Université Sidi Mohamed Ben Abdellah

Faculté Des Sciences Et Techniques Fès

Département d'Informatique

FST FES

Projet de Fin d'Etudes

Master Sciences et Techniques

Systèmes Intelligents & Réseaux

Alignement d'ontologies du web sémantique

LSIA

Lieu de stage : Laboratoire SIA – FST de Fès

Réalisé par :

- Yassine namir
- Achraf ragui

Soutenu le

16/06/2016

Encadré par :

Pr. Ilham Chaker

Devant le jury composé de :

Pr. Rachid Ben Abbou

Pr. Arsalane Zarghili

Pr. Azeddine Zahi

Pr. Ilham Chaker

Année Universitaire 2015-2016

Remerciements :

Nous présentons nos respects et nos remerciements aux membres du jury qui nous ont fait l'honneur d'avoir accepté d'évaluer ce travail ainsi que notre encadrante Pr. Ilham Chaker.

Nous remercions vivement notre encadrante, Pr. Ilham Chaker, d'avoir accepté de diriger ce travail et pour son aide précieuse pour le temps qu'elle a pris pour échanger et pour nous corriger. En plus de remplir brillamment sa fonction de pourvoyeuse d'idées, et de relectrice rigoureuse.

Nous tenons à remercier également nos familles pour leurs soutiens et encouragements, ainsi que toute autre personne qui a contribué de près ou de loin au bon déroulement de ce projet.

Merci,

Table de Matières

Liste des sigles et des abréviations

1. API : Application Programming Interface
2. ASMOV: Automated Semantic Mapping of Ontologies with Validation
3. DAML : Darpa Agent Markup Language
4. DOE : Differential Ontologie Editor
5. F-logic : frame logic
6. FMA: Foundational Model of Anatomy
7. GA : Graphe d'Association
8. I^3CON: The Information Interpretation and Integration Conference
9. IA : Intelligence Artificiel
10. IC: Ingénierie de connaissance
11. IDF: Inverse Document Frequency
12. KIF: Knowledge Interchange Format
13. MeSH: Medical Subject Heading
14. MPV: Multi-point de vues
15. NCI : National Cancer Institute
16. NOM : Naive Ontology Matching
17. O-Graphe : Ontology Graphe
18. OIL : Ontology Interchange Language
19. OLA : Owl Lite Alignment
20. OntoEdit : Ontology Editor
21. OWL: Ontology Web Language
22. PWN : Princeton WordNet
23. QOM : Quick Ontology Matching
24. ROMIE: Resource based Ontology Mapping within and Interactive and Extensible environment
25. SAT : la satisfiabilité propositionnelle
26. SBC : Système de base de Connaissances
27. SE : Systèmes experts
28. SNOMED CT: Systematized Nomenclature Of Medicine Clinical Terms
29. SODA: Strucural Ontlology OWL-DL Alignement
30. SPARQL: Simple Protocol and RDF (Resource Description Framework) Query Language
31. TF : Term Frequency
32. UMLS : Unified Medical Language System
33. URN: Uniform Ressource Name
34. W3C: World Wide Web Consortium

Liste de Figures

Liste de Tableaux

Introduction générale

Aujourd'hui, des milliards de pages web sont recensés sur Internet. Par contre, la majorité de ces pages web ne sont compréhensibles que par l'être humain, par conséquent, les machines ou bien les agents logiciels ne peuvent effectuer que des recherches syntaxiques dans le contenu de ces pages. Ils sont incapables de comprendre, d'interpréter ou d'analyser les informations contenues dans ces pages Web.

Pour faire face à ces insuffisances du Web classique, Tim Berners-Lee [2] a proposé une nouvelle vision du Web, qui se base sur ce qu'on appelle « Ontologies ». Ces dernières permettent de décrire la structure et la sémantique de données contenues dans un document. Elles assurent aussi l'organisation des informations sous forme d'une taxonomie de concepts et de relations entre ces derniers, d'autant plus que ça, les machines et les agents logiciels peuvent interpréter leurs sémantiques.

Cependant, il n'existe pas d'ontologie universelle partagée, adoptée par tous les utilisateurs d'un domaine donné. Les problématiques et les tentatives d'amélioration de l'interopérabilité du système comptent donc sur la réconciliation des différentes ontologies utilisées dans un domaine par les différents systèmes. Cette réconciliation consiste à trouver les liens de correspondances entre ces ontologies, on parle donc de l'**alignement d'ontologies**.

C'est dans ce cadre que s'introduit ce projet de fin d'études, intitulé « Alignement d'Ontologies du Web Sémantique ». En effet, il s'agit tout d'abord de faire une étude de l'état de l'art sur le sujet de l'alignement des ontologies. Ensuite, étudier en détail l'algorithme d'alignement ASCO3. Ce dernier a été implémenté pour pouvoir l'évaluer.

Dans ce rapport, on va dans un premier lieu définir c'est quoi le Web Sémantique, puis donner une définition détaillée de la notion d'ontologie, ses composants, ainsi que son rôle indispensable dans le Web Sémantique, etc. Ensuite, dans le deuxième chapitre va détailler le concept d'alignement d'ontologies, puis on va présenter des travaux existants reliés à ce concept.

Puis, dans le troisième chapitre, on va présenter un algorithme permettant de faire la correspondance entre des ontologies représentées en OWL. L'algorithme proposé, nommé ASCO3, compare la similarité entre deux graphes représentant les structures de l'ontologie. Il applique l'algorithme de recherche du sous-graphe commun maximal de deux graphes, qui se base sur la recherche de la clique maximale du graphe associé (une clique est un sous-graphe complet dont les sommets sont tous connectés deux à deux).

Enfin, une évaluation de l'algorithme d'alignement d'ontologies proposé, ASCO3, est présentée dans le chapitre IV. L'implémentation de l'algorithme a été effectuée en Java, et les évaluations ont été faites sur des ontologies qu'on va détailler dans le même chapitre.

Chapitre I : Du Web Classique au Web Sémantique

Introduction :

Le défi majeur actuel pour le Web et en particulier pour les organisations hétérogènes est de tendre vers « le développement des ontologies MPV » où il est nécessaire d'atteindre un certain niveau de consensus pour que les connaissances représentées par ces ontologies puissent être réutilisées, partagées et échangées efficacement par différentes communautés.

Dans le contexte du Web Sémantique, une ontologie correspond à une modélisation du monde dans un domaine de discours. Un tel consensus dans une ontologie requiert idéalement que les membres de groupes d'utilisateurs concernés s'engagent à utiliser cette ontologie, et reposent ainsi sur une modélisation commune du monde réel [1]. Cependant, si l'on passe d'un seul groupe ou d'une petite communauté des utilisateurs à plusieurs groupes ou à une plus grande communauté, ou bien si l'on considère une organisation telle qu'une entreprise, voire une entreprise virtuelle constituée de plusieurs organisations en collaboration, la maintenance d'un tel consensus et d'une modélisation commune du monde devient très difficile. On a besoin de pouvoir exprimer le consensus dans un point de vue particulier pour chaque groupe ou chaque communauté, et de décomposer le monde selon les différents points de vue de différents groupes ou différentes communautés [1].

Dans ce chapitre, nous tentons de revenir sur une petite définition du Web Sémantique. Puis, la définition formelle d'une ontologie, donc la notion d'ontologie sera présentée à travers les besoins auxquels elle répond dans notre contexte, tout en revenant plus particulièrement sur les différents types d'ontologies qui nous paraissent fondamentaux (pour introduire à la définition des ontologies MPV), quelques langages de représentation, et quelques approches de construction de ces ontologies ont été développés récemment. À la fin, le chapitre se termine par une conclusion.

I. Web Sémantique :

"...the semantic web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation. »

- *Tim Berners-Lee, James Hendler And Ora Lassila*

D'après Tim Berners-Lee, James Hendler et Ora Lassila, « *le Web sémantique est une extension du Web actuel avec une signification bien définie des données et des ressources. Il permet aux ordinateurs et aux utilisateurs de coopérer et d'échanger l'information facilement* ».

Le Web nous permet aujourd'hui d'accéder à des documents et à des services par l'intermédiaire d'Internet. L'interface d'accès aux services est représentée par des pages Web écrites dans du langage naturel, qui doit être compris par l'être humain. Le Web sémantique est une extension du Web actuel dans lequel l'information est munie d'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 [2]. L'utilité du Web sémantique peut être vue au travers des exemples suivants : supposons que vous cherchiez à comparer les prix des canapés qui se fabriquent dans votre région (c.-à-d. le même code postal que le vôtre), ou que vous cherchiez 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. Vous avez toujours besoin d'une personne capable de discerner et de vous donner la signification de ces réponses et de leur importance par rapport à vos besoins.

Le but du Web sémantique est de rendre explicite le contenu sémantique des ressources dans le Web (documents, pages web, services…). Les ordinateurs et les agents logiciels pourraient donc « comprendre » les informations contenues dans ces ressources et aider les utilisateurs à exécuter et compléter leurs tâches, leurs requêtes de façon plus automatique et plus efficace. Les machines maintenant pourraient « savoir » qu'une voiture est une sorte de véhicule, que c'est la même chose qu'une bagnole, que l'âge d'une personne est un nombre entier positif et ne peut pas être plus grand que 150 ans… Elles pourraient donc raisonner sur des connaissances déjà existantes dans le Web pour fournir des informations plus précises, des services plus évolués qui s'adaptent mieux aux besoins des utilisateurs.

Cependant, le Web sémantique n'est pas construit à partir de zéro. Le Web actuel est déjà un trésor immense des connaissances et des informations dans l'histoire humaine, où les informations sont sauvegardées, organisées et représentées de façon passive, non structurée, arbitraire et ad hoc. Le Web sémantique hérite de ce trésor des connaissances, mais il permet aussi à des machines d'accéder à ce trésor et de l'exploiter. Le Web sémantique est donc une extension du Web actuel.

La Figure 1 montre en image les extensions du Web sémantique par rapport au Web actuel. Nous pouvons trouver, en plus des ressources dans le Web actuel, leur sémantique rendue explicite dans le Web sémantique, souvent sous forme d'annotations sémantiques de ces ressources. Les pages web codées en HTML (HyperText Markup Language, langage pour mettre en page des documents, des textes, des images, des vidéos… pour les pages web), transférées sur le Web par le protocole de communication HTTP (HyperText Transfer Protocol), référencées (la localisation) par des URL (Uniform Resource Locator) dans le Web actuel seront identifiées par des URI (Uniform Resource Identifier) dans le Web sémantique. En plus, la sémantique des ressources sera encodée selon le modèle de métadonnées standardisé RDF (Resource Description Framework), sérialisées en format XML (Extensible Markup Language) et basées sur des ontologies en format OWL (Web Ontology Language). Dans le Web sémantique, les ressources seront ainsi accessibles et compréhensibles par des logiciels, qui pourront effectuer des raisonnements, des recherches plus « intelligentes » que les recherches par des mots clés comme aujourd'hui.

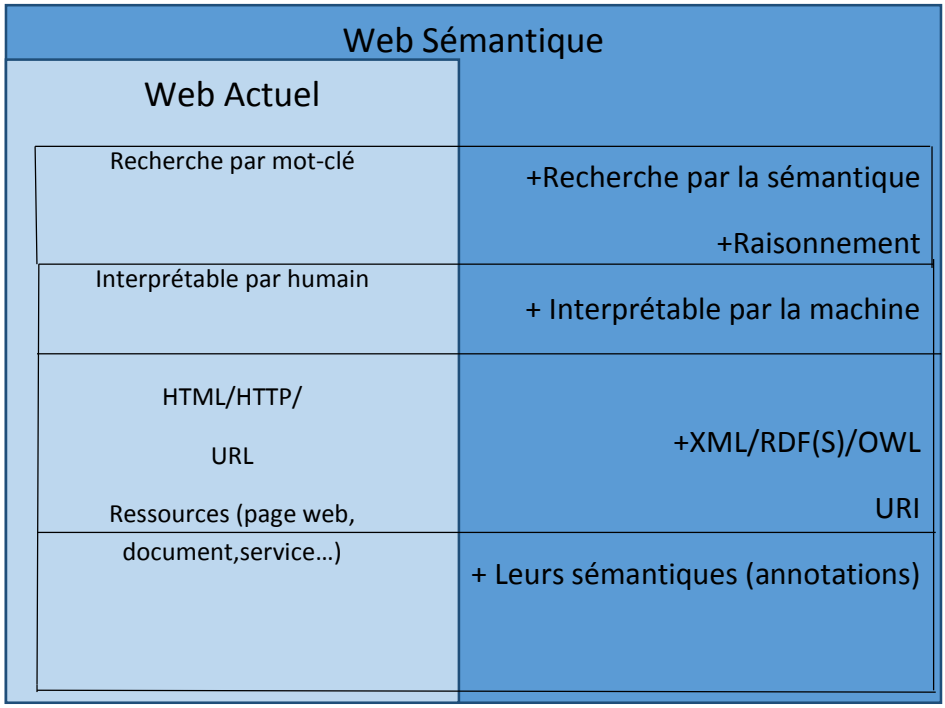


Figure 1: Le Web actuel et son extension, Le Web Sémantique

Le schéma de la Figure 2 montre les couches du standard du Web sémantique. Elle contient les éléments suivants :

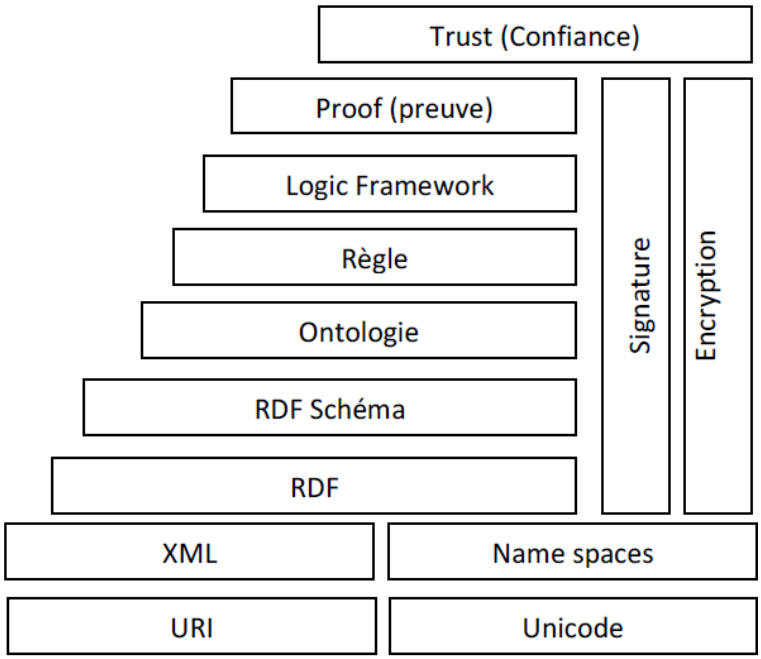


Figure 2: Les couches du standard du Web Sémantique

Dans ce qui suit, nous présenterons plus en détail les différents niveaux de cette architecture.

La couche URI (Uniform Resource Identifiers), Unicode : L'URI (Uniform Resource Identifier) est justement un protocole simple et extensible pour identifier, d'une manière unique et uniforme, toute ressource sur le web. Il s'agit d'un aspect central de l'infrastructure, c'est pour cette raison que cet élément se trouve à la base de l'architecture en couches proposée.

La spécification de la syntaxe et de la sémantique de l'URI dérive des concepts introduits par l'initiative de globalisation de l'information dans le World Wide Web, qui date des années 1990. Elle faisait alors l'objet du protocole RFC1630. Son utilisation est décrite dans l'ouvrage « Universal Resource Identifiers in WWW » et vise à réunir plusieurs recommandations concernant la localisation d'une ressource (URL) et son identification (URN). C'est pourquoi on dit qu'il est le pilier du web syntaxique avant d'être celui du web sémantique. La notion d'URI a évidemment évolué depuis, pour être décrite par un nouveau standard du web qu'est le RFC2396.

URI est en fait un acronyme de « Uniform Ressource Identifier », qui veut dire en français Identificateur uniforme de ressources. D'une façon générale, le terme identificateur désigne une clé capable de référencer un objet ayant une identité. Dans le cas du web sémantique, l'URI est une séquence de caractères avec une syntaxe restreinte, qui permet d'identifier toute ressource utilisée dans le cadre d'une application web sémantique. On entend par ressource n'importe quel objet ayant une identité, telle qu'un document électronique, une page HTML, un fichier, une image, une vidéo, un service, une couleur, etc. Une ressource n'est pas nécessairement un objet faisant partie du réseau, il peut s'agir d'un internaute, d'un livre, d'une corporation, etc. La notion d'uniformité, ou universalité, quant à elle, est d'autant plus importante que riche d'avantages. Elle permet d'abord à différents types d'identificateurs de ressources d'être utilisés dans le même contexte, même si le mécanisme qui leur permet d'y accéder est différent. Ensuite, elle assure une interprétation sémantique uniforme des conventions syntaxiques communes pour les différents identificateurs. Par ailleurs, elle permet d'introduire de nouveaux types d'identificateurs de ressources sans interférer avec ceux existants. Enfin, cette uniformité permet la réutilisation de ces identificateurs dans différents contextes. Nous voyons là au moins 4 avantages de l'uniformisation des identificateurs des ressources.

Par ces propos, nous sous-entendons qu'il existe plusieurs types d'identificateurs de ressources. Un URI peut être classé, en effet, en 3 catégories, selon qu'il soit destiné à la localisation, au nommage ou au deux. Le terme URL (Uniform Resource Locator) désigne un sous-ensemble d'URI qui identifie les ressources via une représentation de leur mécanisme d'accès, plutôt que par le nom ou autre attribut de cette dernière, comme il en est le cas pour l'URN (Uniform Resource Name). L'URL et l'URN sont donc des cas particuliers d'URI.

La couche XML, Name Spaces : À ce niveau d'architecture, nous ne sommes toujours pas au point d'affecter une sémantique à l'information, c'est-à-dire de la décrire et lui donner un sens. Il s'agit seulement d'une couche syntaxique, de bas niveaux, qui permet de structurer

les données et organiser selon un format de message standard, et ce, grâce au langage de balisage extensible XML (*eXtensible Markup Language*). En d'autres termes, XML permet d'indiquer l'organisation logique de l'information d'un document, mais, a priori, ne permet pas d'en décrire le contenu.

Avant de commencer à organiser les informations dans un document XML, il est impératif de définir la structure de ce dernier, afin de permettre notamment de vérifier sa validité. DTD (Document Type Definition) et XML-S (XML Schema) sont des langages de description de format de document XML. Cependant, on peut noter certaines différences entre eux. XML-S permet par exemple de définir des domaines de validité pour la valeur d'un champ, alors que cela n'est pas possible dans une DTD; en revanche, il ne permet pas de définir des entités. Par ailleurs, il est à noter qu'un fichier XML-S est lui-même un fichier XML. La norme XML n'impose pas l'utilisation d'une DTD pour un document XML, mais elle impose par contre le respect exact des règles de base de la norme XML, c'est pourquoi on fait la différence entre un document valide, pour un fichier XML qui comporte une DTD, et un document bien formé, pour un fichier XML qui respecte juste les règles de base du XML.

Concrètement, un fichier XML peut faire appel à plus d'un document DTD ou XML-S. Sa structure peut donc être organisée selon plusieurs grammaires. Afin d'éviter les conflits de noms, dus au fait que ces documents sont en général développés indépendamment les uns des autres, le W3C a mis en place un nouveau standard, baptisé « Namespace », qui veut dire espace de nommage. Selon la définition de ce consortium, les espaces de nommage fournissent un moyen simple pour qualifier les noms des éléments et des attributs dans le langage XML, en les associant avec des espaces de noms identifiés par URI.

Jusqu'à ce niveau de l'architecture, le problème de l'interprétation de la sémantique de l'information par la machine n'est toujours pas résolu. Le XML a su donner une nouvelle dimension aux informations en les structurant à la guise du développeur, mais ne permet pas d'en décrire le contenu. Cela parait plutôt logique, puisque la structuration et la description sont deux processus différents.

Dans ce qui suit, nous présentons les standards de représentation des connaissances qui enrichissent les informations apportées par XML, à savoir RDF/RDFS et OWL.

RDF (Resource Description Framework) est un modèle conceptuel, normalisé par le W3C, servant à encadrer la description de ressources, et ce, d'une façon simple et non ambiguë [74]. Ses applications visent initialement le web sémantique, mais elles peuvent s'étendre plus largement à l'ingénierie des connaissances. Ce modèle est associé à une syntaxe dont le but est de permettre à une communauté d'utilisateurs de partager les mêmes métadonnées pour des ressources partagées.

La construction de base en RDF consiste en un triplet d'éléments (Ressource, Propriété, Valeur), qu'on appelle déclaration RDF. Par analogie, un triplet RDF est similaire à la déclaration <sujet — prédicat — objet> :

- **Ressource (Sujet)** : Cela peut être n'importe quel objet référencé par un URI, qu'il concerne le web (Page HTML, document PDF, fichier multimédia…), ou non (Personne, Région, Etc.).
- **Propriété (prédicat)** : Critère, caractéristique, attribut ou relation qui peut décrire la ressource (titre, couleur, taille, auteur, etc.). Une propriété ses valeurs permises et ses relations avec les autres propriétés.
- **Valeur (objet)** : C'est la valeur qui sera affectée à la propriété de la ressource. Cette affectation peut être soumise à certaines restrictions.

Chaque élément de ce triplet peut être un URI, un littéral ou une variable.

RDFS est une extension de RDF. Un schéma RDF permet de décrire un vocabulaire et une sémantique des types de propriétés utilisées par une communauté d'utilisateurs ; est un vocabulaire de base pour décrire les déclarations RDF, au même titre que le XML-S pour le langage XML. Il ajoute à RDF la possibilité de définir des hiérarchies de classes et de définir les genres et les propriétés des ressources, d'assigner des contraintes spécifiques sur la nature des documents et de fournir des informations sur l'interprétation des déclarations RDF. Les schémas RDF permettent donc de garantir qu'un document RDF est sémantiquement consistant.

La couche ontologie : Le terme ontologie est emprunté de la philosophie où il fait référence à la science qui « étudie l'être en tant qu'être ». Avec l'émergence de l'ingénierie des connaissances et du web sémantique, ce terme a pris une tout autre tournure pour désigner la problématique de représentation et de manipulation des connaissances dans un système informatique. Elles sont définies comme « une spécification explicite d'une conceptualisation » [22].

L'ontologie n'est en fin de compte qu'une modélisation du monde réel en concept et relation entre ces concepts. Les principales composantes qu'on peut distinguer sont donc les suivantes [4] :

- **Concept** : C'est la représentation abstraite des éléments du domaine. On peut également les appeler termes ou classes. Ces concepts peuvent être classés selon différents critères dans la taxonomie (niveau d'abstraction, atomicité, etc.)[7].
- **Relations** : Elles expriment les associations entre les différents concepts définis dans la taxonomie. Les différents types de relations qui peuvent exister sont : « Spécialisation/Généralisation », « Agrégation ou Composition », « associé à », « composé de », Etc.
- **Fonctions** : Il s'agit des relations particulières ou un élément est défini par les n-1 autres éléments.
- **Axiomes :** Constituent des assertions considérées toujours comme vraies.
- **Instances** : Ce sont des exemples particuliers de concepts.

Nous reviendrons plus en détail sur la définition formelle d'une ontologie, dans la partie suivante.

La couche logique : La couche logique repose sur les langages ontologiques dans l'architecture recommandée par le W3C. En général, nous utilisons la couche logique pour exprimer les règles d'inférences.

Une large variété de logiques a été conçue jusqu'à présent. Étant le formalisme le mieux apprécié dans la représentation de la connaissance, la logique descriptive est celle qui est, généralement, la plus adoptée pour la représentation des règles d'inférences. La logique descriptive est définie comme étant « *une famille de formalismes de représentation de la connaissance basée sur la logique. Elle est conçue pour représenter et raisonner sur la connaissance d'un domaine d'application d'une manière structurée et bien comprise. Elle dérive des réseaux sémantiques* » [75].

Il existe deux concepts de base dans les logiques descriptives, à savoir, les concepts (prédicats) et les rôles (relations binaires). Les notions de base à prendre en considération sont la **Satisfiabilité** et la **subsomption**. Un concept « C » est dit satisfaisable quand son expression ne réfère pas à un ensemble vide. En d'autres termes s'il existe une interprétation possible pour ce concept. La Satisfiabilité est un cas particulier de la subsomption, qui n'est autre que la relation de généralisation/spécialisation. C'est pourquoi on parle d'une relation entre la Satisfiabilité et la subsomption.

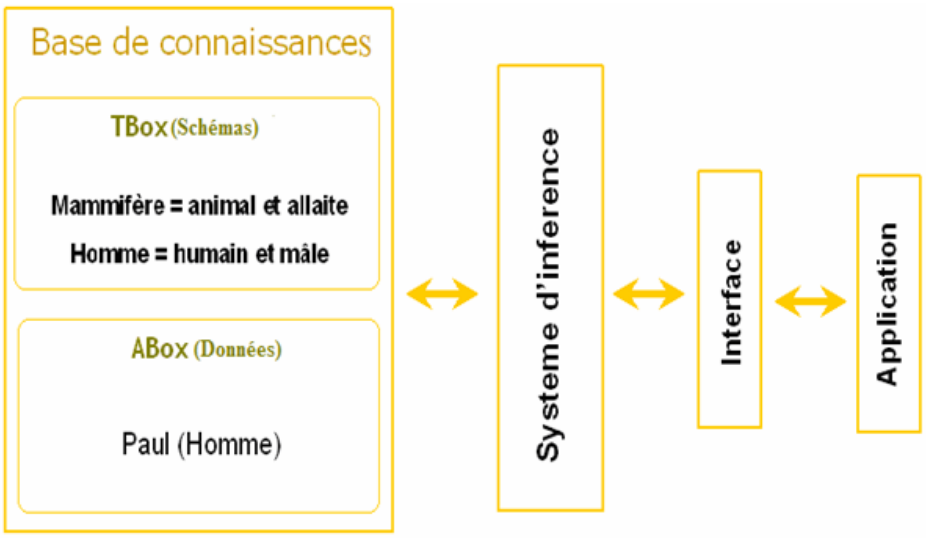


Figure 3:Architecture des bases de connaissances

Les deux éléments principaux d'une telle base de connaissances, sont le **TBox** (***T****axonomy Box*) et le **ABox** (**A**ssertional Box). Le premier, représente la connaissance conceptuelle du domaine et englobe un ensemble de formules relatives aux informations terminologiques. Le second, quant à lui, définit les instances, c'est-à-dire l'ensemble de formules relatives aux informations sur les assertions. Celles-ci sont instables et dépendantes du domaine [76]. Ainsi, toutes les informations connues sont alors modélisées comme un couple ***<T, A>***.

Si les trois premières couches ont déjà été standardisées et recommandées par le W3C, dans le cadre de l'architecture du web sémantique, il en est tout autrement pour les couches de haut niveau, que très peu d'auteurs évoquent dans leurs littératures. En effet, les recherches menées jusqu'à présent, dans ce contexte, se sont plus concentrées sur le référencement des ressources, la structuration de l'information (XML), l'affectation des métadonnées (RDF), la modélisation du monde réel sous forme d'ontologies (OWL), en faisant abstraction de la sécurité et des preuves qui servent fondamentalement à justifier la pertinence de l'information.

La couche de la preuve a pour but de prouver la pertinence de l'information retournée par les couches de plus bas niveau et des déductions obtenues à partir des inférences. Une des façons de le faire est de garder trace des sources d'information et des raisonnements effectués. Malheureusement, à l'heure où nous rédigeons ce mémoire, il n'existe encore aucun langage de preuves standardisé par le W3C. Un langage de preuve constitue un moyen simple pour prouver si une déclaration est juste ou pas. Une instance de ce dernier consiste en général en une liste de toutes les étapes d'inférence par lesquelles a transité l'information en question.

La couche confiance : Le web est un environnement très ouvert et dynamique. De ce fait, toute personne est donc en mesure d'éditer et de publier des informations de façon très simple. La couche Confiance, dans l'architecture proposée par Tim Bernes-Lee, a pour objectif d'évaluer la fiabilité de l'information et des raisonnements. Cette couche repose sur les signatures numériques, le cryptage des données et sur la fiabilité des sources d'information (agents de confiances, certifications, etc.). Le web ne pourra atteindre son plein potentiel que si les utilisateurs ont confiance dans les transactions et la qualité de l'information fournie.

Comme nous l'avons indiqué ci-dessus, la sémantique des ressources dans le Web sémantique est rendue explicite et représentée sous forme d'annotations sémantiques. L'interprétation des annotations, donc la sémantique, est précisée entre des agents logiciels, des machines ou voire des gens grâce à un des éléments les plus importants du Web sémantique : l'ontologie, que nous allons étudier dans la section suivante pour voir son rôle important dans le monde du Web sémantique.

II. Ontologies :

A. La notion d'Ontologie :

Les ontologies sont largement utilisées dans des domaines qui sont à la base du Web sémantique tels que la recherche d'informations, la gestion des connaissances, l'intégration sémantique de données, etc. Cependant, la problématique est qu'il est assez difficile de donner une définition claire et consensuelle d'une ontologie.

« Ontologie » est un terme emprunté à la philosophie qui implique une branche de la philosophie qui traite la nature et l'organisation de la réalité. Dans le domaine de l'IA, de façon moins ambitieuse, on ne considère que des ontologies, relatives aux différents domaines de

connaissances. En fait, plusieurs définitions d'ontologies sont données, mais celle qui caractérise l'essentiel d'une ontologie est fondée sur la définition donnée par [22] :

« Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée » Tom Gruber, avec la signification des termes suivants :

- ***Formelle*** : réfère au fait qu'une ontologie doit être compréhensible par la machine, c'est-à-dire que cette dernière doit être capable d'interpréter la sémantique de l'information fournie ;
- ***Explicite*** : signifie que le type de concepts utilisés et les contraintes sur leur utilisation doivent être explicitement définis ;
- ***Conceptualisation*** : se réfère à un modèle abstrait de certains phénomènes dans le monde qui identifie les concepts appropriés de ce phénomène ;
- ***Partagée*** : indique que l'ontologie supporte la connaissance consensuelle, et elle n'est pas restreinte à certains individus, mais est acceptée par un groupe.

