sur la similarité*

- ✓ **Les types de relations**

  La grande majorité des méthodes s'intéressent seulement à la relation d'équivalence (⟺), notée également (=), cependant certaines méthodes sont plus expressives et distinguent les relations d'inclusion notée par (⊆).

- ✓ **La symétries ou asymétrie**

  Une procédure d'alignement est symétrique si l'ordre dans lequel les hiérarchies sont considérées n'a pas d'influence sur l'alignement produit.

- ✓ **La cardinalité des alignements**

  Une méthode d'alignement peut produire des alignements (voir la *Figure 2.4*):

  - Fonctionnels (de cardinalité 0,1 – 0,n) ;
  - Fonctionnels et injectifs (de cardinalité 0,1 – 0,1) ;
  - De cardinalité 0,n – 0,m.

La plupart des méthodes d'alignement produisent des alignements au moins fonctionnels. En effet, elles mettent en correspondance chaque élément d'une hiérarchie source à, au maximum, un élément de la hiérarchie cible.

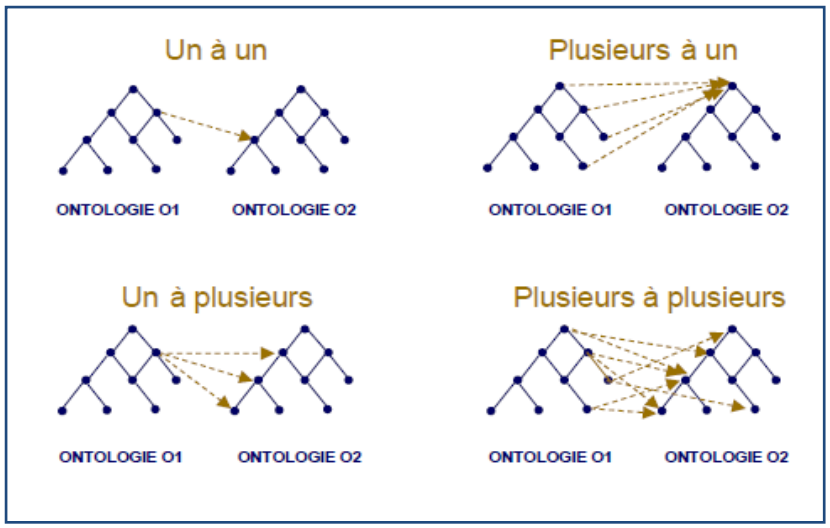


Figure 2.4 : *Les différents types de cardinalité d'un alignement*

2.5 Les techniques d'alignement

L'objectif de l'alignement est d'identifier les relations entre les entités de différentes ontologies. Ces relations sont découvertes à travers *les mesures de similarité* entre ces entités. Dans cette section, nous étudions les méthodes d'alignement. Pour cela, nous focalisons sur les techniques permettant la comparaison des entités et nous faisons quelques rappels sur les

définitions de distances et similarités qui sont des mesures largement utilisées par les méthodes d'alignement.

La similarité *est la quantité qui reflète la force du rapport entre deux objets ou deux caractéristiques*. La notion de similarité sémantique est vue comme celle de la similarité topologique en mathématiques, ou on l'associe a une fonction, appelée fonction de la similarité.

Ou on a, La similarité $S : O \times O \_ R$ est une fonction d'une paire d'entités a un nombre réel exprimant la similarité entre ces deux entités telle que:

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

La dissimilarité *est parfois utilisée au lieu de la similarité. Elle est définie de manière analogue à la similarité*, sauf qu'elle n'est pas transitive : La dissimilarité $DS : O \times O \_ R$ est une fonction d'une paire d'entités à un nombre réel exprimant la dissimilarité entre ces deux entités telle que:

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

La distance *est une mesure utilisee aussi souvent que les mesures de similarité. Elle mesure la dissimilarité de deux entités, elle est proportionnellement inverse a la similarité* : Si la valeur de la fonction de similarité de deux entités est élevée, la distance entre elle sera petite et vice-versa.

Elle est donc définie dans *[Euzenat & al, 2004]* comme suit : La distance $D : O \times O \_ R$ est une fonction de la dissimilarité satisfaisant la définitvité et l'inégalité triangulaire :

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

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

Elle est définie comme suit :

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

En outre, Les mesures de la similarité, de la dissimilarité et de la distance peuvent être classées selon la nature des entités que l'on veut comparer : *des termes, des structures, des instances ou des modèles théoriques.*

2.5.1 Les méthodes terminologiques

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

- ***Les techniques terminologiques syntaxiques*** : une chaîne de caractères peut être modélisée soit par une séquence de caractères, soit par une séquence de sous-chaîne. Le moyen le plus simple de comparer les deux chaînes de caractères est de mesurer la similarité entre ces dernières. Les méthodes syntaxiques s'exécutent après une certaine normalisation syntaxique des deux chaînes de caractères à comparer.

Dans la littérature, il existe plusieurs mesures pour calculer la similarité entre deux chaînes de caractères. Nous considérons une chaîne de caractère *s* comme un ensemble de caractères *S* :

- ***Similarité 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 $S_{Jaccard}$: S*S→[0, 1] telle que :

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

- ***Distance d'édition*** : Soit un ensemble d'opération d'édition *OP*, *op* ∈ *OP*, *op : S→S*, et une fonction de coût d'édition *w :OP → R* pour n'importe quelle paire des chaînes de caractères, il existe une séquence des opérations d'édition qui transforme la première en la seconde (et vice versa), la distance d'édition de deux chaînes de caractères *s* et *t* est une fonction de la dissimilarité DS_{de} *: S*S→*[0, 1] telle que $DS_{de}(s,t)$ est le coût de la séquence des opérations la moins coûteuse qui transforme *s* en *t*.

$$\overline{\boldsymbol{DS_{de}}}(\mathbf{s,t}) = \boldsymbol{min}_{(op1)I:opn(..op1(s))=t}\ \left(\sum_{i\in I} \boldsymbol{w_{opi}}\right)$$

- ***TF/IDF*** (Term Frequency/ Iverse Document Frequency): Soit *D* un corpus des documents, |D| dénote le nombre des documents dans le corpus *D*. soit t un terme à considérer, *n(t)* étant le nombre d'occurrences du terme *t* dans un document, et *N* étant le nombre des termes dans ce document, et *d(t)* étant le nombre des documents qui contiennent au moins une fois le terme *t*. les mesures de TF et TF/IDF sont définies comme suit :

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

La similarité entre deux chaînes sera alors définie comme le cosinus entre les vecteurs représentant les chaînes.

Un autre type de méthode dites ***hybrides*** proposent de combiner une mesure de type distance d'édition pour comparer mot à mot les deux chaînes de caractères, puis une mesure vectorielle pour agréger les similarités.

- ***La similarité hybride*** : Soit $s=a_1..a_k$ et $t=b_1..b_l$ deux chaînes de caractères, où a_i et b_i sont des sous-chaînes de *s* et *t* respectivement. Soit *S* une mesure de la similarité entre deux chaînes de caractères.

La similarité hybride est une fonction de la similarité $S_{Hybride}$: S*S→[0, 1] telle que :

$$\boldsymbol{\overline{S_{Hybride}}\,(s,t) = \frac{1}{K}\sum_{i=1,j=1}^{K,l} \max S(a_i, b_i)}$$

Les méthodes syntaxique prennent en compte seulement la ressemblance physique de chaînes de caractères. Cependant, il existe dans toutes les langues des synonymes pour désigner une même entité, et l'utilisation de similarité syntaxique sur chaînes de caractères ne permet pas de repérer de proximité dans ce cas. Afin de dépasser ces limites, d'autres méthodes basées sur des connaissances linguistiques existent.

2.5.2 Les méthodes linguistiques

La similarité entre deux entités représentées par des termes peut aussi être déduite en analysant ces termes à l'aide des méthodes linguistiques. Ces méthodes exploitent essentiellement des propriétés expressives et productives de la langue naturelle *[Maynard & Ananiadou, 1999]*. Les informations exploitées peuvent être celles intrinsèques (des propriétés linguistiques internes des termes telles que des propriétés morphologiques ou syntaxiques) ou celles extrinsèques (employant des ressources externes telles que des vocabulaires ou des dictionnaires) tel que l'API de WordNet *[Miller, 1995]*.

A. Les méthodes intrinsèques

Une même entité ou un même concept peut être référencé par plusieurs termes (synonymie) ou par plusieurs variantes d'un même terme. Les méthodes intrinsèques fonctionnent avec le

principe de chercher la forme canonique ou représentative d'un mot ou d'un terme (lemme) à partir de ses variantes linguistiques (lexème).

B. Les méthodes extrinsèques

Ces méthodes calculent la valeur de similarité entre deux termes en employant des ressources externes telles que des dictionnaires, des lexiques ou des vocabulaires. La similarité est déterminée grâce aux liens sémantiques déjà existants dans ces ressources externes tels que des liens synonymes (pour l'équivalence), des liens hyponymes/hyperonymes (pour la subsomption).

2.5.3 Les méthodes structurelles

Une méthode d'alignement structurelle compare deux entités en s'appuyant sur les relations sémantiques ou syntaxiques entre les entités en question. *[Shvaiko et Euzenat, 2005]* distingue deux types de méthodes structurelles :

A. **Les méthodes structurelles internes** : ces méthodes exploitent des informations des structures internes des entités, telles que les informations concernant les attributs de l'entité, co-domaine des attributs, cardinalité des attributs, caractéristique d'attributs (transitivité, symétrie,..),.. .etc.
B. **Les méthodes structurelles externes** : ces méthodes exploitent l'hiérarchie des entités et les relations entre les entités elles-mêmes (relation de subsomption, méréologie). L'idée ici, est que si deux entités sont similaires, leurs voisins pourraient également être d'une façon ou d'une autre similaires.

2.5.4 Les méthodes extensionnelles

La description extensionnelle d'une entité représente l'ensemble de ses instances. De ce fait, une mesure extensionnelle de deux entités produite une similarité qui est en fait la similarité entre les deux ensembles de leurs instances en se basant sur la comparaison exacte des éléments dans les deux ensembles.

2.5.5 Les méthodes sémantiques *[Euzenat & shvaiko, 2007]*

A. Les techniques basées sur les ontologies externes

Lorsque deux ontologies doivent être alignées, il est préférable que les comparaisons se fassent selon un capital de connaissances commun. Ce type de techniques s'intéresse à l'utilisation d'ontologie formelle intermédiaire pour répondre à ce besoin.

L'idée est que cette ontologie, avec une couverture appréciable du domaine d'intérêt des ontologies (ou une ontologie encore plus générale comme une ontologie de haut niveau), va permettre de lever le voile sur les ambigüités concernant les différentes significations possibles des termes.

B. Les techniques déductives

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

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

Résumé : Les techniques d'alignement présentées sont les outils dont dispose les concepteurs de systèmes d'alignement pour trouver les solutions aux problèmes de l'hétérogénéité des ontologies. La conception et le développement de ces systèmes nécessite un traitement plus global, en effet, et selon *[Euzenat et Shvaiko, 2007]*, ces systèmes doivent pouvoir :

- *Sélectionner et combiner* les techniques de base de l'alignement afin d'améliorer la qualité de l'alignement résultant.
- *Apprendre* à partir des données les meilleurs techniques et les meilleurs paramètres
- Tenir compte des remarques des experts du domaine de l'ontologie et des utilisateurs d'une manière générale.

2.6 Quelques Systèmes D'alignement

La problématique de la recherche d'alignement entre deux structures a été posée dans de nombreux domaines. Cette dernière décennie, de nombreuses méthodes d'alignement dédiées aux schémas (bases de données, objets, xml) et aux ontologies ont vu le jour. Ces méthodes ont été étudiées dans plusieurs états de l'art *(*[Shvaiko et Euzenat, 2005], [Kalfoglou et Schorlemmer 2003], [Rahm et Berstein, 2001]) qui les ont classées en fonction des techniques, des types d'information, des critères qu'elles utilisent, et de la façon dont elles les exploitent.

2.6.1 Le système PROMPT *[Noy et Musen, 2001]*

Est un outil semi-automatique de fusion et d'alignement d'ontologies. Il commence par les matchers linguistiques pour la comparaison initiale mais produit par la suite une liste de

suggestions à l'utilisateur basée sur la connaissance linguistique et structurale, puis dirige l'utilisateur pour choisir les meilleurs mappings. Prompt est un plugin de PROTEGE2000.

2.6.2 COMA *[Doan et al, 2002]*, *[Doan et al, 2003]*

COMA (COmbinaison of schema MAtching approche) est un système permettant de mettre en correspondance des schémas (des bases de données, de XML) automatiquement ou bien manuellement. Le système fournit une bibliothèque des matchers qui se compose de (i) 6 matchers simples qui emploient des techniques linguistiques; (ii) 5 matchers hybrides qui combinent les matchers simples précédents en exploitant quelques informations structurelles. Ainsi, le système présente quelques stratégies pour agréger les résultats de similarité pour chaque paire d'éléments calculés par différents matchers

2.6.3 GLUE *[Doan et al, 2002]*

GLUE est une approche qui utilise la technique d'apprentissage pour trouver des correspondances entre deux ontologies. Il comprend plusieurs modules d'apprentissage (learners), qui sont entraines par des instances des ontologies. Ces modules emploient la technique d'apprentissage Bayes naïf en exploitant différentes caractéristiques des instances. Les prévisions de ces modules de mise en correspondance sont combinées par un méta module de mise en correspondance en employant la somme pondérée. Le résultat final des correspondances sera déduit à partir des valeurs de similarité agrégées en employant la technique d'optimisation de contraintes.

2.6.4 OLA (OWL Lite Alignment) *[Euzenat et al ,2004]*

Est un système implémentant un algorithme d'alignement des ontologies décrites en OWL. OLA mesure la similarité entre deux entités à partir des calculs de similarité entre leurs caractéristiques (leurs types : classe, relation ou instance, leurs liens avec d'autres entités : sous-classes, domaine, …). La valeur de similarité finale est la somme pondérée des valeurs de similarité de chaque caractéristique. Les poids sont associés suivant le type d'entité à comparer et ses caractéristiques. L'algorithme applique un calcul du point fixe, avec des itérations pour améliorer la similarité de deux entités. Quand il n'y a plus d'améliorations possibles, des alignements entre deux ontologies sont proposés.

2.7 Étude Comparative des méthodes d'alignement

Quelques systèmes d'alignement qui sont présentes dans la partie précédant sont résumés dans ce tableau :

Systèmes d'Alignements				
Critère de comparison	**PROMPT**	**COMA**	**GLUE**	**OLA**
Entrée	Ontologies	Schémas : bases des données relationnelles, xml	Ontologies + instances	Ontologies en OWL
Similarité	égalité des chaînes de caractères pour les étiquettes des concepts et des relations	-Similarité des préfixes, des suffixe, des synonymes, des types de donnée, distance d'édition -Similarité agrégée des enfants, des feuilles ou des chemins entre des éléments	Techniques terminologiques : tokenisation, égalité des chaînes de caractères pour des tokens	-égalité des chaînes de caractères pour les URIs des entités Similarité des chaînes des caractères pour des étiquettes (non précise) -Similarité des types des données
Sortie	RDF(S), graphe RDF	Graphes acycliques orientés	Non précisée	OL-graphes
Optimisation	(-)	(-)	(-)	(-)

Tableau 2.2 : *comparaison entre quelques systèmes d'alignement*

Conclusion

Après avoir présenté un cadre générale sur l'alignement d'ontologies, on conclu que ce concept est une pratique qui devient nécessaire dés que différentes ontologies coexistent et couvrent partiellement les mêmes domaines. L'ensemble des ontologies et des alignements qui les interconnectent forme aujourd'hui un vaste réseau de connaissances distribuées. Ce qu'il nous renvoi a cherché une méthode d'optimisation pour cette notion dans les chapitres suivants de notre projet.


Chapitre
III
Appariement des Graphes

Introduction

Evaluer la similarité de deux objets est un problème qui se pose dans de nombreuses applications informatiques, un corollaire de ce problème étant de retrouver, parmi un ensemble d'objets, l'objet le plus similaire à un autre. Citons par exemple la recherche d'informations, où il s'agit de trouver des documents similaires à un document donné, la reconnaissance d'images, où l'on confronte une image à des modèles, le raisonnement à partir de cas, où il s'agit de trouver un problème déjà résolu ressemblant à un nouveau problème à résoudre dans le but d'adapter la solution du problème déjà résolu au nouveau problème à résoudre, ou encore dans les application d'alignement des ontologies, où l'ontologie est considérée comme un ensemble d'entités (objets, classes, propriétés, etc) et dont le but est de trouver les rapport entre ses entités.

Lorsque les objets sont décrits par des graphes, il s'agit d'évaluer la similarité de graphes, problème qui se ramène à chercher un « meilleur » appariement des sommets des graphes, i.e., une mise en correspondance des sommets des graphes permettant de retrouver un maximum de caractéristiques communes aux deux graphes.

Sur ce fait, ce chapitre a porté sur les problèmes d'appariement des graphes. Il commence par quelques rappels de mathématiques et de théories des graphes utilisés dans ce mémoire. La description formelle d'appariement des graphes, toute on expose une partie sur les mesures de similarité, et on finalise par la présentation de quelques algorithmes connu dans cet intitulai « *Appariement des graphes* ».

3.1 Introduction aux Graphes

Un graphe permet de représenter la structure, les connexions d'un ensemble complexe en exprimant les relations entre ses éléments : réseau de communication, réseaux routiers, interaction de diverses espèces animales, circuits électriques,. . .

Il constitue une méthode de pensée qui permet de modéliser une grande variété de problèmes en se ramenant à l'étude de sommets et d'arcs. Les derniers travaux en théorie des graphes sont souvent effectués par des informaticiens, du fait de l'importance qu'y revêt l'aspect algorithmique. *[Champin, 2003]*

Définition 1 : ***Un Graphe*** est une structure de données utilisées notamment pour modéliser des objets en termes de composants appelés *nœuds*, et de relations binaires entre composants appelées *arcs*. De façon plus formelle, un graphe est défini par un couple $G = (V, E)$ tel que :

- V est un ensemble fini de nœuds,
- $E \subseteq V \times V$ est un ensemble d'arcs.

Définition 2 : ***Graphes orientés*** est défini par un couple $G = (V, E)$ où V est un ensemble fini de sommets et $E \subseteq VxV$ est un ensemble fini d'arêtes orientées. Chaque arête (u, v) est un couple ordonné de sommets représentant un arc dirigé du sommet u au sommet v. Dans un graphe orienté, il y a deux arcs possibles (u, v) et (v, u) entre deux sommets u et v. Le fait que si (u, v) est une arête, alors (v, u) n'est pas obligatoirement un arc du graphe.

Définition 3 : ***Un Graphe Étiqueté*** est un graphe orienté auquel on a associé un ensemble non vide d'*étiquettes* à chacun des nœuds et des arcs. Plus formellement, étant donné L_V un ensemble fini d'étiquettes de nœuds et L_E un ensemble fini d'étiquettes d'arcs, un graphe étiqueté est défini par un triplet $G = (V, r_V, r_E)$ tel que :

- V est un ensemble fini de nœuds,
- $r_v \subseteq V \times L_v$ est la relation reliant nœud et étiquettes, i.e., r_V est l'ensemble des couples (v_i, l) tels que le nœud v_i est étiqueté par l,
- $r_E \subseteq V \times V \times L_E$ est la relation reliant arcs et étiquettes, i.e., r_E est l'ensemble des triplets (v_i, v_j, l) tels que l'arc (v_i, v_j) est étiqueté par l. l'ensemble E des arcs peut être défini par $E = \{ (v_i, v_j) \mid \exists l, (v_i, v_j, l) \in r_E \}$.

3.2 Appariement

Les problèmes d'appariement de graphes consistent à mettre en correspondance les sommets de deux graphes, l'objectif étant généralement de comparer les objets modélisés par les graphes. *[Champin, 2003]*

3.2.1 Appariements de Graphes Étiquetés

Formellement, un appariement entre deux graphes étiquetés *G1* et *G2* est une relation m contenant tous les couples (u, v) tel que u est apparié au sommet v.

À partir d'un appariement, en ajoutant ou en enlevant un ou plusieurs couples de sommets, nous obtenons un nouvel appariement. Autrement dit, un appariement est aussi un ensemble mathématique. Nous pouvons donc appliquer des opérateurs ensemblistes sur des appariements.

