N° d'ordre : 18/2021-C/INF

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté d'électronique et d'informatique

Thèse de Doctorat

Présentée pour l'obtention du **grade** de **DOCTEUR.**

En : INFORMATIQUE

Spécialité : Intelligence Artificielle

Par : **BELHADI Hiba**

Sujet :

Application des méthodes intelligentes pour le web des données (linked data)

Soutenue publiquement, le 06/02/2021, devant le jury composé de :

Mr AIT ZAI Abdelhakim	Professeur	à l'USTHB	Président
Mme AKLI-ASTOUATI Karima	Professeur	à l'USTHB	Directrice de thèse
Mr BABA ALI Ahmed Riadh	Professeur	à l'USTHB	Examinateur
Mme BOUSTIA Narhimane	Professeur	à l'Univ.BLIDA	Examinateur
Mr DJENOURI Youcef	Maitre de recherche/A	SINTEF Digital, Oslo, Norway	Invité

A la mémoire de Amira.

Je rends grâce à Dieu, miséricordieux de m'avoir soutenue et donné la volonté, la persévérance et l'obstination pour réaliser ce modeste travail.

Ce travail a été effectué au Laboratoire de **Recherche en Informatique Intelligente, Mathématiques et Applications (RIIMA)** de l'université des sciences et de la technologie Houari Boumediène (**USTHB**).

Ma gratitude va tout particulièrement à ma directrice de thèse Madame **AKLI-ASTOUATI Karima**, Professeur à l'USTHB pour toute la confiance qu'elle m'a accordée dès le début de mes travaux, pour sa patience, sa constante disponibilité, ses orientations, ses conseils avisés, son soutien moral et ses grandes qualités humaines. Je n'aurais pas pu rêver mieux...

Je suis très honorée que Monsieur **AIT ZAI Abdelhakim,** Professeur à l'USTHB ait accepté de présider le jury de cette thèse. Je lui exprime mes plus vifs remerciements.

Je remercie les membres de jury Monsieur **BABA ALI Ahmed Riadh** et Madame **BOUSTIA Narhimane** d'avoir accepté d'évaluer et de juger mon présent travail.

Je tiens aussi à remercier Monsieur **DJENOURI Youcef,** Maitre de recherche à SINTEF Digital, Oslo, Norway, pour sa précieuse aide durant ma formation doctorale. C'est grâce à lui que j'ai pu achever ma thèse.

Je dédie cette thèse à mes chers parents, qui ont toujours été là pour moi.

Je remercie mon fiancé ***Abdellatif***, mes sœurs ***Asma*** et ***Sana*** et mon beau-frère ***Youcef*** pour leur soutien et leurs encouragements.

Sans oublier mon petit-neveu ***Sohayb*** qui a fleuri à sa manière cette dernière année.

Mes remerciements vont à tous ceux qui, par leurs encouragements et leur aide précieuse, m'ont soutenue tout le long de mon travail.

Abstract

This thesis presents our contribution in the field of linked open data by exploiting certain data mining techniques and a meta-heuristic, the genetic algorithm. We have opted for an architecture composed of two (2) processes : « the selection of relevant attributes » and « the alignment process». For the data mining approach, we have developed two systems DMOM (Data Mining for Ontology Matching based instances) and COMI (Clustering for Ontology Matching-based Instances). The first system focuses on an attribute selection process to then trigger a classic alignment process. For the second system, we have directly developed the alignment process. Regarding the genetic algorithm, we focused on the selection of relevant attributes and then the alignment system developed for DMOM was used.

To validate our systems and show their efficiency in comparison with state-of-the-art systems, in-depth experiments were carried out using well-known datasets, those of the OAEI (Ontology Alignment Evaluation Initiative) campaign[1] and that of DBpedia [2]. The experiments show the efficiency of our algorithms compared to existing algorithms.

Key words : data web, linked open data, feature selection, alignment, incomplete data, data mining, meta-heuristics, genetic algorithm, clustering.

[1]http ://oaei.ontologymatching.org
[2]https ://wiki.dbpedia.org

Résumé

Ce travail de thèse présente notre contribution dans le domaine du liage de données en exploitant certaines techniques de fouille de données et une méta-heuristique, l'algorithme génétique. Nous avons opté pour une architecture composée de deux (2) processus : « la sélection des attributs pertinents » et « le processus d'alignement ». Pour l'approche fouille de données, nous avons développé deux systèmes DMOM (Data Mining for Ontology Matching based instances) et COMI (Clustering for Ontology Matching-based Instances). Le premier système se focalise sur un processus de sélection des attributs pour ensuite déclencher un processus d'alignement classique. Pour le second système, nous avons développé directement le processus d'alignement. En ce qui concerne l'algorithme génétique, nous nous sommes concentrés sur la sélection des attributs pertinents puis le système d'alignement développé pour DMOM a été utilisé.

Pour valider nos systèmes et montrer leur efficacité en comparaison avec d'autres systèmes, des expérimentations approfondies ont été menées à l'aide de jeux de données bien connues, ceux de la campagne OAEI (Ontology Alignment Evaluation Initiative)[3] et celui de DBpedia[4]. Les expérimentations montrent l'efficacité de nos algorithmes par rapport aux algorithmes existants.

Mots Clés : web des données, liage des données, sélection des attributs, alignement, données incomplètes, fouille de données, méta-heuristique, algorithme génétique, clustering.

[3]http ://oaei.ontologymatching.org
[4]https ://wiki.dbpedia.org

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

L'évolution rapide dans le domaine de l'ingénierie des connaissances, a encouragé les chercheurs à réaliser des progrès dans ce domaine. Parmi ces progrès, on trouve le web sémantique (WS) qui étend le réseau des hyperliens entre des pages web classiques par un réseau de lien entre données structurées. Ceci permet aux agents automatisés d'accéder plus intelligemment aux différentes sources de données contenues sur le web. La plupart du temps, lorsqu'on prononce le terme de web sémantique, on parle des différentes technologies qui se cachent derrière. Parmi les plus connues, on peut citer RDF (Ressource Description Framework) qui correspond à un modèle standard pour l'échange de données sur le web. Il a ses propres caractéristiques pour faciliter l'intégration des données. La sémantique des données est décrite par des ontologies avec des langages prévus pour fournir une description formelle de concepts, termes ou relations d'un domaine quelconque. L'objectif principal du WS vise donc à aider l'émergence de nouvelles connaissances à partir de celles qui existent déjà. Pour y parvenir, le web sémantique met en œuvre le web des données (Linked Data) qui consiste à lier ces données. Il devient un web des données ouvertes grâce à leur diffusion de manière structurée selon une méthodologie et une licence ouverte garantissant un libre accès et une réutilisations par tous, sans aucune restriction technique, juridique ou financière. Sans vraiment une maîtrise du contenu du web, l'utilisateur passe une grande partie de son temps à examiner un grand nombre de pages web en cherchant ce qui lui convient. Grâce à cette nouvelle vision du web, les ressources web seront non seulement compréhensibles par les humains mais également par des machines. L'objectif donc de ce web des données est la construction d'un réseau d'informations disponibles en ligne et facilement réutilisables dans de nombreux contextes par la création d'un réseau de liens entre données structurées.

Cadre de la thèse et Problématique

L'idée d'un réseau de liens entre données structurées favorise leurs publications non pas sous la forme de bases de données isolées les unes des autres, mais en les reliant entre elles pour constituer un réseau global d'informations. En janvier 2007, il y a eu l'identification de 12 jeux de données dont DBpédia, FOAF, Geoname, WikiData,,,. Ils sont disponibles en RDF, sous des licences ouvertes et ils sont diffusés sur le web. Le nombre de jeux de données et de vocabulaires disponibles a grandi très rapidement, pour atteindre par exemple à la fin de 2010, 203 jeux de données regroupant près de 27 milliards de triplets et près de 400 millions de liens RDF. En mai 2020, l'ensemble des données a atteint 1260 jeux de données avec 16187 liens [8].

Les liens existants sont réalisés manuellement grâce à la primitive owl :sameAs. Mais, avec cette masse grandissante de données, la tâche manuelle devient difficile. L'automatisation du processus de liage de données devient indispensable. Comme on a souvent affaire à des données imprécises et manquantes, périmées, fausses ou soumises à des restrictions d'usage, il est primordial d'en tenir compte. Dans le contexte de cette thèse, nous nous intéressons à des données manquantes, n'ayant pas forcément la même structure. C'est donc au problème d'hétérogénéité des données [Heflin and Song, 2016]) et à leur liage [Nentwig et al., 2017] que nous traitons dans le cadre de notre présente étude. Il s'agit de trouver des entités équivalentes, déclarées dans des jeux de données différents.

Apport de la thèse

A travers la lecture des différents travaux sur le liage de données, nous avons adopté un processus en deux phases pour réaliser nos systèmes de liage de données :

- La phase de sélection des attributs pertinents.
- La phase d'alignement des données.

Des techniques intelligentes ont été exploitées par l'application de l'algorithme génétique et certaines techniques de fouille de données. Nous résumons nos différentes contributions dans ce qui suit :

- Nous nous sommes intéressées d'abord au processus de sélection des attributs pertinents. Nous avons testé et modélisé d'une part une méta-heuristique, l'algorithme génétique [Belhadi et al., 2018] et d'autre part une technique de fouille de

[8] https ://lod-cloud.net

données, les item set fréquents (FIM) [Belhadi et al., 2020,Belhadi et al., 2019]. Dans la 2ème approche, trois techniques ont été implémentées. La première explore l'arborescence de recherche d'une manière aléatoire. La deuxième approche utilise certaines valeurs statistiques pour sélectionner les propriétés les plus pertinentes. Et la troisième appelée FIM (Frequent Itemsets Mining), explore la corrélation entre les différentes propriétés et sélectionne les propriétés les plus fréquentes décrivant les instances globales qu'on peut assimiler à un jeu de données.

- Pour les deux systèmes développés, le processus d'alignement est identique.
- Proposition d'une approche qui exploite une technique de fouille de données, le clustering pour le liage de données. Il s'agit de regrouper les instances hautement corrélées dans des clusters à l'aide de l'algorithme K-means puis lier les clusters entre eux.

Organisation du document

Notre document, organisé en deux parties (état de l'art et contributions), inclut six chapitres. Dans la première partie, le premier chapitre englobe les notions essentielles pour aborder "le web des données". Ensuite, les algorithmes évolutifs, conduisant aux méta-heuristiques, plus précisément l'algorithme génétique fait l'objet de présentation dans le chapitre 2. Enfin, dans le chapitre 3, seront présentées certaines techniques de fouille de données, pour choisir les items sets frequents et le clustering comme techniques d'implémentation. La 2ème partie est consacrée à nos contributions. Trois systèmes ont été développés. Le premier basé sur l'algorithme génétique dans la phase de sélection des attributs pertinents est présenté dans le chapitre 4. Le chapitre 5 développe une technique de fouille de données pour sélectionner les meilleurs attributs, les items set fréquents. Le 3ème système est développé pour réaliser tout le processus de liage de données en utilisant le clustering.

Partie I : Etat d'art

- **Chapitre 1** Nous donnons un aperçu sur les aspects fondamentaux du web des données avec un aperçu sur ses principes, son architecture et ses langages (RDF, RDF(S), OWL). Quelques systèmes de liage des données avec leurs différentes composantes sont présentés.

- **Chapitre 2** Nous nous intéressons à certaines heuristiques et méta-heuristiques afin de maîtriser ce domaine. Des algorithmes comme A*, les recherches aveugles, les algorithmes génétiques, ... sont présentés indépendamment du problème qui nous intéresse. Puis, nous avons fait le lien avec notre problématique pour trouver par la suite des solutions alternatives.

- **Chapitre 3** Cette partie est dédiée aux différentes techniques d'exploration de données qui ont été étudiées pour traiter et analyser plusieurs types de modèles de données. Nous nous sommes focalisées sur les tâches d'exploration de données les plus populaires. Il s'agit de la classification, la synthèse, l'exploration de règles d'association et le clustering. Comme dans le chapitre précédent, le travail a été fait indépendamment du problème de liage de données. Ensuite, pour aborder notre problématique, un paragraphe lui a été consacré.

Partie II : Contributions

- **Chapitre 4** Dans cette partie, nous présentons les principaux composants du système proposé appelé GFSOM (Genetic Feature Selection for Ontology Matching) qui est une approche de sélection des attributs. Elle est basée sur l'approche génétique pour extraire les attributs les plus pertinents afin de leur appliquer ensuite le processus d'alignement des données . Des expérimentations de GFSOM ont été menées sur le jeu de données DBpédia et certains systèmes de la littérature.

- **Chapitre 5** Cette partie représente le système appelé DMOM (Data Mining for Ontology Matching based instances). Une approche basée sur des techniques de fouille de données, les items sets fréquents (FIM) a été utilisée. Des expérimentations de DMOM ont été menées sur des jeux de données de la campagne d'évaluation OAEI (Ontology Alignment Evaluation Initiative), et DBpedia pour évaluer notre système en termes de temps d'exécution et de précision.

- **Chapitre 6** Ce chapitre est consacré au système COMI (Clustering for Ontology Matching-based Instances). C'est un système de liage de données basé sur une des techniques de fouille de données, le clustering. COMI regroupe d'abord les instances hautement corrélées de chaque ontologie (jeu de données) dans des clusters similaires à l'aide de l'algorithme k-means. Pour démontrer la précision de COMI, plusieurs expérimentations sur le jeux de données DBpedia et ceux de la campagne OAEI ont été menées et présentées dans ce chapitre.

Conclusion Générale : A la fin de ce mémoire, nous dresserons un bilan général de ce présent travail et nous abordons quelques perspectives.

Première partie

État de l'art

Chapitre 1

Aperçu du problème de liage des données

1.1 Introduction

Sur le Web, avec la prolifération des jeux de données, un nouveau challenge est apparu. Revoir la vision du Web pour penser à lier des données plutôt que des documents. Donc, l'idée est d'étendre le réseau d'hyperliens entre des pages Web classiques par un réseau de liens entre données structurées, permettant ainsi d'accéder plus intelligemment aux différentes sources de données contenues sur le Web. De cette façon, des tâches (l'apprentissage, la recherche,..) sont effectuées d'une manière plus précise pour les utilisateurs.

Cette nouvelle technologie appelée Web sémantique (WS) ou plus récemment web des données fait référence à une base de données à échelle mondiale où les communications homme - machines, machine-machine, sont possibles grâce aux ontologies. Les données dans ce cas sont formalisées en logique et elles sont représentées sous forme de triplets RDF (Resource Description Framework).

Suite aux recommandations du W3C [1] un grand nombre de triplets RDF sont disponibles sur le web. On parle alors de données liées ouvertes (LOD) du fait que les données soient liées entre elles. D'une manière générale ce liage est réalisé manuellement grâce à une primitive logique. Mais, par rapport à la masse de données existante 1260 jeux de données, avec 16187 liens (en mai 2020)[2], il est primordial d'automatiser ce processus qui trouve la même entité déclarée dans différents jeux de données. Plusieurs travaux ont vu le jour, comme synthèse de travaux [Nentwig et al., 2017, Heflin and Song, 2016, Shvaiko and Euzenat, 2013, Otero-Cerdeira et al., 2015, Abubakar

[1] https ://www.w3.org
[2] https ://lod-cloud.net

et al., 2018], ou comme analyse et examen de systèmes de liage [Mohammadi et al., 2019, Mohammadi et al., 2018]. Afin de mieux appréhender notre problématique de recherche, nous allons présenter dans ce chapitre les notions essentielles liées au LOD.

1.2 Les technologies du web

Cette section présente le web sémantique et ses différents concepts : RDF et ontologie. Le reste de la section est réservé pour les données liées (Linked Open Data) et l'alignement des données.

1.2.1 Web sémantique

Web Sémantique est une expression proposé par **Tim Berners-Lee** [Berners-Lee et al., 2001]. Cette proposition fait référence à la vision du Web du futur comme une extension du Web actuel supportant des fonctions avancées pour la collaboration (homme-homme, homme-machine, machine-machine) en vue de partager et de raisonner sur le contenu de grands volumes d'informations.

Le principe du Web sémantique consiste à attacher des annotations à tous types de ressources disponibles (documents, pages Web, services...) sur le Web en vue de rendre explicitement leur contenu sémantique. L'interprétation des annotations, donc la sémantique, est précisée entre des agents logiciels, des machines ou voire des gens grâce à un des éléments les plus importants du Web sémantique : l'ontologie. Le principe des ontologies consiste à définir une interprétation commune d'une partie du monde réel, et de modéliser les concepts et les relations entre ces concepts (pour plus de détails voir section 1.2.5).

1.2.2 Architecture du web sémantique

Une architecture représentative a été mise en oeuvre lors de la conférence internationale du WWW (fig 1.1), nomé aussi : cake layer. C'est Tim Berners-Lee qui a expliqué le besoin d'ajouter une couche sémantique au web actuel. Elle se compose de trois niveaux : adressage, syntaxique et sémantique.

- **Niveau adressage** : Il représente le plus bas niveau dans l'architecture et se compose de deux notions. La première représente l'Unicode qui est un encodage universel afin d'échanger des symboles. La deuxième est URI (Uniform Resource Identifier) qui caractérise d'une manière unique et non ambigue une ressource. URI fournit aussi un adressage standard universel, on la considère comme étant

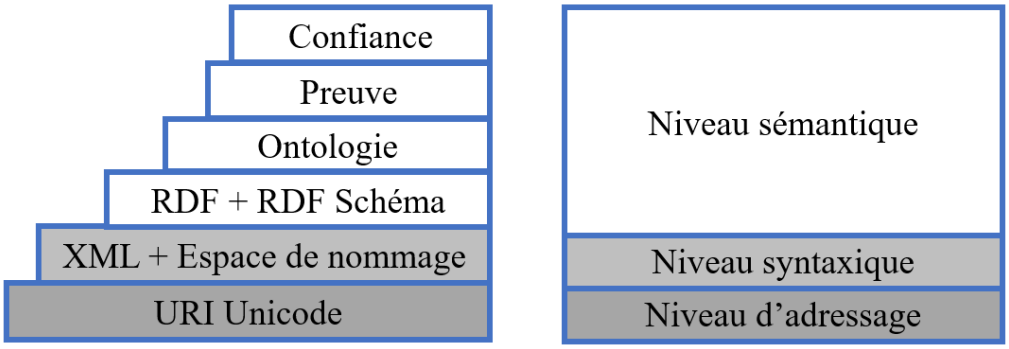


FIGURE 1.1 : Architecture du web sémantique [Berners-Lee et al., 2001].

un URL (Uniform Ressource Locator) dans l'actuel web. Sauf que les URL sont utilisées pour des ressources localisées et les URI peuvent représenter des ressources abstraites (à titre d'exemple des personnes).

- **Niveau syntaxique** : Ce niveau se compose d' XML (eXtensible Markup Language) celui qui mène à la création de vocabulaire à travers un ensemble de règles. Il structure les documents et les données publiés sur le web. Ainsi, il inclut l'espace de nommage (namespace) pour l'identification des différentes balises utilisées dans le document.

- **Niveau Sémantique** : Afin de donner une organisation plus structurée des informations présentes sur le Web à travers une description sémantique des données fournies par XML, RDF (Resource Description Framework) est utilisé. C'est un standard permettant la mise en place des annotations. Il permet d'affirmer des relations entre les ressources, et de représenter l'information sous forme de triplets (sujet, prédicat, objet), dont les éléments peuvent être des URIs, des variables ou des littéraux. L'information représentée par RDF est conservée principalement sous forme de déclarations RDF.

  RDF Schéma permet ensuite de décrire les hiérarchies de concepts et les relations entre les concepts.

  La couche Ontologie décrit des ressources d'information de différentes natures communes d'un domaine et partagée par plusieurs personnes, sociétés, etc.

  La couche Preuve a pour but de prouver la pertinence des informations des niveaux plus bas.

Tandis que la couche Confiance est dédiée à la fiabilité des informations et des raisonnements. Elle se base sur des principes numériques et le cryptage des sources d'information (agents de confiances. . .)

1.2.3 RDF : Resource Description Framework

RDF [Lassila, 1999] est un modèle de données W3C qui permet l'expression des données d'une manière standardisée. Grâce à RDF, les informations sur le Web peuvent être transformées d'un format non structuré en données structurées. Une ressource est tout ce qui peut être référencé à une information, il peut s'agir d'une page Web, d'un document XML, ou d'un service Web. Une ressource RDF est décrite par des triplets. Un triple se compose d'un sujet, d'un prédicat et d'un objet **(sujet, prédicat,objet)**.

- **Ressources (Sujet) :** désigne tout élément qui peut être référencé par une URI (Uniform Resource Identifier). Une ressource peut être un livre, une image, une page web, un événement ou une personne.
- **Propriétés (Prédicat) :** désigne un attribut (ou une caractéristique) utilisé pour décrire la ressource.
- **Valeurs (Objet) :** peut être soit une simple chaîne de caractères, soit une ressource qui peut être une URI.

Un *Sujet* peut devenir un *Objet* dans un autre triplet. Ce qui fait qu'une description RDF correspond à un graphe orienté étiqueté, dont le nœud source est le sujet, le nœud cible est l'objet et l'arc orienté est le prédicat. Considérons l'exemple dans la figure 1.2 la représentation graphique de cette phrase. Omar est le créateur de la page "http ://www.w3.org/Home/Omar". Le sujet c'est le site de la page. Le prédicat correspond au verbe "créer par", et l'objet est "Omar".

FIGURE 1.2 : Exemple d'un graph RDF.

Le langage RDF permet de définir formellement des métadonnées pour une large catégorie de données. A travers sa syntaxe les applications peuvent comprendre une grande partie de la traduction des données. RDF Schema (RDFS) est une extension de RDF dont le but est d'enrichir les descriptions de ressources dans les graphes. Un schéma RDF permet de déclarer des contraintes sémantiques entre les classes et les propriétés utilisées dans ces graphes.

1.2.4 RDF Shéma

RDF a été rapidement complété par RDF Schéma (RDFs) qui fournit des prédicats essentiels pour représenter (en RDF) une ontologie. RDFS a pour but d'étendre le RDF en décrivant plus précisément les ressources utilisées pour étiqueter le graphe. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les ressources sont des instances où des propriétés. RDFS s'écrit toujours à l'aide de triplets RDF. Quelque constructeurs sont les suivants.

- **rdfs :class** qui permet de désigner une classe.
- **rdfs :subClassOf** est une propriété transitive restreinte à la classe "rdfs :Class". Une classe peut être une sous-classe de plusieurs classes.
- **rdfs :property** qui permet de spécifier des propriétés d'une classe.
- **rdfs :subPropertyOf** est une instance de "rdf :Property". Elle est employée pour spécifier qu'une propriété est une spécialisation d'une autre.
- **rdf :type** qui permet de désigner un typer pour ressource à une propriété ou une classe.
- **rdfs :domain** qui permet de spécifier le domaine d'une propriété, pour indiquer aux classes sur quels membres une propriété peut être utilisée.
- **rdfs :range** elle est utilisée pour indiquer à quelles classes les valeurs d'une propriété appartiennent.
- **rdfs :label** elle permet d'ajouter un nom à une ressource.

RDFS ajoute à RDF les possibilités de définir des hiérarchies de classes et de propriétés dont l'applicabilité et le domaine de valeurs peuvent être contraintes à l'aide des attributs "rdfs :domain" et "rdfs :range". A chaque domaine applicatif, on peut associer un schéma identifié par un préfixe particulier et correspondant à une

URI. Les ressources instances sont ensuite décrites en utilisant le vocabulaire donné par les classes définies dans ce schéma. Les applications peuvent alors leur donner une interprétation opérationnelle.

1.2.5 Les ontologies

Nous examinons dans cette section la composante essentielle du Web sémantique : l'ontologie. Les ontologies sont la technologie dorsale pour le Web sémantique.

Une ontologie est une définition formelle des concepts, propriétés et relations des entités qui existent dans un certain domaine. Une ontologie fournit un vocabulaire partagé qui peut être utilisé pour décrire le domaine, classer et catégoriser les éléments qu'il contient. En 1993, T. Gruber [Gruber, 1993] a proposé une définition qui reste jusqu'à présent la définition la plus citée dans les écrits en intelligence artificielle : une ontologie est une spécification d'une conceptualisation d'un domaine de connaissances. La notion de conceptualisation nous la définissons comme l'identification par des termes et/ou des symboles, des concepts du domaine et des relations existantes entre ces concepts. Mais cette conceptualisation est forcément basée sur une vision partielle des connaissances d'un domaine.

Les ontologies ont plusieurs classifications, dont chaque chercheur ou communauté développe une selon une approche bien précise telle que la représentation des connaissances, et le domaine d'ontologie. En 1998 Guarino [Guarino, 1998] a proposé une classification des ontologies (voir figure 1.3) selon leurs niveaux de dépendance à une tâche particulière à savoir :

- **Les ontologies de haut niveau :** Décrivent les concepts les plus généraux, valables dans plusieurs domaines tel que le temps, évènements, actions et l'espace.

- **Les ontologies de domaine et de tâches :** Fournissent respectivement, le vocabulaire lié à un domaine précis ou les tâches génériques du domaine. Ces types d'ontologies spécialisent les concepts introduits dans les ontologies de haut niveau.

- **Les ontologies d'applications :** Ce sont des ontologies qui représentent les concepts d'un domaine précis ainsi qu'une tâche particulière.

Le W3C a recommandé OWL (Web Ontology Language) comme un langage standard pour la représentation des ontologies, puisqu'il permet de mieux exprimer une

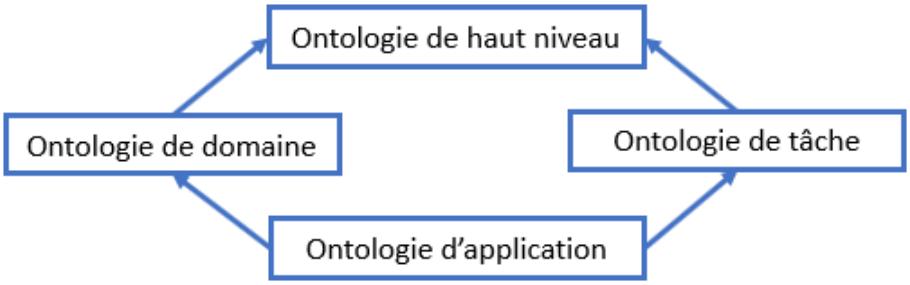


FIGURE 1.3 : Relation entre les différents types d'ontologies [Guarino, 1998].

description sémantique que celle d'XML et RDFs. Cela se fait par l'ajout d'un vocabulaire plus riche en termes de description de concepts et de relations. La dernière version est dénotée "OWL2" [3]. La syntaxe de cette version est beaucoup plus proche du graphe RDF que celle d'OWL1.

Une ontologie peut être représentée par un tuple O = (C, R, A, $\mathcal{R}$) où :

- **C** représente l'ensemble des classes (concepts). Un concept représente un objet ou une notion du domaine.
- **R** est un ensemble de relations entre les différents concepts d'une ontologie. Il existe plusieurs relations entre concepts comme la relation de subsomption "is-a" ou "est-un", qui indique qu'un concept est en hiérarchie avec un autre et que les propriétés d'un concept sont inclues dans les propriétés du deuxième concept.
- **A** l'ensemble des axiomes tels que les relations entre classes ou propriétés, les relations de disjonction entre classes.
- $\mathcal{R}$ l'ensemble de règles d'inférences des connaissances, une règle est une contrainte explicite sur les différents comportements des individus ou instances de l'ontologie.

En pratique, de nombreuses ontologies représentent un mélange des niveaux de généralité proposés par Guarino, attribuant à des inadéquations qui rendent la tâche d'alignement des ontologies difficile.