Les taxonomies ou les thésaurus sont aussi des ontologies car elles se cantonnent à décrire des liens sémantiques du type "est-une-sorte de" et son inverse "est-représenté-par" ou, plus spécifiquement, "est-une-sous-classe-de". Des ontologies plus complexes permettent la représentation de liens sémantiques plus spécifiques, par exemple, "est-localisé-dans". Mais surtout, les ontologies les plus abouties permettent également l'intégration de propriétés particulières, de règles d'utilisation et de contraintes.

B. Rôle des Ontologies :

Nées des besoins de représentation des connaissances, les ontologies sont à l'heure actuelle au cœur des travaux menés dans le Web sémantique. Visant à établir des représentations à travers lesquelles les machines peuvent manipuler la sémantique des informations, la construction des ontologies demande à la fois une étude des connaissances humaines et la définition de langages de représentation, ainsi que la réalisation de systèmes pour les manipuler. Les ontologies participent donc pleinement aux dimensions scientifiques et techniques de l'Intelligence Artificielle (IA) : scientifiques comme étude des connaissances humaines et plus largement de l'esprit humain, ce qui rattache l'IA aux sciences humaines, et techniques comme création d'artefacts possédant certaines propriétés et capacités en vue d'un certain usage.

Au fur et à mesure des expérimentations, des méthodologies de construction d'ontologies et des outils de développement adéquats sont apparus. Émergeant des pratiques artisanales initiales, une véritable ingénierie se constitue autour des ontologies, ayant pour but leur construction, mais plus largement leur gestion tout au long d'un cycle de vie. Les ontologies apparaissent ainsi comme des composants logiciels s'insérant dans les systèmes d'information et leur apportant une dimension sémantique qui leur faisait défaut jusqu'ici.

Le champ d'application des ontologies ne cesse de s'élargir et couvre les systèmes conseillers (systèmes d'aide à la décision, systèmes d'enseignement assisté par ordinateur –

e-learning, etc.), les systèmes de résolution de problèmes et les systèmes de gestion de connaissances (par exemple dans le domaine du biomédical). Un des plus grands projets basés sur l'utilisation des ontologies consiste à ajouter au Web une véritable couche de connaissances permettant, dans un premier temps, la recherche d'informations aussi bien au niveau syntaxique qu'au niveau sémantique.

L'enjeu de l'effort engagé est de rendre les machines suffisamment sophistiquées pour qu'elles puissent intégrer le sens des informations, qu'à l'heure actuelle, elles ne font que manipuler formellement. Mais en attendant que des ordinateurs « chargés » d'ontologies et de connaissances nous soulagent en partie du travail de plus en plus lourd de gestion des informations dont le flot a tendance à nous submerger, de nombreux problèmes théoriques et pratiques restent à résoudre.

C. L'origine des Ontologies

L'Ingénierie des Connaissances (IC) est une branche de l'IA issue de l'étude des Systèmes Experts (SE). Si ces derniers n'avaient pour objet que la résolution automatique de problèmes, les Systèmes à Base de Connaissances (SBC), qui leur ont succédé sont censés permettre le stockage et la consultation de connaissances, le raisonnement automatique sur les connaissances stockées (sans préjugé sur le type de raisonnement à mener), la modification des connaissances stockées (ajout ou suppression de connaissances), et avec le développement des réseaux, le partage de connaissances entre systèmes informatiques. De manière générale, il ne s'agit plus de faire manipuler en aveugle des connaissances à la machine, qui restitue à la fin la solution du problème, mais de permettre un dialogue, une coopération entre le système et l'utilisateur humain (systèmes d'aide à la décision, systèmes d'enseignement assisté par ordinateur, e-learning, recherche d'informations sur le Web). Le système doit donc avoir accès non seulement aux termes utilisés par l'utilisateur, mais également à la sémantique que ce dernier associe aux différents termes, faute de quoi aucune communication efficace n'est possible. Plus précisément, les représentations symboliques utilisées dans les machines doivent avoir du sens aussi bien pour la machine que pour les utilisateurs, « avoir du sens » signifiant ici que l'on peut relier les informations représentées à d'autres informations.

Pour cela, la représentation des connaissances sous forme de règles logiques, utilisées dans les SE, ne suffit plus. Pour modéliser la richesse sémantique des connaissances, de nouveaux formalismes sont introduits, qui représentent les connaissances au niveau conceptuel, y compris la « structure cognitive » d'un domaine.

> *" most Knowledge Representation formalisms differ from pure first-order logic in their structuring power, i.e. their ability to make evident the structure of a domain »*
>
> - GUARINO, N., CARRARA, C., & GIARETTA, P. (1994).

Les langages à base de frames, les logiques de description et les graphes conceptuels sont des exemples de tels formalismes. Ces langages permettent de représenter les concepts sous-jacents à un domaine de connaissances, les relations qui les lient, et la sémantique de ces relations, indépendamment de l'usage que l'on souhaite faire de ces connaissances. Ainsi, une même base de connaissances peut être utilisée en consultation ou comme base de raisonnement.

Toutefois, il convient de souligner que la conceptualisation d'un domaine de connaissances ne peut se faire de manière non ambiguë que dans un contexte d'usage précis. Par exemple, un même terme peut désigner deux concepts différents dans deux contextes d'usage différents [3]. On ne peut donc mener de façon totalement indépendante la représentation des connaissances d'un domaine et la modélisation des traitements que l'on souhaite leur appliquer.

En d'autres termes, modéliser des connaissances ne peut se faire que dans un domaine de connaissances donné, et pour un but donné, condition nécessaire à l'unicité de la sémantique associée aux termes du domaine. Certains auteurs estiment cependant que les ontologies sont, par nature, destinées à être réutilisées [4], et s'attachent à construire des ontologies dont la sémantique est indépendante de tout objectif opérationnel. Au niveau ontologique, les primitives utilisées pour représenter les connaissances ne sont plus des mots du langage naturel, ou des primitives conceptuelles, et pas encore des prédicats logiques, mais des énoncés qui donnent le sens des connaissances, avec une interprétation contrainte.

Les ontologies sont donc des représentations de connaissances, contenant des termes et des énoncés qui spécifient la sémantique d'un domaine de connaissances donné dans un cadre opérationnel donné.

Le terme « ingénierie ontologique » a été proposé dans [5] pour désigner un nouveau champ de recherche ayant pour but la construction de systèmes informatiques tournés vers le contenu, et non plus vers les mécanismes de manipulation de l'information.

Avant de présenter les langages et les outils disponibles pour la manipulation d'ontologies, nous allons maintenant préciser quelles sont les primitives utilisées dans les ontologies.

D. Les constituants d'une Ontologie :

1. Les concepts :

Les connaissances portent sur des objets auxquels on se réfère à travers des concepts. Un concept peut représenter un objet matériel, une notion, une idée [6]. Un concept peut être divisé en trois parties :

- un terme (ou plusieurs)
- une notion et
- un ensemble d'objets.

La notion, également appelée ***intension*** du concept, contient la sémantique du concept, exprimée en termes de propriétés et d'attributs, de règles et de contraintes.

L'ensemble d'objets, également appelé ***extension*** du concept, regroupe les objets manipulés à travers le concept ; ces objets sont appelés ***instances du concept***. Par exemple, le terme « table » renvoie à la fois à la notion de table comme objet de type « meuble » possédant un plateau et des pieds, et à l'ensemble des objets de ce type.

Il est à noter qu'un concept peut très bien avoir une extension vide. Il s'agit alors d'un concept générique, correspondant généralement à une notion abstraite (par exemple, la « vérité », prise dans le sens de « ce qui est vrai » et non pas du « degré de vérité »). Deux concepts peuvent partager la même extension sans pour autant avoir la même intension. C'est le cas des concepts d'« étoile du matin » et d'« étoile du soir », qui désignent tous deux Vénus. De plus, des concepts partageant la même extension, mais pas leur intension peuvent être désignés par le même terme. Ceci correspond à des points de vue différents sur un même objet. Par exemple, les chiens peuvent être considérés comme des animaux de compagnie, ou comme des ressources culinaires dans certaines cultures.

Le langage naturel contienne de nombreux termes désignant plusieurs concepts sémantiques différents (par exemple « table » pour un meuble et « table » pour un tableau de valeurs numériques), et de telles ambiguïtés ne sont pas gérables en machine, où on identifie généralement un concept à l'aide de ses termes. Néanmoins, la restriction à un domaine de connaissances permet généralement d'éviter les homonymies de concepts. Il apparaît par contre souhaitable de gérer les synonymies et de permettre la désignation d'un concept par plusieurs termes, pour assurer une plus grande souplesse d'utilisation de l'ontologie [7].

2. Les propriétés :

Un autre débat non tranché porte sur les propriétés contenues dans les intensions des concepts. En effet, certaines propriétés sont essentielles à la caractérisation d'un concept, dans le sens où la suppression de cette propriété entraîne la disparition du concept en tant que tel [8]. D'autres propriétés peuvent cependant être considérées pour caractériser le concept dans un contexte donné. Ces propriétés ne sont vraies que dans ce cadre et leur disparition ne modifie pas le concept. N. Guarino ne considère comme ontologique que les propriétés nécessaires [9]. G. Kassel admet dans les ontologies des propriétés incidentes, c'est-à-dire vraies seulement dans le cadre applicatif [10].

L'exemple du concept de « table » montre que cette notion ne peut se définir qu'en utilisant d'autres concepts comme « meuble », « plateau » et « pied ». De fait, les concepts manipulés dans un domaine de connaissances sont organisés au sein d'un réseau de concepts. L'ensemble des concepts y est structuré hiérarchiquement et les concepts sont liés par des propriétés conceptuelles. La propriété utilisée pour structurer la hiérarchie des concepts est la « subsomption (super-concept) », qui lie deux concepts : un concept C1 subsume un concept C2 si toute propriété sémantique de C1 est aussi une propriété sémantique de C2, c'est-à-dire

si C2 est plus spécifique que C1. L'extension d'un concept est forcément plus réduite que celle d'un concept qui le subsume. Son intension est par contre plus riche.

Une liste des principales propriétés pouvant être associées à un concept est donnée ci-dessous. Les propriétés portant sur les extensions de deux concepts sont utiles dans le cas où on ne définit un concept que par son intension, mais où l'on veut préciser qu'aucune instance commune aux deux concepts ne doit être créée ultérieurement sans respecter certaines propriétés.

Les propriétés portant sur deux concepts sont :

- L'équivalence : deux concepts sont équivalents s'ils ont même extension. Par exemple : « étoile du matin » et « étoile du soir » ;
- La disjonction : (on parle aussi d'incompatibilité) deux concepts sont disjoints si leurs extensions sont disjointes. Par exemple : « homme » et « femme » ;
- La dépendance : un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C2. Par exemple : « parent » est un concept dépendant de « enfant » (et vice-versa).

Ces propriétés ont été formalisées, afin d'être traduites dans des langages d'ontologie.

Il est également possible de classer les propriétés que l'on utilise à l'aide d'autres critères. C. Welty propose ainsi de distinguer **les propriétés indispensables**, qui ne sont liées qu'au concept lui-même, comme la généricité, et les propriétés extrinsèques, qui font intervenir d'autres concepts dans leur définition [11].

Des modalités peuvent également être introduites au niveau des propriétés qui ne sont pas de nature modale, comme l'identité. Un concept pourrait alors porter possiblement (i.e. dans certains mondes) ou nécessairement (i.e. dans tous les mondes possibles) un critère d'identité.

3. Les instances :

En plus des propriétés, l'intension d'un concept peut contenir des attributs. Par exemple, un « chien » possède quatre attributs « pattes » et un attribut « queue », entre autres. Un attribut peut être une instance de concept. Par exemple, un « président de la République française » a toujours le Palais de l'Élysée comme « résidence officielle ». D'autre part, on peut avoir besoin d'affecter des attributs (concepts ou instances) à une instance d'un concept. Par exemple, une « Ferrari », instance d'« automobile », porte un attribut « couleur rouge » instance du concept « couleur ».

Si certains liens conceptuels existant entre les concepts peuvent s'exprimer à l'aide de propriétés portées par les concepts, d'autres doivent être représentés à l'aide de relations autonomes. Une relation permet de lier des instances de concepts, ou des concepts génériques. Elles sont caractérisées par un terme (voire plusieurs) et une signature qui précise

le nombre d'instances de concepts que la relation lie, leurs types et l'ordre des concepts, c'est-à-dire la façon dont la relation doit être lue. Par exemple, la relation « écrit » lie une instance du concept « personne » et une instance du concept « texte », dans cet ordre.

4. Les relations de spécialisation :

Tout comme les concepts, les relations peuvent être spécifiées par des propriétés dont une liste, non exhaustive, est donnée ci-après. Les relations sont organisées de manière hiérarchisée à l'aide de la propriété de subsomption décrite précédemment. Les remarques faites au sujet de l'organisation des propriétés s'appliquent également dans ce cas.

Les propriétés fondamentales d'une relation sont :

- Les propriétés algébriques : symétrie, réflexivité, transitivité ;
- La cardinalité : nombre possible de relations de même type entre les mêmes concepts (ou instances de concept). Les relations portant une cardinalité représentent souvent des attributs. Par exemple, une « pièce » a au moins une « porte », un « humain » a entre zéro et deux « jambes ».

Les propriétés liant deux relations sont :

- L'incompatibilité : deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts. Par exemple, les relations « être rouge » et « être vert » sont incompatibles ;
- L'inverse : deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I1 et I2, l'autre lie I2 et I1. Par exemple, les relations « a pour père » et « a pour enfant » sont inverses l'une de l'autre ;
- L'exclusivité : deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lie pas ces instances, et vice-versa. L'exclusivité entraîne l'incompatibilité. Par exemple, l'appartenance et la non-appartenance sont exclusives.

Les propriétés liant une relation et des concepts sont :

- Le lien relationnel : (propriété proposée par Kassel [10]). Il existe un lien relationnel entre une relation R et deux concepts C1 et C2 si, pour tout couple d'instances des concepts C1 et C2, il existe une relation de type R qui lie les deux instances de C1 et C2.
- La restriction de relation : (propriété proposée par Kassel [10]). Pour tout concept de type C1, et toute relation de type R liant C1, les autres concepts liés par la relation sont d'un type imposé. Par exemple, si la relation « mange » portant sur une « personne » et un « aliment » lie une instance de « végétarien », concept subsumé par « personne », l'instance de « aliment » est forcément instance de « végétaux ».

Ces propriétés associées aux concepts et relations complètent la sémantique différentielle de l'ontologie, au sens où elles contribuent à préciser les liens et différences entre les primitives cognitives du domaine de connaissances. Les aspects différentiels et référentiels sont cependant fortement imbriqués et la construction d'une ontologie est un processus complexe qui demande en particulier des choix de représentation délicats.

Nous présentons par la suite les langages et les outils de présentation et de création d'ontologies.

E. Classification des ontologies :

Les ontologies peuvent être classifiées en fonction de plusieurs dimensions : selon leur formalisation, leur type de conceptualisation, leur propos, leur niveau de complexité, et selon le nombre de points de vue des concepteurs ou d'utilisateurs, etc.

Selon l'état de formalisation : selon leur formalisation, les ontologies peuvent être classées en trois types :

Les ontologies semi-informelles : une ontologie est dite semi-informelle, si elle est exprimée dans une forme restreinte et structurée du langage naturel.

Les ontologies semi-formelles : une ontologie est dite semi-formelle, si elle est définie dans un langage artificiel et formellement défini.

Les ontologies rigoureusement formelles : une ontologie est dite rigoureusement formelle, si elle est définie dans un langage avec une sémantique formelle, des théories et des preuves des propriétés telles que la solidité et la perfection.

Selon leur conceptualisation : selon leur nature de conceptualisation, les ontologies peuvent être classées comme suit :

Les ontologies de représentation de connaissances [13] [12] : les ontologies de représentation des connaissances regroupent les concepts impliqués dans la formalisation des connaissances (classes, instances, propriétés, relations, restrictions, etc.).

Les ontologies de haut niveau (Top-level ontologies) [9] [12] : les ontologies de haut niveau sont des ontologies générales. Elles décrivent des concepts de manière générale, et de haute abstraction, c'est-à-dire indépendamment d'un domaine ou problème particulier, comme : les entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations, et les propriétés.

Les ontologies génériques [13] [12] : une ontologie générique appelée également *méta-ontologie* véhicule des connaissances génériques moins abstraites que celles véhiculées par l'ontologie de haut niveau, mais assez générales néanmoins pour être réutilisées à travers différents domaines.

Les ontologies de domaine (Domain ontologies) (14) [12] : les ontologies du domaine expriment des conceptualisations spécifiques à des domaines particuliers et utilisées par des

communautés bien définies. Ces conceptualisations mettent des contraintes sur la structure et les contenus des connaissances du domaine. La plupart des ontologies existantes dans le Web sémantique sont des ontologies de domaine.

Les ontologies de tâche (Task ontologies) (14) [12] : ces ontologies sont utilisées pour conceptualiser des tâches spécifiques dans les systèmes, telles que les tâches de diagnostic, de planification, de conception, de configuration de tutorat, soit tout ce qui concerne la résolution de problèmes.

Selon leur propos : selon leur propos, les ontologies peuvent être classées en :

Les ontologies d'application : l'ontologie d'application est la plus spécifique, dont les concepts correspondent souvent aux rôles joués par les entités du domaine tout en exécutant une certaine activité (15) *[12]*. Elle contient toutes les définitions nécessaires pour décrire la connaissance requise pour une application particulière.

Les ontologies de référence : elles sont définies par *[12]* comme des ontologies conçues pour décrire un domaine correctement. Les ontologies de référence sont utilisées durant le processus de développement d'applications pour une compréhension mutuelle entre des agents appartenant à des communautés différentes, pour établir un consensus dans une communauté qui a besoin d'adopter de nouveaux termes, ou tout simplement pour expliques le sens des termes aux nouveaux arrivants dans la communauté.

Selon le niveau de complexité :

Les ontologies légères : sont des structures taxonomiques simples, composées de primitives ou de termes et les définitions qui leur sont associées. Les ontologies légères ne sont pas ou peu axiomatisées, et ne garantissent pas un engagement ontologique. Elles sont réduites à des structures taxonomiques entre des termes considérés comme appropriés.

Les ontologies lourdes : ces ontologies ont recours à une axiomatisation riche, à la différence des ontologies légères, les ontologies lourdes représentent explicitement l'engagement ontologique. Le but de l'axiomatisation est d'exclure des ambiguïtés terminologiques et conceptuelles, dues aux interprétations fortuites.

Selon le nombre de points de vue de concepteurs : Yiling Lu dans sa thèse [16] [12], a classé les ontologies en deux types :

Les ontologies inspirationnelles : les ontologies inspirationnelles sont conçues selon un seul point de vue du concepteur de domaine.

Les ontologies collaboratives : à la différence des ontologies inspirationnelles, les ontologies collaboratives sont conçues selon multiples points de vue de différents concepteurs et acteurs de domaines.

Selon le nombre de points de vue des futurs utilisateurs : À travers ce que nous avons présenté dans les sections précédentes, il ressort que la notion d'ontologie constitue l'une des

approches les plus efficaces pour représenter les connaissances d'un domaine, mais l'existence de plusieurs façons d'appréhender ces connaissances selon différents points de vue donne lieu à la naissance d'une autre notion d'ontologie qui est l'ontologie multipoints de vue. Dans nos travaux, nous nous sommes rendu compte que nous pouvons aussi classer les ontologies selon le nombre de points de vue de futurs utilisateurs de l'ontologie. Nous distinguons alors :

Les ontologies mono-point de vue : elles sont conçues selon à un point de vue global des futurs utilisateurs.

Les ontologies multipoints de vue : Une ontologie étant une représentation subjective de la réalité, il semble difficile (voire impossible) d'imaginer que les concepteurs d'ontologies puissent se mettre d'accord sur une représentation commune de la réalité, et que celle-ci soit adaptée aux besoins de tous les utilisateurs (17) [12]. Les ontologies MPV sont conçues selon les différents points de vue de différents futurs utilisateurs et c'est là où se situe notre travail.

La figure (Figure 3) montre un résumé de la classification présentée.

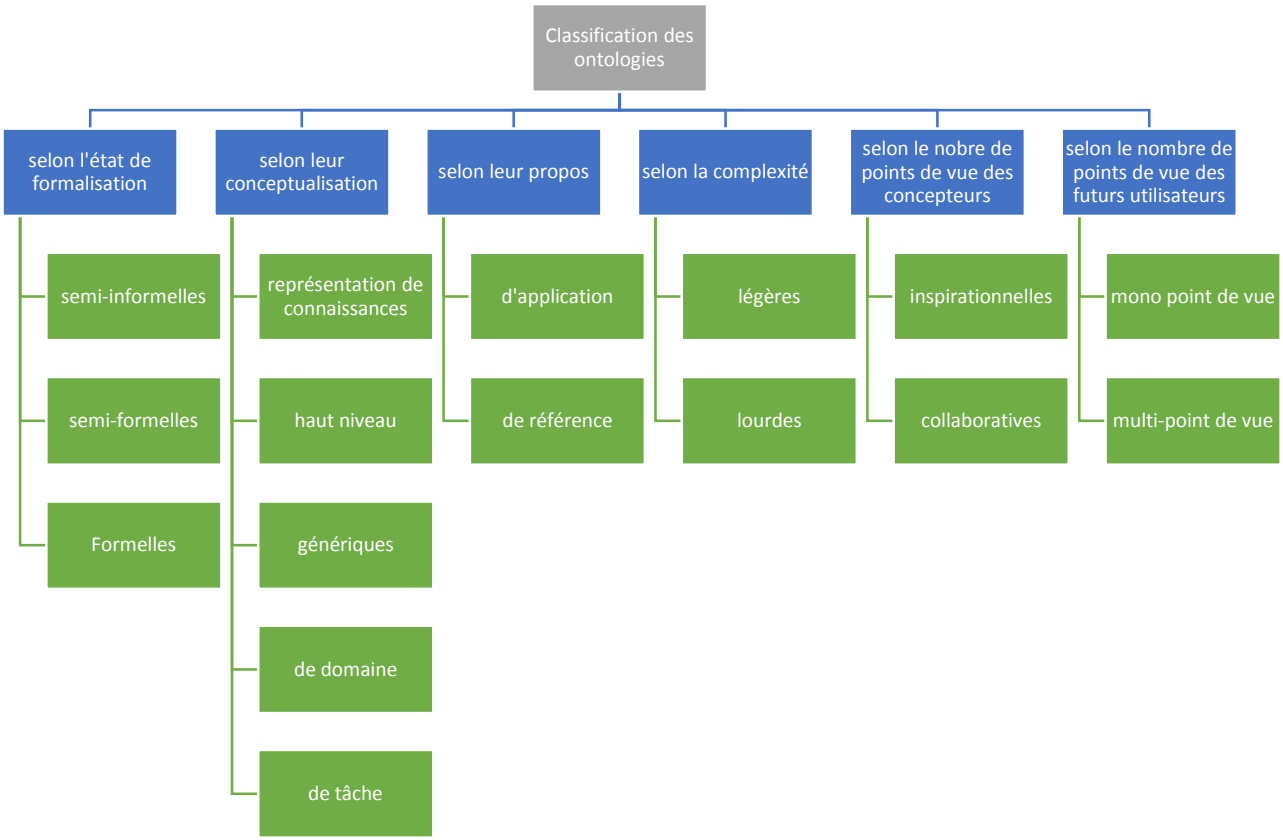


Figure 4:Classification des ontologies

F. Les langages de représentation :

Dans cette section, nous allons présenter quelques langages de représentation des ontologies.

1. KIF :

KIF [18] est un langage basé sur les prédicats du premier ordre avec des extensions pour représenter des définitions et des méta-connaissances, la logique du premier ordre étant un langage de bas niveau pour l'expression d'ontologies. Une extension du langage KIF, ONTOLINGUA, est utilisée dans le serveur d'édition d'ontologies, Ontolingua1 du même nom.

2. RDF et RDF Schéma :

Le W3C a adopté le langage RDF (Ressource Description Framework) comme un des formalismes standards de représentation de connaissances sur le Web2. Utilisant la syntaxe XML (Extended Markup Language3) qui constitue déjà un standard, le RDF permet de décrire des ressources Web en termes de ressources, propriétés et valeurs. Une ressource peut être une page Web (identifiée par son URI, United Ressource Identifier) ou une partie de page (identifiée par une balise). Les propriétés couvrent les notions d'attributs, relations ou aspects et servent à décrire une caractéristique d'une ressource en précisant sa valeur. Les valeurs peuvent être des ressources ou des littéraux. RDF dispose d'une sémantique formelle analogue à celle des graphes conceptuels, c'est-à-dire identique à celle d'un fragment de la logique du premier ordre [19].

Pour décrire n'importe quel type de connaissances à l'aide de ce formalisme, nous devons d'abord écrire en RDF le modèle sémantique à utiliser. Par exemple, pour décrire des connaissances en termes de concepts et de relations hiérarchisés, l'introduction des types « concepts » et « relations » et des propriétés de subsomption et d'instanciation est nécessaire. Un schéma de base incluant les primitives sémantiques généralement utilisées a ainsi été ajouté au RDF et constitue ce qu'on appelle le RDF SCHEMA (RDF-S1). La Figure 2 montre les primitives de RDF(S). Les concepts et les relations sont déclarés dans un document RDF(S) comme instances de « Class » et de « Property ».

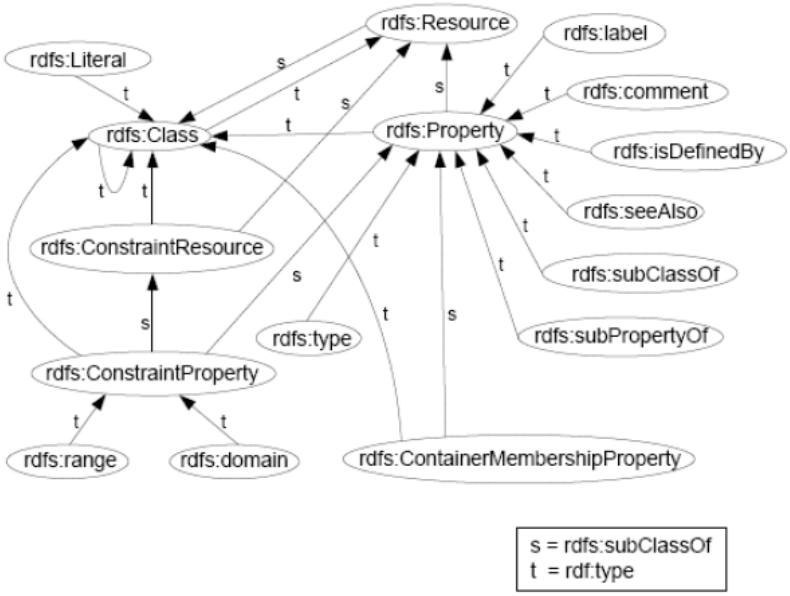


Figure 5: Le schéma de RDF(S)

Une fois ce schéma stocké sur le Web, les primitives qui y sont décrites peuvent être utilisées dans une page si on y inclut une référence à l'URI du schéma. Une application nécessitant l'accès à la sémantique de la page utilisera alors le schéma d'interprétation. Le RDF-S n'est cependant pas un langage opérationnel de représentation, au sens où il ne permet pas la représentation des axiomes et leur utilisation pour raisonner.

3. DAML + OIL :

Dans l'optique d'une utilisation d'ontologies sur le Web, le langage RDF-S a été enrichi par l'apport du langage OIL (Ontology Interchange Language) qui permet d'exprimer une sémantique à travers le modèle des frames tout en utilisant la syntaxe de RDF-S. OIL offre de nouvelles primitives permettant de définir des classes à l'aide de mécanismes ensemblistes issus des logiques de description (intersection de classes, union de classes, complémentaire d'une classe). Il permet également d'affiner les propriétés de RDF-S en contraignant la cardinalité ou en restreignant la portée [20].

Le langage OIL a été fusionné avec le langage DAML pour former le DAML+OIL. DAML (Darpa Agent Markup Language) est conçu pour permettre l'expression d'ontologies dans une extension du langage RDF. Il offre les primitives usuelles d'une représentation à base de frames et utilise la syntaxe RDF [21]. L'intégration de OIL rend possibles les inférences compatibles avec les logiques de description, essentiellement les calculs de liens de subsomption.

4. OWL :

La combinaison de RDF/RDF-S et de DAML+OIL a permis l'émergence de OWL (Web Ontology Language), un langage standard de représentation de connaissances pour le Web.

Développé par le groupe de travail sur le Web Sémantique du W3C, OWL peut être utilisé pour représenter explicitement les sens des termes des vocabulaires et les relations entre ces termes. OWL vise également à rendre les ressources sur le Web aisément accessibles aux processus automatisés [22], d'une part en les structurant d'une façon compréhensible et standardisée, et d'autre part en leur ajoutant des méta informations. Pour cela, OWL a des moyens plus puissants pour exprimer la signification et la sémantique que XML, RDF, et RDF-S. De plus, OWL tient compte de l'aspect diffus des sources de connaissances et permet à l'information d'être recueillie à partir de sources distribuées, notamment en permettant la mise en relation des ontologies et l'importation des informations provenant explicitement d'autres ontologies.

Sous langages d'OWL :

OWL a trois sous langages de plus en plus expressifs : OWL Lite, OWL DL, et OWL Full :

1. OWL Lite : Il supporte les utilisateurs ayant besoin principalement d'une hiérarchie de classification et des contraintes simples (un ensemble est limité à 0 ou 1 élément, par exemple). Il a une complexité formelle inférieure à celle

d'OWL DL. OWL Lite supporte seulement un sous-ensemble de constructions du langage OWL.