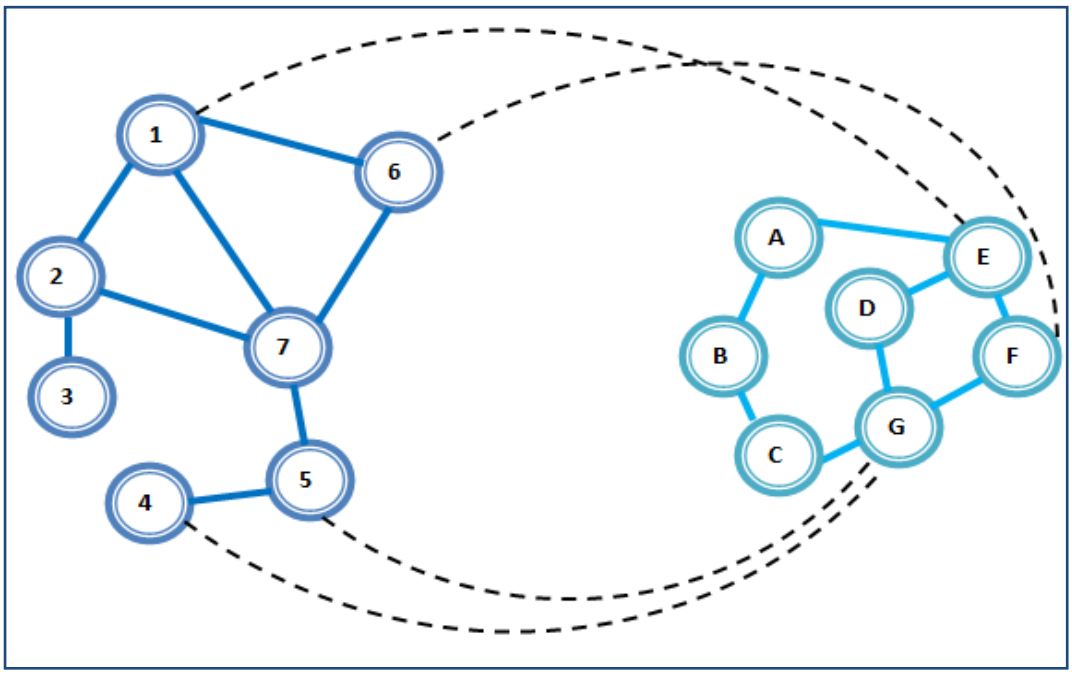


Figure3.1 : *Un appariement entre deux graphes.*

Avec les deux graphes représentés dans la *Figure 3.1*, il y a plusieurs appariements possibles. Nous définissons ci-dessous un matching de graphes:

$$m = \{(1, E), (6, F), (4, G), (5, G)\}$$

Cet appariement accouple respectivement les sommets 1et 6 avec les sommets E, F et les sommets 4 et 5 sont tous les deux apparié au sommet G. dans cet appariement l'ensemble associé au sommet *G* et m(*G*)= {4,5}.

Si nous enlevons le couple (*4, G*) de l'appariement m, nous obtenant un nouvel appariement m`= {(*1*,*E*), (*6, F*),(*5, G*)}.

Dans ce nouvel appariement, le sommet *4* n'est donc accouplé a aucun sommet de deuxième graphe. L'ensemble de sommets associé a *4* est vide : m(4) = ∅ .

3.2.2 Classification d'Appariements

Il existe 4 types d'appariement :

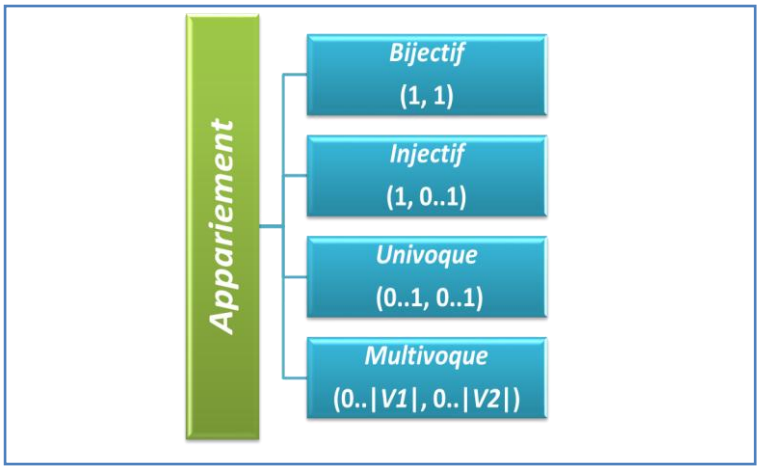


Figure 3.2 : *Types d'appariement des graphes*

A. Appariement bijectif

Tous les sommets d'un graphe sont appariés à un et un seul sommet de l'autre graphe et inversement. Il n'existe pas donc deux sommets d'un graphe liés à un même sommet de l'autre graphe.

Le problème associé à ce type d'appariement est l'isomorphisme de graphes visant à déterminer si les deux graphes donnés sont isomorphes ou non en indiquant un appariement bijectif 1-1 entre eux.

B. Appariement injectif

Un appariement injectif de *G1* à *G2* lie chaque sommet du graphe *G1* à au maximum un sommet du graphe *G2*. Il est impossible d'apparier deux sommets de *G1* à un même sommet de *G2*.

Ce type d'appariement associe au problème d'isomorphisme de sous graphe. L'objectif de ce problème est de déterminer si un graphe est l'isomorphisme d'un sous-graphe de l'autre graphe ou non en indiquant un appariement injectif entre eux.

C. Appariement univoque

Dans un appariement univoque entre deux graphes G1 et G2, chaque sommet d'un graphe est apparié au maximum à un sommet de l'autre graphe. Chaque sommet n'a donc que deux possibilités : soit il n'est pas apparié, soit il est apparié à un sommet unique.

Cet appariement concerne le problème de plus grand sous-graphe commun. Il s'agit de trouver le sous graphe le plus grand dans chacun des graphes de telle sorte que ces deux sous-graphes soient isomorphes.

D. Appariement multivoque

Chaque sommet d'un graphe est apparié à un ensemble (éventuellement vide) de sommets de l'autre graphe. Dans un appariement multivoque, un sommet est peut-être apparié en même temps à plusieurs sommets. *[Champin, 2003]*

3.2.3 Les Problèmes d'Appariement de Graphes

Les problèmes d'appariement les plus connus sont :

Définition 3.1* : *Isomorphisme de graphes permet de vérifier que deux graphes sont structurellement identiques.

Plus formellement, étant donné deux graphes G = (V, E) et G'= (V', E'), G est isomorphisme à G' s'il existe une bijection Ø : V→ V' qui préserve les arcs, i.e., $\forall (v_1, v_2) \in V^2, (v_1, v_2) \in E \Leftrightarrow (\boldsymbol{\phi}(v_1), \boldsymbol{\phi}(v_2)) \in E'$.

Définition 3.2* : *Isomorphisme de sous-graphes *(partiel)* permet de vérifier qu'un graphe est inclus dans un autre graphe.

Plus formellement, étant donné deux graphes *G = (V, E)* et *G'= (V', E')*, *G* est un sous-graphe isomorphe d'un graphe *G'* s'il existe une injection *Ø : V→ V'* qui préserve les arcs de *G*, i.e., $\forall (v_1, v_2) \in V^2, (v_1, v_2) \in E \Leftrightarrow (\boldsymbol{\phi}(v_1), \boldsymbol{\phi}(v_2)) \in E'$.

L'isomorphisme de sous-graphes impose non seulement de retrouver tous les arcs de G dans G', mais aussi que tous couple de sommets de G non reliés par arc soit apparié à un couple de sommets de *G'* également non reliés par un arc. L'isomorphisme de sous graphe partiel relâche cette deuxième condition, i.e., il s'agit de trouver une injection *Ø : V→ V'* telle que $\forall (v_1, v_2) \in V^2, (v_1, v_2) \in E \Longrightarrow (\boldsymbol{\phi}(v_1), \boldsymbol{\phi}(v_2)) \in E'$.

Définition 3.3* : *Le plus grand sous-graphe *(partiel) commun* permet d'identifier la plus grande partie commune à deux graphes. Plus formellement, le plus grand sous-graphe commun de deux graphes *G = (V, E)* et *G' = (V', E')* est le plus grand graphe qui soit un sous-graphe isomorphe de *G* et *G'*, la taille du sous-graphe commun pouvant être définie par le nombre de nœuds et/ ou le nombre d'arcs. Cette définition est étendue à la notion de plus grand sous-graphe partiel commun en recherchant le plus grand sous-graphe partiel isomorphe à *G* et *G'*.

Définition 3.4 **: *La distance d'édition de graphes*** permet d'identifier le plus petit nombre d'opération effectuée pour transformer un graphe en un autre. La distance d'édition entre deux graphes G et G' est le coût minimal pour transformer G en G'. Dans le cas général, on considère des graphes étiquetés, pour cette transformation, on dispose de six opérations élémentaires : l'insertion, la suppression et le ré-étiquetage de nœuds et d'arcs. Chaque opération a un coût. L'ensemble d'opérations le moins coûteux permettant de transformer G en un graphe isomorphe à G' définit la distance d'édition entre G et G'.

Définition 3.5 **: *Un appariement multivalent*** est un appariement où chaque sommet du premier graphe est apparié à zéro, un ou plusieurs sommets de l'autre graphe. Plus formellement, un appariement multivoque de deux graphes G et G' est une relation $m \subseteq V \times V'$ contenant tous les couples $(v, v') \in V \times V'$ tels que le nœud v est apparié au nœud v'.

Définition 3.6 **: *Les problèmes de sélection de sous-ensembles*** *(SS-problème)* ont pour but de trouver un sous-ensemble consistant et optimal d'objets. Plus formellement, un SS-problème est défini par un triplet $(S, S_{consistant}, f)$ tel que : *[Solnon, 2005]*

- S est l'ensemble d'objet ;
- $S_{consistant} \subseteq P(S)$ est l'ensemble de tous les sous-ensembles de S qui sont consistants, afin de pouvoir construire chaque sous-ensemble de $S_{consistant}$ de façon incrémentale (en partant de l'ensemble vide et en ajoutant à chaque itération un objet choisi parmi l'ensemble des objets consistants avec l'ensemble en cours de construction), on impose la contrainte suivante : pour chaque sous-ensemble consistant non vide $S' \in S_{consistant}$, il doit exister au moins un objet $o_i \in S'$ tel que $S' - \{o_i\}$ est aussi consistant.
- $f : S_{consistant} \rightarrow$ IR est la fonction objectif qui associe un coût $f(S')$ à chaque sous-ensemble d'objets consistant S' $\in S_{consistant}$.

Le but d'un SS-problème $(S, S_{consistant}, f)$ est de trouver $S^* \in S_{consistant}$ tel que $f(S^*)$ soit maximal.

3.3 Mesure De Similarité

Dans cette section, nous présentons la mesure de similarité proposée par *Pierre-Antoine Champin* et *Christine Solnon* *[Champin, 2003]*. Ils proposent une fonction de similarité pour un appariement entre deux graphes. Cette fonction est directement proportionnelle aux nombres de caractéristiques communes aux deux graphes par rapport au nombre total de caractéristiques. Nous exposerons au début dans cette section les caractéristiques d'un graphe et comment déterminer les caractéristiques communes partagées par deux graphes. Finalement, nous donnons la mesure de similarité de graphes.

3.3.1 Caractéristiques communes

Un graphe étiqueté est caractérisé par ses composants et les étiquettes attachées aux composants. Chaque couple (sommet, étiquette) ou (arête, étiquette) est considéré comme une caractéristique du graphe. Le graphe étiqueté $G = <V, rV, rE>$ défini dans donc par un ensemble de caractéristiques *descr*(*G*) associés aux sommets et aux arcs :

$$descr(G) = rV \cup rE = \{(v, l) \in rV\} \cup \{(vi, vj, l) \in rE\}$$

Où *: rV* est l'ensemble de caractéristiques des sommets et *rE* l'ensemble de caractéristiques des arcs du graphe étiqueté.

Deux caractéristiques de sommets (ou d'arcs) appartenant à deux graphes sont similaires s'ils sont associés à une même étiquette. Étant donnés deux graphes *G1*, *G2* et un appariement *m* entre eux, les caractéristiques communes aux deux graphes sont les caractéristiques des sommets ou des arcs appariés par *m* à au moins un sommet ou un arc de l'autre graphe ayant la même étiquette.

Formellement, l'ensemble $descr(G) \sqcap_m descr(G')$ des caractéristiques communes aux deux graphes étiquetés G et G‟ par rapport à l'appariement *m* est défini par :

$$\begin{aligned} descr(G) \sqcap_m descr(G') = & \{(v, l) \in r_V \mid \exists\, (v, v') \in m, (v', l) \in r_{V'}\} \\ & \cup \{(v', l) \in r_{V'} \mid \exists\, (v, v') \in m, (v, l) \in r_V\} \\ & \cup \{(v_i, v_j, l) \in r_E \mid \exists\, (v_i, v_i') \in m, \exists\, (v_j, v_j') \in m, (v_i', v_j', l) \in r_{E'}\} \\ & \cup \{(v'_i, v'_j, l) \in r_{E'} \mid \exists\, (v_i, v'_i) \in m, \exists\, (v_j, v_j') \in m, (v_i, v_j, l) \in r_E\} \end{aligned}$$

La similarité entre deux graphes est une fonction directement proportionnelle au nombre des caractéristiques communes aux deux graphes

$$sim(G, G')m \sim | descr(G) \sqcap m\ descr(G')| \text{ où :}$$

sim(*G, G'*)*m* : la similarité entre deux graphes G, G' en respectant l'appariement *m* .
|*descr*(*G*) ⊓*m descr*(*G'*)| : la taille de l'ensemble des caractéristiques communes aux deux graphes

En fait, nous pouvons apparier deux sommets quelconques ayant deux étiquettes différentes mais cette mise en correspondance ne produit pas des caractéristiques communes aux quelles la similarité de graphes est directement proportionnelle. Une bonne mise en correspondance est celle entre les sommets ayant les mêmes étiquettes.

3.3.2 Éclatement

L'ensemble de sommets éclatés par un appariement. C'est l'ensemble de sommets appariés à plus d'un sommet.
Les deux graphes dans la *Figure 3.1* sont similaires. Ils deviennent identique si les deux sommets *e* et *f* du premier graphe sont enlevés et remplacés par un seul sommet. L'éclatement du sommet *5* de *G2* en deux sommets *e* et *f* dans *G1* diminue la similarité de deux graphes. Donc, pour évaluer mieux la similarité, nous avons besoin d'appliquer une pénalité sur l'éclatement des sommets.
Donc, la similarité entre deux graphes sera une fonction inversement proportionnelle au nombre de sommets éclaté

$$sim(G, G')_m \sim | descr(G) \sqcap_m descr(G')|$$
$$\sim - |splits(m)|$$

Où : |*splits*(*m*)| est la taille de l'ensemble des sommets éclatés.

3.3.3 Similarité de graphes par un appariement

En se basant sur la mesure de similarité entre deux ensembles de *Tversky, Pierre-Antoine Champin et Christine [Champin, 2003]* ont proposé une fonction de similarité entre deux graphes étiquetés *G* et *G'* par rapport à un appariement *m* suivante :

$$sim(G, G')_m = \frac{f\left(descr(G) \sqcap_m descr(G')\right) - g(splits(m))}{f(descr(G) \cup descr(G'))}$$

Où :

descr(*G*) ∪ *descr*(*G'*) : l'union de toutes les caractéristiques des deux graphes *G* et *G'*

f et *g* : deux fonctions positives, monotones et dépendantes de l'application considérée.

3.3.4 Similarité de graphes

La similarité sim(*G, G'*) de deux graphes est définie comme la plus grande similarité possible obtenue par le meilleur appariement

$$sim(G, G') = \max_{m \in V \times V'} \frac{f(descr(G) \sqcap_m descr(G')) - g(splits(m))}{f(descr(G) \cup descr(G'))}$$

Pour évaluer exactement la similarité de deux graphes, nous devons rechercher le meilleur appariement. Le matching de graphes est donc un problème consistant à mesurer la similarité car la recherche du meilleur appariement se base sur l'optimisation de la fonction de similarité par rapport à un appariement.

3.4 Les Algorithmes d'Appariement de Graphes

On considère maintenant le problème de calcule de similarité maximum de deux graphes multi-étiquetés, ce problème très général d'appariement de graphes apparait difficile à résoudre par une approche complète. On va citer dans cette partie trois algorithmes : un algorithme glouton, un algorithme tabou, et l'algorithme d'optimisation par colonie de fourmis. *[Solnon, 2005]*

3.4.1 L'algorithme Glouton

Voila ci-dessous, la *Figure 3.4* présente l'algorithme Glouton

✓ **Fonction**

fonction Glouton(G_1=$\langle V_1, r_{V_1}, r_{E_1}\rangle$, G_2=$\langle V_2, r_{V_2}, r_{E_2}\rangle$)
retourne un appariement $m \subseteq\mid V_1 \times V_2$
  $m \leftarrow \emptyset$
  itérer
    $cand \leftarrow \{(u_1,u_2) \in V_1 \times V_2 - m \mid score(m \cup \{(u_1,u_2)\})$ est maximal$\}$
  sortir si $\forall (u_1,u_2) \in cand', score(m \cup \{(u_1,u_2)\}) \leq score(m)$
    $cand' \leftarrow \{(u_1,u_2) \in cand \mid f(look\_ahead(u_1,u_2))$ est maximal$\}$
      où $look\_ahead(u_1,u_2)$=$\{(u_1,v_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (u_2,v_2,l) \in r_{E_2}\}$
        $\cup \{(u_2,v_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (u_1,v_1,l) \in r_{E_1}\}$
        $\cup \{(v_1,u_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (v_2,u_2,l) \in r_{E_2}\}$
        $\cup \{(v_2,u_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (v_1,u_1,l) \in r_{E_1}\}$
        $-descr(G_1) \sqcap_{m \cup \{(u_1,u_2)\}} descr(G_2)$
    choisir aléatoirement un couple (u_1,u_2) dans $cand'$
    $m \leftarrow m \cup \{(u_1,u_2)\}$
  fin itérer
  retourner m

Figure 3.3: *Recherche gloutonne d'un appariement*

✓ **Processus**

L'algorithme glouton démarre d'une solution vide. Il ajoute itérativement à l'appariement *m* un couple de sommet tel que ce couple augmente le plus la similarité de graphes. À chaque étape, les candidats à considérer sont des couples de sommets qui ne sont pas encore appariés (qui n'appartiennent pas à *m*) et qui améliorent l'appariement actuel *m*. Le couple à ajouter est choisi selon une heuristique gloutonne : l'ajout du couple doit maximiser la similarité. S'il y a plusieurs candidats maximisant la similarité, nous choisirons aléatoirement un parmi eux. L'algorithme s'arrête quand aucun couple de sommets ne peut augmenter la similarité en l'ajoutant à *m*. À ce moment-là, l'ensemble de candidats est vide.

3.4.2 L'algorithme Tabou

La recherche *Tabou* *[Glover, 1989]*, *[Dorne et Hao ,1998]* est une des meilleures heuristiques connues pour choisir le prochain voisin à explorer. La *Figure 3.5* montre l'algorithme du Tabou.

✓ **Fonction**

fonction Tabou($G = \langle V, r_V, r_E \rangle, G' = \langle V', r_{V'}, r_{E'} \rangle, k, limiteQualite, maxMouv$)
retourne un appariement $m \subseteq V \times V'$
  $m \leftarrow Glouton(G, G')$; $best_m \leftarrow m$; $nbMouv \leftarrow 0$
  tant que $sim_{best_m}(G, G') < limiteQualite$ **et** $nbMouv < maxMouv$ **faire**
    $cand \leftarrow \{m' \in voisinage(m) / sim_{m'}(G, G') > sim_{best_m}(G, G')\}$
    si $cand = \emptyset$ **alors**/* *pas d'aspiration* */
      $cand \leftarrow \{m' \in voisinage(m)$/le mouvement inverse (de m' à m) n'a pas été réalisé durant les k dernières itérations$\}$
    fin si
    $cand \leftarrow \{m' \in cand / m'$ est maximal par rapport au critère de Glouton$\}$
    choisir aléatoirement $m' \in cand$
    $m \leftarrow m'$; $nbMouv \leftarrow nbMouv + 1$
    si $sim_m(G, G') > sim_{best_m}(G, G')$ **alors** $best_m \leftarrow m$ **fin si**
  fin tant que
  retourner $best_m$

Figure 3.4 : *Algorithme Tabou*

✓ **Processus**

A chaque itération, le voisin est choisi par rapport au critère de l'algorithme glouton. Notons cependant que le meilleur voisin d'un appariement m localement optimal est de moins bonne qualité que m. Afin de ne pas cantonner la recherche autour d'un maximum local en ajoutant puis en retirant continuellement le même couple de sommets, une liste taboue est utilisée. Cette liste de longueur k mémorise les k derniers mouvements effectués afin d'interdire un mouvement inverse à un mouvement récemment. Une exception nommée "aspiration" est ajoutée : si un mouvement interdit permet d'atteindre un appariement de meilleure qualité que le meilleur appariement connu jusqu'alors, le mouvement est quand même réalisé.