[3]https ://www.w3.org/TR/owl2-overview/

1.3 Les données liées ouvertes (Linked Open Data)

Les données sont accessibles et publiques pour la réutilisation, redistribution, reproduction et sans aucune restriction. Elles sont considérées comme ouvertes. Vu la grande quantité de données disponibles sur le Web (en mai 2020, l'ensemble de données a atteint 1260 jeux de données avec 16187 liens [4]) (voir fig 1.4 et fig1.5), de nombreux liens entre les ensembles de données existent. Puisque les recherches se sont intéressées aux approches qui génèrent automatiquement des liens entre les données RDF. Dans le Web sémantique, la tâche de génération de liens est généralement appelée liage de données, liage d'instances, ou interconnexion de références. La liaison de données peut être considérée comme une opération qui prend en entrée deux ensembles de données, chacun contenant un ensemble de descriptions d'instances et donne en sortie des liens entre les instances (c'est-à-dire des liens owl :sameAs).

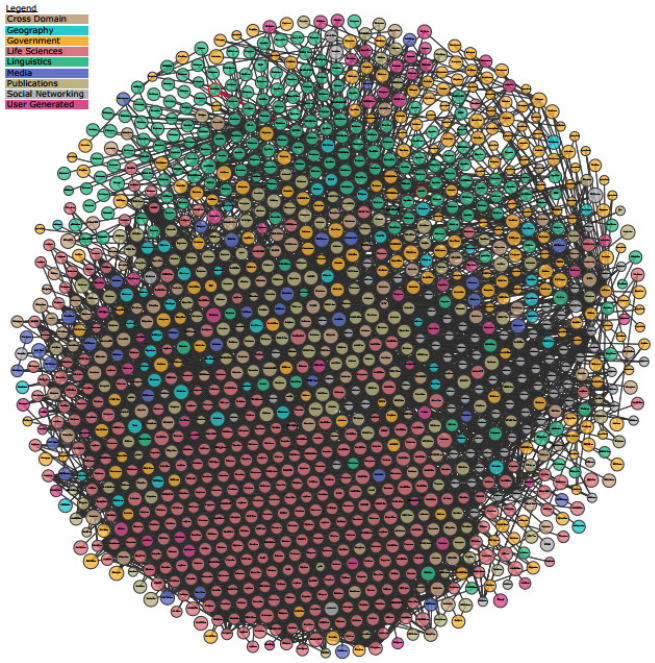


FIGURE 1.4 : Diagramme de nuage de données liées ouverte. [5]

Le concept de données ouvertes est apparu dans les années 2000 et s'est répandu dans les domaines scientifiques, gouvernementaux et de la société civile.

Berners-Lee [Bizer et al., 2011] a décrit un ensemble de règles pour publier des données sur le Web de manière structurée qui font partie d'un seul espace de données mondial :

[4]https ://lod-cloud.net
[5]https ://lod-cloud.net/clouds/lod-cloud.svg

1. Utilisez des URI comme identifiant pour les objets.

2. Utilisez les URI HTTP pour que les utilisateurs puissent rechercher ces identifiants.

3. Lorsque quelqu'un recherche une URI, fournissez des informations utiles, en utilisant les normes (RDF, SPARQL).

4. Incluez des liens vers d'autres URI, afin qu'ils puissent découvrir plus de choses.

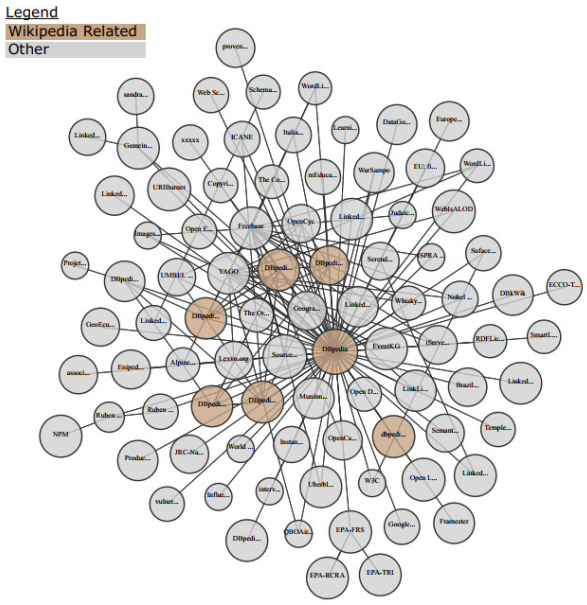


FIGURE 1.5 : Les domaines du cloud de données ouvertes interconnectés. [6]

Cependant, en raison du nombre croissant des données ouvertes sur le Web, elles ne peuvent pas être traitées manuellement. De là vient un réseau ouvert, permettant d'accéder aux données à interroger automatiquement quel que soit l'endroit où elles sont stockées.

Le Web ouvert de données contient des informations exprimées avec des formats standard lisibles par la machine. Il dénote non seulement les données ouvertes mais également leur mise en relation pour constituer un réseau global liées, afin qu'à partir d'une information donnée, nous puissions accéder plus facilement et plus rapidement aux autres informations. Le processus de liaison des entités entre elles est appelé liage des données.

[6] https ://lod-cloud.net/clouds/cross-domain-lod.svg

1.3.1 Concepts de base liés au domaine de l'appariement

Cette partie est dédiée à l'explication du liage des données avec les notions de base de la thèse.

Étant donné que les ensembles de données sont créés indépendamment par les utilisateurs, il peut y avoir plusieurs URI indiquant la même information (instances) dans différents jeux de données. Par conséquence, nous avons besoin de résoudre le problème de liage, afin d'identifier et relier la même instance à travers plusieurs jeux de données.

Différents types de disparités ou d'hétérogénéités rendent le processus d'alignement difficile. De nombreux algorithmes d'appariement appliqués sont assez naïfs (par exemple, les algorithmes d'appariement de chaînes). Il existe différentes classifications de ces disparités, à différents niveaux et avec un chevauchement. Une classification est tirée de Klein [Klein, 2001], est illustrée sur la figure 1.6.

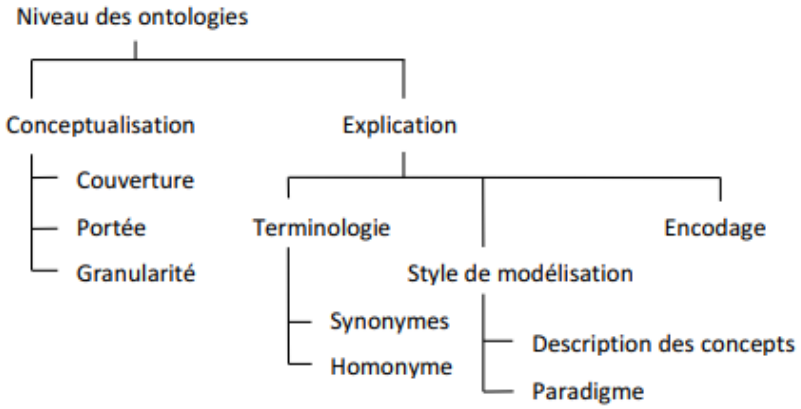


FIGURE 1.6 : Disparité des ontologies [Klein, 2001].

Les disparités de conceptualisation : ce type de disparités ne peut pas être résolu automatiquement. Il exige les décisions et la connaissance d'un expert du domaine. Il distingue trois types de disparités :

1. **La portée**, surgit lorsque deux classes semblent représenter le même concept, sans posséder les mêmes instances.

2. **La couverture**, les ontologies diffèrent aussi souvent par leur couverture d'un domaine particulier,

3. Ainsi que, **la granularité**, une ontologie de fine granularité est une ontologie très détaillée, possédant ainsi un vocabulaire riche capable d'assurer une description détaillée des concepts d'un domaine.

Les disparités d'explication : sont de trois types :

1. **Style de modélisation**, les ontologies diffèrent dans leurs styles de modélisation sur les paradigmes et les descriptions des concepts qu'elles utilisent.

2. **les disparités terminologiques** sont des disparités liées au processus de nommage qui associe un objet linguistique aux entités décrites dans une ontologie.

3. **L'encodage des données** au sein des ontologies diffère souvent, que ce soit pour les dates, les unités (monnaie, distances, ...), etc. Dans la plupart des cas, une étape de transformation est suffisante pour éliminer ces disparités.

1.3.2 Problématique

Considérons un graphe RDFs d'un jeux de données **G** . Une instance est représentée par des triplets **(s,p,o)** (sujet, prédicat, objet) avec $s \in U$, $p \in U$ et $o \in U \cup L$. Avec U est l'ensemble des URI et L ensemble des littéraux.

Definition 1.3.1. Prédicat
Un prédicat p est appelé propriété dans G lorsque dans le triplet (s, p, o) l'objet est un littéral ie $o \in L$.

Definition 1.3.2. Littéral
Un littéral est une valeur de type chaîne de caractère ou entier d'une propriété.

Considérons un jeu de données $\mathcal{D}$, représenté par un ensemble d'instances $\mathcal{I} = \{i_1, ..., i_k\}$. En plus, chaque instance est représentée par un ensemble de propriétés $\mathcal{P} = \{p_1, ..., p_n\}$ qui peuvent avoir une ou plusieurs valeurs. Dans nos travaux nous nous intéressons aux propriétés de type prédicat déjà définie.

Un exemple illustratif de liage de données est présenté dans la figure 1.7. On considère deux jeux de données $\mathcal{D}_1$ et $\mathcal{D}_2$. Dans le processus de liage, la première étape est de transformer chaque ensemble des données sous forme tabulaire, ou l'élément $[i, j]$ représente le littéral du $j^{ème}$ propriété de la $i^{ème}$ instance. Ensuite, le processus de liage entre les deux jeux de données est construit et son résultat nommé alignement.

Definition 1.3.3. Liage de données
soient $\mathcal{D}_1$ et $\mathcal{D}_2$ deux jeux de données qui sont représentés par un ensemble de k_1 d'instances $\mathcal{I}^1 = \{i_1^1 ... i_{k_1}^1\}$, et un ensemble $\mathcal{P}^1 = \{p_1^1 ... p_{n_1}^1\}$ de n_1 propriétés pour $\mathcal{D}_1$, et un ensemble de k_2 d'instances $\mathcal{I}^2 = \{i_1^2 ... i_{k_2}^2\}$, et un ensemble $\mathcal{P}^2 = \{p_1^2 ... p_{n_2}^2\}$ de

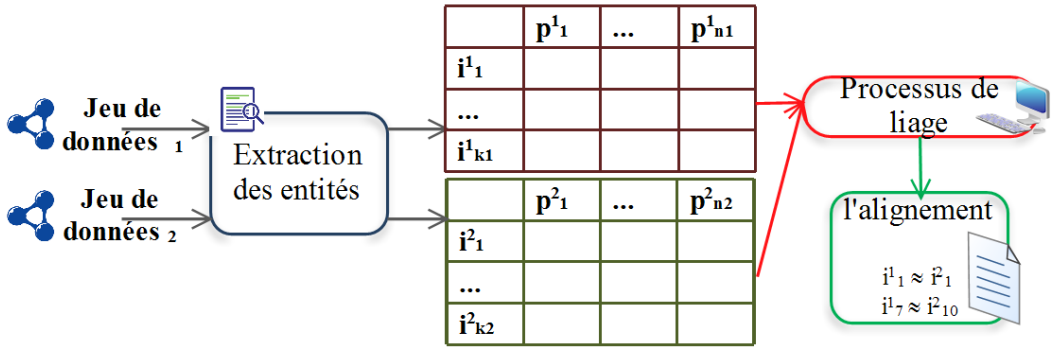


FIGURE 1.7 : Exemple illustratif.

n_2 propriétés pour $\mathcal{D}_2$. Le liage des données a pour but de calculer la fonction $\mathcal{M}$ qui détermine les instances en commun entre $\mathcal{D}_1$ et $\mathcal{D}_2$ comme suit :

$$\mathcal{M}(\mathcal{D}^1, \mathcal{D}^2) = \bigcup_{l \leq k_1, \wedge s \leq k_2} \mathcal{M}_{ls}(i_l^1, i_s^2) \tag{1.1}$$

où

$$\mathcal{M}_{ls}(i_l^1, i_s^2) = \{p | p \in i_l^1 \wedge p \in i_s^2\} \tag{1.2}$$

L'appariement des ontologies est le processus qui compare deux ontologies et découvre les relations ou de correspondances entre leurs entités [Euzenat et al., 2008].

Le résultat du processus d'appariement est appelé alignement. Ce dernier exprime des correspondances entre les entités appartenant à deux ontologies différentes. Une correspondance doit considérer les entités correspondantes ainsi que la relation supposée exister entre elles. Sachant que les entités d'une ontologie peuvent être les concepts, les relations, les fonctions, les instances, les axiomes, les règles, etc.

Dans ce qui suit, nous utilisons l'alignement pour désirer l'appariement des jeux de données, donc les entités sont des instances.

Le nom d'une propriété peut être différent dans chaque jeux de données. Cela rend le liage plus difficile. Ainsi, toutes les valeurs de l'instance du premier jeu de données doivent être comparés avec toutes celles d'une entité du deuxième jeu de données. A travers cela, on constate que l'objectif de liages des données est de trouver les entités en commun entre deux jeux de données, définies différemment.

Dans le processus de liage, il existe plusieurs techniques : celle qui essayent de minimiser le nombre de propriété, ou bien de minimiser la comparaison 1 :1 entre les instances en les regroupant. L'idée dans ce travail est de sélectionner les prédicats les plus pertinents avant le processus de liage dans une partie, et en deuxième lieux nous testerons le regroupement des instances.

Definition 1.3.4. Sélection de propriétés
Considérant un jeu de données $\mathcal{D}$ et $\mathcal{P}$ l'ensemble de tout les prédicats, Sp est l'ensemble des prédicats sélectionnés comme suit :

$$Sp = \{p | freq(p) > \mu \wedge p \in \mathcal{P}\} \tag{1.3}$$

ou μ est le seuil de degré d'intérêt, et $freq(p)$ est la fonction qui détermine comment le prédicat est sélectionné.

1.3.3 Problème du liage des données

La publication des données sur le web d'une manière non organisée a engendré la multiplication de milliers d'ensembles de données. Une première idée était de créer des ontologies spécifiques pour chaque domaine. Mais en raison de décentralisation, de nombreuses ontologies ont apparu pour un même domaine. Les informations publier dans le web peuvent être incomplètes, imprécises et souvent même des information erroné. La liaison de données essaye d'améliorer la qualité du Web, facilitant l'accès à des informations distribuées et décentralisées. Le liage des données est défini comme le processus d'établissement d'une relation entre deux informations provenant de deux jeux de données distincts.

Nous distinguons l'appariement des ontologies. Cela signifie la recherche de concepts ainsi que de propriétés en commun entre deux schémas structurant les données. La figure suivante (Fig : 1.8) montre deux ontologies à apparier ainsi que les différentes correspondances entre leurs entités. sachant que la relation et le degré de confiance par défaut sont respectivement l'équivalence (=) et le 1 ; autrement, ils sont mentionnés.

Mais ,vu l'ensemble de données qui a atteint 1260 jeux de données avec 16187 liens (en mai 2020) [7], le liage des ontologies n'est pas suffisant et de nouveaux algorithmes doivent être développés spécialement pour identifier des instances similaires.

En revanche une des visions prometteuses du Web sémantique est la possibilité de partager une très grande quantité de données sémantiques dans un contexte ouvert. En conséquence, le Web des données, et en particulier le projet Linked Open Data (LOD), a gagné en popularité ces dernières années, avec des centaines d'ensembles de données publiés sur le Web. Étant donné que les données sont publiées de manière décentralisée, cela pose de nouveaux défis pour découvrir les liens identiques dans le contexte ouvert du Web.

[7] https ://lod-cloud.net

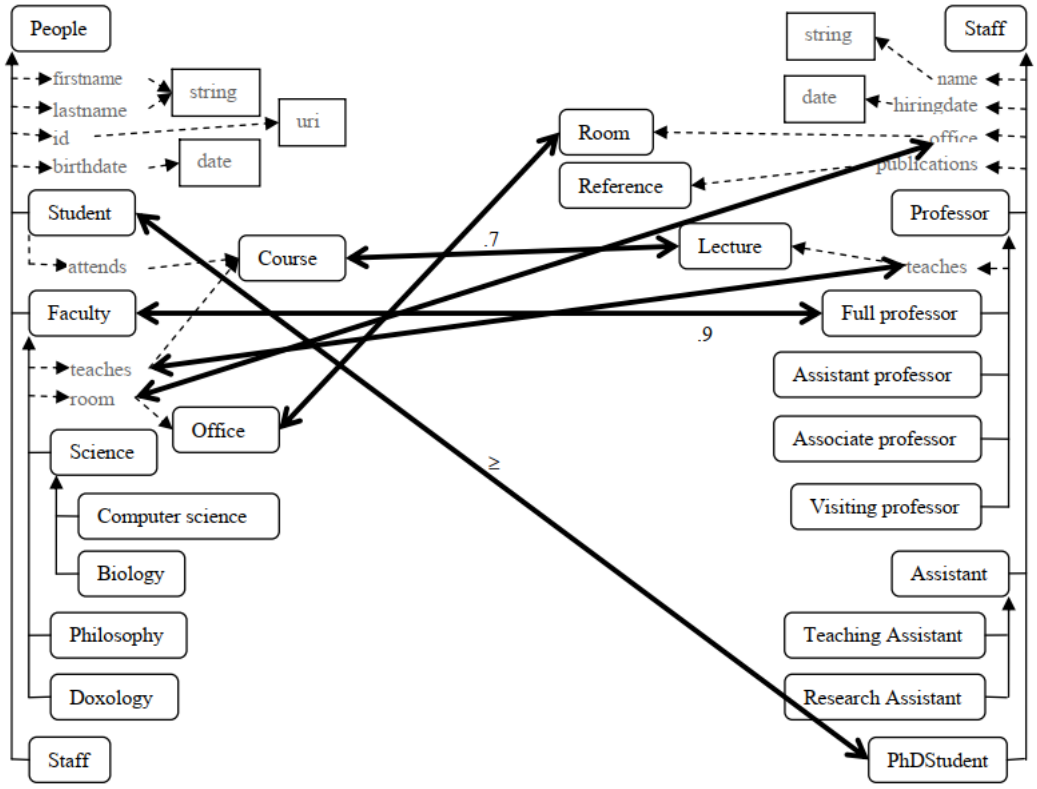


FIGURE 1.8 : Exemple d'alignement entre deux ontologies [Euzenat et al., 2007].

1.4 Etat de l'art

La recherche de la même instance à partir de différents jeux de données peut être traitée sous différents noms tels que le couplage d'enregistrements [Fellegi and Sunter, 1969], la duplication [Sitas and Kapidakis, 2008], la correspondance de noms [Bilenko et al., 2003] ou l'identification d'instance et la correspondance d'enregistrement [Hernández and Stolfo, 1995].

Le problème de découvrir les correspondances entre les entités dans plusieurs jeux de données remonte aux années 1960. Il est appliqué dans divers domaines tels que les données biomédicales [Smith et al., 2007], l'apprentissage [Cerón-Figueroa et al., 2017] et le traitement de langage naturel [Iwata et al., 2017].

En effet, diverses solutions d'alignement ont été proposées et des états de l'art ont été faits [Scharffe et al., 2011,Nentwig et al., 2017], et des livres ont été publiés [Christen, 2012, Euzenat et al., 2007] . Dans ce qui suit, on présente quelques algorithmes qui ont marqué leur performance dans le problème de liage des données.

Le problème du liage a été d'abord étudié par la communauté des bases de données comme les problèmes de duplication ou de couplage d'enregistrements. Dans ce cadre

se situe le travail de [Elmagarmid et al., 2007] qui est basé sur différentes techniques pour résoudre les problèmes d'hétérogénéité dans les ontologies. Il a suggéré une approche pour traiter un ensemble de données d'enregistrement segmentées en propriétés structurées.

Ensuite, il y a eu Rosaci [Rosaci, 2007] qui montre que la correspondance d'ontologies peut être utilisée pour établir une connexion entre différents agents intelligents. CILIO *(Connectionist Learning and Inter-Ontology Similarities)* exploite un modèle d'ontologie capable de représenter des concepts, des collections de concepts, des fonctions et des implications causales parmi des événements dans un environnement multi-agents. Les connaissances d'un agent simulent son comportement, et chaque agent de la communauté doit connaître la similitude entre lui et un autre agent pour guider une négociation si un agent fait une offre. Le même auteur [Rosaci, 2015] a proposé l'utilisation d'un modèle hiérarchique pour trouver des associations sémantiques entre les données du Web. Les relations sémantiques représentées par les métadonnées sont explorées dans le contexte d'un groupe d'entités Web. L'approche prouve son utilité dans des systèmes de recommandation.

Par ailleurs, Li et al. [Wang et al., 2013] ont proposé une approche qui repose sur l'hypothèse que deux entités sont liées si elles sont liées à des entités précédemment déjà liées. Les auteurs expliquent qu'une méthode d'agrégation est basée sur le vote pour combiner les résultats d'alignement de plusieurs stratégies. L'approche se compose de deux étapes : i) d'abord utiliser les informations lexicales pour identifier des correspondances sémantiques ; ii) ensuite, identifier des résultats d'alignement supplémentaires en fonction des informations structurelles et des correspondances précédemment trouvées grâce à une stratégie de liage structurel. Selon les résultats, cette approche a permis d'obtenir une bonne précision pour des jeux petits ou moyens dans le concours OAEI 2010.

L'idée d'Alam et al. [Alam et al., 2017] est de proposer un outil de correspondance des graphes de connaissances (à partir d'un texte). L'outil appelé MERGILO a pu améliorer les résultats dans OAEI par rapport aux approches de base, en utilisant la similarité entre graphe et similarité entre mot. Afin de réunir une représentation sémantique complète d'un évènement, MERGILO utilise un lecteur de machine Web sémantique $FRED$[8], conjointement avec $Framester$, un hub de données liées linguistiques représentées à l'aide d'une nouvelle sémantique formelle pour les trames.

[8]un outil qui génère automatiquement des ontologies RDF / OWL et des données liées à partir de texte en langage naturel multilingue. http ://wit.istc.cnr.it/stlab-tools/fred/

Framester est utilisé pour améliorer la connaissance des événements extraits avec des trames sémantiques (semantic frames).

Hu et al. [Shao et al., 2016] ont implémenté RiMOM-IM l'un des systèmes qui a participé dans les campagnes d'évaluation OAEI 2013 et OAEI 2016 [9]. C'est un algorithme itératif basé sur la stratégie de blocage pour réduire l'ensemble des instances candidates.

Wang et al. [Li et al., 2013] ont développé l'approche nommé VMI qui construit pour chaque instance deux vecteurs. Le premier, *s'appelle vecteur de Nom* contient les prédicats noms et étiquetages (labels), le deuxième contient les prédicats de description et différentes informations des instances voisines, *nommé vecteur de Document Virtuel.* VMI construit des index inversé pour ces types de vecteurs et sélectionne les candidats de liage en fonction des index.

SiGMa [Lacoste-Julien et al., 2013] est un algorithme de propagation itératif. Il exploite les informations structurelles en graphe de relations, ainsi que des mesures de similitude flexibles entre les propriétés des entités dans une recherche locale gourmande, ce qui le rend évolutif. SiGMa est un algorithme d'alignement de jeux de données qui peut gérer des millions d'entités. L'algorithme est facilement extensible avec des fonctions de notation personnalisées pour incorporer la connaissance du domaine. Il fournit également un équilibre entre précision et rappel, ainsi qu'entre le calcul et le rappel.

Afin de traiter des données incomplètes et des connaissances incertaines, des recherches ont choisi d'extraire des clés de liage [Atencia et al., 2020,Atencia et al., 2014]. De modéliser le problème dans la théorie d'analyse de concept formel (FCA) [Ganter and Wille, 2012], en montrant comment cette modélisation peut-être étendue dans la sélection de clé de liage et comment l'analyse conceptuelle relationnelle peut être utilisée pour gérer les dépendances circulaires. Linkex [Abbas et al., 2019] présente une formulation du problème afin de découvrir la clé de liage en utilisant Pattern Structures (PS) [Ganter and Kuznetsov, 2001], la généralisation de l'analyse de concept formel traitant des ensembles de données non binaires.

D'autres recherches ont modélisé le liage de données comme un problème de raisonnement avec incertitude [Al-Bakri et al., 2016], Ils ont conçu l'algorithme $ProbFR$ qui modélise le raisonnement sur des faits et des règles RDF incertains. Ils se sont basés sur la sémantique du *Datalog* [Fuhr, 2000] probabilistes. Pour le même auteur [Al-Bakri et al., 2015], on trouve un algorithme d'importation par requête qui alterne les étapes de réécriture de sous-requête, et sur d'interrogation sur mesure du

[9] http ://oaei.ontologymatching.org

cloud de données liées. Afin d'importer des données aussi spécifiques que possibles pour inférer ou contredire une cible de la même manière que les faits.

Bien que les différents systèmes dans la littérature fonctionnent bien et sont puissants. Ils présentent quelques inconvénients. Pour certain, leurs performances d'exécution et la qualité des solutions ne sont pas satisfaisantes pour les grandes tailles (c'est-à-dire un grand nombre d'instances) et les données de grandes dimensions (instances avec un grand nombre de data propriétés). Pour faire face à ces deux défis, le tableau ci-dessous 1.1 présente les avantages et les inconvénients des stratégies :

Système	Avantages	Inconvénients
RiMOM [Shao et al., 2016] Li et al. [Wang et al., 2013] MERGILO [Alam et al., 2017] LogMap [Jiménez-Ruiz and Grau, 2011]	Résultat meilleurs avec une bonne précision dans des petits jeux de donnée.	Impossible de traiter une grande échelle de données. Consomment du temps en raison combinaison de différentes mesures.
VMI [Li et al., 2013] SiGMa [Lacoste-Julien et al., 2013]	Meilleurs résultats quand l'utilisateur précise quelque paramètres de liage.	L'intervention de l'utilisateur.
Linkex [Abbas et al., 2019] [Atencia et al., 2020] [Atencia et al., 2014]	Modélisation d'une façon théorique, et testée sur de base spécifique.	

TABLE 1.1 : Avantages et inconvénients des systèmes étudiés.

Système RiMOM-IM : Risk Minimization based Ontology Mapping - Instance Matching

Hu et al. [Shao et al., 2016] ont implémenté RiMOM-IM l'un des systèmes qui a participé dans OAEI 2013 et 2016 [10]. C'est un algorithme itératif basé sur la stratégie de blocage pour réduire l'ensemble des instances candidates. Le choix de l'index de l'entité est basé sur les caractéristiques des prédicats et leurs entités. Le système utilise aussi des approches d'agrégation de similarité, basées sur une fonction exponentielle pondérée pour assurer un liage d'instance de haute précision. La nouvelle méthode de blocage réduit le temps d'exécution sans réduire la précision et le rappel. RiMOM-IM atteint la performance de 99 % de précision pour les petits jeux de données et moyennes. Le système RiMOM-IM se compose de cinq modules illustrés dans la figure

[10]Depuis 2004, l'OAEI organise des campagnes d'évaluation visant à évaluer les technologies d'appariement d'ontologies. http ://oaei.ontologymatching.org

1.9 qui inclus la configuration interactive initiale, génération des paires candidates, calcul d'alignement, alignement d'instances, et la validation. Les diffèrents modules sont détaillé dans ce qui suit.