2. OWL DL : D'après son nom, OWL DL utilise la logique de description DL [19]. Il a été défini pour les utilisateurs qui réclament une expressivité maximale tout en retenant la complétude informatique (toutes les conclusions sont garanties 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.
3. OWL Full : Il a été défini pour les utilisateurs qui veulent une expressivité maximale et une liberté syntaxique de RDF mais sans les garanties informatiques. OWL Full permet à une ontologie d'augmenter la signification du vocabulaire prédéfini (RDF ou OWL). Il est peu probable que n'importe quel logiciel de raisonnement soit capable de supporter le raisonnement complet de chaque caractéristique d'OWL Full. Autrement dit, en utilisant OWL Full en comparaison avec OWL DL, le support de raisonnement est moins prévisible puisque l'implémentation complète d'OWL Full n'existe pas actuellement.

OWL Full et OWL DL maintiennent le même ensemble de constructions d'OWL. La différence se situe dans les restrictions sur l'utilisation de certaines de ses caractéristiques et sur l'utilisation des caractéristiques de RDF. OWL Lite permet le mélange libre d'OWL avec RDFS et, comme RDFS, n'impose pas une séparation stricte des classes, des propriétés, des individus, et des valeurs de données. OWL Full peut être vu comme étant une extension de RDF, tandis que OWL Lite et OWL DL peuvent être vus comme des extensions d'une vue restreinte de RDF. Alors, les utilisateurs de RDF devraient se rendre compte qu'OWL Lite n'est pas simplement une extension de RDFS. OWL Lite met des contraintes sur l'utilisation du vocabulaire de RDF (par exemple, disjointes des classes, des propriétés, etc.). OWL Full est conçu pour la compatibilité maximale de RDF. Quant à opter pour OWL DL ou OWL Lite, il faut considérer si les avantages du OWL DL/Lite (par exemple, support de raisonnement) l'emportent par rapport aux restrictions de DL/Lite à l'utilisation des constructions de OWL et de RDF.

G. Les outils de construction d'Ontologies

De nombreux outils de construction d'ontologies utilisent des formalismes variés et offrent différentes fonctionnalités. Seuls les plus connus seront cités ici. Tous ces outils offrent des supports pour le processus de création d'ontologies, mais peu offrent une aide à la conceptualisation.

1. DOE

DOE (Differential Ontologie Editor) [23] [24] offre la possibilité de construire les hiérarchies de concepts et relations en utilisant les principes différentiels énoncés par B. Bachimont, puis en ajoutant les concepts référentiels. La sémantique des relations est ensuite précisée par des contraintes. Ce n'est qu'une fois l'ontologie ainsi structurée qu'elle soit formalisée en utilisant la syntaxe XML.

2. PROTEGE2000 :

Parmi les outils non liés à des formalismes de représentation, citons l'outil PROTEGE2000 [25], [26]. PROTEGE2000 est une interface modulaire permettant l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, et la fusion semi-automatique d'ontologies à l'aide du plug-in Prompt [27].

Le modèle de connaissances sous-jacent à PROTEGE-2000 est issu du modèle de frames et contient des classes (concepts), des slots (propriétés) et des facettes (valeurs des propriétés et contraintes), ainsi que des instances de classes et des propriétés. Il autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

Il présente une interface graphique très agréable et simple à utiliser. Son utilisation dans le cadre de notre étude a été bénéfique pour pouvoir manipuler et/ou modifier les ontologies (de type rdf ou owl) ou encore pour créer des ontologies de test.

3. OntoEdit :

OntoEdit (Ontology Editor) [28] est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. OntoEdit permet d'éditer et de créer des ontologies de type 'obo'. Il est très utilisé dans le domaine biomédical. OntoEdit intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition. Enfin, un plug-in nommé ONTOKICK offre la possibilité de générer les spécifications de l'ontologie par l'intermédiaire de questions de compétences.

Conclusion :

Nous avons décrit dans ce chapitre le cadre technique de notre travail. Nous avons présenté les concepts de base des ontologies et du web sémantique, qui sont la base de notre travail. Nous avons ensuite présenté les deux domaines d'application cibles sur lesquels nous avons illustré notre travail. Voici un résumé des principaux points que nous avons abordés :

- Le Web sémantique est établi sous forme de couches et l'ontologie est l'élément noyau du Web sémantique ;
- Une ontologie est une spécification explicite et formelle d'une conceptualisation partagée ;
- La conception d'une ontologie est un processus itératif ;
- Il existe plusieurs langages de spécification d'ontologie avec des caractéristiques différentes. Parmi eux, DAML+OIL, RDF, OWL et F-Logic ;

- Pour permettre la création et la manipulation d'ontologies, une série d'outils sont nécessaires. Parmi eux, PROTEGE2000 et OboEdit sont sans doute les plus utilisés.

Après ce panorama des technologies qui constituent le paysage technique de cette recherche, le chapitre suivant se focalise sur les travaux de recherche autour de l'alignement d'ontologies.

Chapitre II:État de l’art sur l’alignement d’ontologies

Introduction :

Dans ce chapitre, nous présentons un état de l'art sur l'alignement des ontologies dans le web sémantique qui constitue la base de notre recherche. Et une fois que nous clarifions : comment est né le besoin à l'alignement des ontologies, nous nous intéressons à sa définition, ses objectifs, ses difficultés et aux différentes notions connexes. Nous enchaînons plus en profondeur avec les méthodes de base utilisées pour réaliser l'alignement, ces dernières sont combinées pour construire des systèmes automatiques d'alignement qui à leurs tours sont classés selon différents critères. Enfin, nous présentons des travaux dans la littérature qui attaquent ce difficile problème en décrivant succinctement les systèmes les plus cités dans les articles de recherche comme étant les plus performants, ainsi que quelques des outils les plus utilisés dans les systèmes d'alignement d'ontologies.

I. L'alignement des ontologies :

A. Comment est né le besoin à l'alignement des ontologies ?

L'évolution du web a permis d'intégrer la sémantique au web, en donnant un sens à ces ressources. Cette sémantique est représentée avec des ontologies (l'abstraction des connaissances d'un domaine). La réalisation de cette abstraction se fait de différentes manières selon la personne ou l'organisation qui modélise l'ontologie, ce qui va créer un ensemble d'ontologies hétérogènes, donc de l'hétérogénéité dans le web sémantique.

Pour bien expliquer d'où vient cette hétérogénéité dans la modélisation d'ontologie, on va donner l'exemple suivant. On a deux personnes ou organisations qui veulent modéliser une ontologie pour le même domaine, c'est sûr que chacun entre eux va donner une modélisation selon le contexte dont il voit l'information ou la connaissance, aussi il va utiliser sa propre langue et ses propres termes pour réaliser la représentation terminologique de cette modélisation, par ex. : une personne américaine peut modéliser le sport « football » avec le terme « soccer » alors qu'une autre en Europe utilise le terme « football ». Aussi ils peuvent utiliser de différentes sources de connaissances, car il peut y avoir plusieurs ressources qui représentent la même connaissance. De même pour l'extraction des connaissances et informations peut être fait différemment à partir d'une même ressource.

Donc, en général, selon l'utilisateur et la source de connaissance se produit l'hétérogénéité dans le Web sémantique au niveau des ontologies. Cette hétérogénéité posera par la suite beaucoup de problèmes au niveau de l'échange, traitement, intégration et recherche de l'information.

B. Problème issu de l'hétérogénéité :

L'hétérogénéité n'est pas seulement due à la divergence des domaines que peuvent couvrir les ontologies mais aussi aux formalismes requis pour leur développement. Dans la littérature, plusieurs classifications des types d'hétérogénéité sont recensées. Quelques classifications se basent sur l'étude du décalage sémantique et structurel (29) qui peut exister entre les ontologies. D'autres classifications élucident le degré l'hétérogénéité selon les

niveaux d'interopérabilité sémantique (30). La littérature recense quatre types d'hétérogénéités, à savoir : l'hétérogénéité syntaxique, l'hétérogénéité terminologique, l'hétérogénéité conceptuelle et l'hétérogénéité sémiotique.

L'hétérogénéité syntaxique se produit quand deux ontologies sont décrites avec deux langages ontologiques différents. Ce type d'hétérogénéité se manifeste lors de la comparaison d'un répertoire avec un modèle conceptuel. Cette hétérogénéité se produit aussi quand deux ontologies sont modélisées en utilisant des formalismes différents, par exemple OWL et F-Logic (Frame Logic). Cette classe d'hétérogénéité survient au niveau théorique, notamment, quand il s'agit d'établir des équivalences entre les primitives de différents langages ontologiques. Il est possible dans certains cas de traduire les ontologies dans différents langages ontologiques à condition de préserver la signification.

L'hétérogénéité terminologique se manifeste dans l'éventualité où deux entités sont référencées par deux noms différents alors qu'elles désignent le même objet. La cause d'une telle hétérogénéité revient à l'utilisation de différents langages naturels, ou des sous-langages techniques spécifiques à un domaine de connaissances bien déterminé. Elle se manifeste aussi par l'utilisation des synonymies.

L'hétérogénéité conceptuelle est appelée aussi hétérogénéité sémantique [30] ou la différence logique [31]. Elle concerne la diversité des modélisations d'un même domaine de connaissances. Elle découle principalement de l'utilisation de différents (ou équivalents) axiomes décrivant les concepts ontologiques. Elle se manifeste aussi lors de l'utilisation de concepts totalement différents. Respectivement [31] et [32] évoquent la différence de conceptualisation et la différence de l'explication. La différence de conceptualisation se manifeste à travers la différence entre les concepts inclus dans la modélisation. La différence des explications se base sur la manière avec laquelle les concepts sont exprimés. [32] proposent une classification précise de ces différences. Par ailleurs [33] présentent les trois principales raisons de la différence de conceptualisation :

- Différence de convergence qui survient lorsque deux ontologies décrivent différentes connaissances avec le même niveau de détail pour une unique perspective ;
- Différence de granularité qui se produit quand deux ontologies décrivent le même domaine avec une même perspective, mais avec différents degrés d'expression des détails ;
- Différence de perspectives qui se manifeste quand deux ontologies décrivent un même domaine, avec un même degré d'expression des détails, mais avec des points de vue et des perspectives différents.

L'hétérogénéité sémiotique est appelée aussi hétérogénéité pragmatique [29]. L'hétérogénéité sémiotique s'intéresse à la manière dont les entités ontologiques sont interprétées par leurs utilisateurs. Ainsi, les entités ayant les mêmes interprétations sémantiques peuvent être interprétées de différentes manières par l'Homme.

Ces différences d'interprétation sont dues principalement à la diversité des contextes et des domaines d'application des ontologies. Par conséquent, la manière de mettre en œuvre les entités ontologiques influence leurs interprétations. De plus, ce type d'hétérogénéité reste difficile à détecter par la machine.

Mais grâce à l'alignement des ontologies, on peut lier entre les ontologies du web sémantique même si l'hétérogénéité existe. Cette liaison se fait en cherchant les correspondances entre les ontologies. Afin de pouvoir réaliser quelques tâches comme l'échange, l'intégration… etc.

C. Définition de l'alignement d'ontologies :

L'alignement dans son sens général, désigne l'ajustement des objets par rapport aux autres selon une orientation voulue. Dans le domaine d'informatique, plus particulièrement dans le Web Sémantique, on parle de l'alignement des ontologies qui représente une tâche cruciale dans plusieurs sous-domaines d'application : la communication dans les systèmes multi-agents, data warehouse, etc. Et comme nous avons mentionné dans un premier temps que le problème actuel, c'est que dans un même domaine ou des domaines connexes, on risque d'avoir plusieurs ontologies différentes (hétérogénéité). C'est pour cela qu'on a introduit la notion de l'alignement d'ontologies qui sert à comparer deux ontologies et de gérer le plus automatiquement possible, des appariements sur les ontologies, et qui consiste à trouver des 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 ces correspondances est ce qu'on appelle ***alignement***, et qu'on exploite après conjointement dans le même système.

- ***le processus d'alignement :***

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

L'input ou bien l'entrée qui se constitue des structures destinées à être alignées, elles peuvent être des schémas XML, schémas relationnels, des ontologies. Dans notre cas, ce sont les dernières.

NB : l'input peut être enrichi par un alignement en entrée, comme l'alignement initial A dans la figure 6 (par exemple, la plupart des méthodes d'alignement utilisent un alignement terminologique simple qui sera l'input de la méthode d'alignement principale).

Le processus d'alignement : est une tâche pendant laquelle à partir d'une ontologie O_1 et une autre O_2, il détermine un alignement A' entre ces deux ontologies, cette tâche est réalisée en utilisant une stratégie ou une combinaison de techniques d'alignement de bases. On verra dans la suite de ce chapitre ces techniques en détail.

L'output : ensemble d'alignement reliant les entités qui constituent les deux ontologies qui comporte :

- Id : un identifiant de l'alignement
- e : une entité à aligner et qui appartient à O_1
- e' : une entité à aligner appartenant à l'ontologie O_2
- r : la relation qui lie e à e'
- n : la mesure de confiance de r comprise dans [0,1], 1 étant une relation forte.

La figure qui suit est une représentation schématique du processus d'alignement :

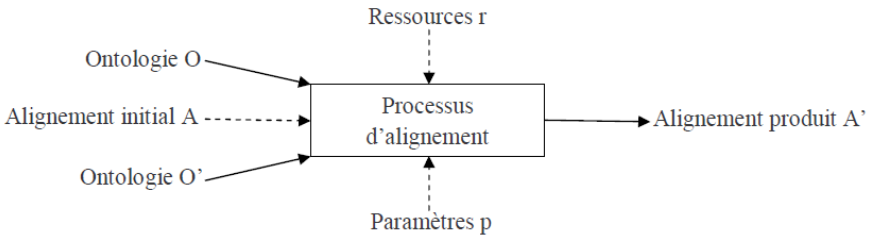


Figure 6: Schéma résumant le processus d'alignement

D. Les objectifs de l'alignement d'ontologies :

L'objectif de l'alignement d'ontologies est de réaliser l'interopérabilité sémantique. L'interopérabilité sémantique est la capacité de deux ou plusieurs systèmes d'information à trouver une compréhension commune, à partir des données échangées, afin de produire des résultats utiles. Cela en mettant deux ontologies hétérogènes dans un accord mutuel en détectant un ensemble de correspondances, entre les entités de ces ontologies sémantiquement liées, afin de permettre aux données d'être échangées, traitées, intégrées… grâce à l'utilisation de différentes méthodes et approches.

E. Les difficultés de l'alignement d'ontologies :

L'alignement des ontologies est venu pour gérer le problème de l'hétérogénéité dans le web sémantique en produisant un ensemble de correspondances liant les entités des deux ontologies à aligner (des concepts, instances, propriétés, etc.) deux à deux par une relation (équivalence, subsomption, etc.). Mais même si le nombre de méthodes progresse régulièrement, il n'y a pas de solution intégrée qui représente un clair succès, et qui est suffisamment robuste pour être la base des futures études, et qui soit utilisable même par les utilisateurs non-experts. Cela est essentiellement à cause de la nature exploratoire du problème d'alignement puisque l'alignement est une génération d'hypothèse sur la sémantique des entités en n'ayant accès qu'à une connaissance descriptive. Par conséquent, les méthodes vont typiquement appliquer des combinaisons techniques élémentaires de nature heuristique (fouille de données, analyse de similarité, analyse de concept, raisonnement formel, etc.), implémentées par des approches indépendantes, et la contribution et l'orchestration de ces approches sont régies par un nombre important de

paramètres spécifiques à chaque méthode. Ceci rend l'analyse de l'impact d'une technique, ou paramètre difficile à évaluer et plus particulièrement si lorsque le nombre et le volume des schémas de données sont importants. En effet, dans les domaines d'applications réelles où les ontologies sont volumineuses et complexes, les exigences de l'exécution du temps et de l'espace mémoire sont les deux facteurs significatifs qui influencent directement la performance d'un algorithme d'alignement.

Malgré le nombre toujours croissant de méthodes et d'études comparatives dans le domaine, il manque encore un corpus de connaissance consensuelle sur la façon d'aborder les problèmes de l'alignement ainsi que sur les mérites respectifs des diverses techniques et méthodes.

F. Les Méthodes d'alignement :

Les méthodes d'alignement d'ontologies se basent sur les méthodes de calcul de similarité. Dans cette partie, nous allons voir la relation entre l'alignement et la similarité, la définition de la similarité et les différentes méthodes de calcul de similarité (terminologiques, structurelles, extensionnelles et sémantiques).

1. La relation entre l'alignement et la similarité :

L'alignement des ontologies consiste à chercher la similarité entre les ontologies à aligner, donc pour chercher des correspondances entre les entités des ontologies il faut analyser les hiérarchies des entités (la structure locale et globale des entités), cela en utilisant les techniques de l'alignement qui sont les méthodes de calcul de similarité (terminologiques, structurelles, extensionnelles et sémantiques).

2. La similarité c'est quoi ?

La similarité est un concept important et largement utilisé, qui est lié à un domaine d'application particulier ou à une forme de représentation des connaissances.

Dans notre contexte, la notion de similarité est plutôt celle de la similarité sémantique, qui est également appelée la proximité sémantique. Elle est déterminée grâce à l'association à des documents, des termes ou des entités, d'une métrique basée sur la similitude de leurs significations ou de leurs contenus sémantiques. La similarité est la quantité qui reflète la force du rapport entre deux objets ou deux caractéristiques.

La similarité sémantique, c'est-à-dire l'appréhension de la liaison entre deux concepts, est une capacité de l'homme que les machines ne savent que très mal reproduire. Ainsi, pour un humain, il est évident que les concepts de crayon et de papier sont liés, beaucoup plus que ceux de parapluie et fer à repasser en tout cas. Mais il est très difficile de le formaliser, car rien, en surface, ne permet de le décider. Pour ce faire, il faut utiliser des ressources sémantiques : les ontologies, c'est-à-dire des bases de connaissances. Elles seules permettent de montrer les liens (hyperonymie, antonymie, etc.) entre des concepts.

Les recherches sur ce sujet se font sur plusieurs domaines : intelligence artificielle, psychologie, sciences cognitives, et ce depuis de nombreuses années. Les modèles de calcul

de la similarité sémantique se retrouvent dans de multiples applications, avec pour but de donner à ces dernières des connaissances supplémentaires pour raisonner sur leurs données.

Dans la plupart des approches dans notre contexte, la notion de similarité sémantique est vue comme celle de la similarité topologique en mathématiques, où on l'associe à une fonction, appelée fonction de la similarité. La définition de cette fonction de la similarité peut changer selon les approches, selon les propriétés souhaitées. La valeur de cette fonction est souvent comprise entre 0 et 1, ce qui permet des possibilités d'interprétation probabiliste de la similarité. Des propriétés ou des caractéristiques communes possibles de la fonction sont des caractéristiques positives, auto similaires ou maximales, symétriques ou réflexives. On peut aussi trouver d'autres caractéristiques telles que la finitude ou la transitivité.

Définition 1 (Similarité). La similarité $S : O \times O$ R est une fonction d'une paire d'entités à 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 $S(a, a) = S(a, b) \Leftrightarrow a = b$ (autosimilarité ou maximalité)
- $\forall a, b \in O, S(a, b) = S(b, a)$ (symétrie)
- $\forall a, b, c \in O, S(a, b) = S(b, c) \Rightarrow S(a, b) = S(a, c)$ (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 :

Définition 2 (Dissimilarité). La dissimilarité $DS : O \times O \rightarrow R$ 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 utilisée aussi souvent que les mesures de similarité. Elle mesure la dissimilarité de deux entités, elle est inverse de la similarité : si la valeur de la fonction de similarité de deux entités est élevée, la distance entre ces entités est petite et vice-versa. Elle est donc définie dans [34] comme suit :

Définition 3 (Distance). La distance $D : O \times O \rightarrow R$ est une fonction de la dissimilarité satisfaisant la définitivité 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 $\bar{S}$ et $\overline{DS}$, alors on a $\bar{S}$ +$\overline{DS}$ =1.

Définition 4 (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 $\bar{f}$.

3. Les méthodes d'alignement basées sur la similarité :

a) Les méthodes terminologiques :

Les méthodes terminologiques comparent les chaînes de caractères afin d'en déduire la similarité (ou dissimilarité) en exploitant les relations d'hyperonymie ou d'hyponymie. Certaines méthodes terminologiques se basent sur la comparaison des chaînes de caractères et sont appelées les méthodes syntaxiques. Ces méthodes syntaxiques comparent les structures des deux chaînes. Deux chaînes qui partagent des caractères ou des mots en commun seront considérées comme similaires. D'autres méthodes terminologiques qui ont recours à une base de données lexicale, sous forme de réseau sémantique, sont les méthodes linguistiques. Ces méthodes calculent la similarité entre deux chaînes en fonction des relations qu'elles entretiennent.

On peut diviser les méthodes terminologiques en deux catégories : les méthodes syntaxiques et les méthodes linguistiques [77].

(1) Les méthodes syntaxiques :

Les méthodes syntaxiques comparent, les termes ou les chaînes de caractères ou bien les textes, des entités à aligner. Ces méthodes permettent de calculer la valeur de la similarité des entités textuelles. Ces entités sont caractérisées par des noms, des étiquettes, des commentaires, des descriptions, etc. Ces méthodes se déclinent en deux sous-catégories. La première sous-catégorie englobe des méthodes qui comparent des termes en se basant sur les caractères contenus dans ces termes. La deuxième sous-catégorie utilise les distances basées sur les tokens [35].

(a) Les méthodes se basant sur les chaînes de caractères [78] :

La structure des chaînes des caractères, l'ordre des caractères dans la chaine, le nombre d'occurrences d'une lettre dans une chaine pour concevoir des mesures de la similarité. Par contre, elles n'exploitent pas la signification des termes.

Les résultats de la comparaison des chaines seront améliorés si chaînes passent une phase qu'on appelle la phase de normalisation, pendant laquelle ces chaines sont « nettoyées » ou « traitées », avant de les fournir aux formules calculant la similarité. Dans ce qui suit, on présente les différents types de normalisations :

- Normalisation des caractères : ce type rend une chaîne de caractère soit tout en majuscules, soit tout en minuscule.
- Normalisation des espaces : remplace toutes les séquences consécutives des espaces, tabulations… trouvées dans une chaîne de caractères par un seul caractère d'espace.
- Suppression des signes diacritiques ou des accents (aigus, grave…) : ce type de traitement remplace des caractères avec des signes diacritiques par caractères correspondants sans signes diacritiques.
- Suppression des chiffres.
- Élimination des ponctuations.
- Élimination des mots vides (les mots contenant peu d'informations tels que : est, un ,une…).
- Suppression des affixes (préfixe, suffixe)
- Extension des abréviations.
- Tokenisation.
- Lemmatisation (passer au singulier, à l'infinitif pour les verbes…)

Pour calculer la valeur de similarité ou bien la distance entre deux chaines de caractères, il existe plusieurs mesures pour faire cela : la similarité de Jaccard, la distance de Hamming, la distance de Levenshtein… Nous, on va présenter ceux qui sont les plus utilisées dans les approches d'alignement d'ontologies.

Définition 5 *(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* $S_{Jaccard}$ *:* $S \times S \rightarrow [0, 1]$ *telle que* :

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

(NB. Les opérations d'édition : Des opérations d'édition typiques sont celles de l'insertion, de la suppression, et de la substitution de caractère, et à chaque opération est affecté à un coût**).**

Définition 6 *(Distance d'édition). Soit un ensemble d'opérations d'édition* OP, $op \in OP$, $op : S \rightarrow S$, *et une fonction de coût d'édition* $w : OP \rightarrow \mathbf{R}$. *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} *:* $SxS \rightarrow [0, 1]$ *telle que* $DS_{de}(s, t)$ *est le coût de la séquence des opérations la moins coûteuse qui transforme s en t.*

$$\overline{DS_{de}}(s,t) = min_{(op_i)_I : op_n(\ldots op_1(s)) = t}\left(\sum_{i \in I} w_{op_i}\right)$$

Les variantes de cette mesure sont différentes au niveau des coûts assignés aux opérations d'édition. On peut trouver :

- La distance de Levenstein où tous les coûts sont égaux à 1.

- La distance de Damerau qui est presque identique à la distance de Levenshtein, mais qui peut tolérer des caractères adjacents qui ont été permutés, une erreur typographique habituelle.
- La distance de Smith-Waterman [36] qui affecte des coûts relativement inférieurs à la séquence des insertions ou des suppressions.
- La distance de Needleman-Wunsch avec des matrices des coûts pour chaque paire des caractères dans des opérations des insertions ou des suppressions.

La métrique ([37] et [38]) produit la similarité entre deux chaînes de caractères en se basant sur le nombre et l'ordre des caractères communs entre elles.

Définition 7 *(Distance de Jaro). Soit s et t deux chaînes de caractères. Soit Nc le nombre des caractères communs apparaissant dans les deux chaînes dans une distance de moitié de la longueur de la chaîne la plus courte. Soit Nt le nombre des caractères transposés, qui sont des caractères communs apparaissant dans des positions différentes. La distance de Jaro est une fonction de la dissimilarité DSJaro* : S x S ➔ [0, 1] telle que :

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

Définition 8 *(Distance de Jaro-Winkler variante de la distance de Jaro). Soit s et t deux chaînes de caractères. Soit P la longueur du préfixe commun le plus long de s et t. Soit n un nombre positif. La distance de Jaro-Winkler est une fonction de la dissimilarité DSJaroWinkler : S x S ➔ [0, 1] telle que :*

$$\overline{DS_{\text{JaroWinkler}}}(s,t) = \overline{DS_{Jaro}}(s,t) - \frac{max(P,n)}{10}\overline{DS_{Jaro}}(s,t)$$

(b) Les distances basées sur les tokens :

Les mesures qu'on vient de présenter ci-dessus sont convenables lorsqu'on veut comparer deux termes ou bien deux courtes chaines de caractères, mais il y a des cas où on aurait besoin de comparer de longs textes, ou même des documents textuels. Dans ces cas, on découpe ces entités en plusieurs petits morceaux appelés *tokens.*

Il existe bien sûr plusieurs mesures de similarité dans cette catégorie : similarité de Dagan(1999), distance de Jensen-Shannon… Nous, on va présenter celles qui sont les plus utilisées.

La similarité de Jaccard (Définition 5) peut aussi être étendue pour comparer des ensembles des tokens.

Une autre mesure largement utilisée : TF/IDF (Term Frequency/ Inverse Document Frequency) qui est employée pour mesurer la pertinence d'un terme l'ensemble de documents. La fréquence de terme, TF, dans un document donné montre l'importance de ce

terme dans le document en question. La fréquence inverse de document, IDF, est une mesure de l'importance générale du terme dans l'ensemble de documents.

Définition 9 *(TF/IDF). 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 suivante :*

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

La similarité des entités textuelles peut être construite comme la valeur cosinus de deux vecteurs représentant ces entités, où chaque dimension correspond à un terme et sa valeur correspond à la valeur TF/IDF de ce terme.

[39] propose une méthode ***hybride*** pour comparer des chaînes de caractères longues, qui découpe ces deux chaînes en plusieurs chaînes plus courtes. Ensuite, ces dernières sont comparées par une mesure de distance quelconque citée ci-dessus. Enfin, les résultats obtenus sont combinés.

Le Tableau suivant (synthèse de Cohen [40]) résume des domaines d'applications pour des mesures présentées dans cette partie.

Mesure de Similarité	Domaine d'application
N-Gram	Bigrams (n = 2) est efficace avec des erreurs typographiques mineures
Distance d'édition	Peut être appliquée aux entités ayant une longueur variable. Pour atteindre une exactitude raisonnable, les coûts des opérations de modification dépendent de chaque domaine
Distance de Hamming	Utilisée principalement pour les entités numériques ayant des tailles fixes, comme les codes postaux ou les numéros de sécurité sociale
Distance de Monge-Elkan	La meilleure performance au niveau des résultats dans plusieurs expériences. Peut être employée dans plusieurs domaines
Distance de Jaro/Jaro-Winkler	Presque même performance au niveau des résultats que Monge-Elkan mais beaucoup plus rapide
Distance basée sur TF/IDF	La meilleure pour la comparaison des textes longs (basée sur des tokens)

Tableau 1:Critères principaux d'utilisation des mesures de la similarité

(2) Les méthodes linguistiques :

La similarité entre deux entités représentées par des termes peut aussi être déduite en les analysants à l'aide des méthodes linguistiques. Les méthodes linguistiques permettent de déterminer la similarité entre deux entités. Ces entités sont représentées par des termes (ou mots). Ces méthodes syntaxiques prennent en charge les propriétés expressives et productives du langage naturel qui peuvent être intrinsèques ou extrinsèques [35].

(a) Les méthodes intrinsèques :

Les informations intrinsèques sont des propriétés linguistiques internes des termes, telles que des propriétés morphologiques ou syntaxiques. Un même concept (ou entité) peut être décrit par plusieurs termes (synonymie) ou par plusieurs variantes d'un même terme. Les méthodes intrinsèques cherchent la forme canonique ou représentative d'un mot ou d'un terme (lemme) en exploitant ses variantes linguistiques (lexème). La similarité entre deux termes est mesurée en comparant leurs lemmes. La recherche du lemme d'un mot peut être effectuée à l'aide d'un dictionnaire. Une approche automatique utilise les stemmers. Le stemmer est un algorithme qui permet de déterminer la forme radicale d'un terme. Cette forme est déduite à partir d'une forme infléchie ou dérivée d'un mot donné. Les radicaux trouvés par les stemmers n'ont pas besoin d'être identiques à la racine morphologique du mot. Il suffit que les mots similaires soient associés à un même radical, même si ce radical n'est pas une racine de mot valide. La lemmatisation est une approche qui détermine le radical exact d'un mot. Ce processus extrait la catégorie lexicologique d'un mot et applique des règles de normalisation différentes pour chaque partie du discours. Cette approche nécessite la connaissance de la grammaire ainsi que les différentes règles d'un langage.

