وزارة التعليم العالي والبحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTE GENIE ELECTRIQUE ET INFORMATIQUE

DEPARTEMENT INFORMATIQUE

Domaine : Mathématique et informatique

Filière : informatique

Spécialité : ingénierie des systèmes d'information

Mémoire en vue de l'obtention

Du diplôme de master II

Promotion : 2016/2017

Techniques et outils pour l'alignement d'ontologies

Réalisé par :

SLIMANE AMIAR

Encadré par :

Professeur MALIK SIMOHAMED

Jury

Mme SINI : Présidente

Mme ILTACHE : Examinatrice

Remerciements

Je présente mes respects et mes remerciements aux membres du jury qui nous ont fait l'honneur d'avoir accepté d'évaluer ce travail ainsi que notre encadrant monsieur le Professeur MALIK SI MOHAMED.

Je remercie vivement notre encadrant, monsieur le Professeur MALIK SI MOHAMED, d'avoir accepté de diriger ce travail, de nous avoir soutenu et encouragé, pour son aide précieux, pour le temps qu'il a pris pour échanger et pour nous corriger, et de ses guidelines.

Je remercie également la doctorante L. MESTOUR qui travaille sur le même thème, pour sa collaboration dans ce travail.

Je tiens à remercier également ma famille pour son soutien et encouragements, ainsi que toute autre personne qui a contribué de près ou de loin au bon déroulement de ce projet.

Merci ^_^.

Table des matières

Introduction générale

Au cours des années précédentes, un effort a été fait pour trouver des moyens de représenter les données d'une manière plus organisée et plus liée qui soit mieux adaptée aux systèmes informatiques. Les ontologies ont été considérées comme un moyen potentiel de représentation de l'information de manière structurée. Ceci est obtenu grâce à l'introduction d'une couche sémantique de niveau supérieur pour les descriptions de données. Bien que les ontologies aient été considérées comme une solution possible au problème de l'hétérogénéité des données, maintenant il existe un problème d'hétérogénéité entre les ontologies. Une approche qui a été proposée pour répondre à certains des défis de l'hétérogénéité sémantique entre les ontologies est connue sous le nom d'alignement ontologique.

Notre travail consiste à réaliser un système capable de résoudre ce problème d'hétérogénéité entre les ontologies, en se basant sur des techniques de comparaison de chaine de caractère, et de technique sémantique par utilisation de ressource externe, cette solution que nous allons proposer sera applicable dans le domaine médical.

La plupart des systèmes d'alignement d'ontologies existant utilisent généralement une seule méthode, et même ceux utilisant plus d'une, ils ne permettent pas de faire un choix ou d'avoir une souplesse dans la configuration du système, notre but aussi est de permettre cette configuration afin de rendre le système adaptable à plusieurs cas, et ça va permettre de retrouver la bonne configuration pour chaque scénario.

Ainsi le choix ne se limitera pas seulement aux méthodes à utiliser, mais même aux relations à identifier, notre système permettra d'identifier des relations d'équivalence sémantiques et non sémantiques, des relations de généralisations et de spécialisations.

1. Ontologies

1.1. Introduction

Le mot ontologie a été initialement introduit dans la philosophie où Aristote a défini l'ontologie comme « la science qui étudie l'être en tant qu'être », C'est-à-dire l'étude des différentes modalités et propriétés des choses en raison de leur nature même. Indépendamment de toute autre considération. Par exemple, il est même possible d'étudier l'ontologie d'une chose fictive (inexistante).

En informatique une ontologie est une forme de représentation de connaissance qui décrit une partie de notre monde, c'est un moyen de modéliser formellement la structure d'un système, c'est-à-dire les entités pertinentes et les relations survenant de l'observation de ce système et qui sont utiles à nos fins, elles sont utilisées dans plusieurs domaines tel que l'intelligence artificielle, le Web sémantique, l'ingénierie des systèmes, l'ingénierie des logiciels, l'informatique biomédicale etc.

1.2. Définition

Plusieurs définitions existent, En se basant sur [1] une ontologie consiste à définir de manière formelle et explicite (claire et précise) un ensemble de classes (concepts), propriétés (attributs), types de relations et entités (individus, instances) lisible par machine exprimant une vision partagée entre plusieurs parties (consensus).

Représentés par des prédicats unaires et binaires, une ontologie est basée sur une hiérarchie de concepts de généralisation/spécialisation, c'est-à-dire d'une taxonomie.

Les ontologies ont été utilisées avec succès afin de résoudre des problèmes tels que l'interopérabilité et l'hétérogénéité, dérivant de la gestion des connaissances partagées et distribuées et l'intégration efficace de l'information dans les applications.

Le mot ontologie seul réfère à une discipline philosophique et une ontologie designe son utilisation en informatique.

1.3. Composants d'une ontologie

Concepts (ou classes) :

Un concept représente les abstractions utilisées pour décrire les objets dans le monde. Il est décrit par un terme (un symbole), c'est l'élément fondamental du domaine, il est représenté dans des graphes hiérarchiques, La hiérarchie des classes est exprimée en classes de niveau supérieur (superclasse ou classe parent) et classes de niveau inférieur (sous-classe ou classe enfant), par exemple, une entreprise peut être représentée comme une classe avec des sous-classes telles que les départements et les employés.

Relations :

Elles sont utilisées pour exprimer des relations entre deux concepts dans un domaine donné. Plus précisément, une relation décrit le lien entre le premier concept, représenté dans le domaine, et le

second, représenté dans une portée. Par exemple, la déclaration (Employé, Travail, Faculté) la relation est « Travail », la notion de « Employé » représente son domaine et « Faculté » représente sa portée.

Axiomes :

Ensemble d'assertions, considérées comme vrai, Ils peuvent être inclus dans une ontologie à plusieurs fins, comme la définition de la signification des composants de l'ontologie, la définition de contraintes complexes sur les valeurs des attributs, les arguments des relations, etc. les axiomes sont généralement exprimés en utilisant des langages basés sur la logique tels que la logique de premier ordre , qui permettent de vérifié l'exactitude des informations spécifiées dans l'ontologie ou de déduire de nouvelles information.

Instances (individu ou objet) :

Ce sont des représentations spécifiques des éléments des classes, par exemple une classe « étudiant », chaque étudiant est une instance de cette classe.s

1.4. Critères de conception d'une ontologie

- **Déformation d'encodage minimale :**
  La conceptualisation doit être spécifiée au niveau du savoir sans être dépendant d'un langage ou d'un outil particulier.
- **Engagement ontologique minimal :**
  Une ontologie devrait définir un vocabulaire minimal suffisant pour décrire un domaine, et partagé la connaissance liée à ce domaine.
- **Clarté :**
  Une ontologie devrait communiquer efficacement la signification voulue des termes définis, de manière aussi objective que possible (indépendante du contexte). Une définition doit de plus être complète (c'est-à-dire définie par des conditions à la fois nécessaires et suffisantes) et documentée en langage naturel
- **La cohérence :**
  Une ontologie est dite cohérente si toutes les inférences conformes aux définitions, autrement dit si elle est logiquement et formellement compatible.
- **Extensibilité :**
  L'ajout de nouveaux concepts devraient être faciles et ne nécessitent pas des modifications des fondations de l'ontologie.

1.5. Langages d'ontologie

Récemment, différent langages spécialisées dans l'expression d'ontologies ont évolué, principalement dans le cadre du web sémantique. En plus de couvrir les caractéristiques essentielles d'une ontologie en tant qu'état de représentation de la connaissance, elles tiennent également compte des caractéristiques du Web, telles qu'une identification unique d'entités via une IRI (Internationalized Resource Identifier) ou des formats de sérialisation XML[1](Extensible Markup Language), JSON[2](JavaScript Object Notation), etc. Nous allons voir brièvement les langages d'ontologie du Web sémantique les plus répandues, la figure suivant représente l'architecture du Web sémantique :

[1] https://www.w3.org/XML/s
[2] http://www.json.org/s

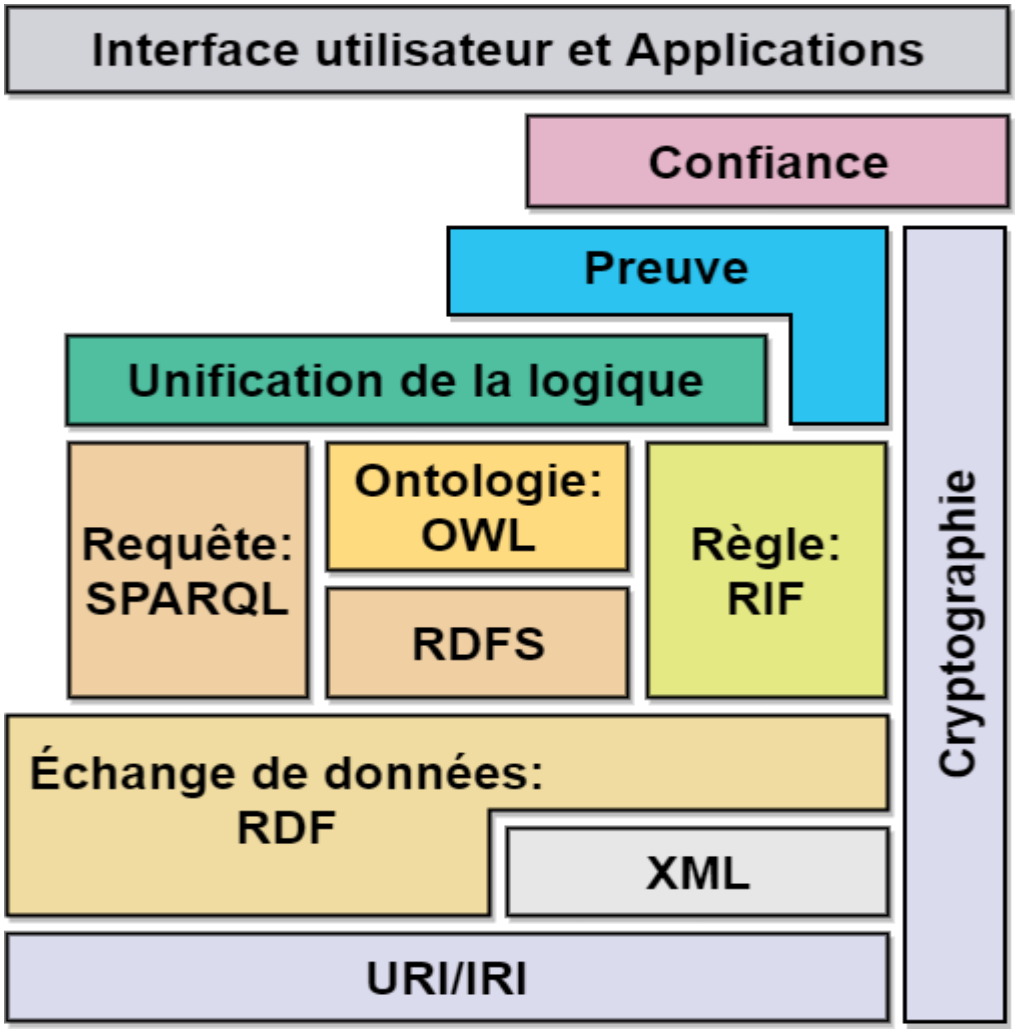


Figure 1 pile du web sémantique,[2] *traduite par S.AMIAR*

Chaque couche est une évolution de la couche en dessous et non pas un remplacement.

1.5.1. RDF

(RDF) Resource Description Framework est une recommandation du W3C comme un modèle standard d'échange de donnée sur le web et un langage d'expression de graphs de données dirigé sous forme de triples sujet-prédicat-objet respectivement (ressource, propriété, valeur), une ressource est identifiable par une IRI (Internationalized Resource Identifier), un objet peut être un sujet ou un littéral ou d'autre instance de triples (Sujet, Prédicat, Objet), une propriété c'est une relation entre un sujet et un objet.

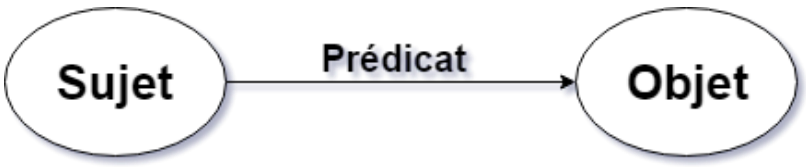
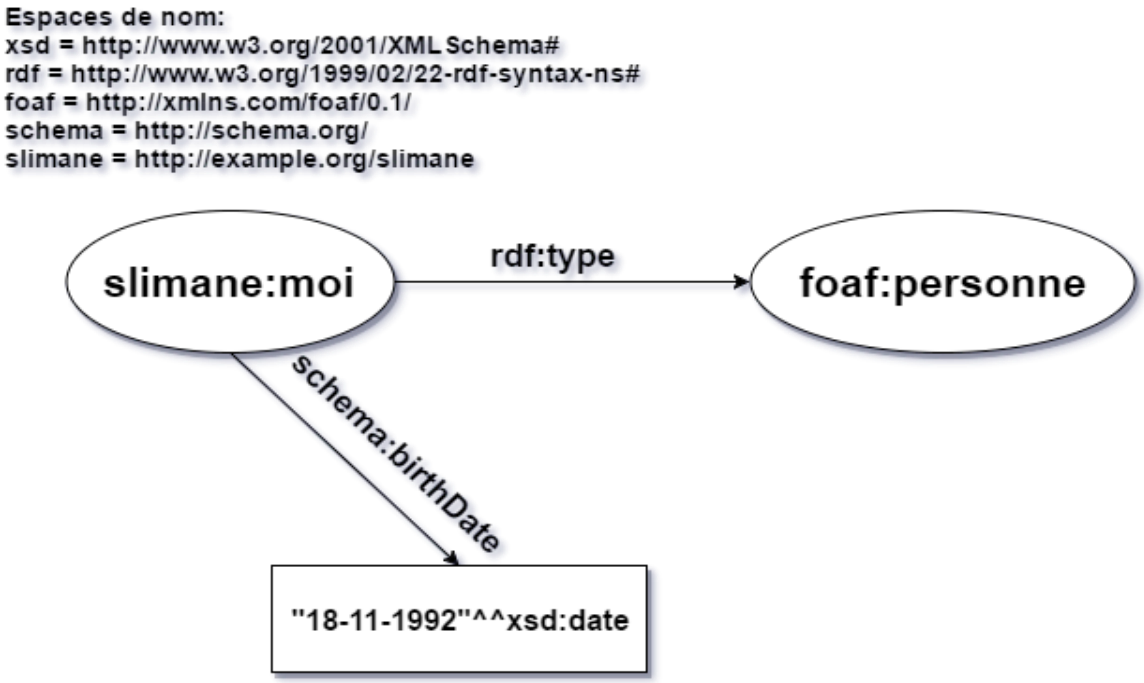


Figure 2 triple sujet-prédicat-objet

- Le sujet est une IRI ou un nœud vide
- Le prédicat est une IRI
- L'objet est une IRI ou littéral (chaine de caractère, nombre, date …) ou nœud vide

Voici un exemple me décrivant en tant qu'une personne ayant né le 18-11-1992 :



Un document RDF est un document qui encode un graph RDF ou un ensemble de données RDF dans une syntaxe RDF concrète qui peut être soit en une sérialisation RDF/XML, RDFa, N-Triples, Turtle, JSON-LD, TriG, N-Quads

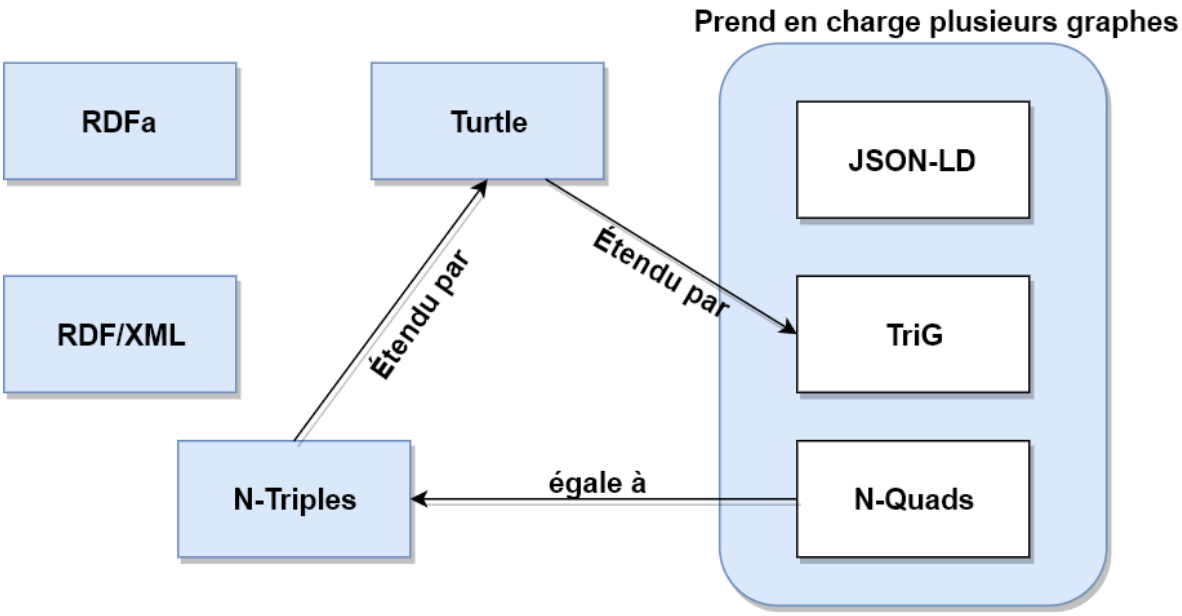


Figure 3 Formats de sérialisation RDF 1.1 [2], traduite par S.AMIAR

Cette utilisation sous forme de graph permet de profiter de la force de la théorie des graphs cependant RDF ne permet pas de représenter une ontologie vue son expressivité faible, alors une couche de plus a été porté sur RDF en lui introduisant des éléments de plus dans RDFS (RDF schéma).

1.5.2. RDFS

RDF Schema est une extension sémantique de RDF. Il fournit des mécanismes pour décrire des groupes de ressources connexes et des relations entre ces ressources [4]. RDF Schema définit des entités tell que rdfs:class, rdfs:subclass, rdfs:subproperty, rdfs:domain, et dfs:range, permettant de modéliser des classes et des propriétés avec une restriction du domaine et de portée, cependant il ne permet pas d'exprimé l'exclusion et la négation , et il limite l'axiomatisation aux restriction, pour cela il est considéré comme un langage ontologique simple, La combinaison de RDF et RDF Schema est appelée RDF(S).

1.5.3. OWL

OWL (Web Ontology Language) étend RDFS et mettent l'accent sur le soutien d'une inférence logique plus riche, c'est est un successeur de deux autres langages OIL et DAML+OIL. OWL est disponible en plusieurs variantes avec une expressivité croissante :

- **OWL-Lite** est la version la plus simple du langage OWL, Des algorithmes décidables existent pour la totalité de OWL-Lite, Il correspond à la logique de descriptions SHIF.
- **OWL-DL** est concentrée Sur le formalisme de la logique de description (DL), il correspond à la logique de descriptions SHOIN(D), il est décidable.
- **OWL-Full** possède une couche propre sur RDF (S), permettant des fonctionnalités de la méta-modélisation et de la réification, Il n'existe aucun algorithme d'inférence décidable pour OWL-Full

OWL a introduit les fonctionnalités suivant :

Pour **OWL-lite**

- Restriction de propriétés.
- Intersection de classe.
- (In)Égalité.
- Cardinalité restreinte.
- Versions.
- Caractéristiques de propriété.
- Informations d'en-tête.
- Propriétés d'annotation.

Pour **OWL-DL** et **OWL-FULL** en plus de celle de **OWL-lite** on leurs a ajouté les fonctionnalités suivantes :

- Axiomes de classe.
- Cardinalité arbitraire.
- Combinaisons booléennes des expressions de classe.
- Informations sur la valeur d'une propriété.

1.5.4. OWL2

OWL 2 est une extension et une révision du Langage d'ontologie Web OWL développé par le Groupe de travail sur l'ontologie Web du W3C, Il est compatible avec OWL, et il offre de nouvelles fonctionnalités tel que :

- Clés : Une collection de propriétés (données ou objet) peut être affectée comme une clé à une expression de classe.
- Chaînes de propriété.
- Types de données plus riches, porté de données.
- Restrictions de cardinalité qualifiées.
- Des propriétés asymétrique, réflexive et disjoint.
- Capacités d'annotation améliorées y compris l'annulation d'entités OWL spécifiques, et des annotations pour axiomes et ontologies.
- Importation : permet d'importer d'autres ontologies afin d'accéder à leurs entités, expressions et axiomes, fournissant ainsi l'installation de base pour la modularisation de l'ontologie.

1.5.4.1. Structure

La figure suivante donne un aperçu de ce langage montrant ses blocs de constructions principales et leur relation entre eux. L'ellipse au centre représente la notion abstraite d'une ontologie, qu'on peut penser soit comme une structure abstraite, soit comme un graph RDF. Au sommet, différentes syntaxes concrètes peuvent être utilisées pour sérialiser et échanger des ontologies. En bas sont les deux spécifications sémantiques qui définissent le sens des ontologies OWL 2 [5].

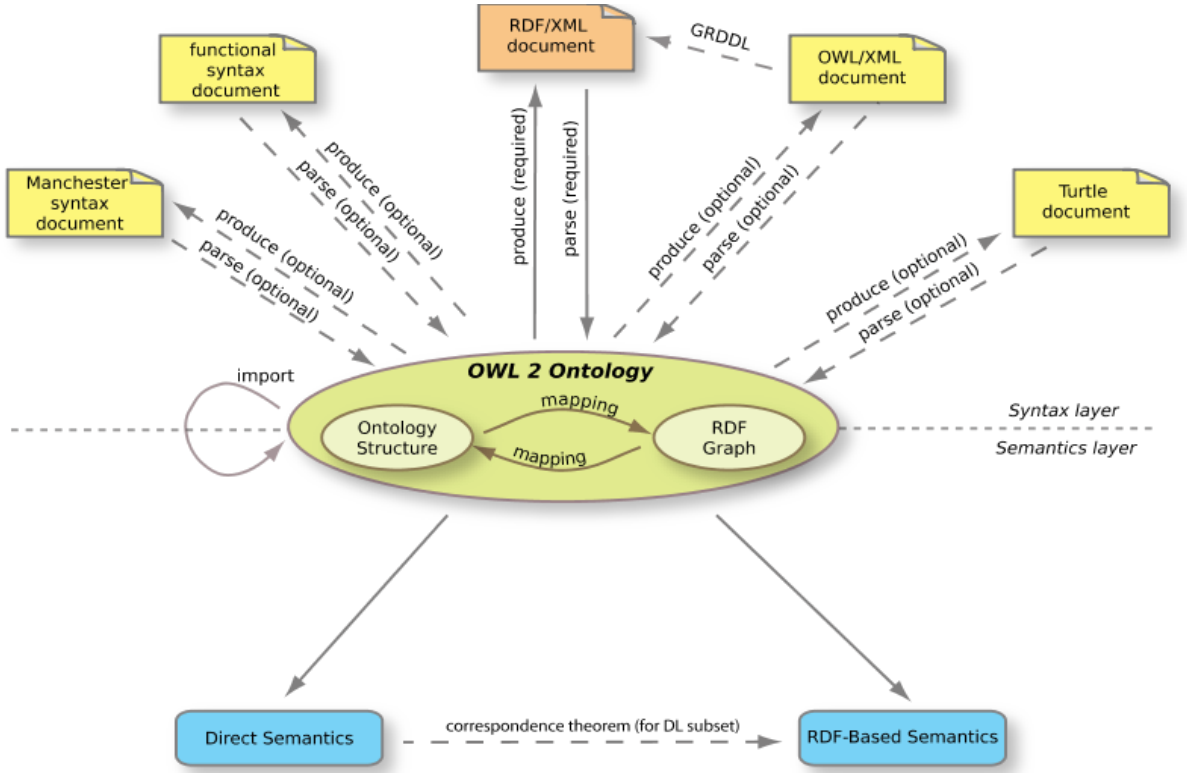


Figure 4 Structure de OWL 2 [5]

1.5.4.2. Profiles

OWL 2 est composé de trois profiles (sous langages) OWL 2 EL, OWL 2 QL, and OWL 2 RL Chaque profil est défini comme une restriction syntaxique de la spécification structurelle OWL 2

- **OWL 2 EL** permet des algorithmes de temps polynomiaux pour tous les types d'inférence standard. Il a été conçu comme un langage particulièrement adapté pour définir des ontologies qui incluent des hiérarchies de classes et de rôles très grands tout en n'utilisant qu'une quantité limitée de fonctionnalité OWL. Une application d'exemple typique est SNOMED CT une grande ontologie médicale qui définit plus de trois cent vingt mille classes en 2017.
- **OWL 2 QL** Permet la mise en œuvre d'une réponse de requête conjoncturelle à l'aide de systèmes de base de données relationnels classiques, Il est particulièrement adapté aux applications où des ontologies relativement légères sont utilisées pour organiser un grand nombre d'individus et où il est utile ou nécessaire d'accéder aux données directement via des requêtes relationnelles (par exemple, SQL).
- **OWL 2 RL** Permet la mise en œuvre d'algorithmes de raisonnement de temps polynomial à l'aide des technologies de base de données étendues par règles fonctionnant directement sur des triples RDF, Il est particulièrement adapté aux applications où des ontologies relativement légères sont utilisées pour organiser un grand nombre d'individus et où il est utile ou nécessaire d'opérer directement sur des données sous la forme de triples RDF.

1.5.4.3. Ontologie en OWL 2

Une ontologie en OWL 2 se compose des trois catégories syntaxiques suivantes [6]:

- **Les entités** telles que les classes, les propriétés et les individus, sont identifiées par les IRI. Ils forment les termes primitifs d'une ontologie et constituent les éléments fondamentaux d'une ontologie. Par exemple, une classe a:Personne peut être utilisée pour représenter l'ensemble de toutes les personnes. De même, la propriété objet a:parentOf peut être utilisée pour représenter la relation parent-enfant. Enfin, l'individu a:nomPersonne peut être utilisé pour représenter une personne particulière appelée « nomPersonne ».
- **Les expressions** représentent des notions complexes dans le domaine décrit. Par exemple, une expression de classe décrit un ensemble d'individus en termes de restrictions sur les caractéristiques des individus.
- **Les axiomes** sont des affirmations qui sont affirmées comme étant vraies dans le domaine décrit. Par exemple, en utilisant un axiome de sous-classe, on peut indiquer que la classe a:Etudiant est une sous-classe de la classe a:Personne.

1.5.5. SPARQL

SPARQL (Simple Protocole AND RDF Query Language) est un langage de requête standardisée pour les graphs RDF. Il est similaire au langage de requête SQL, il permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF avec un certain nombre d'ajouts importants et puissants, y compris la capacité de filtrer les résultats et de construire de nouveaux graphs en fonction des requêtes.

1.6. Types d'ontologies

Il existe de nombreux types d'ontologies, selon différente dimension :

1.6.1. Portée

selon leur niveau de généralité, Guarino [7] considère que les types d'ontologie suivants existent :

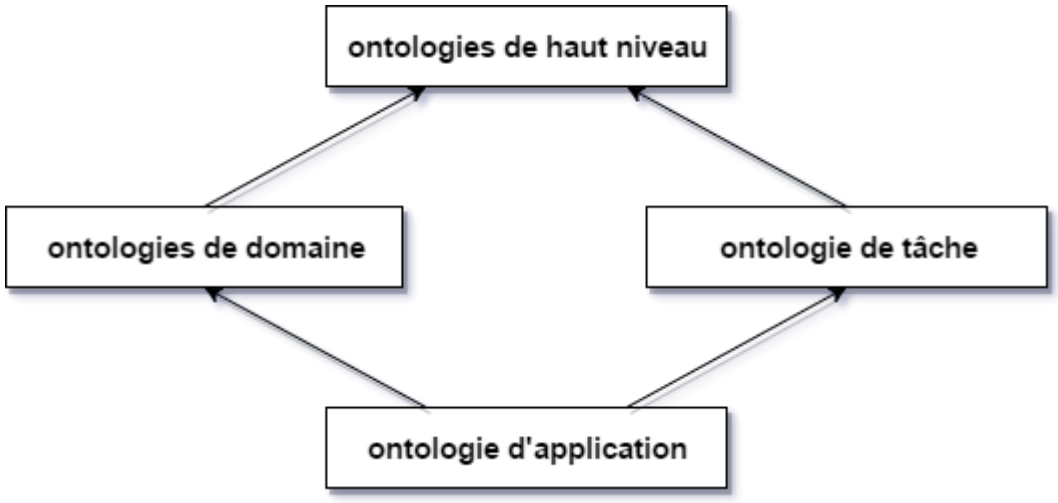


Figure 5 : les types d'ontologies d'après Guarino [6] traduite par S.AMIAR

Les ontologies de haut niveau

Décrit des concepts très abstraits et généraux qui peuvent être partagés dans de nombreux domaines et applications, telles que « objet physique » ou « objet d'information ». Ils sont indépendants d'un domaine ou d'un problème spécifique, souvent, elles sont utilisées pour guider simplement la conception d'ontologies de domaine, leur but est d'unifier les critères entre les grandes communautés d'utilisateurs, un exemple concret est l'ontologie YAMATO[3] qui a été développé par Riichiro Mizoguchi pour couvrir trois fonctionnalités qui sont :

- Une description avancée de la qualité.
- Une ontologie de représentation.
- Une description avancée des processus et des événements.

Mieux que les ontologies existantes.

Les ontologies de domaine

Représentation de concepts dans un domaine générique (par exemple, systèmes d'information ou médecine), et peuvent généralement être considérés comme des spécialisations des concepts introduits d'ontologies de haut niveau. Un exemple concret est le modèle fondamental d'Anatomie (FMA)[4] développé à l'Université de Washington dans le domaine médical de l'anatomie.

Les ontologies de tâche

Représente des concepts spécifiques à une tâche ou une activité générique (par exemple, développement ou diagnostic), au moyen de la spécialisation des concepts introduits d'ontologies de haut niveau.

Les ontologies d'application

Elles décrivent les concepts appartenant simultanément à un domaine et à une tâche spécifique. Elle sont souvent une spécialisation des ontologies de domaine et de tâches et correspondent aux rôles joués par les entités de domaine lorsqu'ils effectuent certaines tâches, par exemple la « fièvre » comme symptôme de la tâche du diagnostic dans le domaine médical, un exemple d'une ontologie d'application l'ontologie du facteur expérimental (EFO)[5] fournit une description systématique de nombreuses variables expérimentales , elle combine des parties de plusieurs ontologies biologiques, telles que l'anatomie UBERON, les composés chimiques ChEBI et l'ontologie cellulaire.