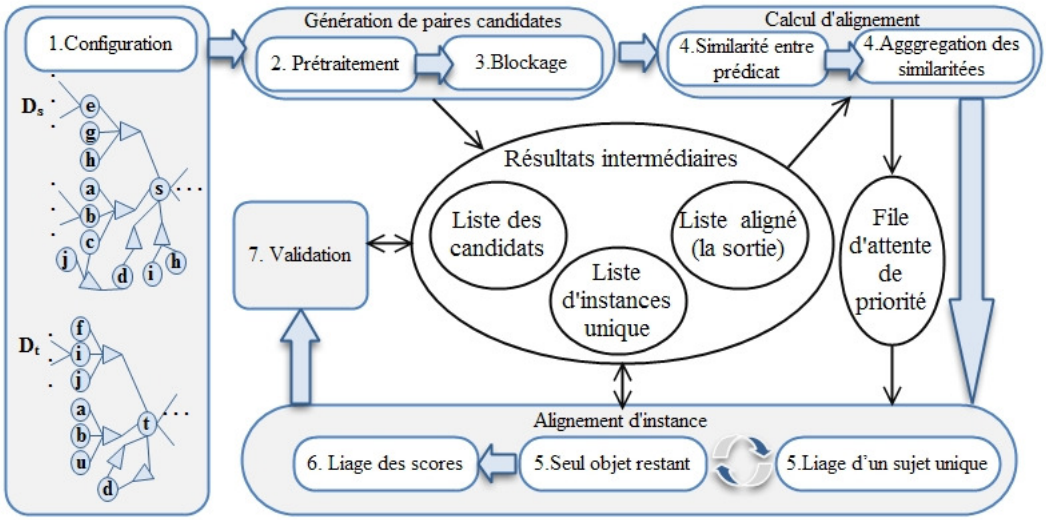


FIGURE 1.9 : Architecture du système RiMOM-IM [Shao et al., 2016].

Configuration interactive initiale

Le système commence par une configuration interactive. Il permet à l'utilisateur de charger les jeux de données et configurer les paramètres dans les diffèrents modules. Le module permet la sélection des prédicats d'alignement. Google traduction est utilisé pour la partie liage. Plusieurs similarités d'agrégation peuvent être sélectionnées, dans ce travail ils utilisent la fonction *ExpAgg* (définie dans la partie 1.4). Ainsi que le seuil de liage.

Génération des paires candidates

Dans cette partie, le module conduit au prétraitement des données, unifie le format et les valeurs pour quelques prédicats.

- Prétraitement des données : contient la suppression des mots vides(le, la, une,..), ainsi que les caractères spéciaux (♯, @,..). Ensuite le système calcule **TF IDF** [11] pour chaque mot dans la base de connaissances. Notez qu'un objet associé à

[11]Dans la recherche d'informations, TF IDF abréviation de term frequency–inverse document frequency, est une statistique numérique destinée à refléter l'importance d'un mot pour un document d'une collection ou d'un corpus.

un prédicat et un sujet précis consiste un pseudo document. En plus, puisque le format de différentes bases est inconsistant, donc le module applique quelque unification. A titre d'exemple le champ date, il y en a qui écrivent "2014, 05, 01" d'autres "Mai 1^{ier}, 2014" le module les unifie à "05-01-2014".

- Blocage : le blocage a pour objectif de prendre relativement un sous-ensemble de pairs candidats depuis tout l'ensemble des paires. A cause de la largeur de jeux de données, il est impossible de calculer le score de liage pour toutes les instances paires. Le système prend un prédicat comme étant les cinq premières objets (triés par ordre TF-IDF) comme l'index d'instances. Dans le cas où l'objet est une instance, toute l'URI est considéré comme un mot. Avec cette méthode le système a réduit considérablement les pairs avec le même prédicat et des objets distinctifs. Le système avec cette idée à réduit considérablement la comparaison de similarité, et prouve son efficacité. l'algorithme 1 est composé de trois phases, première boucle (de la ligne 4 à 19) index tous les instances pour chaque jeux de données. La deuxième boucle (de la ligne 20 à 26) parcours la table des indexes et génère candidat de l'ensemble C. La dernière étape (de la ligne 27 à 34) génère deux instances uniques des ensembles U_S et T_T un pour $\mathcal{D}_S$ et l'autre pour $\mathcal{D}_T$, qui vont être utilisées pour l'alignement.

Calcul d'alignement

Dans cette partie le système essaye de trouver le score entre l'ensemble des pairs candidats. Premièrement, la similarité entre prédicats : le système utilise la similarité de Jaccard dans le cas où l'objet de pairs de prédicat est les instances . Dans le cas où les pairs d'objets sont des textes, le système utilise la similarité de cosine. En plus d'autres similarités qui sont implémenté citer dans [Euzenat and Valtchev, 2004].

Deuxièmement, similarité d'agrégation : le système propose sa propre fonction d'agrégation exponentielle pondérée ($ExpAgg$) définie comme suit :

$$ExpAgg(S) = \frac{\sum_{s_i \in s} w'_i \times exp(w"_i \times s_i)}{\sum_{s_i \in s} w'_i \times exp(w"_i \times 1)} \tag{1.4}$$

sachant que s_i est la similarité de la $i^{ème}$ prédicat aligné. Quand s est large cela signifie que le prédicat comporte beaucoup d'informations. Le paramètre w' reflète l'importance du prédicat, et $w"$ contrôle l'importance du prédicat informatif. Les paramètres sont fixés d'une manière expérimentale, les meilleurs résultats sont obtenus avec ($w' = 1$ et $w" = 5$).

Algorithm 1 RiMOM- Blocking

```
Input : D_S, D_T
Output : liste des candidats C, ensembles des instances unique U_S, U_T
C ← ∅, U ← ∅, A← ∅, B← ∅.
for each D ∈ { D_S, D_T } do
    for each triple (s, p, o) ∈ D do
        if not A.ContainKey(p+o) then
            A.addKey(p+o) ;
        end if
        A.addValue(p+o, D, s) ;
        Divisé o en words(o) après triés les words(o) avec TF IDF dans l'ordre croissant ;
        Extraire les 5 premiers words(o) apparaissant dans l'autre base de connaissances au moins une fois.
        for each word w_i ∈ top 5 TF IDF words do
            if not B.ContainKey(p + w_i) then
                B.addKey(p+w_i)
            end if
            B.add(p+w_i, D, s) ;
        end for
    end for
end for
for each key ∈ A.AllKeys() do
    for each subject s_i ∈ A.getValues(key, D_S) do
        for each s_j ∈ A.getValues(key, D_S) do
            C.add(s_i,s_j) ;
        end for
    end for
end for
for each key ∈ B.AllKeys() do
    if B.getValues(Key, D_S).size()==1 then
        U_S.add(key, D_S, B.getValues(Key, D_S)) ;
    end if
    if B.getValues(Key, D_T).size()==1 then
        U_T.add(key, D_T, B.getValues(Key, D_T))
    end if
end for
return C,U_S,U_T ;
```

Alignement d'instances

Le système propose trois stratégies pour l'alignement qui s'exécutent d'une manière itérative.

- Liage d'un sujet unique : si une clé d'index contient le même ensemble d'instances dans U_S et U_T, alors les instances vont être alignées. L'alignement des paires va entraîner un nouvel alignement (par exemple, si (a,b) sont alignés, et c $\in U_S$ est indexé avec a+p, et d $\in U_T$ est indexé avec p+b, alors (c,d) sont alignés maintenant). Le système génère un nouvel alignement à partir des paires déjà alignées.

- Seul objet restant : si deux sous ensembles d'instances (de taille m-1) sont alignés par rapport à un prédicat précis, sachant que l'ensemble de taille m, l'objet restant de chaque prédicat peut être déduit comme aligné. De même, nous utilisons de manière itérative l'alignement des paires d'objets pour générer plus d'objets alignés jusqu'à ce qu'aucune nouvelle paire d'objets ne soit alignée.

- Liage des scores : étant donné que les processus précédents ont généré des paires alignées fiables qui peuvent être utilisées pour le calcul du score de similarité des paires d'instances, Ils considèrent la paire d'instances avec le score de correspondance le plus élevé au-dessus d'un seuil prédéfini σ comme alignée. Notez que pour garantir la précision, ils ont opté à chaque fois qu'une seule paire d'instances alignées avec le score le plus élevé. Ensuite, ils mettent à jour les ensembles d'instances uniques et les scores correspondants associés. Avec l'algorithme glouton [12] ils gardent uniquement la paire la plus correspondante à chaque fois. Comme ils ne peuvent pas garantir une optimisation globale avec l'algorithme glouton, ils ont rajouté le processus de validation.

Validation

Étant donné que de nombreux objets d'instances sont des URI faisant référence à d'autres instances, il existe encore une certaine non-détermination dans l'alignement de deux instances en raison de l'incertitude dans la situation d'alignement de leurs voisins compatibles. Ils ont ajouté un module de validation pour corriger certaines erreurs en recalculant les scores finaux de correspondance des paires d'instances alignées.

1.5 Conclusion

Nous avons montré dans ce chapitre l'importance du liage de données sur le web en insistant sur la masse de données à considérer. Ces données toujours en croissance, leur liage ne peut se faire manuellement. Une automatisation de ce processus, devient primordiale. Pour le faire et partant des travaux existants, nous proposons, dans un premier temps, de suivre une démarche en deux phases : - Une phase de pré-traitement des données qui permettra de sélectionner les informations les plus pertinentes. - Une phase d'alignement des instances qui permettra de mettre en évidence la propriété d'équivalence entre les instances.

Pour la concrétisation de nos objectifs, des techniques d'intelligence artificielle vont être testées. Il s'agit de méta-heuristiques et de techniques de fouille de données. Les notions liées à ces approches vont faire l'objet des deux chapitres suivants.

[12] Le principe de l'algorithme glouton est de faire un choix localement dans l'espoir que ce choix mènera à une solution globalement optimale.

Chapitre 2

Les méta-heuristiques et le liage

2.1 Introduction

Comme nous l'avons vu dans l'état de l'art du chapitre 1, nous devons traiter des données de masse comme ceux de DBpedia[1], ou autres jeux volumineux. La phase de pré-traitement des données est importante. Il est judicieux de ne traiter que les informations qui peuvent sembler pertinentes. Pour le faire, nous orientons nos recherches vers un problème d'optimisation.

Partant du fait que les méta-heuristiques et les algorithmes évolutifs sont des techniques indépendantes du problème, elles peuvent être parfaitement appliquées à notre problématique. Les méta-heuristiques se composent d'opérations simples, partant d'un état et de données initiales, le but de ces algorithmes est d'atteindre une ou plusieurs fonctions objectives.

Puis nous définissons dans la section 2.3 les heuristiques, d'où les méthodes aveugles et méthode exhaustive. La section 2.4 présente les métaheuristiques. Quant à la section 2.5, elle est consacrée à l'algorithme génétique. Nous terminons par des travaux de l'état de l'art.

2.2 Problème d'optimisation

L'optimisation existe dans plusieurs domaines, de la conception technique à la planification commerciale. La conception d'un produit doit maximiser les performances, la durabilité et l'efficacité énergétique et minimiser les coûts. Par conséquent, l'optimisation est importante pour de nombreuses applications. Par conséquent, l'optimisation est très importante pour de nombreuses applications.

[1] On y recense $4,233,000$ instances et $2,795$ propriétés différentes

Les algorithmes sont des instructions de calcul itératif. Les instructions sont des équations ou des descriptions, par exemple vérifier si un nombre est premier, ou générer des nombres aléatoires etc. La généralité des algorithmes dépend de la configuration initiale, donc effectue souvent des recherches localement. Un bon algorithme devrait être capable de changer sa configuration initiale. Car, c'est la recherche globale, qui tente de trouver des solutions finales moins sensibles au point de départ initial.

Comme la plupart des problèmes du monde réel sont aléatoires, la programmation mathématique non linéaire constitue une partie importante des méthodes d'optimisation mathématiques. Une large classe de problèmes de programmation non linéaire concerne la minimisation ou la maximisation.

Dans ce qui suit nous nous intéressons aux heuristiques et méta-heuristiques qu'on modélisera pour notre problématique dans les chapitres de contribution.

2.3 Les heuristiques

Une heuristique est généralement conçue pour un problème particulier. Une heuristique est un algorithme d'optimisation qui essaye de fournir une solution rapide, simple et adaptée à un problème d'optimisation. Les approches heuristiques parcourent l'espace des combinaisons en prenant en considération l'explosion combinatoire. Dans cette partie nous allons voir les méthodes aveugles avec les deux types de balayage (d'exploration de graphe), ensuite nous présentons l'algorithme A*.

2.3.1 Les méthodes d'exploration aveugles

Le processus de recherche d'une solution dans l'approche espace des états basés sur la représentation de graphe suit trois étapes : i) spécification d'un nœud initial correspond à l'état initial, ii) les successeurs d'un nœud sont calculés à l'aide des opérateurs, pour calculer tous les successeurs. iii) parmi les successeurs, s'il existe un état final, la solution est obtenue en parcourant le graphe de cet état vers l'état initial, donc les arcs doivent être orientés dans les deux sens, sinon, on continue l'expansion des noeuds.

Dans les algorithmes de graphes, il existe deux façons de balayage de l'espace de solution [Cormen et al., 2009] : soit par largeur (2.3.1.1) soit par profondeur (2.3.1.2). Une explication plus détaillée de ces deux types de recherches est donnée dans ce qui suit.

2.3.1.1 Recherche en largeur d'abord

Une recherche en largeur a été découverte par Moore [Moore, 1959] dans le contexte de la recherche de chemins à travers des labyrinthes. Elle est nommée ainsi car elle exploite la frontière entre les sommets découverts et non découverts uniformément sur toute la largeur de la frontière. Autrement dit, l'algorithme découvre tous les sommets à la distance k avant de découvrir les sommets à la distance $k + 1$.

La recherche en largeur d'abord construit un arbre en largeur, ne contenant initialement que sa racine (sommet) s. Chaque fois que la recherche découvre un nouveau noeud au cours du balayage de la liste d'adjacence d'un noeud u déjà découvert, le nouveau noeud v est ajouté à l'arborescence. Nous disons que u est **le prédécesseur** ou **le parent** de v dans l'arbre de largeur. Puisqu'un sommet est découvert au plus une fois, il a au plus un parent.

La **relations ancêtre** et **descendant** dans l'arbre de largeur sont définies par rapport à la racine s comme suit : si le sommet u est sur le chemin simple dans l'arbre de la racine s au sommet v, alors u est un ancêtre de v et v un descendant de u.

La figure 2.1.(a) illustre la progression de la recherche en largeur d'abord sur un exemple graphique.

2.3.1.2 Recherche en profondeur d'abord

La recherche en profondeur d'abord est largement utilisée depuis la fin des années 50, en particulier dans les programmes d'intelligence artificielle. Hopcroft et Tarjan [Hopcroft and Tarjan, 1973] ont préconisé l'utilisation de la représentation de liste d'adjacence sur la représentation de matrice d'adjacence pour les graphiques clairsemés et ont été les premiers à reconnaître l'importance algorithmique de la recherche en profondeur d'abord.

La stratégie qui suit la recherche en profondeur d'abord, comme son nom l'indique, consiste à chercher plus profondément dans le graphe chaque fois que cela est possible. La recherche en profondeur d'abord explore l'arête d'un noeud v le plus récemment découvert qui a encore des arêtes inexplorées. Une fois que toutes les arêtes de v sont explorées, la recherche revient en arrière pour explorer les arêtes sortant du noeud v à partir duquel elle a été découverte. Ce processus se poursuit jusqu'à ce que nous ayons découvert tous les noeuds possibles à partir du sommet source d'origine. S'il reste des noeuds non découverts, la recherche en profondeur d'abord sélectionne l'un d'entre eux comme nouvelle source et répète la recherche à partir de cette source. L'algorithme répète le processus jusqu'à ce qu'il ait exploité tous les noeuds.

La figure 2.1.(b) illustre la progression de la recherche en profondeur d'abord sur un exemple graphique.

(a) Recherche en largeur d'abord. (b) Recherche en profondeur d'abord.

FIGURE 2.1 : Illustration des méthodes aveugles.

2.3.2 méthode exhaustive : l'algorithme A^*

Les méthodes aveugles sont des méthodes **exhaustives**. Pour chaque méthode, il existe des limites : pratiques sur le temps d'exécution et l'espace mémoire, pour appliquer ces méthodes sur une grande catégorie de problèmes. Le A^* est un algorithme de recherche ordonnée basé sur une fonction d'évaluation.

Pour cela, d'autres stratégies ont été proposées comme par exemple l'algorithme A^* qui va tenter à chaque itération de se rapprocher du but. Il s'agit de privilégier les possibilités directement plus proches du but, en mettant de côté toutes les autres dans une liste de possibilités à explorer si jamais la solution explorée actuellement s'avère mauvaise.

L'algorithme de recherche ordonnée A^* est basé sur une fonction d'évaluation [Lauriere, 1987] :

$$f(X) = g(X) + h(X) \tag{2.1}$$

où : $g(X)$ est le coût de la chaîne allant du sommet s à un noeud X. $h(X)$ appelée heuristique est une estimation du coût de la chaîne reliant le noeud X à un nœud final. h est spécifique au domaine d'application.

L'algorithme 2 va donc d'abord se diriger vers les chemins les plus directs et prometteuse. Ensuite, dans le cas où ces chemins n'aboutissent pas ou bien s'avèrent mauvais par la suite, il examinera les solutions mises de côté (dans une liste ouverte). C'est ce retour en arrière pour examiner les autres possibilités qui nous garantit que l'algorithme essaye de nous trouver une solution.

Algorithm 2 L'algorithme A^*

```
1: Sélectionner le sommet S.
2: while la liste ouverte est non {∅} et le noeud but ∉ dans liste ouverte do
3:    Récupérer un sommet X de coût minimum.
4:    Ajouter X à la liste fermé.
5:    Ajouter les successeurs de X non développé à la liste ouverte en identifiant leur
      prédécesseur.
6:    if un successeur ∈ liste ouverte et nouveau coût est inférieur à l'ancien then
7:       Mettre à jour le coût.
8:       Mettre à jour le prédécesseur.
9:    end if
10: end while
```

Exemple d'application de l'algorithme A^*

Le jeu du taquin est imaginé par Sam Loyd dans les années 1870. Constitué d'une matrice carrée, subdivisée en neuf cases, sur laquelle les carrés sont numérotés de un à huit. Il reste donc une case libre ou le trou. Le seul mouvement autorisé consiste à faire bouger l'une des pièces adjacentes au trou vers celui-ci, ce qui revient à échanger leurs positions respectives [Lauriere, 1987].

Soit S_0 la configuration initiale et S_but la configuration finale qu'on veut atteindre (voir tab : 2.1).

$S_0 =$

2	8	3
1	6	4
7	□	5

$S_{but} =$

1	2	3
8	□	4
7	6	5

TABLE 2.1 : L'état initial et but du taquin.

Le symbole □ désigne la case trou. On veut minimiser le nombre de coups. Les fonctions sont définies comme ceci :

- g(S) = nombre de coups de S_0 jusqu'à S,
- h(S) = nombre de cases mal placées.

La figure 2.2 donne l'arbre de recherche développé par A^* avec la fonction $f(S)$, et pour chaque noeud le couple de valeur (g(S), h(S)) calculer par l'algorithme.

2.4 Les métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs : méta, du grec signifiant "au-delà" et heuristique. En effet, ces algorithmes peuvent optimiser

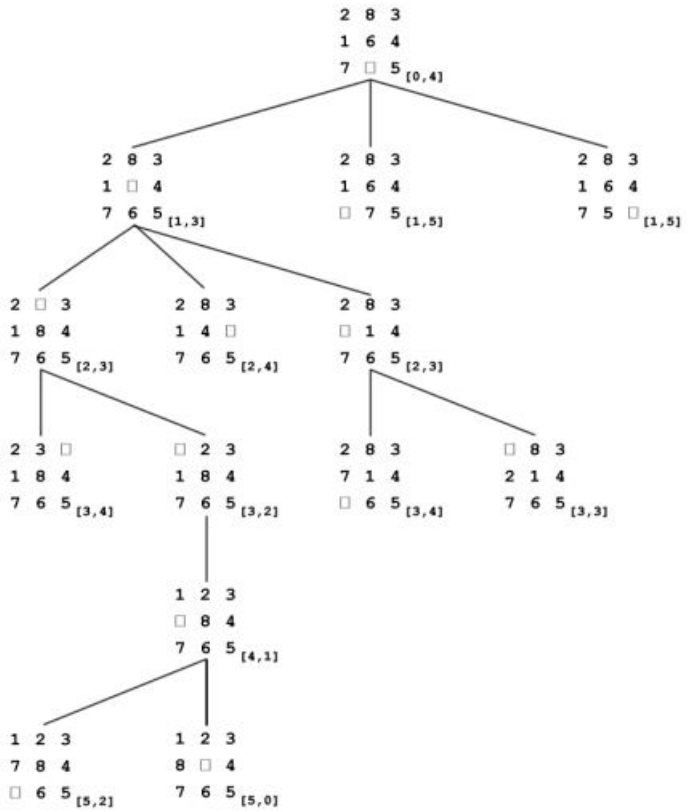


FIGURE 2.2 : L'arbre de recherche A^* pour le taquin .

une large gamme de problèmes différents, sans avoir besoin de changements profonds dans l'algorithme employé.

Les métaheuristiques sont conçus indépendamment des problèmes donnés. elles permettent d'obtenir une solution réalisable dont le coût est proche de la solution en un temps raisonnable. Les métaheuristiques sont généralement adaptables pour résoudre plusieurs problèmes.

Les métaheuristiques forment une famille d'algorithmes visant à résoudre des problèmes difficiles d'une manière approchée pour lesquels on ne connaît pas de méthodes d'algorithmes classiques plus efficaces (voir fig 2.3).

On distingue deux grands types de métaheuristiques :

- Les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. A titre d'exemple : Le recuit simulé, la recherche tabous et de recherche gloutonne aléatoire adaptative (GRASP : Greedy Randomized Adaptive Search Procedure).

- Les métaheuristiques à base de population de solutions. Ces dernières sont plutôt exploratoires et permettent une meilleure diversification de l'espace de recherche(d'où notre choix).

Principes des métaheuristiques

Bien qu'il existe de petites différences entre les métaheuristiques, dans [Mosteghanemi, 2010] un ensemble de principe ont été définis. Ils consistent tous une boucle itérative caractérisée par :

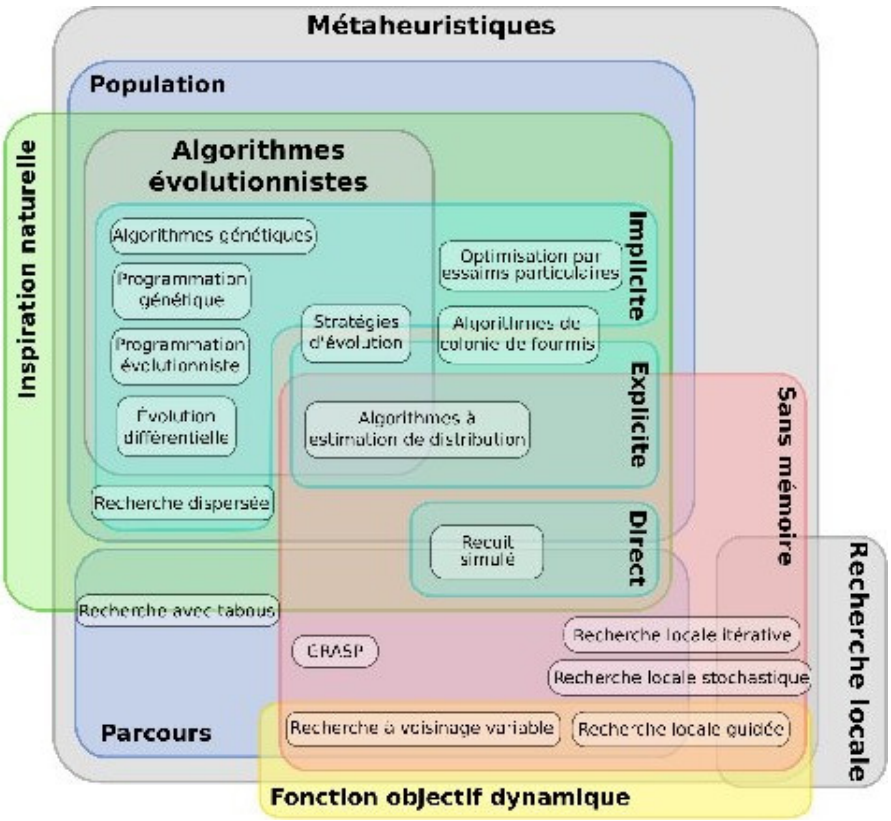


FIGURE 2.3 : Classification des métaheuristiques [2].

- Un état initial S_0 qui peut être une solution unique ou un groupe de solutions appartenant à l'espace de recherche.
- Un ensemble d'opérateurs qui permettent la construction du voisinage de l'état courant. Ce sont les techniques de transition d'un état à un autre.

[2]Par Johann "nojhan" Dréo — Travail personnel, CC BY-SA 3.0, `https://commons.wikimedia.org/w/index.php?curid=2942568`

- Une règle de transition qui permet la sélection de la prochaine solution parmi l'ensemble des successeurs.
- Une condition d'arrêt qui représente un critère d'arrêt. Généralement, il s'agit du nombre d'itérations au bout desquelles aucune amélioration sur la solution n'est observée.

Notions utiles dans la suite

Intensification/exploitation : Afin de parcourir des zones non exploitées. Un processus de combinaisons des bonnes propriétés des solutions trouvées est intégré dans les métaheuristiques, se nomme : ***L'intensification***

Diversification/exploration : Le processus qui est responsable d'introduire de nouvelles propriétés inexistantes au niveau des solutions exploitées nommé ***la diversification***, permet de mener la recherche vers des zones peu ou pas encore explorées.

Dans les métaheuristiques, les deux notions d'intensifications et de diversifications sont prépondérantes dans la conception. Les deux notions sont complémentaires, donc doivent atteindre un équilibre délicat entre elles. Il existe de nombreuses stratégies mêlant à la fois l'un et l'autre des aspects.

Un optimum local : Un optimum local est la meilleure solution que l'algorithme a pu avoir pendant l'exécution. L'inconvénient c'est que se n'est pas possible d'exploiter une autre zone de recherche qui pourrait être plus prometteuse. La diversification est la technique utilisée pour pouvoir éviter de se bloquer sur des optimums locaux.

Classification des métaheuristiques basées sur la population

Il existe plusieurs classifications des métaheuristiques, mais la plus connue est la catégorisation des algorithmes basés sur la population (voir Fig 2.4). Les métaheuristiques se composent de quatre classes [Dhiman and Kaur, 2018] :

- **Des algorithmes évolutionnaires :** s'inspirent des processus évolutifs tels que la reproduction, la mutation, la combinaison et la sélection. On cite quelques-uns de cette catégorie :
  - Algorithme génétique (GA) [Goldberg and Holland, 1988] : les origines de ces algorithmes remontent au début des années 1970, une technique très populaire et la plus largement utilisée. Ces algorithmes se détachent en

grande partie par la représentation des données du génotype, sous forme d'un vecteur binaire où sous forme d'une chaîne de caractères.

- Évolution différentielle (DE) [Storn and Price, 1995] : optimise un problème en conservant une population de solutions candidates et en créant de nouvelles solutions candidates en combinant celles existantes en utilisant des formules mathématiques simples. Puis en conservant la solution candidate qui a le meilleur score ou la meilleure aptitude au problème en cours.
- Stratégie d'évolution (ES) [Beyer and Schwefel, 2002] : l'algorithme le plus simple, il génère par mutation un nouveau individu à partir de l'individu parent et sélectionne l'un ou le nouveau afin de le conserver dans la population.
- Optimisation basée sur la biogéographie (BBO) [Ergezer et al., 2008] : elle consiste à l'étude de la répartition spatiale des espèces vivantes (végétales et animales) et des causes de cette répartition.