3.4.3 L'algorithme d'optimisation par colonie de fourmis

L'idée de s'inspirer des colonies de fourmis pour résoudre un problème d'optimisation combinatoire est due à A. Colorni, M. Dorigo et V. Maniezzo *[Colorni, 1992]*. L'optimisation par colonies de fourmis, s'inspire du comportement des fourmis lorsque celles-ci sont à la recherche de nourriture. Les fourmis se dirigent de manière probabiliste en tenant compte de la quantité de phéromone qui est autour d'elles et qui a été précédemment déposée par les autres membres de la colonie.

Les algorithmes de colonies de fourmis s'inspirent de la capacité collective qu'ont les fourmis de résoudre certains problèmes, qu'elles ne pourraient pas résoudre individuellement étant donné leurs capacités limitées. Grâce à l'utilisation qu'elles font de leurs phéromones, les fourmis peuvent collectivement trouver le plus court chemin entre deux points. *[Bichot, 2007]*

- ✓ **Algorithme**

Procédure COLONIES DE FOURMIS
répète % Boucle externe
construction d'une solution par déplacement des fourmis
évaporation de la phéromone
affinage de la solution % étape optionnelle
jusqu'`a ce que critère d'arrêt
retourner Pk
fin Procédure

- ✓ **Processus**

L'algorithme présente de manière générique la méthode d'optimisation par colonies de fourmis. Les auteurs précisent que la planification et la synchronisation des trois étapes de la boucle externe sont à la discrétion de la personne qui implémentera leur méthode.

Conclusion

Ce chapitre a montré une introduction sur les graphes et le problème d'appariement de graphes, la mesure de similarité basée sur des appariements multivoques des sommets des graphes ce qui permet de prendre en compte les problèmes de granularité, par la suite les bases de la recherche locale permettant de résoudre efficacement plusieurs problèmes d'optimisation.

Nous avons aussi décrit un algorithme d'optimisation méta-heuristiques et son mécanisme de la recherche « l'algorithme d'optimisation par colonie de fourmis » que on va les bien détaille à travers du reste de notre projet, pour résoudre le problème d'optimisation « L'appariement multivoque ».


Chapitre
IV
Conception de L'application

Introduction

Ce chapitre présente notre contribution au problème posé dans ce mémoire, à savoir l'utilisation des méta-heuristiques dans l'alignement des ontologies. Contrairement aux méthodes présenter dans le deuxième chapitre, notre travail à l'originalité d'appliquer les colonies de fourmis pour optimiser le résultat d'alignement d'ontologies.

En vue de réaliser notre contribution, nous proposons de prendre en entrée deux ontologies décrites en OWL et de les transformer en des structures de graphes, en particulier, en deux graphes étiquetés appelés OWLGraph.

Les graphes sont considérés comme d'excellents outils permettant la représentation de données structurées. L'intérêt de les utiliser dans notre processus d'alignement se trouve certainement dans leur puissance à modéliser les dépendances entre les éléments d'une ontologie, et que les colonies de fourmis sont adéquat et applicable sur la structure graphique.

Ainsi, l'idée est d'appliquer un modèle de calcul de similarité entre les deux ontologies qui se réduit à la comparaison des deux graphes. D'un point de vue mathématique, le calcul de similarité entre deux graphes est réalisé par une recherche de morphisme de graphes. Ce genre de problème s'appelle *appariement de graphes* et est un problème NP-complet.

Dans le processus d'alignement, nous nous intéressons à la recherche du *meilleur appariement* parmi les *appariements multivoques* des nœuds des graphes, où chaque nœud du graphe de la première ontologie peut être apparié à un ou à plusieurs nœuds du graphe de la deuxième ontologie ou inversement. Cette recherche peut se traduire en un problème de *sélection de sous-ensemble (SS-problème)*, dans le but de trouver un sous-ensemble qui *satisfait certaines propriétés*.

4.1 Présentation du problème

Étant données deux ontologies, l'alignement produit un ensemble de correspondances chacune liant deux entités (par exemple, des concepts, des instances, des propriétés, des termes, etc.) par une relation (équivalence, subsomption, incompatibilité, etc.), éventuellement munie d'un degré de confiance. L'ensemble de correspondances, aussi appelé alignement, peut par la suite être utilisé pour fusionner les ontologies, migrer des données

entre ontologies ou traduire des requêtes formulées en fonction d'une ontologie vers une autre.

A l'heure actuelle, ce n'est plus le manque des méthodes d'alignement qui pose problème mais, de pouvoir détecter l'ensemble d'alignement optimal et le plus similaire. Différentes méthodes sont proposées dans la littérature, cependant elles requièrent toutes des temps de calcul important. Il est nécessaire d'appliquer des algorithmes d'optimisation exploitant la structure de graphe particulière des ontologies pour accélérer la détection des correspondances. En réponse à cet enjeu, nous essayons d'appliquer les colonies de fourmis pour optimiser le résultat d'alignement.

Bien que l'algorithme de colonie de fourmis soit conçu au départ pour le problème du voyageur de commerce, il offre finalement beaucoup de souplesse. Notre choix est motivé par la flexibilité de cette méta-heuristique qui rend possible son application à différents problèmes qui ont pour point commun d'être NP-difficile (appariement de graphes). Ainsi l'utilisation d'un modèle parallèle (colonies des fourmis) diminuent le temps de calcul et améliorent la qualité des solutions obtenues par le processus d'alignement.

4.2 Travaux dans le domaine

Ces dernières années des nouvelles méthodes ont été mise en place pour l'alignement des ontologies. La section suivante passe en revue quelques méthodes d'alignement qui utilisent les algorithmes d'appariement de graphes.

4.2.1 EDOLA *[Zghal et Ben Yahia, 2007]*

C'est une nouvelle méthode d'alignement d'ontologies décrites en OWL-Lite. Les ontologies à apparier sont transformées sous forme d'un graphe, qui permet de représenter toutes les informations contenues dans l'ontologie OWL-Lite. EDOLA est une approche reposant sur un modèle de calcul des similarités locale et globale. Le calcul de la similarité locale s'effectue en deux étapes successives. La première étape, permet de calculer la similarité linguistique pour chaque couple de nœuds appartenant à la même catégorie. La deuxième étape, permet de calculer la similarité structurelle en exploitant la structure du voisinage.

Dans la méthode EDOLA, l'algorithme ***glouton*** est implémenté pour effectuer de choix locaux. En effet, lorsqu'il est confronté à un choix, il prend ce qui lui semble le meilleur pour avancer, et espère ensuite que la succession de choix locaux contribue à une solution optimale. Il choisit un couple d'entité ayant la plus grande similarité et qui est supérieur ou égal à un seuil fixé, et continue la vérification pour chaque couple jusqu'à ce qu'il n'existe plus de couples ayant une mesure de similarité supérieure au seuil.

4.2.2 ASCO3 *[Bach et al, 2004]*

L'idée principale de l'algorithme ASCO3 est d'exploiter l'expressivité du langage d'ontologie OWL pour déduire la similarité entre les entités de deux ontologies représentées par un réseau sémantique.

Les liens ont des sémantiques spécifiées par les primitives du langage. La connexion entre les nœuds reflète la modélisation de l'ontologie dans le domaine pour lequel l'ontologie a été conçue et représentée. De cette observation, il s'en suit que si deux ontologies dans le même domaine sont similaires, leurs réseaux sémantiques sont aussi d'une certaine façon similaires. Ainsi, les nœuds (entités) se trouvant aux places correspondantes dans les parties similaires de deux réseaux sémantiques, sont considérés similaires. L'algorithme ASCO3 cherche donc les parties communes les plus grandes de deux réseaux sémantiques représentant les deux ontologies, et ensuite décide si les entités aux positions correspondantes dans les parties trouvées sont similaires ou pas (***sous-graphe commun le plus grand des deux graphes*****).**

4.3 Approche proposée

L'originalité de notre approche réside dans le fait d'appliquer les méta-heuristiques dans la tâche d'alignement des ontologies. Donc l'objectif est de mettre en correspondances les deux ontologies décrites en OWL représentées par des graphes et d'optimiser par la suite ces correspondances afin d'obtenir un alignement optimal et plus similaire. L'approche est composée de trois phases principales, contenants les différents traitements dirigés par le système (voir la *Figure 4.1*).

4.3.1 Format des ontologies en entrée

Dans cette section, on va présenter l'ensemble d'arguments justifiant notre motif du choix du format d'ontologies.

A. Critères de choix des ontologies

Les ontologies à aligner doivent :

1. Etre de bonne qualité : bonne conception, couverture significative du domaine de l'ontologie, syntaxe correcte du langage dans lequel est spécifiée l'ontologie.

2. Décrire des situations du monde réel.

3. Permettre de discriminer entre les différentes approches d'alignement afin de mettre en évidence leurs forces et leurs faiblesses.

B. Motivation du choix de OWL

D'après une enquête publiée dans *[Sean & al, 2007]*, et en réponse à une question sur le format des ontologies les plus utilisées dans le contexte de l'alignement des ontologies, *OWL* (Ontology Web Language) s'est classe largement en tête de liste (voir *Figure 4.1*), et qui s'est pratiquement donne de bon résultats.

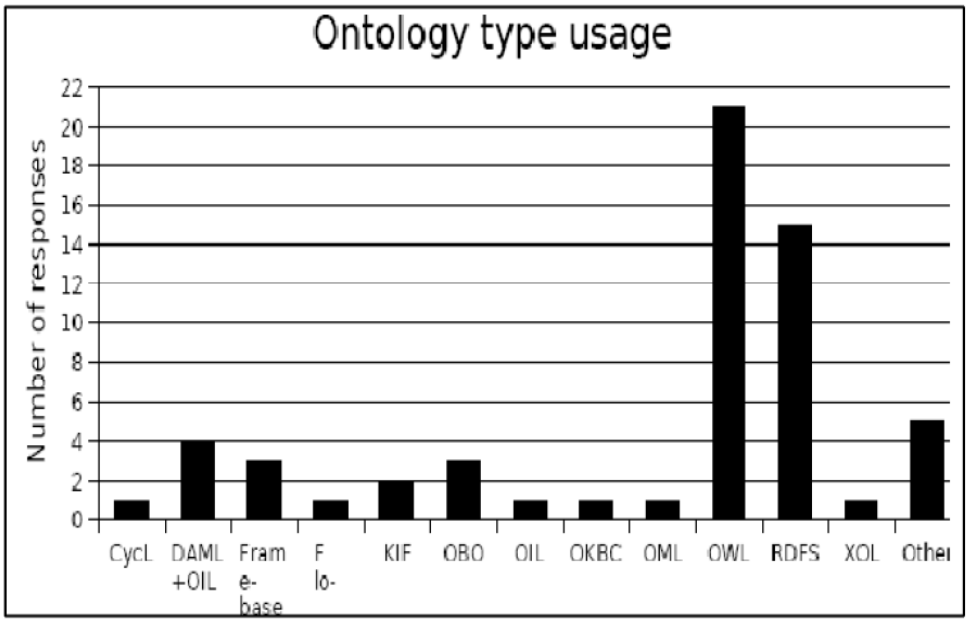


Figure4.1 : *Résultats d'une enquête sur les langages de spécification des ontologies à aligner*

4.3.2 Architecture du système

Le système est composé de trois phases principales, contenant les différents traitements dirigés par le système (voir la *Figure 4.2*) :

- Unité ***Pré-Alignement*** pour l'importation des ontologies et la transformation en graphe.
- Unité ***Processus-Alignement*** qui englobe:
  - L'extraction des concepts de chaque ontologie.
  - Le calcul de la similarité et l'alignement des ontologies
- Unité ***Post-alignement*** montre :
  - L'optimisation de l'alignement par l'application d'une méta-heuristique qui est les colonies de fourmis.

Que l'on va détailler comme suit :

PHASE1 : Pré-Alignement

Dans cette phase, nous identifions d'abord les informations d'entrée (Input) qui constituent essentiellement les structures destinées à être alignées. Puis nous formalisons ces structures sous forme de graphe.

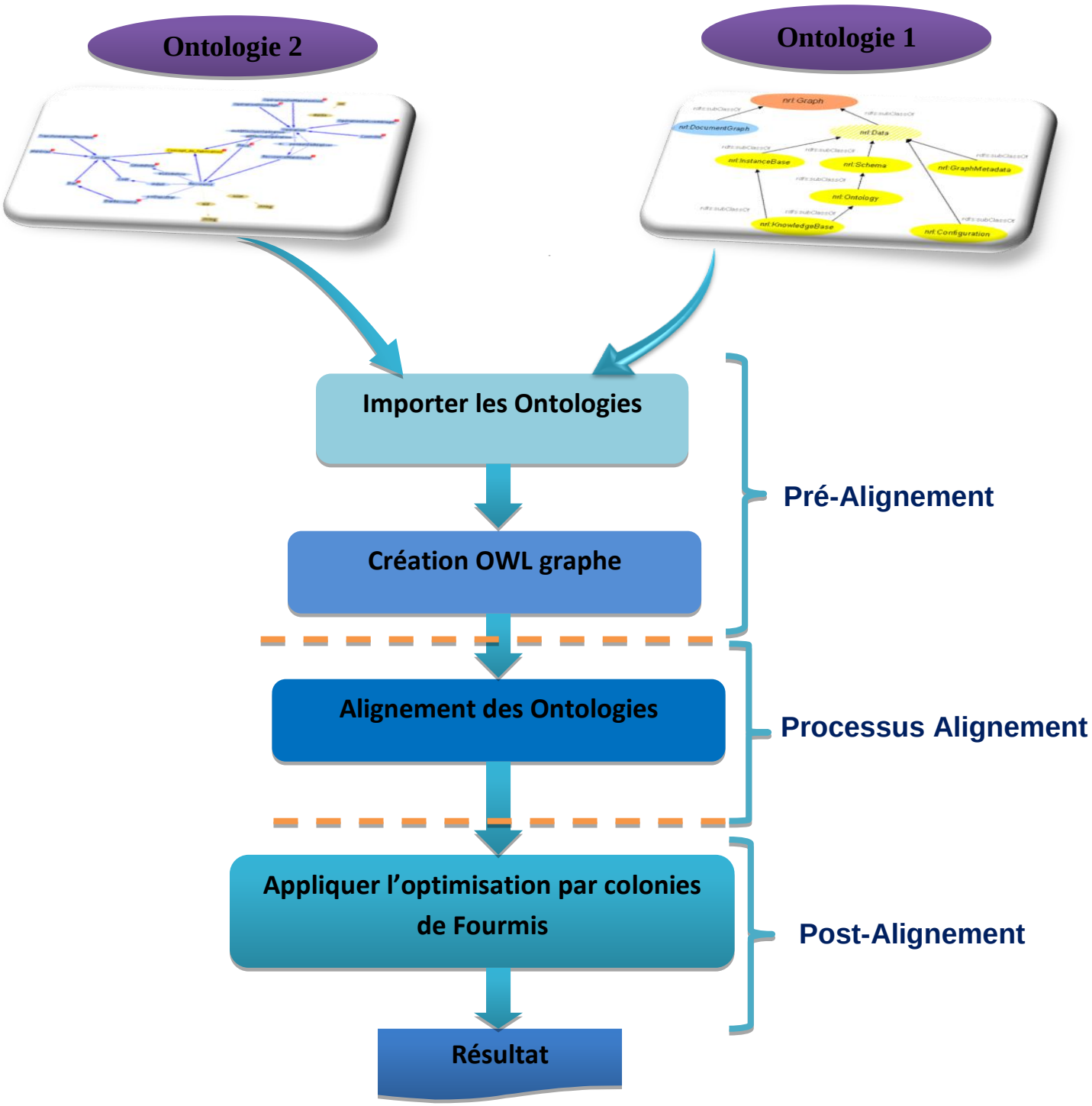


Figure 4.2: *Architecture de Système*

A. Importation des Ontologies

Dans ce module, nous avons utilisé la bibliothèque ***Jena*** (Cf. § chapitre 5, section 5.3), qui permet l'importation des ontologies via des modèles qui offrent des opérations de lecture et écriture de différents formats d'ontologies.

B. Création de OWLGraph

Dans cette section, une ontologie OWL est considérée comme un ensemble d'entité (classes, restrictions, propriétés, instances, etc.) et de liens sémantiques (instanciation, subsomption, équivalence, etc.) entre ces entités. La figure suivante présente une description générale d'une ontologie OWL.

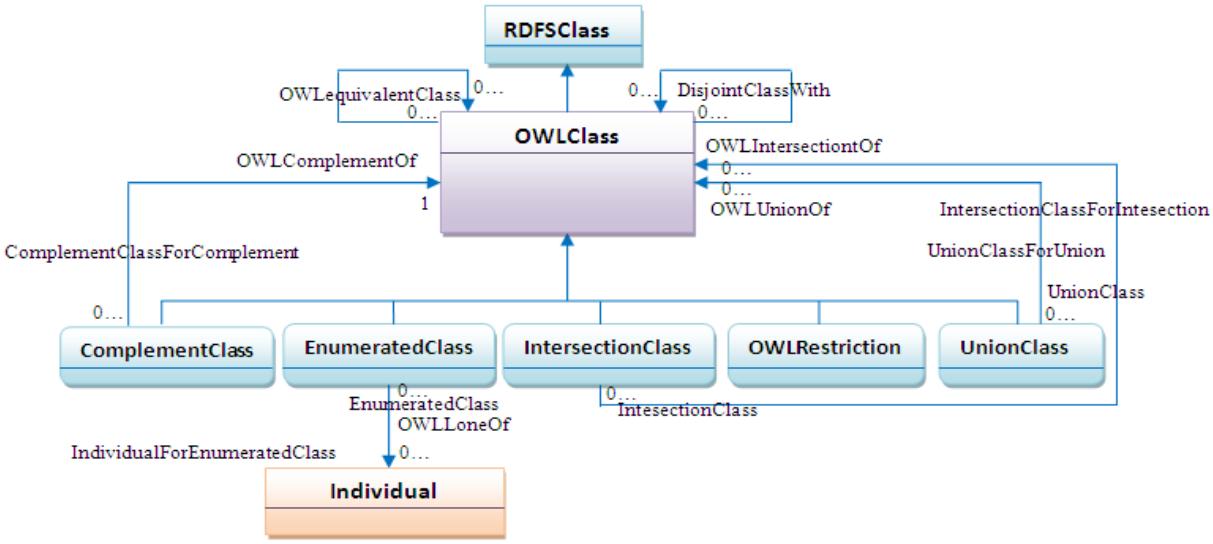


Figure 4.3 : *Diagramme de description d'OWL*

Pour pouvoir être alignées, les ontologies sont au préalable représentées sous forme de graphe. Les deux ontologies sont transformées automatiquement sous forme de deux graphes étiquetés appelés respectivement $OWLGrah_1$, $OWLGraph_2$. Ces deux graphes permettent de représenter toutes les informations contenues dans les deux ontologies décrites en OWL.

Les nœuds de chaque graphe représentent des entités d'une ontologie et possède deux étiquettes. L'une des étiquettes de nœud est le nom de l'entité représentée alors que l'autre identifie la catégorie de cette entité. Comme catégorie d'entité on peut avoir CLASSE, PROPRIETE, INSTANCE, etc.

Deux nœuds de graphe sont reliés par un arc si et seulement s'il existe un lien sémantique entre les entités qu'ils représentent. Chaque arc est étiqueté par le nom du lien qu'il encode. La figure suivante montre le schéma général de transformation.