1.6.2. Niveau de formalisme

Le degré de formalisme d'une ontologie détermine dans quelle mesure elle est axiomatisée au moyen d'énoncés logiques sur le domaine. Les ontologies légères ne possèdent pas ou possèdent seulement quelques axiomes limitant l'utilisation des entités dans leur signature. D'autre part, les ontologies lourdes se caractérisent par une axiomatisation étendue pour interconnecter les éléments

[3] http://download.hozo.jp/onto_library/upperOnto.htm
[4] http://si.washington.edu/projects/fma
[5] http://www.ebi.ac.uk/efo/

de signature d'une manière sophistiquée, de sorte que presque toutes les entités s'accompagnent de nombreux axiomes qui limitent leur utilisation et soutiennent le raisonnement à leur sujet.

Uschold et Grüninger [8] distinguent quatre type d'ontologies selon le niveau de formalisme :

Fortement informelles

L'ontologie est exprimée en langage naturel.

Semi informelle

L'ontologie est exprimée sous une forme restreinte et structurée du langage naturel.

Semi formelle

L'ontologie est exprimée dans un langage artificiel formellement définie.

Rigoureusement formelles

L'ontologie est exprimée avec des termes méticuleusement définis avec une sémantique formelle, des théorèmes et des preuves de propriétés telles que la solidité et l'intégralité.

Note : ces ontologies doivent être traitables par machine sinon elles ne sont pas considérées comme des ontologies.

McGuinness [9] a introduit un continuum ontologique spécifiant un ordre total entre les types de modèles courants :

Modèles informel Ordonnés dans un ordre croissant comme suivant :

Catalogues, glossaires, thésaurus et taxonomies informelles.

Un catalogue est une ontologie qui a des définitions de classe sans étiquettes ni commentaires.

Un glossaire est une ontologie qui a des définitions de classe avec des étiquettes ou des commentaires.

Un thésaurus est une ontologie qui a des classes et des assertions équivalentes de classe.

Une taxonomie informelle est une ontologie comprenant des classes et des relations de sous-classe hiérarchiques strictes et d'informations entre les classes.

Modèles formel en partant des Taxonomies formelles en ajoutant plus de sémantique dans l'ordre suivant :

Taxonomies formelles, instance formelles, les propriétés/frames, les restrictions de valeur, les contraintes logiques générales, la disjonction.

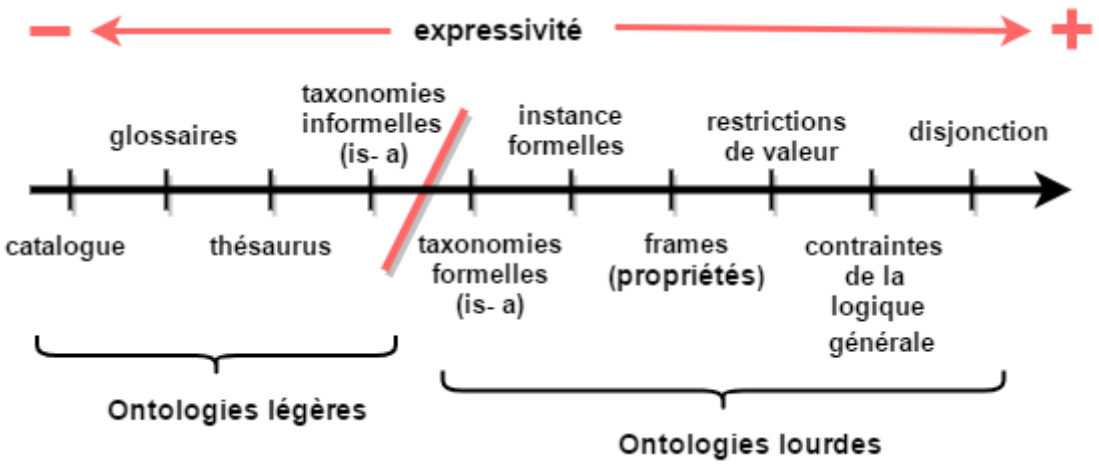


Figure 6 types d'ontologies par ordre de formalité [10] traduite par S.AMIAR

Ontologies de base (Core-ontologies)

Elles fournissent des concepts et des propriétés très basiques (les plus importants) d'un domaine de connaissance. Par exemple, le réseau sémantique dans UMLS[6] (un compendium de nombreux vocabulaires contrôlés dans les sciences biomédicales (créé en 1986). Il fournit une structure de cartographie parmi ces vocabulaires et permet ainsi de traduire parmi les différents systèmes de terminologie, Il peut aussi être considéré comme un thésaurus complet et l'ontologie des concepts biomédicaux. UMLS fournit également des installations pour le traitement du langage naturel. Il est destiné à être utilisé principalement par des développeurs de systèmes en informatique médicale.) contient des concepts médicaux généraux tels que la maladie, le syndrome, etc. ce qui constitue une ontologie médicale fondamentale.

Ontologies de représentation (Meta-ontologies)

Elles fournissent des entités de représentation sans indiquer ce qui devrait être présenté, c'est-à-dire qu'ils ne s'engagent pas sur un domaine particulier. Un exemple est les langages Web sémantiques W3C RDFS[7] et OWL[8].

1.7. Utilisation des ontologies

Il existe de nombreuses applications potentielles d'ontologies, Fikes [11] a énuméré l'importance des ontologies dans quatre domaines d'application clés: collaboration, interopérabilité, éducation et modélisation.

- **Collaboration** Les ontologies offrent une base de connaissance comme référence commune et partagé entre les membres d'une équipe qui travaille sur un projet, ou entre agents intelligents qui se communique, ce qui rend l'échange de connaissances plus efficace vu qu'ils se basent tous sur le même modèle d'ontologie.
- **Interopérabilité** Les ontologies facilitent l'intégration de l'information, en particulier dans les systèmes distribués, ces systèmes peuvent avoir besoin d'accéder à différentes sources de connaissances afin d'obtenir toutes les informations pertinentes disponibles, l'utilisation de la

[6] https://www.nlm.nih.gov/research/umls/
[7] https://www.w3.org/TR/rdf-schema/
[8] https://www.w3.org/OWL/

même ontologie par ces différent parties permet la conversion des données et l'intégration de l'information plus facilement et d'une manière automatique.

- **Éducation** Les ontologies peuvent être utilisées comme moyen de publication du savoir d'un domaine particulier et comme une source de référence, ce qui permet aux experts de partagé leur compréhension de la conceptualisation et la structure d'un domaine.
- **Modélisation** Les ontologies peuvent être utilisées comme module de connaissance prédéveloppée réutilisable dans les systèmes de modélisation d'applications.

Uschold et Gruninger [12] on définit trois cas d'utilisation des ontologies :

Communication, les ontologies sont utilisées en communication afin de réduire la confusion conceptuelle et terminologique en fournissant un unifiant Framework au sein d'une organisation selon différents aspect, par exemple elle peut être utiliser comme un modèle de normalisation, Dans n'importe quel système de logiciel intégré à grande échelle, où différentes personnes doivent avoir une compréhension partagée du système et de ses objectifs En utilisant une ontologie, nous pouvons construire un modèle normatif du système .

Interopérabilité L'échange de données entre différents utilisateurs ou systèmes lorsque différents outils logiciels sont utilisés, souvent implique des problèmes d'interopérabilité. Vu la possibilité d'utilisé différent termes et langues, En ce sens, les ontologies peuvent servir comme un langage de communication commun entre les différentes utilisateurs et systèmes c'est-à-dire qu'elles peuvent être utilisées pour soutenir la traduction entre différentes langues et représentations, La figure suivante montre l'efficacité d'utilisé une ontologie d'échange comme traducteur pour toutes les parties impliquées(interlingua), que d'avoir un traducteur pour chaque partie, ce que réduit le nombre de traduction entre les paires.

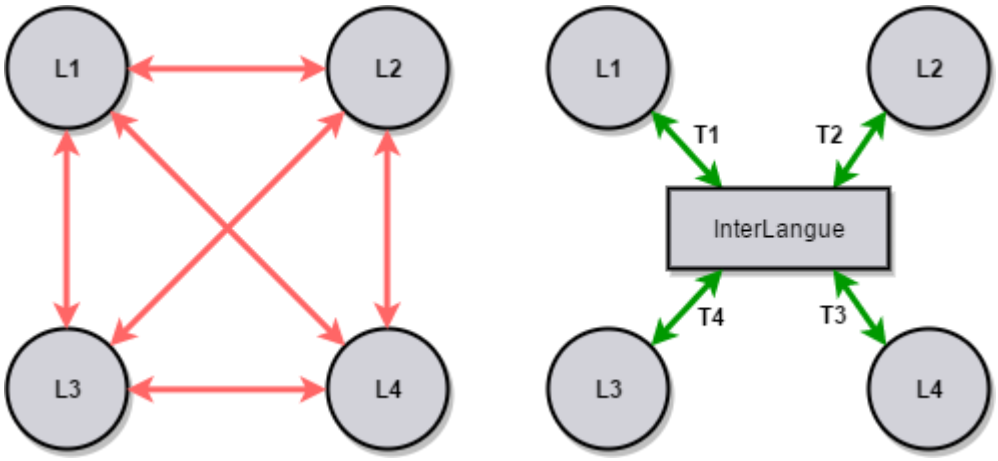


Figure 7 : interopérabilité par utilisation d'ontologie comme interlingua[8, p. 10]

Ingénierie de systèmes : Les ontologies peuvent être utilisées pour la conception et le développement de système logiciel notamment : **Spécification, Fiabilité, réutilisabilité, etc.**

1.8. Raisonnement

Le raisonnement dans les ontologies est l'une des raisons pour lesquelles une spécification doit être formelle. Les raisonneurs (moteurs d'inférence) tirent de nouveaux faits des ontologies existantes et vérifient l'intégrité des ontologies. Cela se fait généralement en appliquant des règles simples du raisonnement déductif. Les services standards offerts par un raisonneur sont :

Raisonnement TBox (base de règles) :

- **Satisfaction du concept**. Vérifie si une description de concept est satisfaisante c'est-à-dire il vérifie si une classe peut être instancié ou avoir des individus. Si une classe, en raison de sa définition, n'est pas en mesure de contenir des individus, cela signifie que toute l'ontologie n'est pas cohérente.
- **Subsomption de concept, classification**. Il peut calculer l'arbre hiérarchique des classes / sous-classes. Cet arbre peut être utilisé pour évaluer les requêtes sur une classe et renvoyer indifféremment ses superclasses ou sous-classes. Par exemple, pour vérifier si le concept (EtudiantA, est, MajeurDePromotion) est impliqué dans l'ontologie O.

Raisonnement ABox (base de faits) :

- **Vérification de cohérence**. Assure que l'ontologie ne contient pas de faits contradictoires. Selon la terminologie DL, ce service assure que l'ensemble des faits qui se réfèrent à des individus spécifiques (Assertional Box ou ABox) sera cohérent par rapport à l'ensemble des axiomes qui décrivent les concepts généraux d'un domaine (Terminological Box ou TBox).
- **Vérification d'instance (Réalisation)**. Il peut calculer les classes auxquelles appartient un individu. En d'autres termes, il peut calculer le type individuel. La réalisation ne peut avoir lieu qu'après la classification, car la hiérarchie conceptuelle est nécessaire.

En plus de ces quatre services, un raisonneur offre le service de **réponse aux requêtes** qui renvoie un ensemble de réponses adéquat à une requête. Par exemple, pour trouver tous les étudiants du cycle master spécialité ingénierie des systèmes d'information.

En raison des différents algorithmes mis en œuvre pour le raisonnement de description logique. Tous les raisonnants ne peuvent évaluer toutes les inférences possibles. Une caractéristique commune entres les raisonneurs est le raisonnement ABOX, le raisonnement des individus qui couvre la vérification de l'instance, la réponse aux requêtes conjoncturelles et la vérification de la cohérence.

1.9. Raisonneurs

Parmi les raisonneurs existant les plus populaires on trouve :

FaCT++[9] : est un raisonneur fort basé sur les tableaux pour la logique de description (DL) expressives. Il couvre OWL DL et OWL 2 DL (manque de support pour les contraintes clés et certains types de données) et les langages d'ontologie basés sur DL, Il peut être utilisé comme classificateur de logique de description et pour vérifier la satisfaction logique modale, il est Open Source écrit en C++.

HermiT[10] : est un raisonneur pour les ontologies écrites en utilisant OWL. Étant donné un fichier OWL, HermiT peut déterminer si oui ou non l'ontologie est cohérente, identifier les relations de subsistance entre les classes et bien plus encore. HermiT est basé sur un nouveau calcul "hypertableau" qui fournit un raisonnement efficace. HermiT utilise la sémantique directe et passe tous les tests de conformité OWL 2 pour les raisonneurs sémantiques directs.

[9] http://owl.cs.manchester.ac.uk/tools/fact/
[10] http://www.hermit-reasoner.com/

Pellet[11] : un raisonneur Open Source OWL DL pour Java, il fournit des fonctionnalités pour vérifier la cohérence des ontologies, calculer la hiérarchie de classification, expliquer les inférences et répondre aux requêtes sSPARQL.

1.10. Editeurs d'ontologie

Différent outils de création, d'édition et de visualisation d'ontologies existe, cependant nous allons seulement voir ceux qui sont d'une licence non commerciale.

RDFLib[12] : est une bibliothèque Open source du langage Python pour travailler avec RDF, elle est simple mais puissant pour représenter une ontologie sous forme de graph, elle contient des analyseurs et des sérialisateurs pour RDF / XML, N3, NTriples, N-Quads, Turtle, TriX, RDFa et Microdata, elle présente une interface de graph qui peut être sauvegardée par l'une des nombreuses implémentations de sauvegarde, elle comprend des implémentations pour un stockage en mémoire et un stockage persistant sur Berkeley DB, Une implémentation SPARQL 1.1 est incluse - prenant en charge les requêtes SPARQL 1.1 et les instructions de mise à jour, RDFLib dispose d'une architecture de plugin pour la mise en œuvre d'implémentation de sauvegarde, ainsi que des analyseurs / sérialisateurs, plusieurs autres projets existent qui étendent les fonctionnalités RDFLib tell que Rdflib-jsonld - Serializer et parser pour json-ld.

Protégé[13] : est un environnement d'édition d'ontologie riche en fonctionnalités avec un support complet pour le langage d'ontologie Web OWL 2 et des connexions directes en mémoire à des rapporteurs logiciels de description comme HermiT et Pellet. Protégé Desktop prend en charge la création et l'édition d'une ou plusieurs ontologies dans un seul espace de travail via une interface utilisateur entièrement personnalisable. Les outils de visualisation permettent une navigation interactive des relations ontologiques. Le support d'explication avancée aide à repérer les incohérences. Les opérations de ré factorisation disponibles, y compris la fusion de l'ontologie, les axiomes mobiles entre les ontologies, possibilité de renommée plusieurs entités etc.

NeOn Toolkit[14] : est un environnement d'ingénierie ontologique multiplateforme open source développé à l'origine dans le cadre du projet NeOn, il fournit un support complet pour le cycle de vie de l'ingénierie ontologique. NeOn Toolkit est basée sur la plate-forme Eclipse, un environnement de développement leader, et fournit un ensemble complet de plug-ins (actuellement disponibles 45 plug-ins) couvrant une variété d'activités d'ingénierie ontologique, y compris l'annotation et la documentation, le développement, l'ontologie humaine d'interaction, acquisition de connaissances, gestion, modélisation et personnalisation, plugins de néon, évaluation de l'ontologie, adaptation de l'ontologie, raisonnement et inférence, réutilisation.

FluentEditor[15] : est un outil pour éditer, manipuler et interroger des ontologies complexes écrites dans OWL, RDF ou SWRL. Fluent Editor est entièrement compatible avec la plupart des normes sémantiques W3C standard (OWL, RDF, SPARQL, SKOS, ...), mais a en même temps une interface utilisateur intuitive qui utilise le langage naturel contrôlé Ontorion (OCNL) qui est une Alternative conviviale au langage d'ontologie XML comme OWL ou RDF

[11] https://github.com/stardog-union/pellet
[12] https://github.com/RDFLib/rdflib
[13] http://protege.stanford.edu/
[14] http://neon-toolkit.org/wiki/Main_Page.html
[15] http://www.cognitum.eu/semantics/FluentEditor/

mais est complètement compatible avec OWL2, RDF et SWRL. De plus, l'OCNL peut également être utilisé comme langage de requête compatible avec SPARQL. Fluent Editor dispose d'outils qui aident l'utilisateur à gérer des ontologies complexes : une fenêtre de raisonnement pour interroger l'ontologie, une fenêtre SPARQL où les requêtes SPARQL peuvent être exécutées, une fenêtre de prévisualisation XML pour voir comment la phrase OCNL écrite se trouve dans son équivalent OWL, Une vue d'arbre de taxonomie et une fenêtre d'annotation. En outre, il existe deux plugins : un plugin d'interopérabilité Protegé (pour exporter, importer vers, depuis Protegé) et un plugin R qui utilise les packages ROntorion pour tracer et répertorier le contenu actuel de l'ontologie.

Eddy[16] : est un éditeur graphique pour la spécification et la visualisation des ontologies Graphol[17]. Eddy dispose d'un environnement de conception spécialement conçu pour générer des ontologies de Graphol grâce à des fonctionnalités ad hoc. Les fonctions de dessin permettent aux concepteurs de modifier facilement les ontologies dans une zone de visionnement centrale, Eddy prend également en charge les profils standard de OWL 2, c'est-à-dire OWL2 QL, OWL 2 RL et OWL 2 EL.

1.11. Base de données RDF

Redland[18] : est un ensemble de bibliothèques de logiciels C gratuits qui fournissent un support pour RDF, c'est une bibliothèque modulaires, basées sur des objets et API pour manipuler des graphs RDF, triples, URI et Literals. Stockage pour les graphes en mémoire et persistant avec Oracle Berkeley DB, MySQL 3-5, PostgreSQL, OpenLink Virtoso, SQLite, fichiers ou URI, Elle prend en charge de syntaxes multiples pour la lecture et l'écriture de RDF sous forme de syntaxes RDF / XML, N-Triples et Turtle, RSS et Atom via la bibliothèque Raptor RDF Syntax. Elle permet la consultation avec SPARQL et RDQL à l'aide de la bibliothèque de requêtes RDQ de Rasqal, l'agrégation des données et l'enregistrement du soutien de provenance avec les contextes de Redland, la liaison avec des langages tell que Perl, PHP, Python et Ruby via le package Redland Bindings. Elle offre des programmes d'utilité de ligne de commande rdfproc (RDF), Rapper (analyse) et Roqet (requête). Portable, rapide et sans fuites de mémoire connues.

AllegroGraph[19] : est une base de données Triple Store qui est conçu pour stocker des triples RDF moderne, à haute performance, il utilise efficacement la mémoire en combinaison avec le stockage sur disque, ce qui lui permet d'étendre à des milliards de quads tout en conservant des performances supérieures. AllegroGraph prend en charge SPARQL, RDFS++ et le raisonnement Prolog de nombreuses applications clientes.

Neo4j[20] : est une base de données qui permet de stocker des graphs, elle est extrêmement performante pour traiter les relations, elle possède un langage de requête puissant, Cypher, qui

[16] https://github.com/danielepantaleone/eddy
[17] http://www.dis.uniroma1.it/~graphol/
[18] http://librdf.org/
[19] https://allegrograph.com/allegrograph/
[20] https://neo4j.com/product/

permet d'interroger un graphe pour obtenir toutes sortes d'informations sur les nœuds, leurs liens et le contenu de ces derniers.

1.12. Conclusion

Nous avons vu dans ce chapitre la notion d'ontologie, sa définition, son utilisation dans plusieurs domaines, et l'ensemble de concepts et méthodes qui nous permettrons de bien manipuler les ontologies, et qui va nous permettre aussi de découvrir un domaine de l'ingénierie des ontologies qui est l'alignement d'ontologies, afin de traiter le problème d'hétérogénéité entre ontologies, et c'est ce que nous allons voir dans le chapitre qui suit.

2. Chapitre 2 : Alignement d'ontologies

2.1. Introduction

Les ontologies sont un élément essentiel dans la résolution du problème d'interopérabilité entre les systèmes hétérogènes, cependant ces ontologies sont construites par différents auteurs en utilisant différentes notations et langages ce que les rend eux-mêmes hétérogènes, et le but de l'alignement d'ontologie est de les rendre interopérable.

2.2. Définition

Un alignement d'ontologie est un ensemble de correspondances entre les entités (classes, propriétés, prédicats, etc.) formant des ontologies, et le processus d'alignement appelé *Matching* en anglais est l'action qui permet de retrouver ces correspondances qui sont des relations par exemple d'équivalence (≡), de subsomption (plus général (⊇), plus spécifique (⊆)) et de disjonction (⋁). Le schéma suivant illustre le processus d'alignement en générale :

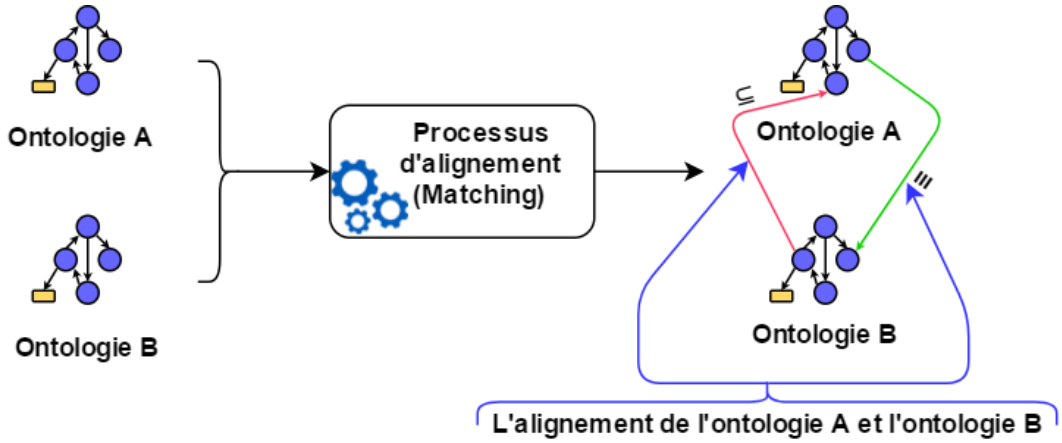


Figure 8 processus d'alignement d'ontologies

La figure suivante montre un exemple d'alignement de deux ontologies

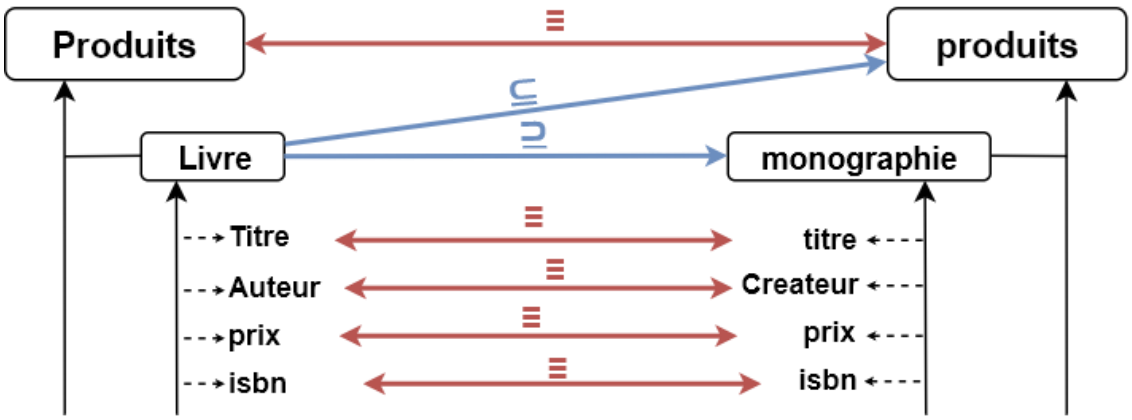


Figure 9 exemple d'alignement entre deux ontologies

2.3. Cas d'utilisations de l'alignement d'ontologies

L'alignement d'ontologie fait de plus en plus l'intérêt dans plusieurs domaines, J.Euzenat et P.Shvaiko [13] ont identifié des cas d'utilisation, parmi ces cas d'utilisation on trouvent les suivants :

- **Ingénierie ontologique :** l'activité de concevoir, de mettre en œuvre, et de maintenir des applications basées sur des ontologies est une activité dans lequel les utilisateurs sont confrontés à des ontologies hétérogènes, venu de source multiple, évolutive, distribué, ce qui rend nécessaire de les aligner.
- **Intégration de l'information :** l'intégration de l'information est la fusion d'informations provenant de sources hétérogènes avec différentes représentations conceptuelles.
- **Le Web des données :** qui est en cours d'existence à travers le « Linked Open Data » (LOD) données ouvertes et liées, est une étape majeure dans la réalisation de la vision Web sémantique. Cependant, le développement d'applications basées sur LOD est confronté à des difficultés du fait que les différents ensembles de données de LOD sont des éléments d'information peu liés, donc par l'alignement c'est possible d'identifier des correspondances pour les lier.
- **Systèmes de communication autonomes :** des applications conçues de manière indépendante et qui implique des entités autonomes peuvent se rencontrer sur un réseau, lorsque ces entités sont des logiciels, elles ont été considérées comme des agents depuis longtemps. Cependant, s'ils sont une combinaison de matériel et de logiciels, ils sont une question d'informatique ambiante. Ces entités ne peuvent pas partager une ontologie commune. Ainsi, s'ils veulent communiquer, il est utile de faire correspondre leurs ontologies.
- **Navigation et réponse aux requêtes sur le Web :** plusieurs applications qui étendent l'expérience Web au Web sémantique en utilisant des ressources telles que des ontologies formelles, fonctionnant dans un environnement ouvert, ces applications nécessitent souvent d'aligner les ressources utilisées.

Plusieurs autres cas d'utilisation existent, tels que le partage d'information Peer-to-Peer, Composition des services Web, etc.

2.4. Incompatibilités entre les ontologies

On est souvent confronté à plusieurs problèmes lorsque l'on essaie d'utiliser des ontologies développées indépendamment, ou lorsque des ontologies existantes sont adaptées à de nouvelles fins. Les incompatibilités entre les concepts dans différentes ontologies peuvent exister selon deux niveaux comme défini par M.Klein [14, pp. 33–37]:

2.4.1. Incompatibilité au niveau du langage

Les incompatibilités au niveau du langage se produisent lorsque des ontologies écrites dans différents langages ontologiques sont combinées, les points suivant montrent ces incompatibilités au niveau du langage :

- **Syntaxe :** différents langages de modélisation d'ontologie utilisent différentes syntaxes pour créer les composants d'une ontologie, ce genre d'incompatibilité est dû aux différentes structures du langage utilisé pour construire l'ontologie.

- **Expressivité du langage ou pouvoir expressif :** certains langages peuvent représenter des expressions ou des objets qui ne peuvent pas être représentés dans d'autres.
- **Sémantique des primitifs :** une différence plus subtile possible au niveau du métamodèle est la sémantique des constructions linguistiques. Malgré le fait que parfois le même nom est utilisé pour une construction du langage en deux langues, la sémantique peut différer.
- **Représentation logique :** ce niveau d'incompatibilité concerne la différence de représentation des notions logiques.

2.4.2. Incompatibilité au niveau d'ontologie

- **Niveau de contenu**
  - **Portée :** lorsque deux termes ont la même signification, mais avec des cas différents.
  - **Modèle de couverture et granularité :** ce type pourrait impliquer une incompatibilité dans le style de la modélisation ou des incompatibilités terminologiques.
- **Niveau organisationnel**
  - **Structuration de concepts** : Concerne la différence dans la conception des ontologies, par exemple une personne, peut utiliser le concept « adresse » en tant qu'entité unique alors que d'autre peuvent le séparer sur plusieurs entité « village » « commune », « daïra », « wilaya ».
  - **Termes synonymes** : les incompatibilités multilangues se produisent lorsque l'on utilise différents synonymes pour le même concept, comme dans « maison » et « logement ».
  - **Termes Homonymes :** c'est ce qu'on appelle la terminologie qui se chevauche. Par exemple, « table » peut être un meuble ou une liste d'informations disposées en colonnes et en lignes.
  - **Codage** : les incompatibilités se produisent lorsque les ontologies sont construites dans différents formats tels qu'une date représentée par « jj/mm/aaaa » ou « jj-mm-aaaa ».
  - **Paradigme** : des incompatibilités apparaissent lorsque plus d'un paradigme est utilisé pour représenter un concept, par exemple un modèle utilise des représentations temporelles basées sur une logique d'intervalle, tandis qu'un autre utilise une représentation basée sur des points.

2.5. Techniques d'alignement

2.5.1. Vue d'ensemble

La figure suivante représente l'ensemble des techniques d'alignement comme représenter par J.Euzenat et P.Shvaiko :

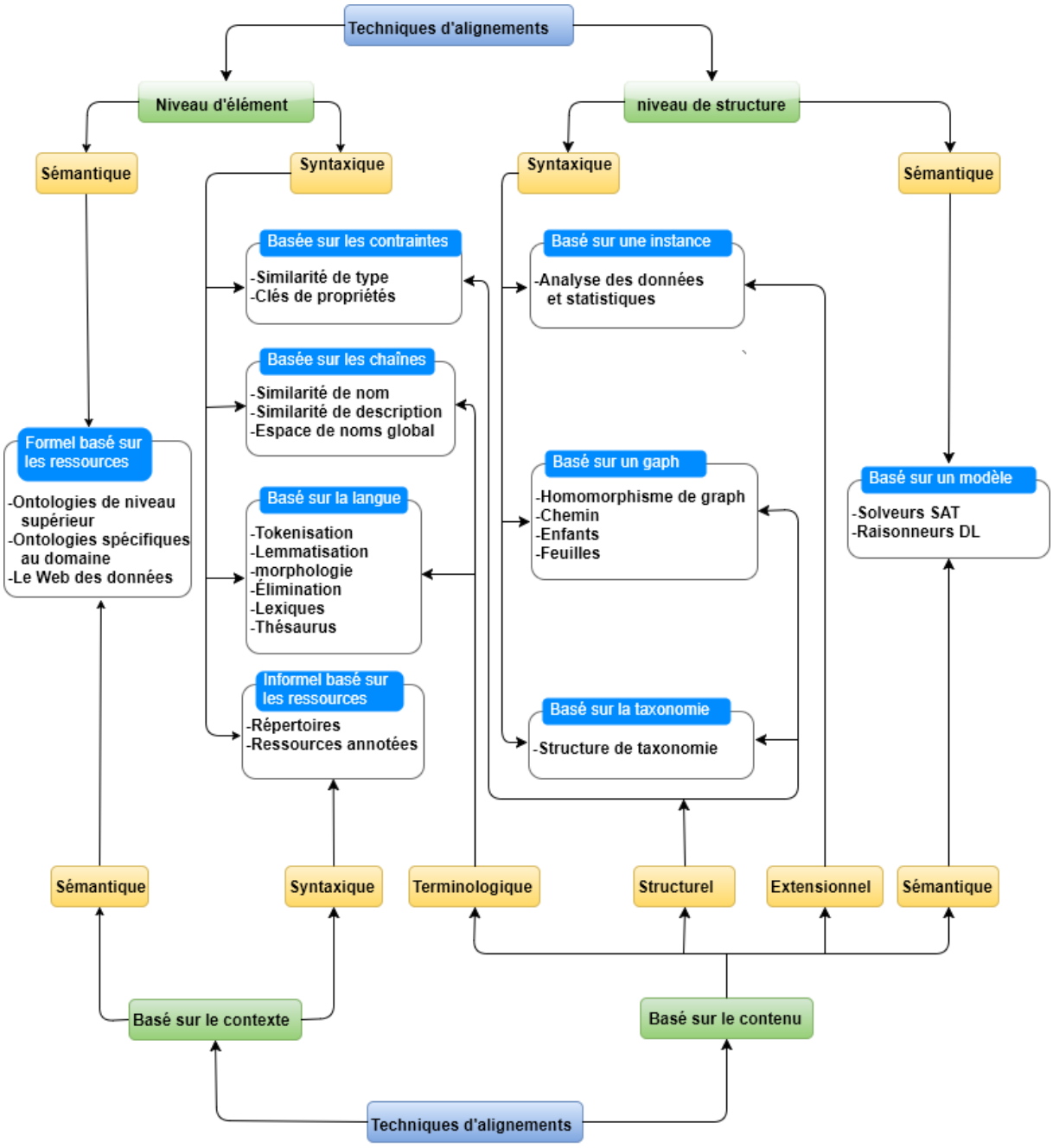


Figure 10 les techniques d'alignements [13, p. 77] traduite par S.AMIAR

2.5.2. Méthodes basé sur les chaines de caractère

Une classe est représentée par une IRI, une ou plusieurs labels, qui sont composés d'un ou plusieurs mots compréhensibles par l'humain et ces représentations sont uniques, on ne peut pas retrouver une duplication dans une même ontologie, comme elle peut aussi contenir des commentaires.