- **Des algorithmes basés sur les lois de la physique :** s'inspirent des processus physiques selon des règles de la physique telles que la force de gravité, la force électromagnétique, le chauffage et le refroidissement des matériaux. Parmi les algorithmes de cette catégorie on trouve :

  - Algorithme de recherche gravitationnelle (GSA) [Rashedi et al., 2009] : les agents sont considérés comme des objets et leurs performances sont mesurées par leurs masses. Tous ces objets s'attirent les uns les autres par la force de gravité, et cette force provoque un mouvement global de tous les objets vers les objets avec des masses plus lourdes. Par conséquent, les masses coopèrent en utilisant une forme directe de communication, par la force gravitationnelle. Les masses lourdes - qui correspondent à de bonnes solutions - se déplacent plus lentement que les plus légères, ce qui garantit l'étape d'exploitation de l'algorithme.
  - Algorithme de trou noir (BH) [He et al., 2009] : à chaque itération du trou noir, le meilleur candidat est sélectionné pour être le trou noir et le reste forme les étoiles normales. Après le processus d'initialisation, le trou noir commence à attirer des étoiles autour de lui. Si une étoile s'approche trop près du trou noir, elle sera avalée par le trou noir et disparaîtra pour toujours. Dans un tel cas, une nouvelle étoile (solution candidate) est générée

aléatoirement et placée dans l'espace de recherche et lance une nouvelle recherche.

- Recherche de système chargé (CSS) [Kaveh and Talatahari, 2010] : dans le CSS, chaque solution candidate contenant un certain nombre de variables de décision est considérée comme une particule chargée. La particule chargée est affectée par les champs électriques des autres agents. La quantité de la force résultante est déterminée en utilisant les lois électrostatiques et la qualité du mouvement est déterminée en utilisant les lois de la mécanique newtonienne. Il semble qu'un agent avec de bons résultats doit exercer une force plus forte que les mauvais, donc le montant de la charge sera défini en tenant compte de la valeur de la fonction objective.
- Algorithme de recherche basé sur la galaxie (GbSA) [Shah-Hosseini, 2011] : est une technique d'optimisation développée récemment par Hamed Shah-Hosseini à l'Université Shahid Beheshti-Iran. GbSA est une méta-heuristique qui utilise un algorithme d'escalade modifié comme recherche locale et ressemble aux bras en spirale de certaines galaxies pour rechercher l'optimum.

- **Des algorithmes basés sur des essaims de particules intelligentes :** basés sur l'intelligence d'essaims sont inspirés de l'intelligence collective des essaims dans la nature. On cite :

  - Optimisation d'essaim de particules (PSO) [Kennedy and Eberhart, 1995] : Elle s'inspire des déplacements collectifs de certains animaux sociaux comme les poissons et les oiseaux migrateurs.
  - Optimisation basée sur les colonies de fourmis(ACO) [Dorigo et al., 2006] : inspiré depuis le comportement des fourmis réelles à la recherche de nourriture. Les fourmis parviennent à trouver le chemin le plus court entre leur nid et une source de nourriture. Elles explorent d'abord les environs de leur nid en effectuant une marche aléatoire. Ensuite, elles déposent sur leur chemin une substance chimique nommée phéromone pour marquer les chemins favorables afin de guider leurs congénères à la source de nourriture.
  - Algorithme d'essaim de poissons artificiels (AFSA) [Neshat et al., 2014] : est l'une des méthodes d'optimisation parmi les algorithmes d'intelligence d'essaim. Cet algorithme s'inspire du mouvement collectif des poissons et de leurs différents comportements sociaux. Sur la base d'une série de comportements instinctifs, les poissons essaient toujours de maintenir leurs

colonies et manifestent en conséquence des comportements intelligents. La recherche de nourriture, l'immigration et la gestion des dangers se produisent tous sous une forme sociale et les interactions entre tous les poissons d'un groupe se traduiront par un comportement social intelligent.

- Colonie d'abeilles artificielles (ABC) [Karaboga, 2005] : l'idée provient de l'observation de l'exploitation des ressources alimentaires chez les fourmis. En effet, celles-ci, bien qu'ayant individuellement des capacités cognitives limitées, sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid.

- **Des algorithmes basés sur le comportement biologique :** se sont des mété-heuristiques qui s'inspirent du comportement des être vivants, on cite :

  - Algorithme firfly (FA) [She Yang, 2010] : est un algorithme métaheuristique qui s'inspire du comportement clignotant des lucioles et du phénomène de communication par bioluminescence.
  - Cuckoo searche (CS) [Yang and Deb, 2009] : Il a été inspiré par le parasitisme obligatoire des couvées de certaines espèces de cuckoo en pondant leurs œufs dans les nids d'autres oiseaux hôtes (d'autres espèces). Certains oiseaux hôtes peuvent entrer en conflit direct avec les cuckoo intrus.
  - Algorithme de chauve-souris (BA) [Yang, 2010] : il a été inspiré par le comportement de recherche de nourriture des chauves-souris, l'algorithme effectue le processus de recherche en utilisant des chauves-souris artificielles comme agents de recherche imitant le volume sonore naturel des impulsions et le taux d'émission de vraies chauves-souris.

2.5 L'algorithme génétique

Nous nous intéressons plus particulièrement à l'algorithme génétique (GA) par rapport aux nombreux travaux existants qui traitent le problème de liage de données. Il s'agit d'une métaheuristique bio-inspirée basée population qui fait partie des algorithmes évolutionnaires, et qui modifie une population de solutions en appliquant des opérateurs génétiques [Goldberg and Holland, 1988].

L'idée de base des GA est de simuler un processus évolutionnaire pour trouver de meilleurs individus (solutions). Un individu est basé sur un encodage sous forme de

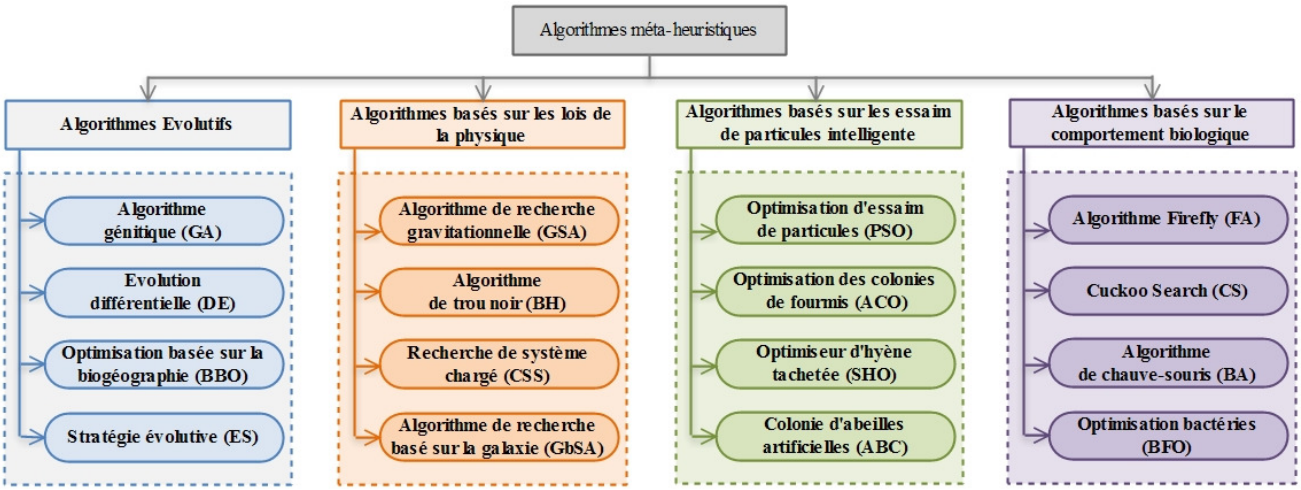


FIGURE 2.4 : Classification des méta-heuristiques [Dhiman and Kaur, 2018].

chaîne 0/1 de longueur fixe. L' individu sous l'encodage est appelé chromosome. Les opérateurs génétiques utilisés sur les individus d'une population sont :

- **Population initiale :** la population initiale est un ensemble d'individus. Chaque individu présente une solution possible. Il existe deux méthodes principales pour initialiser une population dans un GA : i) *initialisation aléatoire*, donc remplissage de la population initiale avec des solutions aléatoires.ii) *initialisation heuristique*, le remplissage de la population initiale en utilisant une heuristique connue pour le problème.

- **La fonction objectif :** est une fonction qui prend comme entrée une solution(un individu) et renvoie en sortie un score. Le score représente la pertinence de la solution, la probabilité qu'un individu soit sélectionné pour la reproduction est basée sur son score.

- **Sélection :** qui conserve les individus (des parents). Les principales utilisées sont la sélection par tirage à la roulette, la sélection par tournoi, la sélection par rang, etc... [Goldberg and Deb, 1991, Blickle and Thiele, 1996]

- **Croisement :** permet de combiner des individus parents afin de générer de nouveaux individus enfants. Le principe du croisement est de préserver les meilleurs gènes des parents, ainsi l'individu résultant du croisement à de grandes chances de ressembler aux parents. Le croisement peut se faire de différentes manières : uni-point, bipoint.

- **Mutation :** Un gène du chromosome (solution) est modifié. C'est-à-dire, on modifie aléatoirement une partie de la solution. D'abord une suppression de quelque gène aléatoirement, ensuite la reconstruction pour recouvrer les informations perdues.
- **Condition d'arrêt :** l'algorithme se termine si la population a convergé.

Algorithm 3 Algorithme génétique

```
1: Généré aléatoirement une population initiale de solutions
2: repeat
3:    Sélection
4:    Croisement
5:    Mutation
6:    Remplacement
7: until un critère d'arrêt est vérifié
```

Les avantages des algorithmes génétiques

Parmi les avantages de ces algorithmes et qui sont nombreux, on cite [Fogel, 1997] :

- Un des avantages capital est qu'il est conceptuellement simple. Les GA sont plus rapides et plus efficaces que les méthodes traditionnelles.
- les algorithmes génétiques peuvent être appliqués à pratiquement n'importe quel problème qui peut être formulé comme une fonction d'optimisation. Il nécessite une structure de données pour les solutions, une fonction de performance pour évaluer les solutions et des opérateurs pour générer de nouvelles solutions à partir d'anciennes.
- Les algorithmes offrent un pseudo code, tel qu'il est comparativement facile d'incorporer des connaissances où de s'hybrider avec d'autres méthodes. Ils optimisent les fonctions continues et discrètes ainsi que les problèmes multi-objectifs.
- Ce sont des processus hautement parallèles, puisque les ordinateurs de traitement distribués deviennent plus accessibles.
- Les algorithmes peuvent être utilisés pour adapter des solutions à des circonstances changeantes. La population disponible de solutions fournit une base pour de nouvelles améliorations.

- Le plus grand avantage vient de la capacité pour résoudre des problèmes pour lesquels il n'y a pas d'experts humains. Ils sont utiles lorsque l'espace de recherche est très grand et qu'un grand nombre de paramètres sont impliqués.

2.6 Liage des données et les métaheuristiques

Pour appliquer l'algorithme génétique sur le liage des ontologies, deux étapes de base doivent être formulées pour l'application : la présentation de l'individu qui présente le codage du problème, et la formulation de la fonction objective qui attribue à chaque individu une mesure de la performance.

2.6.1 GAOM : Genetic Algorithm based Ontology Matching

Le système GAOM (Genetic Algorithm based Ontology Matching) proposé par [Wang et al., 2006] est un algorithme génétique utilisé pour le problème du liage des ontologies.Il vise à trouver la similitude entre les concepts de deux ontologies. L'utilité des algorithmes génétiques (GA) est de réaliser un cas d'approximation proche de l'alignement optimal entre deux ontologies.

- **La présentation de l'individu** : soit n_1 le nombre de concepts de l'ontologie O_1, et n_2 le nombre de concepts de l'ontologie O_2. GA est appliqué pour trouver la fonction de liage optimale $\mathcal{M}$ entre les concepts de ces deux ontologies. Chaque solution ou individu dans la population serait un tableau à une dimension avec n_1 éléments pouvant prendre des valeurs entre 1 et n_2, noté ainsi : $N_1, N_2, ...N_{n1}$ où $N_i = \mathcal{M}(i) \in \{1, , n_2\}$, $i \in \{1, , n_1\}$ et cela signifie que le $i^{ème}$ concept dans O_1 est lié au $N_i^{ème}$ concept dans O_2.

- **La fonction objectif** : les auteurs ont opté pour l'utilisation du modèle de similitude de Tversky [Tversky, 1977] (eq 2.2) .

$$Sim_{O_1,O_2}(\mathcal{M}) = \frac{f((F_{o_1} \cup F_{o_2})|\mathcal{M})}{f((F_{o_1} \cup F_{o_2})|\mathcal{M}) + \alpha f((F_{o_1} - F_{o_2})|\mathcal{M}) + \beta f((F_{o_2} - F_{o_1})|\mathcal{M})} \tag{2.2}$$

où : $F_{o_1} et F_{o_2}$ sont des ensembles de caractéristiques de l'ontologie O_1 et O_2 respectivement. $f((F_{o_1} \cup F_{o_2})|\mathcal{M})$ représente un ensemble des correspondants entre O_1 et O_2 par rapport au liage $\mathcal{M}$. $f((F_{o_1} - F_{o_2})|\mathcal{M}) et f((F_{o_2} - F_{o_1})|\mathcal{M})$ sont deux ensembles d'éléments incomparables par rapport au liage $\mathcal{M}$. α et β sont deux paramètres compris entre 0 et 1. Ils déterminent l'importance relative du liage. f est une fonction définie la cardinalité de l'ensemble.

2.6.2 The compact hybrid Evolutionary Algorithm

Xue et al. [Xue and Liu, 2017] propose une approche évolutive hybride compacte, qui incorpore un algorithme génétique et une méthode de recherche locale au niveau du schéma et au niveau de l'instance pour le problème de liage des données dans le LOD.

- le processus de recherche locale est établi à chaque génération et mis en œuvre par un mécanisme de croisement binaire. L'approche utilise un vecteur de probabilité (PV) [Parsopoulos, 2009] pour caractériser la population dans une EA[3] hybride. Le nombre d'éléments dans PV est égal au nombre de bits de gènes de l'individu.

- L'approche suggère également une mesure de similitude basée sur le profil asymétrique (eq 2.3 et 2.4), qui est basée sur l'agrégation de la mesure de syntaxe et de la mesure linguistique pour évaluer les valeurs de similarité de la cartographie d'entités.

$$Sim_1(e_1, e_2) = \frac{f(e_1) \cap f(e_2)}{f(e_1)} \tag{2.3}$$

$$Sim_2(e_1, e_2) = \frac{f(e_1) \cap f(e_1)}{f(e_2)} \tag{2.4}$$

  où : $f(e_1)$ et $f(e_1)$ sont les cardinalités des profils e_1 et e_2 respectivement. $f(e_1) \cap f(e_1)$ est le nombre d'éléments identiques entre e_1 et e_2. De plus, si $0 \leq |sim1(e1, e2) - sim2(e1, e2)| \leq \alpha$, alors e_1 et e_2 sont identifiés comme sémantiquement similaires, sachant que α un seuil pour mesurer l'équivalence sémantique, les auteurs ont suggéré la plage de $\alpha \in [0.01, 0.15]$.

Cette approche nécessite un temps de calcul élevé, en particulier pour les grands ensembles de données.

Dans le même cadre [Xue et al., 2018] propose un algorithme co-évolutionnaire compact pour faire correspondre des entités d'ontologies biomédicales. D'une autre manière, une combinaison de l'EA compact et de l'algorithme co-évolutionnaire pour économiser la consommation de mémoire et le temps d'exécution, permet de surmonter l'amélioration des biais des solutions. Des expérimentations ont été faites sur le jeu de données Anatomy track and Large Biomed Piste pendant la campagne OAEI 2017.

[3]Evolutionary Algorithm

2.6.3 GOAL : Genetics for Ontology Alignments

GOAL (Genetics for Ontology Alignments) [Martinez-Gil et al., 2008] une approche qui calcule la fonction d'alignement d'ontologie optimale pour un ensemble d'entrées d'ontologie donné. L'algorithme travaille avec plusieurs objectifs : maximiser la précision d'alignement, maximiser le rappel d'alignement, maximiser la f-mesure ou réduire le nombre de faux positifs. GOAL est conçu pour fonctionner avec de nombreux algorithmes de base, tous sont choisis sur la base de leurs résultats, au lieu de leur comportement, ils peuvent donc être considérés comme des boîtes noires. L'idée derrière GOAL est que chaque composante du vecteur de poids est associée à un algorithme d'appariement de base. Les mêmes auteurs [Martinez-Gil and Aldana-Montes, 2011] présentent un méta-modèle pour sélectionner automatiquement les mesures de similitude appropriées pour le problème de liage d'ontologie. Il explore l'approche GOAL en tant que cadre de boîte noire, pouvant être étendu avec divers algorithmes de correspondance d'ontologies. Cette approche est hautement évolutive et permet d'obtenir des résultats plus précis. Cependant, de simples opérateurs génétiques sont utilisés dans le processus de recherche.

2.6.4 Le système VMI

Wang et al. [Li et al., 2013] ont développé l'approche nommée VMI qui construit pour chaque instance deux vecteurs. Le premier (NV) *vecteur de Nom*, contient les noms et étiquetages (labels). Le deuxième (VD)*nommé vecteur de Document Virtuel*, décrit les prédicats et donne les différentes informations des instances voisines. VMI construit des indexes inversés pour ces types de vecteurs et sélectionne les candidats de liages en fonction des indexes. Le VMI réduit le nombre d'espace de liage ce qui permet d'améliorer son efficacité. L'approche fonctionne efficacement avec un grand jeu de données pour un petit ensemble de prédicats. Par contre, les meilleurs résultats sont obtenus lorsque tous les prédicats de liage et les méthodes de récupération des valeurs sont spécifiées par l'utilisateur. Cependant, VMI repose sur l'algorithme génétique pour liage de données. Ainsi, leur approche repose sur un algorithme générique d'appariement d'instances, alors que quelques processus sont personnalisés pour des domaines spécifiques ; à savoir, une simple comparaison des chaînes de caractères est utilisée. Après, une combinaison des indexes une sélection des prédicats de liages est sélectionnée. Par contre, le VMI fonctionne bien avec les petites bases de données ontologiques dans OAEI 2009. Cependant, sa qualité diminue avec l'augmentation

du nombre d'instances. La figure ci dessous 2.5 représente l'architecture du système VMI, avec la description détaillée pour chaque composant à la suite :

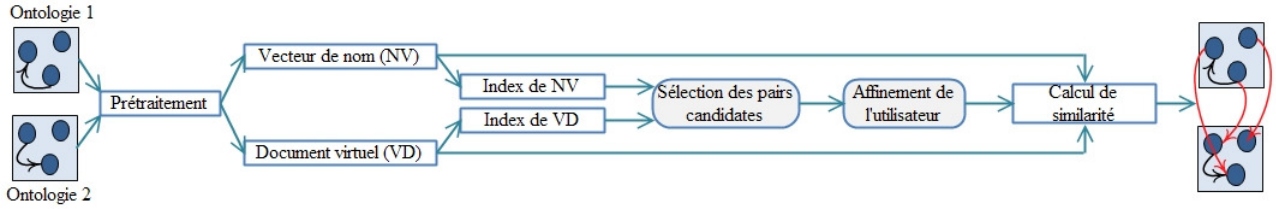


FIGURE 2.5 : L'architecture du système VMI [Li et al., 2013].

Dans ce qui suit une description détaillé pour chaque composant :

Construction des vecteurs et des indexes

Le système construit deux vecteurs, avec i comme index qui réfère au jeu de données i :

- Vecteur de Nom ($NV(i)$) : est obtenu à partir des noms des propriétés, après avoir éliminé les mots vides. Le poids de chaque mot est fixé par rapport à la fréquence d'apparition, car le système croit que les mots dans NV sont très importants. Pour chaque terme dans NV un index est créé.

- Vecteur de document virtuel ($VD(i)$) : la construction de ce vecteur est plus complexe, puisqu'elle va dépendre de ces propriétés de type chaîne de caractères. Après avoir éliminé les mots vides, on aura le vecteur $LD(i)$. Ensuite les informations voisines sont déduites à partir de l'expression 2.5.

$$NBI(i) = \sum_{i' \in NB(i)} (NV(i') + LD(i')) \tag{2.5}$$

  et $VD(i)$ est définie par

$$VD(i) = LD(i) + \gamma.NBI(i) \tag{2.6}$$

  où γ est le facteur qui indique la force des informations voisines dans le document virtuel.

  A la place, il est nécessaire de choisir des termes importants dans $VD(i)$ pour représenter l'instance. Pour évaluer la signification des termes dans le système, ce dernier utilise leur TF-IDF.

Après le calcul de TF-IDF, le système construit les indexes inversés [Zobel and Moffat, 2006] pour les vecteurs de noms et les documents virtuels respectivement. L'index inversé en règle générale utilise un dictionnaire pour stocker tous les termes uniques dans les documents. Pour chaque terme t du dictionnaire, il existe un lien vers une liste de publications $Lt = < d_t^1, d_t^2, ..., d_{nt}^t >$. L'élément d_i^t désigne l'identifiant de document du $i^{ème}$ document contenant t. En utilisant l'index inversé, nous pouvons trouver rapidement les documents qui contiennent un terme donné. Dans l'approche, seuls les termes dont les poids sont supérieurs à un seuil t_{vd} sont ajoutés au dictionnaire de l'index inversé.

Sélection des paires candidates

Pour chaque instance $i_s \in I_1$, l'instance $i_t \in I_2$ qui satisfait à l'une des règles suivantes est sélectionnée comme une paire candidate (5 est un paramètre fixé par les auteurs) :

- Si |NV (i_s)| ≥ 5 et |NV (i_t)| ≥ 5 et que les deux vecteurs ont au moins 2 termes en commun.
- Si |NV (i_s)| ≤ 5 ou |NV (i_t)| ≤ 5 et qu'ils ont au moins un terme en commun.

  Les deux premières règles proviennent de l'idée que l'approche est prête à comparer des instances avec des noms similaires. Lorsque les instances ont des noms courts, un terme de nom en commun pourrait signifier une paire candidate. Mais, lorsque les instances ont des noms plus longs, il est très possible que si elles correspondent elles devraient avoir au moins deux termes communs dans le vecteur de nom.
- Si VD(i_s) et VD (i_t) ont au moins 1 mot clé commun.

  La troisième règle est un complément aux deux premières règles pour s'assurer que si les instances ne sont pas similaires dans leurs noms mais elles ont des descriptions sémantiques similaires, elles seront également choisies comme candidats.

Dans le processus de sélection des pairs candidates, les deux index inversés sont utilisés pour générer des paires d'instances candidates. Prenons l'exemple inversé des vecteurs de noms NV. VIM génère des paires d'instances candidates qui ont au moins un terme commun dans leurs noms comme suit :

- Pour chaque terme unique t, obtenez sa liste L_t à partir de l'index inversé des vecteurs de noms NV ;

- Générer l'ensemble des paires d'instances du terme t : $\mathcal{T}_t = \{< i_s, i_t > | i_s \in L_t \cup I_1, i_t \in L_t \cup I_2\}$;
- Obtenez toutes les paires d'instances souhaitées en fusionnant les paires d'instances générées à partir de termes à indice inversé : $\mathcal{T} = \cup_{t \in T} \mathcal{T}_t$, ici $t \in T$ désigne l'ensemble de tous les termes uniques.

Toutes les paires d'instances dans $\mathcal{T}$ ont au moins un terme en commun. Ces paires ayant au moins 2 termes en commun peuvent également être trouvées dans $\mathcal{T}$. L'index inversé des documents virtuels est également utilisé de la même manière pour trouver des paires d'instances qui ont au moins un terme en commun.

Affinement de l'utilisateur

En prenant en considération les règles de sélection vu précédemment, on remarque qu'elles sont relativement simple. Comme il est vraiment difficile d'obtenir automatiquement la correspondance de propriété, VMI s'appuie sur l'utilisateur comme information prioritaire.

Pour les propriétés spécifiées par l'utilisateur, il existe deux types de scénarios. L'un consiste à vérifier si les valeurs de propriété se trouvent dans l'ensemble donné par l'utilisateur, et l'autre consiste à vérifier si les valeurs de propriété de deux instances sont identiques.

VMI permet aux utilisateurs de récupérer une partie particulière de valeur d'une propriété. Toutes les propriétés liées, les valeurs possibles, ainsi que les modes de récupération des valeurs sont spécifiés par les utilisateurs dans le fichier de configuration de VMI.

Calcul de similarité

Une fois les paires d'instances candidates sélectionnées, la similitude entre elles est calculée en utilisant respectivement le vecteur de nom et le document virtuel selon le modèle d'espace vectoriel. VMI utilise la distance $COSINE$ entre deux vecteurs pour obtenir la similitude. Elle est définie comme suit :

$$Sim(V_s, V_t) = \frac{\sum_{i=1}^{V}(v_{si}.v_{ti})}{\sqrt{(\sum_{i=1}^{V}(v_{si}^2)(\sum_{j=1}^{V}(v_{tj}^2)}} \tag{2.7}$$

où V est la dimension du vecteur, les éléments de V correspondent aux poids des termes calculés par la formule de $TF - IDF$. Ensuite, le système combine les deux

similitudes à un carré de la racine moyenne comme similitude initiale de la paire candidate.

$$Sim(i_s, i_t) = \sqrt{w_n.Sim_{nv}(i_s, i_t)^2 + (1 - w_n).Sim_{vd}(i_s, i_t)^2} \tag{2.8}$$

où w_n est le poids de la similitude du vecteur de nom. Une fois la similitude finale calculée, VMI effectue un filtrage de seuil sur la similitude pour extraire les résultats finaux.

2.6.5 SigMa : Simple Greedy Matching

SiGMa [Lacoste-Julien et al., 2013] est un algorithme de propagation itératif. Il exploite les informations structurelles en graphe de relations, ainsi que des mesures de similitude flexibles entre les propriétés d'entité dans une recherche locale gourmande, ce qui le rend évolutif. SiGMa est un algorithme d'alignement de jeux de données qui peut gérer des millions d'entités. L'algorithme est facilement extensible avec des fonctions de notation personnalisées pour incorporer la connaissance du domaine. Il fournit également un équilibre entre précision et rappel, ainsi qu'entre le calcul et le rappel. Le haut niveau de l'algorithme SiGMa fonctionne comme suit :

1. Il commence par un alignement partiel initial de données de bonne qualité m_0.
2. À chaque itération, il augmente le liage précédent avec une nouvelle paire appariée qui maximise le nombre de liage m_0, avec un petit ensemble de candidats qui préservent la faisabilité de la nouvelle correspondance.
3. L'algorithme s'arrête quand un seuil est atteint.

Dans le cadre du test de l'algorithme, deux ensembles de données d'alignement ont été construit (les bases de connaissances partiellement étiquetées).

Notions de SiGMa

L'algorithme SiGMa peut être vu comme l'optimisation gloutonne d'une fonction objective, qui évalue globalement l'adéquation d'un liage m particulier pour deux bases de connaissances. L'approche est motivée par le problème de maximisation du nombre d'entités similaires.

Cette fonction objective utilisera deux sources d'informations utiles pour choisir un liage : une fonction de similarité entre les paires d'entités définies à partir de leurs propriétés ; et un graphe de contribution de voisinage utilisant les voisins appariés.

SiGMa définit la fonction objectif suivante qui marque globalement l'adéquation d'un liage m :

$$obj(m) = \sum_{i,j \in D_1 \times D_2} m_{ij}[(1-\alpha)s_{ij} + \alpha g_{ij}(m)] \tag{2.9}$$

où ;

$$g_{ij}(m) = \sum_{k,l \in N_{ij}} m_{k,l} w_{ij,kl} \tag{2.10}$$

L'équation (2.9) contient un coefficient linière s_{ij} qui encode la similarité entre entités i et j, ainsi que le coefficient $w_{ij,kl}$ qui contrôle la tendance de l'algorithme de lier i avec j si k est déjà lié à l. N_{ij} est choisi d'une manière à respecter la structure graphique. Sa contribution à l'objectif code de manière cruciale le fait qu'un voisin k de i étant apparié à un voisin compatible l de j devrait encourager i à être apparié à j. $g_{ij}(m)$ compte le nombre d'entités voisines des éléments i et j d'une façon pondérée. $\alpha \in [0..1]$ est un paramètre qui troque entre le linéaire et $g_{ij}(m)$.