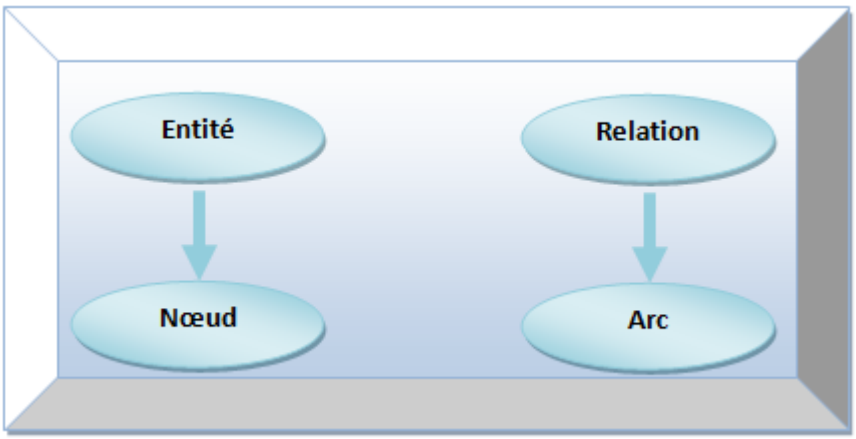


Figure 4.4 : *Schéma générale de transformation*

- **Transformation des Classes :** Dans la figure suivante, nous présentons le digramme UML des classes OWL :

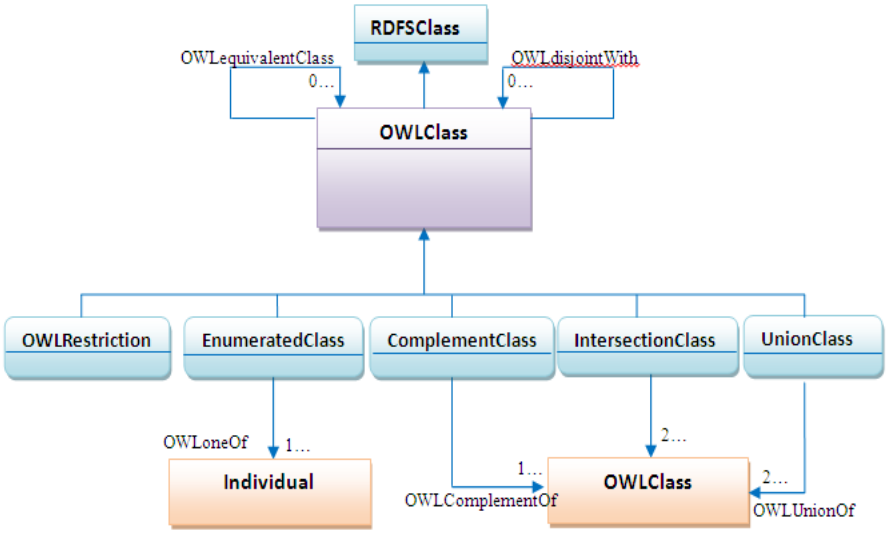


Figure 4.5: *Diagramme UML des classes OWL*

L'ensemble de ces classes peut être transformé en éléments de OWLGraph selon le tableau suivant :

Nom de l'élément	Catégorie	Elément de graphe
OWLOntology	Class	Nœud
OWLClass	Class	Nœud
ComplementClass	Class	Nœud
EnumeratedClass	Class	Nœud
DisjointClass	Class	Nœud
IntersectionClass	Class	Nœud
RestrictionClass	Class	Nœud
UnionClass	Class	Nœud

Tableau 4.1 : *Tableau d'élément Class OWL*

➢ **Transformation des Restrictions** : Dans la figure suivante, nous présentons le digramme UML des différentes restrictions d'OWL :

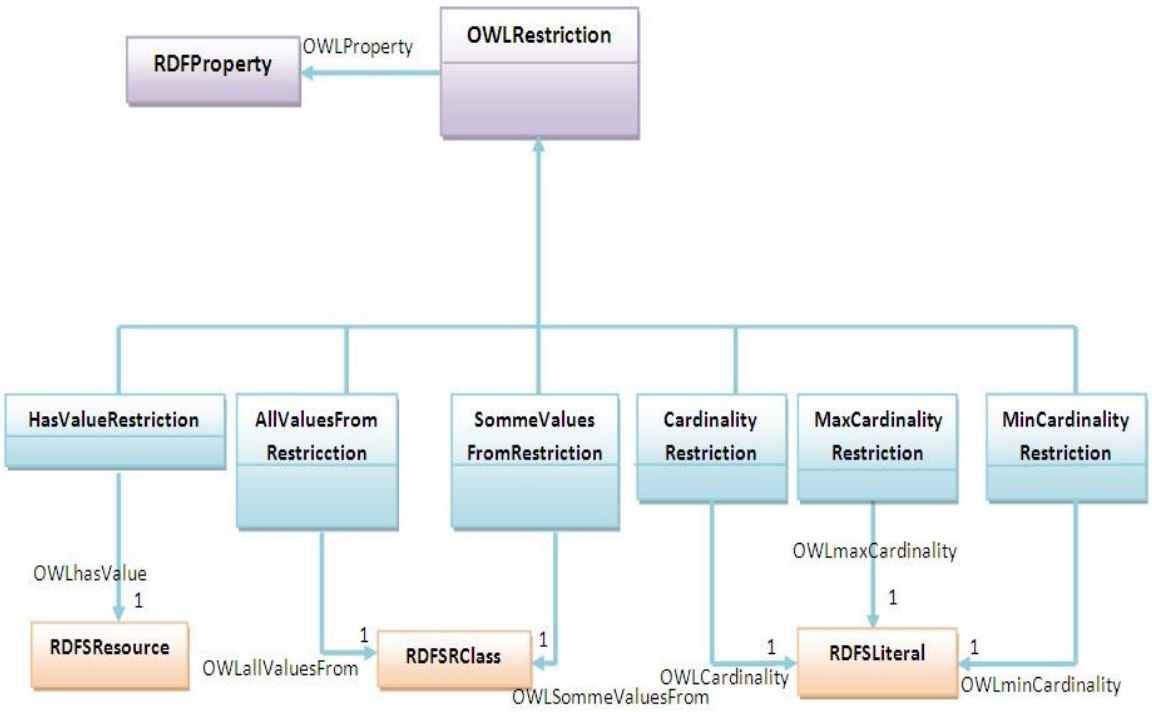


Figure 4.6 : *Diagramme UML des restrictions d'OWL*

L'ensemble de ces restrictions peut être transformé en éléments de OWLGraph selon le tableau suivant :

Nom de l'élément	Catégrie	Elément de graphe
OWLRestriction	Restriction	Nœud
AllValuesFromRestriction	Restriction	Nœud
SomeValuesFromRestriction	Restriction	Nœud
CardinalityRestriction	Cardinality	Nœud
MaxCardinalityRestriction	Cardinality	Nœud
MinCardinalityRestriction	Cardinality	Nœud
OWLDataRange	DataType	Nœud

Tableau 4.2: *Tableau d'élément Restriction OWL*

➢ **Transformation des Propriétés :** Dans la figure suivante, nous présentons le digramme UML de quelques propriétés d'OWL :

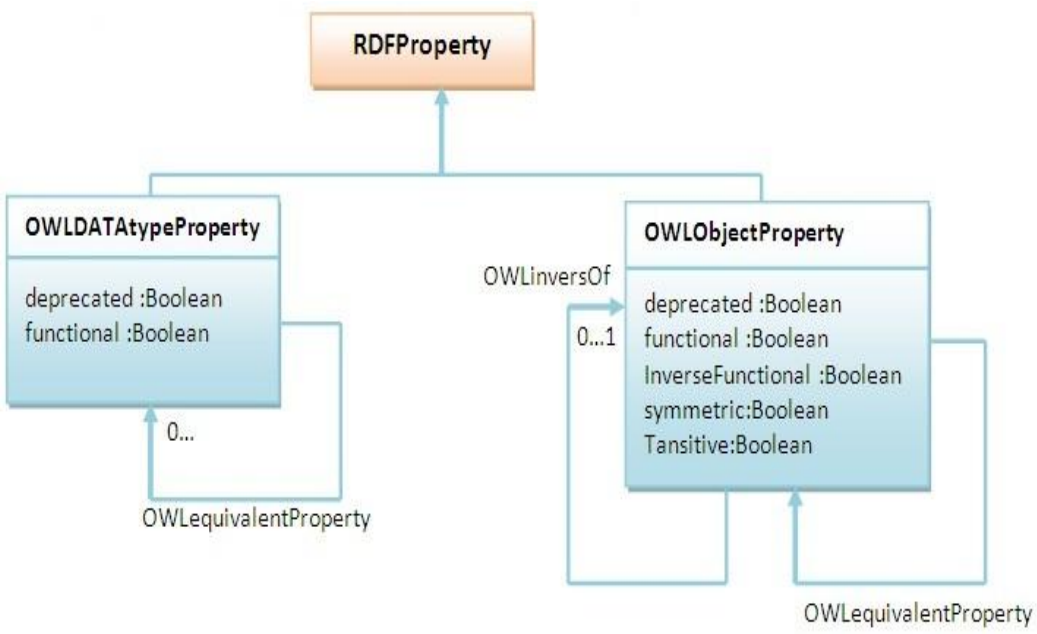


Figure 4.7 : *Diagramme UML des Propriétés OWL*

Le tabeau suivant présente la transformation des différentes propriétés d'OWL en élément de graphe :

Nom de l'élément	Catégorie	Elément de graphe
Property	Property	Nœud
OW LAnnotationProperty	Property	Nœud
OW LOntologyProperty	Property	Nœud
FunctionalProperty	Property	Nœud
OW LDatatypeProperty	Property	Nœud
OW LObjectProperty	Relation	Arc
InversFunctionalProperty	Property	Nœud
SY mmetricProperty	Property	Nœud
TransitiveProperty	Property	Nœud

Tableau 4.3: *Tableau d'élément Propriétés OWL*

- **Transformation des Instances :** le tableau suivant présente les différentes transformations des proprités des instances en éléments de OWLGraph,

Nom de l'élément	Catégorie	Elément de graphe
Individual	Object	Nœud
OWL :sameAs	Property Instance	Nœud
OWL :differentFrom	Property Instance	Nœud
OWL :AllDifferent	Property Instance	Nœud
Individual Value	Value	Nœud

Tableau 4.4: *Tableau d'élément des instances en OWL*

- **Transformation des Relations :**

  - ***Subsumption*** : Class-Class, Relation-Property, Relation-Relation
  - ***Attribution*** : Property-Class, Property Instance-Object,
  - ***Instantiation*** : Object-Class
  - ***Valuation*** : Object-PropertyInstance, Value- PropertyInstance
  - ***Domain***: Class-Relation,

- ***Range***: Class-Relation, Datatype-Relation,
- ***All***: Class-Property, Datatype-Property,
- ***Card***: Cardinality-Property,

Le graphe doit garder la même structure et hiérarchie de l'ontologie qu'il présente. La figure suivante présente une partie de OWLGraph d'une ontologie conference. Dans cette figure, les nœuds représentent des entités d'ontologies et sont étiquetés par la ***catégorie de l'entité*** (classe, propriété, relation, mots et cardinalité) et le nom de l'entité. Les arcs représentent les liens entre entités et sont étiquetés par les noms des liens.

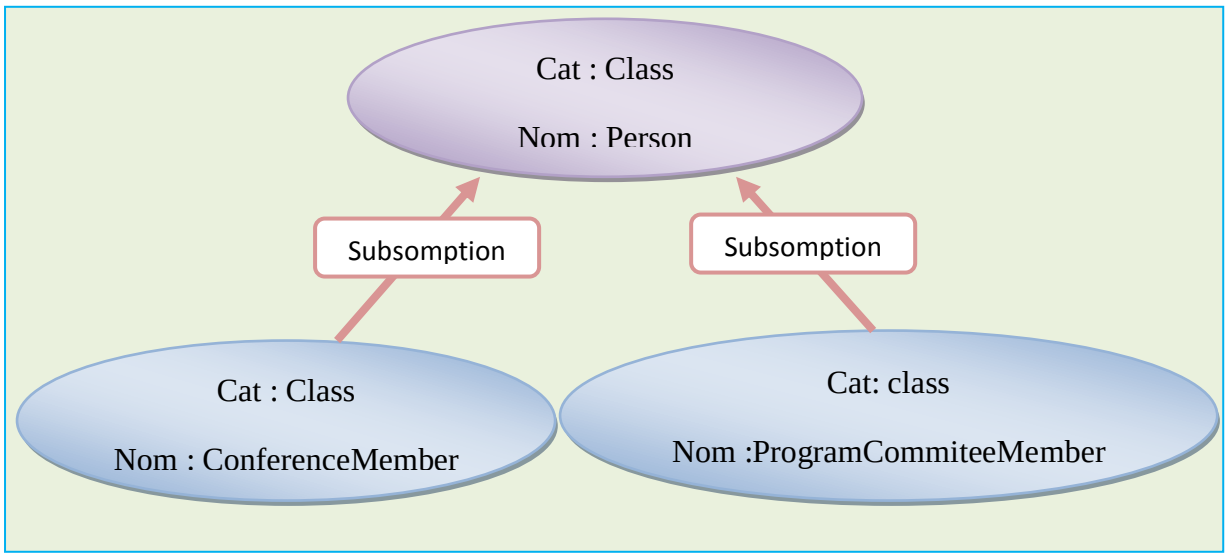


Figure 4.8 : *Extrait d'un OWLGraph*

PHASE2:Processus Alignement

Dans ce module, nous utilisons les traitements faits dans la phase précédente, il se divise lui même en deux parties :

A. La mise en Correspondance

La mise en correspondances des entités ou l'alignement des entités s'effectué en utilisant une mesure de similarité linguistique des couples d'entités de même catégorie (classe propriété et instance). Elle est calculée par l'intermédiare de fonctions de Editdistance (Cf. § chapitre 2, section 2.5.1), qui est une distance normalisée, utilisée pour la conparaison des chaînes de caractères. Notre choix est mitivé par sa robustesse face aux fautes d'orthographes.

Donc la phase de mise en coresspodaces permet de calculer les similarités des couples de nœuds des deux ontologies. Elle prend en entrée les deux ontologies Ontologie1 et ontologie2 à aligner, représentées sous la forme de deux graphes OWLGraph, ainsi la fonction de similarité linguistique à utiliser, et donne en retour une matrice de coresspondances entre les entités des deux ontologies.

Algorithme_Mise_En_Coresspondance

Entrée : $OWLGraph_1$; $OWLGraph_2$; //graphes des deux ontologies à aligner,
Fonction ; // la fonction de calcul de similarité,
Sortie : Matrice ; // matrice de similarité

Début

/*parcours des nœuds de la première ontologie*/
Pour chaque $nœud_1$ du $OWLGraph_1$ **faire**
/*parcours des nœuds de la deuxième ontologie*/
Pour chaque $nœud_2$ du $MPV\text{-}Graph_2$ **faire**
Sim=Mise_En_Coresspondace ($nœud_1$, $nœud_2$, fonction) **;**
Matrice[$nœud_1$**,** $nœud_2$**] = Sim;**
Fin pour;
Fin pour;
Retourner Matrice;
Fin.

B. Filtrage des Couples Candidats à Base de Seuil

Pour sélectionner les couples de nœuds candidats pour l'optimisation, nous déterminons une valeur de seuil située entre 0 et 1.

Après le calcul de similarité entre chaque couple de nœuds provenant de deux ontologies OWL en entrée, on a une matrice contenant ces valeurs de similarité. Le filtrage consiste à éliminer les couples dont la valeur de similarité est inférieure à ce seuil. Nous obtenons donc une matrice des couples les plus similaires. Où nous pouvons trouver qu'un nœud du $OWLGraph_1$ peut apparier à un ou plusieurs nœuds du $OWLGraph_2$ ou inversement (i.e., appariement multivoque).

La matrice de similarité est exploitée par la suite dans l'optimisation d'alignement. La figure suivante présente le résultat d'alignement où une entité de la première ontologie est appariée à plusieurs entités de la deuxième ontologie.

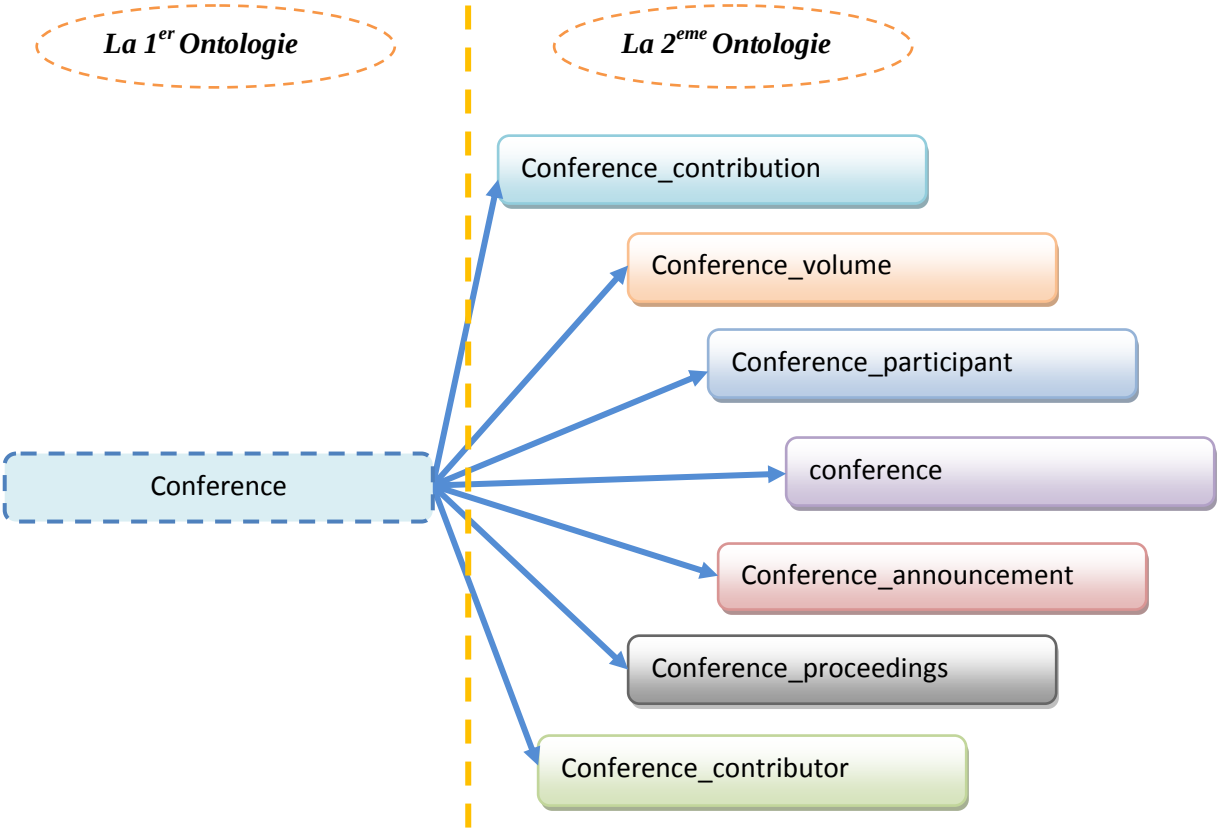


Figure 4.9 : *Extrait d'un Résultat d'Alignement*

PHASE 03 : Post-Alignement

La plupart des méthodes d'alignement existantes dans la littérature, utilisent seulement des filtres pour optimiser le résultat d'alignement. Dans notre approche, nous adaptons un algorithme d'optimisation par les colonies de fourmis pour réduire le résultat d'alignement.

A. Optimisation d'alignement

Dans notre contexte, le problème d'alignement d'ontologies est un problème *d'appariements multivoques* de deux graphes *OWLGraph*$_s$, ce problème se ramène au problème de sélection de sous-ensemble. Afin de trouver les meilleures correspondances entre les deux ontologies, nous adaptons l'algorithme d'optimisation par colonies de fourmis (ACO) proposé dans *[Solnon, 2005]*.

➢ ***Construction du graphe de similarité*** : le graphe de similarité est un graphe orienté, étiqueté et pondéré. Il est le cœur de l'algorithme d'optimisation dans notre contribution.

Soient $OWLGraph_1$ et $OWLGraph_2$ deux graphes d'ontologies et S le graphe de similarité correspondant. Les nœuds de S sont des couples $(v_1, v_2) \in OWLGraph_1 \times OWLGraph_2$ tels que $v_1$et v_2 représentent des entités appartenant à la même catégorie. La catégorie du nœud (v_1, v_2) est la catégorie commune de ses composantes. Contrairement aux nœuds du graphe OWLGraph, on associe aux nœuds du graphe de similarité une valeur représentant leur similarité. Cependant, le poids de chaque arc est affecté en fonction de son étiquette. Tous les arcs de même étiquette ont le même poids.

Un arc d'étiquette l existe entre deux nœuds (v_1, v_2) et (v'_1, v'_2) du graphe de similarité si et seulement s'il existe simultanément un arc d'étiquette l entre les nœuds v_1 et v'_1 de $OWLGraph_1$ d'une part et les nœuds v_2 et v'_2 de $OWLGraph_2$ d'autre part.

L'ensemble des nœuds du graphe de similarité représente ainsi l'ensemble de tous les couples d'entités qui sont potentiellement mis en alignement dans l'étape de mise en correspondance. En effet, deux nœuds sont alignables si les entités qu'ils représentent appartiennent à la même catégorie et partagent le même ensemble de liens.

➢ ***Sélection d'un sous ensemble dans le graphe de similarité*** : Rappelons que nous cherchons un sous ensemble dans le graphe de similarité et nous adaptons l'algorithme proposé dans *[Solnon, 2005]*.