(b) Les méthodes extrinsèques :

Les informations extrinsèques exploitent des ressources externes telles que des dictionnaires ou des vocabulaires. Ces méthodes calculent la valeur de similarité entre deux termes en employant des ressources externes. Ces ressources regroupent les dictionnaires, les lexiques ou les vocabulaires. La similarité entre deux termes est calculée en exploitant les liens sémantiques existants dans ces ressources externes. Ces liens regroupent les synonymes (pour l'équivalence), des liens d'hyponymes ou d'hyperonymes (pour la subsomption). Typiquement, l'API WordNet [41] est un système lexicologique, qui a été employé pour chercher des relations telles que la synonymie entre les termes. WordNet est aussi exploité pour calculer la distance sémantique entre les termes. Les ressources externes utilisées dans les méthodes extrinsèques peuvent aussi être des vocabulaires ou des dictionnaires multilingues, ou d'autres systèmes tels qu'EuroWordNet et Polylex.

b) Les méthodes structurelles :

Les méthodes structurelles se basent sur les informations structurelles pour calculer la similarité, vu que les ontologies peuvent être représentées avec des graphes ou bien des taxonomies qui rendent les liens sémantiques et syntaxiques entre les entités plus claires.

Ces méthodes s'intéressent aux algorithmes qui traitent les graphes aussi ils peuvent se baser sur les résultats des méthodes terminologiques et sémantiques vu que l'information structurelle est présente avec les liens sémantiques et syntaxiques.

Les méthodes structurelles se devisent en deux : les méthodes structurelles internes qui s'intéressent aux informations sur les attributs de l'entité, les méthodes structurelles externes qui traitent les relations et les liens entre les entités.

(1) Les méthodes structurelles internes :

Ces méthodes utilisent les informations des structures internes des entités qui sont des informations liées à l'entité comme les informations concernant les attributs de l'entité à savoir les informations du co-domaine des attributs, les informations de cardinalité des attributs aussi les informations des caractéristiques des attributs.

Dans ce cas pour ces méthodes qui se basent sur les caractéristiques des attributs c'est presque le cas du domaine des bases de données, il existe plusieurs méthodes pour calculer la similarité entre deux éléments de deux schémas de base de données, en se basant sur les contraintes à propos de ces éléments. Donc ces méthodes pour chercher la similarité entre deux éléments ils peuvent se baser sur les informations concernant les attributs, les types de données, des contraintes de co-domaine ou de valeur, l'autorisation des valeurs nulles…

(2) Les méthodes structurelles externes :

Les méthodes structurelles externes traitent la structure externe de l'entité, donc la structure de l'ontologie qui représente les relations entre les entités comme les relations de subsomption (is-a ou spécialisation). L'ensemble de ces relations nous donne tout la hiérarchie de l'ontologie qu'est souvent représenté par des graphes. La comparaison de similarité entre deux entités de deux ontologies peut être basée sur la position des entités dans leurs hiérarchies.

L'idée de base c'est que si deux entités de deux ontologies sont similaires, leurs voisins le sont également d'une certaine façon. On peut donc comparer la similarité des entités selon leurs positions dans la hiérarchie alors on aura des liens directs (voisinage direct où on a une relation qui lie les deux entités directement) et des liens indirects (voisinage indirect où les deux entités sont liées, mais pas d'une manière directe).

Pour considérer la similarité entre deux entités avec des liens directs, on a les 2 cas suivants :

- Leurs super-entités directes sont similaires.
- Leurs sous-entités directes sont déjà similaires.

Pour considérer la similarité entre deux entités avec des liens indirects, on a les cas suivants :

- Leurs sœurs (ou toutes leurs sœurs, qui sont les entités ayant la même super entité directe avec les entités en question) sont déjà similaires.
- Leurs descendants (entités dans le sous-arbre ayant pour racine l'entité en question) sont déjà similaires.

- Toutes (ou presque toutes) leurs feuilles (les entités de même type, qui n'ont aucune sous-entité, dans le sous-arbre ayant pour racine l'entité en question) sont déjà similaires.
- Toutes (ou presque toutes) les entités dans les chemins de la racine aux entités en question sont déjà similaires.

Il existe plusieurs mesures qui se servent de la structure hiérarchique de l'ontologie pour le calcul des distances pour enfin déterminer la similarité sémantique entre les entités.

➢ *Mesure Valtchev et Euzenat :*

C'est une mesure récursive de la dissimilarité topologique structurelle qui se base sur la distance du chemin le plus court dans un graphe.

➢ *Mesure Mädche et Staab :*

Cette mesure introduit la notion de « upward cotopy » qui est définie :

UC(c, H) = {c'∈ H ;c ≤ c'}, l'ensemble d'entités qui sont super-entités de l'entité c dans la hiérarchie H. La mesure de similarité, inspirée de la distance de Jaccard, est alors définie ainsi :

Définition 10 (Similarité de « upward cotopy »). Soit H1 et H2 deux hiérarchies. Soit e et e' deux entités dans H1 et H2 respectivement. La similarité de « upward cotopy » entre e et e' est une fonction de la similarité S_{cotopy} : O x O ➔ [0, 1] telle que :

$$\overline{S_{cotopy}}(e,e') = \frac{|UC(e,H_1) \cap UC(e',H_2)|}{|UC(e,H_1) \cup UC(e',H_2)|}$$

Avec le même principe, on peut déduire la similarité entre deux entités en utilisant les relations reliant ces deux entités. L'idée est que si l'on a deux entités similaires A et A', et si elles sont connectées par un même type de relation R avec deux autres entités B et B', alors on peut déduire que B et B' sont d'une façon ou d'une autre similaires. De même, si on sait que A et A' sont similaires, B et B' sont aussi similaires, alors les relations de A-B et de A'-B' peuvent être similaires. On peut généraliser cette idée pour un ensemble d'entités et de relations : si on a un ensemble de relations R1… Rn qui sont similaires avec un autre ensemble de relations R1'… Rn', alors les entités qui sont les domaines (ou les co-domaines) de ces relations sont considérées comme similaires.

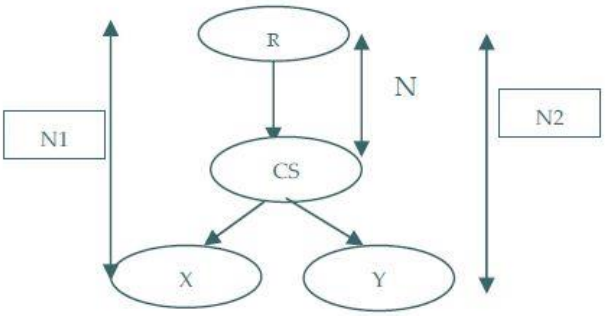
Toujours avec la mesure de Mädche et Staab mais cette fois-ci pour le calcul de la similarité entre des relations définie d'après ce principe :

Définition 11 (Similarité des relations). Soit H1 et H2 deux hiérarchies. Soit r et r' deux relations dans H1 et H2 respectivement. Soit dom(r) et ran(r) deux fonctions qui retournent le domaine et le co-domaine de la relation r respectivement. La similarité des relations entre r et r' est une fonction de la similarité $S_{relation}$: O x O ➔ [0, 1] telle que :

$$\overline{S_{relation}}(r,r') = \sqrt{\overline{S_{costopy}}(dom(r),dom(r')) * \overline{S_{costopy}}(ran(r),ran(r'))}$$

➢ *Mesure de Wu et Palmer :*

Cette mesure de similarité basée sur le principe suivant : Étant donnée une ontologie O formée par un ensemble de nœuds et un nœud racine R. Soit X et Y deux éléments de l'ontologie dont nous allons calculer la similarité. Le principe de calcul de similarité est basé sur les distances (N1 et N2) qui séparent les nœuds X et Y du nœud racine et la distance qui sépare le concept subsumant (CS) de X et de Y du nœud R. Comme c'est montré dans l'exemple suivant :



La mesure de Wu et Palmer est définie par la formule suivante :

$$Simc(X,Y) = \frac{2*N}{N1+N2}$$

On peut généraliser cette mesure pour pouvoir traiter plusieurs ontologies.

➢ *Mesure de Rada et al :*

Cette mesure est adoptée dans un réseau sémantique et elle est fondée sur le fait qu'on peut calculer la similarité en se basant sur les liens hiérarchiques « is-a ».

Pour calculer la similarité de deux concepts dans une ontologie, on doit calculer le nombre des arcs minimums qui les séparent. Cette mesure, basée sur le calcul de la distance entre les nœuds par le chemin le plus court, présente un moyen des plus évidents pour évaluer la similarité sémantique dans une ontologie hiérarchique.

Toutes ces mesures peuvent avoir des limites par exemple, la mesure de Wu et Palmer sa limite c'est qu'elle vise essentiellement à détecter la similarité entre deux concepts par rapport à leur distance de leur plus petit généralisant. Donc pour mieux calculer la similarité entre les entités on peut généraliser ces mesures ou bien les combiner pour dépasser ces limites. Mais on peut toujours rencontrer quelques difficultés dans les cas, où les hiérarchies des ontologies sont différentes au niveau de granularité.

c) Les méthodes extensionnelles :

Les méthodes extensionnelles se basent sur l'analyse des extensions des entités (concepts ou classes), donc pour déduire la similarité entre deux entités on va traiter leurs ensembles des instances qui représentent leurs extensions.

On aura deux cas vu qu'on va comparer les ensembles des instances, le 1er cas c'est où nous aurions une partie commune entre les deux ensembles, dans cette situation il existe des mesures qu'on peut appliquer pour qu'on puisse faire des opérations sur ces ensembles.

Il y a la mesure de Hamming adapté pour les mesures extensionnelles qui en général compte le nombre d'éléments différents entre deux ensembles à comparer.

Définition 12 (Distance de Hamming, version adaptée pour les ensembles des instances). Soit S et T deux ensembles. La distance de Hamming (appelée aussi la différence symétrique) entre S et T est une fonction de la dissimilarité $DS_{Hamming} : 2^E \times 2^E \rightarrow [0, 1]$ telle que :

$$\overline{DS_{\text{Hamming}}}(S,T)=\frac{|S\cup T-S\cap T|}{|S\cup T|}$$

Il y a la mesure de Jaccard adapté pour les mesures extensionnelles qui est en général le rapport entre l'intersection des ensembles à comparer et leur union.

Définition 13 (Distance de Jaccard, version adaptée pour les ensembles des instances). Soit S et T deux ensembles. Soit P(x) la probabilité d'une instance aléatoire être dans l'ensemble X. La distance de Jaccard est une fonction de la dissimilarité $DS_{Jaccard} : 2^E \times 2^E \rightarrow [0, 1]$ telle que :

$$\overline{DS_{\text{Jaccard}}}(S,T)=1-\frac{P(S\cap T)}{P(S\cup T)}$$

Dans le 2eme cas où les ensembles des instances ne partagent aucune partie commune, ces deux mesures ne sont plus applicables (le résultat retourné sera toujours égal à 1, c.-à-d. les entités à comparer sont toujours différentes). Parce que ces deux mesures se basent sur la comparaison exacte des éléments dans deux ensembles pour chercher la similarité entre les deux ensembles. Dans ce cas, on va utiliser la mesure de Valtchev.

La mesure de Valtchev calcule la similarité basée sur des correspondances (match-based similarity). Ici, la similarité entre deux ensembles est la similarité moyenne des paires des éléments dans Pairing, où Pairing est l'ensemble de correspondances ayant la somme maximale de toutes les similarités des paires dans l'ensemble. Le calcul de l'ensemble Pairing(S, T) est un problème d'optimisation qui maximise le total des similarités des paires des éléments de S et T.

Définition 14 (Similarité basée sur des correspondances). Soit S et T deux ensembles d'entités. Soit Sq(s, t) une mesure de similarité quelconque de deux entités s et t. Soit Pairing(S, T) l'ensemble de correspondances entre S et T ayant la somme maximale des valeurs de

similarité de ces correspondances. La similarité basée sur des correspondances entre deux ensembles S et T est une fonction de la similarité S_{Corr} : 2^E x 2^E ➔ [0, 1] telle que :

$$\overline{S_{\text{Corr}}}(S,T)=\frac{1}{max(|S|,|T|)}\sum_{(s,t)\in Pairing(S,T)}S_q(s,t)$$

Une autre mesure pour calculer la similarité entre deux ensembles emploie une technique d'analyse multidimensionnelle dans le domaine de la statistique. L'hypothèse de cette technique est que si deux entités ont les distances très similaires à toutes autres entités, elles doivent être très similaires. Les entités dans des ensembles sont représentées par des vecteurs, dont la valeur d'une dimension est la similarité de l'entité en question avec une autre entité dans les deux ensembles. La similarité entre deux ensembles est donc la similarité (la valeur du cosinus) de deux vecteurs moyens de ces deux ensembles.

Définition 15 (Similarité des ensembles). Soit S = {s_1, s_2,…} et T = {t_1, t_2,…} deux ensembles d'entités. Soit $S_q(s_i, t_j)$ une mesure de similarité quelconque de deux entités s et t.

Soit $\vec{S}_i$ = (sim(e_i,e_1), sim(e_i,e_2),… sim(e_i,f_1), sim(e_i,f_2),…) le vecteur représentant de l'entité e_i. La similarité des ensembles entre S et T est une fonction de la similarité S_{Set} : 2^E x 2^E ➔ [0, 1] telle que :

$$\overline{S_{\text{Set}}}(S,T)=\frac{\sum_{s\in S}\vec{s}}{\left|\sum_{s\in S}\vec{s}\right|}\otimes\frac{\sum_{t\in T}\vec{t}}{\left|\sum_{t\in T}\vec{t}\right|}$$

d) Les Méthodes Sémantiques :

Les méthodes sémantiques se basent principalement sur deux approches. La première approche repose sur les modèles de la logique, tandis que la deuxième approche regroupe les méthodes de déduction afin de déduire la similarité entre deux entités.

Les approches logiques sont la satisfiabilité propositionnelle (SAT), la SAT modale ou les logiques de descriptions. Giunchiglia et al. (2003)[42] et Bouquet et al. (2003) [43] respectivement, emploient des techniques issues de la satisfiabilité propositionnelle (SAT). Ces techniques permettent la vérification de la validité d'un ensemble de formules propositionnelles. Ce dernier est construit en traduisant des relations déjà connues et des relations à vérifier entre des entités vers des formules propositionnelles. [42] étendent les méthodes proposées vers le modèle de la SAT modale, qui peut aussi contenir des prédicats binaires.

Le premier modèle n'accepte que des prédicats unaires qui sont des entités comme des concepts ou des classes. Le second modèle permet de calculer en plus, des attributs ou des propriétés (slots). Il emploie des opérateurs de la logique modale. La validité de l'ensemble de formules en logique modale est aussi vérifiée en utilisant des procédures de recherche de

la satisfiabilité (SAT). Dans le cas où la validité est satisfaite, les relations hypothétiques entre des entités, qui sont des traductions de la requête sur la relation entre ces entités en logique modale, sont confirmées.

Les techniques des logiques de description (le test de subsomption) peuvent être employées. Elles permettent de vérifier les relations sémantiques entre les entités telles que l'équivalence (la similarité est égale à 1), la subsomption (la similarité est comprise entre 0 et 1) ou l'exclusion (la similarité est égale à 0). Elles assurent aussi la déduction de la similarité de deux entités.

4. Les méthodes d'alignement combinées :

Il existe plusieurs vues ou aspects sous lesquels une entité peut être observée ou considérée, son nom, ses attributs, ou sur ses relations avec d'autres entités. La similarité entre deux entités peut être calculée donc suivant plusieurs aspects, et on compare les caractéristiques d'une entité sur un aspect avec les caractéristiques correspondantes de l'autre entité en employant une des mesures de similarité de base qu'on a vues ci-dessus. Par conséquent, il faut un moyen pour combiner ces valeurs de similarité obtenues pour chaque aspect afin de produire une valeur de similarité unique pour chaque deux entités à comparer. Dans ce qui suit, on va présenter quelques approches qui existent dans la littérature.

a) La distance de Minkowski :

Définition 16 *(Distance de Minkowski). Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. La distance de Minkowski entre x et y est une fonction de la dissimilarité* $DS_{Minkowski}$ *: O x O ➔ R telle que :*

$$DS_{\text{Minkowski}}(x,y)=\sqrt[p]{\sum_{i=1}^{n} DS(x_i,y_i)^p}$$

La distance de Minkowski, si on veut dire est une mesure généralisée avec différentes valeurs de *p,* puisque suivant la valeur de p, on peut tomber dans d'autres mesures, par ex. pour p=1, elle devient égale à la distance de Manhattan, pour p=2 à la distance euclidienne, pour p=∞ elle devient la distance de Chebyshev.

Cette mesure n'est une fonction linéaire que quand p = 1 ou p =∞. Si p = 1, une variante de cette mesure avec des poids est souvent utilisée. L'avantage ici c'est que nous pouvons contrôler l'influence (ou l'importance) de chaque dimension sur la valeur finale de la distance. Les dimensions plus importantes seront associées avec les poids plus élevés, donc les valeurs de ces dimensions influenceront mieux à la valeur agrégée finale.

b) La somme Pondérée :

Définition 17 (*Somme pondérée). Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit wi le poids de la dimension i. Soit DS(xi, yi) la dissimilarité de la paire des objets à la dimension i. La somme pondérée entre x et y est une fonction de la dissimilarité* DS_{sp} *: O x O ➔ R telle que :*

$$DS_{sp}(x,y)=\sum_{i=1}^{n} w_i * DS(x_i,y_i)$$

En general, $\sum_{i=1}^{n} w_i = 1$ ➔ Version normalisé.

Une autre mesure similaire à la somme pondérée est le produit pondéré. Cependant, un inconvénient de cette mesure est que le résultat sera égal à 0 si une des dimensions est égale à 0.

c) Produit Pondéré :

Définition 18 (Produit pondéré). *Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit w_i le poids de la dimension i. Soit DS(xi, yi) la dissimilarité de la paire des objets à la dimension i. Le produit pondéré entre x et y est une fonction de la dissimilarité DSpp : O x O ➔ R telle que :*

$$DS_{sp}(x,y) = \prod_{i=1}^{n} DS(x_i,y_i)^{w_i}$$

Le schéma ci-dessous, résume les différentes méthodes qu'on a présentées avant :

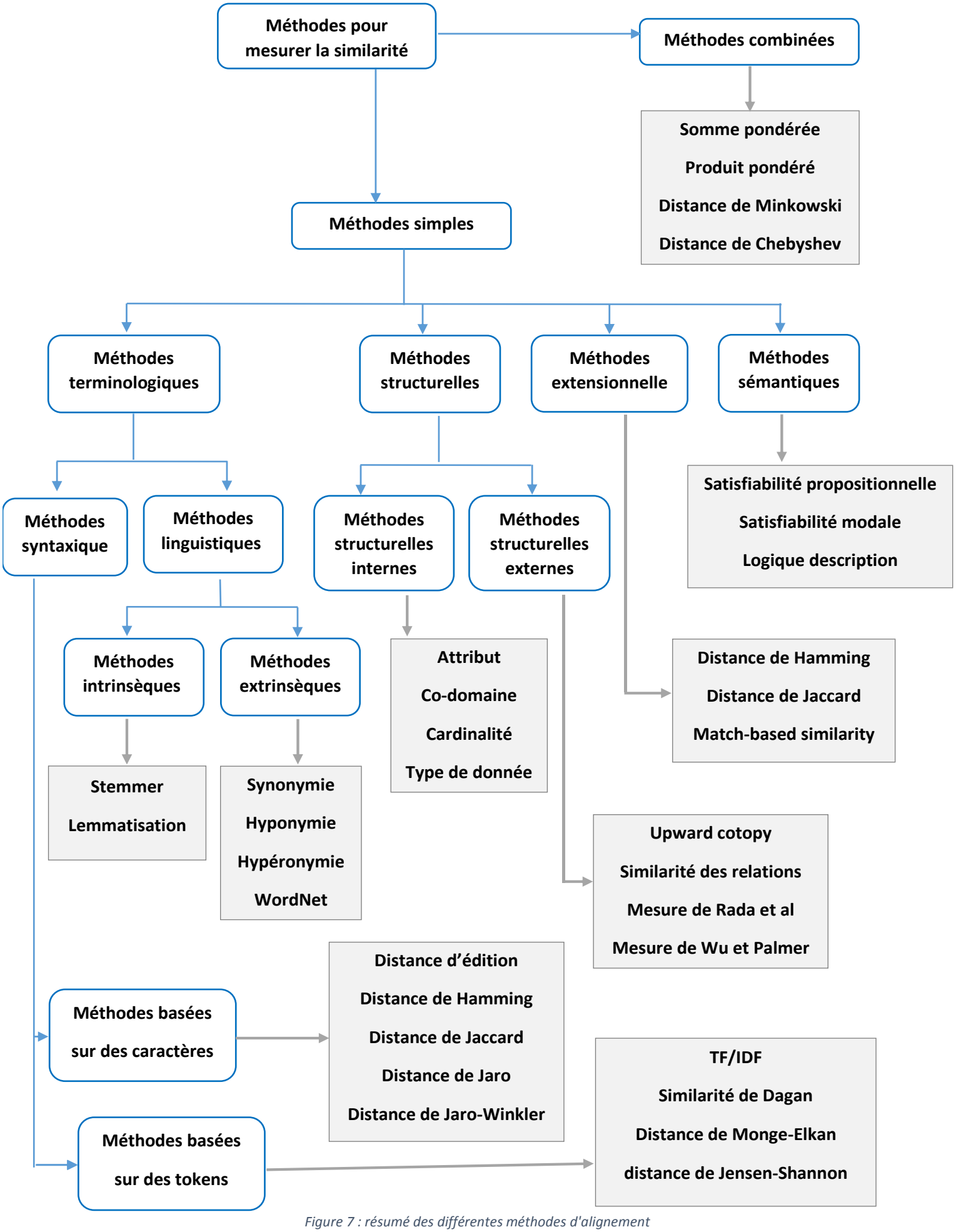


Figure 7 : résumé des différentes méthodes d'alignement

G. Quelques approches de l'alignement d'ontologies :

Pour aligner deux ontologies, on doit chercher des correspondances entre eux en utilisant les méthodes de calcul de similarité, comme nous avons vu. Il existe plusieurs méthodes et techniques pour calculer la similarité, aussi différentes façon pour combiner les résultats de similarité. Donc la diversité d'approches vient du fait que chaque approche utilise les méthodes adéquates selon les types d'ontologies à aligner. Nous allons voir un petit résumé sur quelques approches.

- *Une approche utilisant WordNet et une nouvelle mesure structurelle [44] :*

Cette technique commence par un prétraitement sur les termes de chaque ontologie, ce qui est important pour améliorer les résultats d'alignement, selon les réalisateurs de cette approche. Cette approche se compose de trois phases principales, la première est basée sur les chaînes de caractères plus particulièrement sur la mesure de Levenstein qui sert à déterminer les relations d'équivalence et l'heuristique d'inclusion des labels afin de déterminer les relations spécialisations entre les concepts. La deuxième phase se focalise sur l'exploitation d'une ressource externe, le WordNet, ainsi que l'utilisation de la synonymie, pour qu'on puisse déduire les relations d'équivalence entre les concepts dont les labels ne sont forcément pas similaires, syntaxiquement parlant. La dernière phase exploite les résultats obtenus dans les deux phases précédentes en utilisant une technique structurelle, et elle combine la structure des deux ontologies ainsi que WordNet pour trouver d'autres possibles alignements. Lors de la phase structurelle, pour générer des correspondances, on a proposé une nouvelle mesure structurelle entre deux ontologies qui n'est qu'une généralisation de la mesure de Wu & Palmer.

- *ASMOV [45], [46] :*

Asmov (Automated Semantic Mapping of Ontologies with Validation) comme son nom l'indique est un nouvel algorithme qui automatise le processus d'alignement ainsi que l'intégration d'un unique processus de validation sémantique afin de s'assurer que l'alignement obtenue ne contient pas d'incohérences sémantiques.

L'algorithme ASMOV calcule la similarité entre les entités pour une paire d'ontologies selon quatre caractéristiques : éléments lexicaux (id, label, commentaires), structure relationnelle (hiérarchie : ascendant-descendant), structure interne (restrictions des propriétés des concepts ; types, domaines, ranges ; data-values pour les individus), extension (les instances des classes et les valeurs des propriétés). Et enfin, les différentes mesures obtenues en comparants ces quatre caractéristiques sont fusionnées en une valeur unique utilisant une somme pondérée.

Pour le calcul de la similarité lexicale, ASMOV, utilise par défaut WordNet. Cependant, autres dictionnaires peuvent être mieux appropriés pour d'autres domaines ; par exemple, UMLS est mieux convenable pour les ontologies médicales que le WordNet. Par conséquent, ASMOV définit les interfaces qui permettent aux utilisateurs experts d'implémenter des

adaptateurs pour d'autres sources de connaissances. ASMOV inclut les adaptateurs WordNet et UMLS.

Et les résultats des différents benchmarks qui ont été réalisés montrent qu'ASMOV est un outil pratique pour les applications du monde réel qui demandent des alignements d'ontologies à la volée.

- *ROMIE [47]:*

ROMIE (Resource based Ontology Mapping within and Interactive and Extensible environment), est une approche qui regroupe les méthodes d'alignement syntaxiques, linguistiques, structurelles et sémantiques pour offrir un alignement semi-automatique.

Cette approche se divise en deux phases : la phase d'enrichissement sémantique et la phase d'alignement. La première est basée sur l'analyse des informations développées par les ontologies à comparer (ressources web, données, documents, …) et qui sont associées aux concepts de l'ontologie. A la fin de la phase d'enrichissement, une ontologie contient plus de relations sémantiques entre les concepts qui seront exploitées dans la deuxième phase. La phase d'alignement, prend deux ontologies et calcule la similarité entre les couples de chaque concept d'ontologies en employant plusieurs méthodes syntaxiques qui calculent par exemple la distance d'édition entre deux concepts, et/ou encore des méthodes linguistiques basées sur le dictionnaire WordNet, par exemple. Puis selon les valeurs de similarité retournées, on génère un ensemble d'alignements candidats et en utilisant des méthodes de filtrage, on élimine les alignements les moins pertinents. Ces méthodes sont basées sur des relations structurelles et sémantiques entre les concepts. En fin l'ensemble des hypothèses d'alignements fournis par l'étape précédente est ordonné selon les valeurs de similarité.

- *SODA : un système d'alignement basé sur OWL-DL [48] :*

SODA (Strucural Ontlology OWL-DL Alignement), est une approche qui aligne deux ontologies OWL-DL (Ontology Web Language Description Logics) et utilisant des mesures de similarité. Les deux ontologies OWL-DL sont transformées en graphes DL-GRAPH correspondants qui décrivent toutes les informations dans les ontologies. L'approche SODA, utilise le DL-GRAPH pour aligner les deux ontologies. Elle opère en deux étapes successives, la première calcule la similarité locale (similarités linguistiques et structurelles), alors que la deuxième calcule la similarité sémantique.

Le calcul de la similarité locale est elle aussi divisée en deux phases. La première permet le calcul de la similarité linguistique pour chaque couple du nœud de la même catégorie. La seconde phase permet de calculer la similarité structurelle en employant la structure des voisins des nœuds à aligner.

La similarité globale est une mesure de similarité agrégée des deux similarités linguistique et structurelle.

Les travaux réalisés sur cette approche ont montré que la méthode SODA n'est pas encore très bien adaptée pour les ontologies de grande taille. La prise en considération des ontologies réelles permet de rendre la méthode exploitable dans la confrontation des points de vue multiples.

- *QOM [49] :*

L'approche QOM (Quick Ontology Matching) est basée sur l'approche NOM (Naive Ontology Matching) cette dernière a montré qu'elle est effective, mais inefficace, donc on a pensé à l'optimiser, la chose qui a donné naissance à l'approche QOM. Cette approche comme son prédécesseur exploite RDF-triples, et pour mesurer la similarité entre les ontologies à aligner elle utilise la similarité terminologique et structurelle. Et, après la phase de l'analyse de la similarité et ses interprétations, de nouvelles décisions doivent être prises, comme : quels alignements candidats doit-on ajouter à l'agenda pour l'itération suivante.

Cette approche a montré de très bons résultats en termes de temps d'exécution (n.log (n) au lieu de n^2, n étant le nombre d'entités dans les ontologies) en comparaison avec d'autres approches de la même classe de complexité.

- *Anchor-PROMPT [62] :*

Anchor-PROMPT est un algorithme qui permet de trouver des concepts qui sont sémantiquement similaires dans des ontologies différentes, cela en se basant sur : une liste des paires d'ancres comme entrée de l'algorithme, ces paires d'ancres sont les paires de concepts similaires définis par les utilisateurs ou trouvés en utilisant des méthodes de correspondance lexicologique. Aussi sur le graphe qui représente l'ontologie, où les nœuds sont des concepts (classes) et les arcs représentent les relations (slots) entre les concepts, cet algorithme fait une différence entre les relations de type «is-a » et les autres relations car les classes liées par « is-a » constituent des groupes d'équivalences qui sont considérés comme un seul nœud. L'algorithme analyse les chemins dans les sous-graphes limités par les ancres pour déterminer quels concepts apparaissent fréquemment en positions similaires sur les chemins similaires. Ces classes sont susceptibles de représenter les concepts qui sont sémantiquement similaires. Les résultats de cet algorithme sont liés à la taille de la liste des paires d'ancre initiale et la longueur maximale du chemin. Cet algorithme trouve des problèmes si les structures des ontologies sont différentes car il ne trouvera pas les chemins similaires avec la même longueur et si les noms de relations sont différents.

- *GLUE [63] :*

GLUE est un système qui se base sur les techniques d'apprentissage automatique pour trouver les correspondances. Étant donné deux ontologies, pour chaque concept dans une ontologie GLUE trouve le concept le plus proche dans la deuxième ontologie. GLUE peut travailler avec plusieurs mesures de similarité. GLUE peut utiliser plusieurs modules d'apprentissage (learners), chacun exploite un type d'information différent soit dans les données d'instances ou dans la structure de la hiérarchie des ontologies, ces résultats sont

combinés par un méta module de mise en correspondance en utilisant la somme pondérée. Pour améliorer encore ces résultats GLUE utilise la technique d'optimisation de contraintes « relaxation labelling ».

- *S-Match [64] :*

S-Match est un algorithme et un système qui cherche les correspondances sémantiques où il cherche ces correspondances en calculant les relations sémantiques entre les concepts assignés aux nœuds et non pas les étiquettes dans les graphes des concepts, pour trouver des relations d'équivalence, over lapping, différence. Cet algorithme utilise SAT (le moteur de la satisfiabilité propositionnelle) pour déduire les rapports à partir les informations sémantiques implicitement ou explicitement codifiées dans les nœuds et les arcs. Utilise aussi WordNet pour chercher le sens des lemmes qui représentent les étiquettes normalisées.