On peut appliquer les méthodes terminologiques sur ces entités afin d'identifier les correspondances entre ontologies, si deux labels ou IRI sont identiques alors les deux classes qu'ils représentent sont identiques aussi, avant d'appliquer ces méthodes on peut normaliser les chaines de caractères pour faire des calculs efficaces et avoir des résultats plus corrects.

2.5.2.1. Normalisation

Afin d'améliorer les résultats de comparaisons entre chaines de caractères on peut normaliser par les méthodes suivantes :

- **Normalisation de la casse** : rendre tout caractère majuscule en minuscule exemple : « Algérie » devient « algérie ».
- **Suppression des diacritiques** : remplacer tous les diacritiques par leurs remplaçant les plus fréquents par exemple « algérie » devient « algerie ».
- **Normalisation des blancs et des liaisons** : convertir toute tabulation, retour chariot, répétition d'espace, apostrophe, tiret … en un espace blanc par exemple « tizi-ouzou » devient « tizi ouzou ».
- **Suppression de chiffres** : par exemple « vehicule115 » devient « vehicule ».
- **Suppression des de ponctuations** : par exemple « U.M.M.T.O » devient « UMMTO ».

Il faut faire attention à l'utilisation et au choix de normalisations adéquates, par exemple dans certains cas c'est possible de perdre de l'information par suppression de chiffre, ou avoir de résultat différent suivant un certain ordre de normalisation etc.

Une fois que les mots sont normalisés en applique des méthodes de comparaison.

2.5.2.2. Égalité des chaines de caractère

Cette comparaison est très simple elle compare si deux chaines sont égales, on peut commencer par comparer leurs tailles puis les caractères un à un, elle retourne 1 si oui ou 0 sinon, cette mesure de similarité ne donne pas l'information sur combien deux chaines de caractère sont différentes.

2.5.2.3. Test de sous chaines de caractères

Cette comparaison peut considérer que deux chaines de caractère sont similaires d'un score de 0 à 1 si l'une des chaines contient l'autre, plusieurs variations existent pour calculer cette mesure de similarité l'une d'elles est la suivante :

Soit x, y deux chaines de caractère et t la plus longue sous chaine commune entre eux, et δ la fonction de calcul de similarité entre x et y

$$\delta(x, y) = \frac{2|t|}{|x| + |y|}$$

Une autre variation est la similarité **n-Gram**, elle calcule le nombre de n-gram commun entre deux chaines, par exemple 4-Gram du mot « hacking » et « hack, acki, ckin, king ».

soit $ngram(x, n)$ un ensemble de sous chaines de taille n de x, et $\delta(x, y)$ la fonction de calcul de similarité n-Gram entre x et y

$$\delta(x,y) = \left|ngram(x,n) \bigcap ngram(y,n)\right|$$

La version normalisée

$$\bar{\delta}(x,y) = \frac{|ngram(x,n) \cap ngram(y,n)|}{\min(|x|,|y|) - n + 1}$$

Cette fonction est assez efficace lorsque seulement certains caractères manquent.

2.5.2.4. Distance d'édition

La distance d'édition entre une chaine de caractère A et B consiste à calculer le nombre minimal de modifications qui il faut apporter à A pour que A soit égale à B

$$\delta(x,y) = \min_{(op_i)_I;\, op_n(\ldots op_1(s)) = t} \left(\sum_{i \in I} w_{op_i}\right)$$

Les différentes définitions d'une distance d'édition utilisent différents ensembles d'opérations de chaîne de caractères. Les opérations de la distance **Levenshtein** sont la suppression, l'insertion ou la substitution d'un caractère dans la chaîne de caractères. Étant la métrique la plus commune, la distance de Levenshtein est généralement ce qu'on entend par "distance d'édition"

2.5.2.5. Distance basé sur les Tokens

Cette technique est efficace beaucoup plus avec des textes longs, pour cela on profit des différentes chaines de caractère composant un concept (labels, commentaire, documentation, etc.) et par agrégation de ces chaines de caractères on obtient un document qu'on peut comparer avec d'autre documents, pour cela il faut extraire tous les mots représentant ce document pour avoir un sac de mots après certains prétraitements (normalisation, suppression de mots vide, etc.), et représenter ce sac de mot par un vecteur **V** dans un espace métrique **E** dans lequel chaque dimension est un terme et chaque position dans le vecteur est le nombre d'occurrence du terme lui correspondant dans le sac de mots.

La similarité de **cosinus** est l'une des similarités qu'on peut appliquer sur des vecteurs tel que présenter précédemment, soit $\vec{s}$ et $\vec{t}$ les vecteurs correspondants aux chaines de caractère s et t respectivement, la similarité de cosinus σ_V est calculé comme suivant :

$$\sigma_V(s,t) = \frac{\sum_{i \in |V|} \vec{s_i} \times \vec{t_i}}{\sqrt{\sum_{i \in |V|} \vec{s_i}^{\,2} \times \sum_{i \in |V|} \vec{t_i}^{\,2}}}$$

D'autres distance de l'espace métrique peuvent être utilisées comme la distance euclidienne, la distance de Manhattan et toute instance de la distance de Minkowski.

2.5.2.6. Méthodes basées sur la langue

Dans ces méthodes on considère les chaines de caractères représentant un concept d'une ontologie comme étant du texte sur lequel on peut appliquer des techniques de traitement du langage naturel pour aider à extraire les termes significatifs du texte, J.Euzenat et P.Shvaiko [13, p. 97] distinguent deux types de méthodes, celles qui reposent sur des algorithmes et celles qui utilisent des ressources externes :

2.5.2.7. Méthodes intrinsèques : la normalisation linguistique (reposent sur des algorithmes) :

- **Tokenisation** : consiste à segmenter une chaine de caractère en segments appeler Tokens reconnus par la ponctuation, les espaces, etc. par exemple « utilisation des ontologies » on obtient trois Tokens « utilisation » « des » « ontologies »
- **Suppression des mots vides** : telle que les prépositions, les articles, les pronoms, exemple « le », « la », « de », « du », « donc » etc.
- **Lemmatisations** : consiste à réduire un mot à sa forme canonique « lemme », par exemple enlever la forme du pluriel, le suffixe, le préfixe, etc. c'est possible d'utiliser des techniques de stemming qui sont une lemmatisation approximative telle que Lovins [15], Porter [16].

2.5.2.8. Méthodes extrinsèques (utilisent des ressources externes) :

Dans ces méthodes on ait recours à des ressources externes pour rendre l'identification de similarités plus efficace, par exemple par l'enrichissement de chaque terme d'un concept par ces synonymes qu'on retrouve dans des ressources externes, on peut considérer trois types de ressource externe :

- **Lexique :** est un ensemble de mot contenant des définitions (dictionnaire).
- **Thésaurus :** est une sorte de lexique auquel on a ajouté certaines informations relationnelles, comme la relation synonyme, hyponyme et hyperonyme.
- **Terminologies :** est un thésaurus qui est souvent spécifique à un domaine spécial et utilise beaucoup plus des phrases que des mots**.**

2.6. Stratégies d'alignement

Une fois que la base du système d'alignement est faite en se basant sur les méthodes d'alignement vues précédemment, on a besoin d'assurer certains points tels que la possibilité de gérer des ontologies à grande échelle, la combinaison de différentes similarités ou algorithmes d'alignement, l'amélioration des alignements par désambiguïsation, débogage et réparation, etc. pour cela des traitements plus complets sont nécessaires, dans ce qui suit on va voir certains de ces traitements.

2.6.1. Partitionnement d'une ontologie et élagage de l'espace de recherche

L'alignement peut se faire sur des ontologies de grande taille (des centaines de milliers d'entité), pour faire cela d'une manière efficace, on peut faire cela par deux manières : le partitionnement ou l'élagage.

2.6.2. Partitionnement

Cette méthode consiste à séparer l'ontologie en petites parties, et les aligner d'une manière indépendante, et à la fin agréger le résultat des alignements, par exemple c'est possible d'utiliser des algorithmes du Data Mining comme l'algorithme ROCK du clustering agglomératif [17], une des implémentations existantes est pyclustering[21], la figure suivante montre le principe général du partitionnement :

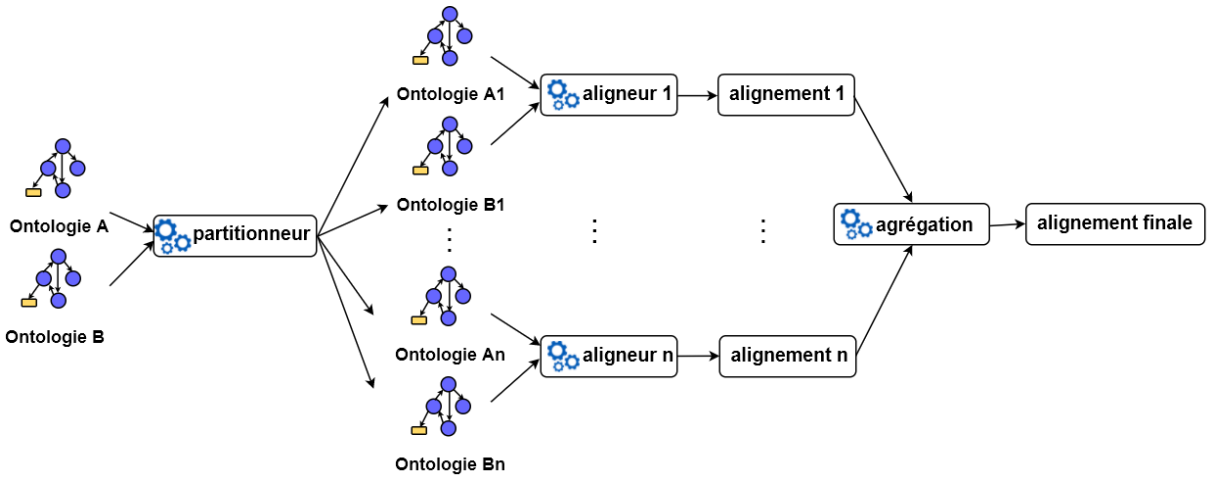


Figure 11 partitionnement d'ontologies

2.6.3. Élagage de l'espace de recherche

Les techniques d'élagage permettent de manière dynamique de réduire le nombre de comparaisons, par exemple à chaque fois qu'ont identifié deux concepts similaires, on les enlève de l'itération de comparaison, ce que réduira le nombre d'itérations pour la prochaine comparaison.

2.6.4. Agrégation de similarité et d'alignement

L'agrégation de similarité est la combinaison de plusieurs similarités donner par différentes approches et en résultat une seule similarité, ce calcule de similarité peut-être porter sur une même entité par différents algorithmes de calcul de similarité exemple une comparaison de deux annotations labelA et labelB par plusieurs algorithmes (cosinus, Leveinshtein, Jaccard, Hammer, …) ou sur différentes entités d'un même concept par exemple calcule de similarité entre noms des classes, labels, noms classes parent, propriétés etc. Dans les points qui suivent quelques approches d'agrégation parmi celles lister par J.Euzenat et P.Shvaiko [13].

2.6.5. Pondération

La similarité d'une classe avec une autre peut dépendre de plusieurs entités et algorithmes, d'où le besoin d'une seule mesure de similarité pondéré, parmi les techniques de pondérations existantes on retrouve les suivantes.

- **Normes triangulaires :** les normes triangulaires sont utilisées comme opérateurs de conjonction dans des calculs incertains, une des normes triangulaires pour agréger plusieurs dimensions est le produit pondérer.
    - **Produit pondéré :** Soit o un ensemble d'objets qui peuvent être analysés en n dimensions, le produit pondéré entre deux objets $x \; et \; \acute{x}, \forall x, \acute{x} \in o$ est

[21] https://github.com/annoviko/pyclustering

$$\delta(x, \acute{x}) = \prod_{i=1}^{n} \delta(x_i, \acute{x}_i)^{w_i}$$

Tel que $\delta(x, \acute{x})$ est la dissemblance de la paire d'objets le long de la $i$ème dimension et w_i est le poids de la dimension i.

- **Distances multidimensionnelles et somme pondérées**

Ces mesures sont bien adaptées aux dimensions indépendantes et tendent à équilibrer les valeurs entre les dimensions

Distance de Minkowski : soit o un ensemble d'objets qui peuvent être analysés en n dimensions, la distance de Minkowski entre deux objets $x \; et \; \acute{x}$, $\forall x, \acute{x} \in o$ est

$$\delta(x, \acute{x}) = \sqrt[p]{\sum_{i=1}^{n} \delta(x_i, \acute{x}_i)^p}$$

Tel que $\delta(x_i, \acute{x}_i)$ est la dissimilarité entre paires d'objets le long de la $i$ième dimension.

Somme pondérée : soit o un ensemble d'objets qui peuvent être analysés en n dimensions, la somme pondérée entre deux de ces objets est $\forall x, \acute{x} \in o$

$$\delta(x, \acute{x}) = \sum_{i=1}^{n} w_i \times \delta(x_i, \acute{x}_i)$$

Tel que $\delta(x_i, \acute{x}_i)$ est la dissimilarité entre paires d'objets le long de la $i$ième dimension et w_i est le poids de la dimension i.

- **Moyenne pondérée :**

Soit o un ensemble d'objets qui peuvent être analysés en n dimensions, la moyenne pondérée entre deux objets est $\forall x, \acute{x} \in o$:

$$\delta(x, \acute{x}) = \frac{\sum_{i=1}^{n} w_i \times \delta(x_i, \acute{x}_i)}{\sum_{i=1}^{n} w_i}$$

Tel que $\delta(x_i, \acute{x}_i)$ est la dissimilarité entre paires d'objets le long de la $i$ième dimension et w_i est le poids de la dimension i.

- **Somme harmonique pondérée adaptative**

Soit o et $\acute{o}$ deux ensembles d'objets qui peuvent être comparés par n mesures, la somme harmonique pondérée adaptative de telles mesures entre deux de ces objets est $\forall x \in o \, , \acute{x} \in \acute{o}$:

$$\delta(x, \acute{x}) = \sum_{i=1}^{n} h(\delta_i) \times \delta_i(x, \acute{x})$$

Tel que $\delta(x_i, \acute{x}_i)$ est la dissimilarité entre paires d'objets le long de la $i$ième mesure et h est

tel que :

$$h(\delta_i) = \frac{\left|\left\{\langle e, é\rangle \in o \times ó \;;\; \wedge \begin{array}{l} \forall y \in ó \setminus \{é\}, \sigma(e,y) > \sigma(e,é) \\ \forall x \in o \setminus \{e\}, \sigma(x,é) > \sigma(e,é) \end{array}\right\}\right|}{\min(|o|, |ó|)}$$

Cette mesure peut être normalisée en divisant les résultats par la somme des poids. Il favorise les similarités ou distances qui sont plus discriminantes.

- **Moyenne pondérée ordonnée**

Soit o un ensemble d'objets qui peuvent être analysés en n dimensions, un opérateur moyen pondéré ordonné f est une fonction de $D^n \to D$ (avec D un ensemble commander par $\leq$ et muni d'une borne supérieure $\top$) Satisfaisant $\forall x, x_1, \ldots, x_n \in$ D, défini comme

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i \times \acute{x}_\iota$$

Tels que :

$w_1, \ldots, w_n$ est un ensemble de poids dans [0 1] tel que $\sum_{i=1}^{n} w_i = 1$;

$\acute{x}_i$ est le premier élément de $(x_1, \ldots, x_n)$.

2.6.6. Vote

Ici on prend plusieurs alignements de deux ontologies chaque alignement est indépendant de l'autre, et le choix du bon alignement se fait par vote majoritaire, par exemple par le nombre de concepts se correspondant par rapport à un seuil donner.

Soit $\{A_i\}_{i \in I}$ un ensemble d'alignement sur les mêmes ontologies O et $\acute{O}$ et C les concepts alignés :

- **Vote majoritaire**

$$A = \left\{ c \in \bigcup_{i \in I} A_i \middle| |\{A_i | c \in A_i\}_{i \in I}| > \frac{|I|}{2} \right\}$$

Souvent pour aligner deux ontologies on utilise plusieurs méthodes d'alignement ou mesures de similarité, et pour avoir l'alignement final on peut utiliser l'une des méthodes qu'on vient de voir telle que l'agrégation et le vote, il existe d'autres méthodes comme l'argumentation dans laquelle le choix se fait par préférence.

Les stratégies que on vient de voir sont très importante pour rendre un système d'alignement plus performant en termes de vitesse et de mémoire. D'autres stratégies existent, comme faire apprendre aux aligneurs comment aligner à partir d'alignement corrects (par utilisation de réseau bayésien, réseau de neurones, arbre de décision, etc.), cependant on ne va pas les cités tous, on s'est contenté seulement de ce que nous allons utiliser par la suite dans la phase de réalisation d'un système d'alignement.

2.7. Alignement sémantique

2.7.1. Introduction

Le processus d'alignement sémantique permet d'identifier les relations sémantiques telles que la synonymie, l'hyponymie, et l'hyperonymie. Comme cité dans la figure 3 sur les techniques d'alignements, l'alignement sémantique peut-être considérer selon deux niveaux, le niveau d'éléments et le niveau structurel, basé soit sur le contexte soit sur le contenu, nous nous intéressant à l'alignement sémantique au niveau d'élément basé sur le contexte, par utilisation de ressource externe (connaissances de base), pour identifier des relations sémantiques qui ne peuvent pas être identifiées par les techniques traditionnelles basées seulement sur les ontologies alignées.

Définition : Soit $A_1 = < C_1, C_2, SC_1 >$, $A_2 = < C_3, C_4, SC_2 >$, $A_3 = < C_5, C_6, SC_3 >$ trois alignements différents (correspondance entre deux concepts), C est un concept, CS est le score de similarité entre deux concepts, et s est une chaine de caractère qui représente un concept. La figure suivante montre les trois exemples d'alignements sémantiques **A1, A2, et A3**

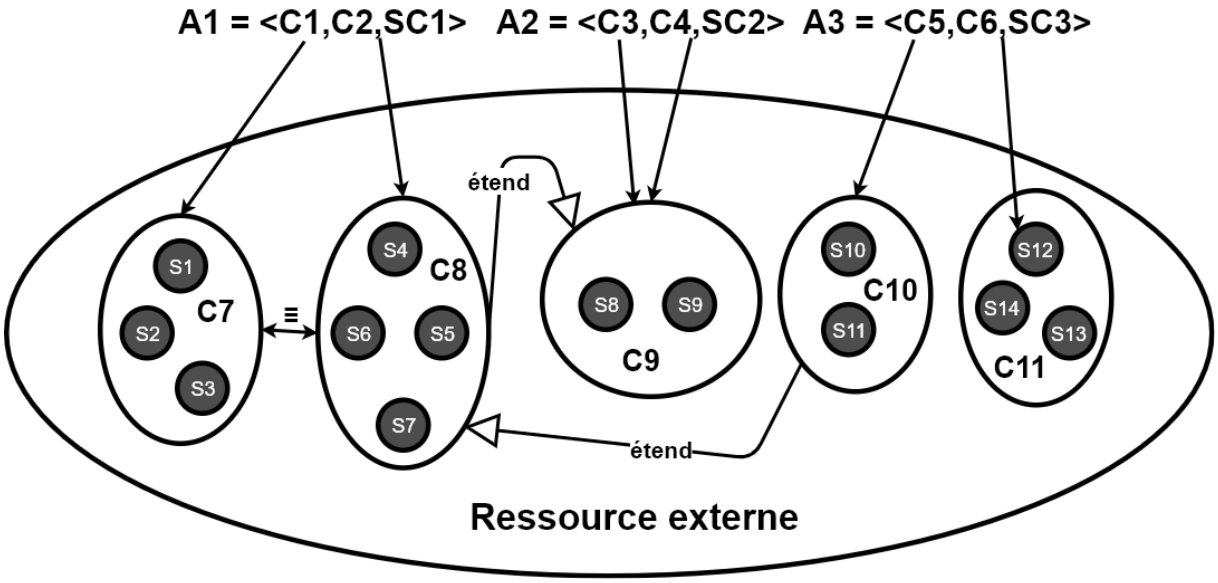


Figure 12 exemple d'alignement sémantique par l'utilisation d'une ressource externe

- Le concept C7 est équivalent au concept C8 et C1 correspond à C7 et C2 correspond à C8, ce qui implique que le concept C1 est équivalent à C2, C9 est parent (hyperonyme) de C8, C10 est enfant (hyponyme) de C8, ce qui implique que C3 et C4 sont hyperonymes de C1 et C2, et C5 est hyponyme de C1 et C2.
- Le concept C3 et C4 correspond au même concept C9, ce qui implique que le concept C3 et C4 sont équivalents, le concept C8 est enfant de C9, ce qui implique que C1 et C2 sont hyponymes de C3 et C4.

- Le concept C10 et le concept C11 sont indépendants, et le concept C5 correspond au concept C10, le concept C6 correspond au concept C11, ce qui implique que le concept C5 n'est pas équivalent au concept C6, le concept C10 enfant de C8, ce qui implique que le concept C1, C2, C7, C8 sont hyperonymes de C5.

2.7.2. Connaissances de base

Le processus d'alignement d'ontologie avec l'utilisation de connaissances de base est plus adapté pour identifier les correspondances entre des ontologies hétérogènes, car les connaissances de base sont utilisées comme intermédiaires ou une référence pour identifier les relations entre deux concepts en correspondance, ce qui permet de surmonter deux grandes limitations des techniques d'alignement basé sur la syntaxe et la structure, ces méthodes échouent quand il y a un petit chevauchement lexical entre les étiquettes des entités ontologiques ou lorsque les ontologies ont des structures faibles ou dissemblables, par exemple le mot « voiture » et « véhicule » se réfèrent à la même chose , mais qui ne peut pas être identifiée comme semblable d'où le besoin d'une partie tierce qui contient des relations sémantiques telles que la synonymie, hyponymie, hyperonymie, si on prend l'exemple précédent et par l'utilisation de connaissances de base tel que WordNet[22] qui est une base de données lexicale de la langue anglaise, il regroupe les mots en un ensemble de synonymes, on arrivera à identifier que « voiture » est synonyme de « véhicule », le schéma suivant explique cet exemple et montre la différence entre les méthodes traditionnelles et les méthodes utilisant des connaissances de base.

[22] https://wordnet.princeton.edu/

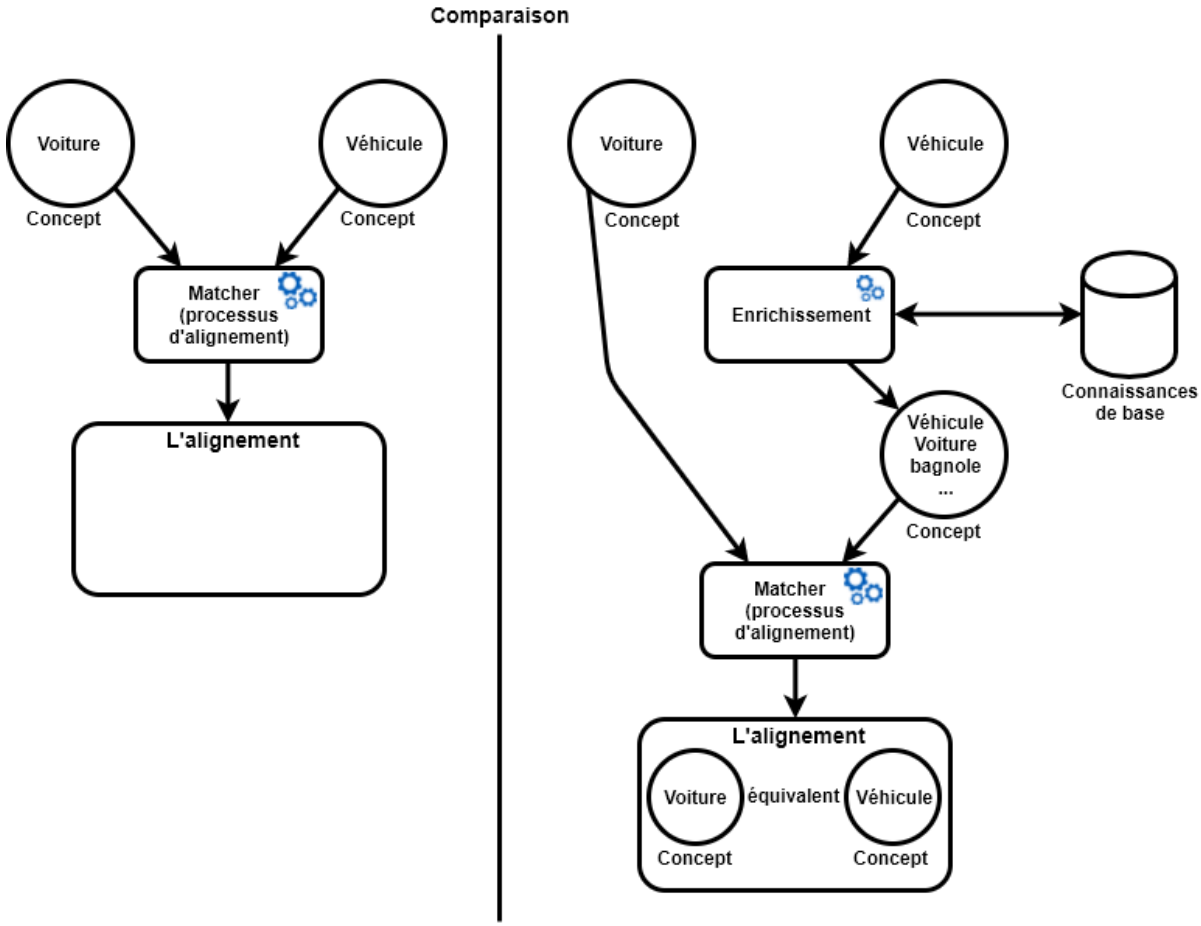


Figure 13 comparaison entre alignement sans enrichissement et alignement avec enrichissement

La deuxième méthode est appelée alignement indirect, avant le processus d'alignement, nous enrichissons les termes représentant un concept par leurs synonymes.

2.7.3. Alignement sémantique avec UMLS

Le choix d'une base de connaissance appropriées est une étape très importante, le choix se fait par rapport au domaine des ontologies à aligner, dans notre cas nous allons aligner des ontologies d'un domaine médicales, d'où le besoin d'utiliser une ressource externe appartenant au domaine médicale, parmi les ressource existante, on retrouve UMLS qui est un thésaurus de terminologie biomédicale qui regroupe de nombreux vocabulaires et normes biomédicales et de santé pour permettre l'interopérabilité entre les systèmes informatiques, il intègre des données provenant de différentes sources, à ce jour plus de 150 terminologies biomédicales. Il est conservé par la « National Library of Medicine » (NLM), organisme à but non lucratif, UMLS se compose de trois parties principales :

Metathesaurus : Le Metathesaurus est une base de données de vocabulaire très large, polyvalente et multilingue qui contient des informations sur les concepts biomédicaux et liés à la santé, leurs différents noms et les relations entre eux. Conçu pour être utilisé par les développeurs de systèmes, le Metathesaurus est construit à partir des versions électroniques de divers thésaurus, classifications, ensemble de codes et listes de termes contrôlés utilisés dans les soins des patients, la facturation des services de santé, les statistiques de santé publique, l'indexation et le catalogage de la littérature

biomédicale et/ou recherche fondamentale, clinique et de services de santé, y compris CPT®[23], ICD-10 -CM[24], LOINC®[25], MeSH®[26], RxNorm[27] et SNOMED CT®[28].

Réseau sémantique : grandes catégories (types sémantiques) et leurs relations (relations sémantiques).

Lexique spécialisé et outils lexical : outils de traitement du langage naturel.

UMLS est un graphe dirigé de concepts. Ce graph dirigé contient des relations de haut niveau entre les concepts, fournissant ainsi une sémantique pour le processus d'alignement. Chaque concept dans l'UMLS possède un identifiant unique connu sous le nom de CUI (Concept Unique Identifier).

Suivant quelques relations entre les concepts dans UMLS [18, pp. 73–74]:

- **PAR** : cette relation signifie qu'un concept A est un « parent » d'un autre concept B dans l'UMLS.
- **RB** : il s'agit d'une relation « plus large que » entre un concept A et un concept B.
- **CHD** : cela signifie qu'un concept A est un enfant d'un un concept B.
- **RN** : il s'agit d'une relation « plus étroite que » entre un concept A et un concept B.
- **RO** : cela signifie qu'il existe une relation entre un concept A et concept B, non synonyme, plus étroit ou plus large.
- **RQ** : cela signifie qu'un concept A et un concept B sont liés et éventuellement synonymes.
- **SIB** : il s'agit d'une relation de fratrie (sibling) entre un concept A et un concept B dans le metathesaurus.

UMLS est donnée sous format RRF[29], est accompagné d'outils permettant de le mettre sur une base de données relationnelle SQL, nous avons opté à utiliser une base de données orientée document (NoSQL), qui est MongoDB pour des raisons de performance, pour cela on a utilisé **humumls**[30] qui est une bibliothèque python[31] qui permet d'installer UMLS sur MongoDB[32] et offre des requêtes de base prêtes pour interroger UMLS.