A chaque cycle de l'algorithme, chacune des fourmis construit un sous-ensemble. En partant d'un sous-ensemble S_k vide, les fourmis ajoutent à chaque itération un couple de nœuds à S_k choisi parmi l'ensemble des couples non encore sélectionnés. Le couple de nœuds à ajouter à S_k est choisi selon une probabilité qui dépend des traces des phéromones et deux facteurs heuristiques, l'un vise à favoriser les couples qui ont la similarité la plus forte et l'autre vise à favoriser les couples qui font le plus augmenter la similarité. Une fois que chaque fourmi a construit son sous-ensemble, une procédure de recherche locale est lancée afin d'essayer d'améliorer la qualité du meilleur sous-ensemble trouvé lors de ce cycle. Les fourmis arrêtent leur construction quand tous les couples de nœuds candidats font décroitre la similarité de sous-ensemble ou quand les trois derniers ajouts n'ont pas permis d'accroître cette similarité.

D'une manière générale, l'algorithme ci-dessous explique en détaille la démarche :

Procédure-construction-sous-ensemble

Entrée : le graphe de similarité,

Sortie : un sous-ensemble le plus similaire,

Initialiser les traces de phéromone

Début

Répéter

Pour chaque fourmi *k* dans 1..*nbFourmis*, construire une solution S_k comme suit :

Choisir aléatoirement le premier couple

Ajouter le couple sélectionné à S_k

Candidats ← *tous les couples non encore sélectionnés*

Tant que *Candidats* $\neq \phi$ **faire**

Choisir un couple de nœuds de l'ensemble *Candidat* avec une Probabilité *P qui maximise la similarité*

Ajouter le couple choisis à S_k

Enlever *le couple choisis* de *Candidats*

Fin tant que

Fin pour

Mettre à jour les traces de phéromone en fonction de $\{S_1, \ldots, S_{nbfourmis}\}$

Jusqu'à nombre maximal de cycles atteint ou solution trouvée.

Fin.

➢ **Résultat de l'optimisation :** Le résultat de l'algorithme d'optimisation est un sous ensemble du graphe de similarité, qui représente la meilleure solution que les fourmis peuvent sélectionner lors de la construction de leurs solutions. Cette solution représente les meilleures correspondances entre les deux ontologies. Donc pour chaque nœud du graphe $OWLGraph_1$ qui appartient au sous ensemble trouvé, il y a un et seulement un nœud du graphe $OWLGraph_2$ qui lui correspond. La figure suivante présente le résultat de l'optimisation de l'alignement présenté dans la *Figure 4.8*.

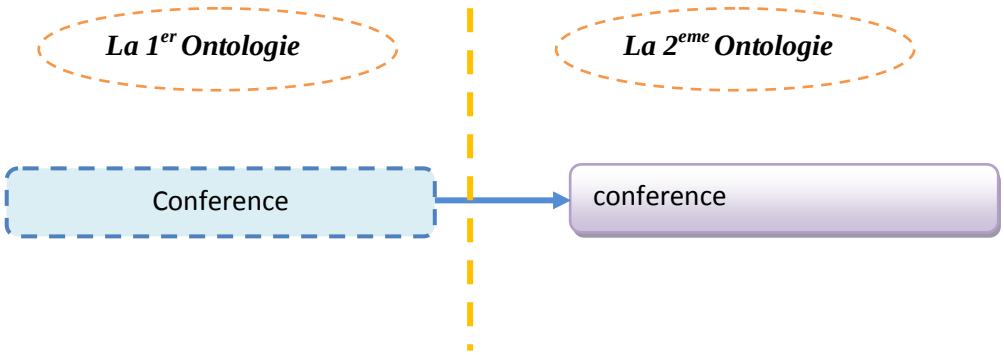


***Figure 4.10** : Résultat de l'optimisation*

Conclusion

Dans ce chapitre, nous avons présenté une approche d'alignement entre deux ontologies. L'approche proposée se compose de trois phases principales : (i) la phase de pré-alignement qui permet d'identifier les formats des ontologies en entrée et de les transformer en des structures de graphes étiqueté. (ii) Le processus d'alignement qui permet de détecter les appariements multivoques entre les deux graphes. (iii) La phase de post-alignement : dans cette phase, l'optimisation de l'appariement est effectuée par une technique de méta-heuristique de colonies de fourmis.

Dans le chapitre suivant, nous envisageons d'implémenter les différentes phases du processus d'alignement, et de calculer les différentes mesures d'évaluation pour évaluer la qualité d'alignement de la méthode.


Chapitre
V
Implémentation de L'application

Introduction

Dans ce chapitre nous allons aborder l'aspect implémentation de l'application réalisée ‹‹Application des Colonies de Fourmis pour L'optimisation de L'alignement des Ontologies››. Pour évaluer la qualité de notre contribution nous présenterons tout d'abord les métriques d'évaluations les plus utilisées. Puis nous détaillerons l'aspect matériel, l'environnement de développement et les différents modules qui composent le logiciel.

Ensuite la réalisation de l'approche proposée. Et à la fin nous exposons la discussion des résultats obtenus, qui ont été menées dans le but d'étudier le comportement et la performance de notre approche.

5.1 Mesures d'évaluation

La première phase dans le processus d'évaluation de la qualité d'alignement d'un système d'alignement consiste à résoudre le problème manuellement. Nous utilisant un alignement de référence, qui est construit à la main, et nous comparons les résultats produits par l'approche par rapport à cet alignement de référence.

Nous employons les mesures de ***précision***, de ***rappel*** et de ***f-mesure***. Ce sont les mesures bien utilisées dans le domaine de recherche d'information et ensuite appliquées dans le domaine d'alignement d'ontologies pour permettre une analyse fine des performances de système.

L'utilisation de ces mesures est nécessaire pour l'automatisation du processus de comparaison des méthodes d'alignement ainsi que pour l'évaluation de la qualité des alignements obtenus.

La comparaison du résultat de l'*alignement de référence* avec celui de l'*appariement obtenu* par la méthode d'alignement produit trois ensembles :

- ✓ N_{found} : représente l'ensemble des couples alignées avec la méthode d'alignement,
- ✓ $N_{expected}$: représente l'ensemble des couples appariés dans l'alignement de référence,
- ✓ et $N_{correct}$: est l'intersection des deux ensembles précédents. Il représente l'ensemble des couples appartenant à la fois à l'alignement obtenu et l'alignement de référence.

A partir de ces trois ensembles, nous pouvons obtenir les mesures suivantes :

➢ Le ***rappel*** est la proportion de correspondances correctes renvoyées par l'approche parmi toutes celles qui sont correctes (en incluant aussi des correspondances correctes que l'algorithme n'a pas détectées). Le rappel mesure l'efficacité d'un algorithme. ***Plus la valeur de rappel est élevée, plus le résultat de l'algorithme couvre toutes les correspondances correctes.***

- *Rappel*= $|N_{correct}| / |N_{expected}|$,

➢ La ***précision*** est la proportion des correspondances correctes parmi l'ensemble de celles renvoyées par l'approche. Cette mesure reflète la précision d'un algorithme. ***Plus la valeur de précision est élevée, plus le bruit dans le résultat de l'algorithme est réduit, et donc plus la qualité du résultat est imposante.***

- *Précision*= $| N_{correct}| / |N_{found} |$.

➢ La ***f-mesure*** est un compromis entre le rappel et la précision. Elle permet de comparer les performances des algorithmes par une seule mesure. La f-mesure est définie par :

- *F-mesure*= (2**Précision***Rappel*) / (*Précision*+*rappel*).

5.2 Aspect Matériels

Notre projet a été développé sur un **PC** portable pentium **4** avec un système d'exploitation **Windows 7** professionnel, un Microprocesseur dual core **2.0GHz** (Intel), **2Go** de **RAM,** un disque dur **250 Go**, un écran (15.6" LCD).

5.3 Environnement de développement

L'application réalisée a été implémentée avec le langage **JAVA** Version **6** sous l'environnement **Eclipse HELIOS.** Guidée par la bibliothèque **Jena** version **2.5.5.**

est un langage multi plate-forme qui permet, selon le slogan lancé par Sun Microsystems : ‹‹ de fonctionner dans tous les environnements››. L'objectif était de taille, puisqu'il impliquait la définition d'une *machine virtuelle java* (JVM) sur laquelle les programmes écrits devaient fonctionner, ainsi que la réalisation de cette machine virtuelle dans tous les

environnements concernés. Sun Microsystems se chargeait pas ailleurs de la réalisation d'une machine virtuelle dans les environnements Unix et Windows, laissant à d'autres les soins d'en faire autant pour les autres environnements (et en particulier Mac OS, le système d'exploitation des Macintosh). Dans le domaine multimédia, **JAVA** est en mesure d'ajouter des graphiques, des images, des animations et des séquences vidéo.

est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

est un API (application programming interface) pour la construction des applications du Web sémantique.

Jena est open source, son utilisation principale est d'aider à écrire du code Java qui gère les documents RDF, OWL et les descriptions. Il fournit un environnement de programmation pour la lecture et l'écriture des différents modèles d'ontologies : RDF, RDFS et OWL, et comprend une base de règles et de moteur d'inférence.

Pour bien manipuler cette API (voir *Figure 5.1*), on à munie notre mémoire par une annexe qui déroule l'installation et la configuration de Jena avec eclipse et d'autres outils que on a utilisé. (L'annexe est à la fin de ce mémoire).

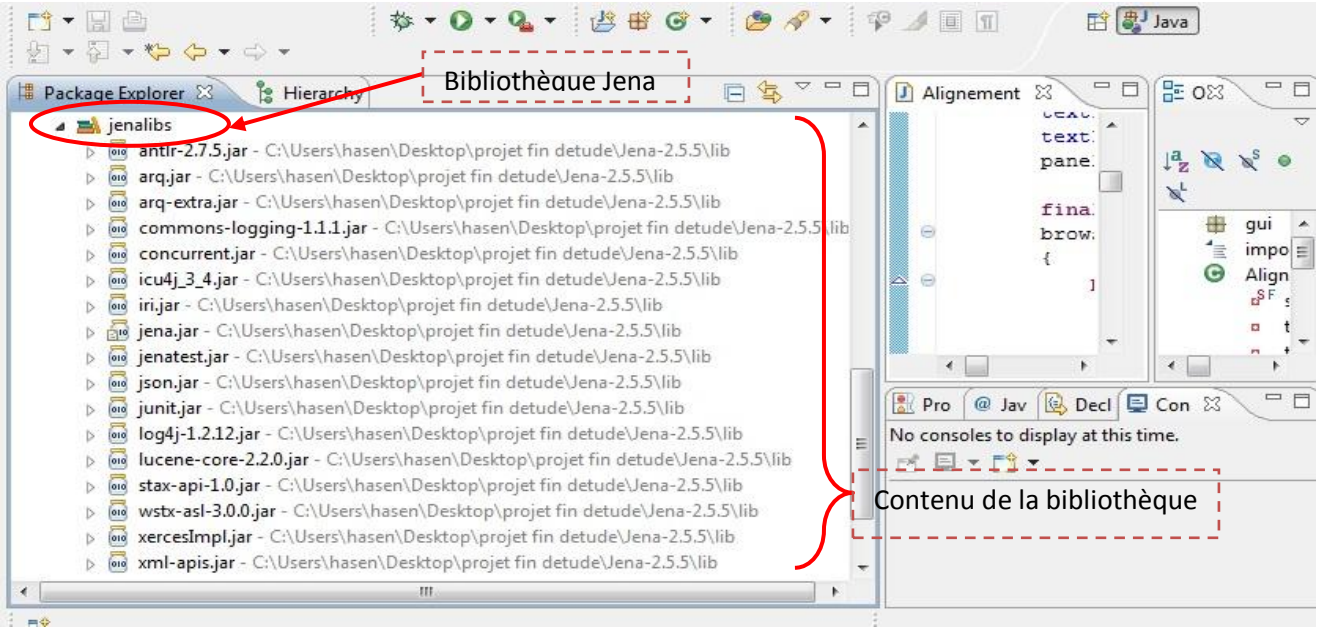


Figure 5.1 *: Contenu de la Bibliothèque Jena*

5.4 Réalisations logicielles

Les composants développés permettent de charger, mettre en correspondances les deux ontologies et d'optimiser le résultat d'alignement. L'approche proposée est réalisée, selon le prototype suivant :

- Chargement des ontologies
- Algorithme de création de *OWL-Graph*
- Algorithme de mise en correspondance
- Algorithme de filtrage
- Algorithme de création de graphe de similarité
- Algorithme d'optimisation par colonie de fourmis

5.5 Les Composants de l'application

La réalisation de cette application est basée sur la programmation orientée objet qui permet la manipulation des classes, et qui généralise la notion de structure. Les langages orientées objet ont

été développés pour faciliter l'écriture et améliorer la qualité des logiciels en terme de modularité.

Notre projet est composé de 17 packages, on citant :

- ✓ ***Package gui*** : qui assure l'interface du logiciel et permet l'accès à l'ensemble des fonctions utilisées dans le logiciel.
- ✓ ***Package AlignementOnto*** : il contient plusieurs classes qui coopérant entre eux pour la manipulation et l'alignement des ontologies, à fin de mettre les résultats dans un fichier texte.
- ✓ ***Package OwlGraph*** : qui fait l'extraction des nœuds et des arcs et ses propriétés à partir des ontologies à fin de les transformer en graphes.
- ✓ ***Package Editedistance***: englobe des classes permettent la normalisation des caractères et le calcule de la similarité terminologique.
- ✓ ***Package SimGraph :*** qui crée le graphe de similarité entre les deux $OWLGraph_s$.
- ✓ ***Package Optimisation*** : pilote une classe qui présente l'algorithme d'optimisation qui fonctionne en collaboration avec les autres packages.

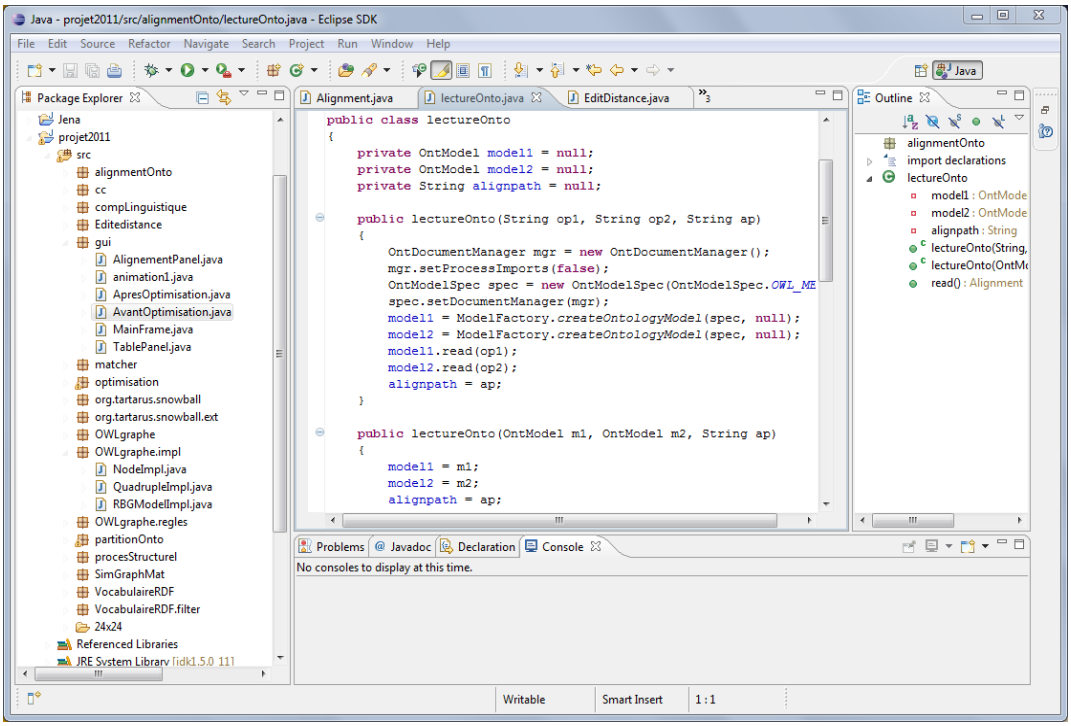


Figure 5.2 : *code de l'application dans l'environnement Eclipse*

5.6 Présentation de L'application

L'interface principale de l'application est présentée dans le *Figure 5.3***,** elle contient deux boutons le premier pour accéder aux fonctions de l'application et, le deuxième pour la quitter.

Figure 5.3*: Interface Graphique de l'application*

L'application offre trois panneaux qui permettent d'assurer divers fonctions :