- *COMA [65] :*

COMA est un système développé pour mieux aider les utilisateurs pour faire correspondre les schémas complexes et pour combiner les algorithmes de correspondance d'une manière flexible. Ce système fournit une bibliothèque extensible des algorithmes de correspondance (matchers) avec la possibilité d'ajouter de nouveaux algorithmes à cette bibliothèque et prend en charge les différentes méthodes pour combiner les résultats des algorithmes de correspondance. COMA reste flexible avec les problèmes de correspondance particuliers où il peut adapter les choix des algorithmes et les méthodes de combinaison pour résoudre le problème. COMA aussi utilisé pour évaluer et comparer l'efficacité des différents algorithmes de correspondance et stratégies de combinaison.

- *OLA [66] :*

OLA pour OWL-Lite Alignement est un algorithme pour aligner des ontologies représentées en OWL-Lite. Cet algorithme utilise les différentes méthodes de calcul de similarité pour trouver les correspondances entre les entités de deux ontologies en se basant sur leurs caractéristiques et leurs rapports avec d'autres entités et pour combiner ces valeurs de similarités calculées pour chaque paire d'entités il utilise la somme pondérée des valeurs de similarité de chaque caractéristique.

Les approches	Méthodes terminologiques	Méthodes structurelles	Méthodes extensionnelles	Méthodes sémantiques	Méthodes combinées
Anchor-PROMPT	String-comparaison des classes/relations	comparaison des positions			
GLUE	tokenisation, String-comparaison pour des tokens				somme pondérée
S-Match	tokenisation, lemmatisation, WordNet			logique propositionnelle SAT	
COMA	comparaison des affixes, tokens, trigrammes, synonymes, types, editDistance, soundex	comparaison des statistiques structurelles			somme pondérée, max, moyenne
OLA	termes comparaison des URIs, étiquettes, noms comparaison des types	comparaison des nœuds voisins			somme pondérée
ASMOV	comparaison lexicale des étiquettes, descriptions, tokenisation, WordNet,	comparaison des nœuds parents/enfants. Comparaison des types, domaines, ranges. Mesure de Wu et Palmer	comparaison des instances classes/propriétés		somme pondérée
ROMIE	WordNet, distance (Hamming, subString, N-gramme, Levenshtein ...)	comparaison des nœuds voisins			somme pondérée
SODA	Jaro-Winkler pour les noms et labels, Monge-Elkan pour les commentaires	comparaison des nœuds voisins			somme pondérée
QOM	String-comparaison des URIs et étiquettes	comparaison des ensembles des voisines directes des concepts / relations	comparaison des ensembles d'instances concepts / relations		Somme pondérée, Fonction de Sigmoid

Tableau 2 : quelques techniques et méthodes utilisé par les approches d'alignement d'ontologie

H. Quelques outils utilisés dans l'alignement d'ontologies :

Comme nous avons vu dans le passage qui précède, il y en a plusieurs approches ayant pour but d'aligner des ontologies. Ces approches-là peuvent se baser sur des outils et techniques, comme des dictionnaires, des outils de visualisation des résultats, des bases de données, etc., ce qui rend la tâche des chercheurs et développeurs de ce domaine un peu plus facile. Dans ce qui suit, nous allons essayer d'en énumérer quelques-unes de ces outils qui représentent un moyen, soit de visualiser les résultats d'alignement obtenu, soit pour mesurer la similarité terminologique, extensionnelle, structurelle ou bien sémantique, qui est une étape indispensable dans le domaine de l'alignement des ontologies, etc.

- *WordNet [50], [51] :*

WordNet est une large base de données lexicale dans son noyau, qui est utilisée dans plusieurs techniques de l'alignement des ontologies. Cela renforce et améliore l'alignement syntaxique ou celui basé sur la comparaison des chaînes de caractères entre les labels des entités ainsi que la capacité d'aligner des mots qui peuvent être des synonymes, hyperonymies, et d'autre sens lexical. Les algorithmes d'alignement utilisent l'outil WordNet dû à l'amélioration potentielle aux termes de rappel des alignements.

Historiquement, le premier WordNet est le PrincetonWordNet (PWN), développé pour l'anglais à la Princeton University. Au fil du temps, il est devenu une des ressources les plus utiles pour de nombreuses applications de compréhension et d'interprétation automatique des langues, telles que la désambiguïsation sémantique, l'extraction d'informations, la traduction automatique, la classification de documents et le résumé de textes. Ceci a provoqué le développement de WordNets pour de nombreuses autres langues.

- *UMLS [52], [53]:*

Le UMLS ou bien Unified Medical Language System est un Meta-Thésaurus est actuellement l'effort le plus compréhensif dans le domaine de l'intégration des ontologies et dictionnaires médicale, et inclut le Thésaurus du National Cancer Institute (NCI), le FMA (Foundational Model of Anatomy) et les Termes SNOMED CT (Systematized Nomenclature Of Medicine Clinical Terms), qui sont des ontologies volumineuses et sémantiquement riches.

Le UMLS peut être utilisé pour l'amélioration ou le développement de plusieurs applications, des records électronique de la santé, outils de classification, dictionnaires, la récupération des moteurs de recherche, datamining, les statistiques de la santé publique de rapports et la recherche terminologique.

Le UMLS, lui aussi comprend trois sous-outils qu'on peut appeler des Sources de Connaissances (Knowledge Sources) :

- Meta-Thésaurus : Les termes et codes de différents vocabulaires (CPT, ICD-10-CM, LOINC®, MeSH®, RxNorm, and SNOMED CT).

- Réseau Sémantique : Grandes catégories (de types sémantique) et leurs relations (sémantiques).
- SPECIALIST Lexicon et Outils lexicaux : outils de traitement du langage naturel.

Et on utilise le Réseau Sémantique et les Outils Lexicaux pour produire le Méta-Thésaurus.

- *SAT [54], [55]:*

Cette approche décompose le problème de mise en correspondance de graphes ou d'arbres en un ensemble de problèmes de mise en correspondance de nœuds (Bouquet 2003). Ensuite chaque paire de nœuds des deux structures à faire correspondre est traduite dans une formule propositionnelle et leur validité testée. Une formule propositionnelle est valide si et seulement si sa négation est insatisfiable. L'insatisfiabilité est vérifiée en utilisant des procédures de recherche de satisfiabilité (solver) SAT. Le système S-Match (Giunchiglia, Shvaiko et Yatskevich 2004) utilise la satisfiabilité propositionnelle pour trouver des correspondances entre ontologies (cf. section 1.8). Donc, elle tend à trouver les alignements en transformant deux ontologies ainsi que la requête d'alignement (le couple d'entités à apparier et la relation possible entre ces entités) en des formules propositionnelles et par la suite de vérifier sa validité.

La limite de cette approche est qu'elle n'utilise que des prédicats unaires, donc des concepts.

- *MeSH [67] :*

MeSH (Medical Subject Heading) est la Bibliothèque nationale du thésaurus vocabulaire contrôlé de médecine à l'origine destiné à l'index des articles scientifiques pour l'Index Medicus et pour la base de données Medline. L'équipe CISMeF à essayer de développer MeSH dans le but de l'adapter dans le domaine des ressources Internet de santé, après ce développement de MeSH thesaurus il peut être utilisé dans la recherche d'informations, la catégorisation et l'indexation automatique. L'ancienne version contient 24,767 descripteurs et 83 qualificatifs qui décrivent le sujet de la ressource. Il y a aussi plus de 97.000 synonymes qui aident à trouver les rubriques MeSH les plus appropriées. Après l'amélioration de MeSH l'équipe CISMeF a ajouté plus de 10.000 synonymes français.

- *Alviz [68] :*

Alviz est un outil pour visualiser l'alignement d'ontologies, les outils existants d'alignement d'ontologies fournissent des résultats comme les listes de correspondances qui rendent la tâche d'analyse très difficile car les utilisateurs doivent analyser chaque paire de correspondances dans ces listes. D'où le besoin d'utiliser les techniques de visualisation pour faciliter la compréhension des résultats de l'alignement par l'utilisateur. Pour rendre les types de similarité entre les entités explicites, Alviz utilise la notion de multi-vue qu'est une combinaison des différentes techniques de visualisation et d'affichage des vues. L'outil Alviz est composé de deux types de vue, J-Trees pour représenter la structure hiérarchique par des

arbres, mais pour la représentation des grandes ou complexes ontologies J-Trees ne fournit pas l'aperçu adéquat parce que les arbres deviennent encombré. Small world graphs qu'utilise des clusters pour regrouper les nœuds d'un graphe en fonction du niveau de détail sélectionné, les nœuds représentent les entités (concepts ou instances) liées les uns aux autres selon les relations sélectionnées.

Conclusion :

Dans ce chapitre, on a introduit la notion de l'alignement des ontologies. On a vu qu'un alignement est défini comme un ensemble de correspondances entre des entités de différentes ontologies classiques, ces entités peuvent être : des concepts, des relations ou des individus. Ainsi que ses objectifs et les difficultés et les défis que connaît le domaine de l'alignement d'ontologies.

On a mentionné que pour trouver les correspondances entre ontologies, les méthodes d'alignement sont utilisées, et exploitées dans des systèmes d'alignement. Ces méthodes sont appliquées pour calculer la similarité entre concepts, relations ou individus. Et ces méthodes-là sont classées dans 4 catégories : terminologique, structurelle, extensionnelle et sémantique. Pour une vue globale, nous avons fourni un schéma montrant les relations entre les différentes méthodes présentées.

Puis, on a essayé de présenter brièvement quelques approches qui existent dans la littérature et qui offrent ce service d'alignement d'ontologies, ainsi que les mesures de similarité que ces approches emploient, ainsi qu'un tableau comparatif de ces approches. Enfin, on a cité quelques outils utilisés non seulement pour améliorer les performances des systèmes d'alignement des ontologies, mais aussi de faciliter leur tâche de réalisation.

Chapitre III : Algorithme d’alignement d’ontologies ASCO3

Introduction

Dans ce chapitre, nous présentons l'algorithme d'alignement d'ontologies ASCO 3. Cet algorithme traite les ontologies représentées en OWL (langage d'ontologie recommandé par W3C). L'algorithme ASCO 3 analyse les rapports sémantiques entre les entités des deux ontologies à aligner, pour trouver les entités correspondantes entre ces deux ontologies.

Nous allons voir dans la suite de ce chapitre, le fonctionnement de cet algorithme et ses différentes étapes, à savoir la création des graphes O-Graphe, la création du graphe d'association et la recherche de la clique maximale.

I. L'algorithme implémenté et pourquoi ?

Comme nous avons vu dans le chapitre précédent, il y a plusieurs approches d'alignement d'ontologies, qui utilisent différentes méthodes et techniques de calcul de similarité. Tant qu'une approche utilise plusieurs méthodes de calcul de similarité plus elle traite les données de plusieurs aspects (terminologique, structurels…).

Alors pour avoir un taux élevé de correspondances correctes il faut traiter le maximum d'informations possible. De là nous avons choisi d'implémenter l'approche ASCO3, qu'utilise plusieurs méthodes de calcul de similarité et traite des ontologies représentées en OWL. On verra dans ce chapitre le fonctionnement de cette approche et l'algorithme en détail.

II. L'algorithme ASCO3 :

L'algorithme ASCO3 était présenté par Bach Thanh Le et Rose Dieng-Kuntz dans l'article « ***A Graph-Based Algorithm for Alignment of OWL Ontologies*** » [79]. C'est un algorithme qui cherche les correspondances entre les entités de deux ontologies représentées en OWL. OWL permet de décrire ces entités avec ses primitives qu'ont déjà une sémantique définie.

Dans cet algorithme, une ontologie est considérée comme un réseau sémantique où les nœuds représentent les entités et les liens sont les primitives de OWL. Pour que deux ontologies qui modélisent le même domaine soient similaires, c'est clair que leurs réseaux sémantiques devraient être aussi similaires. Pareil pour les nœuds se trouvant aux places correspondantes dans les parties similaires des deux réseaux sémantiques, sont considères similaires. Le but de cet algorithme est de trouver les plus grandes parties communes des deux réseaux sémantiques, puis décider si les nœuds se trouvant dans les positions correspondantes des parties trouvées sont similaires ou non.

A. Fonctionnement de l'algorithme :

L'algorithme ASCO3 prend les deux fichiers OWL ontologieA.owl et ontologieB.owl comme données initiales (avec OntologieA.owl et OntologieB.owl sont deux fichiers OWL, chaque fichier contient la description de l'ontologie, à savoir les définitions des entités et les liens entre les entités, en utilisant les primitives de OWL). Pour les transformer en deux O-Graphes (Ontologie Graphe) qui représentent les deux réseaux sémantiques des deux ontologies.

Puis pour chercher les plus grandes parties similaires des deux réseaux sémantiques, il utilise le graphe d'association crée à partir des deux O-graphes pour chercher les cliques maximales (les parties communes les plus grandes des deux réseaux sémantiques). Enfin extraire les entités similaires de la meilleure clique maximale.

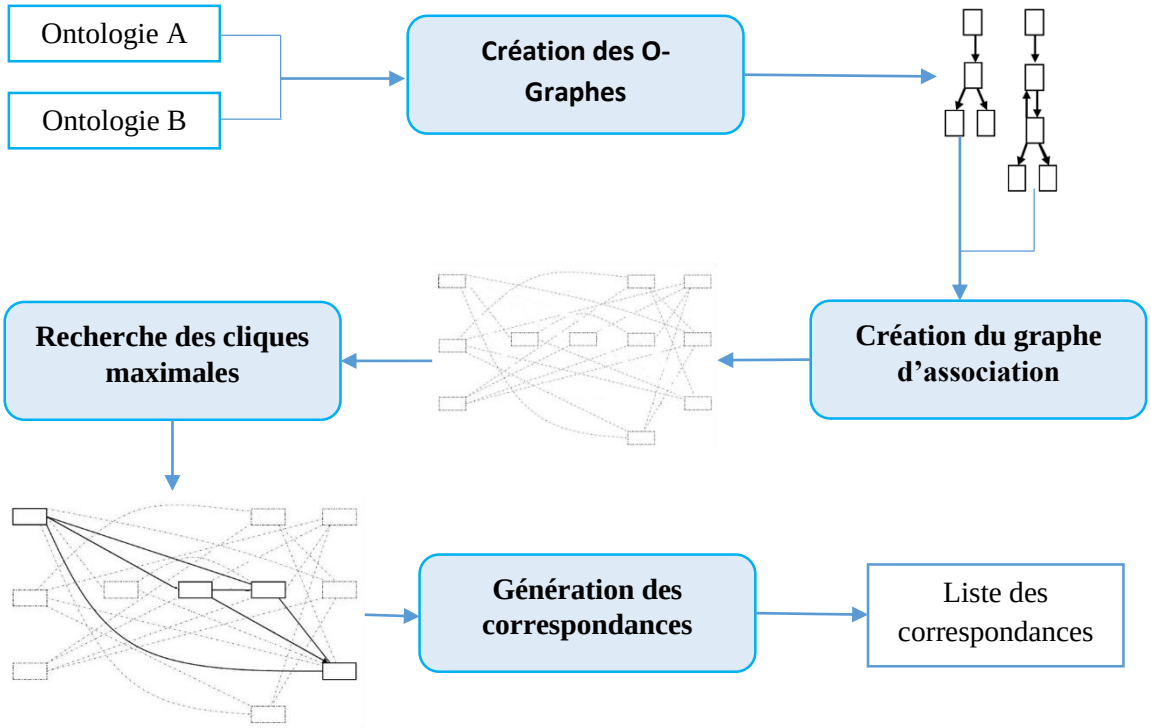


Figure 8 : les étapes de l'algorithme ASCO3

B. L'algorithme en détail :

1. Création des O-Graphes :

Dans l'algorithme ASCO3 l'O-Graphe permet de représenter le réseau sémantique de l'ontologie et c'est un graphe étiqueté, orienté et cyclique, où les nœuds du graphe représentent les entités de l'ontologie (classes, relations, instances), ces nœuds sont liés avec des arcs orientés et étiquetés représentant les primitives de OWL.

Définition 19 (O-Graphe) Un O-Graphe est un 4-uplet og = (V, E, α, β), où :

- V est l'ensemble fini de sommets (également appelés les noeuds)
- E $\subseteq$ V $\times$ V est l'ensemble d'arcs
- α : V → L est une fonction affectant une étiquette à chaque sommet
- β : E → L est une fonction affectant une étiquette (type) à chaque arc

Rappelons qu'une ontologie OWL est un document RDF valide, et que les descriptions des entités en OWL sont interprétables sous forme des triplets RDF. Nous classons les propriétés (les primitives) de OWL en deux catégories : P_NE et P_Entité.

Dans cet algorithme, on ne va pas utiliser les triplets RDF dont les prédicats appartiennent à la catégorie P_NE, pour construire le graphe O-Graphe. Car P_NE se compose des propriétés dont le type de co-domaine n'est pas une entité (classe, propriété et instance), et dans le graphe O-Graphe on a besoin des entités.

P_Entité se compose des 19 propriétés de OWL (dans OWL on a 33 propriétés). Les descriptions des entités sous forme de triplets RDF ayant leur prédicat appartenant à la catégorie P_Entité sont exploitées pour construire le graphe O-Graphe.

Code	Propriété de OWL	Domaine	Co-domaine
1	rdf:type	rdfs:Resource	rdfs:Class
2	rdfs:domain	rdf:Property	rdfs:Class
3	rdfs:range	rdf:Property	rdfs:Class
4	rdfs:subClassOf	rdfs:Class	rdfs:Class
5	rdfs:subPropertyOf	rdf:Property	rdf:Property
6	owl:equivalentClass	owl:Class	owl:Class
7	owl:equivalentProperty	rdf:Property	rdf:Property
8	owl:sameAs	owl:Thing	owl:Thing
9	owl:complementOf	owl:Class	owl:Class
10	owl:inverseOf	owl:ObjectProperty	owl:ObjectProperty
11	owl:differentFrom	owl:Thing	owl:Thing
12	owl:disjointWith	owl:Class	owl:Class
13	owl:onProperty	owl:Restriction	rdf:Property
14	owl:allValuesFrom	owl:Restriction	owl:Class
15	owl:someValuesFrom	owl:Restriction	owl:Class
16	owl:distinctMembers	owl:AllDifferent	rdf:List
17	owl:intersectionOf	owl:Class	rdf:List
18	owl:unionOf	owl:Class	rdf:List
19	owl:oneOf	owl:Class	rdf:List

Tableau 3 : Les propriétés de OWL dans la catégorie P_Entité

Nous allons utiliser l'algorithme 1 pour construire le graphe O-Graphe qui représente l'ontologie à aligner. L'entrée de cet algorithme est l'ensemble de triplets RDF de l'ontologie, qu'étaient extraits à partir du fichier OWL de l'ontologie. Dans l'étape d'extraction, on va retourner que les triplets RDF des propriétés appartenant à la catégorie P_Entité. Ensuite on va parcourir la liste des triplets retournée, et pour chaque triplet on va vérifier si les deux sommets : sujet et objet, n'existent pas dans l'O-Graphe, pour les ajouter on utilisant la fonction Ajouter_Sommet. Un arc orienté de sommet du sujet au sommet d'objet, et étiqueté avec le prédicat du triplet, est également ajouté dans le graphe. Si le prédicat du triplet est une des primitives owl:equivalentClass, owl:equivalentProperty, owl:sameAs, un arc inverse de l'objet vers le sujet est ajouté avec la même étiquette.

Dans le cas où on a une liste d'objets au lieu d'un seul objet, comme c'est le cas pour les primitives owl:distinctMembers, owl:intersectionOf, owl:unionOf, owl:oneOf. On ajoute un sommet anonyme (classe anonyme), liée avec le sujet par la relation owl:equivalentClass. Et entre les objets de la liste et le sommet anonyme, on ajoute les arcs des relations owl:distinctMembers, owl:intersectionOf, owl:unionOf, owl:oneOf.

Algorithme 1 Construire_O-Graphe(Ontologie)

O-Graphe ← initialiser le graphe

Pour chaque triplet t de l'ontologie ontologie

Si prédicat(t) ∈ P_Entité

Si sujet(t) n'existe pas dans O-Graphe

Ajouter_Sommet(O-Graphe, sujet(t))

Fin Si

Si prédicat(t) ∈ { owl:distinctMembers, owl:intersectionOf,owl:unionOf, owl:oneOf }

liste ← Prendre_Liste_Objets(t)

sommet_anonyme ← créer une classe anonyme

Ajouter_ sommet (O-Graphe, sommet _anonyme)

Ajouter_Arc(O-Graphe, owl:equivalentClass, sujet(t), sommet_anonyme)

Ajouter_Arc(O-Graphe, owl:equivalentClass, sommet_anonyme, sujet(t))

Pour chaque objet o dans liste

Si o n'existe pas dans O-Graphe

Ajouter_Sommet(O-Graphe, o)

Fin Si

Ajouter_Arc(O-Graphe, prédicat(t), sommet_anonyme, o)

Fin Pour

Sinon

Ajouter_ Sommet(O-Graphe, objet(t))

Ajouter_Arc(O-Graphe, prédicat(t), sujet(t), objet(t))

Fin Si

Si prédicat(t) ∈ { owl:equivalentClass, owl:equivalentProperty, owl:sameAs }

Ajouter_Arc(O-Graphe, prédicat(t), objet(t), sujet(t))

Fin Si

Fin Si

Fin Pour

Retourner O-Graphe

Figure 9 : Algorithme 1 : Construction d'O-Graphe

2. Le sous-graphe commun maximal :

La recherche des parties communes les plus grandes des deux réseaux sémantiques représentant les deux ontologies correspond donc à la recherche des plus grands sous-graphes communs de deux graphes O-Graphe. Ce problème appartient à la famille des problèmes de recherche de sous-graphe commun maximal. Les entités correspondantes de deux ontologies sont donc déduites des nœuds des deux graphes O-Graphe qui correspondent aux nœuds du sous-graphe commun maximal.

Définition 20 (Sous-graphe commun et Sous-graphe commun maximal) Soit deux O-graphes, $og_1 = (V_1, E_1, \alpha_1, \beta_1)$ et $og_2 = (V_2, E_2, \alpha_2, \beta_2)$. Un sous-graphe commun de og_1 et og_2, nommé $ogc(g_1,g_2)$, est un graphe $og = (V, E, \alpha, \beta)$ tel qu'il existe un isomorphisme de sous-graphe de og à og_1 et un isomorphisme de sous-graphe de og à og_2. Nous appelons og un sous-graphe commun maximal de og_1 et og_2, nommé $ogcm(g_1, g_2)$, s'il n'existe aucun autre sous-graphe commun de og_1 et og_2 qui ait plus de nœuds que og.

Le problème de recherche de sous-graphe commun maximal est un problème NP-complet [69], ne peut être résolu qu'avec des méthodes de force brute qui utilisent la recherche exhaustive pour vérifier tous les cas possibles. Il existe deux grandes approches qui traitent ce type de problème : l'approche de retour en arrière (backtracking) [70] et l'approche de détection de la clique maximale [71], [72]. Parfois, il est possible de réduire la complexité de calcul avec des heuristiques, mais dans le pire des cas, la complexité de ce type de problème est de O(N !) où N est le nombre des nœuds dans le graphe.

Dans cette approche on va chercher le sous-graphe commun maximal en utilisant l'approche de détection de la clique maximale. Parce qu'elle est plus efficace pour les graphes ayant une forte densité des arcs(les graphes de grandes tailles). Une clique est un graphe connexe où tous ses sommets sont deux à deux adjacents (ils sont tous connectés l'un à l'autre). La recherche de la clique maximale est exécutée sur le graphe d'association, qui y est construit à partir des deux graphes orignaux (les deux graphes O-Graphe dans notre cas). Les nœuds du graphe d'association correspondent aux paires de nœuds des deux graphes O-Graphe. Les arcs du graphe d'association représentent la compatibilité des paires de nœuds (i.e. si des arcs de même type connectent les nœuds de deux paires dans deux graphes originaux ou pas).

Le graphe d'association permet d'accélérer la performance de l'algorithme ASCO3, en effectuant les calculs de compatibilité des paires de nœuds des deux O-Graphes une seule fois et les stockées. Cette compatibilité est déduite à partir de plusieurs informations : le type de ces nœuds dans les O-Graphes (classe, propriété, instance), les relations avec les nœuds voisins dans les O-Graphes (les types des arcs dans les O-Graphes), les voisins directs et indirects dans les O-Graphes (leurs types, leurs positions dans les O-Graphes). La construction du graphe d'association prend en compte les différences entre les arcs des graphes O-Graphes et les sémantiques des arcs spécifiées par les primitives de OWL.

3. Construction du graphe d'association :

Le but qui réside derrière la construction du graphe d'association est d'encoder les informations de compatibilité entre les nœuds des graphes O-graphes des deux ontologies à aligner en une seule entité : l'arc du graphe d'association. D'autant plus que ça, en utilisant le graphe d'association, on réduit le temps de calcul requis par l'algorithme chaque fois qu'il veuille chercher une paire de nœuds à ajouter dans le sous-graphe commun maximal.

Dans le graphe d'association GA, construit à partir de deux O-graphes og_1 et og_2, un nœud de GA noté $[s_1, s_2]$ est la combinaison des deux nœuds s_1 d'og_1, et s_2 d'og_2. Un arc est construit entre deux nœuds $[s_1, s_2]$ et $[t_1, t_2]$ Si : (a) La relation entre s_1 et t_1 dans og_1 et celle entre s_2 et t_2 sont compatible ou (b) s_1 et t_1 ne sont pas adjacents dans og_1, et ni le sont s_2 et t_2 dans og_2.

Définition 21 : (Graphe d'association) Soit deux graphes de types O-Graphe $og1= (V_1, E_1, \alpha_1, \beta_1)$ et $og_2= (V_2, E_2, \alpha_2, \beta_2)$. Le graphe d'association de og_1 et og_2, nommé $ga(og_1,og_2)$ est un graphe $ga=(V, E, \alpha, \beta)$ où :

- $V = \{[s_1, s_2] \mid s_1 \in V_1, s_2 \in V_2, s_1$ et s_2 sont n-compatibles$\}$
- $E \subseteq V \times V$. $E = \{([s_1, s_2], [t_1, t_2]) \mid (s_1, t_1) \in E_1, (s_2, t_2) \in E_2, (s_1, t_1)$ et (s_2, t_2) sont p-compatibles ; ou $(s_1, t_1) \notin E_1 \wedge (s_2, t_2) \notin E_2\}$
- $\alpha : V \rightarrow L$ est une fonction affectant une étiquette à chaque sommet
- β *: E ➔ L est une fonction affectant une étiquette à chaque arc*

La n-compatibilité, on peut la définir comme suit : deux nœuds s_1 de og_1 et s_2 de og_2 sont n-compatible, si les entités représentées par ces nœuds ont le même type i.e. Ces entités sont soit deux classe (leur type est soit `rdfs : Class`, soit `owl : Class`, ou bien `owl : DepricatedClass`) ou deux restrictions (leur type est `owl:Restriction`) ou deux relations (leur type est un sous-type de `rdf : Property`), ou bien deux instances (leur type est une classe définie par l'utilisateur).

Et en ce qui concerne les primitives, on dit que deux primitives p_1 et p_2 sont p-compatible si et seulement si :

- Elles sont la même primitive c.-à-d. $p_1 = p_2$, ou bien
- $P_1,p_2 \in$ {`owl :allValuesFrom, owl : someValuesFrom`}, ou
- $P_1,p_2 \in$ {`owl:cardinality, owl:minCardinality, owl:maxCardinality`}

La fonction $p\_comp(p_1, p_2)$ suivante est utilisée pour affecter une étiquette commune à un arc du graphe d'association, qui est créé à partir de deux triplets ayant p_1 et p_2 comme leurs prédicats (voir l'Algorithme 2 – construction de graphe d'association).

Définition 22 *(p_comp) Soit deux primitives de OWL, p_1 et p_2. P_1 et p_2 sont p-compatibles. P_comp est une fonction renvoyant une primitive représentante pour p_1 et p_2 :*

- *$p\_comp(p_1, p_2) = p_1$, si $p_1 = p_2$,*
- *$p\_comp(p_1, p_2) = p_1$, si $p_1, p_2 \in$ {`owl:allValueFrom, owl:someValueFrom`}*

- *p_comp(p_1, p_2) = p_1, Si p_1, p_2 ∈{`owl:cardinality`, `owl:minCardinality`, `owl:maxCardinality`}*

Le graphe d'association est construit par l'algorithme (2), qui prend comme paramètres d'entrées deux O-Graphes construits à l'aide de l'algorithme (1). Chaque paire de deux arcs, un du premier O-Graphe, l'autre du deuxième O-Graphe, est traitée pour concevoir deux nœuds et un arc du graphe d'association. Si les étiquettes des deux arcs (les primitives OWL) sont p-compatible est les nœuds des deux extrémités des deux arcs sont n-compatibles respectivement, donc un arc et deux nœuds sont construits dans le graphe d'association GA. Si les nœuds qu'on vient de créer ne se trouvent pas déjà dans GA, *AjouterSommet* va les ajouter, sinon il récupère le nœud existant.

L'algorithme ASCO3 utilise ce qu'on appelle « *saut* », lorsque la compatibilité entre les nœuds s_1 et s_2 peut être étendu via les links sémantiques OWL, comme : `rdf:type`, `rdfs:domain`, `rdfs:range`, `rdfs:subClassOf`, `owl:equivalentProperty` et `owl:sameAs`. Ces sauts nous permettent de déduire les mappings des nœuds ayant des liens directs ainsi qu'indirects. Et un « *saut* » peut être limité à un rayon de *r* arcs. Par exemple, dans la figure 8, on a la représentation en O-Graphe de deux ontologies OWL. La sémantique de la primitive de OWL, `owl:equivalentClass`, est prise en compte. Le graphe d'association GA comprend les deux arcs : [1, A]-P_0-[2, B] et [1, A]-P_0-[3, B] car le nœud 2 représente une classe équivalente à la classe représentée par le nœud 3 dans l'ontologie O_1. Notons que dans O_1, le nœud 1 est considéré comme adjacent avec le nœud 3, où il a un lien direct avec le nœud 3 (du fait de la primitive `owl:equivalentClass`) ; mais le nœud 1 et le nœud 4 ne sont pas adjacents. Il existe donc les arcs [1, A]-l_n-[4, C], [1, A]-l_n-[4, D], [1, A]-l_n-[5, C], [1, A]-l_n-[5, D] dans le graphe d'association GA. L'arc l_n dans GA indique une relation non adjacente entre des nœuds dans le O-Graphe.