Chaque concept d'une ontologie contient une ou plusieurs étiquettes (labels) représentant ce concept, pour chaque étiquette on lui cherche son concept correspondant dans UMLS, on prend tous les termes représentant ce concept dans UMLS, ainsi que les termes synonymes, hyponyme, hyperonyme, et on les ajoute au concept dans l'ontologie, c'est ce qu'on appelle le processus

[23] https://www.ama-assn.org/practice-management/cpt-current-procedural-terminology
[24] https://www.cdc.gov/nchs/icd/icd10cm.htm
[25] https://loinc.org/
[26] https://www.nlm.nih.gov/mesh/
[27] https://www.nlm.nih.gov/research/umls/rxnorm/
[28] http://www.snomed.org/snomed-ct
[29] https://www.ncbi.nlm.nih.gov/books/NBK9685/
[30] https://github.com/stephantul/humumls
[31] http://python.org/
[32] https://www.mongodb.com

d'enrichissement, voici un petit extrait d'enrichissement avec UMLS pour le terme en anglais « heart » qui signifie « cœur » :

L'identifiant « CUI » du concept représentant le mot « heart » est « C0018787 »

Les termes représentant le concept « C0018787 » dans UMLS :

['heart', 'hearts', 'cardiac', 'heart', 'heart', 'cor', 'cardiac structure', 'heart structure', 'cardiac', 'coronary', 'cardio'].

Les concepts enfants du concept « C0018787 » :

['C0150838', 'C0927231', 'C3887460', 'C0178784', 'C0007226', 'C0221589', 'C0025066', 'C0229962', 'C4037974'], chaque concept est représenté par plusieurs termes.

Les concepts parents du concept « C0018787 » :

Ce concept contient 829 concepts parent voici quelques-uns ['C1866207', 'C4014401', 'C3554475', 'C3554048', 'C1862675', 'C3276919', 'C1970628', 'C3551842', 'C3809142', …].

L'utilisation d'UMLS pour l'enrichissement des concepts des ontologies, permettra de retrouver des relations sémantiques entre les ontologies telles que les synonymes, les hyponymes, les hyperonymes, ce que donnera plus de résultat dans l'alignement.

2.7.4. Conclusion

L'alignement sémantique, par l'utilisation d'une ressource externe dans notre cas, permettra d'avoir des résultats beaucoup plus précis et correct que l'utilisations des méthodes traditionnelles seul.

2.8. Evaluation

2.8.1. Types d'évaluation

Afin d'évaluer un système d'alignement d'ontologie, plusieurs ressources d'évaluations sont disponibles (Anatomie, Conférence, Grandes ontologies biomédicales, Maladie et phénotype, etc.) avec différents ensembles de données et différentes structures, nous avons choisi la Grandes ontologies biomédicales « largebio » de la campagne OAEI pour évaluer notre système. Cette track [19] consiste à trouver des alignements entre le modèle de base de l'anatomie (FMA), SNOMED CT et le National Cancer Institute Thesaurus (NCI). Ces ontologies sont sémantiquement riches et contiennent des dizaines de milliers de classes.

2.8.2. Mesures d'évaluation

Mesures de conformité

Parmi les mesures d'évaluation on a celles basées sur des méthodes de conformité qui consistent à mesure le rappel, la précision, la F-Mesure, et la mesure globale :

Soit R l'ensemble des alignements de référence et A est l'ensemble des alignements obtenus par une certaine méthode :

- Précision

La précision mesure le nombre de correspondances correctes trouvées par rapport au nombre total d'ensemble de correspondances obtenues par une certaine méthode

$$Précision = \frac{|A \cap R|}{|A|}$$

- Rappel

Le rappel mesure le nombre de correspondances correctes trouvées par rapport au nombre total de correspondances existantes.

$$Rappel = \frac{|R \cap A|}{|R|}$$

Une valeur de rappel élevée indique que beaucoup des alignements ont effectivement été trouvés, mais il ne donne aucune information sur le nombre d'alignements faux.

- F-mesure

La mesure F combine les mesures de précision et de rappel comme mesure d'efficacité unique

$$F - mesure = \frac{2 \times précision \times rappel}{précision + rappel} \times 100\%$$

- Global

La mesure globale est le rapport entre le nombre d'erreurs et la taille de l'alignement attendu. Cette mesure est considérée comme représentant la distance d'édition entre un alignement et un alignement de référence dans lequel la seule opération est la "correction de l'erreur". Pour cette raison, il est considéré comme une mesure de l'effort requis pour corriger l'alignement.

$$O(A,R) = 1 - \frac{|(A \cup R) - (A \cap R)|}{|R|} \times 100\%$$

Mesures de performance

- **La vitesse**
  La vitesse est le temps pris par l'algorithme d'alignement pour terminer le processus d'alignement, pour comparer la vitesse de plusieurs systèmes, il faut les tester dans les mêmes conditions (sur un environnement matériel et logiciel pareil).
- **Réseau**
  Certains systèmes peuvent avoir besoin de connexion réseau, pour accomplir le processus d'alignement, et c'est possible que le taux de la bande passante consommé soit limité, donc il faut calculer le taux de la bande passante consommé par le système d'alignement.
- **Mémoire**
  Le taux de mémoire utilisé par le système d'alignement doit être mesuré, car la mémoire est une contrainte qui pourra rendre le système d'alignement non fonctionnel, cette mesure peut être calculer par le maximum de la mémoire vive utilisé pendant le processus d'alignement.
- **Évolutivité**
  Le système peut être mesuré par rapport à sa capacité de gérer la mémoire d'une manière efficace et dans des délais raisonnables pour des ontologies de grande taille.

2.8.3. Formats d'alignement

Les alignements d'ontologie sont des ensembles de relations entre les entités ontologiques. Les alignements peuvent être exprimés en différentes langues. Par exemple, les deux relations « ≡ » et « ⊆ » peuvent être exprimées dans OWL par l'intermédiaire de « owl:equivalentClass » et « rdfs:subClassOf »

Le format d'alignement utilisé par OAEI est composé des éléments suivants [20] :

L'élément alignement :

Les éléments d'un alignement décrivent un alignement particulier. Ses attributs sont les suivants :

- **xml** (Valeur: "yes" / "no") indique si l'alignement peut être lu comme un fichier XML avec la DTD.
- **level** (Valeurs: "0", "1", "2") le niveau d'alignement, caractérisant son type, la valeur 0 est réservée aux alignements dans lesquels les entités alignées sont identifiées par des URI, la valeur 1 était destiné à des alignements dans lesquels les correspondances correspondent à des ensembles d'entités identifiés par les URI, la valeur 2 pour le langage EDOAL[33] qui est plus expressive.
- **type** (Valeurs: "11"/ "1?"/ "1+"/ "1*"/ "?1"/ "??"/ "?+"/ "?*"/ "+1" /"+?"/ "++"/ "+*"/ "*1"/ "*?"/ "?+"/ "**" ; par défaut"11") le type ou l'aridité de l'alignement. Les notations habituelles sont 1:1, 1:m, n:1 ou n:m.
- **onto1** (Valeur: Ontologie) la première ontologie alignée.
- **onto2** (Valeur: Ontologie) la deuxième ontologie alignée.
- **map** (valeur: cell) une correspondance entre les entités des ontologies.

L'élément ontologie :

L'élément ontologie contient des informations sur les ontologies alignées, il a trois attributs :

- **rdf:about** contient l'URI identifiant l'ontologie.
- **location** contient l'URL correspondant à un emplacement où l'ontologie peut être trouvée.
- **formalism** décrit la langue dans laquelle l'ontologie s'exprime par son nom et son URI.

L'élément cell :

Chaque correspondance est identifiée par l'élément « cell » dans un alignement. Un élément « cell » possède les attributs suivants :

- **rdf:about** (Valeur: URI, facultatif) un identifiant pour la cellule « cell ».
- **entity1** (Valeur: URI ou edoal: Expression) la première entité d'ontologie alignée.
- **entity2** (Valeur: URI ou edoal: Expression) la deuxième entité d'ontologie alignée.

[33] http://alignapi.gforge.inria.fr/edoal.html

- **relation** (Valeur: String; « > (hyperonyme), <(hyponyme), = (équivalent), %(incompatible), HasInstance, InstanceOf ») la relation entre les deux entités .
- **mesure** (Valeur: float; entre 0 et 1, par défaut: 1) la confiance que la relation entretient entre la première et la seconde entité.

Voici un exemple d'alignement de deux ontologies :