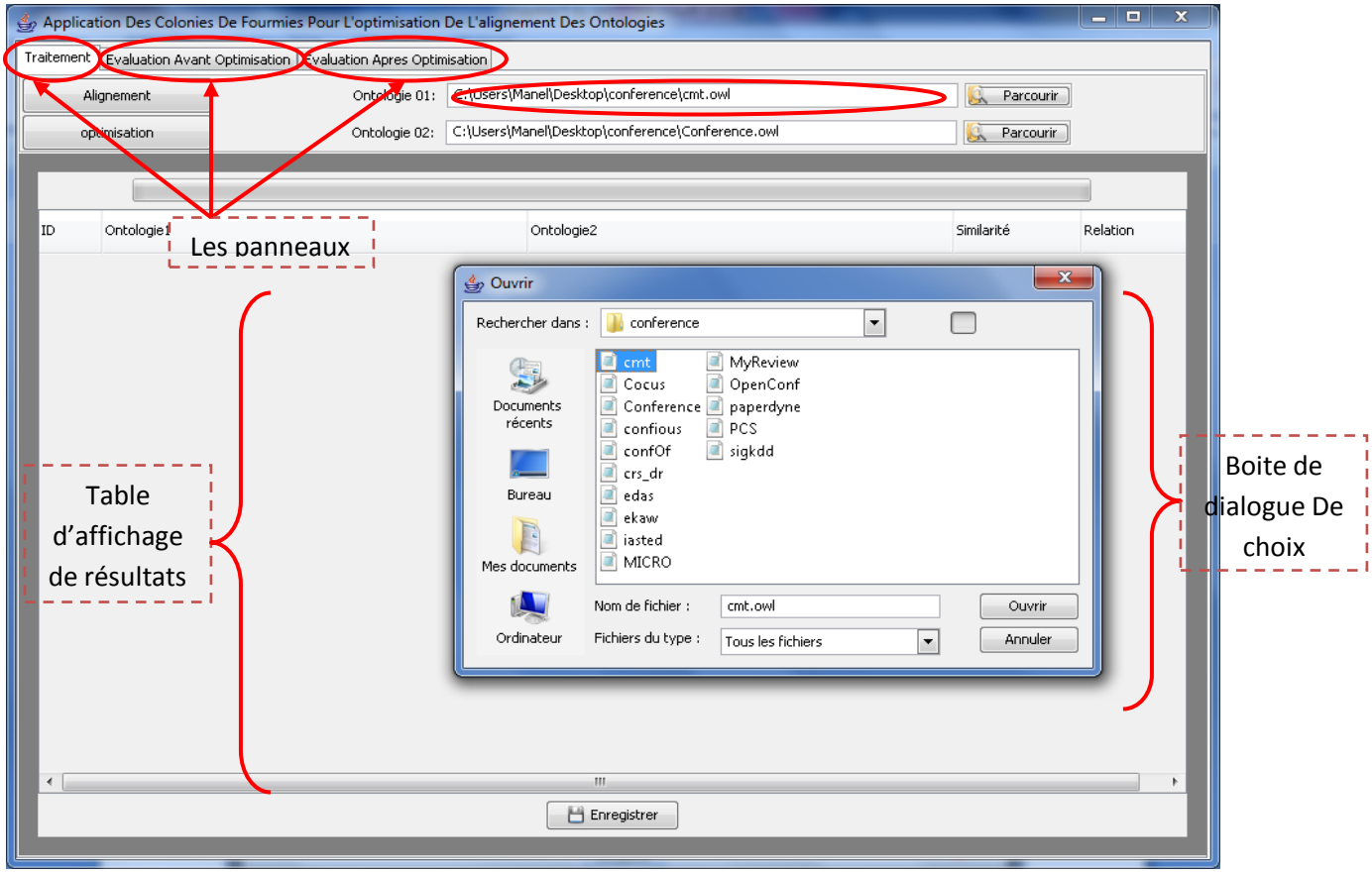


Figure 5.4 *: Interface principale de l'application*

5.6.1 Chargement des Ontologies

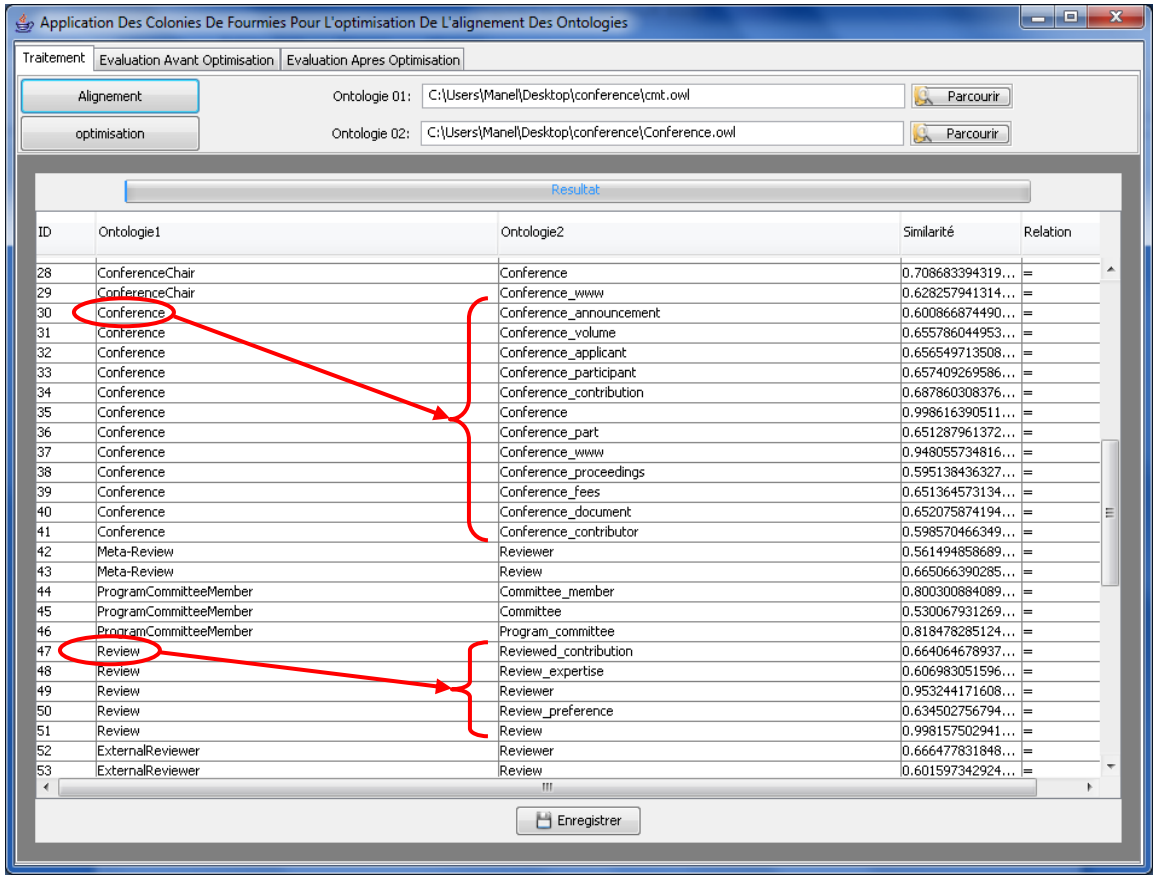
Concernant le chargement des ontologies, nous nous sommes appuyés sur l'API Jena, à partir de l'importation de ces deux bibliothèques principales `Model` et `ModelFactory`, qui fournissent des méthodes pour la création d'un model adapté au type d'ontologies et nous a permis d'ajouter les propriétés que on veut à l'ontologie traitée.

D'après le panneau "***Alignement***" on cliquons sur le bouton ***Parcourir*** pour le chargement des deux ontologies dans la barre de liens, après le fait de les récupérer de la base d'ontologie stockée sur disque. (Voir *Figure 5.4*)

5.6.2 Alignement des Ontologies

Pour avoir le résultat d'alignement des deux ontologies, il faut qu'une simple clique sur le bouton ***Alignement*** et le résultat s'affiche dans une table dans la même fenêtre. A partir de cette

fenêtre on peut enregistrer le résultat obtenu par une clique sur le bouton ***Enregistrer***. La figure ci-dessous montre le résultat d'un alignement où apparait l'appariement multivoque.

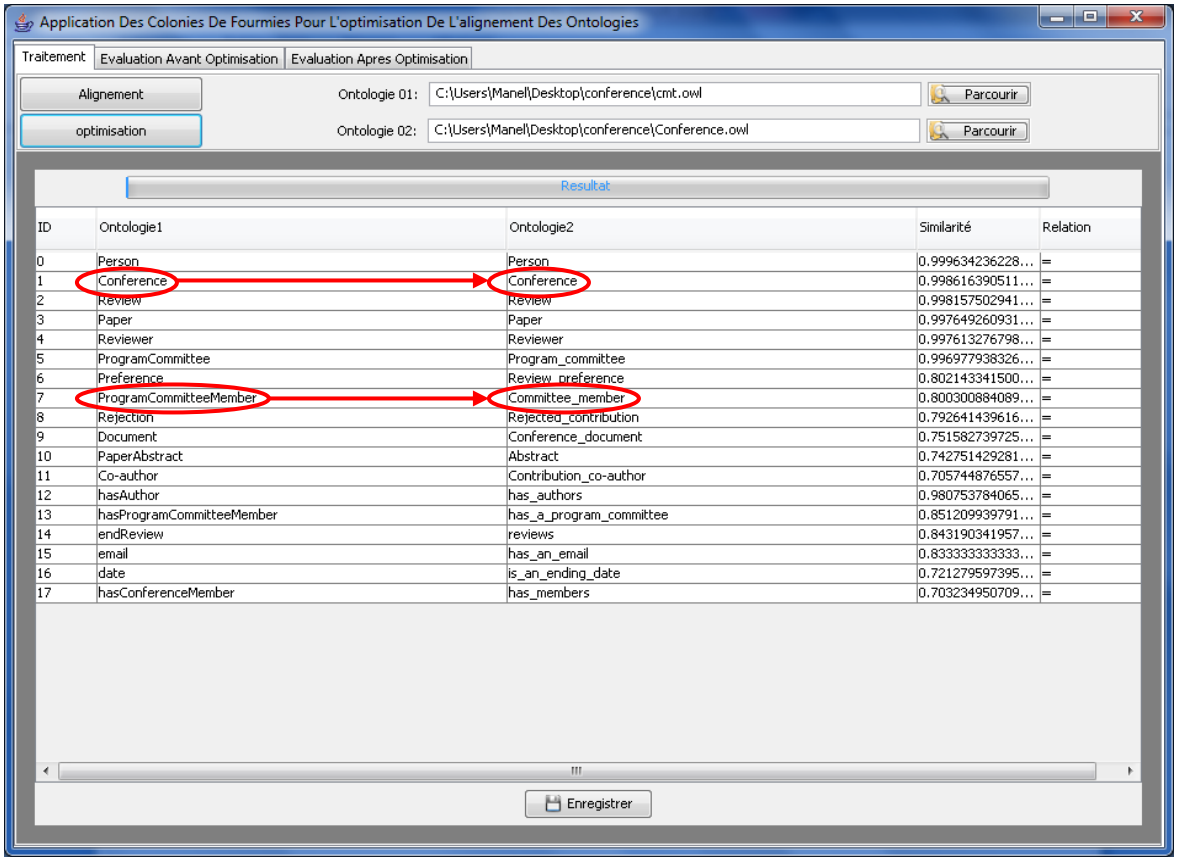


ID	Ontologie1	Ontologie2	Similarité	Relation
28	ConferenceChair	Conference	0.708683394319...	=
29	ConferenceChair	Conference_www	0.628257941314...	=
30	Conference	Conference_announcement	0.600866874490...	=
31	Conference	Conference_volume	0.655786044953...	=
32	Conference	Conference_applicant	0.656549713508...	=
33	Conference	Conference_participant	0.657409269586...	=
34	Conference	Conference_contribution	0.687860308376...	=
35	Conference	Conference	0.998616390511...	=
36	Conference	Conference_part	0.651287961372...	=
37	Conference	Conference_www	0.948055734816...	=
38	Conference	Conference_proceedings	0.595138436327...	=
39	Conference	Conference_fees	0.651364573134...	=
40	Conference	Conference_document	0.652075874194...	=
41	Conference	Conference_contributor	0.598570466349...	=
42	Meta-Review	Reviewer	0.561494858689...	=
43	Meta-Review	Review	0.665066390285...	=
44	ProgramCommitteeMember	Committee_member	0.800300884089...	=
45	ProgramCommitteeMember	Committee	0.530067931269...	=
46	ProgramCommitteeMember	Program_committee	0.818478285124...	=
47	Review	Reviewed_contribution	0.664064678937...	=
48	Review	Review_expertise	0.606983051596...	=
49	Review	Reviewer	0.953244171608...	=
50	Review	Review_preference	0.634502756794...	=
51	Review	Review	0.998157502941...	=
52	ExternalReviewer	Reviewer	0.666477831848...	=
53	ExternalReviewer	Review	0.601597342924...	=

Figure 5.5 : *Résultat de l'alignement de deux ontologies*

5.6.3 Optimisation d'alignement

Pour aperçu le résultat d'optimisation d'alignement, on clique sur le bouton ***Optimisation*** dans le panneau "***Alignement***", avec les même ontologies précédentes sans les recharger et le résultat s'affiche dans la même table des résultats. La figure ci-dessous montre le résultat obtenu après optimisation.



ID	Ontologie1	Ontologie2	Similarité	Relation
0	Person	Person	0.999634236228...	=
1	Conference	Conference	0.998616390511...	=
2	Review	Review	0.998157502941...	=
3	Paper	Paper	0.997649260931...	=
4	Reviewer	Reviewer	0.997613276798...	=
5	ProgramCommittee	Program_committee	0.996977938326...	=
6	Preference	Review_preference	0.802143341500...	=
7	ProgramCommitteeMember	Committee_member	0.800300884089...	=
8	Rejection	Rejected_contribution	0.792641439616...	=
9	Document	Conference_document	0.751582739725...	=
10	PaperAbstract	Abstract	0.742751429281...	=
11	Co-author	Contribution_co-author	0.705744876557...	=
12	hasAuthor	has_authors	0.980753784065...	=
13	hasProgramCommitteeMember	has_a_program_committee	0.851209939791...	=
14	endReview	reviews	0.843190341957...	=
15	email	has_an_email	0.833333333333...	=
16	date	is_an_ending_date	0.721279597395...	=
17	hasConferenceMember	has_members	0.703234950709...	=

Figure 5.6 *: Résultat d'Optimisation*

5.7 Démarche Expérimentale

Notre système a été appliqué sur le jeu de test « Conférence », ce jeu de test est constitué des ontologies hétérogènes qui représentent l'organisation de conférences et qui sont utilisées dans des applications réelles.

Pour examiner le comportement du système, nous avons choisi deux ontologies : ''cmt et conference'' parmi les ontologies du jeu de test conférence. La figure suivante représente l'ontologie ''cmt'' dans visualisée par OWLViz,

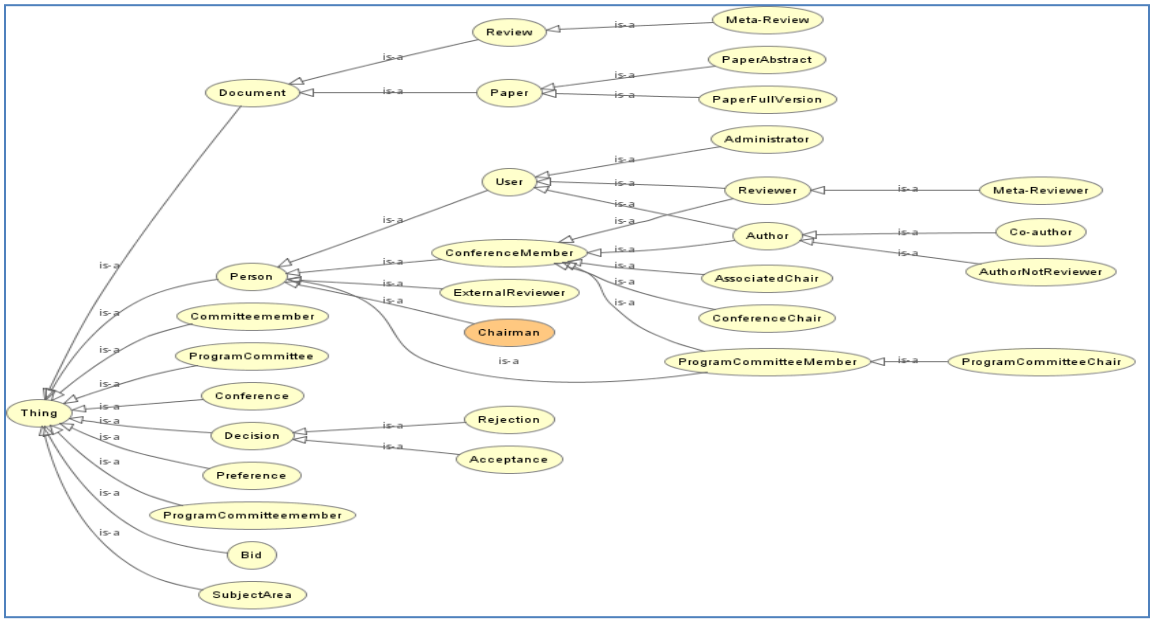


Figure 5.7 : *La hiérarchie des classes de l'ontologie ''cmt''*

La figure suivante représente l'ontologie ''conference'' dans visualisée par OWLViz,

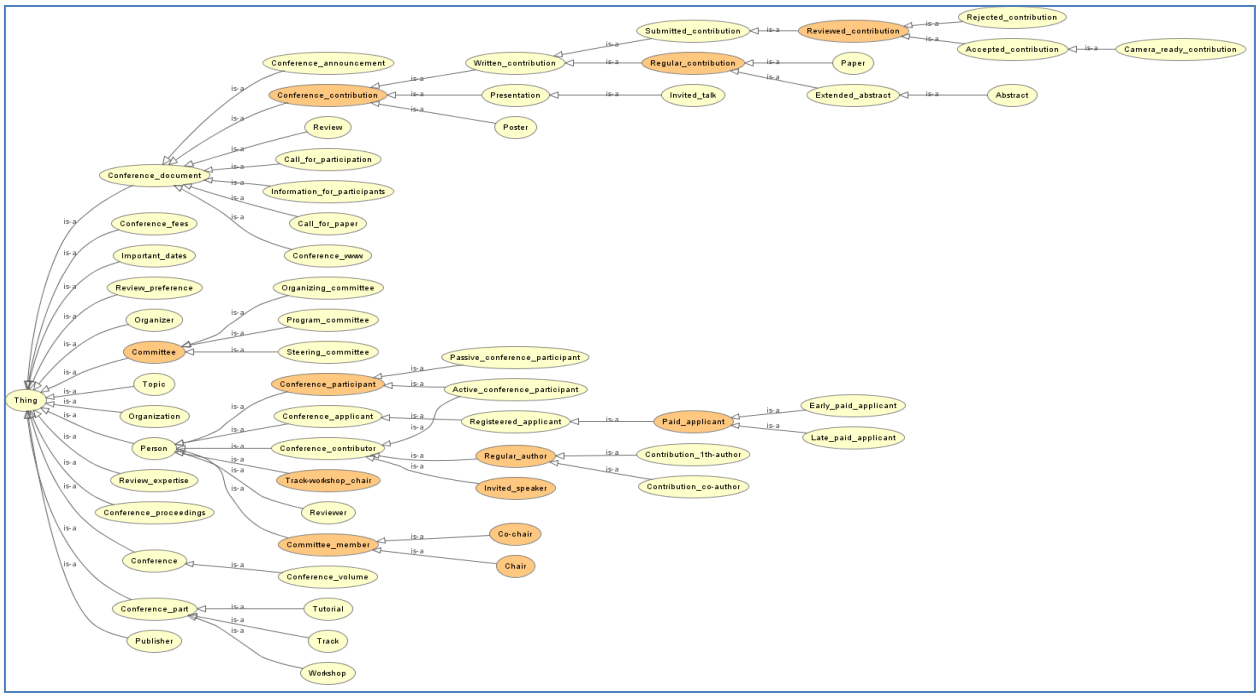


Figure 5.8 : *La hiérarchie des classes de l'ontologie ''conference''*

Les expérimentations contiennent l'analyse des résultats obtenus sur l'optimisation d'alignement des ontologies.

Dans la première phase d'évaluation nous résolvons le problème manuellement. Le résultat de cette phase est un alignement de référence (construit à la main), et nous comparons les résultats produits par le système par rapport à cet alignement de référence. L'alignement de référence est stocké dans un *fichier de référence*, ce fichier est chargé par le système au moment d'évaluation. Ensuite, nous avons réalisé deux essais afin d'évaluer la performance de notre système en produisant des alignements entre les deux ontologies choisies.

La figure suivante représente une partie du fichier de référence des ontologies ''cmt'' et ''conference''

```
<map>
  <Cell>
    <entity1 rdf:resource="http://cmt#Person"/>
    <entity2 rdf:resource="http://conference#Person"/>
    <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.999634236228724</measure>
    <relation>=</relation>
  </Cell>
</map>
<map>
  <Cell>
    <entity1 rdf:resource="http://cmt#Conference"/>
    <entity2 rdf:resource="http://conference#Conference"/>
    <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.9986163905112291</measure>
    <relation>=</relation>
  </Cell>
</map>
<map>
  <Cell>
    <entity1 rdf:resource="http://cmt#Review"/>
    <entity2 rdf:resource="http://conference#Review"/>
    <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.9981575029415095</measure>
    <relation>=</relation>
  </Cell>
</map>
<map>
  <Cell>
    <entity1 rdf:resource="http://cmt#Paper"/>
    <entity2 rdf:resource="http://conference#Paper"/>
    <measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.9976492609311831</measure>
    <relation>=</relation>
  </Cell>
</map>
```

Figure 5.9: *une partie du fichier référence des ontologies 'cmt' et 'conference'*

Dans la deuxième phase, nous analysons l'impact de l'optimisation sur les résultats de l'alignement. Afin d'aboutir à cet objectif, nous avons réalisé les expérimentations suivantes : dans un premier temps, seuls le module d'alignement est appliqué. Ensuite, le module d'optimisation a été activé pour produire l'ensemble d'alignements le plus similaire. Les figures suivantes montrent l'exécution des phases sans et avec optimisation.

- ***Résultat d'évaluation d'Alignement Avant Optimisation***

Après l'acquisition du résultat d'alignement, notre système fourni une évaluation de résultat par à rapport à l'alignement de référence.

Via le panneau " ***Evaluation Avant Optimisation***", notre application charge automatiquement le fichier de résultat, ensuite on charge un alignement de référence de notre base, et par un clique sur le bouton ***Evaluation*** on arrive au résultat d'évaluation montré par la figure 5.10.

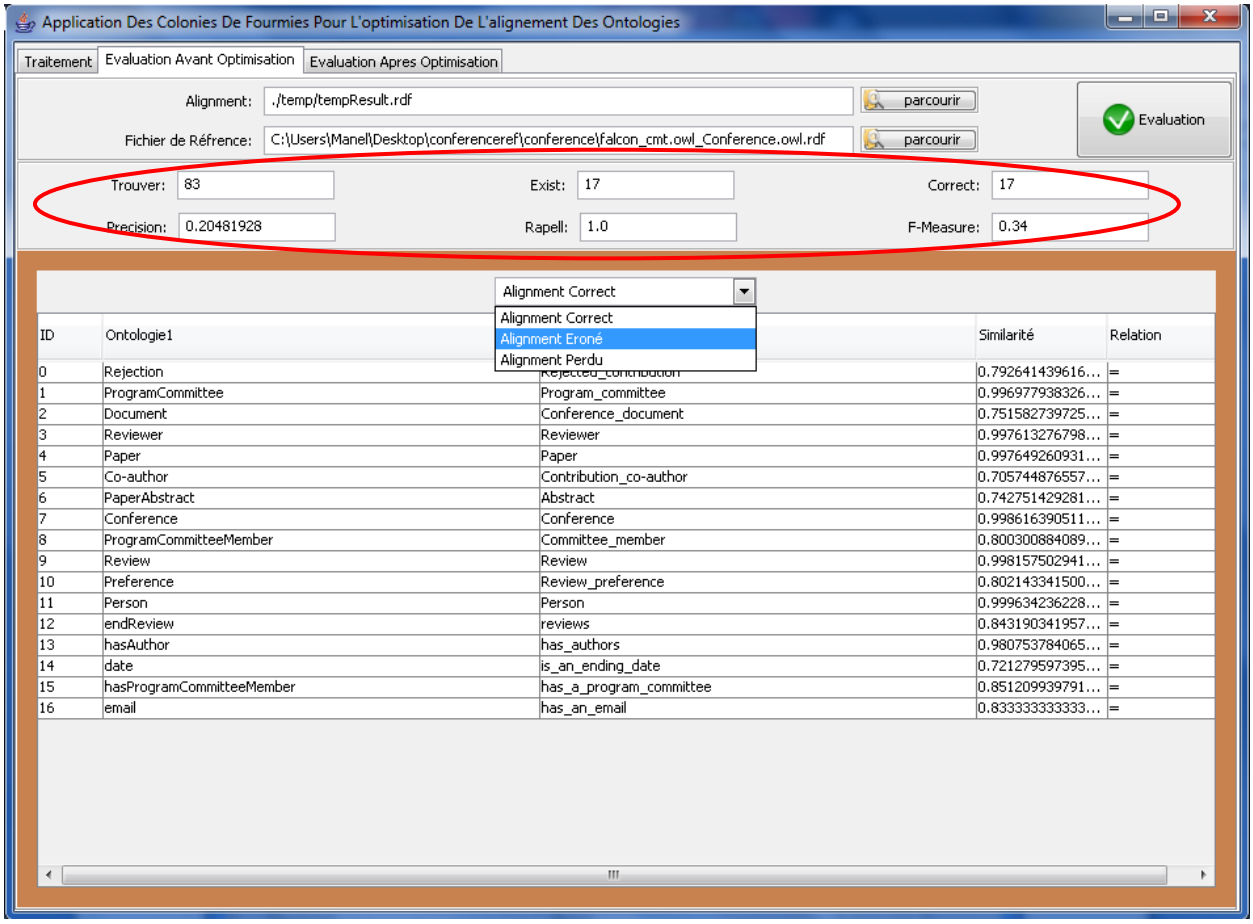


Figure 5.10 : *Résultat d'évaluation Avant Optimisation*

- ***Résultat d'évaluation après Optimisation***

À travers du panneau "***Evaluation après Optimisation***", on peut évaluée l'optimisation tout on clique sur le bouton ***Parcourir*** pour charger le fichier de référence qui correspond. Puis par une clique sur le bouton ***Evaluation*** on aperçoit le résultat monté dans la figure 5.11.