La correspondance m définit la fonction de score de similarité (2.11) dépendante du contexte, qui est utilisée pour choisir la paire appariée à l'étape 2 de l'algorithme4.

$$score(i,j,m)(1-\alpha)s_{ij} + \alpha\delta g_{ij}(m) \tag{2.11}$$

où ;

$$\delta g_{ij}(m) = \sum_{k,l \in N_{ij}} m_{k,l}(w_{kl,ij} + w_{ij,kl}) \tag{2.12}$$

SigMA algorithme

La conception de l'algorithme a été développée. Le score défini dans la ligne 5 pour sélectionner avidement la prochaine paire appariée est composé d'un terme statique s_{ij}, qui ne dépend pas de l'évolution de liage y, et d'un terme dynamique $\delta g_{ij}(y)$ qui dépend de y, bien que uniquement à travers le voisinage de $\mathcal{N}_{ij}$.

- **Structure de liage initiale** m_0 : l'algorithme peut prendre n'importe quel liage initial supposé de bonne qualité. Dans l'implémentation actuelle, cela se fait en recherchant des entités avec la même représentation sous forme de chaîne (avec une standardisation minimale telle que la suppression des majuscules et

Algorithm 4 SigMA algorithme

1: Initialiser le liage $m = m_0$.
2: Initialiser la file d'attente prioritaire S avec les paires candidates suggérées comme $S_0 \cup (\cup_{(i,j)\in m} N_{ij})$ le voisinage compatible de m, avec $score(i, j, m)$ comme clé.
3: **while** la file d'attente prioritaire S n'est pas vide **do**
4: Extraire (score,i,j) de la file d'attente S.
5: **if** score $\leq$ seuil **then**
6: arrêter.
7: **end if**
8: **if** i ou j sont déjà lié à une entité **then**
9: sautez-les et continuez la boucle.
10: **else**
11: ensemble m(i)=j. (mettre à jour la liste des candidats et les scores.)
12: **for** (k,l) $\in N_{ij}$ et non liée **do**
13: Ajouter (score (k,l,m), k, l) à la file d'attente S.
14: **end for**
15: **end if**
16: **end while**

la ponctuation) avec un liage sans ambiguïté $1-1$, c'est-à-dire que nous n'incluons pas une paire de liage exacte lorsque plus de deux entités ont cette même représentation de chaîne, augmentant ainsi la précision.

- **La fonction de score avec dépendance locale** : la fonction de score a un composant s_{ij} qui est statique (fixé au début de l'algorithme) à partir des propriétés d'entités telles que leur représentation sous forme de chaîne, et un composant $\delta gij(y)$ qui est dynamique, en regardant combien de voisins sont correctement liés. La partie dynamique ne peut en fait augmenter que lorsque de nouveaux voisins sont liées, et seuls les scores des voisins peuvent changer lorsqu'une nouvelle paire est liée.

- **Liste optionnelle statique des candidats S0** : facultativement, nous pouvons initialiser S avec une liste statique S_0 qui ne doit être notée qu'une seule fois car toute mise à jour de score proviendra des voisins déjà couverts par la ligne 11 de l'algorithme. S_0 a pour but d'augmenter l'exploration possible du graphique lorsqu'une autre source d'information solide (qui ne provient pas du graphique) peut être utilisée. Dans l'implémentation de l'algorithme, le système utilise un index inversé construit sur des mots pour suggérer efficacement des entités qui ont au moins deux mots en commun dans leur représentation sous forme de chaîne comme candidats potentiels.

- **Structures de données** : SiGMa utilise un tas binaire pour l'implémentation de la file d'attente prioritaire. Les insertions seront donc $O(log(n))$ où n est la taille de la file d'attente. Étant donné que la fonction de score ne peut qu'augmenter à mesure que nous ajoutons de nouvelles correspondances, nous n'avons

pas besoin de suivre les nœuds périmés dans la file d'attente prioritaire afin de mettre à jour leurs scores, ce qui donne une accélération significative.

2.6.6 Autres

Pour résoudre la convergence prématurée causée par l'application des algorithmes génétiques, Acampora et al. [Acampora et al., 2012] propose l'utilisation d'une approche évolutive hybride, qui combine l'algorithme génétique et la méthode de recherche en escalade. Ainsi, après application du croisement et des opérateurs de mutation, la recherche de montée est effectuée pour une convergence rapide vers les optimums locaux. Cette approche est bien équilibrée entre diversification et intensification. Cependant, cela nécessite un temps de calcul élevé pour le processus de recherche. Les mêmes auteurs [Acampora et al., 2013] ont développé une approche évolutive hybride, qui explore les caractéristiques de l'ontologie et l'expert du domaine pour optimiser efficacement les poids des fonctions de similitude agrégées du processus d'appariement de l'ontologie. Dans ce contexte, une approche, qui combine les opérateurs évolutifs et le processus de recherche local est étudiée.

2.7 Conclusion

Suite à notre étude des systèmes appliquant des algorithmes génétiques pour le problème de liage de schémas ou de données, nous avons constaté que dans le cas de traitement de masse de données(individus, attributs, relations...)ces méthodes ne peuvent être efficaces. Le problème peut provenir de l'information traitée qui peut ne pas être pertinente.

En effet, comme nous l'avons signalé dans le chapitre 1, nous allons adopté un processus en deux phases pour le liage de données. On commence par la phase de pré-traitement, qui consiste à extraire les informations pertinentes dans un but d'optimisation. Cette vision va être appliquée en utilisant l'algorithme génétique dans la phase de pré-traitement et fera l'objet de contribution(chapitre4).

Cependant, ces méthodes intelligentes (heuristiques et méta-heuristiques), ne sont pas les seules à avoir contribué à la résolution de notre problématique. Il s'agit de techniques fouille de données. Le chapitre suivant, fera l'objet de présentation de ces techniques et des travaux en rapport avec notre problématique.

Chapitre 3

La fouille des données

3.1 Introduction

La fouille de données, qui est basée sur les principes des statistiques, est le processus d'exploration et d'analyse de grandes bases de données pour découvrir et extraire des modèles à partir de ces données. Les algorithmes sont utilisés pour trouver des relations et des modèles dans les données, puis ces informations sur les modèles sont utilisées pour faire des prévisions et des prédictions. L'exploration de données peut être utiliser pour résoudre un éventail de problèmes commerciaux, tels que la détection de fraude, l'analyse du panier de marché et l'analyse du taux de désabonnement des clients... Traditionnellement, les organisations utilisent des outils d'exploration de données sur de grands volumes de données structurées, telles que des bases de données de gestion de la relation client ou des inventaires de pièces d'avion. Le but de l'exploration de données est d'expliquer et de comprendre les données. L'exploration de données n'a pas pour but de faire des prédictions ou de sauvegarder des hypothèses.

Différentes techniques d'exploration de données ont été étudiées pour traiter et analyser plusieurs types de modèles de données, où les tâches d'exploration de données. Les techniques d'apprentissage automatique non supervisées sont très prometteuses dans le cas de traitement de masse de données, les plus populaires parmi elles, on trouve la classification, l'exploration de règles d'association et le clustering. Ces techniques vont être détaillées pour une meilleure maîtrise. Leurs utilisations vont être présentées à travers des systèmes de l'état de l'art.

3.2 Apprentissage automatique

L'apprentissage automatique est une branche de l'intelligence artificielle qui vise à permettre aux machines d'effectuer leur travail de manière autonome. Étant donné

que les algorithmes d'apprentissage automatique nécessitent des données pour apprendre, et d'extraire une discipline de la base de données. De même, il existe des termes familiers tels que la découverte de connaissances à partir de données (Knowledge Discovery from Data), l'exploration de données et la reconnaissance de formes. Un domaine de recherche principal est que les programmes informatiques apprennent automatiquement à reconnaître des modèles complexes et à prendre des décisions intelligentes basées sur des données

L'apprentissage est, bien entendu, un domaine très vaste. Par conséquent, le domaine de l'apprentissage automatique s'est ramifié en plusieurs sous-domaines traitant de différents types de tâches d'apprentissage.

Puisque l'apprentissage implique une interaction entre l'apprenant et l'environnement, on peut diviser les tâches d'apprentissage en fonction de la nature de cette interaction.

3.2.1 Apprentissage supervisé

Dans l'apprentissage supervisé, l'objectif est de déduire une fonction ou un liage à partir de données d'apprentissage étiquetées. Les données d'apprentissage sont constituées du vecteur d'entrée et du vecteur de sortie d'étiquettes. Il est appelé apprentissage supervisé, car le vecteur de sortie se compose d'étiquettes pour chaque exemple d'apprentissage présent dans les données d'apprentissage. Ces étiquettes de vecteur de sortie sont fournies par le superviseur. Souvent, les superviseurs sont des humains. Les jugements humains sont plus chers que les machines, mais les taux d'erreur plus élevés dans les données étiquetées par les machines (suggèrent la supériorité du jugement humain). Deux groupes ou catégories d'algorithmes relèvent de l'apprentissage supervisé, ce sont : la classification et la régression.

La classification est le processus de recherche d'un modèle qui décrit et distingue des classes de données. Le modèle est dérivé sur la base de l'analyse d'un ensemble de données d'apprentissage (c'est-à-dire, des objets de données pour lesquels les étiquettes de classe sont connues). Le modèle est utilisé pour prédire l'étiquette de classe des objets pour lesquels l'étiquette de classe est inconnue. Un modèle de classification peut être représenté sous diverses formes : des règles IF-THEN, un arbre de décision, ou un réseau de neurones .

La régression modélise des fonctions à valeurs continues. En d'autres termes, la régression est utilisée pour prédire les valeurs de données numériques manquantes ou non disponibles plutôt que les étiquettes de classe. L'analyse de régression est une

méthodologie statistique qui est le plus souvent utilisée pour la prévision numérique, bien que d'autres méthodes existent également.

La classification et la régression peuvent devoir être précédées d'une analyse de pertinence, qui tente d'identifier les attributs qui sont significativement pertinents pour le processus de classification et de régression. Ces attributs seront sélectionnés pour le processus de classification et de régression.

3.2.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, nous manquons de superviseurs ou de données d'entraînement. En d'autres termes, tout ce que nous avons sont des données non étiquetées. L'idée est de trouver une structure cachée dans ces données. Il peut y avoir plusieurs raisons pour lesquelles les données n'ont pas d'étiquette. Cela peut être dû à l'indisponibilité de fonds pour payer l'étiquetage manuel. Avec de nombreux appareils de collecte de données, les données sont désormais collectées à un rythme sans précédent. Obtenir quelque chose de ces données sans le superviseur est important. Deux groupes ou catégories d'algorithmes relèvent de l'apprentissage non supervisé, ce sont : les règles d'association (3.3) et le clustering (3.4), que l'on décrit dans ce qui suit.

3.3 Les approches d'extraction des règles d'association

L'une des approches d'exploration de données les plus populaires consiste à rechercher des ensembles d'éléments fréquents à partir d'un ensemble de données de transaction et à dériver des règles d'association. La recherche d'ensembles d'éléments fréquents (ensembles d'éléments avec une fréquence supérieure ou égale à une prise en charge minimale spécifiée par l'utilisateur) n'est pas inoffensive en raison de son explosion combinatoire. Une fois que des ensembles d'éléments fréquents sont obtenus, il est simple de générer des règles d'association avec une confiance supérieure ou égale à une confiance minimale spécifiée par l'utilisateur.

L'extraction des règles d'association est l'une des approches d'exploration de données la plus populaire [Savasere et al., 1995], puisque c'est une méthode pour extraire des associations et des modèles depuis un ensemble de données gigantesques [Agrawal et al., 1993]. L'utilisation des règles d'association sont utilisées dans plusieurs domaines, tels que panier de consommation [Shaw et al., 2001], domaine médical [Gamberger et al., 1999].

A titre d'exemple, une instance dans une base de données de panier de marché est les articles achetés par un client lors d'une transaction. Les règles d'association extraient de la base de données du panier montrent si certains articles sont achetés lors d'une transaction, il est probable que ces articles peuvent être répétés dans d'autres transactions. Pour cela, les règles d'association peuvent aider dans la disposition des produits côte à côte afin de faciliter l'achat des clients.

Les règles d'association sont des instructions if..then consistant en un antécédent *si* et une partie conséquente *alors*. L'exploration de règles d'association implique la génération fréquente d'ensembles d'éléments suivie de la découverte des règles. Les éléments de l'ensemble de données qui dépassent un minimum support sont considérés comme des ensembles d'éléments fréquents. Les règles sont identifiées à partir de ces ensembles d'éléments fréquents ayant une confiance supérieure à un minimum support.

3.3.1 Concept de base

Considèrent un ensemble d'item set $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ et un ensemble de transaction $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ sachons que $T_i \neq \varnothing$.

Une règle d'association est de la forme : $X \rightarrow Y$, elle se compose de deux parties, X c'est la partie antécédente de la règle alors que Y est la partie conséquente. X et Y sont deux itemsets disjoints c'est à dire $X \cap Y \neq \varnothing$. Un itemset est un ensemble d'items inclus dans I ($X \in I$ et $Y \in I$,), un k-itemset est un itemset de taille k.

L'objectif des règles d'association est d'extraire un ensemble de règles qui couvrent un maximum d'ensemble de transaction. Cependant, puisque la base de transactions est grande, l'utilisateur essaye de chercher uniquement une partie de règles pertinentes. Pour cela, il existe deux mesures pour évaluer une règle, le support et la confiance de la règle.

Definition 3.3.1. Support

Le support d'un itemsets est le rapport entre le nombre de transactions contenant un itemset prédéfini et le nombre de transactions total. Le support de la règle $X \rightarrow Y$ est le support de $X \cup Y$.

Definition 3.3.2. Confiance

$$Confiance(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)} \tag{3.1}$$

L'objectif de ces deux mesures (le support et la confiance) sont de trouver les règles qui satisfont un support supérieur au minimum support (Minsup), et une confiance

supérieure au minimum confiance (Minconf) respectivement. Sachons que le Minsup et Minconf sont deux paramètres fixés par l'utilisateur.

3.3.2 Les algorithmes d'extraction des règles d'association

Le problème d'extraction des règles d'association est de nature NP-Complet, la résolution d'une manière naïve génère 2^n itemset. Dans le cadre de minimiser le temps d'exploitation des itemsset, plusieurs algorithmes ont été proposés dans un cadre d'exactitude, à titre d'exemple Apriori [Agrawal et al., 1993], DIC [Brin et al., 1997], et FPgrowth [Han et al., 2000].

Par la suite, des algorithmes approchés sont apparus. Mat et all [Mata et al., 2001, Mata et al., 2002] ont proposé GENAR (GENetic Association Rules) et GAR (Genetic Association Rules) des algorithmes d'extraction des règles d'association basés sur l'algorithme génétique, Sauf qu'ils ont une mauvaise présentation de l'individu (un mauvais encodage) ce qui a provoquer un calcul important lors de l'étape de l'évaluation. En effet, l'apparition d'autres algorithmes qui ont utilisé une bonne représentation d'individus. AGA [Wang et al., 2011] (Adaptive Genetic Algorithm) et ARMGA [Yan et al., 2009] sont deux stratégies permettant la convergence rapide de la population à un optimum.

Kuo et all [Kuo et al., 2011] ont proposé un algorithme basé sur les essaims de particules. Cet algorithme donne des meilleurs résultats à cause d'une stratégie avant et arrière de chaque particule.

Par la suite, plusieurs algorithmes d'extraction ont choisi le parallélisme soit avec une mémoire distribuée, ou à mémoire partagée, ou bien basé sur la technique GPU.

Dans ce qui suit, on présente quelques uns de ces algorithmes d'extraction.

Algorithme Apriori

L'algorithme Apriori est le plus utilisé pour l'extraction des règles d'associations, il est classé parmi les dix meilleurs algorithmes de fouille des données [Wu et al., 2008]. De nombreux algorithmes sont basés sur Apriori à titre d'exemple Park et All [Park et al., 1995] ont proposé DHP (Direct Hashing Pruning).

La première étape dans l'algorithme Apriori est la génération des itemset fréquents, d'une façon récursif. Les itemssets de taille $k+1$ sont générés à partir des items de taille k. Cependant, le temps de calcul est très long pour chaque itemset, donc, pour

chaque itemset son support est calculé. S'il est supérieur à un minimum support (Minsup), alors l'item sera sauvegardé dans la liste des itemsets fréquents. Le processus s'arrête lorsque aucun nouveau itemset ne sera généré.

La seconde étape est la génération des règles d'association. Elles sont générées à partir des itemsets de la première étape. Pour chaque itemst toutes les règles seront générées. Ensuite, la confiance sera calculée pour chaque règle. Si sa confiance est supérieure à MinConf donc la règle sera retenue sinon rejetée.

Algorithme Dynami Itemset Counting (DIC)

L'algorithme Dynamic Itemset Counting (DIC) a été proposé par Brin et al. [Brin et al., 1997] en 1997. Il permet d'améliorer l'efficacité de la recherche des itemsets fréquents par niveaux en diminuant le nombre de balayages.

Le principe est de définir un block correspondant à un nombre d'objets, et d'effectuer les lectures par bloc. Lorsque la fin de la base de données est atteinte, un balayage complet a été effectué et des lectures reprennent au début de la base. Après la lecture de chaque bloc, les supports des itemsets candidats sont mis à jour, de nouveaux itemsets candidats sont crées et les itemsets pour lesquels tous les objets ont été parcourus sont soit insérés dans l'ensemble des itemsets fréquents, soit inféquents selon leurs supports.

Algorithme Frequent Pattern Growth (FPgrowth)

La première étape consiste à analyser la base de données pour trouver les occurrences des itemset dans la base de données. Cette étape est la même que la première étape d'Apriori. Le nombre d'ensembles de 1 itemset est appelé la fréquence d'un ensemble d'1 éléments. La deuxième étape consiste à construire l'arbre. Pour cela, créez la racine de l'arbre. La racine est initialisée par un null. L'étape suivante consiste à analyser à nouveau la base de données et à examiner les transactions. Examinez la première transaction et découvrez l'ensemble d'éléments qu'elle contient. L'ensemble d'éléments avec le nombre maximum est pris en haut, ensuite l'élément avec un nombre inférieur et ainsi de suite. Cela signifie que la branche de l'arborescence est construite avec des ensembles d'éléments de transaction dans l'ordre décroissant. Les ensembles d'itemset sont classés par ordre décroissant de leur fréquence. Si un itemset de cette transaction est déjà présent dans une autre branche (par exemple dans la 1ère transaction), alors cette branche de transaction partagerait un préfixe en commun à la racine. Cela

signifie que l'itemset en commun est lié au nouveau nœud d'un autre itemset dans cette transaction.

3.4 Les approches de clustering

Le clustering est un apprentissage non supervisé car les classes ne sont pas connues à l'avances. Pour cette raison, le clustering est une forme d'apprentissage par observation, plutôt que d'apprentissage par des exemples. Le regroupement des données est utilisé dans plusieurs domaines : les statistiques, la recherche d'informations, l'apprentissage automatique, la technologie de base de données spatiales, la recherche sur le Web, la biologie, le marketing et de nombreux autres domaines d'application. En raison des énormes données collectées dans les différents domaines, l'analyse en groupes est devenue un sujet très actif dans la recherche data mining.

Pour former les groupes nous pouvons tomber dans l'un des problèmes suivants [Yang, 2019]

- Nous ne connaissons pas d'avances le nombre de cluster qu'on peut former d'après l'échantillon de données.
- L'ensemble de données peut être grand.
- Les données peuvent ne pas être nettoyées (des données bruyantes ou des données inutiles).
- Les données peuvent être incomplètes et peuvent donc manquer d'informations suffisantes pour un regroupement correct.
- Il existe d'autres problèmes, tels que les facteurs de temps, les données non structurées et les mesures de distance.

3.4.1 Concept de base

Un cluster est un groupe d'instances qui partagent des caractéristiques communes. Dans un ensemble de données de taille N, l'objectif est de former la taille N en k groupes ayant un sens (en d'autres termes k différents clusters ayant une similitude), dans le but de minimiser certaines mesures ou objectifs.

Une qualité d'un cluster dépend de : i) la mesure de similarité utilisée : Les distances sont très différentes selon que les domaines d'attributs sont des intervalles (continues), catégories, booléens. et ii) l'implémentation de la mesure de similarité.

Une technique de clustering doit satisfaire :

- **Similarité intra-classe importante** : Une similarité forte au sein des instances du même cluster.
- **Similarité inter-classe faible** : Une similarité faible entre les différents clusters.

3.4.2 Les algorithmes de base

Il existe de nombreux algorithmes de clustering dans la littérature [Han et al., 2011]. Les principaux algorithmes de regroupement fondamentaux peuvent être classés dans les catégories suivantes, bien qu'ils peuvent se chevaucher de sorte qu'une méthode peut avoir des fonctionnalités de plusieurs catégories.

1. **Algorithmes de partitionnement :** Une méthode de partitionnement construit k partitions de données à partir d'un ensemble de n objets, sachant en sorte que chaque groupe doit contenir au moins un objet ($k \leq n$). Les méthodes de partitionnement de base adoptent une séparation de cluster exclusive. En d'autres termes, chaque objet doit appartenir à exactement un cluster. La méthode de partitionnement utilise un partitionnement initial, ensuite une technique de relocalisation itérative qui va améliorer le partitionnement en déplaçant des objets d'un groupe à un autre.

   Les applications adoptent des méthodes heuristiques populaires, telles que des approches glouton comme les algorithmes k-means et k-medoids, qui améliorent progressivement la qualité du clustering et approchent un optimum local.

2. **Algorithmes hiérarchiques :** L'approche ascendante commence avec chaque objet forme un groupe distinct. Ensuite, il fusionne les groupes proches les uns des autres, jusqu'à ce que tous les groupes forment un seul groupe (le niveau le plus global de hiérarchie), ou que bien la condition d'arrêt soit respectée. L'approche descendante, contrairement à la précédente commence avec tous les objets dans un même cluster. Ensuite, pour chaque itération un cluster est divisé en d'autres clusters, jusqu'à ce que chaque objet se trouve dans un cluster, ou que bien la condition d'arrêt soit respectée. Les méthodes hiérarchiques souffrent du fait qu'une fois une étape (fusion ou division) effectuée, elle ne pourra pas être annulée. Cette rigidité est utile dans le cas où le coût de calcul est faible en n'ayant pas à se soucier du nombre de combinaison effectué.

   La figure 3.1 montre l'application d'AGNES (AGglomerative NESting), une méthode de clustering hiérarchique de fusion, et de DIANA (DIvisive ANAlysis),

une méthode de clustering hiérarchique de division, sur un ensemble de données de cinq objets (a, b, c, d, e).

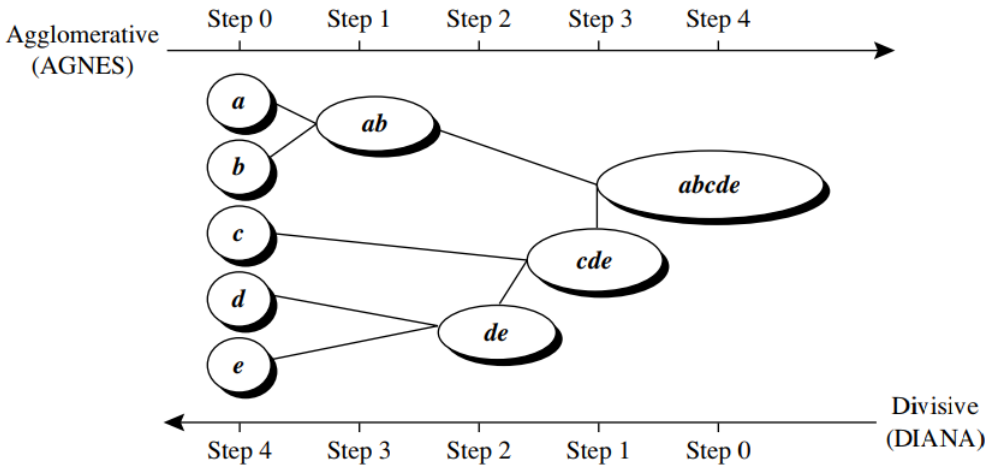


FIGURE 3.1 : L'application d'AGNES et de DIANA des méthodes de clustering hiérarchique.

3. **Algorithmes basés sur la densité :** Basés sur des notions de connectivité et de densité. Ces méthodes ne peuvent trouver que des clusters de formes sphériques et rencontrent des difficultés pour découvrir des clusters de formes arbitraires. L'idée globale est de continuer à développer un cluster tant que la densité (nombre d'objets) dans le voisinage dépasse un certain seuil. Ces méthodes sont utilisées pour filtrer le bruit ou les valeurs aberrantes et découvrir des grappes de forme arbitraire. exemple : Pour chaque point de données d'un cluster, le voisinage d'un rayon donné doit contenir au moins un nombre minimum de points.

   Vous apprendrez les techniques de base du clustering basé sur la densité en étudiant trois méthodes représentatives, à savoir, DBSCAN (Density-Based Clustering Based on Connected Regions with High Density) [Ester et al., 1996], OPTICS (Ordering Points to Identify the Clustering Structure) [Ankerst et al., 1999] et DENCLUE (Clustering Based on Density Distribution Functions) [Hinneburg and Gabriel, 2007].

4. **Algorithmes de grille :** Basés sur une structure à multiniveaux de granularité. Ces méthodes quantifient l'objet en un nombre fini de cellules, qui forment une structure de grille. Toutes les opérations de clustering sont effectuées sur la structure de la grille. L'avantage de cette méthode est le temps de traitement

rapide, qui est généralement indépendant du nombre d'objets, ne dépend que du nombre de cellules dans la dimension de l'objet quantifier. L'utilisation de grilles est souvent une approche efficace pour de nombreux problèmes de data mining, y compris le clustering. Par conséquent, les algorithmes de grille peuvent être intégrées à d'autres algorithmes de clustering telles que les méthodes basées sur la densité et les méthodes hiérarchiques.

STING (STatistical INformation Grid) [Wang et al., 1997] explore les informations statistiques stockées dans les cellules de la grille. CLIQUE (CLustering In QUEst) [Agrawal et al., 1998] représente une approche basée sur la grille et la densité pour le regroupement de sous-espaces dans un espace de données de grande dimension.

5. **Algorithmes à modèles :** Un modèle est supposé pour chaque cluster. Puis vérifier chaque modèle sur chaque groupe pour choisir le meilleur.

Algorithme K-means

L'algorithme k-means [MacQueen et al., 1967] est basé sur le calcul des centroïdes, supposant une data set $\mathcal{D}$ contient n objets. Le partitionnement des objets de $\mathcal{D}$ dans k cluster $\{C_1,C_2,...,C_k\}$ sachant que $C_i \subset \mathcal{D}$ et $C_i \cap C_j = \phi$ pour chaque $1 \leq i,j \leq k$.

Une fonction objectif est importante afin que les objets d'un même cluster soient similaires mais différents des objets d'autres clusters. En d'autres termes, la fonction objectif vise une forte similitude intracluster et une faible similitude intercluster.

Un centre de gravité c_i d'un cluster i est son point central. Le centroïde peut être calculer de différentes manières à titre d'exemple par la moyenne des objets attribués au cluster. La distance entre le centroïde et un abject o est définie par $dist(o, c_i)$ la distance euclidienne entre deux points o et c_i.

La qualité du cluster C_i peut être mesurée par la variation intra-cluster, qui est la somme des erreurs entre tous les objets dans C_i et le centroïde c_i, définie comme

$$\mathcal{E} = \sum_{i=1}^{k} \sum_{p \in C_i} dis(o, c_i)^2 \tag{3.2}$$

où E est la somme des erreurs pour tous les objets de l'ensemble $\mathcal{D}$, o est l'objet dans l'espace des données (multidimensionnels), et c_i (multidimensionnels) est le centre du cluster C_i.