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<rdf:RDF xmlns='http://knowledgeweb.semanticweb.org/heterogeneity/alignment#'
      xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
      xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
<Alignment>
 <xml>yes</xml>
 <level>0</level>
 <type>?*</type>
 <onto1>
  <Ontology rdf:about="http://bioontology.org/projects/ontologies/fma/fmaOwlDlComponent_2_0">
   <location>file:/home/ernesto/Documents/OAEI_2016/LargeBio/Flagged/oaei_FMA_small_overlapping_nci.owl</location>
  </Ontology>
 </onto1>
 <onto2>
  <Ontology rdf:about="http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl">
   <location>file:/home/ernesto/Documents/OAEI_2016/LargeBio/Flagged/oaei_NCI_small_overlapping_fma.owl</location>
  </Ontology>
 </onto2>
 <map>
  <Cell>
   <entity1 rdf:resource='http://bioontology.org/projects/ontologies/fma/fmaOwlDlComponent_2_0#Muscle_cell'/>
   <entity2 rdf:resource='http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Muscle_Cell'/>
   <relation>=</relation>
   <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.93666473031044</measure>
  </Cell>
 </map>
 <map>
  <Cell>
   <entity1 rdf:resource='http://bioontology.org/projects/ontologies/fma/fmaOwlDlComponent_2_0#Chemokine'/>
   <entity2 rdf:resource='http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Chemokine'/>
   <relation>=</relation>
   <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
  </Cell>
 </map>
</Alignment>
</rdf:RDF>
```

2.9. Outils d'alignement existants

On va présenter brièvement les systèmes d'alignement d'ontologies ayant participé dans la campagne OAEI (Ontologie Alignment Evaluation Initiative) en 2016.

❖ ALIN

ALIN [21, p. 1] est un système d'alignement ontologique spécialisé dans l'alignement interactif des ontologies. Sa caractéristique principale est la sélection des correspondances à montrer à l'expert, en fonction des commentaires précédents donnés par l'expert. Cette sélection est basée sur des caractéristiques sémantiques et structurelles. ALIN a obtenu l'alignement avec la plus haute qualité

dans le suivi interactif du jeu de données de la Conférence, la figure suivante montre le workflow d'ALIN.

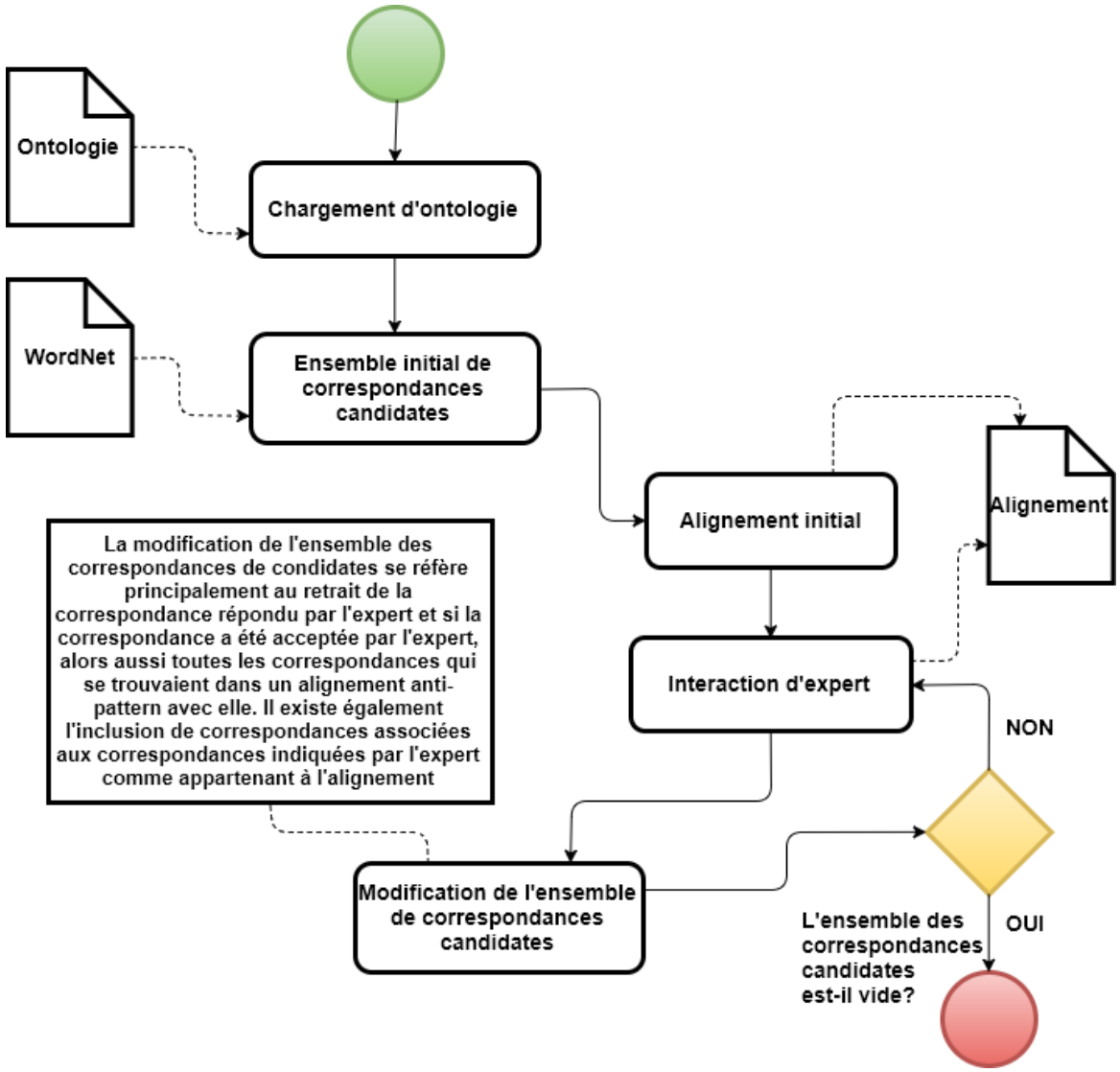


Figure 14 [21, p. 2] *workflow d'ALIN traduite, par S.AMIAR*

❖ AML

AgreementMakerLight (AML) [22, p. 1] est un système automatisé d'adaptation d'ontologies dérivé d'AgreementMaker et conçu pour résoudre des problèmes d'alignement à grande échelle. Il repose principalement sur des techniques d'alignement lexical, en mettant l'accent sur l'utilisation de ressources externes comme connaissances de fond.

❖ CroMatcher

CroMatcher [23, pp. 1–2] est un système d'alignement d'ontologie dans lequel le processus de correspondance s'effectue automatiquement. Il prend en charge l'alignement entre les ontologies exprimées dans Web Ontology Language (OWL). Il existe plusieurs matchers de chaines de caractères et structurelle de base dans le système CroMatcher. Chaque matcher de base détermine la similitude entre les entités en utilisant des informations obtenues à partir d'un ou plusieurs composants des ontologies comparées, les résultats obtenus par tous les matchers de base doivent être agrégés afin d'obtenir les meilleurs résultats d'alignement finals. Les matchers de base de chaînes de caractères,

ainsi que les matchers de base structurelles, sont liés par une composition parallèle de matchers de base. Tout d'abord, les matchers de base de chaînes de caractères sont exécutés. Les résultats obtenus par les matchers de base de chaîne de caractères sont automatiquement agrégés en utilisant une méthode d'agrégation pondérée. Ces résultats agrégés sont ensuite utilisés dans l'exécution des matchers structurels comme valeurs initiales des correspondances entre entités. Encore une fois, les résultats obtenus par les matchers structurels de base sont agrégés en utilisant l'agrégation pondérée. Avant l'alignement final, les résultats agrégés des matchers de chaîne et les résultats agrégés des matchers structurels sont agrégés en utilisant l'agrégation pondérée. Finalement, la méthode d'alignement final itératif est exécutée afin de sélectionner les correspondances appropriées entre les entités des ontologies comparées à partir des résultats d'alignement agrégés, la figure suivante montre les composants de base de CroMatcher.

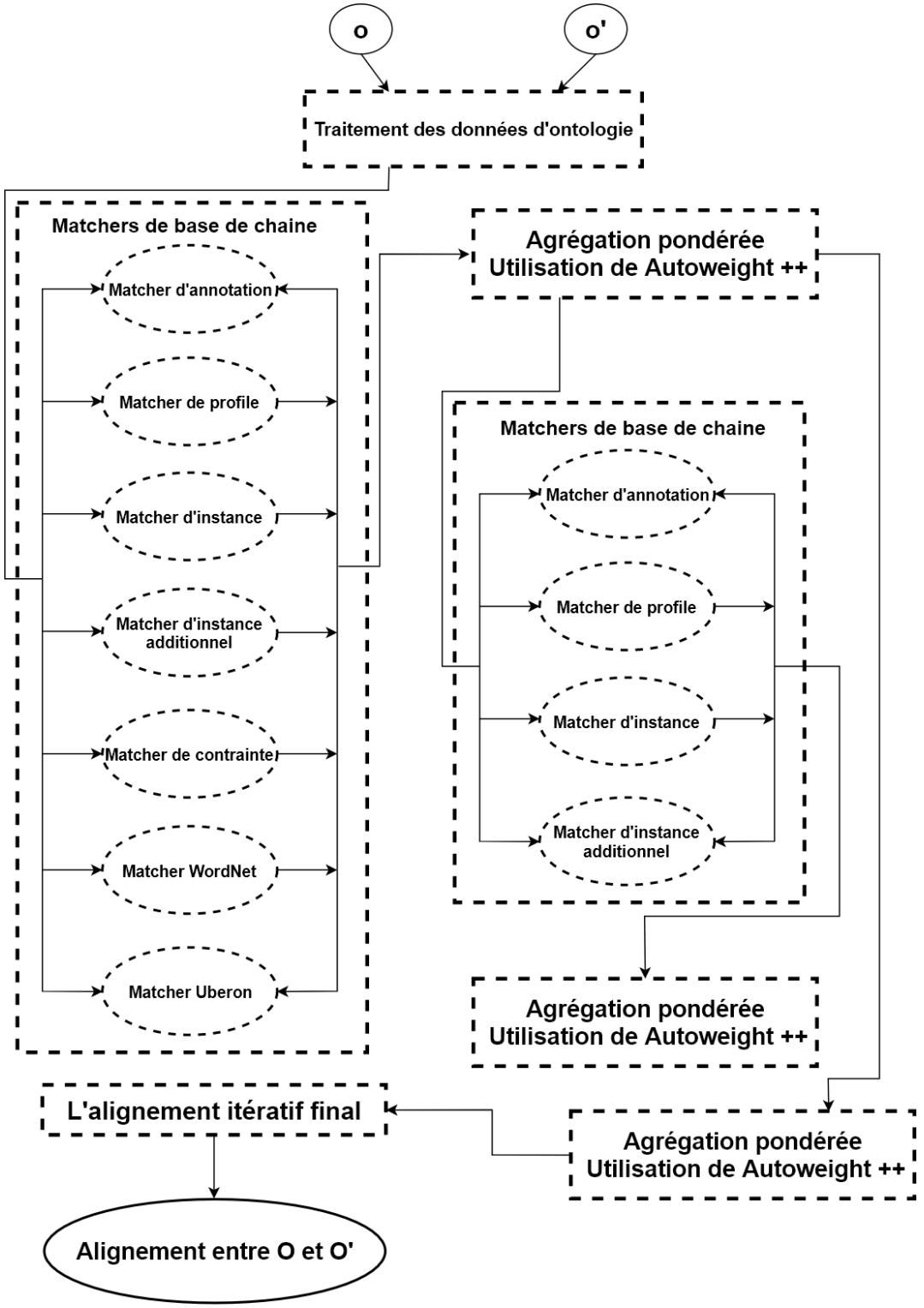


Figure 15 [23, p. 3] *Workflow et les principaux composants de CroMatcher, traduite par S.AMIAR*

❖ DisMatch

DisMatch [24, p. 1] est un système d'alignement d'ontologies expérimental basé sur l'utilisation d'une mesure de distribution basée sur un corpus pour l'approximation de relations sémantiques. Grâce à l'utilisation d'un corpus lié à un domaine, la mesure peut être appliquée à un problème axé sur le domaine du corpus. Pour une paire d'ontologies, DisMatch calcule la matrice des relations sémantiques entre les étiquettes (labels) représentant leurs concepts. Il utilise ensuite cette matrice comme entrée pour l'algorithme classique de Similarity Flooding, afin d'intégrer l'information taxonomique dans les résultats finaux.

❖ DKP-AOM

DKP-AOM [25, p. 1] est un outil de fusion d'ontologie conçu pour fusionner des ontologies hétérogènes. Dans la campagne OAEI, il a participé par son composant de correspondance ontologique qui sert de module de base capable de combiner des ontologies à grande échelle avant leur fusion. DKP-AOM a participé avec deux versions (DKP-AOM et DKP-AOM_lite). DKP-AOM mis en œuvre dans OWLAPI 3, le nom DKP provient du concept de l'analyse de la connaissance disjoint (DKA) et de la Préservation disjoint des connaissances (DKP) pendant le processus de fusion. L'analyse de connaissance disjointe joue un rôle essentiel dans le contrôle de l'espace de recherche pour trouver des similitudes entre les ontologies sources. La recherche dans les partitions disjointes des ontologies source réduit considérablement la complexité du temps de la phase de mappage. La préservation disjoint des connaissances dans l'ontologie fusionnée contribue à préserver les axiomes disjoints dans les sous-hiérarchies de l'ontologie fusionnée pour éviter l'incomplétude dans l'ontologie fusionnée résultante. De cette façon, il identifie également les différents conflits entre les ontologies sources basées sur des axiomes disjoints dans les ontologies source et détecte les mappages incohérents. Les mappages informatiques qui conduisent dans de nombreux cas à un grand nombre de classes insatisfaisantes sont éliminés de sorte que l'ontologie fusionnée résultante ne devrait pas subir d'incohérences

❖ FCA-Map

FCA-Map [26, p. 1] est un système automatique d'alignement d'ontologie basé sur l'analyse de concept formel (FCA), qui est un modèle mathématique bien développé pour analyser les individus et structurer les concepts. Plus précisément, il est construit avec trois types de contextes formels et extrait les mappages des réseaux dérivés. Tout d'abord, le contexte formel basé sur le token décrit comment les noms de classes, les étiquettes et les synonymes partagent des tokens lexicaux, conduisant à des mappages lexicaux (ancres) dans des ontologies. Deuxièmement, le contexte formel basé sur la relation décrit comment les classes sont dans des relations taxonomiques ou disjointes avec les ancres, conduisant à des preuves structurelles positives et négatives pour la validation de la correspondance lexicale. Enfin, après la réparation par incohérence, un contexte relationnel positif peut être utilisé pour découvrir des mappages structurels supplémentaires.

❖ Lily

Lily [27, p. 1] en tant que système d'alignement d'ontologie, est capable de résoudre certains problèmes liés aux ontologies hétérogènes. Il peut traiter des ontologies normales, des ontologies informatives faibles, un débogage du mappage d'ontologie et le réglage du processus d'alignement, à la fois normale et à grande échelle, Lily combine plusieurs techniques effectives et efficaces pour faciliter les alignements. Il existe quatre stratégies d'alignement principales : (1) La Générique Ontology Matching (GOM) est utilisée pour des tâches d'alignement communes avec des ontologies

de taille normale. (2) L'alignement de l'ontologie à grande échelle (LOM) est utilisé pour les tâches correspondantes avec des ontologies de grande taille. (3) Le débogage du mappage d'ontologie sert à vérifier et à améliorer les résultats d'alignement. (4) le réglage du processus d'alignement d'ontologique est utilisé pour améliorer les performances globales.

❖ LogMap

LogMap [28, pp. 273–288], [29] est un système d'alignement d'ontologie hautement évolutif qui met en œuvre les principes de cohérence et de localité. LogMap prend également en charge l'interaction utilisateur (en temps réel) pendant le processus d'alignement, ce qui est essentiel pour les cas d'utilisation nécessitant des mappages très précis. LogMap est l'un des quelques systèmes d'alignement d'ontologie qui (1) peuvent combiner efficacement des ontologies sémantiquement riches contenant des dizaines (voire des centaines) de milliers de classes, (2) intègre des techniques sophistiquées de raisonnement et de réparation pour minimiser le nombre d'incohérences logiques et (3) prend en charge l'intervention de l'utilisateur lors du processus d'alignement.

❖ LPHOM