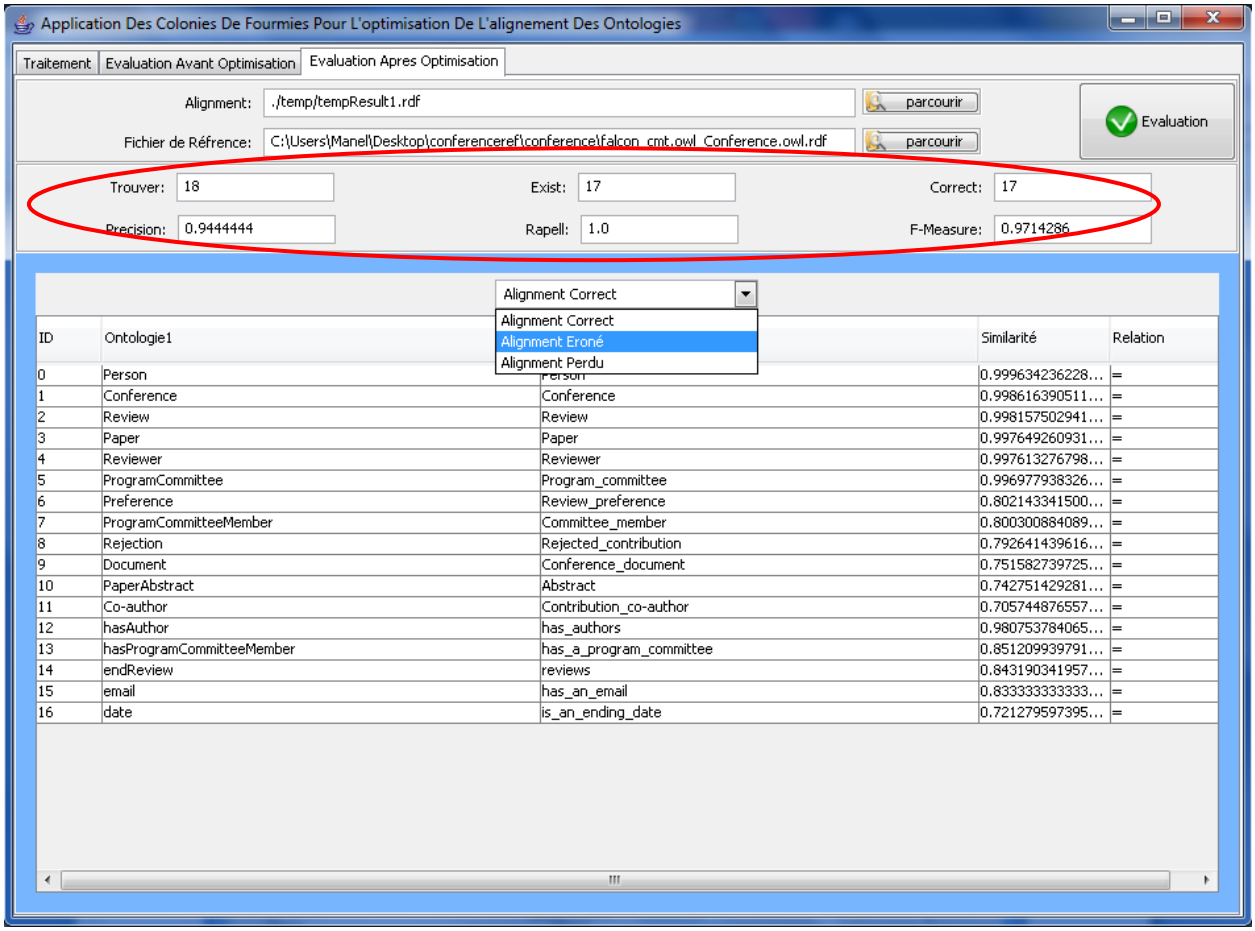


Figure5.11 *: Résultat d'évaluation après Optimisation*

5.8 Résultats expérimentaux

À travers des essais sur des ontologies de notre base de test, nous avons constaté que l'optimisation donne de bon résultat. Cela apparié clairement quand t'on applique l'étape d'alignement qui montre l'appariement multivoque des concepts, et qui sont éliminées par la suite lors de l'application de l'optimisation (Voir figure 5.5 et figure 5.6).

Pour mieux évaluée notre système, nous présentons dans cette section une comparaison entre les résultats obtenus avant optimisation et après optimisation selon les mesures d'évaluations : Rappel, Précision et F-mesure.

L'histogramme suivant récapitule les résultats obtenus selon ces mesures d'évaluation :

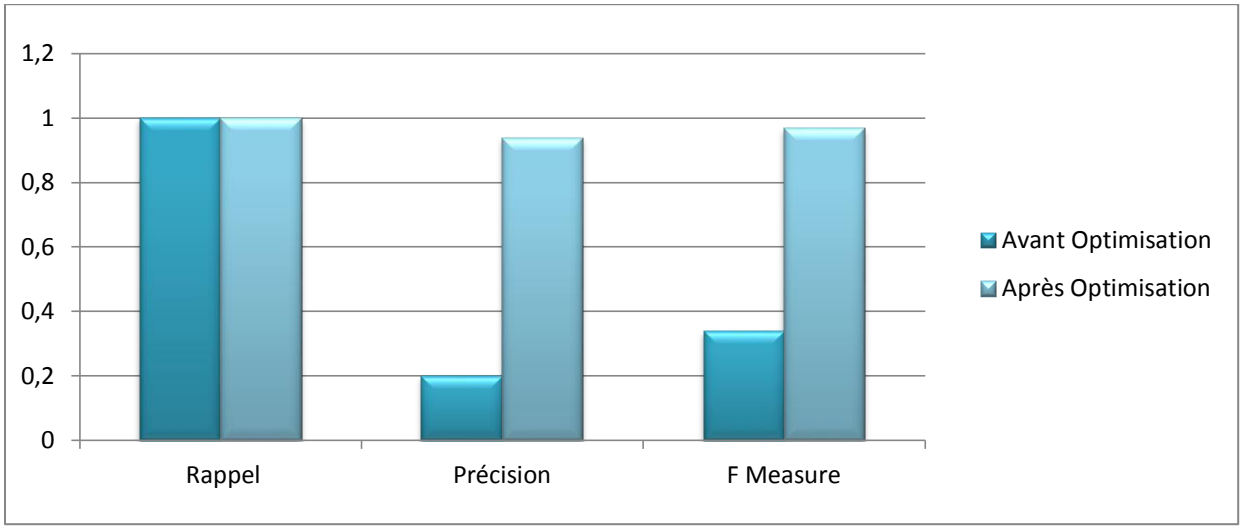


Figure 5.12 : *Evaluation de l'approche proposée*

D'après ces résultats, on constate très bien que l'algorithme d'optimisation donne des bon résultats, vu que :

- Le rappel =1.0 avant et après l'optimisation, ce qu'il assure que notre méthode couvre toutes les correspondances correctes.
- La précision= 0.20 avant optimisation et 0.94 après, cela montre que le taux du bruit dans le résultat de l'algorithme est réduit, qui implique que la qualité du résultat est imposante.
- La F-mesure = 0.34 avant optimisation et 0.97 après, qui justifie bien la garantie de l'exactitude de la qualité d'évaluation des deux mesures précédentes.

Conclusion

Dans ce chapitre, nous avons présenté en détail notre application, qui se base sur l'optimisation de l'alignement par l'application des colonies de fourmis.

D'après les résultats acquises, on conclu que notre algorithme d'optimisation détecte les concepts les plus similaires qui introduit une bonne qualité d'alignement à n'importe quelle complexité d'ontologies. Cette spécificité s'explique par le fait que notre système, fond sur le comportement simple des colonies de fourmis qui permet de mieux guidée à cette mise en correspondance.

Conclusion Générale

Conclusion Générale et Perspectives

Le travail mené dans ce mémoire se situe dans le domaine de l'ingénierie des connaissances et du web sémantique. Notre objectif a été de tirer profit des travaux menés notamment dans le domaine de l'interopérabilité sémantique des connaissances, dans le but d'aligner des ontologies et en particulier d'optimiser le résultat de cet alignement.

Le résultat de notre travail est une méthode d'alignement d'ontologie. Cette méthode exploite les méta-heuristiques pour extraire des correspondances entre des ontologies. Notre méthode est la seule approche appliquant, pour l'alignement des ontologies, l'algorithme d'optimisation de colonies de fourmis et la notion de recherche de sous-ensemble en utilisant l'appariement multivoque de graphes.

Son idée est de considérer des ontologies en OWL, comme des graphes étiquetés dont les nœuds sont les entités, les arcs sont les relations entre ces entités. Ces graphes reflètent la conceptualisation et la modélisation des ontologies, et donc, plus ils sont proches, plus les ontologies sont similaires. Ainsi, notre méthode cherche le sous-ensemble le plus similaire et le plus pertinent de ces deux graphes en utilisant un algorithme de colonies de fourmis et en prenant en compte la structure de voisinage, pour déduire des correspondances entre deux ontologies.

Les résultats obtenus par la méthode, sont très satisfaisantes, en se basant sur les métriques d'évaluation : Rappel, Précision et F-mesure.

Mais, malgré les performances prouvées, notre méthode à également quelques limites telles que :

- Pour les cas plus précis, où les ontologies à aligner sont plus similaires au niveau linguistique qu'au niveau terminologique et structurel, la méthode donne des résultats encourageants.

Perspectives

Plusieurs améliorations sont possibles sur nos propositions, permettant de les rendre plus pertinentes. Ces améliorations incluent :

- **Passage à l'échelle :** Les expérimentations menées jusqu'à présent ont porté sur des ontologies de taille relativement modeste. Il est important d'évaluer nos propositions sur des ontologies de tailles plus réalistes (plusieurs centaines de concepts).
- **Intégration d'un module de calcul de similarité linguistique :** en utilisant par exemple l'API WordNet.

Annexe

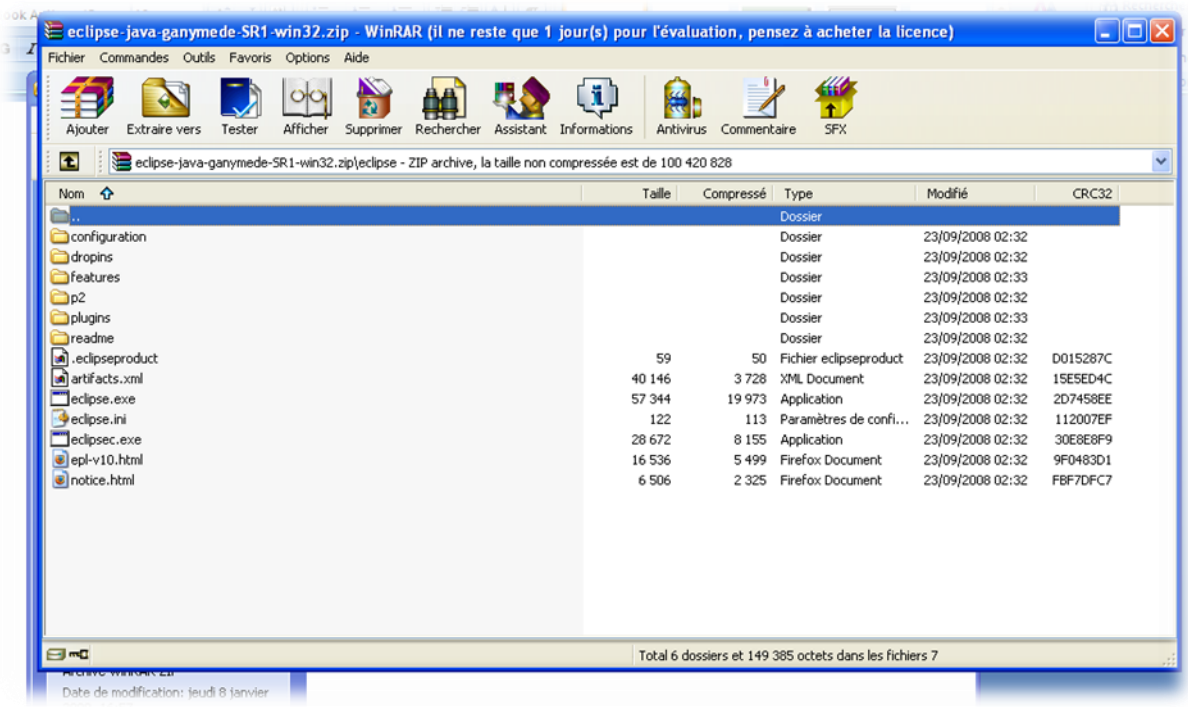
Annexe. Installation et configuration des Outils de développement

Vous trouverez dans cette partie touts les tutoriels nécessaire à l'installation, la configuration et la manipulation des logiciels et outils ayant servi au développement de notre application: l'environnement de développement intégré Eclipse, l'API sémantique Jena, l'éditeur d'ontologie Protégé et l'outil GraphViz.

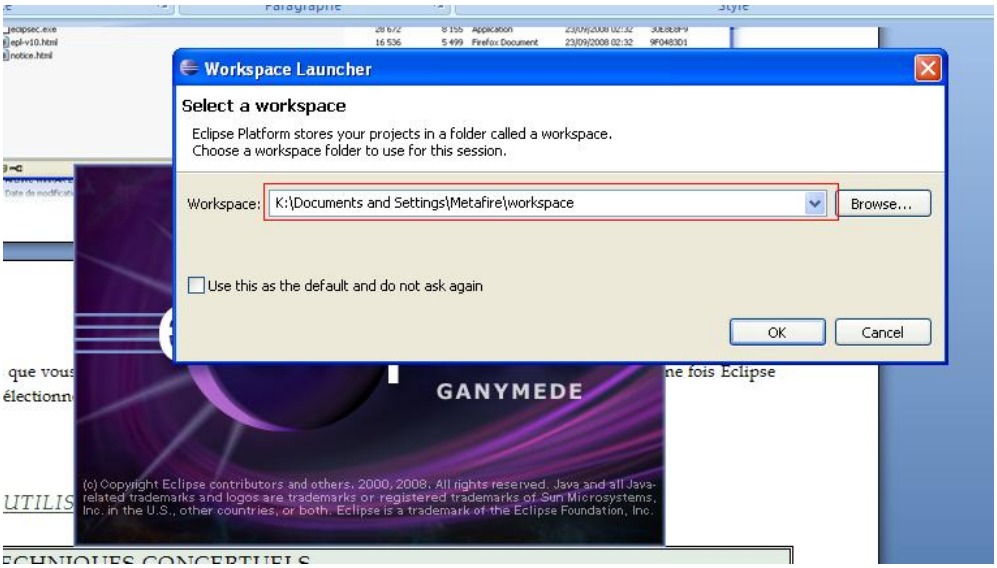
ECLIPSE

Vous trouverez les fichiers nécessaires à l'installation sur le site officiel de l'EDI : http://www.eclipse.org/downloads/

Sélectionner « Eclipse IDE for Java Developers » puis choisissez un miroir de téléchargement. Une archive est alors placé sur votre ordinateur. Décompressez cette archive à l'endroit de votre choix.



Dès lors que vous avez décompressez Eclipse, vous pourrez le lancer à partir d'Eclipse.exe. Une fois Eclipse lancé, sélectionné un espace de travail.



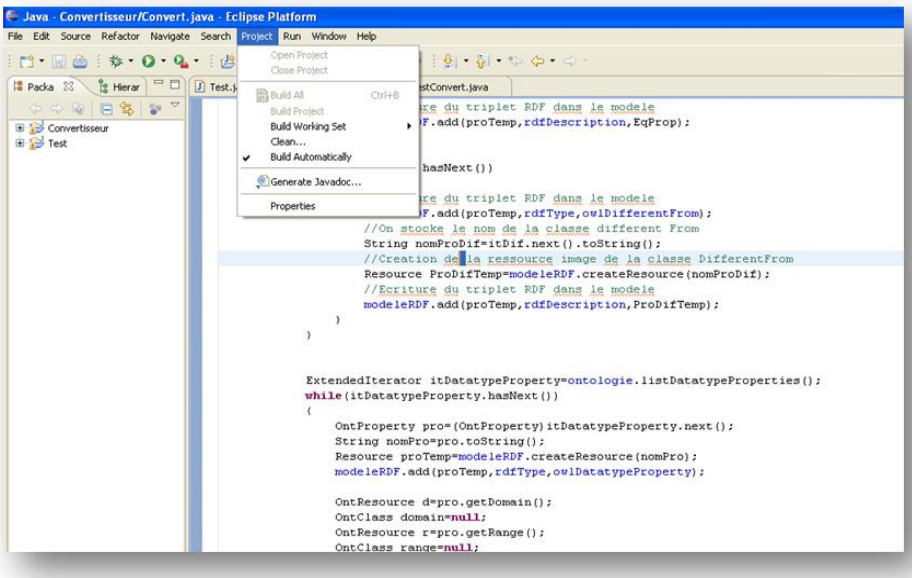
L'EDI Eclipse est alors près à l'emploi.

API JENA

L'API Jena se trouve à l'adresse suivante :

http://sourceforge.net/project/downloading.php?groupname=jena&filename=Jena-2.5.8.zip&use_mirror=heanet

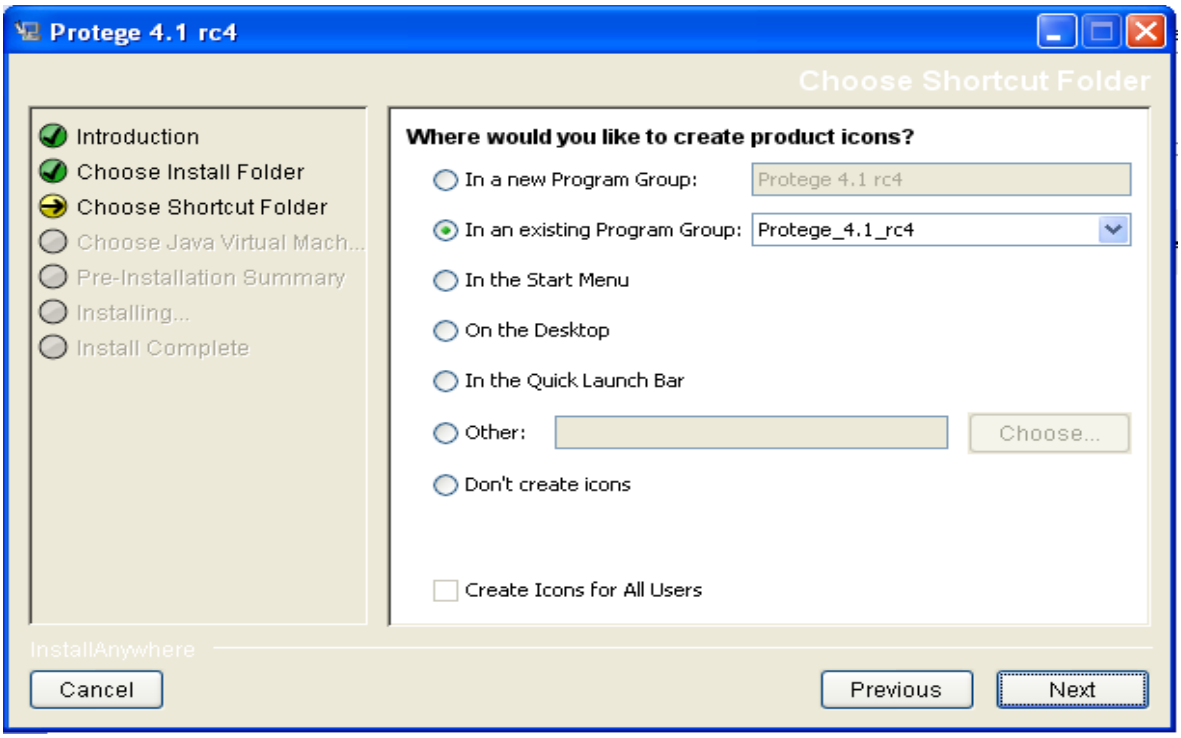
Décompressez ensuite le contenu du téléchargement à l'endroit de votre choix. Nous allons maintenant voir comment importer l'API Jena afin de l'utiliser dans Eclipse. Lancez tout d'abord Eclipse. Allez ensuite dans l'onglet « Projet » puis « Properties »



Dans l'onglet « Java Build Path » , rendez vous dans la section « Libraries » . Cliquez ensuite sur « Add External Jars ». Allez ensuite chercher chacun des fichiers .jar composant l'API téléchargé plus tôt. L'API est maintenant prête à l'emploi. N'oubliez pas de réaliser les opérations d'imports nécessaires dans vos programmes utilisant Jena.