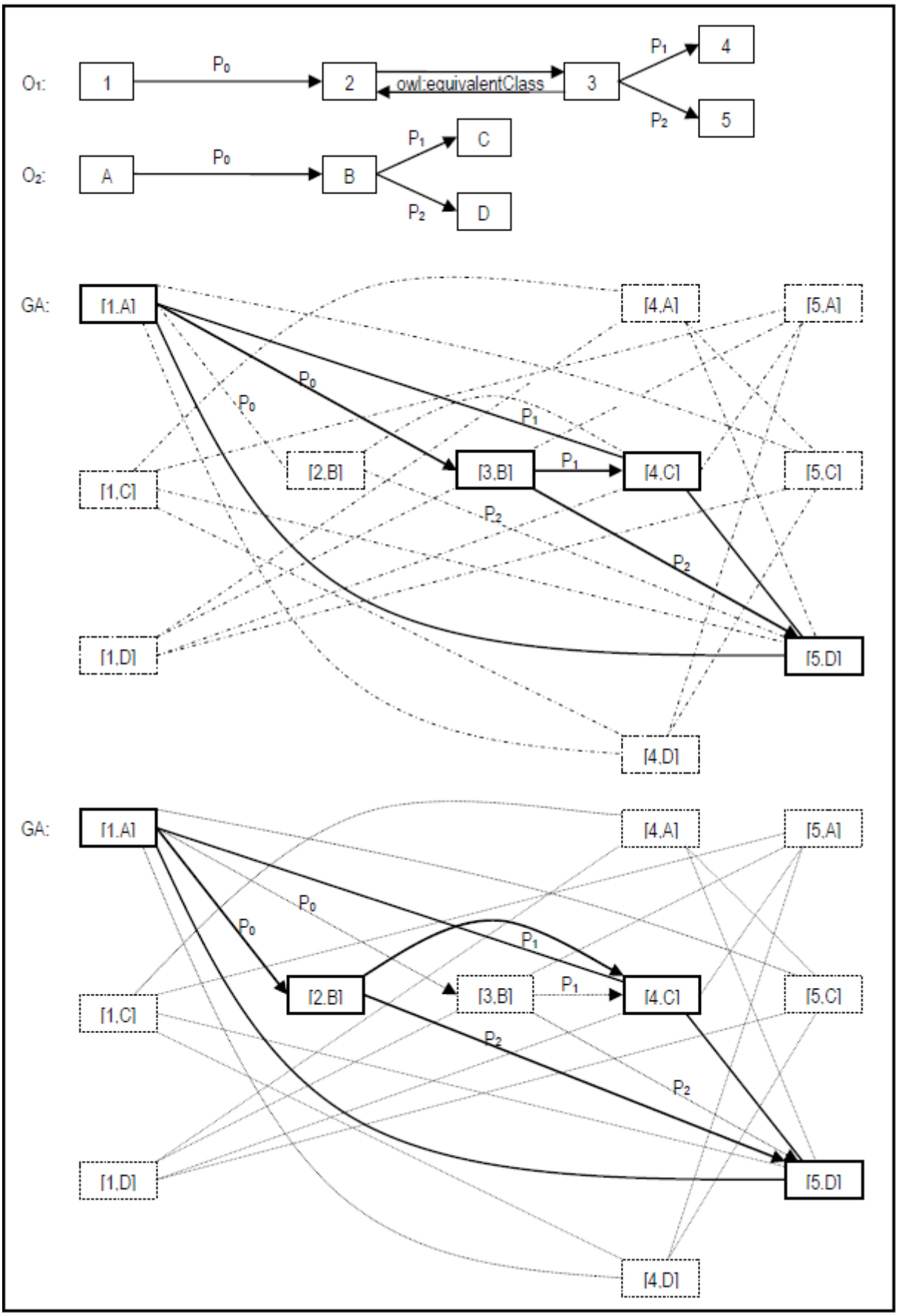


Figure 10 : Exemple de deux ontologies simples, avec leurs représentations en O-Graphe, et leur graphe d'association, et deux cliques maximales

Avant de définir ce que c'est que « saut » dans les différents cas des primitives OWL, nous introduisons la notion « Parents d'une entité », de « Nœuds ancestraux et descendants de la i-ème génération », de « Nœuds ancestraux et descendants dans un rayon » dans les définitions suivantes :

Définition 23 (Parents d'une entité) Soit e une entité (une classe ou une propriété) de l'ontologie O. Les parents de l'entité e, Parents(e), sont les entités de l'ontologie O ayant le même type que e et ayant le lien de subsomption directe avec e :

- *Parents(e) = { f | f est une classe, e est sous-classe directe de la classe f}, si e est une classe*
- *Parents(e) = { f | f est une propriété, e est sous-propriété directe de la propriété f}, si e est une propriété*

Autrement dit, f $\in$ Parents(e) si et seulement s'il existe un triplet soit (e rdfs:subClassOf f) si e est une classe, soit (e rdfs:subPropertyOf f) si e est une propriété, dans l'ensemble de triplets définissant l'ontologie.

Définition 24 *(Nœuds ancestraux de la i-ème génération) Soit n(e) le nœud dans le O-Graphe og représentant une entité e (une classe ou une propriété) de l'ontologie O. Les nœuds ancestraux de la i-ème génération Ai du nœud n(e), i ≥0, sont définies récursivement comme suit :*

- $A^i(n(e)) = \{ n(f) \mid f$ *est une entité de même type que e,* $\exists n(g) \in A^{i-1} \wedge f \in Parents(g)\}$
- $A^1(n(e)) = \{ n(f) \mid f \in Parents(e) \}$
- $A^0(n(e)) = \{ n(e) \}$

Les nœuds descendants de la i-ème génération sont définis par analogie.

Définition 25 *(Nœuds descendants de la i-ème génération) Soit n(e) le nœud dans le O-Graphe og représentant une entité e (une classe ou une propriété) de l'ontologie O. Les nœuds descendants de la i-ème génération Di du nœud n(e), i [0, sont définies récursivement comme suit :*

- $D^{i+1}(n(e)) = \{ n(f) \mid f$ *est une entité de même type que e,* $\exists n(g) \in D^i \wedge g \in Parents(f)\}$
- $D^1(n(e)) = \{ n(f) \mid e \in Parents(f) \}$
- $D^0(n(e)) = \{ n(e) \}$

Définition 26 *(Nœuds ancestraux dans un rayon) Soit n(e) le nœud dans le O-Graphe og représentant une entité e (une classe ou une propriété) de l'ontologie O. Les nœuds ancestraux dans le rayon de r,* A_r*, r ≥ 0, du nœud n(e) sont définis comme suit :*

- $A_r(n(e)) = \bigcup_{i=0\ldots r} A^i(n(e))$

Les nœuds descendants dans un rayon, D_n, sont définis par analogie.

Définition 27 *(Nœuds descendants dans un rayon) Soit n(e) le nœud dans le O-Graphe og représentant une entité e (une classe ou une propriété) de l'ontologie O. Les nœuds descendants dans le rayon de r, Dr, r ≥ 0, du nœud n(e) sont définis comme suit :*

- $D_r(n(e)) = \bigcup_{i=0\ldots r} D^i(n(e))$

Les cas dans lesquels on peut avoir le « saut » :

➢ ***Premier cas :*** avec `rdf:type`

Si deux nœuds représentent des instances, ils deviennent un nœud dans le graphe d'association, ce nœud va être directement connecté aux nœuds représentant les nœuds des classes et les super-classes de ces instances.

Définition 28 (Le saut avec la primitive `rdf:type`) Soit (i_1 `rdf:type` c_1) et (i_2 `rdf:type` c_2) deux triplets dans deux ensembles de triplets définissant deux ontologies O_1 et O_2, respectivement. Les nœuds et les arcs suivants sont ajoutés dans le graphe d'association GA de deux O-Graphes représentant deux ontologies O_1 et O_2, en prenant en compte la notion de « saut dans un rayon de r », avec la primitive `rdf:type` :

- *Nœuds à ajouter dans GA :*
  - *[n(i_1), n(i_2)]*
  - *[n(c_1), n2], n2 ∈ A_r (n (c_2))*
  - *[n1, n (c_2)], n1 ∈ A_r (n (c_1))*
- *Arcs à ajouter dans GA :*
  - *[n(i_1), n(i_2)]-`rdf:type`-[n(c_1), n2] , n2 ∈ A_r(n(c_2))*
  - *[n(i_1), n(i_2)]-`rdf:type`-[n1, n(c_2)], n1 ∈ A_r(n(c_1))*

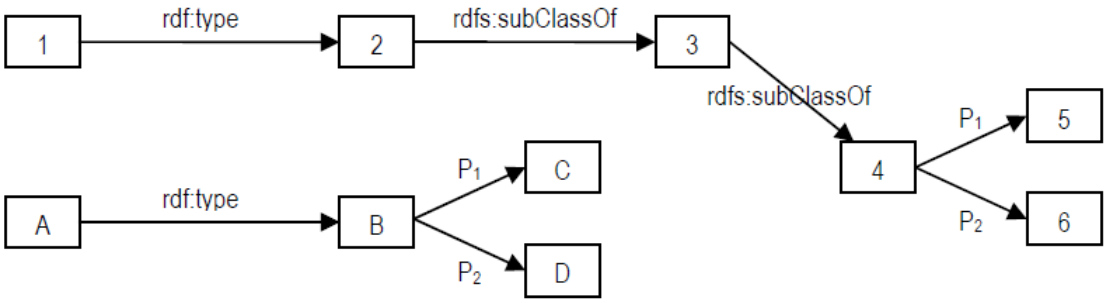


Figure 11 : Le "saut" avec la primitive rdf:type

Dans cet exemple, pour le rayon de 0, nous avons un seul arc dans le graphe d'association : [1, A]-`rdf:type`-[2, B].

Pour le rayon de 1, nous avons deux arcs suivants dans le graphe d'association : [1, A]-`rdf:type`-[2, B] et [1, A]-`rdf:type`-[3, B].

Pour le rayon de 2, nous avons trois arcs suivants dans le graphe d'association : [1,A]-`rdf:type`-[2,B] , [1,A]-`rdf:type`-[3,B] et [1,A]-`rdf:type`-[4,B].

Le « saut » avec le rayon de 2 dans cet exemple permet de trouver une meilleure solution de mettre en correspondance des entités. Grâce à la sémantique bien définie de la primitive `rdfs:subClassOf`, si 1-A est une bonne correspondance, il est possible de déduire non seulement 2-B est une correspondance probable, mais aussi 3-B ou 4-B.

- ***Deuxième cas :*** avec `rdfs:domain` et `rdfs:range`.

Le « saut » est possible pour les sous-classes de la classe qui est l'objet du triplet ayant `rdfs:domain` ou `rdfs:range` comme son prédicat (voir Figure 12).

Définition 29 *(Le saut avec les primitives `rdfs:domain` et `rdfs:range`) Soit p_{dr} une primitive de OWL, p_{dr} est soit `rdfs:domain`, soit `rdfs:range`. Soit (p_1 p_{dr} d_1) et (p_2 p_{dr} d_2) deux triplets dans deux ensembles de triplets définissant deux ontologies O_1 et O_2, respectivement. Les nœuds et les arcs suivants sont ajoutés dans le graphe d'association GA de deux O-Graphes représentant deux ontologies O_1 et O_2, en prenant en compte la notion de « saut dans un rayon de r », avec la primitive p_{dr} :*

- *Nœuds à ajouter dans GA :*
  - *[n (p_1), n (p_2)]*
  - *[n (d_1), n2], n2 $\in D_r$ (n (d_2))*
  - *[n1, n (d_2)], n1 $\in D_r$ (n(d_1))*
- *Arcs à ajouter dans GA :*
  - *[n (p_1), n (p_2)]- p_{dr} -[n (d_1), n2], n2 $\in D_r$ (n (d_2))*
  - *[n (p_1), n (p_2)]- p_{dr} -[n1, n (d_2)], n1 $\in D_r$ (n (d_1))*

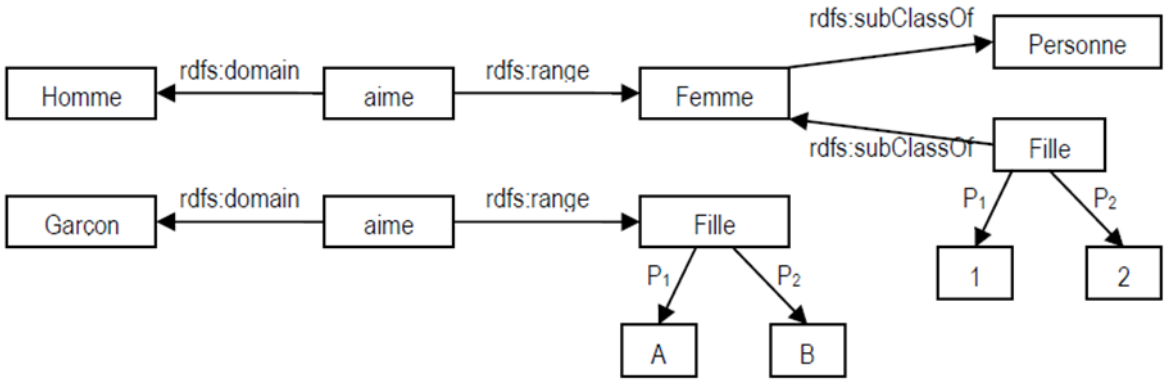


Figure 12 : Le "saut" avec la primitive rdfs:range

Dans cet exemple, pour le rayon de saut descendant de 1, nous avons les arcs suivants dans le graphe d'association : [aime1, aime2]-`rdfs:range`-[Femme1, Fille2], [aime1, aime2] `rdfs:range`-[Fille1, Fille2].

Notons qu'au contraire du saut ancestral dans le cas de la primitive `rdf:type`, où la déduction suivante est correcte : « si i est une instance de la classe c, i est aussi une instance des classes d_k, d_k est une classe ancestrale de la classe c ($d_k \in A_r(n(c))$) », le saut pour les cas avec les primitives `rdfs:domain` et `rdfs:range` est le saut descendant : « si le domaine

d'une propriété p est la classe c, les classes d_k, $d_k \in D_r(n(c))$, sont aussi domaine de cette propriété p ».

- ***Troisième cas :*** avec `rdfs:subClassOf` et `rdfs:subPropertyOf`.

Le « saut » est possible pour les super-entités (super-classes, super-propriétés) de l'entité qui est l'objet du triplet ayant `rdfs:subClassOf` ou `rdfs:subPropertyOf` comme son prédicat (voir Figure 13).

Définition 30 *(Le saut avec les primitives rdfs:subClassOf et `rdfs:subPropertyOf`) Soit p_{cp} une primitive de OWL, p_{cp} est soit `rdfs:subClassOf`, soit `rdfs:subPropertyOf`. Soit (c1 p_{cp} d1) et (c2 p_{cp} d2) deux triplets dans deux ensembles de triplets définissant deux ontologies O1 et O2, respectivement. Les nœuds et les arcs suivants sont ajoutés dans le graphe d'association GA de deux O-Graphes représentant deux ontologies O1 et O2, en prenant en compte la notion de « saut dans un rayon de r », avec la primitive p_{cp} :*

- *Nœuds à ajouter dans GA :*
  - *[n (c_1), n (c_2)]*
  - *[n (d_1), n2], n2 $\in$ Ar (n (d_2))*
  - *[n1, n (d_2)], n1 $\in A_r$ (n (d_1))*
- *Arcs à ajouter dans GA :*
  - *[n (c_1), n (c_2)]- p_{cp} -[n (d_1), n2], n2 $\in A_r$ (n (d_2))*
  - *[n (c_1), n (c_2)]- p_{cp} -[n1, n (d_2)], n1 $\in A_r$ (n (d_1))*

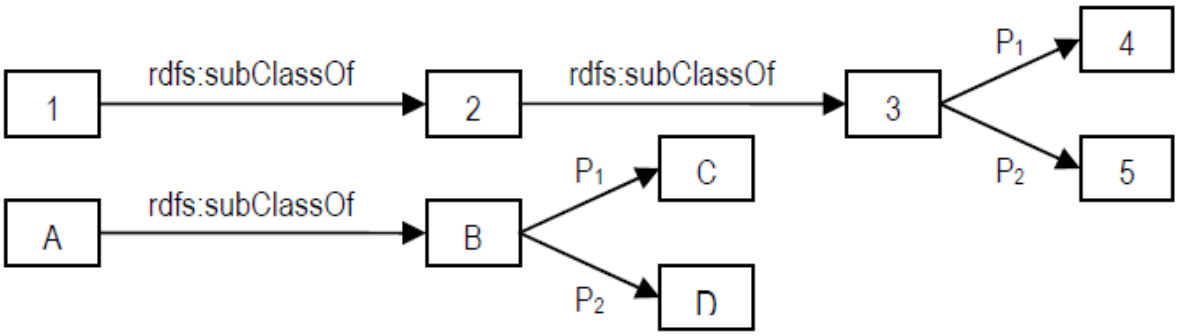


Figure 13 : Le saut avec la primitive rdfs:subClassOf

Dans cet exemple, nous avons les arcs suivants dans le graphe d'association : [1, A]-`rdfs:subClassOf`-[2, B], [1, A]-`rdfs:subClassOf`-[3, B].

- ***Quatrième cas :*** avec `owl:equivalentClass`, `owl:equivalentProperty` et owl:sameAs.

Pour tous les triplets, le « saut » (sans rayon) est possible pour les entités équivalentes avec les entités qui sont le sujet et l'objet du triplet (voir Figure 14).

Définition 31 *(Équivalence des entités) Soit e et f deux entités (classe, propriété ou instance) de l'ontologie O. e et f sont e-équivalentes si et seulement si un des cas suivants est correct :*

- *e et f sont les instances et il existe un triplet soit (e* `owl:sameAs` *f), soit (f* `owl:sameAs` *e) dans l'ontologie O.*
- *e et f sont les classes et il existe un triplet soit (e* `owl:equivalentClass` *f), soit (f* `owl:equivalentClass` *e) dans l'ontologie O.*
- *e et f sont les propriétés et il existe un triplet soit (e* `owl:equivalentProperty` *f), soit (f* `owl:equivalentProperty` *e) dans l'ontologie O.*

Notons qu'une entité e est e-équivalente avec elle-même.

Définition 32 *(Nœuds équivalents) Soit n(e) le nœud dans l'O-Graphe og représentant une entité e (une classe, une propriété ou une instance) de l'ontologie O. Les nœuds équivalents, NE, du nœud n(e) sont définies comme suit :*

- *NE (n(e)) = {n(f) | f est e-équivalente avec e}*

Notons que n(e) $\in$ NE (n(e)).

Définition 33 *(Le saut avec l'équivalence des entités) Soit p_e une primitive de OWL. Soit (e1 p_e f1) et (e2 p_e f2) deux triplets dans deux ensembles de triplets définissant deux ontologies O1 et O2, respectivement. Les nœuds et les arcs suivants sont ajoutés dans le graphe d'association GA de deux O-Graphes représentant deux ontologies O1 et O2, en prenant en compte la notion de « saut», avec l'équivalence des entités :*

- *Nœuds à ajouter dans GA :*
  - *[n1, n2], n1 $\in$ NE ($n(e_1)$), n2 $\in$ NE($n(e_2)$)*
  - *[m1, m2], m1 $\in$ NE ($n(f_1)$), m2 $\in$ NE ($n(f_2)$)*
- *Arcs à ajouter dans GA :*
  - *[n1, n2]- p_e -[m1, m2], n1 $\in$ NE ($n(e_1)$), n2 $\in$ NE ($n(e_2)$), m1 $\in$ NE ($n(f_1)$), m2 $\in$ NE ($n(f_2)$)*

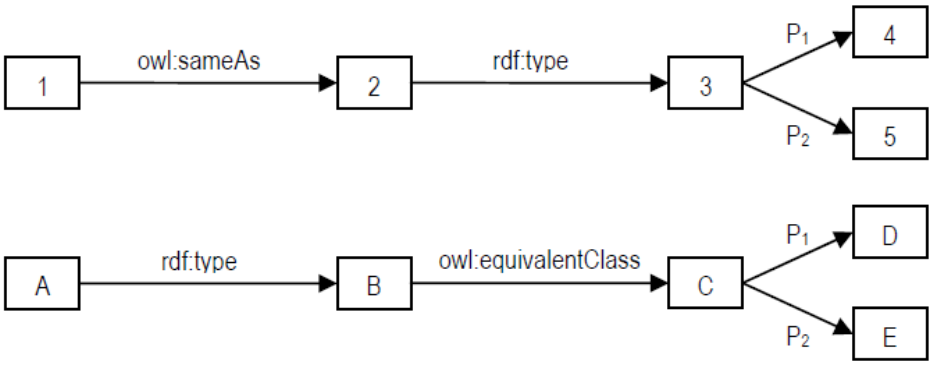


Figure 14 : Le saut avec l'équivalence des entités

Dans cet exemple, nous avons les arcs suivants dans le graphe d'association : [1,A]-`rdf:type`-[3,B], [1,A]-`rdf:type`-[3,C], [2,A]-`rdf:type`-[3,B], [2,A]-`rdf:type`-[3,C].

Le « saut » est limité dans un rayon de r arcs, pour la raison de la sémantique du lien de subsomption. En général, plus r est élevé, plus la différence entre l'entité e et les entités dans la r-ème génération de e ($A_r(e)$ ou $D_r(e)$) est importante. Il est donc nécessaire de limiter r pour que les significations des entités dans un rayon de r de l'entité e soient assez proches. La valeur raisonnable de r dépend de la modélisation des ontologies d'entrée, et devient deux paramètres d'entrée de l'Algorithme de la construction du graphe d'association : `rSuper` et `rSous` pour le saut ancestral et le saut descendant, respectivement.

Notons que les arcs dans le graphe d'association représentent la compatibilité entre des nœuds dans les O-Graphes. Donc, ils sont des arcs non orientés.

Notons que les arcs dans le graphe d'association représentent la compatibilité entre des nœuds dans les O-Graphes. Donc, ils sont des arcs non orientés.

Algorithme 2 : Construction Graphe_association (og1, og2, rSuper, rSous)

```
GA <- initialiser le graphe d'association
Pour chaque arc arc1 dans le O-Graphe og1
Pour chaque arc arc2 dans le O-Graphe og2
    Si étiquette(arc1) et étiquette(arc2) sont p-compatibles et  n_gauche(arc1) et n_gauche(arc2) sont n-compatibles et n_droite(arc1) et
    n_droite(arc2) sont n-compatibles
        EE1_gauche <- équi_entités(n_gauche(arc1))
        EE1_droite <- équi_entités(n_droite(arc1))
        EE2_gauche <- équi_entités(n_gauche(arc2))
        EE2_droite <- équi_entités(n_droite (arc2))
        Pour chaque paire d'entités [e1g, e2g] dans EE1_gauche x EE2_gauche
          Ajouter_Noeud(GA, [e1g, e2g])
         Pour chaque paire d'entités [e1d, e2d] dans EE1_droite x EE2_droite
           Ajouter_Noeud(GA, [e1d, e2d])
           Ajouter_Arc(GA, p_comp(arc1, arc2),[e1g, e2g], [e1d,e2d])
         Fin Pour
 Fin Pour
      Si (étiquette(arc1) est rdfs:subClassOf ou rdfs:subPropertyOf)ou  (étiquette(arc1) est rdf:type et  n_gauche(arc1) est une instance et
n_gauche(arc2) est une instance)
         SEE1 <- super_et_équi_entités(n_droite (arc1), rSuper)
         Pour chaque paire d'entités [e1g, e2g] dans  EE1_gauche x EE2_gauche
            Pour chaque paire d'entités [e1d, e2d] dans SEE1 x {n_droite(arc2)}
             Ajouter_Noeud(GA, [e1d, e2d])
             Ajouter_Arc(GA, p_comp(arc1, arc2),[e1g, e2g],[e1d,e2d])
            Fin Pour
         Fin Pour
         SEE2 <- super_et_équi_classes(n_droite(arc2), rSuper)
         Pour chaque paire d'entités [e1g, e2g] dans EE1_gauche x EE2_gauche
            Pour chaque paire d'entités [e1d, e2d] dans {n_droite(arc1)} x SEE2
               Ajouter_Noeud(GA, [e1d, e2d])
               Ajouter_Arc(GA, p_comp(arc1, arc2), [e1g, e2g], [e1d, e2d])
            Fin Pour
 Fin Pour
      Fin Si
      Si étiquette (arc1) est rdfs:domain ou rdfs:range  // étiquette(arc2) doit exactement être étiquette(arc1)
        sEC1 <- sous_et_équi_classes (n_droite(arc1), rSous)
        Pour chaque paire d'entités [e1g, e2g] dans EE1_gauche x EE2_gauche // e1g, e2g sont des propriétés
           Pour chaque paire d'entités [e1d, e2d] dans sEC1 x {n_droite(arc2)}  // e1d, e2d sont des classes
              Ajouter_Noeud(GA, [e1d, e2d])
              Ajouter_Arc(GA, p_comp(arc1, arc2), [e1g, e2g], [e1d, e2d])
           Fin Pour
 Fin Pour
      sEC2 <- sous_et_équi_classes(n_droite(arc2), rSous)
        Pour chaque paire d'entités [e1g, e2g] dans EE1_gauche x EE2_gauche
           Pour chaque paire d'entités [e1d, e2d] dans {n_droite (arc1)} x sEC2
              Ajouter_Noeud(GA, [e1d, e2d])
              Ajouter_Arc(GA, p_comp(arc1, arc2), [e1g, e2g], [e1d, e2d])
           Fin Pour
 Fin Pour
      Fin Si
 Fin Si
Fin Pour
// Créer des arcs In du graphe d'association, qui indiquent une relation non-adjacent entre des nœuds //dans O-Graphe
Pour chaque nœud n1 dans le O-Graphe og1
   NN1 <- l'ensemble de nœuds dans og1 qui ne sont pas adjacents à n1
   Pour chaque nœud n2 dans le O-Graphe og2
      NN2 <- l'ensemble de nœuds dans og2 qui ne sont pas adjacents à n2
         Si n1 et n2 sont n-compatibles
           Pour chaque nœud nn1 dans NN1
            Pour chaque nœud  nn2 dans NN2
             Si nn1 et nn2 sont n-compatibles
                Ajouter_Noeud(GA, [n1, n2])
                Ajouter_Noeud(GA, [nn1, nn2])
                Ajouter_Arc(GA, In, [n1, n2], [nn1, nn2])
             Fin Si
           Fin Pour Fin Pour
         Fin Si

Retourner GA
```

Figure 15: Algorithme 3 - Construction du graphe d'association

4. Recherche de la clique maximale dans le graphe d'association :

La recherche de la clique maximale dans le graphe d'association est réalisée par l'algorithme 3 d'une manière récursive, cet algorithme est inspiré de l'algorithme proposé dans [71]. L'entrée de l'algorithme est le graphe d'association construit dans l'étape précédente à partir des deux O-Graphes. Au début l'algorithme initialise une liste_sommets vide, qui va représenter la liste des sommets appartenant à la clique maximale. Ensuite pour chaque itération, l'algorithme essaie d'ajouter un nouveau sommet à liste_sommets, obtenue comme résultat dans l'itération précédente. La liste_sommets contient des sommets dans une clique et aussi des sommets nuls SOMMET_NUL. La sortie de l'algorithme est la liste_sommets la plus grande, chaque sommet représente les deux entités considérées comme similaires.

La clique maximale représente le sous-graphe commun maximal des deux graphes O-Graphe. Ce qui veut dire que pour chaque sommet S_i du graphe Og_1 qui appartient au sous-graphe commun maximal trouvé (clique maximale), il y a un et seulement un sommet S_j du graphe Og_2 qui lui correspond et vice versa. Autrement dit, dans la clique maximale représentée par la liste_sommets on ne peut pas avoir deux sommets qui contiennent le même nœud S_i ou S_j, le sommet $[S_i,S_j]$ existe au plus une fois.

C'est pour ça l'algorithme 3 prend la liste des sommets du premier O-graphe, où chaque sommet du Og_1 est considéré comme un niveau de recherche. Pour chaque niveau il vérifie si l'un des sommets de ce niveau (les sommets avec le même nœud du graphe Og_1) est légal pour l'ajouter à la liste_sommets, sinon il ajoute un sommet nul Sommet_Nul pour ce niveau, pour dire qu'aucun sommet de ce niveau n'appartient à la clique maximale (sous-graphe commun maximal de Og_1 et Og_2). Donc le nœud du graphe Og_1 correspondant à ce niveau de recherche n'a aucun nœud candidat correspondant dans le graphe Og_2.

Pour qu'un sommet soit légal, il doit être connecté à tous les sommets non nuls dans la liste_sommets, le sommet nul est toujours légal (connecté à tous les autres sommets). Avec l'ajout des sommets légaux à chaque itération, on augmente la taille de la clique pour avoir à la fin la clique maximale.