LPHOM [30, p. 1] (Programme linéaire pour l'harmonisation holistique de'ontologie) est un système d'alignement holistique d'ontologie, il a participé pour la première fois à la campagne OAEI en 2016. Même si le système a été conçu pour faire face à l'alignement d'ontologie holistique (c.-à-d. Associant plusieurs ontologies simultanément), il est également capable de faire des alignements d'ontologie par paires, la figure suivante montre le workflow du système.

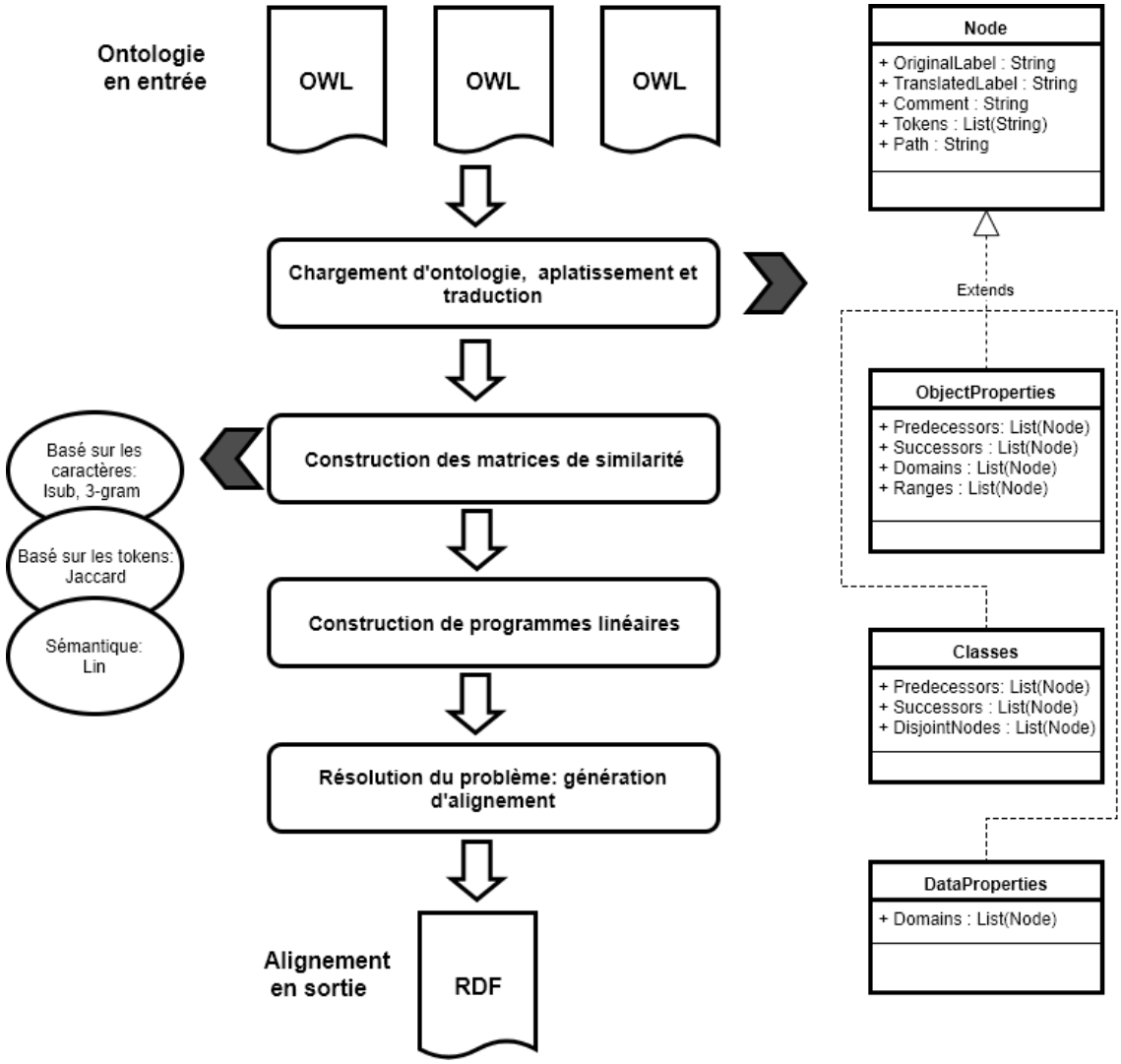


Figure 16 [30] *workflow et composants de LPHOM, traduite par S.AMIAR*

❖ **LYAM++**

LYAM ++ [31, pp. 1–2], est un système d'alignement d'ontologie entièrement automatique basée sur l'utilisation de sources externes. LYAM ++ ne repose pas sur la traduction automatique pour la correspondance entre ontologies. Au lieu de cela, il utilise le réseau sémantique multilingue à vocation générale ouvertement disponible BabelNet[34] afin de recréer le contexte sémantique manquant dans le processus d'alignement, la figure suivante montre le workflow du système.

[34] http://babelnet.org/

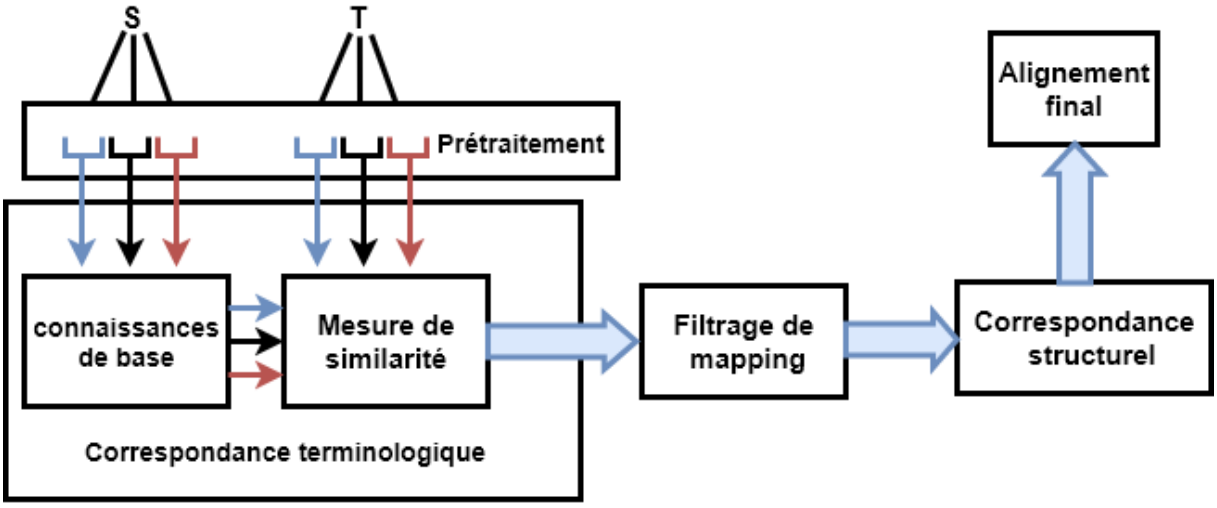


Figure 17 [31, p. 2] *workflow et composants de LYAM++, traduite par S.AMIAR*

❖ **RiMOM**

RiMOM [32, p. 1] fournis un support pour la correspondance entre instances multilingues de manière supervisée ou non supervisée, le système est composé de trois techniques majeures :

- **Le blocage** : permet d'indexer les instances en fonction de leurs objets dans deux bases de connaissances, puis sélectionnez les instances qui contiennent les mêmes clés que les paires d'instances candidates. Cette étape permet l'élimination de nombre de paires à comparer, ce qui améliore considérablement l'efficacité du système.
- **La multi-stratégie** : est l'implémentations de plusieurs correspondants dans le système de correspondance d'instances, puis exécutions des correspondants en parallèle, puis agréger le résultat en fonction des caractéristiques des ontologies source.
- **L'apprentissage par machine** : par utilisation d'alignements existants, et formalisation de l'alignement d'instance en tant que problème de classification binaire et utilisation des mappages de référence pour former un classificateur, qui déterminera si une paire d'instances est équivalente ou non.

La figure suivante montre le Framework sur lequel est basé le système :

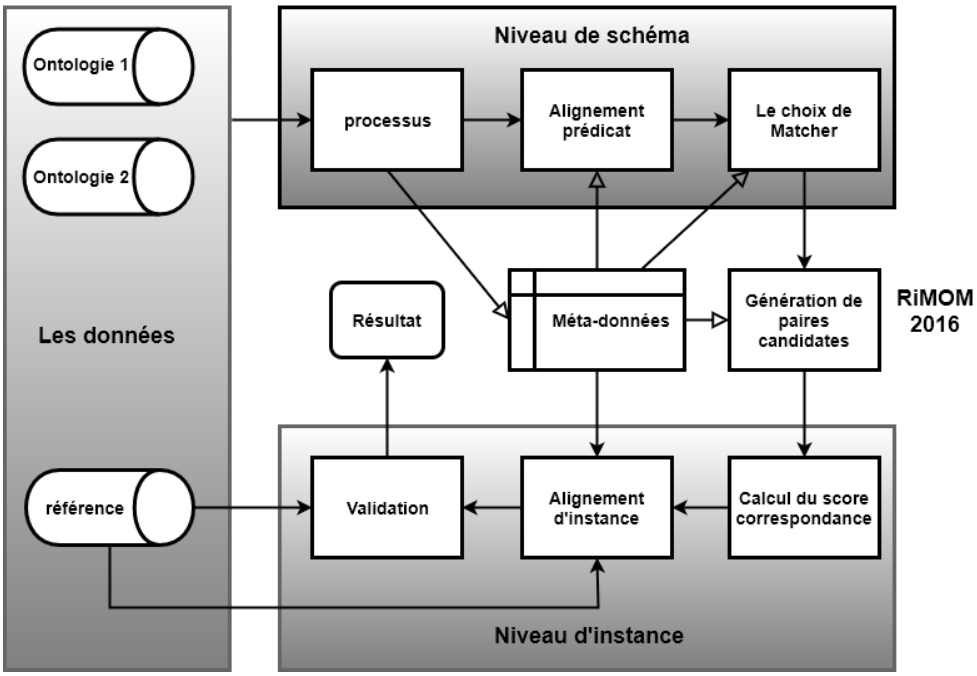


Figure 18 [32] *framework de RiMOM, traduite par S.AMIAR*

❖ **SimCat**

SimCat [33] est un système d'alignement d'ontologie multilingue. SimCat implémente un « matcher » basé sur les catégories de mots et une traduction basée sur le moteur Yandex[35] pour trouver les correspondances sémantiques entre différents concepts des deux ontologies décrites dans différentes langues naturelles, le système comprend les étapes successives suivantes :

- Extraction et normalisation.
- Traduction et nettoyage.
- Calcul de similarité.
- Identification de l'alignement.
- Adaptations réalisées pour l'évaluation.
- Lié le résultat avec l'ensemble des alignements fournis (en alignement).

2.10. Conclusion

Nous avons vu l'ensemble des techniques, méthodes et stratégies qui nous permettront de réaliser un système d'alignement d'ontologies, le chapitre suivant sera consacré pour la conception et la réalisation d'un système d'alignement d'ontologies sémantiques, notre objectif par la réalisation de ce système par l'utilisation d'une solution personnalisée non basé sur des systèmes d'alignement existants est de se libérer des limitations qui se trouvent sur ces derniers, et d'implémenter notre propre vision, afin de maitriser le processus d'alignement et d'arriver à des résultats meilleurs.

[35] https://translate.yandex.ru/

3. Analyse et conception

3.1. Introduction :

Dans ce chapitre nous allons voir les étapes de conception de notre système **SAO**, ainsi que le résultat de son implémentation et les résultats de sa mise en marche, et un résumé des langages et outils que nous avons utilisés.

3.2. Analyse des besoins :

3.2.1. Expression initiale de besoins :

Réalisation d'un système d'alignement d'ontologies performant, capable d'identifier les relations sémantiques de synonymie, hyponymie et hyperonymie, en plus de la relation d'équivalence basée sur les chaines de caractère.

3.2.2. Objectifs :

- Réduire l'hétérogénéité entre les ontologies.
- Identifier toutes les relations sémantiques et non sémantiques existants.
- Atteindre un temps minimal dans le processus d'alignement.

3.2.3. Utilisateurs :

Un seul utilisateur général qui a accès à toutes les fonctionnalités.

3.2.4. Exigences fonctionnelles :

1. **Chargement**
   L'utilisateur doit pouvoir charger une ou plusieurs ontologies, et afficher les informations concernant chaque ontologie (nom, nombre de concept, nombre de labels).
2. **Enrichissement :**
   La fonctionnalité d'enrichissement doit permettre à l'utilisateur d'enrichir les concepts par leur synonyme, hyponyme et hyperonyme, via une ressource externe UMLS.
3. **Alignement :**
   Cette fonctionnalité nous permettra de retrouver les correspondances entre les ontologies chargées selon des mesures de similarité de chaine de caractère, et identifier les relations d'hyponymie et d'hyperonymie.
4. **Visualisation :**
   L'utilisateur doit pouvoir visualiser les ontologies chargées et les relations reliant ces ontologies.
5. **Évaluation :**
   Le système doit permettre de comparer l'alignement généré à un alignement référentiel avec une mesure de précision, rappel et f-mesure.

3.2.5. Exigences non fonctionnelles :

4. **Exigence de qualité :**

- Ergonomie sobre et efficace.

- Un rappel et une précision supérieure à 80%.

5. **Exigence de performance**
   - Supporter les ontologies de grande taille (test sur LargeBio_dataset_oaei2016 : oaei_FMA_whole_ontology et oaei_NCI_whole_ontology).
   - Un temps d'alignement minimal inférieur à deux minutes (test sur LargeBio_dataset_oaei2016 : oaei_FMA_small_overlapping_nci.owl et oaei_NCI_small_overlapping_fma.owl sur un processeur Intel(R) Core™ i5-3320M CPU @ 2.60GHz 2.6 GHz).

3.8.1. Contraintes de conception :

- L'utilisateur doit d'abord chargé des ontologies pour les enrichir ou les aligner.
- L'utilisateur doit pouvoir visualiser chaque ontologie seule, et pouvoir visualiser l'ontologie globale construit par les relations identifiées pendant l'alignement.
- L'alignement doit être en format OWL utilisé par OAEI.

3.2. Conception :

Dans cette partie nous allons utiliser des diagrammes UML pour représenter notre système et modéliser les exigences extraites pendant l'analyse de besoins.

Nous allons utiliser quatre types de diagramme UML dans l'ordre suivant :

- Diagramme de cas d'utilisation générale.
  Et pour chaque incrément.
- Diagramme d'activité.
- Diagramme d'interaction.
- Diagramme de classes.
  Et des maquettes pour intégrer les interfaces graphiques.

C'est possible d'utiliser d'autres diagrammes, mais dans notre cas cela nous suffit largement pour exprimer le fonctionnement de notre système.

La figure suivante montre la démarche que nous allons suivre qui est UP (Unified Process) qui est une méthode agile :

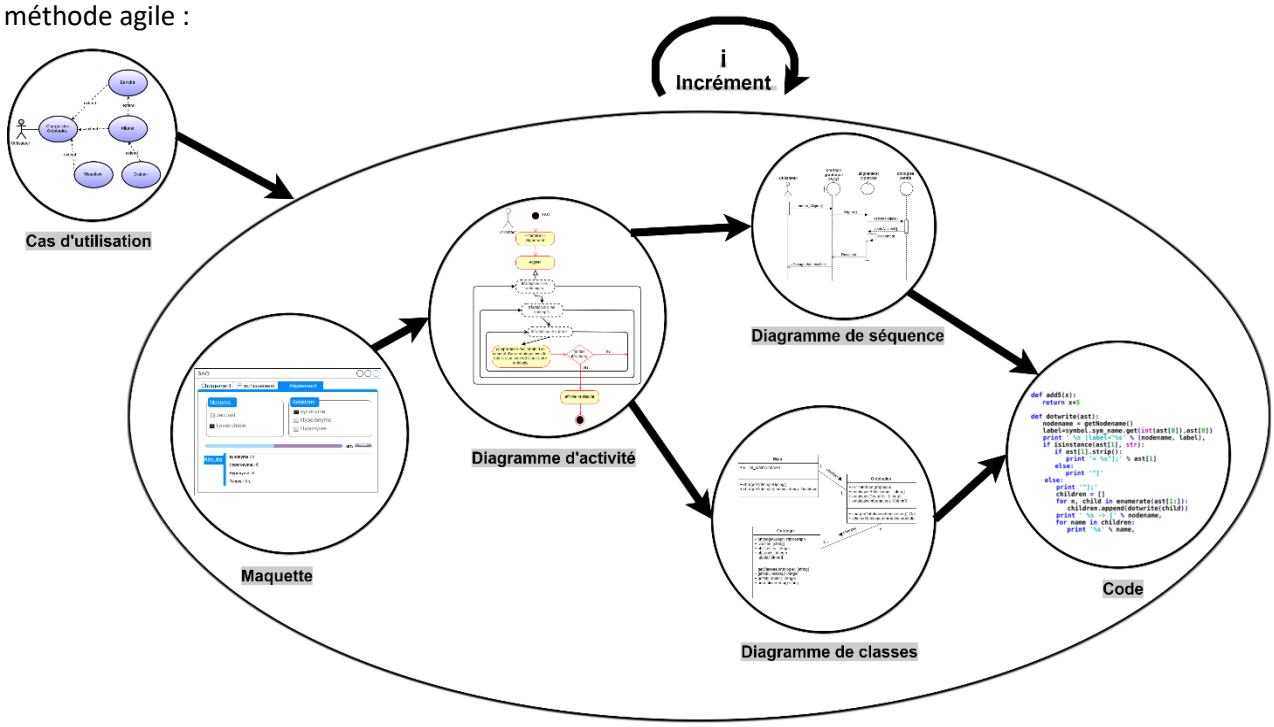


Figure 19 démarche de conception

3.2.1. Diagrammes des cas d'utilisation générale :

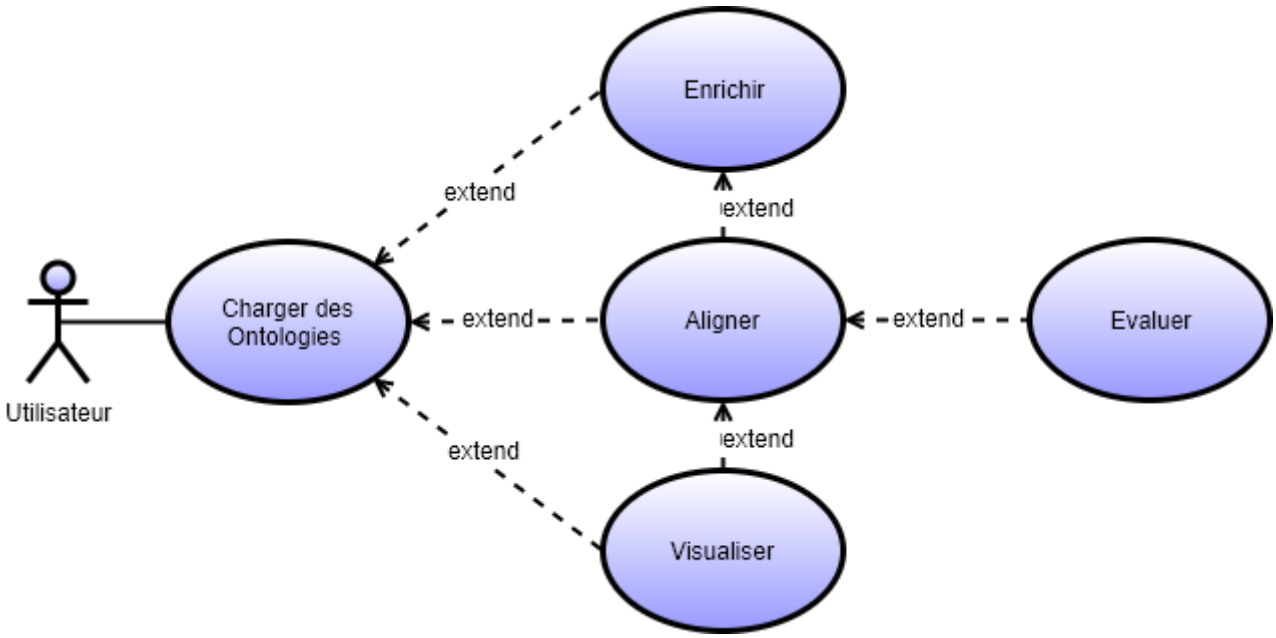


Figure 20 diagramme de cas d'utilisation générale

3.2.2. Les incréments :

L'incrément chargement :

❖ Spécification détaillé du cas d'utilisation chargement :

- **Acteur principale :**
  L'utilisateur.
- **L'objectif :**
  L'utilisateur veut charger des ontologies.
- **Préconditions :**
  - L'utilisateur doit spécifier le chemin des ontologies puis les sélectionner.
- **Post conditions :**
  - Les ontologies sélectionnées sont chargées en mémoires.
  - Affichage des informations concernant chaque ontologie.
- **Scénario nominal :**
  1. L'utilisateur sélection les ontologies a chargé.
  2. Le system de contrôle vérifié si les ontologies sont sous un format supportable.
  3. Le système vérifie pour chaque ontologie si elle est déjà mise en cache.
  4. Le système charge les ontologies en mémoire depuis le cache.
  5. Le système analyse les ontologies et extrait le nom, le nombre de concepts, le nombre de label de chaque ontologie.
- **Alternative :**
  2a l'ontologie n'est pas en cache.

1. Chargement depuis le fichier de l'ontologie.
2. Mise en cache de l'ontologie.

❖ **Diagramme d'activité :**

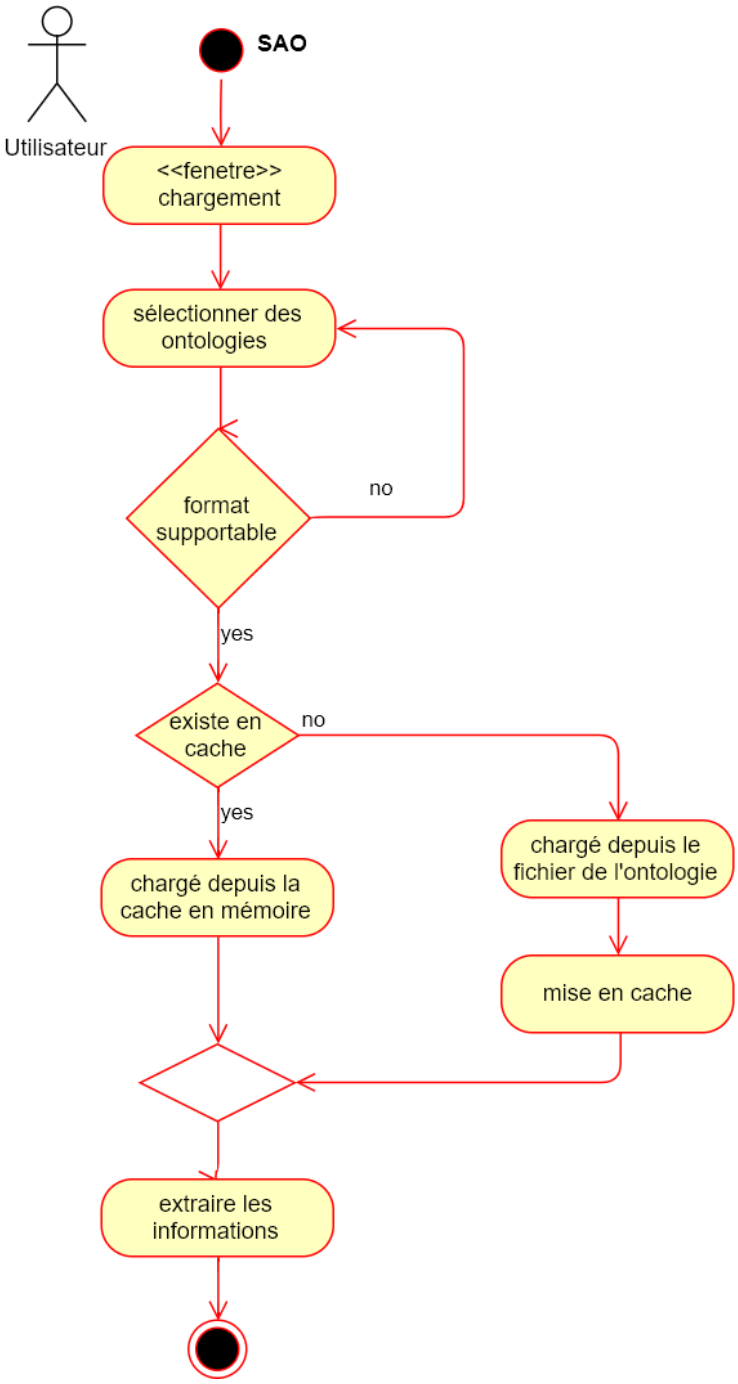


Figure 21 diagramme d'activité "Chargement"

❖ **Diagramme de séquence système :**

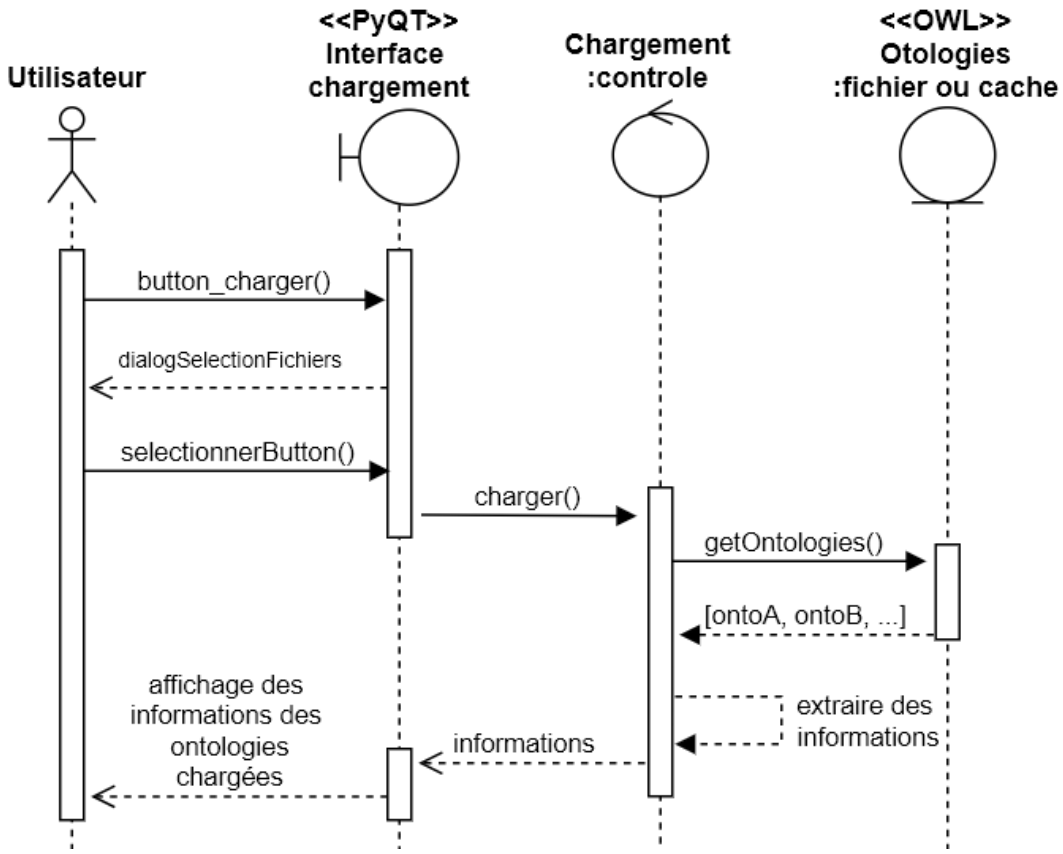


Figure 22diagramme de séquence "Chargement"

❖ **Diagramme de classes :**

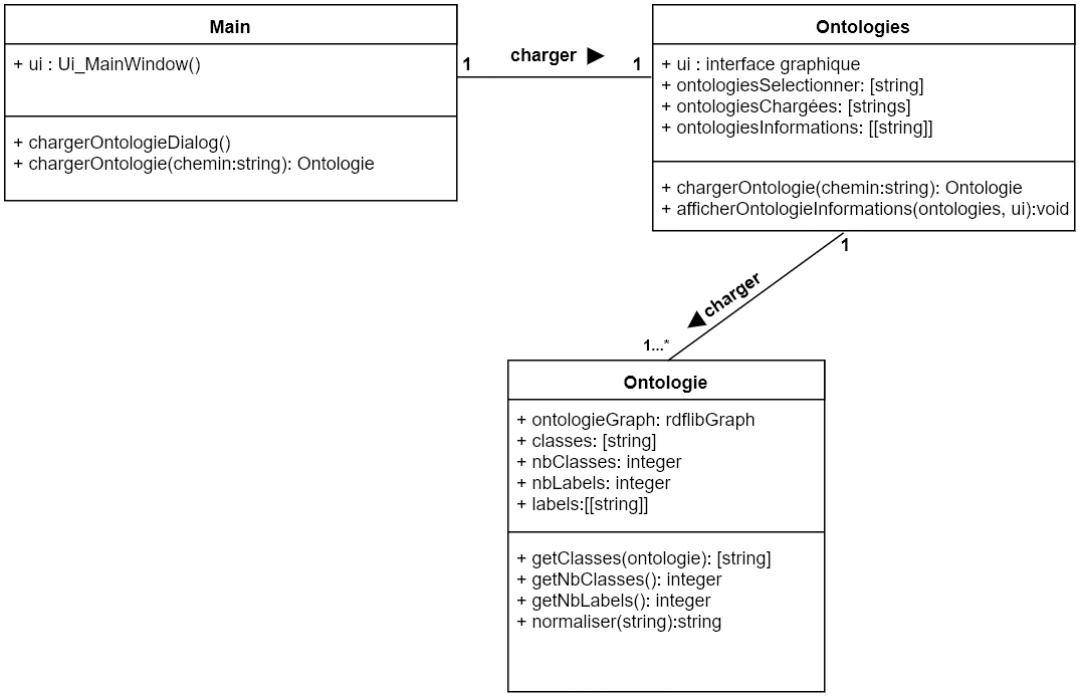


Figure 23 diagramme de classes "Chargement"

❖ **Maquette :**

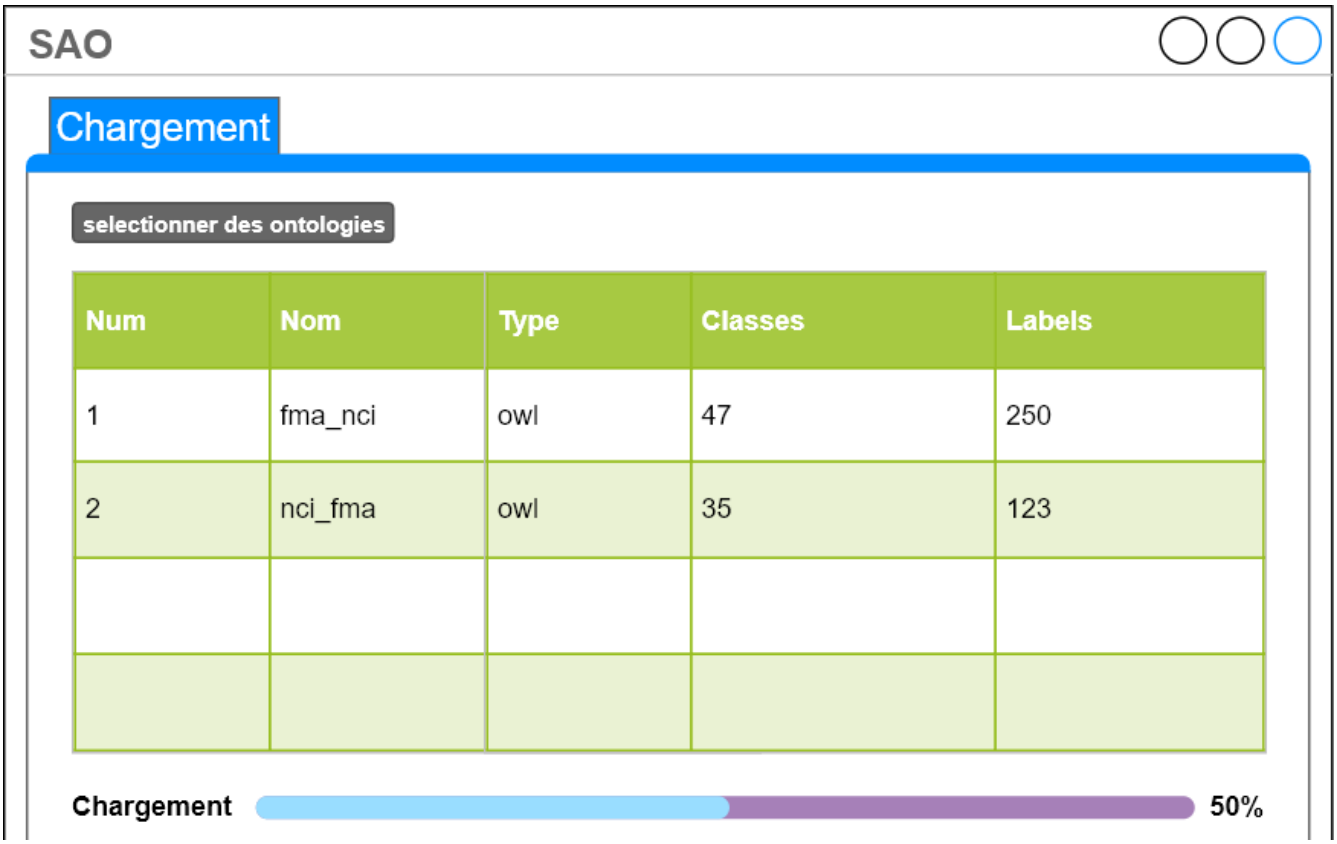


Figure 24 maquette "chargement"

L'incrément enrichissement :

❖ **Spécification détaillé du cas d'utilisation enrichissement :**

- **Acteur principale :** L'utilisateur.
- **L'objectif :**
  L'utilisateur veut enrichir les concepts des ontologies par des synonymes, des hyponymes, et des hyperonymes par utilisations d'une ressource externe qui est UMLS.

- **Préconditions :**
- L'utilisateur doit avoir chargé les ontologies à enrichir.
- UMLS doit être accessible en local ou à distance.
- **Post conditions :**
- Chaque concept retrouvé dans UMLS est enrichi.
- **Scénario nominal :**

1. L'utilisateur appuie sur le bouton enrichir.
2. Le system interroge la ressource UMLS avec les concepts.
3. Les concepts retrouvés sont enrichis.

- **Alternative :**

2a UMLS ne contient pas de correspondant au concept.

3. Le concept n'est pas enrichi.