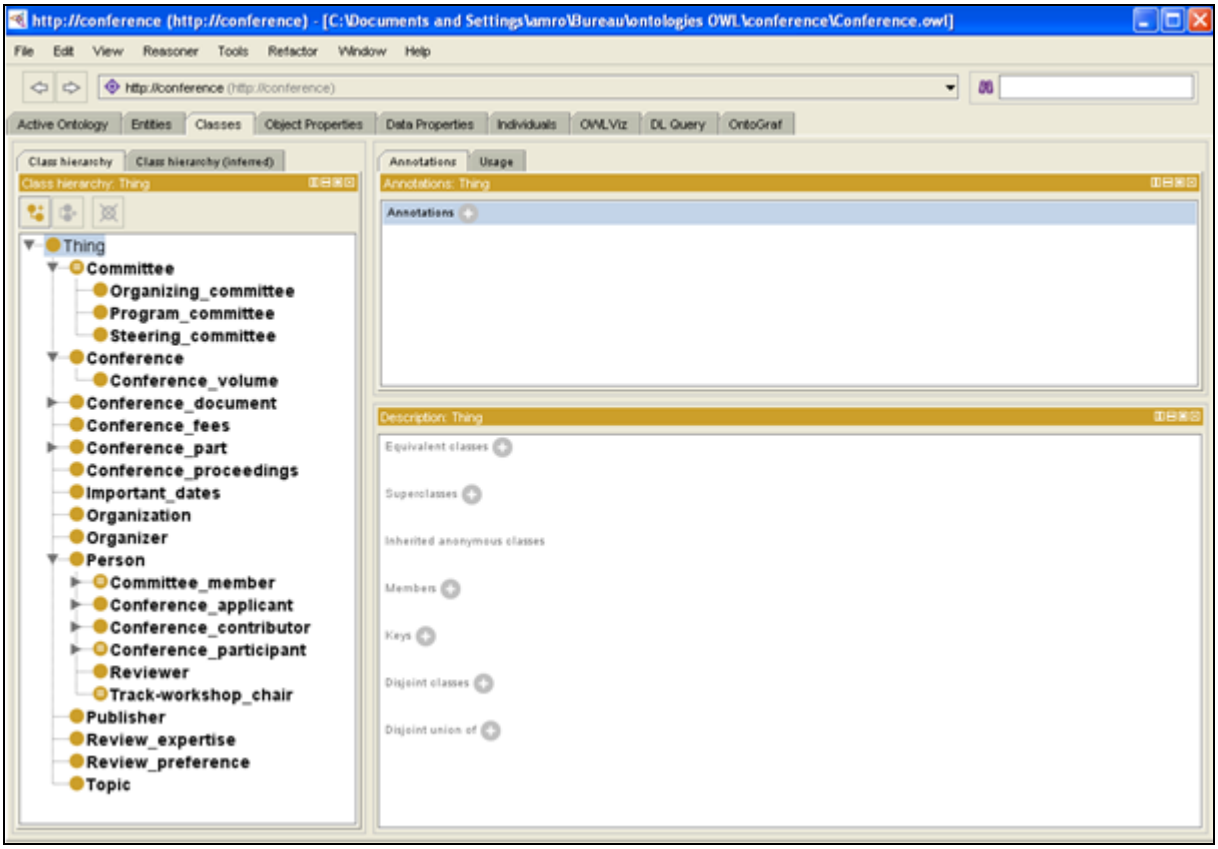
PROTEGE2000

Pour obtenir les fichiers d'installation, vous pouvez vous rendre sur le site de l'université de Stanford, l'auteur du logiciel : http://protege.stanford.edu/download/download.html



Lancez l'installation. Poursuivez jusqu'à ce que l'on vous demande de choisir la version de protégé.

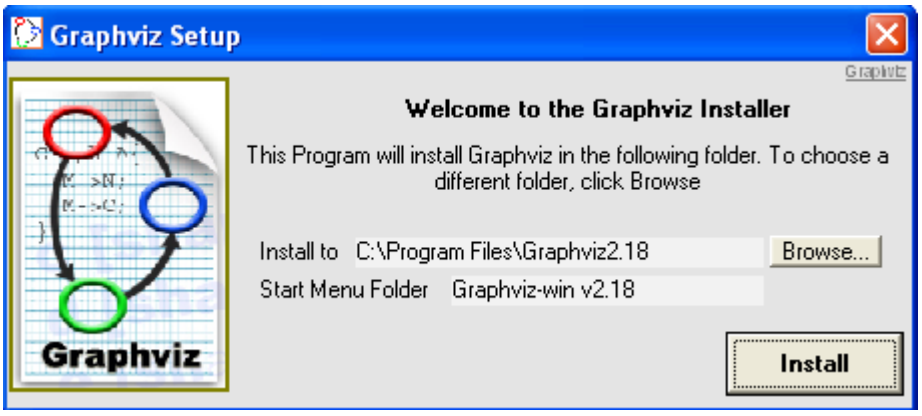
Choisissez le chemin d'installation puis votre machine virtuelle Java. L'installation se lance alors. Après un court instant, Protégé est installé sur votre machine.



GraphViz

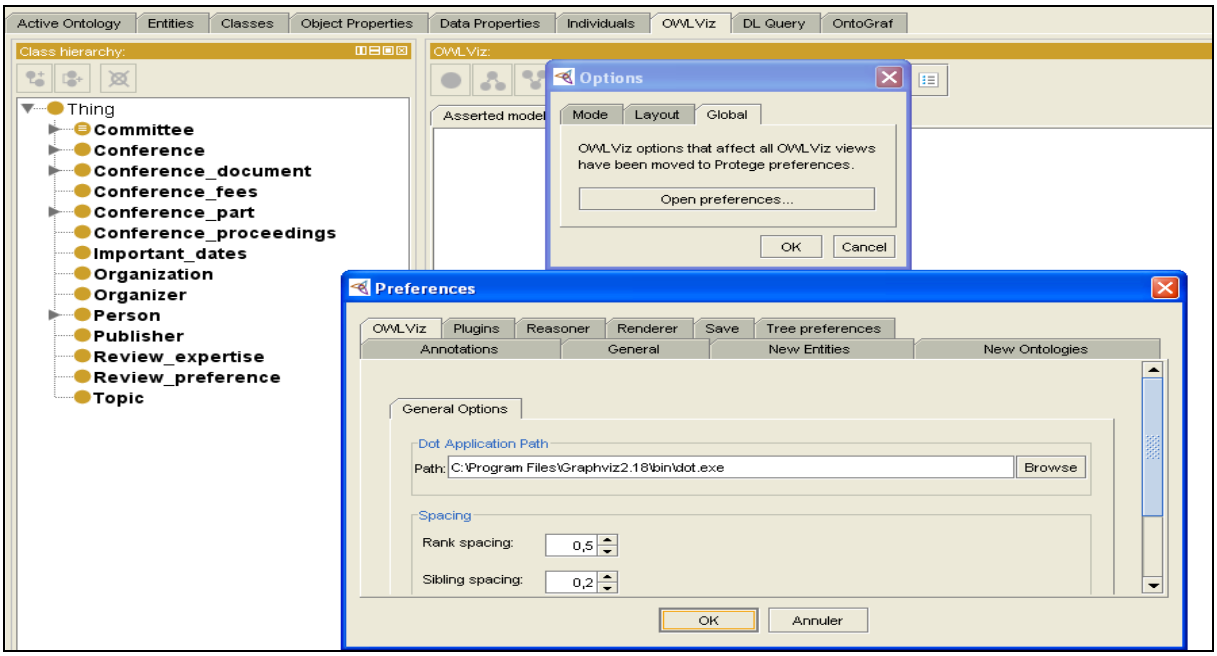
Pour obtenir les fichiers d'installation, vous pouvez vous rendre

sur le site: www.graphviz.org/pub/graphviz/stable/windows/

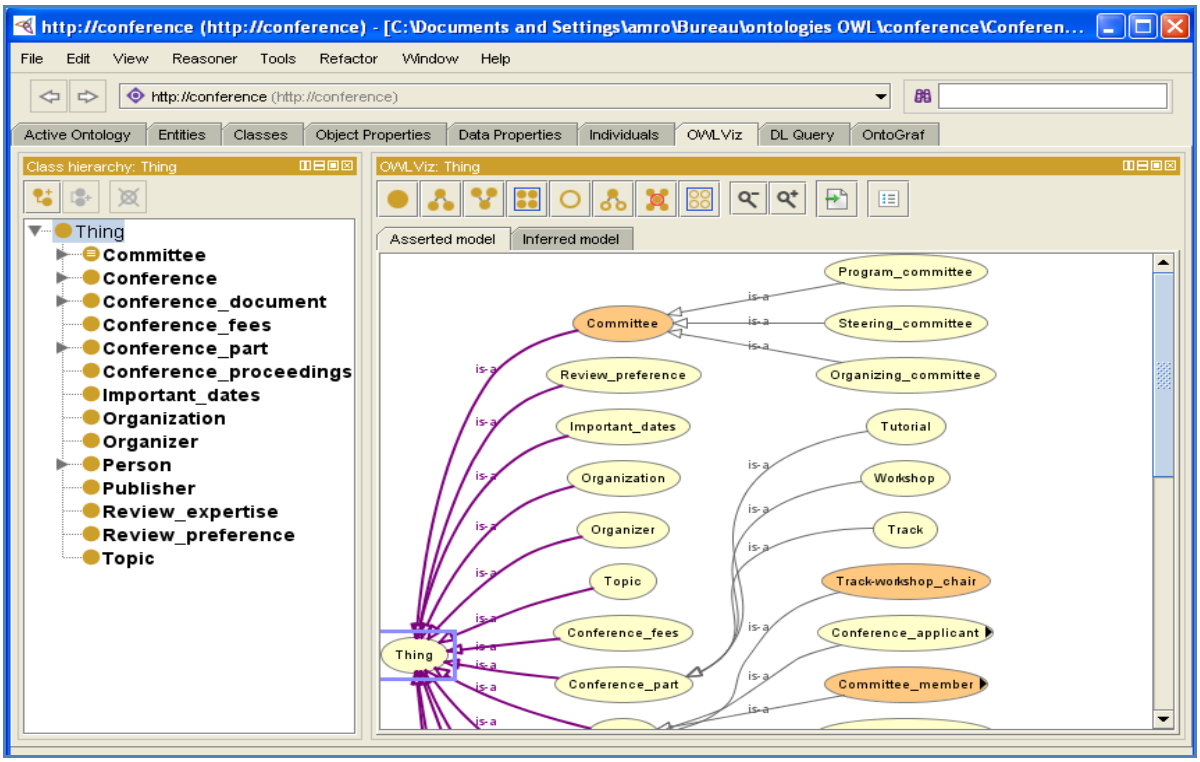


Après l'installation du Grapgviz, nous allons maintenant voir comment le configurer afin de l'utiliser avec Protégé. Lancez tout d'abord Protégé. Allez ensuite dans l'onglet « OWL Viz »

puis sur le bouton Option, ensuite dans l'onglet « Global », un clic sur « Open Preferences », et dans « Dot Application Path », on spécifie l'emplacement de l'application Dot.exe



Maintenant l'ontologie est présentée sous par Graphviz :



Bibliographie

Bibliographies

[Actes, 1999] Actes du Colloque. Terminologie et intelligence artificielle (actes du colloque de Nantes, 10-11 mai 1999) In : Rint, Réseau international de néologie et de terminologie, 1999.

[Baader et al, 1991] F. Baader, D. Mcguinness, D. Nardi et P. Patal-Schneider, "The Description Logic Handbook: Theory, Implementation and Applications '', Cambridge University Press, 1991.

[Bach et al, 2004] Bach T. L., Dien-Kuntz R., Gandon F., « On Ontology Matching Problems - for Building a Corporate Semantic Web in a Multi-Communities Organization », *Proceedings of ICEIS*, Porto, Portugal, 2004.

[Bichot, 2007] Charles-E.BICHOT, ELABORATION D'UNE NOUVELLE M´ETAHEURISTIQUE POUR LE PARTITIONNEMENT DE GRAPHE : LA M´ETHODE DE FUSION-FISSION. APPLICATION AU D´E COUPAGE DE L'ESPACE A´ERIEN, Thèse présentée pour obtenir le titre de DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE TOULOUSE, juillet 2007.

[Blazquez et al, 1998] Blazquez, M., Fernandez-lopez, M., Garcia-Pinar, J.M. & Gomez-Pérez, A. (1998) Building Ontologies at the Knowledge Level using the Ontology Design Environment. In Proceedings of the *Workshop on Knowledge Acquisition, Modelling and Management: KAW'98*, Banff, Canada.

[Champin, 2003] P.-A Champin, C.Solnon. "*Measuring the similarity of labeled graphs*", in 5th International Conference on Case-Based Reasoning (ICCBR 2003), volume Lecture Notes in Artificial Intelligence 2689-Springer – Verlag, 2003 .

[Colorni, 1992] A. COLORNI, M. DORIGO et V. MANIEZZO: Distributed Optimization by Ant Colonies, In Proceedings of the First European Conference on Artificial Life, 1992.

[David, 2007] Jérome David, «AROMA : une méthode pour la découverte d'alignements orientés entre ontologies à partir de règles d'associations », Thèse de doctorat, Université de Nantes, 2007.

[Doan et al, 2002] A Doan, J Madhavan, P Domingos, and A. Halevy, "Learning to Map between Ontologies on the Semantic Web", in the 11th International World Wide Web Conference (WWW'2002), Hawaii, USA, 2002.

[Doan et al, 2003] An-Hai Doan A.H, Domingos P. and Halevy A , "Learning to match the schemas of data sources: A multistrategy approach", Machine Learning, 2003.

[Dorne et Hao, 1998] R. DORNE, J.K. HAO, A new genetic local search algorithm for graph coloring. *Lecture Notes in Computer Science 1498 :* 745-754, Springer-Verlag, 1998.

[Euzenat et al, 2004] Euzenat J., Bach T.L., Barrasa J., Bouquet P., Bo J.D., Dieng-Kuntz R., Ehrig M., Hauswirth M., Jarrar M., Lara R., Maynard D., Napoli A., Stamou G., Stuckenschmidt H., Shvaiko P., Tessaris S., Acker S.V. and Zaihrayeu, "I. State of the art on ontology alignment, deliverable 2.2.3", IST Knowledge web NoE, Juin 2004.

[Euzenat et al, 2007] Euzenat J et al, "Heterogeneity in the semantic web, deliverable 2.2" , Knowledge Web, December 2007.

[Fernandez et al, 1997] M. FERNANDEZ, A. GOMEZ-PEREZ, N. JURISTO, "METHONTOLOGY: from ontological art towards ontological engineering, in Proceedings of the Spring Symposium Series on Ontological Engineering, 1997.

[Glover, 1989] F. GLOVER, Tabu search : part I. *ORSA J. on Computing* 1(3) : 190-206, 1989.

[Gruber, 1993] T.R.Gruber, A Translation Approach to Portable Ontology specifications, 1993.

[Guarino et Giaretta, 1995] N.GUARINO, P. GIARETTA, Ontologies and knowledge bases, towards a terminological clarification, 1995.

[He et al, 2003] He, B., Chang, K., & Han, J. Automatic Complex Schema Matching across Web Query Interfaces: A Correlation Mining Approach, Department of Computer Science. 2003.

[Kalfoglou et Schorlemmer 2003] Kalfoglou Y., Schorlemmer M., « IF-Map : an ontology mapping method based on information flow theory », Journal of data semantics, 2003.

[Klein, 2001] Klein M., "Combining and relating ontologies: an analysis of problems and solutions", in Proc. IJCAI Workshop on Ontologies and Information Sharing, Seattle (WA US), 2001.

[Maynard & Ananiadou, 1999] Maynard D.G. and Ananiadou S., "Term extraction using a similaritybased approach", in Recent Advances in Computational Terminology. John Benjamins, 1999.

[Mellal, 2007] Naçima MELLAL, Réalisation de l'interopérabilité sémantique des systèmes, basée sur les ontologies et les flux d'information, Université de Savoie, thèse pour obtenir le grade de docteur, 19 décembre 2007.

[Mihoubi, 2000] H. Mihoubi : « Une approche déclarative de traduction d'ontologies », Thèse doctorat, Grenoble : Université Stendhal- Grenoble 3, 2000.

[Miller, 1995] Miller G. A., « WordNet: a Lexical Database for English », Communications of the ACM, November, 1995.

[Neches et al, 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, et W.R. Swartout:

[Noy et al, 1999] F. Noy, N. and Musen, M.A, SMART: Automated Support for Ontology Merging and Alignment. In: *Proceedings of the Twelfth Banff Workshop on Knowledge Acquisition, Modeling, and Management*, Banff, Alberta, 1999.

[Noy et al, 2000] F. Noy, N. Fergerson, R.W. and Musen, M.A, The knowledge model of Protégé-2000: combining interoperability and flexibility. Stanford Medical Informatics Technical Report, Stanford University, 2000

[Noy et Musen, 2001] Noy N. and Musen M. Anchor, "PROMPT: Using non-local context for semantic matching", in Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US), 2001

[OntoEdit, 2002] OntoEdit: Collaborative Ontology Development for the Semantic Web, International Semantic Web Conference (ISWC02), Sardinia'Italy', 2001.

[Rahm et Berstein, 2001] Rahm E., and Bernstein P., "A survey of approaches to automatic schema matching" , VLDB Journal, 2001.

[Schank et Abelson, 1988] *Scripts, Plans, Goals and Understanding*, Kaufmann, San Mateo, CA, 1988.

[Sean & al, 2007] Sean M. F , Natalya F. Noy, Storey M.A, "*Ontology mapping - a user survey"*, Workshop Ontology Matching, in the 6th International Semantic Web Conference, Busan, Korea, November 2007.

[Shvaiko et Euzenat, 2005] Shvaiko P., and Euzenat J., "Tutorial on Schema and Ontology Matching", ESWC (European Semantic Web Conference), May 2005.

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

[Solnon, 2005] Christine Solnon, Contributions à la résolution pratique de problèmes combinatoires "des fourmis et des graphes", Mémoire pour l'obtention de l'Habilitation à Diriger des Recherches, l'Université Claude Bernard Lyon 1, 2005.

[Uschold et Gruninger, 1996] Mike USCHOLD et Michael GRÜNINGER, Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 1996.

[Xavier Lacot, 2005] introduction à OWL, *un langage XML d'ontologies Web*,«Enabling Technology for Knowledge Sharing », Juin 2005

[Xu et Embley, 2003] Xu, L., & Embley, D. Using Domain Ontologies to Discover Direct and Indirect Matches for Schema Elements. In: Second International Semantic Web Conference.2003

[Zghal et Ben Yahia, 2007] Sami Zghal, Karim Kamoun, Sadok Ben Yahia, E. Nguifo, Ben Yahya Slimani, EDOLA :" Une nouvelle method d'alignement d'ontologies OWL-Lite" , Faculté de Sciences de Tunis, 2007.