Le problème du clustering est NP-difficile dans l'espace euclidien général, même pour deux clusters. Puisque dans le pire cas : il faudra énumérer le nombre de partitionnements possible qui sont exponentiels au nombre de clusters et calculer la valeur intra-cluster. De plus, si le nombre de cluster k et la dimension de l'espace de données d sont fixes ; le problème peut être résolu dans un temps $O(n^{dk+1}log(n))$ avec n est le nombre d'objets.

L'algorithme K-means procède comme suit : Premièrement, il sélectionne k objets au hasard depuis l'ensemble des données $\mathcal{D}$. Ces dernier représentent les centres initiales des k cluster. Ensuite, pour chaque objet et en fonction de la distance euclidienne entre l'objet et le centre du cluster, l'objet sera affecté au cluster le plus similaire. Au fur et à mesure l'algorithme ajuste le centre de chaque cluster d'une manière itérative, le nouveau centroïde est calculé à partir des objets inclus dans le cluster. Par la suite, les objets seront réaffectés avec les nouveaux centres de chaque cluster. Les itérations se poursuivent jusqu'à ce que la disposition en cours soit la même que celle formée précédemment (voir algo 5).

Algorithm 5 Algorithme k-means

1: **input :** K : nombre de cluster.
2: $\mathcal{D}$: ensemble des données de n objects.
3: Choix des k objets par hasard comme centres des clusters.
4: **repeat**
5: affecter l'objet au cluster le plus similaire.
6: mettre à jour le centre du cluter (calcul de moyenne des objets)
7: **until** aucun changement dans le centre des clusters

Algorithme K-medoids

L'algorithme K-medoids (voir algorithme 6) ressemble à l'algorithme K-means, mais les centres des clusters (les medoids) sont pris parmi les points à classer.

La méthode des k-médoïdes est plus robuste que les k-means en présence de bruit, car un médoïde est moins influencé par des valeurs de bruit ou d'autres valeurs extrêmes qu'une moyenne. Cependant, la complexité de chaque itération dans l'algorithme k-médoïdes est $O(k(n-k)^2)$. Pour de grandes valeurs de n et k, un tel calcul devient très coûteux, et beaucoup plus coûteux que la méthode des k-means. Les deux méthodes nécessitent que l'utilisateur spécifie k, le nombre de clusters.

Algorithm 6 Algorithme k-médoïdes

input : K : nombre de cluster.
$\mathcal{D}$: ensemble des données de n objects.
output : un ensemble de k clusters.
Choix des k objets par hasard comme des représentatifs initiaux.
repeat
    Attribuer chaque objet restant au groupe avec l'objet représentatif le plus proche.
    Sélectionner au hasard un objet non représentatif o.
    Calculer le coût total S, de l'échange d'objet représentatif o_j, avec o.
    si $S < 0$, alors permutez o_j avec o pour former le nouvel ensemble de k objets représentatifs ;
until aucun changement dans les clusters

3.5 Liages des données et les techniques de fouille de données

Au cours de la dernière décennie, plusieurs méthodes de résolution du problème de liage des ontologies ont été proposées. Les techniques de correspondance basées sur les instances conviennent également pour lier des instances dans des bases de données. Par conséquent, de nombreux travaux sont étudiés dans le domaine de fouille de données pour améliorer les performances de correspondance de données.

3.5.1 CSA (Cluster-based Similarity Aggregation)

CSA (Cluster-based Similarity Aggregation) [Tran et al., 2011] est un système qui combine différentes similitudes (cinq mesures, mesure de similarité basée sur des chaînes, mesure de similarité basée sur WordNet, la similarité de profil utilise les informations d'identifiant, d'étiquette et de commentaires contenus dans une entité.) pour extraire l'alignement entre les concepts d'ontologies.

Algergawy et all. [Algergawy et al., 2011] propose une approche basée sur le clustering pour le liage des ontologies à grande échelle. L'idée est de diviser le graphe de schéma des concepts en cluster K (clustering basé ascendant) en utilisant les similitudes des nœuds structurels basés sur le contexte. Après la partition de chaque ontologie, il définit le VSM (vector Space Model) pour extraire les clusters similaires, afin de produire les concepts similaires.

3.5.2 EIFPS (Extended Inverse Functional Property Suite)

Niu et all [Niu et al., 2012] ont développé une approche EIFPS (Extended Inverse Functional Property Suite), qui est un algorithme d'apprentissage semi-supervisé, pour affiner de manière récursive le processus d'alignement dans LOD (Linked Open Data) en utilisant les règles extraites par l'approche d'exploration de règles d'association. Les auteurs ont introduit une métrique basée sur un graphique pour estimer la probabilité et la règle de Dempster pour combiner les valeurs de confiance.

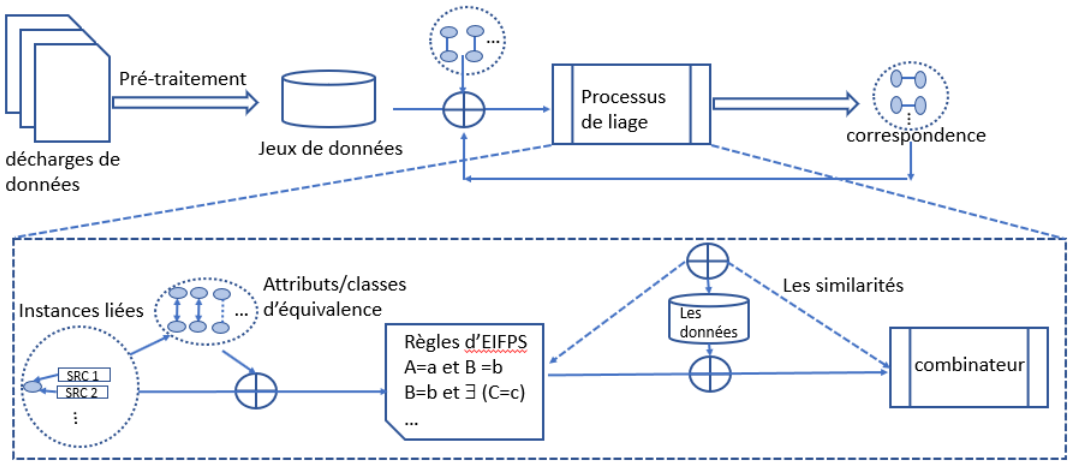


FIGURE 3.2 : L'architecture du système EIFPS [Niu et al., 2012].

La figure 3.2 représente l'architecture du système EIFPS. Après le prétraitement, un vecteur de paires propriété-valeur, similaire à la description du document, est attaché à chaque instance. Ensuite, les correspondances initiales sont importées dans le système pour conduire la découverte de nouvelles correspondances. Après cela, l'ensemble des graines est élargi avec des correspondances de haute qualité nouvellement trouvées et le système entre dans une autre boucle. La même procédure est répétée jusqu'à ce que la condition d'arrêt soit satisfaite. La principale méthode de découverte des correspondances dans approche est l'apprentissage de règles pour déduire les équivalences d'instances. Premièrement, les règles sont extraites dans des correspondances connues étant donné les équivalences de propriétés. Ensuite, ces règles sont appliquées aux instances non appariées pour en trouver de nouvelles équivalentes.

3.5.3 Autres

Sergio et all [Cerón-Figueroa et al., 2017] ont proposé un LOM (Learning Objects Metadata) en présentant la puissance des ressources d'homogénéité dans le contexte du

e-learning. L'application d'un classificateur associatif original pour l'appariement des ontologies est étudiée pour étendre et améliorer les outils disponibles pour l'apprentissage en ligne de manière sémantique. L'approche utilise une fonction de similarité basée sur des fonctionnalités qui nécessite des connaissances préalables sur l'ensemble d'apprentissage. L'approche a été testée dans le cadre du concours OAEI 2014 sur les bases de données d'ontologies. Les résultats montrent 90 % de précision pour certaines grandes bases de données d'ontologies.

Xue et all. [Xue and Liu, 2017] propose une approche évolutive hybride compacte, qui incorpore un algorithme génétique et une méthode de recherche locale à la fois au niveau du schéma et au niveau de l'instance pour le problème de correspondance d'ontologie. Un processus de recherche locale est établi à chaque génération, et il est mis en œuvre par un mécanisme de croisement binaire. De plus, il suggère également une mesure de similarité asymétrique basée sur le profil, qui est basée sur l'agrégation de la mesure de syntaxe et de la mesure linguistique pour évaluer les valeurs de similarité du liage d'entités. Cette approche nécessite un temps de calcul élevé, en particulier pour les grands ensembles de données. En plus [Xue et al., 2018] propose un algorithme co-évolutif compact pour apparier des entités d'ontologies biomédicales. D'une autre manière, une combinaison de l'EA compact et de l'algorithme co-évolutif pour économiser la consommation de mémoire et le temps d'exécution, surmonte l'amélioration du biais des solutions. Les expériences sont présentées dans la piste Anatomy de l'OAEI 2017 et la piste Large BioMed.

3.6 Conclusion

Pour solutionner le problème de liage de données, des techniques de fouille de données ont été utilisées. D'après les travaux des systèmes de l'état de l'art, nous pouvons dire que l'approche de liage de données n'a pas été suivie selon le processus que nous avons adopté. En effet, l'étape de pré-traitement des données n'est pas considérée. Ceci peut soulever le problème de l'information pertinente.

Pour y remédier, nous allons appliquer les règles d'association dans la phase de pré-traitement. Ce travail fait l'objet d'une contribution que l'on présentera dans le chapitre 5.

Dans un second temps, nous avons testé la technique de clustering dans le processus global de liage de données. Cette contribution fait l'objet de présentation dans le chapitre 6.

Deuxième partie

Contributions

Chapitre 4

Le système Genetic Feature Selection for Ontology Matching (GFSOM)

4.1 Introduction

Dans ce chapitre, nous présentons les principaux composants du système proposé appelé GFSOM (Genetic Feature Selection for Ontology Matching). C'est une approche de sélection des attributs qui est basée sur l'approche génétique pour le liage de données dans le web.

L'objectif de GFSOM est d'améliorer le problème de liage de données en prenant en compte les attributs pertinents des deux jeux de données à aligner. En d'autres termes, il s'agit d'abord d'exécuter l'algorithme génétique pour extraire les attributs les plus pertinents pour leur appliquer ensuite le processus d'alignement. Cette réduction permet d'une part de stimuler le processus de liage pour trouver les instances communes entre deux jeux de données, et d'autre part, elle vise à améliorer la qualité de l'alignement obtenu. Une modélisation de l'algorithme génétique sur notre système.

A la fin du chapitre, des expérimentations de GFSOM qui ont été menées vont être présentées. Les résultats sont prometteurs en termes de temps d'exécution et qualité du processus d'alignement par rapport à certaines approches de la littérature.

4.2 L'architecture de l'approche GFSOM proposée

Dans [Belhadi et al., 2018], nous avons proposé l'algorithme GFSOM pour le liage des données qui est basé sur l'algorithme génétique pour la sélection des propriétés(attributs)

les plus pertinents. Le liage des données est déclenché (la figure 4.1 pour plus de détails).

L'étape de sélection des attributs est d'abord effectuée sur l'ensemble des attributs de chaque jeu de données, ce qui donne un sous-ensemble optimal d'attributs qui représente parfaitement les deux jeux de données. Cette étape est considérée comme une étape de pré traitement. Elle ne sera exécutée qu'une seule fois. Pour ce faire, un dossier d'archivage sera construit pour chaque jeu de données du système. Dans cette étape, un algorithme génétique est réalisé pour améliorer le processus de sélection des attributs sans perdre la qualité.

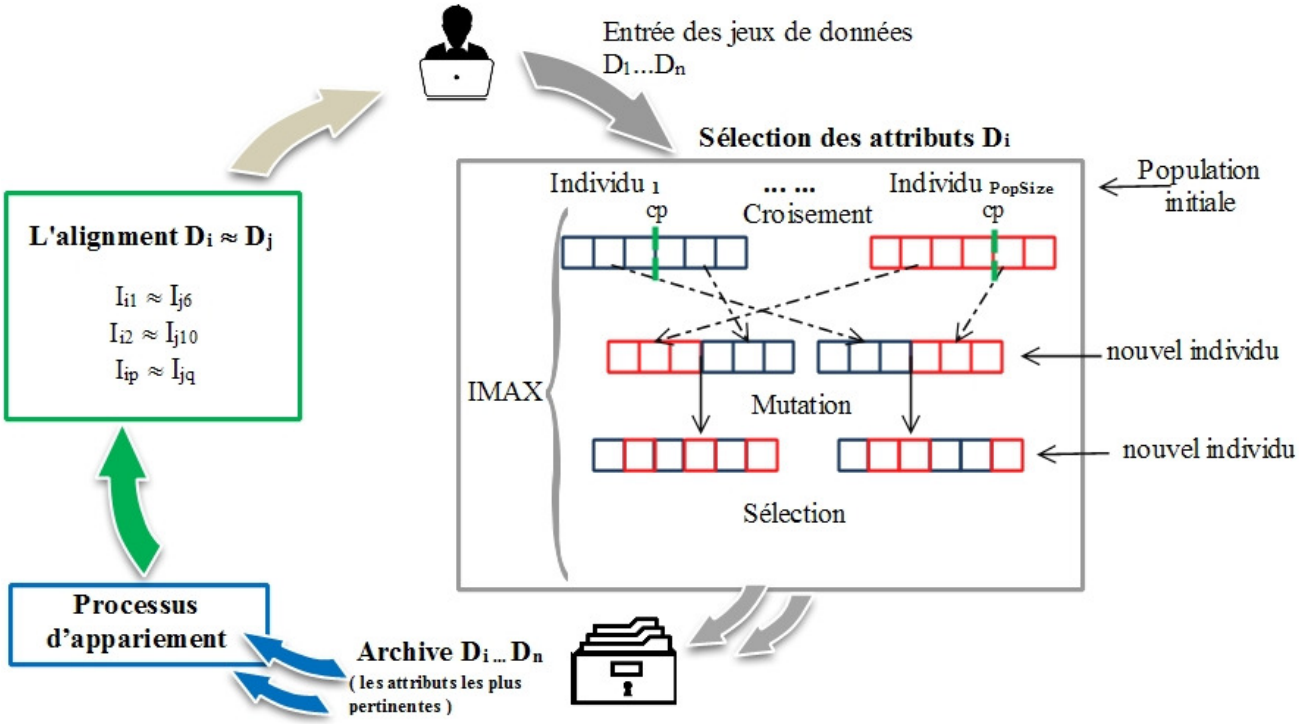


FIGURE 4.1 : Architecture du système GFSOM.

4.2.1 Codage des solutions

Il représente le codage ou représentation d'une solution du problème à traiter. Une solution doit être maniable pour que les autres opérateurs génétiques puissent le manipuler sans risque de perte de données. Le processus commence par générer aléatoirement $PopSize$ individus depuis l'ensemble global de propriétés m. Un individu S est un vecteur binaire de taille m,

$$S[i] = \begin{cases} 1 & \text{si } la\_propriété\_est\_sélectionnée. \\ 0 & \text{sinon.} \end{cases}$$

Exemple Soit un ensemble de propriétés $P = \{Nom, Prénom, Age, Profession\}$
Nous considérons un ensemble d'individus tel que $m = 4$ et $PopSize = 2$:
$S_1 = \{1, 0, 1, 0\}$ signifie que les propriétés (Nom et Âge) sont sélectionnées.
$S_2 = \{1, 0, 1, 1\}$ signifie que les propriétés (Nom, Âge et Profession) sont sélectionnées.

4.2.2 Évaluation d'une solution

Pour s'assurer que l'algorithme est sur la bonne voie, nous devons évaluer chaque solution, ainsi, ne garder que celles qui sont prometteuses afin d'optimiser l'espace mémoire.

Fonction d'évaluation, ou bien fitness, celle-ci permet d'évaluer un individu, elle est équivalente à la fonction objectif dans un problème mathématique.

Pour chaque individu S_i de l'ensemble global $PopSize$, nous calculons la fonction objectif qui va essayer de maximiser le nombre de données liées trouvées. Si on considère un alignement référence A_{ref} et le résultat trouvé avec le sous-ensemble d'attributs A_{result}, alors la fonction objectif essaye de maximiser $A_{ref} \cap A_{result}$.

Exemple Considérant deux résultats d'alignement A_1 et A_2, obtenu lors la sélection des attributs de S_1 et S_2 respectivement. La taille du fichier d'alignement de référence est de 10 instances ($A_{ref} = 10$), donc il y a 10 instances en commun entre les jeux de données $\mathcal{D}_1$ et $\mathcal{D}_2$.
Soit A_1 le $A_{result} = 5$ et supposons que les 5 instances sont des alignements vrais donc $A_{ref} \cap A_{result} = 5$.
Soit A_2 le $A_{result} = 6$ respectivement donc $A_{ref} \cap A_{result} = 6$.
$Objectif(A_2) > Objectif(A_1)$, A_2 est une solution meilleure que A_1.

4.2.3 Calcul de voisinage

À chaque itération appelée génération, une nouvelle population avec le même nombre d'individus est créée. Cette génération est mieux adaptée à la solution optimale. La création d'une nouvelle population se fait à partir d'application des opérateurs génétiques : le croisement, la mutation et la sélection. Un rappel de ces différents opérateurs est donné :

- **Le croisement** permet de produire de nouveaux individus. Il consiste à choisir un couple (parents) aléatoirement de la population initiale S_1 et S_2. Ensuite, une position aléatoire et identique pour les deux individus *cp* sélectionné aléatoirement entre 1 et m. Cela produit deux parties qu'on croise, afin d'obtenir

deux nouveaux enfants S_{11} et S_{12} qui héritent des caractéristiques de leurs parents. Les premières propriétés avant le *cp* de S_1 sont transférées vers S_{11}, et les premières propriétés avant le *cp* de S_2 sont transférées vers S_{12}. Les propriétés restantes de S_1 et S_2 sont transférées à S_{12}, respectivement S_{11}.

Exemple : Si on prend $S_1 = Individu_1$ et $S_2 = Individu_3$ et $cp = 2$.
On aura S_{11}= {**1,0**,1,1}, et S_{12} = {1,0,**1,0**} tel que les bit en gras sont celles de S_1

- **La mutation** l'opération est appliquée pour empêcher que l'évolution de la solution se fige vers l'optimum local, on inverse un bit dans un chromosome. cette opération permet d'assurer une recherche aussi bien globale que locale (diversité). Si une propriété est présente dans l'individu donné, alors on la bascule aléatoirement à 0 sinon à 1.

  Exemple : On prend S_{11}= {1,0, 1,1} et on choisit d'inverser aléatoirement un nombre d'éléments. Nous pouvons avoir S_{11}= {1,1, 0,1} si le nombre aléatoire est deux et les positions sont deuxième et troisième.

- **La sélection** est un procédé qui donne la chance aux individus avec une fonction d'objectif maximale de continuer à faire partie de la génération suivante. Elle consiste à donner une chance aux meilleures solutions trouvées pour s'améliorer encore plus dans la prochaine itération.

  Ensuite, GFSOM ne conserve que les meilleurs individus de taille $PopSize$ (les autres sont supprimés). Ce processus est répété jusqu'à ce que le nombre maximal d'itérations $IMAX$ soit atteint.

4.2.4 Processus de liage

Dans la deuxième partie, le processus d'alignement, le système essaye de trouver les correspondances en comparant chaque instance du jeu de données source à chaque instance du jeu de données cible en prenant en considération les attributs sélectionnés dans la partie précédente. La validation de liage se fait par le plus grand nombre de valeurs en commun entre les deux instances. Ensuite, la performance de l'appariement est de comparer les correspondances trouvées avec le fichier d'alignement référence.

Pour le processus d'appariement, le modèle proposé apprend à fixer les meilleurs paramètres ($PopSize$, $IMAX$). Si le taux d'alignement dépasse le seuil fixé par l'utilisateur, l'alignement pour le test des jeux de données est lancé.

4.3 Étude expérimentale

Dans ce qui suit, plusieurs expérimentations sont effectuées dans le but de valider l'approche proposée. Les tests sont effectués sur le jeux de données DBpedia[1]. Les résultats obtenus seront comparés avec des algorithmes de l'état de l'art .

4.3.1 Description des données

Pour valider les performances de notre système GFSOM, des expérimentations approfondies ont été réalisées sur le jeu de données DBpedia. Son principe est de proposer une version structurée et normalisée au format du web sémantique des contenus de Wikipedia. En 2018, on y recense $4,233,000$ instances et $2,795$ propriétés différentes selon son site[2]. Le choix de cette base au lieu des benchmarks des compagnes d'évaluation classique s'est fait par rapport au nombre important d'attributs trouvés dans les jeux de DBpedia. Ceci nous permet de mieux voir les performances de notre l'application.

4.3.2 Extraction des données

RDF représente des informations en utilisant des triplets sémantiques, qui comprennent un sujet, un prédicat et un objet. Il existe plusieurs syntaxes d'RDF, que nous allons représenter par la suite. Avant cela, nous prenant un exemple de graphe RDF (voir fig 4.2). L'exemple définit un laboratoire qui importe deux membres qui ont un lien entre eux. Nous allons voir les diffère syntaxe qui représente le même graphe.

- **RDF/XML**[3], est une syntaxe définie par le W3C, pour exprimer un graphe RDF comme un document XML. Il est le premier format de syntaxe RDF standard du W3C. C'est une syntaxe difficile à lire par un humain, réservé à la machine. (voir la fig4.3)
- **Turtle (Terse RDF Triple Language)**[4], est similaire à celle de SPARQL, un langage de requête RDF. Turtle fournit un moyen de regrouper trois URI pour en faire un triple, et fournit des moyens d'abréger ces informations, par exemple en prenant en compte les parties communes des URI. (voir la fig 4.4)

[1]http ://wiki.dbpedia.org/Datasets
[2]https ://wiki.dbpedia.org/services-resources/ontology
[3]https ://www.w3.org/TR/rdf-syntax-grammar/
[4]https ://www.w3.org/TR/turtle/

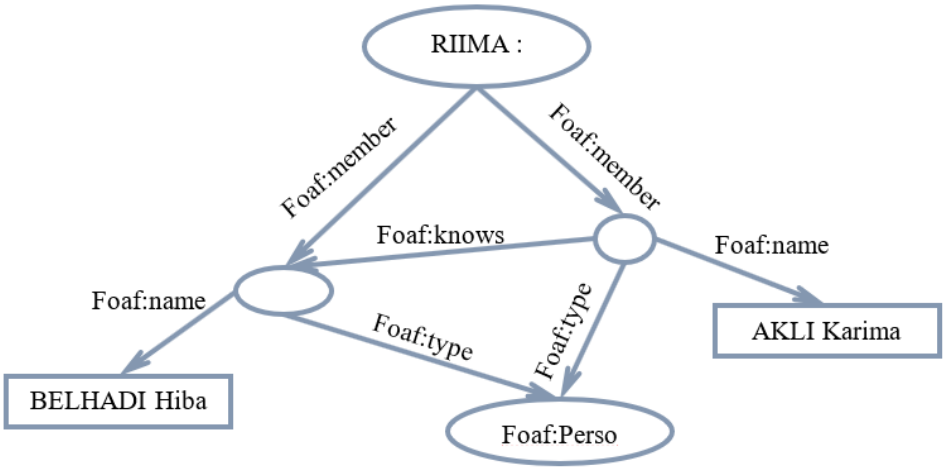


FIGURE 4.2 : Exemple de RDF.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/" >
 <foaf:Group rdf:about="http://usthb.dz/#lab">
  <foaf:member>
   <foaf:Person>
    <foaf:name>Hiba</foaf:name>
    <foaf:knows
     rdf:resource="http://facultyei.dz/#pa"/>
   </foaf:Person>
  </foaf:member>
  <foaf:member>
   <foaf:Person rdf:about="http://facultyei.dz/#pa">
    <foaf:name>AKLI Karima</foaf:name>
   </foaf:Person>
  </foaf:member>
 </foaf:Group>
</rdf:RDF>
```

FIGURE 4.3 : Syntaxe RDF/XML.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://usthb.dz/#lab>
      a foaf:Group ;
        foaf:member [
                  a foaf:Person ;
                  foaf:name "Hiba" ;
                 foaf:knows <http://facultyei.dz/#pa> ],  <http://facultyei.dz/#pa> .
<http://facultyei.dz/#pa>
      a foaf:Person ;
      foaf:name "AKLI Karima" .
```

FIGURE 4.4 : Syntaxe Turtle.

- **N-Triples**[5], chaque triplet est écrit sous la forme : <IRI du sujet> <IRI du prédicat> <IRI de l'objet>. N-Triplet est une syntaxe plus facile à lire par un humain. (voir la fig 4.5)

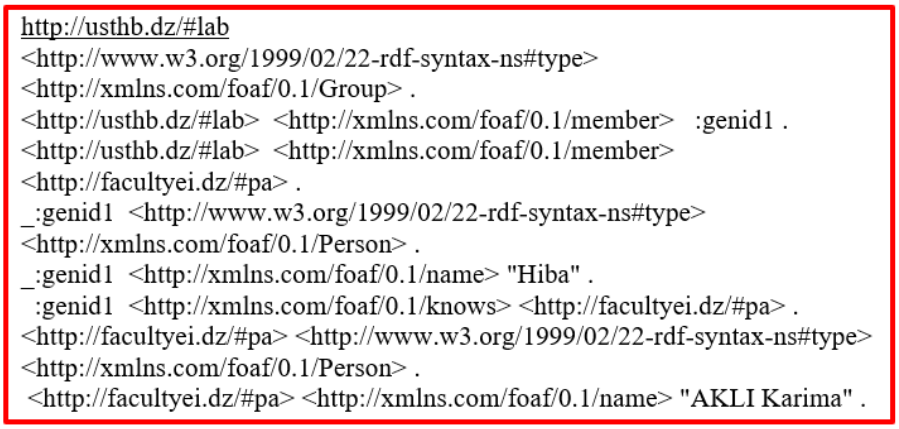


FIGURE 4.5 : Syntaxe N-Triples.

Le challenge est de définir un système qui peut extraire les instances des jeux de données, quelque soit leur syntaxe. Pour cela il existe plusieurs outils, les plus reconnue sont :

- **Apache Jena**[6] Jena est une API Java pour les applications du web sémantique. Jena contient des interfaces pour représenter des modèles, des ressources, des propriétés, des littéraux. Nous l'avons choisie pour la simplicité et la faciliter de son intégration à Java.

- **RDFox**[7] RDFox est hautement évolutive en mémoire et un moteur de raisonnement sémantique. Il prend en charge le raisonnement parallèle en mémoire partagée pour RDF, RDFS, et OWL 2 RL. C'est un logiciel multiplateforme écrit en C ++ qui est livré avec un wrapper Java permettant une intégration facile avec n'importe quel logiciel basé sur Java.

- **Mobi**[8] est une plate-forme gratuite qui relie les sources de données dans un graphe de connaissances, créant un environnement pour les équipes et les communautés. Mobi utilise OWL 2 pour la création d'ontologies, le langage de requête SPARQL. Pour le traitement et la gestion des données utilise RDF.

[5]https ://www.w3.org/TR/n-triples/
[6]https ://jena.apache.org
[7]https ://www.oxfordsemantic.tech/product
[8]https ://www.w3.org/2001/sw/wiki/Mobi

La plate-forme Mobi applique les meilleures pratiques recommandées par le World Wide Web Consortium (W3C) pour soutenir la croissance organique des connaissances dans divers domaines.

4.3.3 Prétraitement des données

Le prétraitement des variables dépend de la qualité des données récolté. Si le processus de récolte est manuel, il faut s'attendre à quelques erreurs de saisie. La phase de prétraitement de données constitue une phase importante dans le processus global. Elle vise à rechercher une représentation des données informative afin d'améliorer la qualité des résultats.

La normalisation des données est souvent importante d'un point de vue numérique où le fait d'avoir des données qui se ressemblent améliore la convergence des algorithmes et la précision des calculs.

Cette phase comporte des opérations de nettoyage. Dans certains cas, le nettoyage des données exige que certaines valeurs soient renseignées ou corrigées, dans d'autres cas, les valeurs devront être tout simplement supprimées. Car les causes les plus courantes d'inexactitude dans les données sont les valeurs manquantes, et les fautes de frappe.