❖ **Diagramme d'activité :**

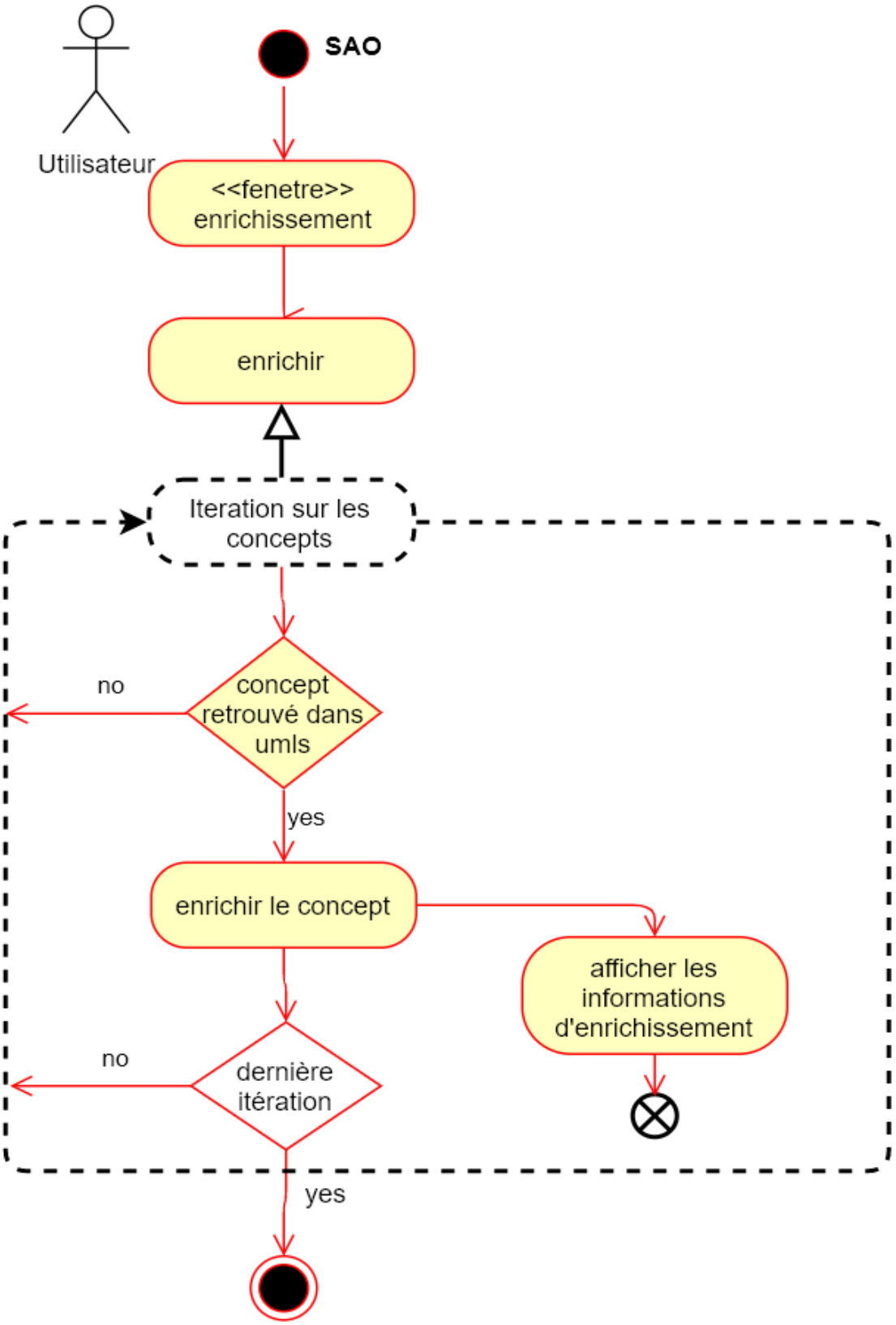


Figure 25 diagramme d'activité "Enrichissement"

Diagramme de séquence :

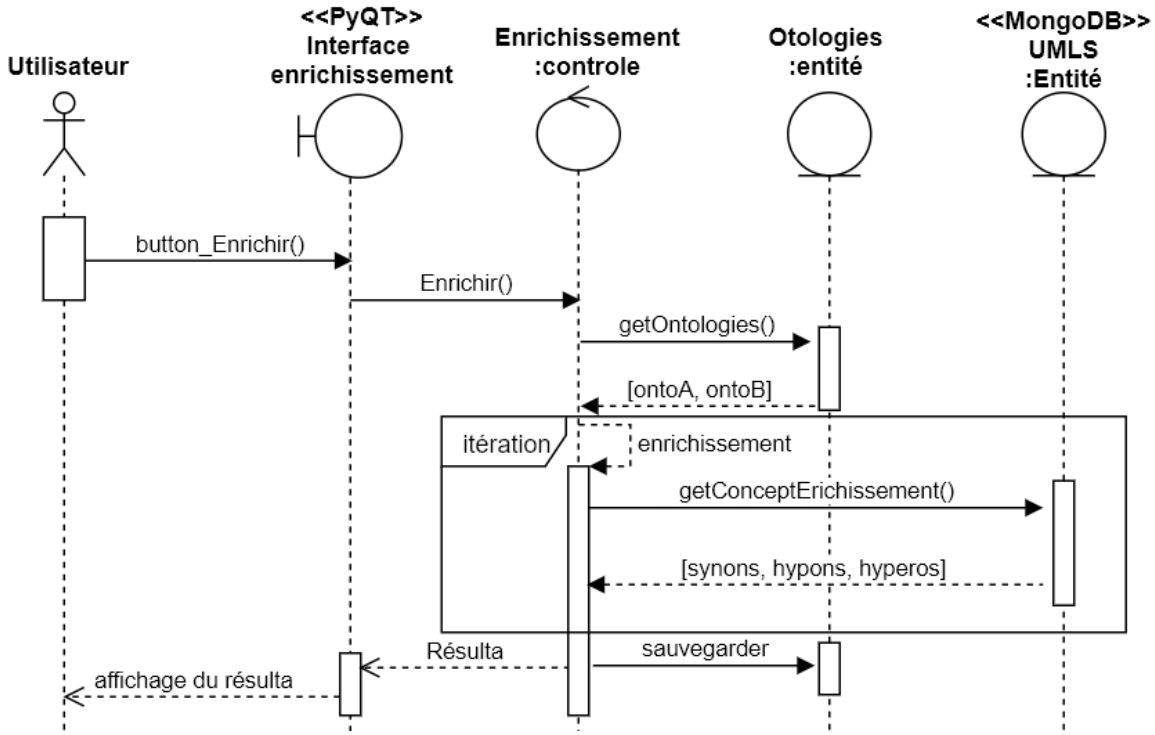


Figure 26 diagramme de séquence "Enrichissement"

❖ **Diagramme de classes :**

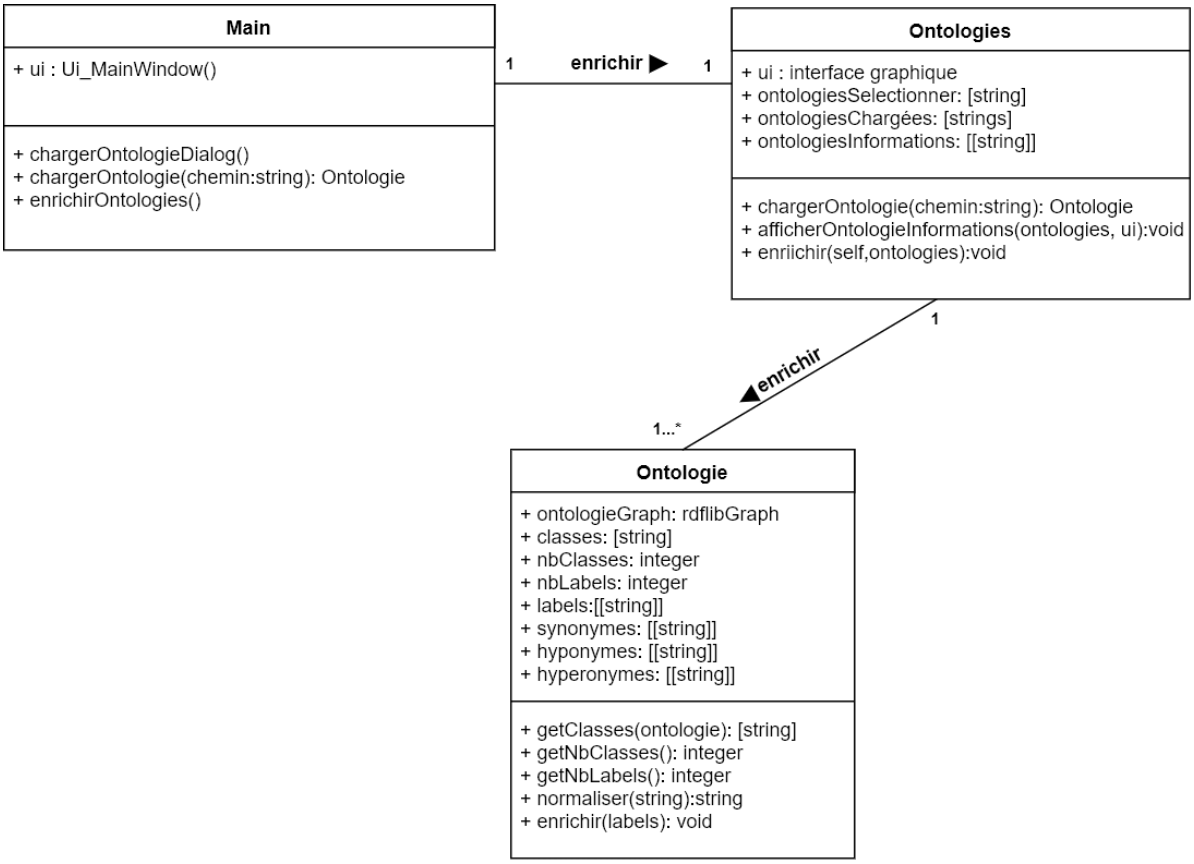


Figure 27 diagramme de classes "Enrichissement"

❖ **Maquette :**

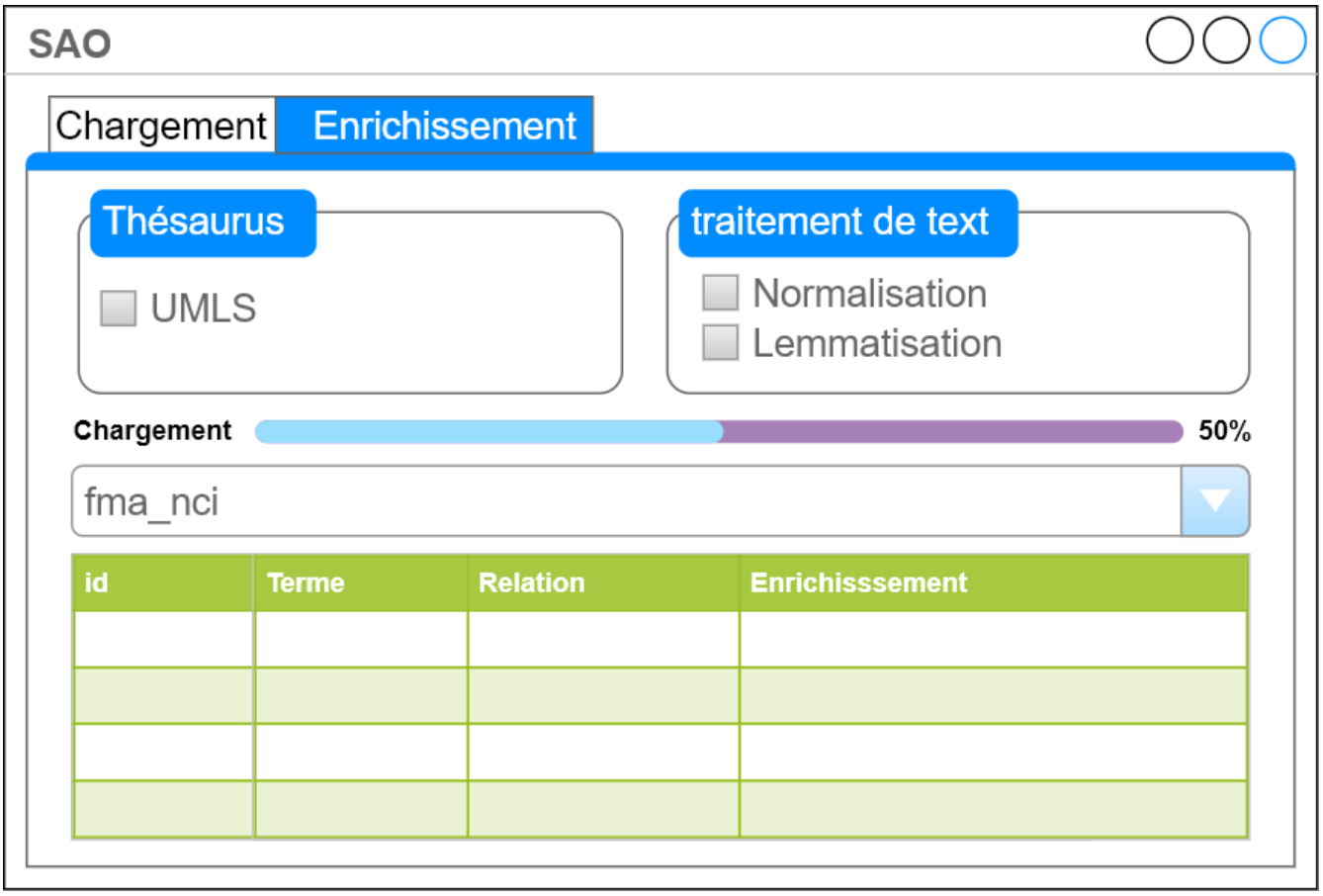


Figure 28 maquette "Enrichissement"

L'incrément alignement :

❖ **Spécification détaillé du cas d'utilisation alignement :**

- **Acteur principale :** Utilisateur.
- **L'objectif :**
  Aligner les ontologies chargées.
- **Préconditions :**
  Chargement des ontologies à aligner.
- **Post conditions :**
  - Un ensemble de correspondances et de relation est retrouvé entre les ontologies.
  - Les relations identifiées sont appliquées pour lier l'ensemble des ontologies aligné.
  - Un fichier d'alignement est généré.
- **Scénario nominal :**
  1. L'utilisateur appuie sur le bouton aligner.
  2. Le système identifie les relations entre les ontologies deux à deux.
  3. Le système ajoute les relations identifiées aux ontologies concernées.
  4. Le système génère l'alignement.

❖ **Diagramme d'activité :**

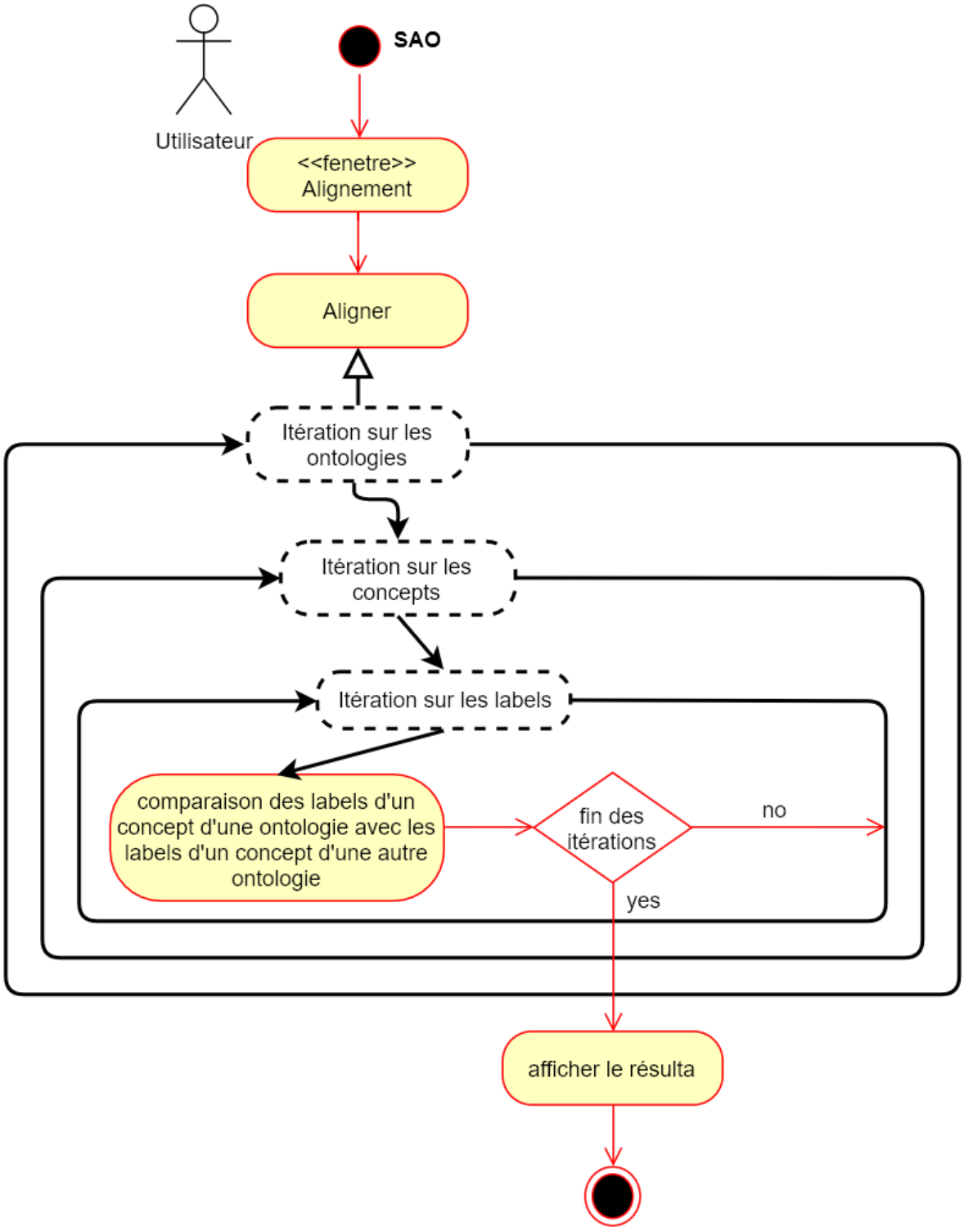


Figure 29 diagramme d'activité "Alignement"

❖ **Diagramme de séquence :**

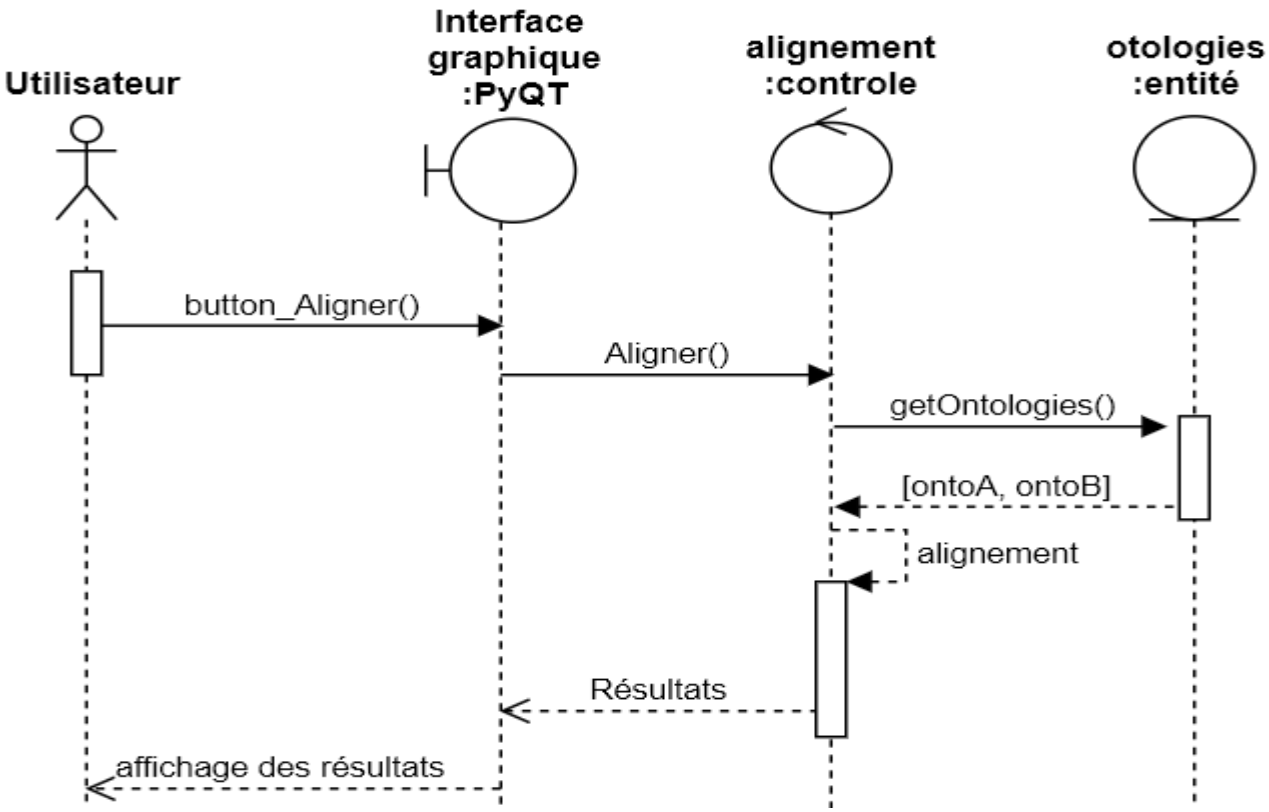


Figure 30 diagramme de séquence "Alignement"

❖ **Diagramme de classes :**

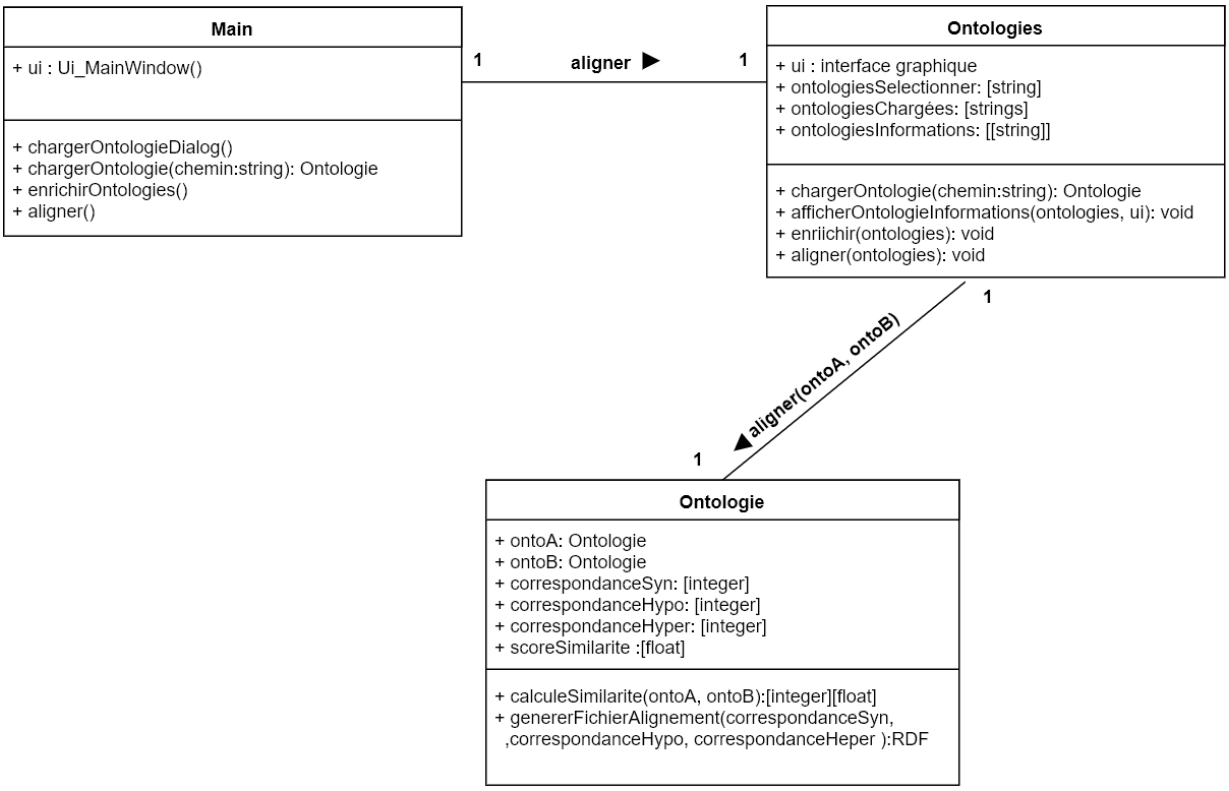


Figure 31 diagramme de classes "Alignement"

❖ **Maquette :**

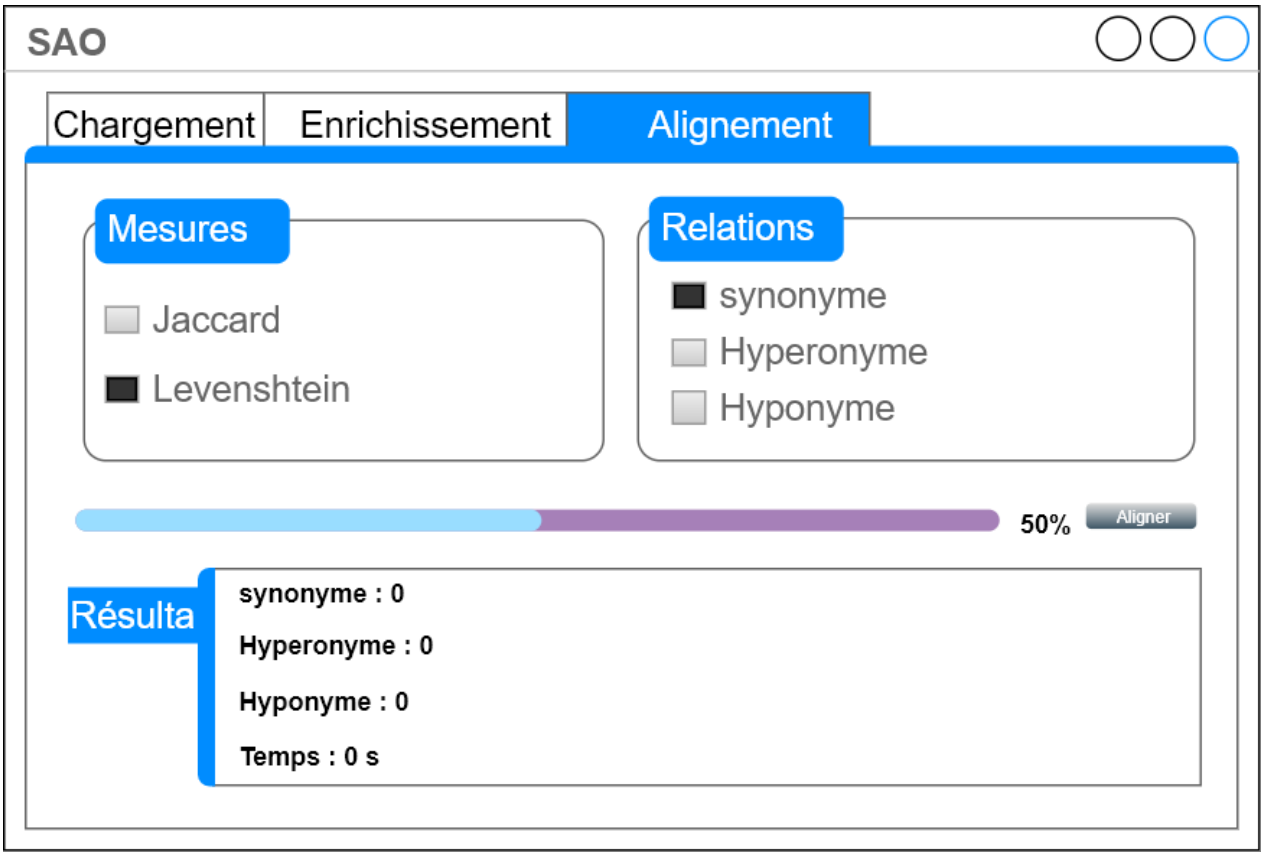


Figure 32 maquette "Alignement"

L'incrément visualisation :

❖ **Spécification détaillé du cas d'utilisation visualisation :**

- **Acteur principale :**
  Utilisateur.
- **L'objectif :**
  Visualiser les ontologies chargées et l'ontologie globale générée par l'alignement.
- **Préconditions :**
  Chargement des ontologies à visualiser.
- **Post conditions :**
  L'ontologie sélectionnée est graphiquement visible.
- **Scénario nominal :**
  5. L'utilisateur sélectionne l'ontologie à visualiser.
  6. Le système convertit l'ontologie au format utilisé pour la visualisation.
  7. Le système charge l'ontologie convertie puis il la visualise.

❖ **Diagramme d'activité :**

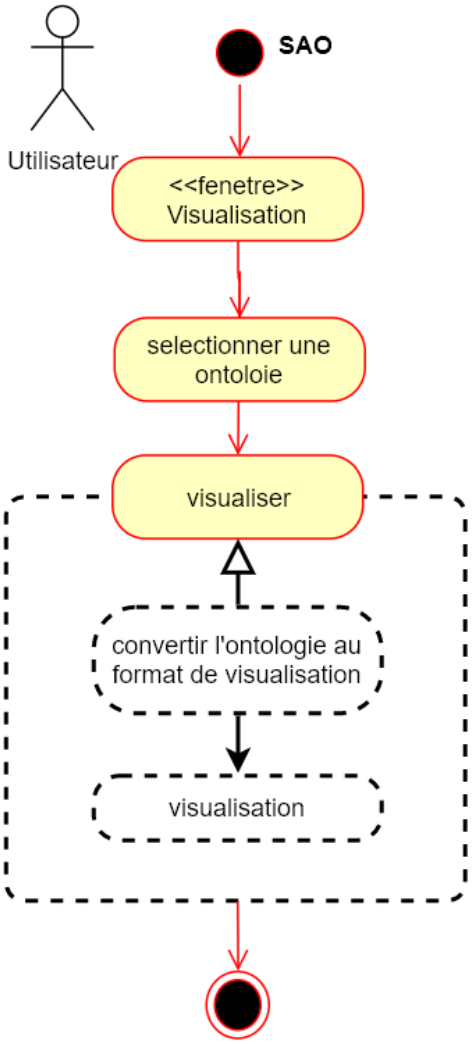


Figure 33 diagramme d'activité "Visualisation"

❖ **Diagramme de séquence :**

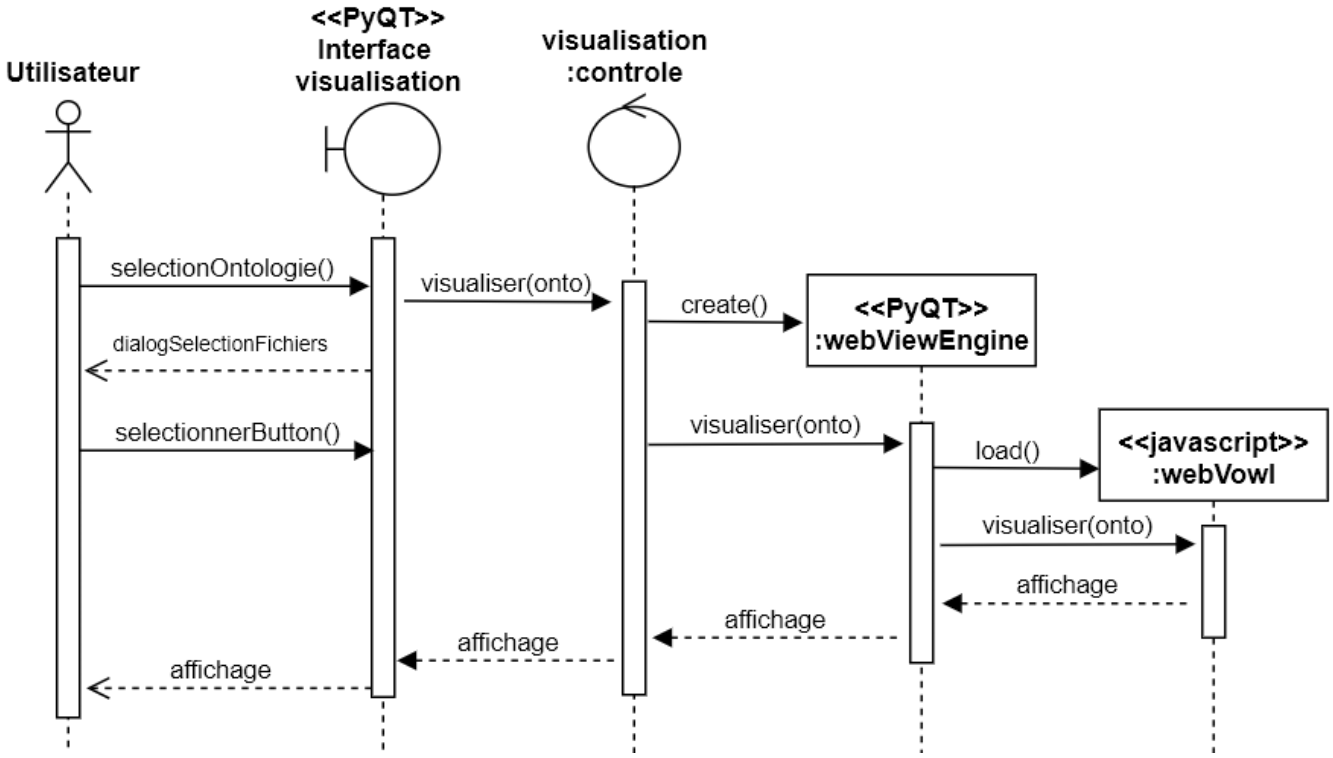


Figure 34 diagramme de séquence "Visualisation"

❖ **Diagramme de classes :**

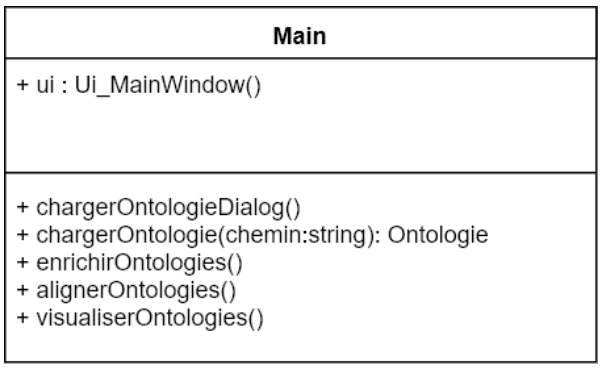


Figure 35 diagramme de classes "Visualisation"

❖ **Maquette :**

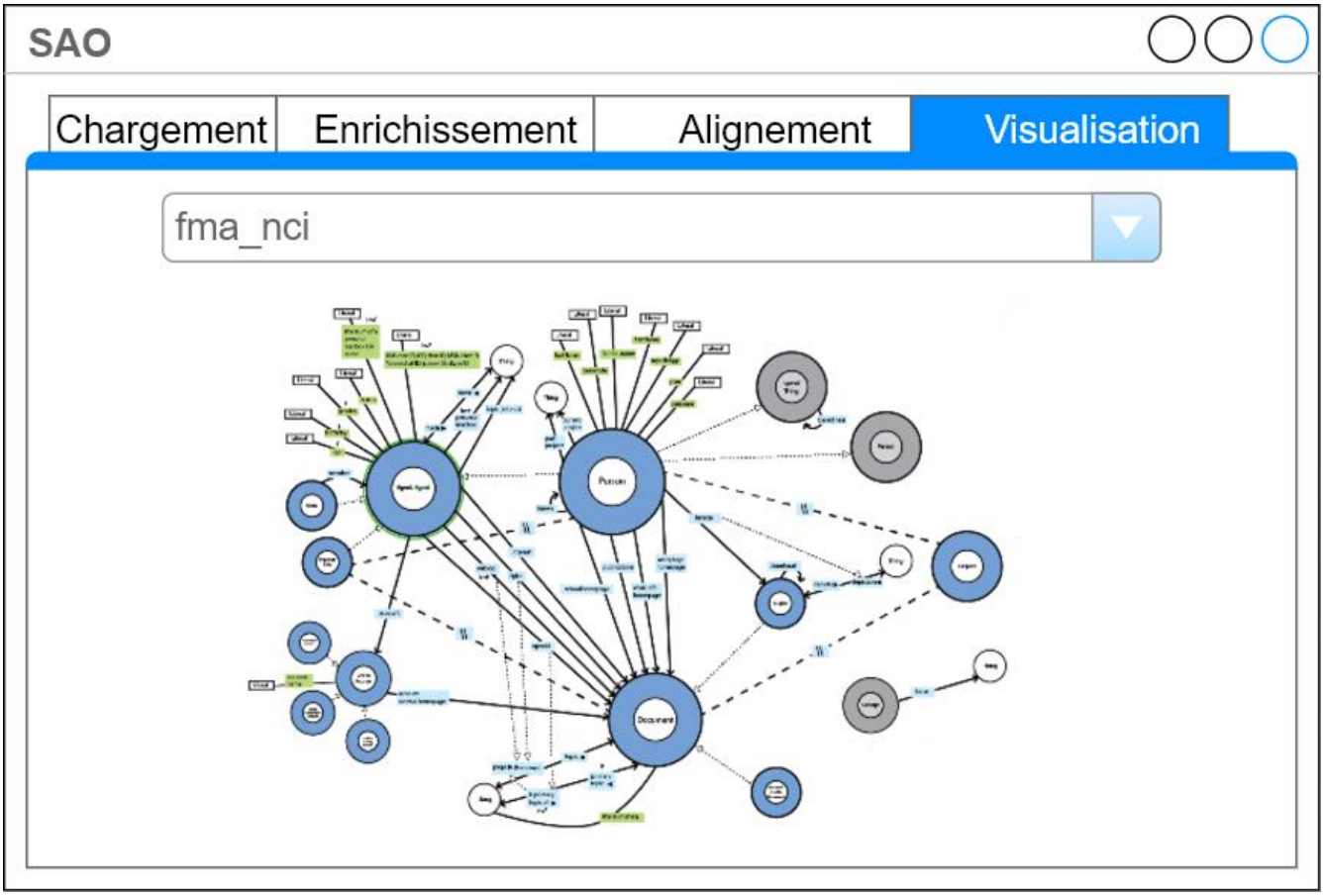


Figure 36 maquette "Visualisation"

L'incrément évaluation :

❖ **Spécification détaillé du cas d'utilisation évaluation :**

- **Acteur principale :**
  Utilisateur.
- **L'objectif :**
  Évaluer l'alignement par rapport à un alignement référentiel.
- **Préconditions :**
  Un alignement est généré.
- **Post conditions :**
  Calcule du rappel, précision et f-mesure.
- **Scénario nominal :**
  8. L'utilisateur se rend à l'onglet Évaluation.
  9. Le système affiche les résultats d'évaluation.

❖ **Diagramme d'activité :**

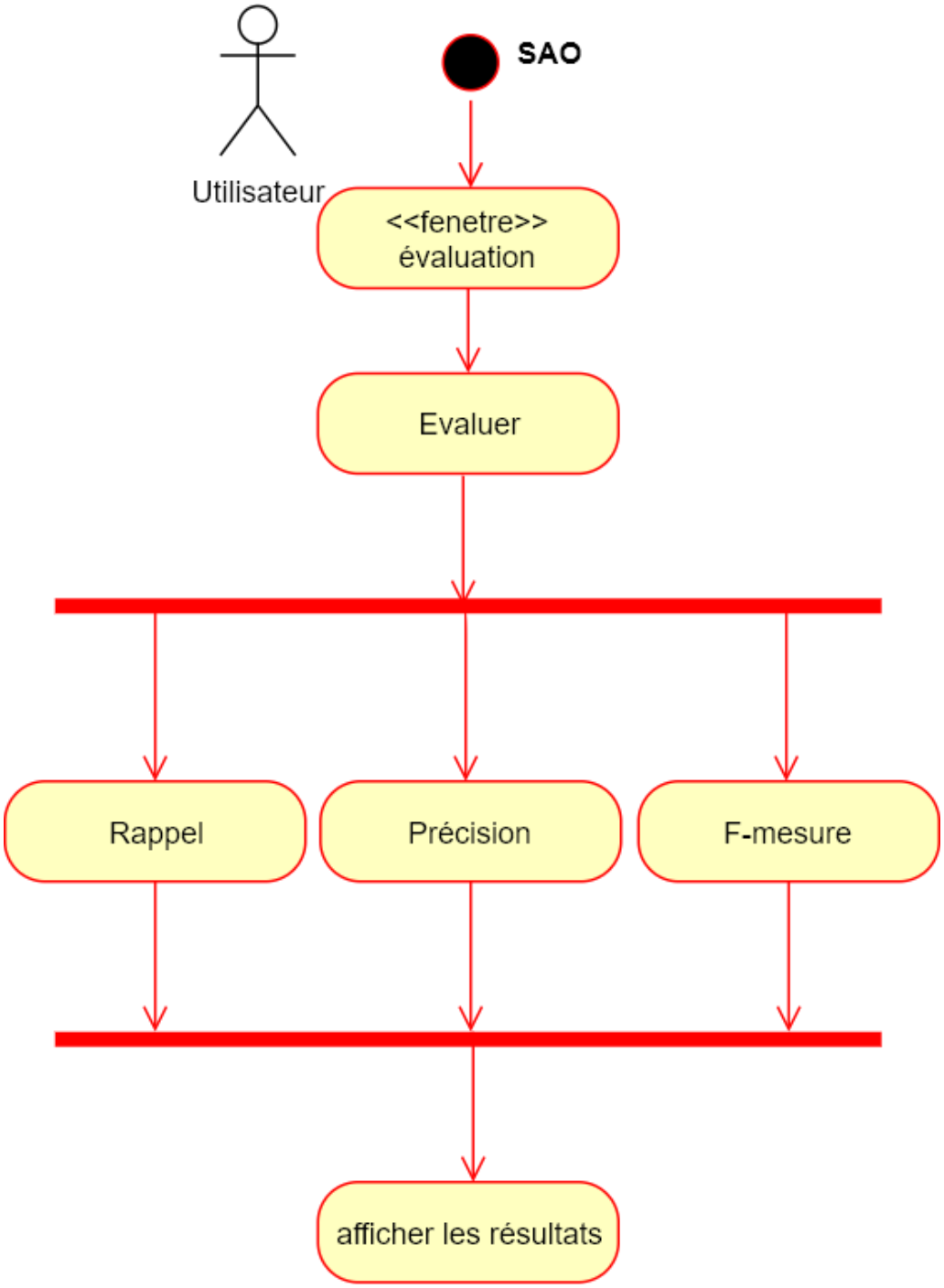


Figure 37 diagramme d'activité "Evaluation"

❖ **Diagramme de séquence :**

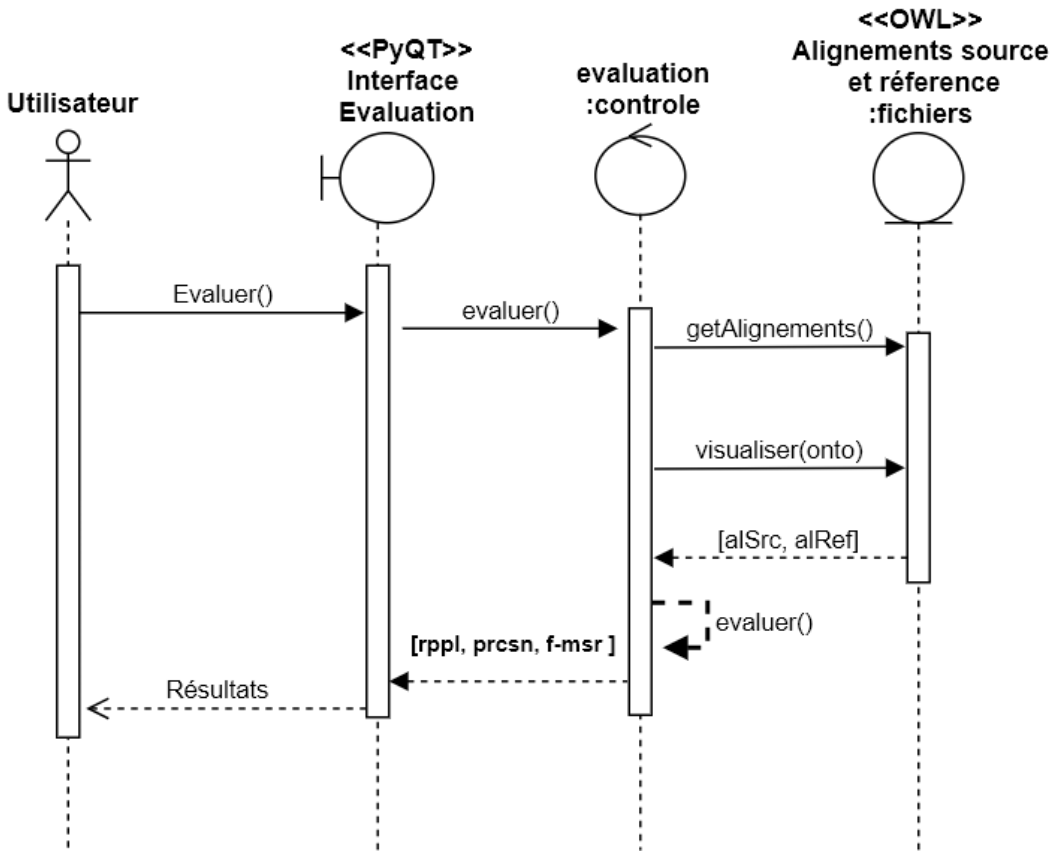


Figure 38 diagramme de séquence "Evaluation"

❖ **Diagramme de classes :**

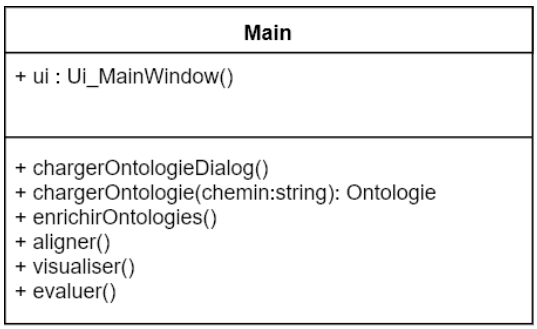


Figure 39 diagramme de classes "Evaluation"

❖ **Maquette :**

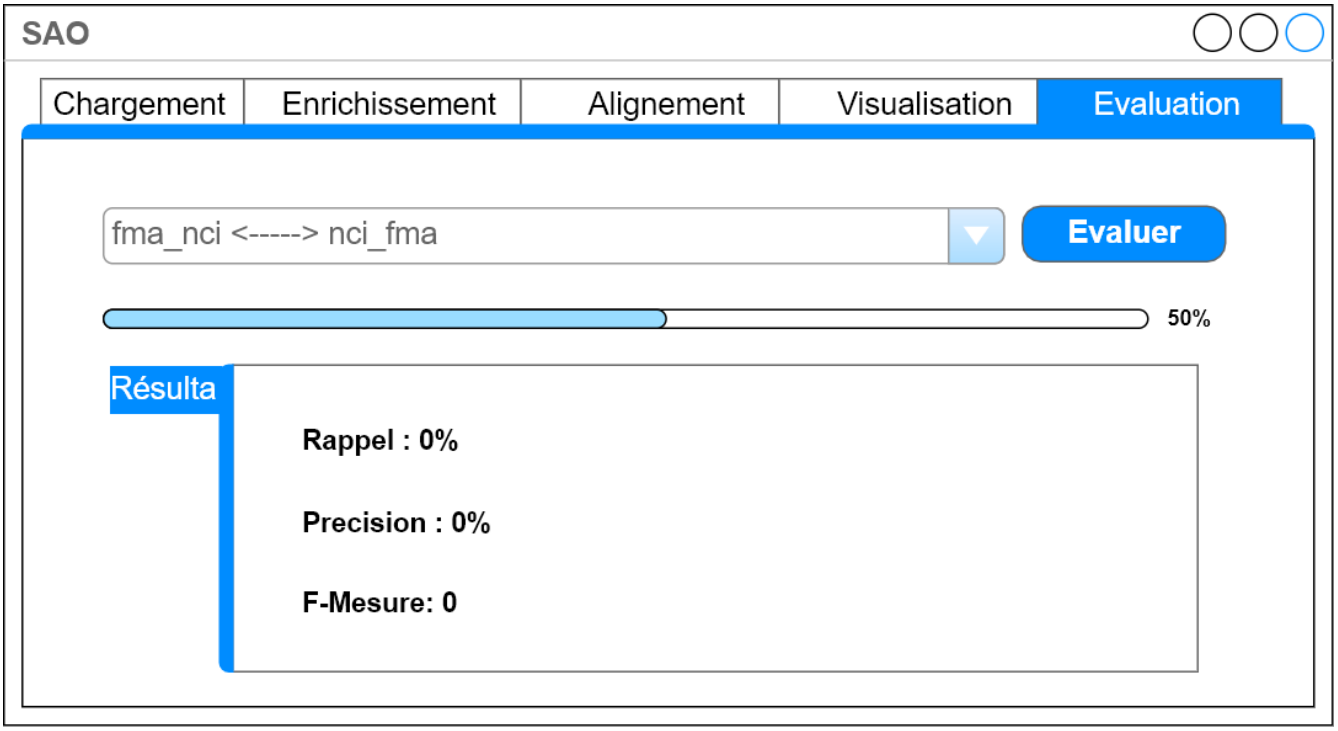


Figure 40 maquette "Evaluation"

L'utilisation de modèles UML nous a permis de représenter notre système en différentes couches d'abstraction, de simuler son fonctionnement et de structurer d'une manière cohérente ses fonctionnalités et données, afin de nous faciliter la phase de développement de chaque fonctionnalité.

Ce qui apporte aussi une compréhension rapide du programme à d'autres développeurs externes en cas de reprise du logiciel et facilite sa maintenance.

3.3. Architecture :

Le schéma suivante montre le système complet avec toutes les parties participantes :

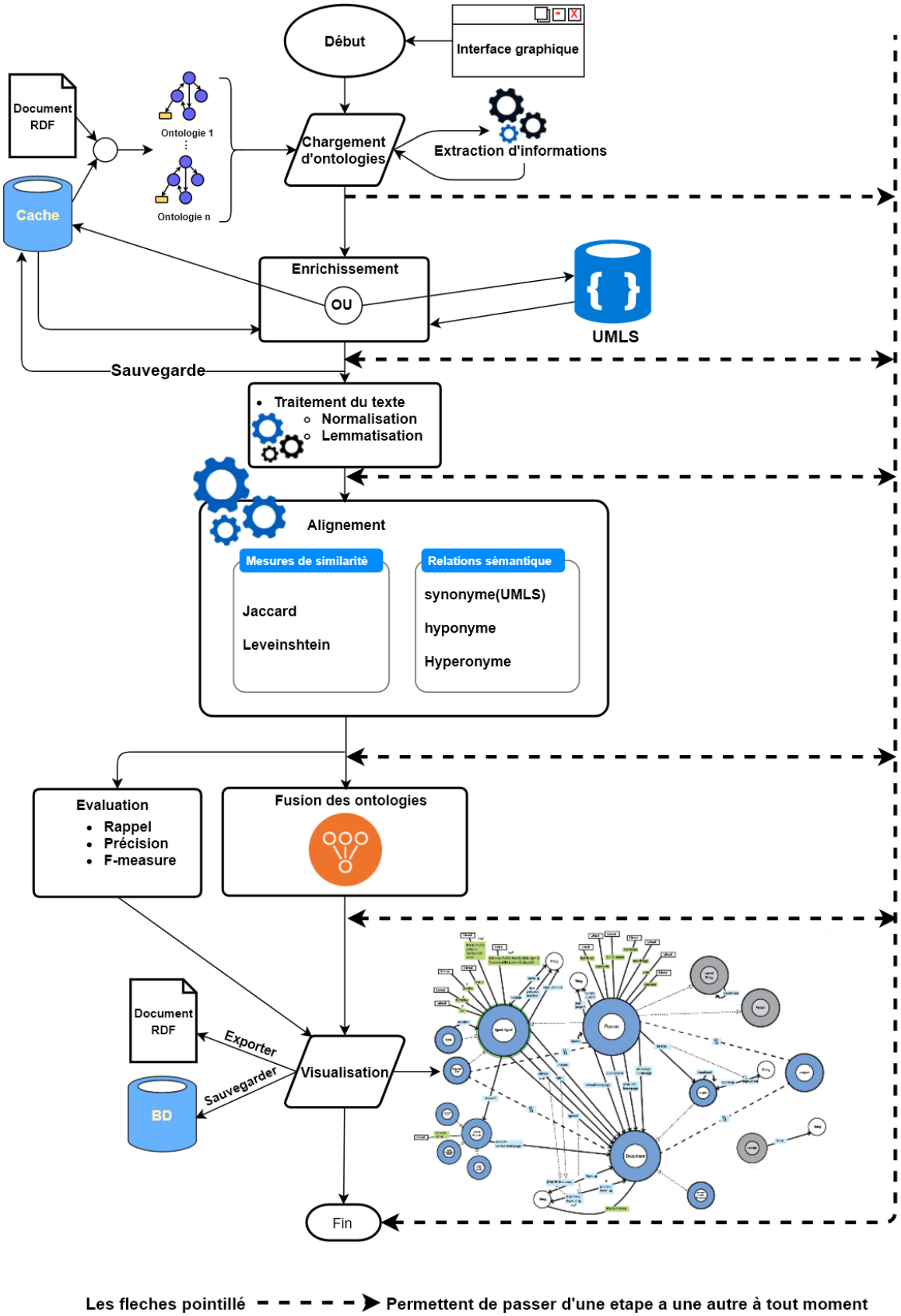


Figure 41 Architecture du système SAO

Conclusion :

Nous venons de voir la phase finale de notre projet qui est la phase de conception et réalisation, dans cette partie nous avons appliqué une méthode agile UP (unified process), qui consiste dans notre cas à identifier les exigences et concevoir globalement le fonctionnement de notre système, puis tirer les tâches existantes et les séparer dans des incréments, et faire une conception détaillée et une implémentation pour chaque tâche d'un incrément. Ainsi cette partie nous à permet d'appliquer les connaissances acquises pendant les phases précédentes (phases de documentation), notamment l'ingénierie des ontologies, les langages de programmation (Python, Cython, PyQT, OWL), la maitrise de nouvelles technologies et bibliothèques (rdflib, nltk (natural language tool kit), webvowl, etc.), outils de maquettage et de dessin de diagramme (Draw.io, Pencil), maitriser des environnements de développement (JetBrains PyCharm, QT Designer), ces derniers point des langages et outils nous allons les voir dans le chapitre suivant

4. Réalisation :

4.1. Captures d'écran :

Nous allons voir des captures d'écran de notre logiciel étant en marche :

❖ Chargement :

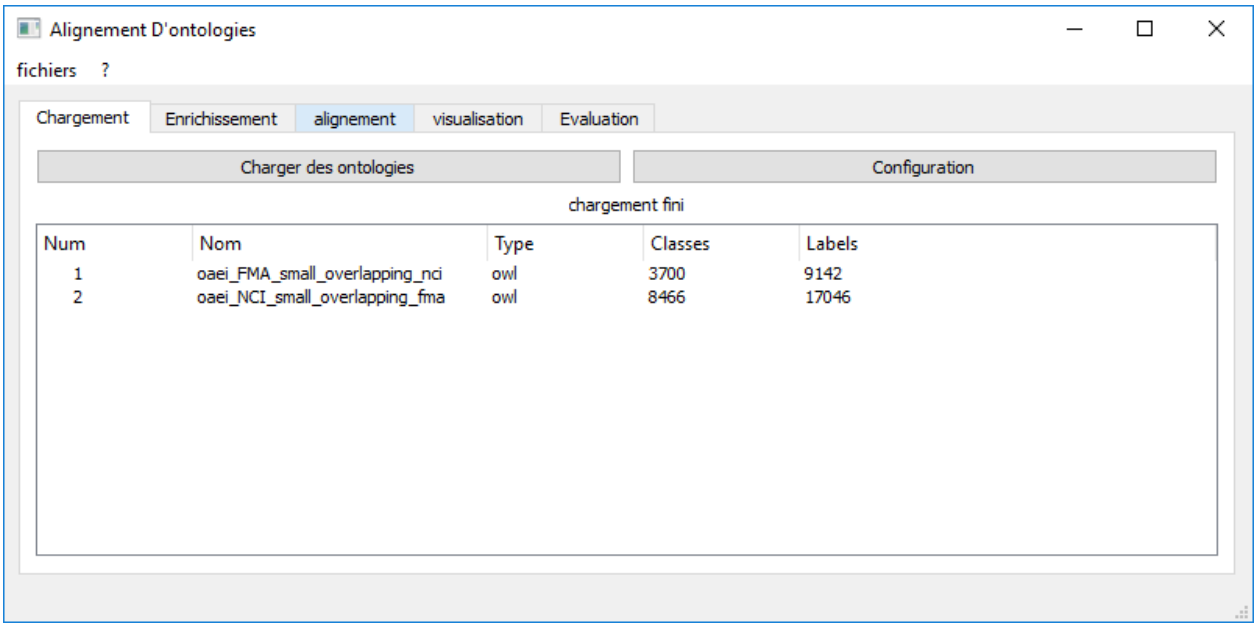


Figure 42 capture d'écran "Chargement"

❖ Enrichissement :

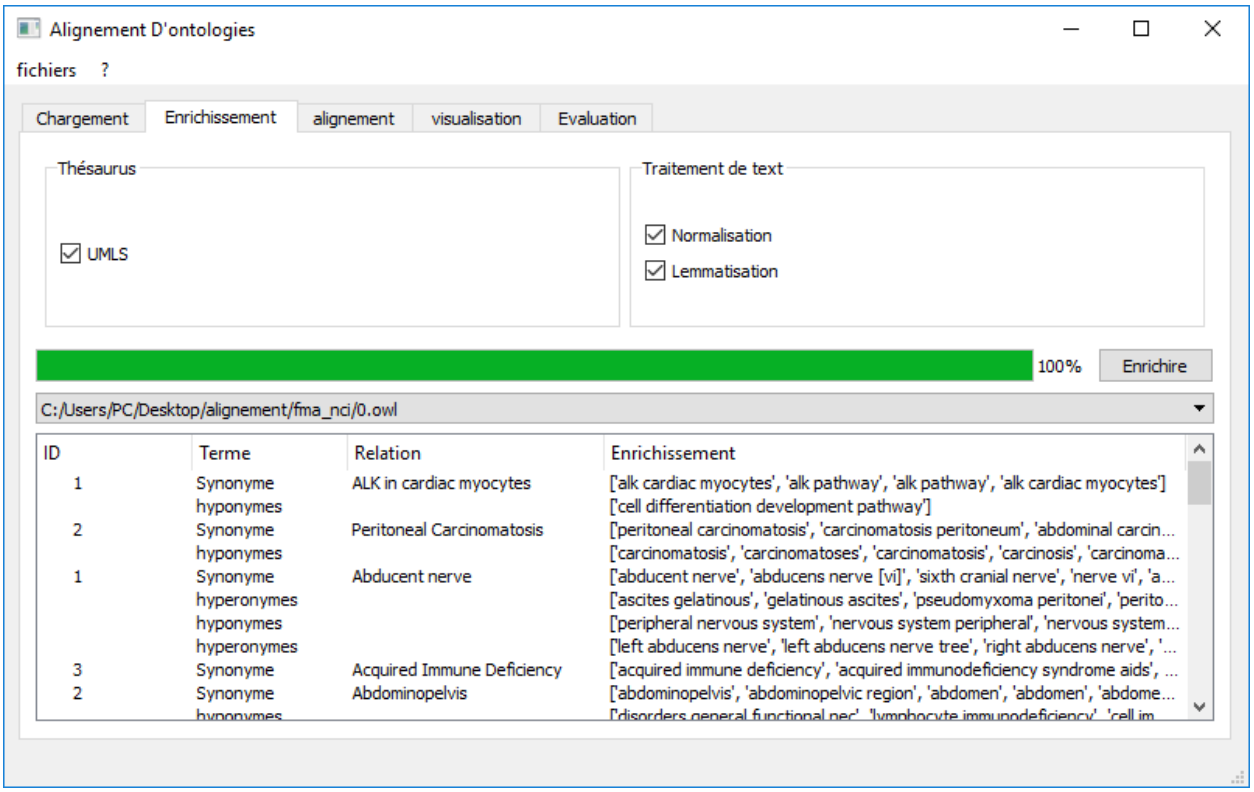


Figure 43 capture d'écran "Enrichissement"

❖ **Alignement :**

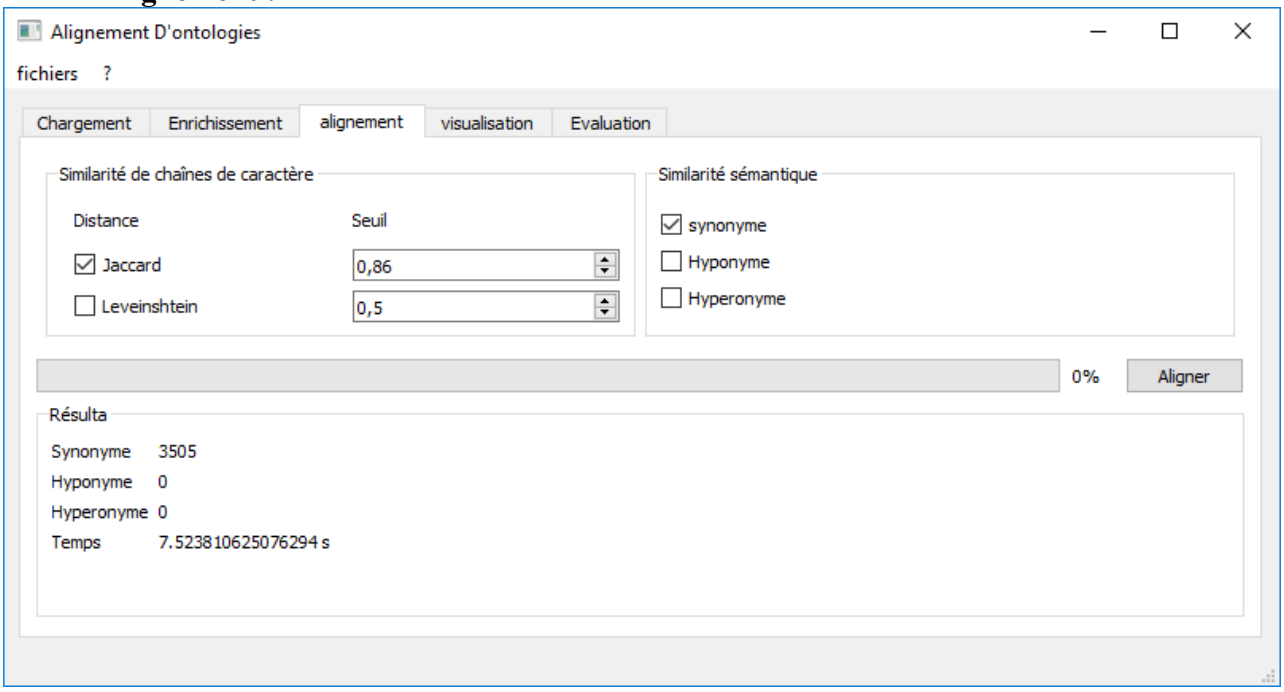


Figure 44 capture d'écran "Alignement"

❖ **Visualisation :**

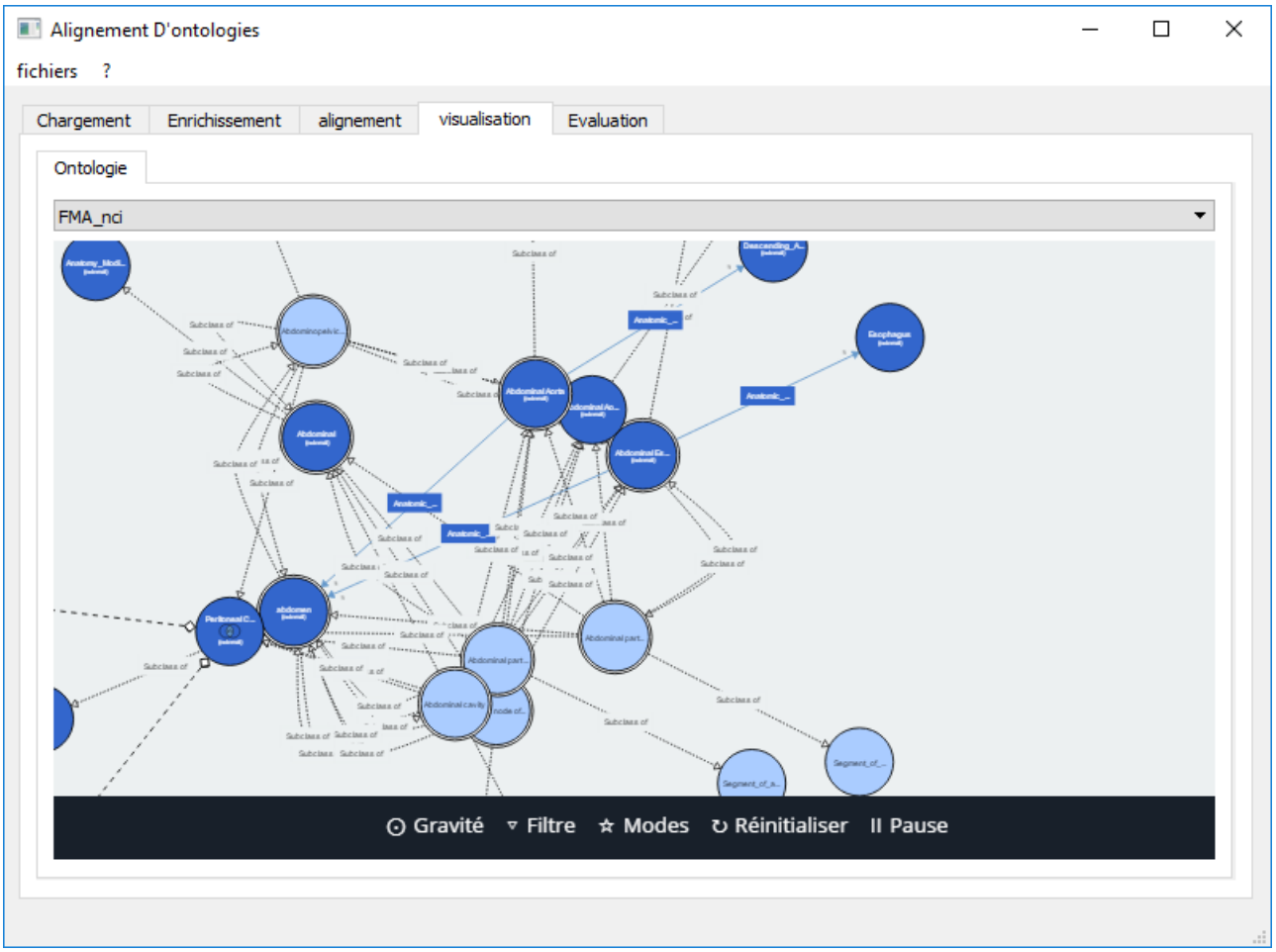


Figure 45 capture d'écran "Visualisation"

❖ **Évaluation :**

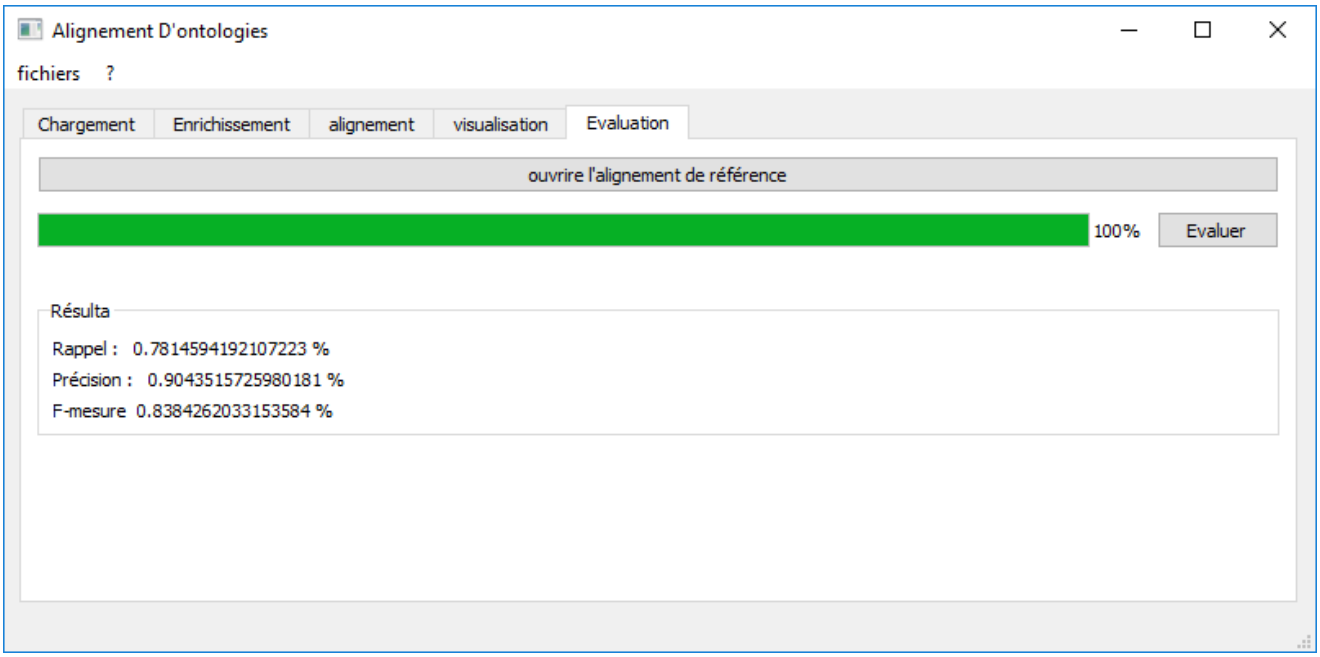


Figure 46 capture d'écran "Evaluation"

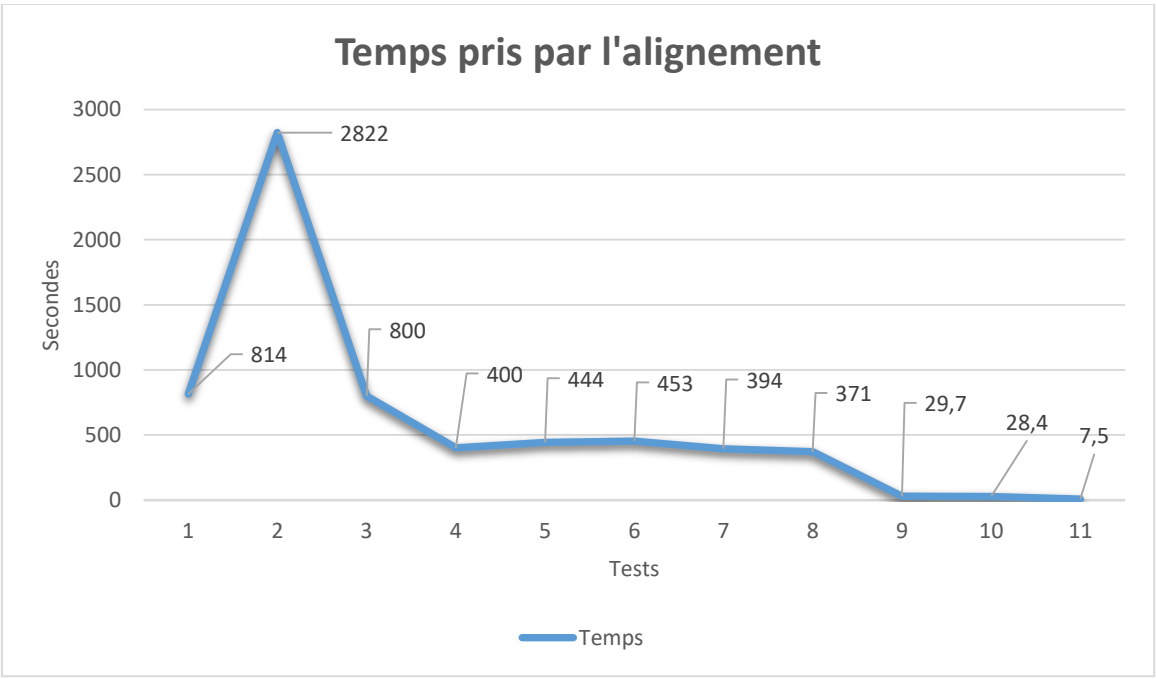
4.2. Evaluation :

Les ontologies sur lesquels nous avons évalué notre système sont :

- oaei_FMA_small_overlapping_nci.owl
- oaei_NCI_small_overlapping_fma.owl

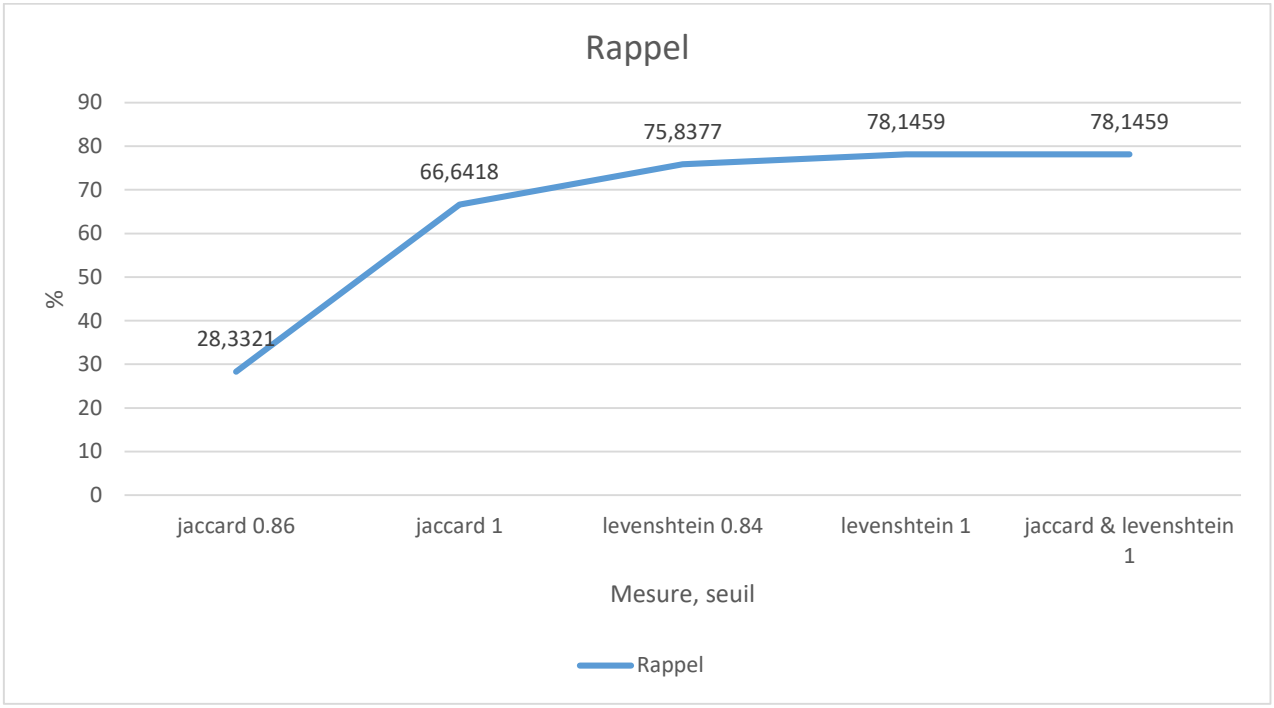
4.2.1. Performance :

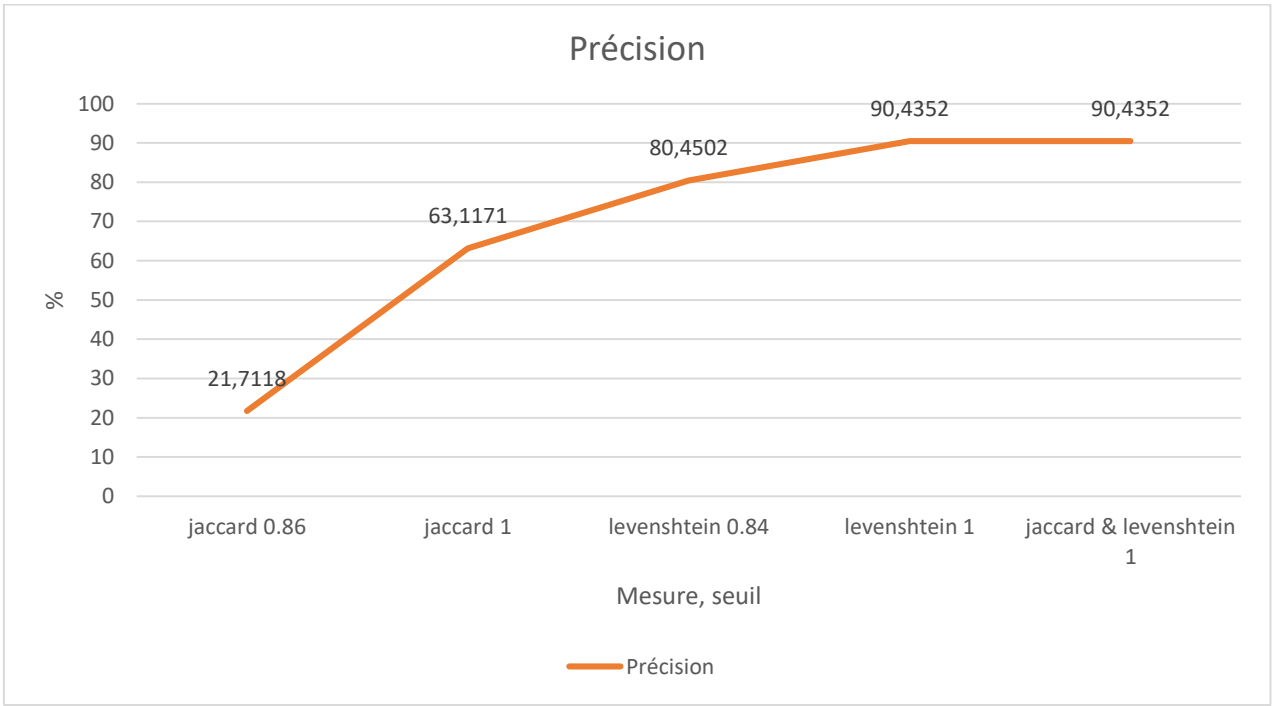
Tests sur un processeur Intel(R) Core™ i5-3320M CPU @ 2.60GHz 2.6 GHz, la mesure utilisé est Jaccard avec un seuil 0,86, les relation à identifier sont les relations de synonymie :

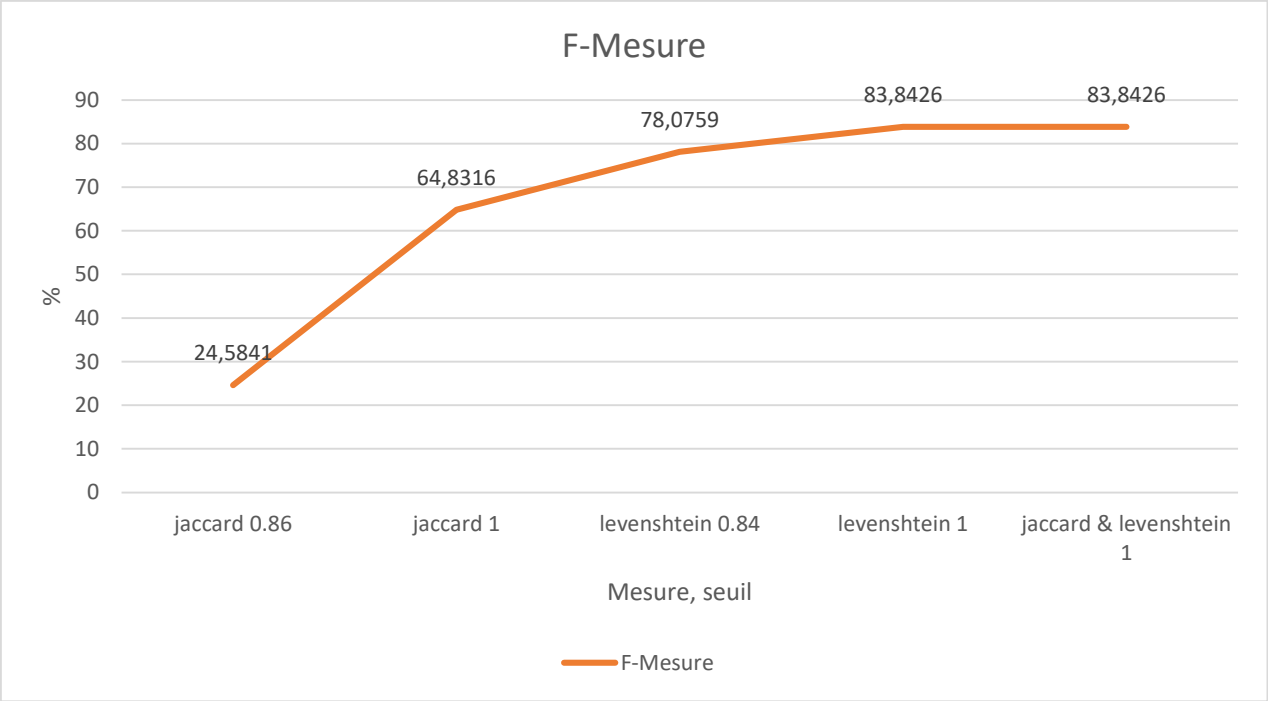


4.2.2. Résultats :

Les graphes suivant montre les résultats obtenus par notre système **SAO** avec deux mesures Jaccard et Levenshtein,







Résultats obtenu dans la conférence OAIE en 2016 pour le problème d'alignement FMA-NCI [19] :

Table 2: Results for the largebio task 1.

System	Time (s)	# Mappings	Scores			Incoherence Analysis	
			Precision	Recall	F-measure	Unsat.	Degree
XMap*	17	2,649	0.977	0.901	**0.937**	2	0.019%
FCA_Map	236	2,834	0.954	0.917	**0.935**	4,729	46.0%
AML	35	2,691	0.963	0.902	**0.931**	2	0.019%
LogMap	10	2,747	0.949	0.901	**0.924**	2	0.019%
LogMapBio	1,712	2,817	0.935	0.910	**0.923**	2	0.019%
LogMapLite	1	2,483	0.967	0.819	**0.887**	2,045	19.9%
Average	1,164	2,677	0.852	0.779	**0.804**	2,434	23.7%
LYAM	1,043	3,534	0.721	0.889	**0.796**	6,880	66.9%
Lily	699	3,374	0.603	0.721	**0.657**	9,273	90.2%

Alin	5,811	1,300	0.995	0.455	**0.625**	0	0.000%
DKP-AOM-Lite	1,698	2,513	0.652	0.577	**0.612**	1,924	18.7%
DKP-AOM	1,547	2,513	0.652	0.577	**0.612**	1,924	18.7%

Nous remarquons que notre système **SAO** peut être classé en 7ème place comparé aux résultats précédant de la compagne OAIE en 2016, avec plus de configuration et d'amélioration de certains points notre système pourra être classé parmi les premiers, parmi ces points on a le processus d'enrichissement qui a besoin d'avoir des traitements de plus pour diminuer le nombre de termes ajouter pour chaque concept par un choix intelligent et enlever les duplications, ajouter de nouvelle mesure de similarités en plus des deux déjà utilisées, utilisation de méthode d'apprentissage automatique, etc.

4.3. Langages et outils :

❖ **Python** [36]:

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et d'une approche de la programmation orientée objet simple mais efficace. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur de nombreuses plateformes.

L'interpréteur Python et sa vaste bibliothèque standard sont disponibles librement, sous forme de sources ou de binaires, pour toutes les plateformes majeures, depuis le site Internet http://www.python.org/ et peuvent être librement redistribués. Le même site distribue et contient des liens vers des modules, des programmes et des outils tiers ainsi que vers de la documentation supplémentaire.

L'interpréteur Python peut être facilement étendu par de nouvelles fonctions et types de données implémentés en C ou C++ (ou tout autre langage appelable depuis le C). Python est également adapté comme langage d'extension pour personnaliser des applications.

[36] https://www.python.org/

❖ **PyQT** [37]:

PyQt est un module qui permet de lier le langage Python avec la bibliothèque Qt. Il permet ainsi de créer des interfaces graphiques en python. Une extension de QtDesigner (utilitaire graphique de création d'interfaces Qt) permet de gérer le code python d'interfaces graphiques. PyQt dispose de tous les avantages liés à Qt.