La sortie de l'algorithme 3 est un ensemble de cliques maximales de même taille, et chaque clique est une solution possible des correspondances entre deux ontologies. Car dans un graphe (graphe d'association dans notre cas) c'est possible d'avoir plusieurs cliques ayant la même taille maximale. Le choix de la meilleure clique maximale est basé sur la similarité linguistique entre les deux nœuds de chaque sommet de la clique. La similarité linguistique des deux entités (nœuds) dans un sommet de la clique est le poids pour ce sommet. Le poids total d'une clique est alors la somme de tous les poids de ses sommets. Donc la meilleure clique maximale est celle qui a le poids total maximal. Chaque sommet de la meilleure clique maximale représente les deux entités considérées similaires.

```
Algorithme 3 : Recherche_Clique_Maximale(liste_sommets)
niveau = taille(liste_sommets)
nombre_sommets_nuls = compter_sommets_nuls(liste_sommets)
clique_taille = niveau – nombre_sommets_nuls
Si nombre_sommets_nuls >= meilleur_nombre_sommets_nuls
        Retourner      liste_sommets
Sinon
        Si niveau == niveau_max
                Si meilleur_nombre_sommets_nuls  !=  nombre_sommets_nuls
                        meilleur_nombre_sommets_nuls = nombre_sommets_nuls
                        supprimer_resultat()
                Fin Si
                sauvegarder_resultat(liste_sommets)
        Sinon
                G = ensemble de sommets [s1,s2] ayant s1 correspond au niveau courant
                G = G ∪ { SOMMET_NUL }
                Pour chaque s dans G
                        Si est_legal(s, liste_sommets)
                                Recherche_Clique_Maximale(liste_sommets + s)
                        Fin Si
                Fin Pour
        Fin Si
Fin Si
Retourner      liste_sommets
```

Figure 16 : l'algorithme 3 - Recherche de la Clique_Maximale

Conclusion :

Dans ce chapitre, nous avons vu les différentes étapes de l'algorithme ASCO3, qu'est utilisé pour aligner les ontologies représentées en OWL, en donnant comme résultats la liste des correspondances entre les entités des deux ontologies à aligner. Cet algorithme se base sur le traitement des graphes, de ce fait il transforme les ontologies de OWL en graphe du type O-Graphe (réseau sémantique). Ce réseau reflète la conceptualisation et la modélisation de l'ontologie, où nous avons codifié les différents types d'informations (structurelle, sémantique…), comme la façon dont les entités sont connectées, les liens sémantiques (correspondant à des primitives de OWL) entre des entités. Plus les deux réseaux sémantiques représentés par les deux graphes O-Graphe d'ontologies sont proches, plus les deux ontologies en question sont similaires. Pour ça l'algorithme ASCO3 essaye de trouver le sous-graphe commun le plus grand des deux graphes O-Graphe représentant les ontologies. Donc cela revient à la recherche de la clique maximale du graphe d'association construit à partir des deux graphes O-Graphe.

La construction du graphe d'association se base sur les deux graphes O-Graphe représentant les ontologies. En prenant en compte la sémantique des primitives de OWL, les liens directs entre les nœuds des graphes O-Graphe et aussi les liens indirects à l'aide de la notion du « saut ». La clique maximale du graphe d'association représente le sous-graphe commun le plus grand des deux graphes O-Graphe représentants les ontologies, autrement dit, la liste des correspondances entre les entités des deux ontologies.

Chapitre IV : Implémentation et évaluation

Introduction :

Dans ce chapitre, nous présentons les implémentations et les évaluations de l'algorithme d'alignement d'ontologies proposé par [79] et que nous avons présenté dans le chapitre 3. L'implémentation de l'algorithme a été effectuée en Java en utilisant l'API Jena, ainsi qu'avec l'aide de la bibliothèque JGrapheT pour la visualisation des O-Graphes. Les évaluations ont été faites sur une base de tests, qui se compose des petites ontologies, ainsi que d'autres, proposées par la compagne I^{3}CON (http://www.atl.lmco.com/projects/ontology/i3con.html).

I. Les ontologies de tests :

A. Les ontologies utilisées et pourquoi ?

1. L'ontologie « Person » :

On a opté à utiliser une paire d'ontologies qui soient petites, pour tester notre application en un temps raisonnable. Ces ontologies se composent d'un nombre très limité d'entités, ce qui rend le temps d'exécution de l'application assez rapide en comparaison avec d'autres ontologies plus grandes. La structure des ontologies de cette paire est assez similaire, mais les entités sont nommées différemment (par ex. Human et Person), ce qui montre la puissance de l'algorithme.

2. L'ontologie « Hotel » :

La paire d'ontologies Hotel, décrit les caractéristiques des chambres d'hôtel. Les deux ontologies sont équivalentes, mais elles décrivent les mêmes concepts de différentes façons. Les deux ontologies de cette paire ont une structure assez différente et les entités sont nommées différemment. Par exemple, la classe OnFloor de la première ontologie correspond à la classe FloorAttribute de la deuxième ontologie.

3. L'ontologie « Network » :

networkA.owl et networkB.owl décrivent les nœuds et les connexions dans un réseau local. networkA.owl se concentre davantage sur les nœuds eux-mêmes, tandis que networkB.owl est plus englobant des connexions. Les deux ontologies dans cette paire ont une structure similaire et les termes nommant les entités sont aussi similaires (Equipment - Equipment, Cable - Cable, NetworkNode - NetworkNode). Par contre, la structure d'ontologie est assez complexe, avec la profondeur maximale de 4. Donc, on a ajouté cette paire d'ontologies à nos tests pour avoir une diversité de type d'ontologies pour mieux tester les performances de l'algorithme.

B. La sérialisation des ontologies de test en OWL avec leurs O-Graphes :

1. Ontologie « Person » :

Pour tester l'algorithme implémenté, on a utilisé un pair de petites ontologies : PersonA et PersonB. Le Tableau 4 montre les sérialisations en OWL de deux ontologies, et les représentations des O-Graphes sont dans la Figure 17.

Ontologie PersonA	Ontologie PersonB
<?xml version="1.0" encoding="US-ASCII" ?> <!DOCTYPE rdf:RDF [<!ENTITY owl "http://www.w3.org/2002/07/owl#" > <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >]> <rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xml:base ="http://wwwsop.inria.fr/acacia/ontologies/onto1.owl#" xmlns="http://wwwsop.inria.fr/acacia/ontologies/onto1.owl#"> <owl:Ontology rdf:about=""> <rdfs:comment>A person ontology</rdfs:comment> <rdfs:label>Person Ontology 1</rdfs:label> </owl:Ontology> <owl:Class rdf:ID="Person" /> <owl:Class rdf:ID="Adult"> <rdfs:subClassOf rdf:resource="#Person"/> </owl:Class> <owl:Class rdf:ID="Adult_Female"> <rdfs:label>Adult Female</rdfs:label> <rdfs:label>Woman</rdfs:label> <rdfs:subClassOf rdf:resource="#Adult"/> </owl:Class> <owl:Class rdf:ID="Man" /> <owl:Class rdf:ID="Adult_Male"> <rdfs:subClassOf rdf:resource="#Person"/> <owl:equivalentClass rdf:resource="#Man"/> </owl:Class> <owl:Class rdf:ID="Book"> <rdfs:subClassOf rdf:resource="#Publication"/> <owl:disjointWith rdf:resource="#Magazine"/> </owl:Class> <owl:Class rdf:ID="Magazine"> <rdfs:subClassOf rdf:resource="#Publication"/> </owl:Class> <owl:Class rdf:ID="Publication" /> <owl:ObjectProperty rdf:ID="write"> <rdfs:domain rdf:resource="#Person"/> <rdfs:range rdf:resource="#Publication"/> </owl:ObjectProperty> </rdf:RDF>	<?xml version="1.0" encoding="US-ASCII" ?> <!DOCTYPE rdf:RDF [<!ENTITY owl "http://www.w3.org/2002/07/owl#" > <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >]> <rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xml:base ="http://wwwsop.inria.fr/acacia/ontologies/onto2.owl#" xmlns="http://wwwsop.inria.fr/acacia/ontologies/onto2.owl#"> <owl:Ontology rdf:about=""> <rdfs:comment>A person ontology</rdfs:comment> <rdfs:label>Person Ontology 2</rdfs:label> </owl:Ontology> <owl:Class rdf:ID="Human" /> <owl:Class rdf:ID="Woman"> <rdfs:label>Woman</rdfs:label> <rdfs:label>Adult Female</rdfs:label> <rdfs:subClassOf rdf:resource="#Human"/> </owl:Class> <owl:Class rdf:ID="Man"> <rdfs:subClassOf rdf:resource="#Human"/> </owl:Class> <owl:Class rdf:ID="Booklet" /> <owl:ObjectProperty rdf:ID="write"> <rdfs:subPropertyOf rdf:resource="#create"/> </owl:ObjectProperty> <owl:ObjectProperty rdf:ID="create"> <rdfs:domain rdf:resource="#Human"/> <rdfs:range rdf:resource="#Booklet"/> </owl:ObjectProperty> </rdf:RDF>

Tableau 4 : Tableau de sérialisation des deux ontologies PersonA et PersonB

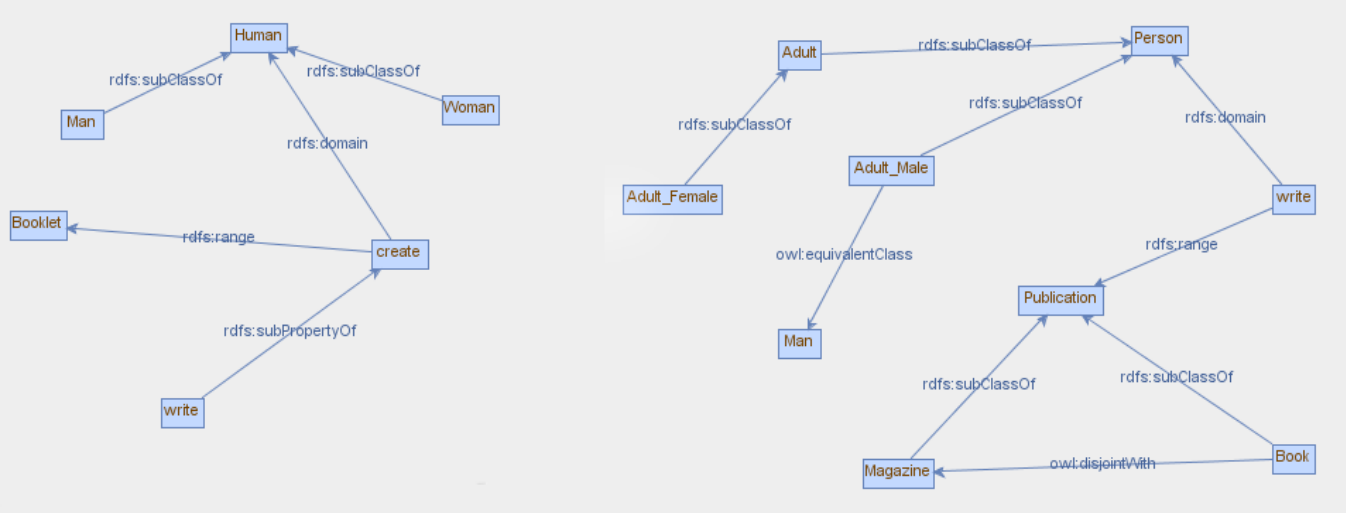


Figure 17: Représentation des OGraphe des deux ontologies PersonA (droite) et PersonB (gauche)

2. Ontologie « Hotel » :

Une autre paire d'ontologies qu'on a utilisée est HotelA.owl et HotelB.owl, qui sont deux ontologies appartenant à la compagne I^{3}CON, I^{3}CON (Information Interpretation and Integration Conference) qui est le premier effort dans la communauté de la recherche de l'intégration et de l'interprétation de l'ontologie et du schéma, pour fournir des outils et un Framework d'évaluation systématique de l'intégration des ontologies et des schémas. I3CON fournit 10 paires d'ontologies accompagnées de leurs fichiers de référence qui contient les correspondances déterminées manuellement entre deux ontologies.

Mais **N.B.** : ***nous n'avons pas utilisé les ontologies proposées par I^{3}CON tout entières, on a tranché quelques entités, vu que le temps d'exécution est très important.***

Le Tableau 5 montre les sérialisations en OWL de deux ontologies, et les représentations des O-Graphes sont dans la Figure 18.

Ontologie HotelA	Ontologie HotelB
<?xml version="1.0" encoding="US-ASCII" ?> <!DOCTYPE rdf:RDF [<!ENTITY owl "http://www.w3.org/2002/07/owl#" > <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >]> <rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xml:base="http://www.atl.lmco.com/projects/ontology/ontologies/hotel/hotelA.owl" xmlns="http://www.atl.lmco.com/projects/ontology/ontologies/hotel/hotelA.owl#"> <owl:Ontology rdf:about=""> <rdfs:comment>A hotel ontology</rdfs:comment> <rdfs:label>Hotel Ontology 1</rdfs:label> </owl:Ontology>	<?xml version="1.0" encoding="US-ASCII" ?> <!DOCTYPE rdf:RDF [<!ENTITY owl "http://www.w3.org/2002/07/owl#" > <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >]> <rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xml:base="http://www.atl.lmco.com/projects/ontology/ontologies/hotel/hotelB.owl" xmlns="http://www.atl.lmco.com/projects/ontology/ontologies/hotel/hotelB.owl#"> <owl:Ontology rdf:about=""> <rdfs:comment>A hotel ontology</rdfs:comment> <rdfs:label>Hotel Ontology 2</rdfs:label> </owl:Ontology> <owl:Class rdf:ID="Room"/> <owl:Class rdf:ID="RoomAttribute"/> <owl:Class rdf:ID="ExclusiveAttribute">

```
<owl:Class rdf:ID="HotelRoom"/>

<owl:Class rdf:ID="Suite">
   <rdfs:subClassOf rdf:resource="#HotelRoom"/>
</owl:Class>

<owl:Class rdf:ID="OneRoom">
   <rdfs:subClassOf rdf:resource="#HotelRoom"/>
</owl:Class>

<owl:Class rdf:ID="SmokingPreference"/>

<owl:Class rdf:ID="OnFloor"/>
<OnFloor rdf:ID="Hardwood"/>

<owl:Class rdf:ID="NumBeds"/>
<NumBeds rdf:ID="one"/>


<owl:ObjectProperty rdf:ID="hasNumBeds">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#HotelRoom" />
   <rdfs:range rdf:resource="#NumBeds"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasOnFloor">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#HotelRoom" />
   <rdfs:range rdf:resource="#OnFloor" />
</owl:ObjectProperty>


</rdf:RDF>
```

```
   <rdfs:subClassOf rdf:resource="#RoomAttribute"/>
</owl:Class>
<owl:Class rdf:ID="SmokingAttribute">
   <rdfs:subClassOf rdf:resource="#ExclusiveAttribute"/>
</owl:Class>
<owl:Class rdf:ID="SizeAttribute">
   <rdfs:subClassOf rdf:resource="#ExclusiveAttribute"/>
</owl:Class>
<SizeAttribute rdf:ID="SingleRoom"/>
<SizeAttribute rdf:ID="Suite"/>
<owl:Class rdf:ID="FloorAttribute">
   <rdfs:subClassOf rdf:resource="#ExclusiveAttribute"/>
</owl:Class>
<FloorAttribute rdf:ID="Hardwood"/>
<owl:Class rdf:ID="NumBedsAttribute">
   <rdfs:subClassOf rdf:resource="#ExclusiveAttribute"/>
</owl:Class>
<NumBedsAttribute rdf:ID="OneBed"/>
<owl:ObjectProperty rdf:ID="RoomProperty"/>
<owl:ObjectProperty rdf:ID="ExclusiveProperty">
   <rdfs:subPropertyOf rdf:resource="#RoomProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="SizeOfRoom">
   <rdfs:subPropertyOf rdf:resource="#ExclusiveProperty"/>
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#Room" />
   <rdfs:range rdf:resource="#SizeAttribute" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="OnFloor">
   <rdfs:subPropertyOf rdf:resource="#ExclusiveProperty"/>
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#Room" />
   <rdfs:range rdf:resource="#FloorAttribute" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="NumBeds">
   <rdfs:subPropertyOf rdf:resource="#ExclusiveProperty"/>
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#Room" />
   <rdfs:range rdf:resource="#NumBedsAttribute"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Tableau 5: Tableau de sérialisation des deux ontologies HotelA et HotelB

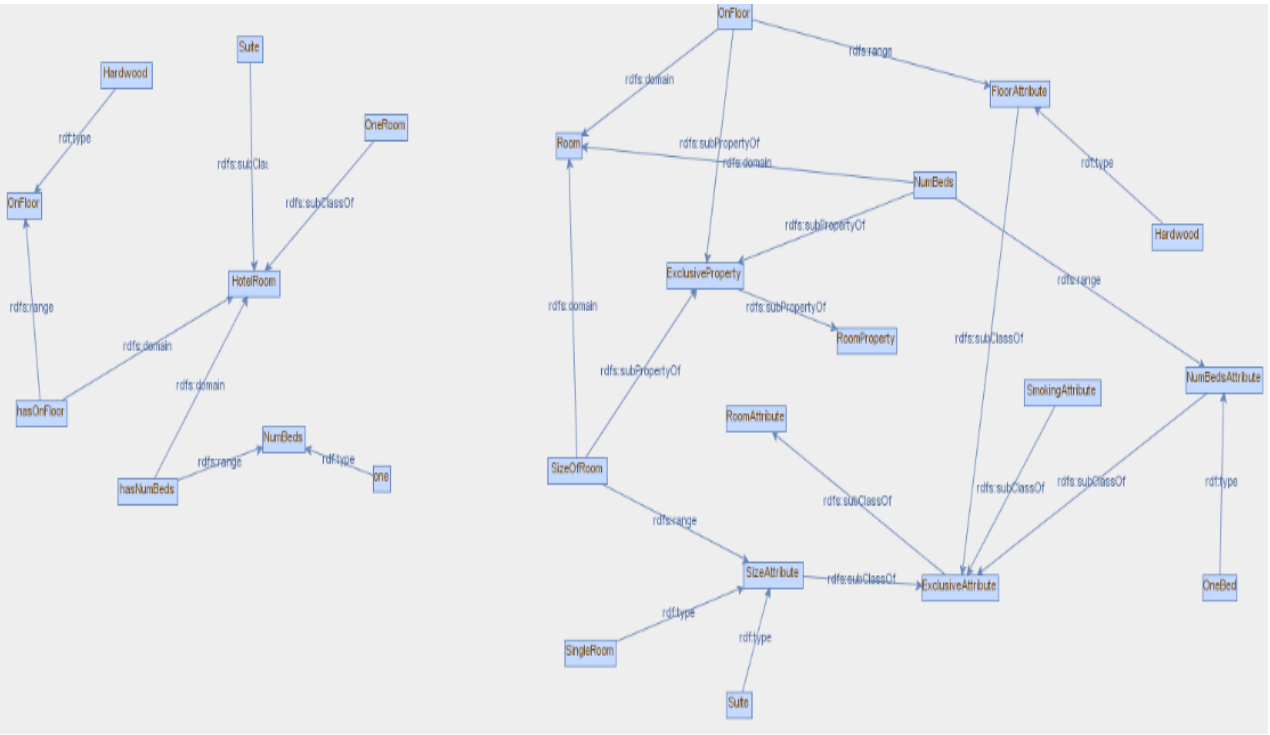


Figure 18 Représentation de OGraphe des deux ontologies HotelA (gauche) et HotelB (droite)

3. Ontologie « Network » :

Un autre pair d'ontologies appartenant à la compagne I^{3}CON qu'on a utilisé est NetworkA.owl et NetworkB.owl, dont on a aussi enlevé quelques entités.

Le Tableau 6 montre les sérialisations en OWL de deux ontologies, et les représentations des O-Graphes sont dans la Figure 19.

NetworkA

```
<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#"
>
    <!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#" >
  ]>
<rdf:RDF
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xml:base="http://www.atl.lmco.com/projects/ontolog
y/ontologies/network/networkA.owl"
   xmlns="http://www.atl.lmco.com/projects/ontology/o
ntologies/network/networkA.owl#">
<owl:Ontology rdf:about="">
   <rdfs:comment>A networking
ontology</rdfs:comment>
   <rdfs:label>Network Ontology 1</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="Equipment"/>
<owl:Class rdf:ID="NetworkNode">
   <rdfs:subClassOf rdf:resource="#Equipment"/>
</owl:Class>
<owl:Class rdf:ID="Computer">
   <rdfs:subClassOf rdf:resource="#NetworkNode"/>
</owl:Class>
<owl:Class rdf:ID="SwitchEquipment">
   <rdfs:subClassOf rdf:resource="#NetworkNode"/>
</owl:Class>
<owl:Class rdf:ID="Cable">
   <rdfs:subClassOf rdf:resource="#Equipment"/>
</owl:Class>
<owl:Class rdf:ID="ServerSoftware"/>
<owl:Class rdf:ID="FTPServer">
   <rdfs:subClassOf rdf:resource="#ServerSoftware"/>
</owl:Class>
<owl:Class rdf:ID="OtherServer">
   <rdfs:subClassOf rdf:resource="#ServerSoftware"/>
</owl:Class>
<owl:Class rdf:ID="NodePair"/>
<owl:ObjectProperty rdf:ID="ConnectedWith">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#NodePair"/>
   <rdfs:range rdf:resource="#Cable"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="NodeA">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#NodePair"/>
   <rdfs:range rdf:resource="#NetworkNode"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="NodeB">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#NodePair"/>
   <rdfs:range rdf:resource="#NetworkNode"/>
</owl:ObjectProperty>
</rdf:RDF>
```

NetWorkB

```
<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#"
>
    <!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#" >
  ]>
<rdf:RDF
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xml:base="http://www.atl.lmco.com/projects/ontolog
y/ontologies/network/networkB.owl"
   xmlns="http://www.atl.lmco.com/projects/ontology/o
ntologies/network/networkB.owl#">
<owl:Ontology rdf:about="">
   <rdfs:comment>A networking
ontology</rdfs:comment>
   <rdfs:label>Network Ontology 2</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="Equipment"/>
<owl:Class rdf:ID="NetworkNode">
   <rdfs:subClassOf rdf:resource="#Equipment"/>
</owl:Class>
<owl:Class rdf:ID="Computer">
   <rdfs:subClassOf rdf:resource="#NetworkNode"/>
</owl:Class>
<owl:Class rdf:ID="CentralHub">
   <rdfs:subClassOf rdf:resource="#NetworkNode"/>
</owl:Class>
<owl:Class rdf:ID="Cable">
   <rdfs:subClassOf rdf:resource="#Equipment"/>
</owl:Class>
<owl:Class rdf:ID="ServerSoftware"/>
<owl:Class rdf:ID="FTP">
   <rdfs:subClassOf rdf:resource="#ServerSoftware"/>
</owl:Class>
<owl:Class rdf:ID="Other">
   <rdfs:subClassOf rdf:resource="#ServerSoftware"/>
</owl:Class>
<owl:Class rdf:ID="PairOfNodes"/>
<owl:ObjectProperty rdf:ID="ConnectedThrough">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#PairOfNodes"/>
   <rdfs:range rdf:resource="#Cable"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="NodeA">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#PairOfNodes"/>
   <rdfs:range rdf:resource="#NetworkNode"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="NodeB">
   <rdf:type rdf:resource="&owl;FunctionalProperty" />
   <rdfs:domain rdf:resource="#PairOfNodes"/>
   <rdfs:range rdf:resource="#NetworkNode"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Tableau 6:Tableau de sérialisation des deux ontologies NetworkA et NetworkB

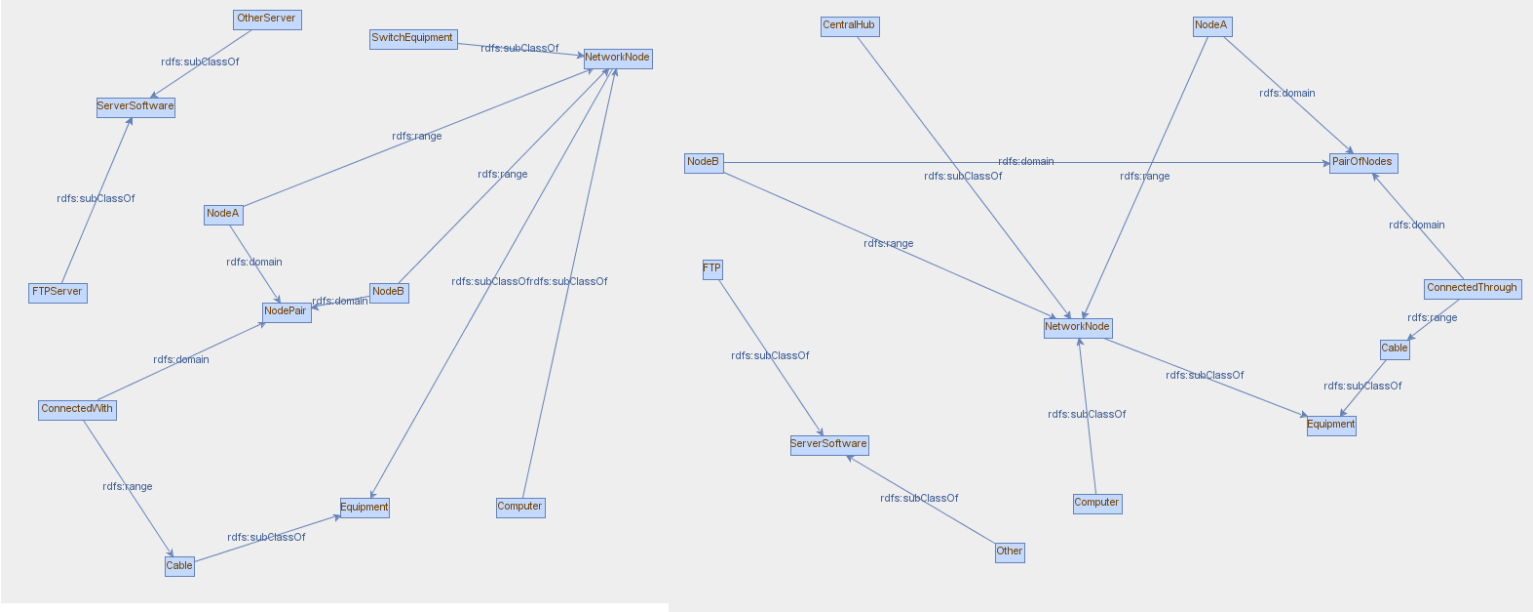


Figure 19:Représentation de O Graphe des deux ontologies NetworkA (gauche) et NetworkB (droite)

II. Implémentation :

L'algorithme d'alignement d'ontologie ASCO3 est implémenté avec Java, dans l'application nommée « AlignementOntologie » afin d'évaluer sa performance. Les fichiers des ontologies en *OWL* sous format en XML sont choisis par l'utilisateur puis chargés dans l'application en employant Jena (celui qui fournit un environnement programmatique pour les applications du Web sémantique).

A. Les outils utilisés :

1. Apache Jena :

Apache Jena est une Framework open-source du Web Sémantique pour Java. Il offre un API qui permet l'extraction de, ou bien l'écriture dans un graphe RDF. Les graphes sont représentés sous forme d'un « model » abstrait. Un modèle peut provenir des données à partir de fichiers, des bases de données, des URL ou une combinaison de ceux-ci. Un modèle peut également être interrogé par SPARQL.

Jena est similaire à Sesame (un Framework open-source pour l'interrogation et l'analyse des données RDF), mais, contrairement à Sesame, Jena fournit un support pour OWL (Web Ontology Language). Ce Framework a diverses raisonneurs internes et le raisonneur Pellet (un raisonneur Open-source Java OWL-DL) peut être mis en place dans Jena.

Jena supporte la sérialisation des graphes RDF en :

- Une base de données relationnelle
- RDF/XML
- Turtle

- N3

2. JGrapheT :

JGraphT est une bibliothèque graphique Java gratuite qui fournit des objets et des algorithmes de la théorie des graphes mathématique. JGraphT prend en charge différents types de graphiques, y compris :

- Les graphes orientés et non orientés
- Les graphes avec des arcs pondérés / non pondérés / étiquetés ou tout type d'arcs définis par l'utilisateur.
- Diverses options de multiplicité des arcs, y compris : simple-graphiques, multigraphes, pseudographs.
- Graphes non modifiables - permettent aux modules de fournir l'accès "lecture seule" à des graphes internes.
- Graphiques modifiables.
- Les sous-graphes qui sont des vues de sous-graphe à mise à jour automatique sur d'autres graphes.
- Toutes les compositions des graphes ci-dessus.

Bien que puissant, JGraphT est conçu pour être simple et de type sécurisé (via les génériques Java). Par exemple, les sommets du graphe peuvent être de tous les objets. On peut créer des graphes à partir de : Cordes, URL, documents XML, etc. ; même des graphes de graphes.

Parmi les fonctionnalités de JGrapheT, il y a la librairie de visualisation des graphes, à laquelle nous sommes intéressés et que nous avons utilisée dans notre application pour tracer les O-Graphes des deux ontologies.

B. Interfaces graphiques :

La figure 20 est l'interface d'accueil de notre application. Elle contient deux boutons « Choisir Fichier » qui permettent à l'utilisateur de choisir les fichiers OWL des deux ontologies à aligner, et cela à travers un explorateur de fichiers (Figure 21).

Après qu'on charge les fichiers des ontologies à aligner, on appuie sur le bouton « Démarrer », pour lancer l'algorithme.