En ce qui concerne la phase de la réduction des données : Une fois les données collectées, les instances en plusieurs exemplaires ou doublons doivent être identifiées et ramenées à un seul exemplaire.

La dernière phase comporte la transformation des données, en d'autre terme la discrétisation des variables numériques, l'encodage des valeurs.

4.3.4 Expérimentations

Tous les algorithmes ont été implémentés en langage de programmation Java et les expériences ont été exécutées sur un ordinateur de bureau équipé d'un processeur Intel $I7$ et d'une mémoire de $16GB$. La qualité du système d'appariement d'ontologies a été évaluée à l'aide de F_mesure, qui est un compromis entre le rappel et la precision. Plus F_mesure augmente, plus l'outil à évaluer est performant. F_mesure est définie pour chaque alignement de référence R et chaque alignement A obtenu comme suit :

$$F\_mesure(A, R) = \frac{2 \times Precision \times Rappel}{Precision + Rappel} \tag{4.1}$$

où la précision est la proportion des correspondances correctes, parmi l'ensemble de celles contenues dans A.

La fonction Precision : A × A → [0..1] tel que :

$$Precision(A, R) = \frac{|R \cap A|}{|A|} \tag{4.2}$$

et le rappel est la proportion de correspondances correctes contenues parmi toutes celles qui sont correctes.

La fonction Rappel : A × A → [0..1] tel que :

$$Rappel(A, R) = \frac{|R \cap A|}{|R|} \tag{4.3}$$

4.3.5 Comparaison avec les systèmes EIFPS et RiMOM-IM

L'objectif de cette expérimentation est de comparer GFSOM avec deux algorithmes EIFPS [Niu et al., 2012] et RiMOM-IM [Shao et al., 2016]. Le choix de ces deux systèmes repose sur une méthode d'apprentissage (EIFPS) et une méthode itérative (RiMOM-IM) performant dans l'alignement des ontologies à base d'instances. Dans le but de diversifier les méthodes utilisées et de situer une méthode génétique entre elles.

RiMOM-IM est déjà vu dans la section 1.4. C'est un algorithme itératif basé sur la stratégie de blocage pour réduire l'ensemble des instances candidates. EIFPS présenté dans la section 3.5.2 est un algorithme d'apprentissage semi-supervisé, pour affiner de manière récursive le processus d'alignement dans LOD (Linked Open Data) en utilisant les règles extraites par l'approche d'exploration de règles d'association. Les deux algorithmes ont été implémentés afin de tester leurs performances avec notre système, par rapport au temps d'exécution et la qualité de l'alignement.

Le liage de données se fait avec le même jeu de données DBpedia. Donc, le jeu de données source et cible sont les mêmes et un alignement de référence n'est qu'un cas spécial d'alignement. On essaye de trouver les correspondances dans un temps d'exécution court et avec un sous-ensemble d'attributs pertinents.

La figure 4.6 montre le temps d'exécution des trois approches en tenant compte de toutes les instances et propriétés.

- Lorsque le nombre d'appariements varie de 100 à $1,000,000$, le GFSOM a surpassé les deux autres approches. De plus, le temps d'exécution de GFSOM s'est stabilisé à 105 secondes.

- Les deux autres approches exigent beaucoup de temps et nécessitent plus de 900 secondes pour effectuer $1,000,000$ alignements dans tout le jeu de données DBpedia, pour un grand nombre d'instances et un grand nombre d'appariements.

Ces résultats ont été obtenus en utilisant l'approche génétique à l'étape de prétraitement, où seules les caractéristiques les plus pertinentes ont été sélectionnées.

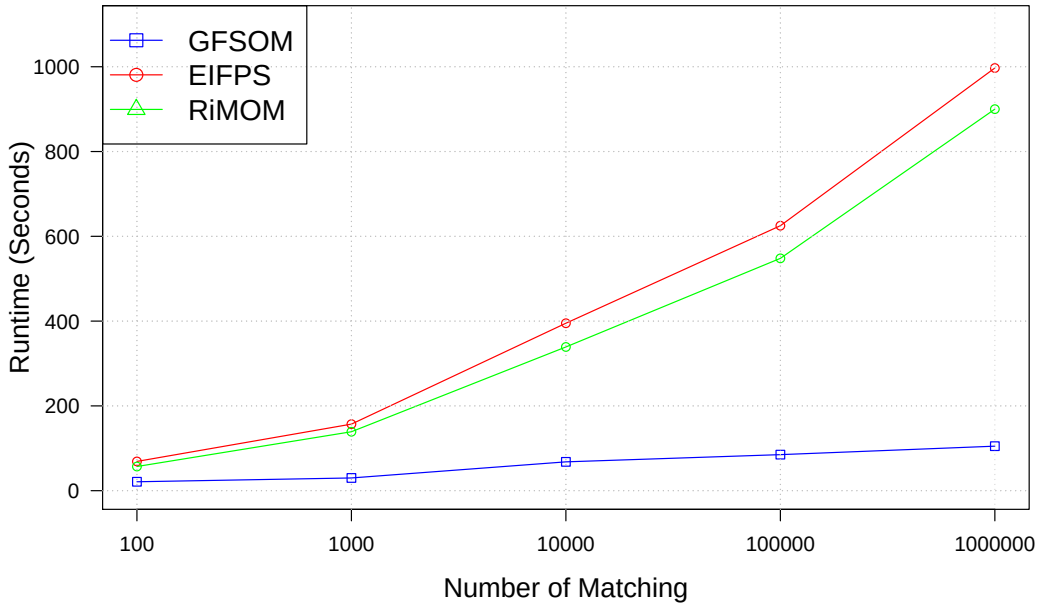


FIGURE 4.6 : Comparaison de temps d'exécution du GFSOM, EIFPS et RiMOM utilisant le jeu de données DBpedia

La dernière expérimentation a été réalisée pour comparer la qualité de l'alignement du système GFSOM avec ceux des algorithmes (EIFPS et RiMOM) sur DBpedia. En faisant varier le pourcentage de propriétés de 20% à 100%, GFSOM a surpassé les deux autres algorithmes en ce qui concerne la valeur F_mesure (un compromis entre le rappel et la précision). La figure 4.7 montre les résultats, ces derniers ont montré que la qualité de GFSOM n'était pas affectée par l'augmentation du nombre de propriétés. Ainsi, la qualité de GFSOM atteignait 90%, tandis que celles de EIFPS et de RiMOM était inférieure à 72%. Ces résultats ont été obtenus grâce à la sélection des attributs pertinents avec l'algorithme génétique.

4.4 Conclusion

Dans ce chapitre, nous avons vu le système GFSOM. L'algorithme génétique est d'abord exécuté pour sélectionner les propriétés les plus pertinentes, le processus de

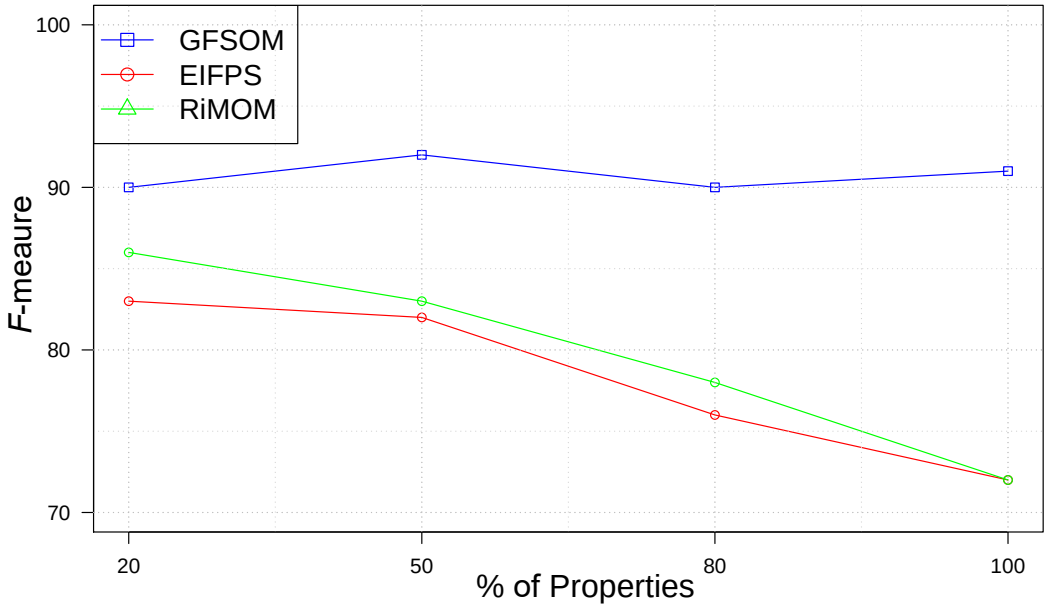


FIGURE 4.7 : Comparaison de F_mesure du GFSOM, EIFPS et RiMOM sur DBpedia.

liage est ensuite appliqué, en ne prenant en considération lors de la comparaison que les propriétés des jeux de données déjà sélectionnées.

D'après les résultats expérimentaux, il est évident de voir que le GFSOM conçu surpasse les algorithmes d'état de l'art sur le jeu de données DBpedia. Nous pouvons également voir que le GFSOM développé bénéficie de la sélection des fonctionnalités et de l'algorithme génétique en termes de temps d'exécution et de qualité du processus de liage.

Dans nos travaux futurs, nous explorerons d'autres techniques d'exploration de données pour le problème de liage des données. Nous viserons en outre à utiliser à la fois le clustering, ou l'exploration des items set fréquente pour traiter de grandes bases.

Chapitre 5

Le système Data Mining for Ontology Matching based instances (DMOM)

5.1 Introduction

L'utilisation de l'approche des règles d'association dans le problème de liage est examinée dans cette partie. Nous allons identifier les correspondances entre les instances et les propriétés de diffèrents jeux de données.

Les principaux composants du système réalisé appelé DMOM (Data Mining for Ontology Matching based instances) pour sélectionner efficacement les propriétés de données des instances va être présenté. Des approches basées sur des techniques de fouille des données sont utilisées afin de voir leur efficacité dans le processus de liage des données. La première approche appelée exhaustive, explore l'arbre de recherche d'énumération de manière aléatoire en générant à chaque itération un sous-ensemble d'attributs. Chaque nœud est évalué en exécutant le processus d'alignement sur les attributs sélectionnés. La seconde approche, dite statistique, utilise des valeurs statistiques pour sélectionner les propriétés les plus pertinentes. La troisième approche appelée FIM (Frequent Itemsets Mining), explore la corrélation entre différentes propriétés et sélectionne les propriétés les plus fréquentes décrivant les instances globales du jeu de données.

L'objectif donc de DMOM est d'améliorer le liage de données en prenant en considération les propriétés pertinentes de deux jeux de données. Cette réduction permet, d'une part de booster le processus de liage pour trouver les instances en commun, et d'autre part de prouver la qualité de l'alignement obtenu.

A la fin du chapitre, on présente des expérimentations de DMOM qui ont été

menées sur des jeux de données d'ontologie de la campagne OAEI (Ontology Alignment Evaluation Initiative) et celui de DBpedia.

5.2 Architecture de DMOM

Vu le nombre d'instances et du nombre de propriétés de données, l'alignement des données dénote un problème polynomial. Par exemple, n et n' correspondent respectivement au nombre d'instances du premier jeu et du deuxième jeu de données. m et m' sont les nombres de propriétés du premier et deuxième jeu de données, respectivement. Une solution triviale pour l'alignement compare chaque instance du premier jeu avec chaque instance du deuxième jeu en prenant en compte toutes les propriétés de données des deux jeux de données. Ainsi, des comparaisons $n \times n' \times m \times m'$ sont nécessaires pour trouver les correspondances, ce qui rend la complexité polynomiale. Cependant, pour DBpedia contient 4 233 000 instances et 2 795 propriétés de données différentes, ce qui signifie que $n = n' = 4233000$ et $m = m' = 2795$, donc il y a 144×10^{18} comparaisons possibles. En conséquence, le processus d'appariement prend beaucoup de temps et la qualité de l'appariement peut être affectée.

Pour améliorer les performances globales de l'alignement des données de haute dimension, et motivés par les techniques d'exploration de données, qui sont utilisées pour résoudre des problèmes complexes du monde réel comme ceux de [Djenouri et al., 2018,Djenouri and Zimek, 2018,Djenouri et al., 2017b], nous proposons trois stratégies de sélection de propriété pour résoudre ce processus d'alignement. Le but est de trouver des similitudes entre les jeux de données de manière efficace. L'idée principale de ce travail est de réduire les propriétés dimensionnelles élevées des données pour chaque jeu au stade du prétraitement, avant le processus de liage.

Notre système DMOM [Belhadi et al., 2020] est composé principalement de deux parties (voir la figure 5.1) :

La sélection des propriétés permet d'extraire un sous-ensemble optimal d'attributs (propriétés) qui représente le mieux le jeu de données. Cette étape est représentée comme un pré traitement pour être exécuté une seule fois. Pour cela, un dossier d'archives va être construit pour chaque jeu. Puis la sélection se fait à travers 3 stratégies différentes, exhaustive, statistique et basée sur les règles d'associations (FIM).

Le processus de liage ou d'alignement est appliqué entre les deux jeux de données en ne prenant en compte que les propriétés sélectionnées dans la partie précédente. Pour le processus de liage, notre modèle apprend à obtenir les meilleurs paramètres

d'alignement (IMAX, les seuils,..). Si le taux d'alignement dépasse le seuil donné, l'alignement du test est effectué et commencé.

Dans ce travail, on s'intéresse à la partie de sélection des propriétés, en proposant un modèle de sélection des propriétés les plus pertinentes en entrée,où n'importe quelle méthode existante peut être utilisée dans la partie d'alignement.

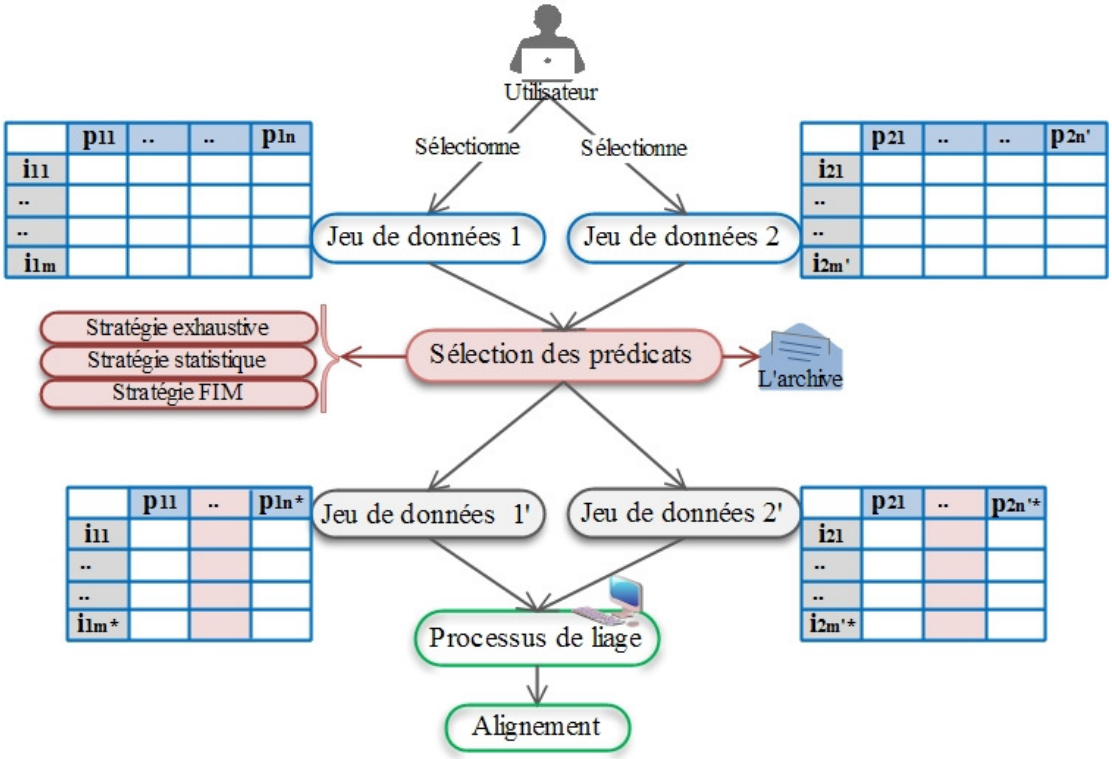


FIGURE 5.1 : Architecture du système DMOM.

Une description plus détaillée des composants de notre système sera présentée dans ce qui suit.

5.2.1 Sélection des propriétés

Afin de sélectionner les propriétés les plus pertinentes, nous proposons trois stratégies, à savoir la stratégie exhaustive, la stratégie statistique et la stratégie FIM. Les stratégies sont détaillées comme suit :

5.2.1.1 Stratégie exhaustive

Nous avons choisi la stratégie exhaustive comme étant une méthode basique pour la sélection des attributs pertinents. À partir d'un sous-ensemble de propriétés A, tous les sous-ensembles possibles APS sont extraits.

Il existe deux méthodes pour construire les sous-ensembles de **APS** : la sélection séquentielle croissante **SFS** (Sequential Forward Selection), et la sélection séquentielle arrière **SBS** (Sequential Backward Selection).

Dans SFS, un ensemble vide est d'abord construit, puis le premier niveau qui contient tous les ensembles possibles qui incluent un attribut de A sont formés. Ensuite, le deuxième niveau contenant tous les ensembles possibles qui incluent deux attributs de A sont construits et ainsi de suite jusqu'à ce que le dernier niveau contenant tous les attributs possibles de A sont obtenus.

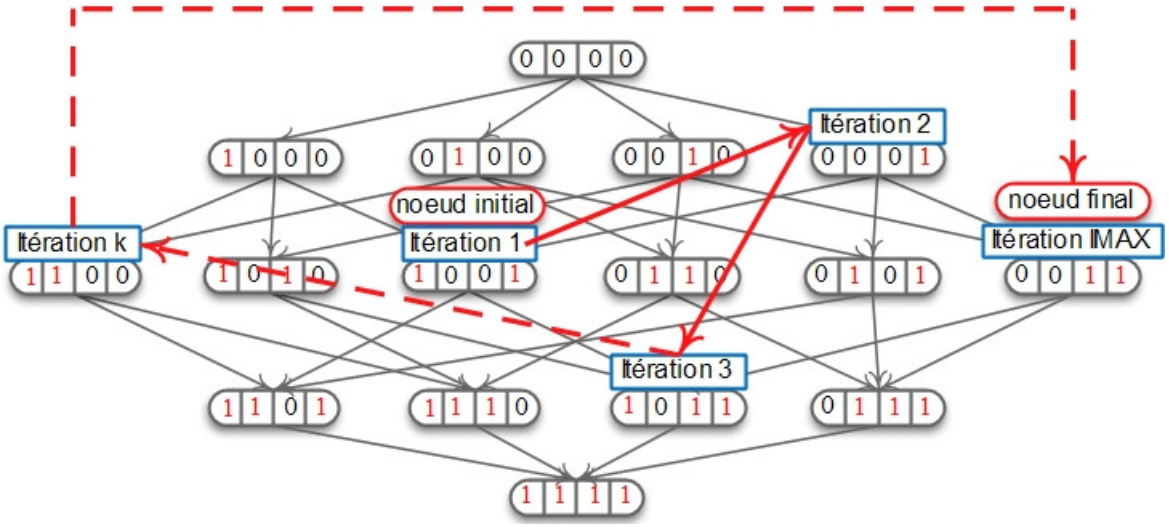
La seconde manière (SBS) est opposée au SFS car elle commence par l'ensemble complet. Le premier niveau contient l'ensemble qui inclut tous les attributs, le second niveau contient tous les ensembles possibles incluant tous les attributs à l'exception d'un attribut du niveau précédent, et ainsi de suite jusqu'à ce que le dernier niveau contenant tous les ensembles possibles qui incluent un seul attribut de A sont formés. Le SBS et le SFS ont la même complexité et le même niveau de difficulté dans l'implémentation.

Dans notre travail, nous explorons l'arbre de manière aléatoire car si l'ensemble A est grand, nous perdrons un certain nombre d'itérations dans la première couche ; à savoir, si nous utilisons les SFS, nous les perdrons dans la couche avec une seule propriété sélectionnée, et si nous utilisons les SBS, nous les perdrons dans la couche avec toutes les propriétés de données sauf une ; sinon la sélection aléatoire nous permet d'explorer le nœud maximum de voisinage non nécessaire (Voir Fig 5.2.(a)).

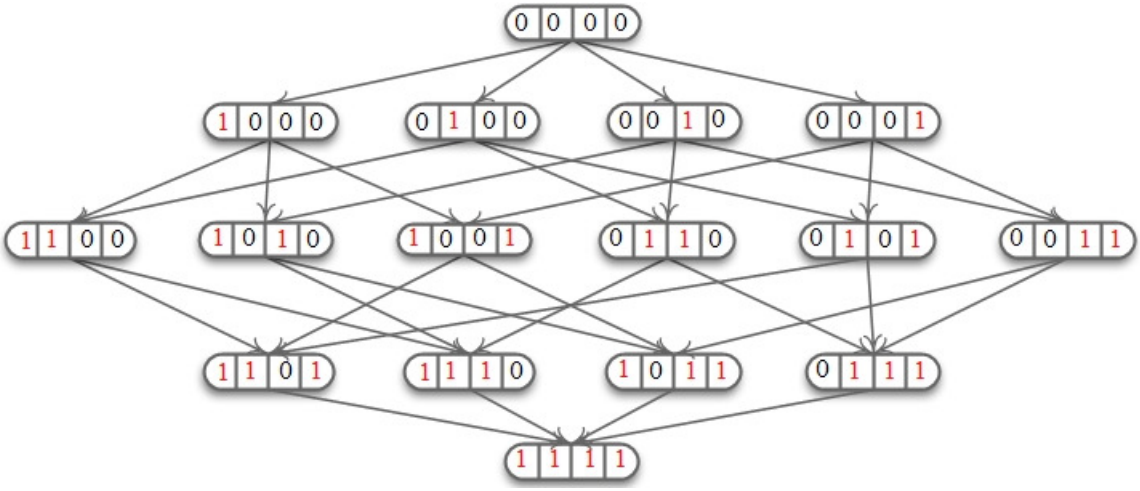
Exemple Nous allons appliquer la stratégie SFS sur l'ensemble A qui contient quatre attributs, et dont l'arborescence est présentée dans la figure 5.2.(b). Le $i^{ème}$ élément dans chaque nœud représente l'état du $i^{ème}$ attribut de A : 1 signifie que l'attribut est sélectionné et 0 signifie que l'attribut n'est pas sélectionné. Chaque nœud de l'arborescence est une liste de quatre cas (nombre d'attributs) qui représente l'état de chaque attribut. Les propriétés sélectionnées dans le nœud sont les observations contenant 1.

Le processus exhaustif commence par l'ensemble vide $\{0, 0, 0, 0\}$.
Le deuxième niveau contient un seul attribut du niveau précédent, ce qui signifie qu'il y a quatre ensembles possibles : $\{1, 0, 0, 0\}$ qui inclut le premier attribut,
$\{0, 1, 0, 0\}$ qui inclut le deuxième attribut,
$\{0, 0, 1, 0\}$ qui inclut le troisième attribut,
et $\{0, 0, 0, 1\}$ qui inclut le quatrième attribut.
Le deuxième niveau contient un attribut en plus et différent du niveau précédent.

Dans le processus de recherche, chaque fois qu'un attribut du niveau précédent est ajouté, le dernier niveau contient tous les attributs $\{1, 1, 1, 1\}$ de A.



(a) Notre stratégie exhaustive.



(b) Exploration de l'arborescence avec la stratégie SFS.

FIGURE 5.2 : Illustration de la recherche exhaustive.

5.2.1.2 Stratégie statistique

Cette stratégie étudie les caractéristiques statistiques des jeux de données pour sélectionner les meilleures propriétés qui sont ensuite utilisées dans le processus de liage. Deux valeurs statistiques sont dérivées de l'ensemble des instances pour sélectionner les propriétés de données appropriées. Le premier qui définit le nombre d'occurrences de la $i^{ème}$ propriété est étiqueté OP_i, et le second qui indique le nombre de toutes les valeurs possibles de la $i^{ème}$ propriété dans l'ensemble entier des instances est étiqueté

VP_i. Deux cas principaux peuvent être distingués : (i) la valeur de OP_i est petite ; dans ce cas, la $i^{ème}$ propriété peut ne pas être prise en compte car cette propriété ne représente pas l'ensemble des instances ; (ii) la valeur de OP_i est grande ; dans ce cas, la $i^{ème}$ propriété peut être considérée en fonction de la valeur VP_i.

En effet, la fonction le degré d'intérêt est utilisée entre OP_i et VP_i, qui calcule le rapport du nombre de toutes les valeurs possibles de la $i^{ème}$ propriété sur le nombre d'occurrences de la $i^{ème}$ propriété (VP_i/OP_i). Selon ce degré et en utilisant le seuil de degré d'intérêt μ, nous décidons si nous sélectionnons la propriété ou non.

Algorithm 7 Stratégie statistique

```
1:  Entrée : I : L'ensemble des instances.
2: OP : Liste du nombre d'occurrences d'un prédicat.
3: VP : Liste des valeurs possibles d'un prédicat.
4: LOP : L'ensemble de toutes les propriétés.
5: LVP : L'ensemble des valeurs possibles pour chaque prédicat.
6: μ : Le seuil degré d'intérêt.
7:  Sortie :  S : L'ensemble des propriétés sélectionnées.
8: LOP ← ∅, LVP ← ∅.
9: for each instance e ∈ I  do
10:    P ← SetProperties(e)
11:    for chaque élément p ∈ P  do
12:       if IDe ∉ LOP(p)  then
13:          AddLOP(p, IDe)
14:          if Value(e, p) ∉ LVP(p) then
15:             AddLVP(p, Value(e,p))
16:          end if
17:       end if
18:    end for
19: end for
20: S ← ∅
21: for chaque élément i ∈ LOP do
22:    OP_i ← SizeLOP(i)
23:    VP_i gets SizeLVP(i)
24:    if VP_i/OP_i > μ  then
25:       S ∪ i
26:    end if
27: end for
28: return  S
```

Dans l'algorithme 7, nous avons présenté la sélection des propriétés en utilisant la stratégie statistique. L'entrée se compose de l'ensemble des instances I, du seuil du degré d'intérêt μ, de l'ensemble de toutes les propriétés LOP et de l'ensemble des

valeurs possibles de chaque propriété LVP.

La première boucle de la ligne 9 à la ligne 19 dans l'algorithme vise à parcourir l'ensemble des instances I. La fonction $SetProperties(e)$ retourne alors P qui est l'ensemble des propriétés de l'instance e. Ensuite, pour chaque propriété $p \in P$ (de la ligne 11 à la ligne 18) l'algorithme vérifie si l'identifiant de l'instance e (IDe), appartient à l'ensemble de toutes les propriétés $LOP(p)$. Sinon, la fonction ($AddLOP(p, IDe)$) ajoute le IDe à l'ensemble de la propriété p. A savoir qu'à ce niveau, l'IDe de l'instance représente un identifiant pour trouver l'instance qui contient la propriété p. Ensuite, à la ligne 14, l'algorithme vérifie si la valeur de la propriété p de l'instance e appartient à la liste $LVP(p)$. Sinon la fonction ($AddLVP(p, Value(e, p))$) ajoute la valeur de la propriété p de l'instance e, $Value(e, p)$, à la valeur possible de la propriété p.