Qt est une bibliothèque logicielle offrant essentiellement des composants d'interface graphique (communément appelés widgets), mais également d'autres composants non-graphiques permettant entre autre l'accès aux données, les connexions réseaux, la gestion des files d'exécution, etc. Elle a été développée en C++ par la société Trolltech et est disponible pour de multiples environnements Unix utilisant X11 (dont Linux), Windows et Mac OS.

❖ **Cython** [38]:

Cython est un langage de programmation qui simplifie l'écriture d'extension compilées pour Python ainsi que l'interfaçage des programmes Python avec des bibliothèques externes. La syntaxe du langage est très similaire à Python mais il supporte l'appel à des fonctions en C et la déclaration de variables et d'attributs de classes de type C. Il est traduit en langage C qui peut être utilisé avec Python. Il permet donc également de générer des exécutables compilés, alors que Python est à l'origine un langage interprété, offrant ainsi des gains de performance en vitesse d'exécution (par rapport au Python).

[37] https://riverbankcomputing.com/software/pyqt/intro
[38] http://cython.org/

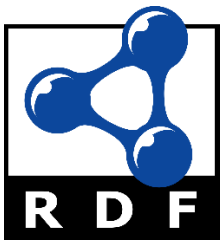
❖ **Qt Creator [39]:**

Qt Creator est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto complétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows.

❖ **MongoDB [40]:**

MongoDB est une base de données **NoSQL** (not only sql) **de type document**, sans schéma. Il est flexible et peut fonctionner efficacement avec de grandes quantités de données. Il gère des collections (équivalents des tables pour MySQL) de documents JSON-like stockés dans un format binaire (BSON), Open Source et écrite en C++.

❖ **RDFLib [41]:**



RDFLib Est une bibliothèque Open source du langage Python pour travailler avec RDF, elle est simple mais puissant pour représenter une ontologie sous forme de graph, elle permet des opérations de modification de création de suppression sur les ontologies.

[39] https://www.qt.io/qt-features-libraries-apis-tools-and-ide/
[40] https://www.mongodb.com/
[41] https://github.com/RDFLib/rdflib

❖ **WebVOWL** [42]:

WebVOWL est une application Web pour la visualisation interactive des ontologies. Il implémente la notation visuelle pour OWL Ontologies (VOWL) en fournissant des représentations graphiques pour les éléments du Web Ontology Language (OWL) qui sont combinés à une disposition de graphe dirigé par force représentant l'ontologie. Les techniques d'interaction permettent d'explorer l'ontologie et de personnaliser la visualisation. Les visualisations VOWL sont générées automatiquement à partir de fichiers JSON dans lesquels les ontologies doivent être converties. Un convertisseur OWL2VOWL basé sur Java est fourni avec WebVOWL.

❖ **JetBrains PyCharm** [43]:

PyCharm est un environnement de développement intégré (abrégé EDI en français ou en anglais : IDE (Integrated Development Environment)) utilisé pour programmer en Python. Il offre l'analyse de code, un débogueur graphique, la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Il est développé par l'entreprise tchèque JetBrains. Il est multi-plateforme et fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, réalisé sous licence propriétaire, et en édition communautaire réalisé sous licence Apache.

❖ **MSVC build tools 2017** [44]:

Ensemble d'outils pour développer en C et C++, permettant de compiler, déboguer un programme en C++ s'exécutant sur Windows.

[42] http://vowl.visualdataweb.org/webvowl.html
[43] https://www.jetbrains.com/pycharm/
[44] https://www.visualstudio.com

❖ **Draw.io** [45]:

Draw.io est une application gratuite open source de dessin pour workflow, BPM, organigrammes, UML, ER, diagrammes de réseau, Maquette, etc.

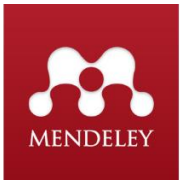
❖ **Pencil** [46]:

Pencil est conçu pour fournir un outil de prototypage GUI gratuit et open source que les utilisateurs peuvent facilement installer et utiliser pour créer des maquettes dans des plates-formes de bureau populaires.

❖ **Microsoft Word** [47]:

Microsoft Word est un logiciel de traitement de texte publié par Microsoft. Nous l'avons utilisé pour rédiger nos documents.

❖ **Mendeley** [48]:



Mendeley est un logiciel de gestion bibliographique, destiné à la gestion et au partage de travaux de recherche. Il est composé d'un logiciel gratuit de bureautique (Windows/Mac/Linux) gérant notamment les PDF, les citations et les références bibliographiques et d'un réseau web.

[45] https://www.draw.io/
[46] https://pencil.evolus.vn/
[47] https://products.office.com/fr/word
[48] https://www.mendeley.com/

Conclusion générale :

L'objectif de ce travail était de résoudre le problème d'alignement d'ontologie, permettant ainsi une interopérabilité sémantique entre différentes parties, ce mémoire présente un cadre basé sur les ontologies et leur alignement. Plus précisément, une nouvelle approche que nous avons développée.

Dans le domaine de l'alignement, l'un des principaux problèmes est le besoin d'algorithmes et d'outils flexibles, capables de s'adapter à différents domaines et aussi à différentes interprétations des notions d'alignement et de similarité. Notre approche mène une variété de techniques basées sur la terminologie, et l'utilisation de ressource externe pour avoir un alignement sémantique.

Par ailleurs la réalisation de ce projet nous a permis de développer de multiples compétences en plus des capacités de documentation de conception et de développement, il nous a fait découvrir un tout nouveau domaine de recherche qui est le domaine de l'ingénierie des ontologies et spécialement le Web sémantique, comme nous avons appris une chose de très important qui est la persévérance, malgré les difficultés rencontrer durant plusieurs étapes, nous avons pu les surmonter et arriver à des résultats satisfaisants.

Références

[1] R. Studer, R. Benjaminsc, and D. Fensela, *Knowledge engineering: Principles and methods* , vol. 25, no. 1–2, March. 1998.

[2] "Publications of the W3C Semantic Web Activity." [Online]. Available: https://www.w3.org/2001/sw/Specs. [Accessed: 27-Sep-2017].

[3] 3 Round Stones Inc. David Wood, "What's New in RDF 1.1," 2014. [Online]. Available: https://www.w3.org/TR/rdf11-new/. [Accessed: 20-Sep-2017].

[4] G. Dan Brickley and G. R.V. Guha, "RDF Schema 1.1," 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/. [Accessed: 27-Apr-2017].

[5] W3C OWL Working Group, "OWL 2 Web Ontology Language Document Overview (Second Edition)," 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/. [Accessed: 28-Apr-2017].

[6] U. of O. Boris Motik, N. C. Peter F. Patel-Schneider, and U. of M. Bijan Parsia, "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)," 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/. [Accessed: 28-Apr-2017].

[7] N. Guarino, "Formal Ontology and Information Systems," in *the International Conference in Formal Ontology and Information Systems - FOIS'98*, 1998, pp. 3–15.

[8] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–136, 1996.

[9] D. L. McGuinness, "Ontologies come of age," *Spinn. Semant. web bringing World Wide Web to its full potential*, pp. 171–192, 2002.

[10] T. Berners-Lee and M. Fischetti, *Weaving the Web : the original design and ultimate destiny of the World Wide Web by its inventor*. HarperSanFrancisco, 1999.

[11] Richard Fikes, "Reusable Ontologies - A Key Enabler for Electronic Commerce." [Online]. Available: http://ksl-web.stanford.edu/Reusable-ontol/. [Accessed: 16-May-2017].

[12] M. Uschold and M. Gruninger, "Ontologies: Principles, Methods and Applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 13–20, 1996.

[13] J. Euzenat and P. Shvaiko, *Ontology matching, 2nd Edition*. 2013.

[14] Michel Klein, "Change Management for Distributed Ontologies," 2004.

[15] J. B. Lovins, "Development of a stemming algorithm. Mech. Transl. Comput. Linguist.11," pp. 22–31, 1968.

[16] M. F. Porter, "An algorithm for suffix stripping. Program .14," pp. 130–137, 1980.

[17] S. Guha∗, R. Rastogi, and K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes."

[18] Jetendr Shamdasani, "Semantic Matching for the Medical Domain," 2012.

[19] "OAEI::Large BioMed Track." [Online]. Available: http://www.cs.ox.ac.uk/isg/projects/SEALS/oaei/. [Accessed: 19-Jul-2017].

[20] J. Euzenat, "A format for ontology alignment." [Online]. Available: http://alignapi.gforge.inria.fr/format.html.

[21] F. A. B. and K. R. Jomar da Silva, "ALIN Results for OAEI 2016," 2016.

[22] and I. F. C. Daniel Faria, Catia Pesquita, Booma S. Balasubramani, Catarina Martins2, Joao Cardoso ˜ , Hugo Curado, Francisco M. Couto, "OAEI 2016 Results of AML," 2011.

[23] M. B. Marko Gulić, Boris Vrdoljak, "CroMatcher - Results for OAEI 2016," 2016.

[24] and J. Maciej Rybinski , Maria del Mar Roldan-Garcia, Jose Garcia-Nieto and F. Aldana-Montes, "DisMatch results for OAEI 2016," 2016.

[25] Muhammad Fahad, "DKP-AOM: results for OAEI 2016," 2016.

[26] M. Zhao and and S. Zhang, "FCA-Map Results for OAEI 2016," 2016.

[27] P. Wang, "Lily Results for OAEI 2016," 2016.

[28] B. Jimenez-Ruiz, E., Cuenca Grau, "LogMap: Logic-based and Scalable Ontology Matching," *Web Conf.*, 2016.

[29] and I. H. E Jimenez-Ruiz, B Cuenca Grau, Y Zhou, "Large-scale Interactive Ontology Matching: Algorithms and Implementation. In ECAI," 2012.

[30] and C. T. Imen Megdiche, Olivier Teste, "LPHOM results for OAEI 2016," 2016.

[31] K. T. Abdel Nasser Tigrine, Zohra Bellahsene, "LYAM++ Results for OAEI 2016," 2016.

[32] J. L. Yan Zhang, Hailong Jin, Liangming Pan, "RiMOM Results for OAEI 2016," 2016.

[33] M. A. B. 3 Abderrahmane Khiat 1 , Elhabib Abdelillah Ouhiba 2 and and C. E. Z. 4, "SimCat Results for OAEI 2016."