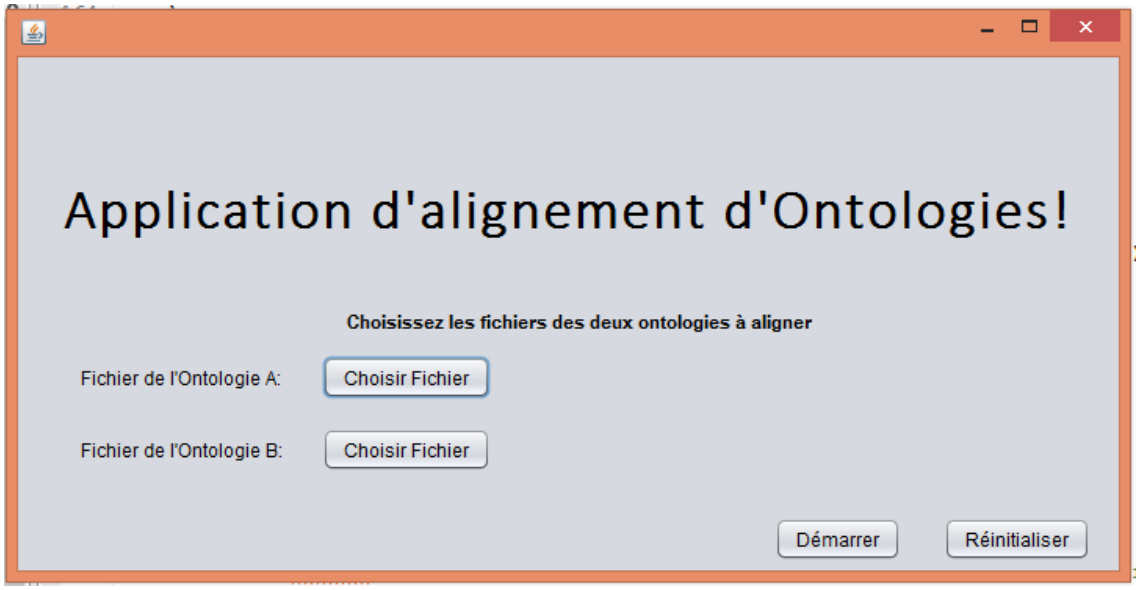


Figure 20: L'interface d'accueil de l'application "AlignementOntologie"

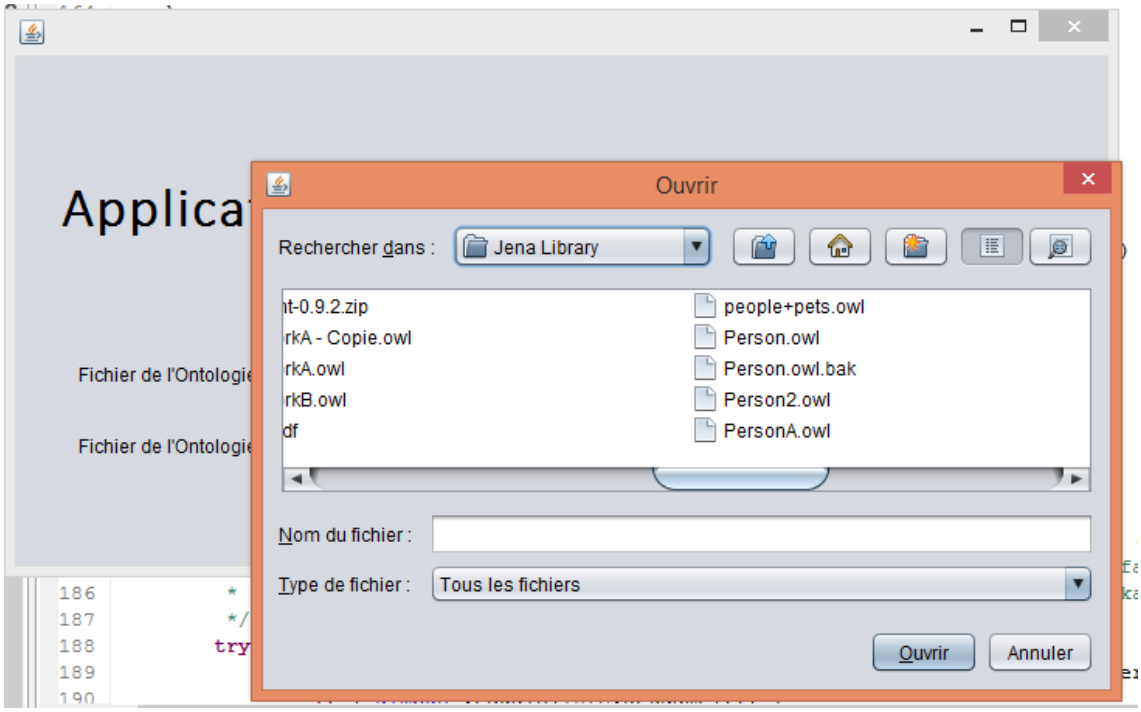


Figure 21: la fenêtre de l'explorateur pour choisir le fichier d'ontologie

Après que l'algorithme finisse son travail, une nouvelle fenêtre apparaît, dans laquelle le résultat des correspondances est affiché sous forme d'un tableau en bas des deux O-Graphes de l'ontologie A et celui de l'ontologie B (Figure 22).

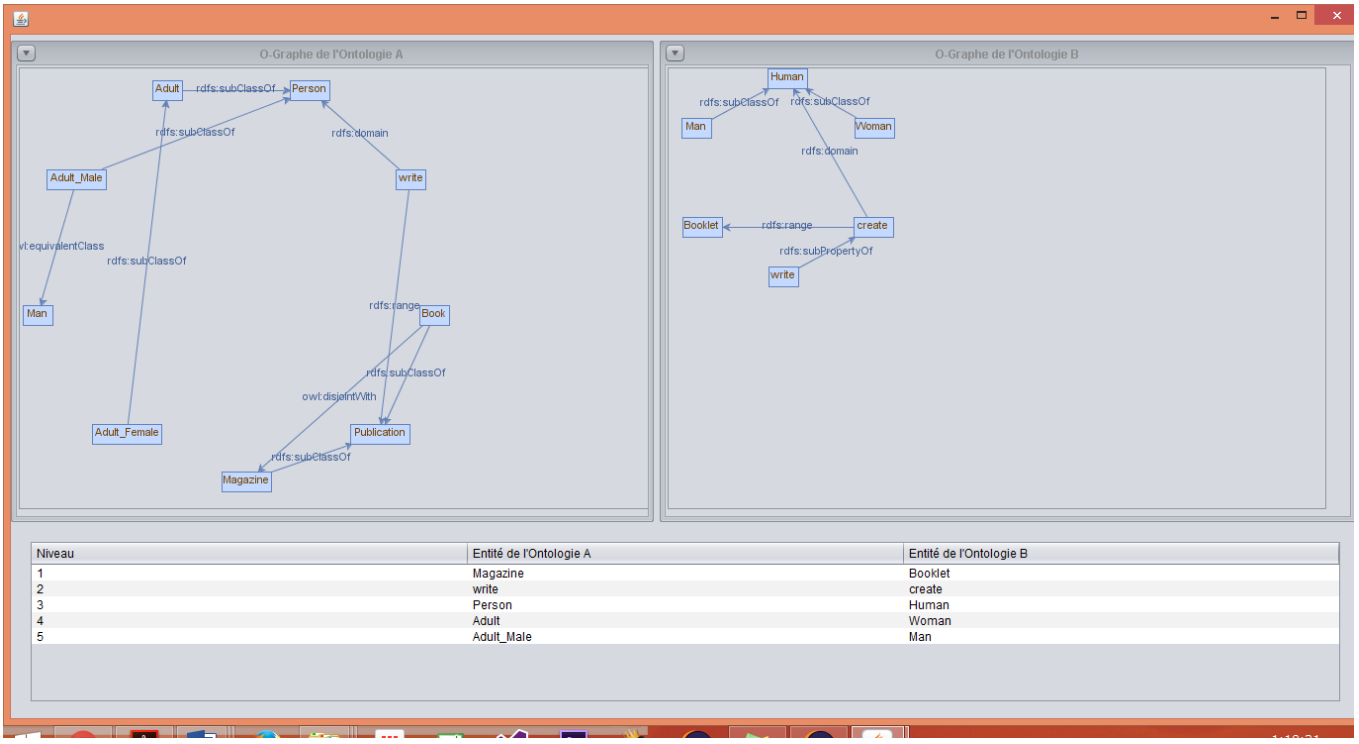


Figure 22:L'interface montrant le résultat de l'alignement

III. Résultats expérimentaux :

A. L'ontologie « Person » :

La Figure suivante représente la liste des correspondances trouvées par l'algorithme ASCO3 :

Niveau	Entité de l'Ontologie A	Entité de l'Ontologie B
1	Magazine	Booklet
2	write	create
3	Person	Human
4	Adult	Woman
5	Adult_Male	Man

Figure 23: Résultats des correspondances pour l'ontologie Person par ASCO3

Ici, on remarque que l'algorithme a trouvé 5 correspondances, et qui sont toute correcte. Donc, l'algorithme ASCO3 montre une performance impressionnante avec un résultat de 100 % de correspondances correctes. Rappelons qu'un nœud du graphe d'association se compose de deux nœuds d'O-Graphe, qui correspondent à deux entités de deux ontologies. Alors, les nœuds dans la clique maximale trouvée dans le graphe d'association correspondent aux paires des entités des deux ontologies, qui sont considérées par ASCO3 comme des correspondances correctes entre deux ontologies.

B. L'ontologie « Hotel » :

La Figure suivante montre les résultats obtenus pour la paire d'ontologies HotelA et HotelB de la compagne I^{3}CON, avec quelques entités enlevées.

Niveau	Entité de l'Ontologie A	Entité de l'Ontologie B
1	hasNumBeds	NumBeds
2	NumBeds	NumBedsAttribute
3	OneRoom	Room
4	Suite	RoomAttribute
5	Hardwood	Hardwood
6	OnFloor	FloorAttribute
7	one	OneBed
8	hasOnFloor	OnFloor

Figure 24: Résultats des correspondances pour l'ontologie Hotel par ASCO3

En comparant ces résultats avec les correspondances correctes déterminées manuellement par des experts (celles dans le fichier de référence) indiquées dans le tableau suivant :

Ontologie HotelA	Ontologie HotelB
HotelRoom	Room
OneRoom	SingleRoom
Suite	Suite
NumBeds	NumBedsAttribute
SmokingPreference	SmokingAttribute
OnFloor	FloorAttribute
hasNumBeds	NumBeds
hasOnFloor	OnFloor
One	OneBed
Hardwood	Hardwood

Tableau 7:La liste des correspondances correctes pour l'ontologie "Hotel" dans le fichier de référence fourni

On obtient les mesures suivantes :

F	T	C	I	M	Précision
8	10	6	2	4	0.75

Tableau 8: Résultats obtenus par Asco3 pour l'ontologie "Hotel"

- F – le nombre des correspondances renvoyées par l'algorithme
- T – le nombre des correspondances correctes déterminées manuellement par des experts (celles dans le fichier de référence)
- C – le nombre des correspondances correctes trouvées (appelé aussi vrais positifs)
- I = F - C – nombre des correspondances incorrectes trouvées (appelé aussi faux positifs)
- M = T - C – nombre des correspondances correctes, mais pas trouvées (appelé aussi faux négatifs).
- Ainsi, Précision = C / F

C. L'ontologie « Network » :

La Figure suivante montre les résultats obtenus pour la paire d'ontologies NetworkA et NetworkB de la compagne I[3]CON, avec quelques entités enlevées.

Niveau	Entité de l'Ontologie A	Entité de l'Ontologie B
1	NetworkNode	NetworkNode
2	Equipment	Equipment
3	NodeB	NodeA
4	NodePair	PairOfNodes
5	Computer	Computer
6	SwitchEquipment	CentralHub
7	OtherServer	Other
8	ServerSoftware	ServerSoftware
9	NodeA	NodeB
10	FTPServer	FTP
11	ConnectedWith	ConnectedThrough
12	Cable	Cable

Figure 25:Résultats des correspondances pour l'ontologie Network par ASCO3

En comparant ces résultats avec les correspondances correctes déterminées manuellement par des experts (celles dans le fichier de référence) indiquées dans le tableau suivant :

Ontologie networkA	Ontologie networkB
Equipment	Equipment
Cable	Cable
NetworkNode	NetworkNode
Computer	Computer
SwitchEquipment	CentralHub
ServerSoftware	ServerSoftware
FTPServer	FTP
OtherServer	Other
NodePair	PairOfNodes
ConnectedWith	ConnectedThrough
NodeA	NodeA
NodeA	NodeB
NodeB	NodeA
NodeB	NodeB

Tableau 9: La liste des correspondances correctes pour l'ontologie "Network" dans le fichier de référence fourni

On obtient les mesures suivantes :

F	T	C	I	M	Précision
12	12	12	0	0	1

Tableau 10: Résultats obtenus par Asco3 pour l'ontologie "Network"

IV. Limites et Perspectives :

A. Les Limites :

Comme nous avons vu dans le chapitre précédent lors de la création du graphe d'association, pour ajouter un nouveau sommet au graphe d'association les deux nœuds des

deux graphes O-Graphe doivent être n-compatibles. Autrement dit, les deux entités représentées par ces deux nœuds doivent être de même type (classe, propriété ou instance). Donc cet algorithme ne peut pas trouver la correspondance entre les entités qui n'ont pas la même nature (classe, propriété ou instance).

Aussi au niveau de la recherche de la clique maximale, on a vu que les nœuds des deux graphes O-Graphe ne peuvent pas se répéter dans les différents sommets de la clique maximale (chaque nœud existe une fois au plus). Donc cet algorithme ne cherche qu'une seule entité correspondante dans la deuxième ontologie pour chaque entité de la première ontologie. Alors si pour une entité de la 1ère ontologie on a plusieurs entités considérées similaires de la 2ème ontologie, l'algorithme ne peut trouver qu'une seule entité.

La complexité de cet algorithme correspond à la complexité de l'algorithme de recherche de la clique maximale dans le graphe d'association. La recherche de la clique maximale est considérée comme approche de recherche exhaustive (méthode de force brute). Le temps d'exécution de l'algorithme augmente exponentiellement avec la taille du graphe d'association. Donc si les ontologies à aligner se composent de beaucoup d'entités, le temps d'exécution de l'algorithme sera très important.

B. Les Perspectives :

Comme perspective, on doit essayer de trouver des solutions pour ces limites, afin d'améliorer cet algorithme. Surtout pour le problème du temps d'exécution de l'algorithme, on doit trouver une manière pour optimiser le temps de la recherche de la clique maximale dans le graphe d'association. Ce qui va nous permettre d'exécuter cet algorithme sur des ontologies de tailles normales et des ontologies réelles.

Conclusion

Dans ce chapitre, nous avons vu les outils qu'on a utilisés pour l'implémentation de l'algorithme ASCO 3, qui sont des API en java. Comme JENA qui nous a permis d'extraire les triplets à partir des fichiers OWL, JGrapheT qu'on a utilisé pour tracer les graphes O-Graphe. On a fait un petit aperçu sur notre interface.

Pour les ontologies de teste on n'avait pas beaucoup de choix, vu la complexité de l'algorithme et le temps d'exécution. On a utilisé que de petites ontologies, qui ont environ des dizaines d'entités. On a même réduit la taille de la paire d'ontologie hôtel (HotelA et HotelB, NetworkA et NetworkB) afin de pouvoir les tester.

Donc on doit trouver une solution pour optimiser le temps de la recherche de la clique maximale, qui va réduire le temps d'exécution.

Conclusion Générale :

L'objectif de notre travail était de réaliser un état de l'art concernant le domaine d'alignement d'ontologies, plus particulièrement les méthodes d'alignements. Ainsi que la présentation d'un algorithme qui a pour but d'aligner des ontologies, ASCO3. L'algorithme ASCO3 qui vise à chercher des alignements entre deux ontologies représentées en OWL (le langage d'ontologie recommandé par W3C pour le Web sémantique). Cet algorithme essaie d'exploiter les avantages d'expressivité du langage d'ontologie OWL pour déduire la similarité de deux entités dans les ontologies en OWL.

L'algorithme ASCO3 est la seule approche appliquant, pour l'alignement des ontologies, la notion d'isomorphisme des graphes et les algorithmes de recherche de sous-graphe commun de deux graphes, qui sont bien étudiés dans la théorie des graphes. Son idée est de considérer des ontologies en OWL, définies en employant les primitives de OWL qui ont les sémantiques bien prédéfinies, comme des réseaux sémantiques dont les nœuds sont les entités, les arcs sont les primitives de OWL. Ces réseaux sémantiques reflètent la conceptualisation et la modélisation des ontologies, et donc, plus ils sont proches, plus les ontologies sont similaires.

L'algorithme proposé a été implémenté en Java pour évaluer la qualité des correspondances entre deux ontologies trouvées. Dans les tests effectués, l'algorithme s'est montré être performante, surtout dans le cas de la dernière paire d'ontologies de test (Network), qui dans ce test nous donne le résultat correct et parfait en employant les «sauts». La sémantique des primitives d'OWL est bien exploitée via cette notion de « saut ». Cependant, il n'est pas encore optimisé au niveau de la performance, puisque l'algorithme ASCO3 applique l'algorithme de recherche de la clique maximale dans le graphe d'association de deux graphes représentant deux ontologies à comparer en utilisant une recherche exhaustive (ce qui peut être très compliqué et gourmand en temps de calcul).

Finalement, bien qu'on constate qu'il y a plusieurs travaux et de recherches qui traitent ce sujet, le problème d'alignement d'ontologies est loin d'être résolu à 100 %, il reste beaucoup encore de travaux à faire dans ce domaine.

Références :

1. Lê Bach Thanh (2006) - Construction d'un Web sémantique multipoints de vue.
2. Berners-Lee, T. (2001). The semantic web. Scientific american, pp. 284(5):35–43.
3. BACHIMONT, B. (2-000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances, in CHARLET J., ZACKLAD M., KASSEL G. & BOURIGAULT D., eds. , Ingénierie des connaissances : évolutions récentes et nouveaux défis (éd. Eyrolles). Eyrols.
4. Fernandez, M., Gomez-Perez, A., & Juristo, N. (1997). From ontological art towards ontological engineering, in Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97), AAAI Press.
5. MIZOGUCHI, R., & IKEDA, M. (1997). Towards ontolgy engineering, in Proceedings of the Joint Pacific Asian Conference on Expert Systems.
6. USCHOLD, M., & KING, M. (1995). Towards a methodology for building ontologies, in Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI'95.
7. GOMEZ-PEREZ, A., FERNANDEZ, M., & DE VICENTE, A. J. (1996). Towards a Method to Conceptualize Domain Ontologies, in Proceedings of the European Conference on Artificial Intelligence ECAI'96. pp. 41-52.
8. Bouaud, J., Bachimont, B., Charlet, J., & Zweigenbaum, P. (1995). Methodological Principles for structuring an ontology, in Proceedings of IJCAI'95 Workshop: Basic Ontological Issues in Knowledge sharing.
9. GUARINO, N., & GIARETTA, P. (1995). Ontologies and knowledge bases, towards a terminological clarification, in MARS N. (I. P. Towards very large knowledge bases: knowledge building and knowledge sharing, Éd.)
10. KASSEL, G. (2002). OntoSpec : une méthode de spécification semi-informelle d'ontologies, in Actes des journées francophones d'Ingénierie des Connaissances (IC'2002). pp. 75-87.
11. WELTY, C., & GUARINO, N. (2001). Supporting ontological analysis of taxonomic relationships, Data et Knowledge Engineering (39). pp. 51-74.
12. Lynda DJAKHDJAKHA, (2014) : Un formalisme pour l'alignement des ontologies multipoints de vue basé sur une extension de la logique de description.
13. VanHeijst, G., Schreiber, A.T. & Wielinga, B.J., 1997. Using explicit ontologies in KBS development. Int. J. Hum.-Comput. Stud., 46(2), pp.183–292.
14. Mizoguchi, R. et al., 2000. Construction and Deployment of a Plant Ontology. In R. Dieng & O. Corby, eds. EKAW. Lecture Notes in Computer Science. Springer, pp. 113–128.
15. Maedche, A., 2002. Clustering Ontology-Based Metadata in the Semantic Web. In T. Elomaa, H. Mannila, & H. Toivonen, eds. PKDD. Lecture Notes in Computer Science. Springer, pp. 348–360.

16. Yiling, L., 2003. Roadmap for tool support for collaborative ontology engineering. University of Victoria.
17. Cerqueus, T., 2012. Contributions au probléme d'hétérogénéité sémantique dans les systèmes pair-à-pair : application à la recherche d'information.
18. GINSBERG, M. L. (1994). Knowledge Interchange Format: the KIF of death. AI Magazine, 12(3):57–63.
19. BAGET, J. (2004). Homomorphismes d'hypergraphes pour la subsomption en RDF/RDFS. In Actes de la 10e conférence Langages et Modèles à Objet (LMO'2004), pages 203–216.
20. Fensel, D., Horrocks, I., van Harmelen, F., & De, S. (2000). Oil in a nutshell. In Proceedings of European Knowledge Acquisition Workshop (EKAW'2000), volume 1937, pages 1–16. Springer-Verlag LNAI.
21. HENDLER, J., & MCGUINNESS, D. (2001). The Darpa Agent Markup Language. http://www.daml.org,.
22. GRUBER, T., & OLSEN, G. (1994). An ontology for enginering mathematics, in J. DOYLE F. S.&TORANO P., eds., Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasonning,. Morgan-Kauffmann.
23. TRONCY, R., & ISSAC, A. (2002). DOE : Une mise en oeuvre d'une méthode de structuration différentielle pour les ontologies, in Actes des journées francophones d'Ingénierie des Connaissances (IC'2002), pages 63-74.
24. DOE. (2002). Differential Ontology Editor Home Page, http ://opales.ina.fr/public.
25. NOY, N., FERGERSON, R. W., & MUSEN, M. A. (2000). The knowledge model of Protégé2000 : combining interoperability and flexibility, in Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW'00).
26. PRO. (2002). PROTEGE2000, Protege2000 Ontology Editor Home Page, http://protege.stanford.edu/.
27. Noy, N. F., & Musen, M. A. (2000). Prompt : algorithm and tool for automated ontology merging and alignment. In Proceeding of Seventeenth National Conference on Artificial Intelligence AAAI.
28. ONTOEDIT. (2004). Ontology Editor Home Page, http ://www.ontoprise.de/com/
29. Euzenat, J. et Shvaiko, P. (2007). Ontology Matching. Springer-Verlag, Heidelberg (DE).
30. Euzenat, J. (2001). Towards a principled approach to semantic interoperability. In Proceedings of the IJCAI Workshop on Ontology and Information Sharing, pages 19 25, Seattle, US.
31. Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. In Proceedings of the IJCAI-Workshop on Ontologies and Information Sharing, pages 5362, Seattle, US.
32. Visser, P. R. S., Jones, D. M., Bench-Capon, T. J. M. et Shave, M. J. R. (1998). Assessing heterogeneity by classiying ontology mismatches. In Proceedings of the 1st

International conference on Formal Ontology in Information Systems (FOIS), pages 148-162, Trento, italy.

33. Benerecetti, M., Bouquet, P. et Ghidini, C. (2001). Contextual reasoning distilled. Journal of Experimental and Theoretical Artificial Intelligence, 12(3):279-305.
34. 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. et Zaihrayeu (2004) State of the art on ontology alignment, deliverable 2.2.3, IST Knowledge web NoE, Knowledge web NoE, 80p., June 2004.
35. Shvaiko, P., Euzenat, J. (2007) A Survey of Schema-based Matching Approaches. Journal on Data Semantics.
36. Durban, R., Eddy, S. R., Krogh, A., et Mitchison, G. (1998) Biological sequence analysis - Probabilistic models of proteins and nucleic acids. Cambridge : Cambridge University Press.
37. Jaro, M. A. (1989) Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 84:414–420.
38. Jaro, M. A. (1995) Probabilistic linkage of large public health data files (disc : P687-689). Statistics in Medicine 14:491–498.
39. Monge, A., et Elkan, C. (1996) The field-matching problem: algorithm and applications. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.
40. Cohen, W., Ravikumar, P., et Fienberg, S. (2003) A comparison of string metrics for matching names and records. Dans Proc. KDD-2003 Workshop on Data Cleaning and Object Consolidation.
41. Miller, A.G. (1995) Wordnet : A lexical database for english. Communications of the ACM, 38(11):39–41.
42. Giunchiglia, F. et Shvaiko, P. (2003) Semantic matching. Dans Proc. IJCAI 2003 Workshop on ontologies and distributed systems, Acapulco (MX), pages 139–146.
43. Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., et Stuckenschmidt, H. C- (2003) OWL : Contextualizing Ontologies. International Semantic Web Conference 2003: 164-179

44. Aissa Fellah, Mimoun Malki, Ahmed ZAHAF - Alignement des ontologies : utilisation de WordNet et une nouvelle mesure structurelle, Université Djillali Liabes de Sidi Bel abbés, département d'informatique, Algérie.
45. Yves R. Jean-Mary, Mansur R. Kabuka - ASMOV: Ontology Alignment with Semantic Validation - 9200 South Dadeland Blvd, Suite 620, Miami, Florida, USA 33156 University of Miami, Coral Gables, Florida, USA 33124
46. Yves R. Jean-Mary, E. Patrick Shironoshita, Mansur R. Kabuka - ASMOV: Results for OAEI 2010 - INFOTECH Soft, 1201 Brickell Ave., Suite 220, Miami, Florida, USA 33131, University of Miami, Coral Gables, Florida, USA 33124.
47. Abdeltif Elbyed - ROMIE, une approche d'alignement d'ontologies à base d'instances - L'institut National Des Télécommunications.
48. Sami Zghal, Sadok Ben Yahia, Engelbert Mephu Nguifo, and Yahya Slimani - SODA: an OWL-DL based ontology matching system- CRIL CNRS FRE 2499, Artois University, IUT of Lens Rue de l'Universit´e - S.P. 16, 62307 Lens Cedex, France - Computer Science Department, Faculty of Sciences of Tunis, Tunisia ,Campus Universitaire, 1060 Tunis, Tunisia.
49. Marc Ehrig and Steffen Staab - QOM - Quick Ontology Mapping - Institute AIFB, University of Karlsruhe, Germany.
50. Uthayasanker Thayasivam and Prashant Doshi - On the Utility of WordNet for Ontology Alignment: Is it Really Worth It? - THINC Lab Department of Computer Science - University of Georgia, USA.
51. Benoît Sagot Darja Fišer - Construction d'un Wordnet libre du français à partir de ressources multilingues – Alpage, INRIA / Paris 7, 30 rue du Ch. des rentiers, 75013 Paris, France & Fac. des Lettres, Uni. de Ljubljana, Aškerceva 2, 1000 Ljubljana, Slovénie.
52. Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks - Exploiting the UMLS Metathesaurus in the Ontology Alignment Evaluation Initiative, Department of Computer Science, University of Oxford.
53. The UMLS® Reference Manual.

54. DOMINGUES ALVES FERREIRA DA SILVA Catarina Eufémia - Découverte de correspondances sémantiques entre ressources hétérogènes dans un environnement coopératif - l'UNIVERSITE CLAUDE BERNARD - LYON 1
55. Graphes et Appariement d'Objets Complexes - Actes du 2nd Atelier GAOC.
56. Warith-Eddine DJEDDI - Alignement sémantique des ontologies de grande Taille, Université Badji Mokhtar –Annaba- Faculté des sciences de l'ingénieur Département d'informatique.
57. Jean François Djoufak Kengue, Jérôme Euzenat et Petko Valtchev - Alignement d'ontologies dirigé par la structure- LATECE, Université du Québec à Montréal
58. Anthony Ventresque - Une mesure de similarité sémantique utilisant des résultats de psychologie, Laboratoire d'Informatique de Nantes Atlantique (LINA).
59. Roua Essayeha, Mourad Abeda - Towards ontology matching based system through terminological,structural and semantic level - LAMIH, University of Valenciennes and Hainaut-Cambrésis, Le Mont Houy, 59313 Valenciennes cedex 9,France.
60. - Thabet Slimani, Boutheina Ben Yaghlane, Khaled Mellouli - Une extension de mesure de similarité entre les concepts d'une ontologie - LARODEC, ISG de tunis, 41 Bouchoucha, Bardo. Tunisia.
61. Sami Zghal - Contributions à l'alignement d'ontologies OWL par agrégation de similarités - DEA en Modélisation et Informatique de Gestion I.S.G. Tunis – Tunisie
62. Noy and Musen (2001) Anchor-PROMPT: Using non-local context for semantic matching. - IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US).
63. AnHai Doan, Jayant Madhavan, Pedro Domingos et Alon Halevy - Learning to Map between Ontologies on the Semantic Web - Computer Science and Engineering University of Washington, Seattle, WA, USA.
64. Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich - S-Match: an algorithm and an implementation of semantic matching - Dept. of Information and Communication Technology University of Trento, 38050 Povo, Trento, Italy.
65. Hong-Hai Do, Erhard Rahm - COMA : A system for flexible combination of schema matching approaches - University of Leipzig.
66. Jérôme Euzenat, Petko Valtchev - Similarity-based ontology alignment in OWL-Lite -
67. Magaly Douyère, Lina F. Soualmia, Aurélie Névéol, Alexandrina Rogozan, Badisse Dahamna, Jean-Philippe Leroy, Benoît Thirion, Stefan J. Darmoni - Enhancing the MeSH thesaurus to retrieve French online health resources in a quality-controlled gateway , CISMeF, Rouen University Hospital
68. Monika Lanzenberger, Jennifer Sampson - AlViz : A Tool for Visual Ontology Alignment - Vienna University of Technology
69. Garey, M. R., Johnson, D. S. (1979) Computers and Intractability : A Guide to the Theory of NPCompleteness. Freeman & Co, New York, 1979.

70. Levi, G., (1972) A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs. Calcolo, Vol. 9, pp. 341-354, 1972.
71. Durand, P. J., Pasari, R., Baker, J. W. et Tsai, C. (1999) An Efficient Algorithm for Similarity Analysis of Molecules . Internet Journal of Chemistry, vol. 2, 1999.
72. Bron, C. et Kerbosch, J. (1973) Finding All the Cliques in an Undirected Graph. Communication of the Association for Computing Machinery 16, pp. 575- 577,1973.
73. Mestiri Mohamed Amine - Vers une approche web sémantique dans les applications de gestion de conférences Maître ès sciences (M.Sc.) Université Laval.
74. Ahcène Benayache (2005) Construction d'une mémoire organisationnelle de formation et évaluation dans un contexte e-learning : Le projet MEMORAe.
75. Ulrike Sattler (Avril 2003) - Description Logics for Ontologies.
76. Natalya F. Noy and Deborah L. McGuinness. (2003) « Ontology Development 101: A Guide to Creating Your First Ontology ».
77. Sami Zghal (2010) Contribution à l'alignement d'ontologies OWL par aggregation de similarités – Université de Tunis El Manar.
78. Warith-Eddine Djeddi (2013) Alignement sémantique des ontologies de grande taille– Université Badji Mokhtar, Annaba.
79. Bach Thanh Le, Rose Dieng-Kuntz (2007) - A Graph-Based Algorithm for Alignment of OWL Ontologies. 2007 IEEE/WIC/ACM International Conference on Web Intelligence.