En d'autres termes, la première boucle consiste à analyser toutes les instances, où les propriétés de données de chaque instance sont extraites. La ligne 12 vise à vérifier si la propriété n'existe pas dans la liste de toutes les propriétés LOP. L'identifiant de l'instance contenant la propriété traitée est ensuite enregistré. La ligne 14, vérifie si la valeur de la propriété est également sauvegardée, sinon la valeur sera sauvegardée dans la valeur possible de la propriété (LVP). De la ligne 21 à la ligne 27, l'algorithme parcourt l'ensemble des propriétés de données LOP et calcule OP_i qui représente la taille de l'ensemble des ID des instances de la $i^{ème}$ propriété et VP_i qui représente le nombre de valeurs possibles de la $i^{ème}$ propriété. Après cela, à la ligne 24, le degré d'intérêt de VP_i sur OP_i est déterminé et comparé au seuil μ ; s'il est supérieur à μ, alors la $i^{ème}$ propriété sera ajoutée à l'ensemble des propriétés de données sélectionnées S.

5.2.1.3 Stratégie FIM

Cette stratégie étudie la corrélation entre les propriétés des jeux de données afin d'en sélectionner les meilleures pour le processus de liage. Il est inspiré de l'extraction des items set fréquent (FIM) [Agrawal et al., 1993] qui est utilisé pour extraire les propriétés les plus pertinentes qui couvrent le nombre maximum d'instances possibles. FIM désigne l'extraction des itemsets pertinents qui satisfont un minimum support ($minsup$) de transactions dans le jeu de données. Nous adoptons un processus FIM classique au problème de liage de données en développant les trois étapes (fouille, réduction et sélection) pour récupérer efficacement les propriétés pertinentes des jeux de données. Une différence clé entre les stratégies FIM précédentes et notre approche basée sur la FIM réside dans le processus de réduction. Les stratégies FIM existantes

énumèrent tous les modèles qui dépassent les contraintes de support minimales, l'approche prend en compte une autre mesure pour élaguer l'espace de recherche en sélectionnant un sous-ensemble de modèles pertinents qui couvre un maximum de transactions dans le jeu de données (des instances dans notre cas). Le pseudo-code de cette stratégie est donné dans l'algorithme 8. Une explication détaillée des trois étapes de la stratégie FIM est donnée ci-dessous.

Processus de fouille Récemment, le SSFIM (Single Scan for Frequent Itemsets Mining) [Djenouri et al., 2017a] a été proposé pour extraire les itemsets fréquents en utilisant un seul passage, et il s'est avéré insensible à la valeur du support minimum. L'étude expérimentale rapportée dans [Djenouri et al., 2017a] révèle que le SSFIM surpasse les algorithmes FIM. Par conséquent, dans ce travail, la SSFIM est adoptée pour extraire les littéraux fréquents (étiquetés comme S) de l'ensemble des instances I. Le SSFIM est réalisé en deux étapes principales : la génération et l'extraction. Dans l'étape de génération, à partir de la première instance I_1, nous référons $Itemset(I_1)$ par l'ensemble de toutes les combinaisons possibles des littéraux de cette instance. Le résultat est ajouté à la table de hachage H en créant une entrée pour chaque ensemble d'éléments dans $Itemset(I_1)$. La fréquence de chaque jeu d'éléments est initialisée par 1 dans la table de hachage H. Ensuite, les itemsets de la seconde instance I_2 sont générés pour chaque itemset dans $Itemset(I_2)$; si cet ensemble d'éléments existe dans H, alors, sa fréquence est incrémentée de un. Sinon, une nouvelle entrée est créée avec la fréquence fixée à 1. Ce processus est répété jusqu'à ce que toutes les instances I soient traitées. La deuxième étape vise à extraire les itemsets fréquents, littéraux fréquents dans notre cas, de la table de hachage H. Pour cela, le support de chaque jeu d'éléments t est calculé (voir Eq 5.1) ; si le support de t est supérieur à $minsup$, alors t est appelé littéral fréquent et il est ajouté à l'ensemble des littéraux fréquents S.

$$Support(t) = \frac{h(t) * freq}{|I|} \tag{5.1}$$

Un seuil μ défini entre $[0, 1]$, est utilisé pour sélectionner les propriétés appropriées. En effet, pour chaque prédicat, si sa probabilité est supérieure à μ alors elle est ajoutée à la liste SP.

Processus de réduction Un inconvénient de l'extraction des itemsets fréquents est que nous pouvons générer un grand nombre d'itemsets. Cela est particulièrement vrai pour les jeux de données générés à partir de grands jeux de données. L'analyse d'un grand nombre d'ensembles d'éléments prend du temps et est peu pratique pour

Algorithm 8 Stratégie FIM

Entrée : I : Ensemble d'instances.
m : La taille des instances.
P : L'ensemble de toutes les propriétés.
S : L'ensemble de tous les littéraux fréquents.
S^* : L'ensemble des littéraux fréquents réduits.
σ : Seuil du support minimum.
IMAX : Nombre d'itération maximum du processus de réduction.
μ : Seuil du degré d'intérêt.
Sortie : SP : L'ensemble des propriétées sélectionnés.
********** **Processus de fouille** **********
for chaque instance e ∈ I **do**
  $F_e \leftarrow$ Itemset(e)
  for chaque élément i ∈ F_e **do**
    if i ∈ H **then**
      $Freq_i$++
    else
      AddH(i,1)
    end if
  end for
end for
S ← ∅
for chaque élément h ∈ H **do**
  if Support(h) > σ **then**
    S ∪ h
  end if
end for
********** **Processus de réduction** **********
sol ← InitialSol(S)
$S^* \leftarrow$ S
iter ← 0.
while $Pruning_{max}$(S)< m and iter < IMAX **do**
  neighbors ← ComputeNeighbors(sol).
  best ← BestNeighbors(neighbors).
  if $Pruning_{max}$(best)> S^* **then**
    $S^* \leftarrow$ best.
  end if
  iter ← iter + 1.
end while
********** **Processus de sélection** **********
SP ← ∅
for chaque propriété i ∈ P **do**
  if Probability(i, S) > μ **then**
    SP ∪ i
  end if
end for
return SP

l'utilisateur. Pour résoudre ce problème, nous avons proposé une nouvelle stratégie pour filtrer les ensembles d'itemsets trouvés dans l'étape de fouille. Le but est de montrer à l'utilisateur un petit ensemble d'éléments représentatifs qui décrit bien le jeu de données. Une stratégie d'élagage est conçue sur la base d'un nouveau concept de couverture pour ne conserver que les plus petits ensembles d'éléments qui recouvrent le plus grand nombre d'instances. On suppose que ces ensembles d'éléments sont plus utiles pour l'utilisateur, conformément au principe de longueur minimale de description [Barron et al., 1998], car ces modèles sont les plus petits qui expliquent la plus grande quantité de données. L'application de la stratégie proposée peut réduire considérablement le nombre d'éléments fréquents. Il est à noter que la proposition faite est différente des autres études sur l'extraction fréquente d'ensembles d'éléments qui réduisent le nombre d'items en découvrant des ensembles d'items qui sont maximaux [Gouda and Zaki, 2001] et fermés [Pei et al., 2000]. La différence dans ces études se concentrent sur la recherche d'itemsets qui sont respectivement maximaux par rapport à un jeu de données ou à des ensembles de transactions, plutôt que de trouver des ensembles d'éléments minimaux. La proposition est également différente des études sur les ensembles d'éléments de générateur car ces études trouvent des ensembles d'éléments minimaux par rapport à des ensembles de transactions qui ne couvrent pas nécessairement de nombreuses instances d'un jeu de données.

Definition 5.2.1. La couverture Soit $S = \{S_1, S_2, ..., S_r\}$ l'ensemble de tous les itemsets fréquents découvert par le processus de fouille de données. Nous définissions la *couverture du problème de réduction* en maximisant $Réduction_{max}$, avec :

$$\begin{array}{rcccc} Réduction_{max} & : & S & \rightarrow & \Re \\ & & S' & \mapsto & Réduction_{max}(S') \end{array}$$

Definition 5.2.2. Nous définissons une $Réduction_{max}$ comme une fonction qui couvre le maximum de nombre d'instances du jeu de données. Formellement, $\mathcal{I}(S_i)$ représente l'ensemble des instances couvert par l'itemset S_i. La fonction de couverture du processus de réduction vise à retrouver un sous-ensemble $S' \subset S$ qui maximise la valeur de couverture, définie par :

$$\begin{array}{rcccc} Réduction_{max} & : & S & \rightarrow & \Re^{+} \\ & & S' & \mapsto & |\bigcup_{S_i \in S'} \mathcal{I}(S_i)| \end{array}$$

Definition 5.2.3. La solution optimale pour le problème de réduction dans un jeu de données de m instances, est un sous-ensemble minimal $S^* \subset S$, tel que S^* est un sous-ensemble qui couvre toutes les instances. Nous définissions S^* par :

$$\begin{cases} Pruning_{max}(S^*) = m \\ \forall S' \subset S, \\ Pruning_{max}(S') = Pruning_{max}(S^*) \Rightarrow |S'| \geq |S^*| \end{cases}$$

Trouver un sous-ensemble optimal qui satisfait les contraintes de la fonction de réduction est un problème NP-complet, puisque pour l'ensemble des itemsets fréquents S, il existe 2^r sous-ensembles possibles de S à choisir. Par conséquent, une recherche exhaustive prendrait beaucoup de temps, voire elle est peu pratique si la cardinalité de S est grande. Pour faire face à cela, une combinaison entre la méthode de recherche gourmande et la recherche du voisinage est développée, pour réduire l'espace de recherche et trouver une bonne solution plutôt qu'une solution globalement optimale.

Processus de sélection Basé sur des littéraux fréquents S découverts et réduits ci-dessus, l'ensemble pertinent SP est sélectionné. La probabilité $P(i, S)$ désigne la probabilité d'apparition de la $i^{ème}$ propriété dans l'ensemble des littéraux fréquents S. Un seuil μ compris entre [0,1], est utilisé pour sélectionner les propriétés de données appropriées. En effet, pour chaque propriété, si sa valeur de probabilité est supérieure à μ alors, elle est ajoutée au sous-ensemble SP.

Definition 5.2.4. Sélection d'une propriété On considère une propriété p et un ensemble de littéraux fréquents S dérivés depuis le processus de réduction. p est sélectionné pour le processus de liage si et seulement si :

$$P(p, S) > \mu \tag{5.2}$$

Où, $P(p, S)$ est la probabilité de l'apparition de la propriété p parmi les fréquents littéraux S et μ est un seuil de degré d'intérêt.

5.2.2 Processus de liage

Après la sélection des propriétés appropriées, il est temps de comparer les instances du jeu de données de base avec un autre jeu de données. Dans cette partie, on considère deux ensembles de données, un jeu de base $\mathcal{BD}$ qui va être liée avec un second jeu de données $\mathcal{D}$. $< P, I >$ est l'ensemble des propriétés P et les entités (instances) I du jeu de données de base. $< P', I' >$ est l'ensemble des propriétés P' et les entités (instances) I' du second jeu de données. P et P' sont les ensembles résultant de la sélection des propriétés décrites avant. Le liage itératif consiste à parcourir toutes les entités I du jeu de données de base, et de les comparer avec toutes les entités I'

Algorithm 9 Processus de liage

Entrée : I : ensemble d'instances de $\mathcal{BD}$
I' : ensemble d'instances de $\mathcal{D}$
P : ensemble des propriété sélectionner de $\mathcal{BD}$
P' : ensemble des propriété sélectionner de $\mathcal{D}$
Sortie : AL : L'ensemble d'alignement (le résultat du liage)
for chaque instance i ∈ I **do**
  P ← SetProperties(i)
  for chaque instance j ∈ I' **do**
    P' ← SetProperties(j)
    L ← ∅
    for chaque propriété p ∈ P **do**
      for chaque propriété p' ∈ P' **do**
        if Value(p,p') **then**
          L ∪ {p,p'}
        end if
      end for
    end for
    if L !=∅ **then**
      AL ∪ ($\{ID_i$,$ID_j\}$∪L)
    end if
  end for
end for
return AL

du second jeu de données. La comparaison entre les entités s'effectue en comparant chaque littéral de la $i^{ème}$ entité de $\mathcal{BD}$ avec tous les littéraux de la $j^{ème}$ entité de $\mathcal{D}$.

L'algorithme 9 parcourt toutes les instances I de $\mathcal{BD}$ (dans la boucle de la ligne 6 à la ligne 22), et la seconde boucle (de la ligne 8 à la ligne 21) parcourt les instances I' de $\mathcal{D}$. La fonction $SetProperties()$ récupère les propriétés de la $i^{ème}$ et la $j^{ème}$ instances respectivement. Ensuite, depuis la ligne 13 à la ligne 15, l'algorithme compare la valeur des propriétés p et p' de la $i^{ème}$ et la $j^{ème}$ instances. Si elles sont identiques, elles seront ajoutées dans L. Enfin, si l'ensemble L n'est pas vide, alors la $i^{ème}$ et la $j^{ème}$ instances sont ajoutées à l'ensemble d'alignement AL.

5.3 Illustration de DMOM par l'exemple

	P_1	P_2	P_3	P_4	P_5	P_6
i_1	A_1	B_1	C_1	D_2	E_3	H_1
i_2	A_3	B_2	C_1	–	–	H_3
i_3	A_3	–	C_1	D_1	–	H_2
i_4	A_2	B_3	–	D_4	–	H_4
i_5	A_1	B_4	C_2	D_2	E_3	H_4
i_6	A_2	B_5	C_1	–	E_2	H_3
i_7	A_1	–	C_2	D_2	E_3	H_1
i_8	A_3	B_6	C_2	–	–	H_2

TABLE 5.1 : Jeu de données I

	P'_1	P'_2	P'_3	P'_4	P'_5
i'_1	D_2	F_1	E_3	–	G_1
i'_2	D_1	F_2	E_2	B_7	–
i'_3	D_2	–	E_3	B_1	G_3
i'_4	D_4	F_3	E_4	B_3	G_4
i'_5	D_2	–	E_3	B_4	G_3
i'_6	–	–	E_2	B_8	G_2
i'_7	–	F_2	E_2	B_6	G_2

TABLE 5.2 : Jeu de données II

Dans cette partie, nous allons présenter un exemple illustratif. Considérons deux ensembles de données (voir Tables 5.1, et 5.2) qui représentent deux jeux de données différents. Les lignes représentent l'ensemble des instances $\{i_1, \ldots i_8\}$ du premier jeu de données et l'ensemble des instances $\{i'_1, \ldots i'_7\}$ du deuxième jeu de données. Les colonnes identifient les propriétés $\{p_1, \ldots p_6\}$ du premier jeu de données et les propriétés $\{p'_1, \ldots p'_5\}$ du deuxième jeu de données, respectivement. Les lettres majuscules font référence aux valeurs des propriétés. L'application du système DMOM utilisant les trois stratégies de sélection des propriétés pertinentes présentées dans la section 5.2.1 est donnée comme suit.

1. **Sélection des propriétés**

   - **Stratégie exhaustive**

     Avec cette stratégie, la sélection des propriétés pertinentes est aléatoire, en utilisant des valeurs 0/1. Ce processus est répété $IMAX$ fois, et la meilleure

configuration est enregistrée en parallèle. Le premier jeu de données (resp, $2^{ième}$ jeu de donnée), est décrit avec six différentes propriétés (resp, cinq différentes propriétés), l'arbre de recherche contient 2^6 nœuds possibles (resp, 2^5 dans le deuxième jeu de données). Supposons que $IMAX = 4$, la génération du premier jeu démarre d'une façon aléatoire en définissant des valeurs booléennes sur les propriétés. Supposons la solution initiale est $(0, 0, 1, 1, 0, 0)$, ce qui signifie que la troisième et la quatrième propriétés sont sélectionnées. Ensuite, nous évaluons cette solution en exécutant le processus de liage, et nous enregistrons le score. En utilisant cette solution, nous générons d'autres solutions pour la deuxième, troisième et quatrième itérations (puisque $IMAX = 4$). Pour chaque itération, nous sauvegardons le score. Et à la fin, la meilleure correspondance avec les meilleures propriétés est enregistrée.

- **Stratégie statistique**

  Nous calculons OP_i qui représente le nombre d'occurrences de la $i^{ème}$ propriété et VP_i qui représente le nombre de toutes les valeurs possibles de la $i^{ème}$ propriété. Dans la Table 5.3, les deux paramètres des deux jeux de données sont résumés. Nous définissions la fonction d'intérêt I qui représente la division entre VP_i et OP_i. Le seuil du degré d'intérêt $0.5 < \mu < 0.7$. Comme nous pouvons le voir dans le tableau 5.3, l'ensemble des propriétés pour le premier jeu et le deuxième qui seront sélectionnées sont : $\{P_4, P_5, P_6\}$ et $\{P'_1, P'_2, P'_5\}$, respectivement, puisqu'ils satisfont le seuil du degré d'intérêt.

- **Stratégie FIM**

  Avant l'application du FIM, nous supposons que le minimum support est à 25 % pour les deux jeux de données. Le processus de génération des itemsets est exécuté instance par instance, jusqu'à ce que toutes les instances soient parcourues. Les itemsets (l'ensemble des propriétés avec leurs valeurs) qui satisfont à la contrainte de prise en charge minimale sont enregistrés, mais seules les propriétés appartenant à ces itemsets sont prises en compte. Dans Table 5.4, tous les itemsets qui satisfont aux contraintes minimales de support pour les deux jeux de données sont présentés. On peut remarquer que l'ensemble des propriétés sélectionnées pour le premier jeu et le second sont respectivement $\{P_1, P_4, P_5\}$ et $\{P'_1, P'_3\}$.

Data		OP	VP	I
$\mathcal{DI}$	P_1	8	3	0.37
	P_2	6	6	1
	P_3	7	2	0.28
	P_4	5	3	0.6
	P_5	4	2	0.5
	P_6	8	4	0.5
$\mathcal{DII}$	P'_1	5	3	0.6
	P'_2	4	3	0.75
	P'_3	7	3	0.42
	P'_4	6	6	1
	P'_5	6	4	0.66

TABLE 5.3 : Stratégie statique.

Data	$Item_1$	$Supt(\%)$
$\mathcal{DI}$	$\{(P_1, A_1)\}$	37
	$\{(P_1, A_3)\}$	37
	$\{(P_3, C_1)\}$	50
	$\{(P_3, C_2)\}$	37
	$\{(P_4, D_2)\}$	37
	$\{(P_5, E_3)\}$	37
	$\{(P_1, A_1), (P_4, D_2)\}$	37
	$\{(P_1, A_1), (P_5, E_3)\}$	37
	$\{(P_1, A_1), (P_4, D_2), (P_5, E_3)\}$	37
$\mathcal{DII}$	(P'_1, D_2)	43
	(P'_3, E_2)	43
	(P'_3, E_3)	43
	$(P'_1, D_2), (P'_3, E_3)$	43

TABLE 5.4 : Stratégie FIM.

2. **Processus de liage** Dans cette étape, nous comparons toutes les instances du premier jeu avec chaque instance du second jeu en tenant compte des propriétés sélectionnées dans la partie sélection de fonctionnalités. Dans ce cas, on fait correspondre :

   - $\{P_1, .., P_6\}$ avec $\{P'_1, ..., P'_5\}$ dans le cas exhaustif. Toutes les valeurs sont sélectionnées, mais A_1 à A_3, C_1, C_2 et H_1 à H_4 n'existent pas dans le deuxième jeu. Pour faciliter la lecture des similarités, nous comparerons l'ensemble des similitudes à un ensemble de propriétés. Les similitudes sont les suivantes :
     - $\{i_1, i_5, i_7\} \approx \{i'_1, i'_3, i'_5\}$ par rapport à la valeur D_2 and E_3.
     - $\{i_4\} \approx \{i'_4\}$ par rapport à la valeur D_4 and B_2.
     - $\{i_6\} \approx \{i'_2, i'_6, i'_7\}$ par rapport à la valeur E_2.
     - $\{i_3\} \approx \{i'_2\}$ par rapport à la valeur D_1.
     - $\{i_1\} \approx \{i'_3\}$ par rapport à la valeur B_1.
     - $\{i_5\} \approx \{i'_5\}$ par rapport à la valeur B_4.
     - $\{i_8\} \approx \{i'_7\}$ par rapport à la valeur B_6.
   - $\{P_4$,P_5,$P_6\}$ avec $\{P'_1$, P'_2,$P'_5\}$ dans le cas statistique. Les similitudes sont les suivantes :
     - $\{i_1, i_5, i_7\} \approx \{i'_1, i'_3, i'_5\}$ par rapport à la valeur D_2 et E_3.
     - $\{i_6\} \approx \{i'_2, i'_6, i'_7\}$ par rapport à la valeur de E_2.

  - $\{i_3\} \approx \{i'_2\}$ par rapport à la valeur de D_1.
  - $\{i_4\} \approx \{i'_4\}$par rapport à la valeur de D_4.

- $\{P_1,P_4,P_5\}$ with $\{P'_1,P'_3\}$ dans le cas de FIM. Les similitudes sont les suivantes :
  - $\{i_1, i_5, i_7\} \approx \{i'_1, i'_3, i'_5\}$ par rapport à la valeur de D_2 et E_3.
  - $\{i_6\} \approx \{i'_2, i'_6, i'_7\}$ par rapport à la valeur de E_2.
  - $\{i_3\} \approx \{i'_2\}$ par rapport à la valeur de D_1.
  - $\{i_4\} \approx \{i'_4\}$ par rapport à la valeur de D_4.

5.4 Études expérimentales

Pour valider notre système $DMOM$, des expériences approfondies ont été menées à l'aide de jeux de données des ensemble de données bien connues. Tous les algorithmes ont été implémentés dans le langage de programmation Java et les expériences ont été exécutées sur une machine de bureau équipée d'un processeur Intel $I7$ et d'une mémoire de $16Go$.

5.4.1 Description des données

Deux ensembles de bases de données bien connus qui sont souvent utilisés par la communauté concernée par notre recherche pour faire l'objet d'expérimentation :

1. Les bases de données OAEI (Ontology Alignment Evaluation Initiative), extraites de "http ://oaei.ontologymatching.org". La table 5.5 répertorie le nombre d'instances et les propriétés de ces bases de données.

2. Le DBpedia [1]. Il s'agit d'un hub de données qui peut être trouvé sur Wikipedia. Cette base de données contient 4,233,000 instances et 2,795 propriétés différentes.

Tout d'abord, les paramètres des stratégies de sélection des propriétés proposées ont été définis. Par la suite, deux types de tests ont été réalisés : (i) en utilisant les bases de données OAEI, avec les trois stratégies proposées (la stratégie exhaustive, la stratégie statistique et la stratégie FIM). Une étude comparative a été réalisée pour sélectionner la meilleure stratégie. Ensuite, (ii) la meilleure stratégie a été comparée aux algorithmes de liage des données en utilisant la base de données DBpedia.

[1] http ://wiki.dbpedia.org/Datasets

Nom de la base	Nombre d'instances	Nombre de propriétés
OntoA_dis	29645	11
OntoB_dis	15556	11
OntoA_rec	15556	11
OntoB_dis	1708	11
Onto_a_id	1330	5
Onto_b_id	2649	4
Onto_a_sim	173	5
Onto_b_sim	172	5
*onto*101	57	46
*onto*104	56	46
*onto*202	57	46
*onto*230	47	49
$IIMB000$	12333	13
$IIMB104$	12338	13
*person*11	500	14
*person*12	500	12
*person*21	600	14
*person*22	400	12

TABLE 5.5 : Description des jeux de données de OAEI.

Dans ce travail, le modèle de K-validation est utilisé, où à chaque itération d'algorithme, le liage de traitement et de test sont effectuées. Dans le processus d'appariement de traitement, le modèle proposé apprend à ajuster au mieux les paramètres. Si le taux de correspondances dépasse un seuil donné, alors le test de correspondances est lancé.

5.4.2 Expérimentations

Des tests approfondis ont été effectués pour ajuster empiriquement les paramètres des stratégies proposées en utilisant la base de données *Person* disponible sur l'OAEI. Cette base de données se compose de deux jeux de données Person1 et Person2, où l'instance d'origine a été modifiée en ajoutant les valeurs de propriétés pour générer l'autre instance. Dans Person1, la différence entre les deux instances était une modification ; tandis que dans Person2, la différence maximale entre les deux instances était de 3 modifications, et la différence maximale par rapport à l'instance d'origine était de 10 doublons. De plus, Person1 contenait 500 instances dans les deux fichiers (Person11 et Person12) avec 400 instances dans l'alignement de référence, tandis que Person2 contenait 600 instances dans Person21 et 400 instances dans Person22, avec

400 instances dans l'alignement de référence. A chaque test, la valeur de $F\_measure$ a été déterminée. Les résultats de ces stratégies sont les suivants :

1. **Exhaustive** : En faisant varier le nombre maximum d'itérations de 1 à 100, la valeur optimale était de 20.
2. **Statistique** : En faisant varier la valeur du seuil du degré d'intérêt de 0 à 1, la valeur optimale était de $0,7$.
3. **FIM** : En faisant varier le seuil de support minimum de 1% à 100%, la valeur optimale était de 30%.

Évolutivité du DMOM

Data	Exhaustive			Statistique			FIM		
	Fmeas.	CPU Temps	Mem.	Fmeas.	CPU Temps	Mem.	Fmeas.	CPU Temps	Mem.
Dis_1	0.76	3.25	115	0.84	**2.12**	**95**	**0.93**	2.61	102
Dis_2	0.77	4.26	158	0.88	**3.91**	**112**	**0.95**	3.99	115
Rec_1	0.79	5.21	136	0.93	**4.96**	**118**	**0.97**	4.99	119
Rec_2	0.75	7.12	159	0.91	**6.02**	**124**	**0.96**	6.15	126
ID_1	0.79	6.15	171	0.92	6.36	**105**	**0.98**	**6.21**	101
ID_2	0.80	10.23	166	0.91	9.12	**99**	**0.95**	**9.02**	103
Sim_1	0.74	12.98	147	0.91	10.12	**113**	**0.95**	**9.85**	117
Sim_2	0.72	14.13	151	0.90	**10.18**	**88**	**0.94**	11.02	92
O_{101}	0.74	11.02	110	0.92	11.00	101	**0.93**	**10.02**	**97**
O_{104}	0.73	14.15	187	**0.94**	**12.25**	102	0.92	13.02	**95**
O_{202}	0.81	18.69	**84**	0.92	17.65	111	**0.93**	**14.23**	88
O_{230}	0.82	19.36	82	**0.95**	21.02	78	0.92	**18.26**	**67**
B_{000}	0.89	**8.26**	**83**	0.95	9.26	87	**0.95**	10.29	93
B_{104}	0.81	12.25	112	**0.96**	**4.12**	92	0.95	5.26	**89**
P_{11}	0.83	5.26	129	0.97	4.26	**81**	**1.0**	**3.77**	85
P_{12}	0.85	12.25	127	0.97	10.25	83	**1.0**	**9.36**	**81**
P_{21}	0.84	11.29	124	0.98	12.03	80	**1.0**	**8.76**	**78**
P_{22}	0.87	15.23	119	0.99	**12.36**	**75**	**1.0**	13.62	77

TABLE 5.6 : Comparaison de F_mesure, du temps CPU (sec) et de l'utilisation de la mémoire (Mo) des trois stratégies (la stratégie exhaustive, la stratégie statistique et la stratégie FIM).

Ces expériences ont été réalisées pour tester l'évolutivité du système DMOM. Pour cela, la qualité des solutions, les performances d'exécution et l'utilisation de la mémoire ont été déterminées de la même manière que lors de la campagne OAEI. On

fait remarquer que l'utilisation de la mémoire a été mesurée à l'aide de l'API Java standard. Le tableau 5.6 présente comment Fmeasure, le temps CPU (en secondes) et l'utilisation de la mémoire (en méga octets) du DMOM ont été modifiés avec diffèrents jeux de données et la stratégie appliquée. Dans le tableau 5.6, on peut remarquer que la stratégie FIM a surpassé les deux autres stratégies concernant Fmeasure dans 15 cas sur 18 cas. De plus, la qualité de la FIM était de 92% dans tous les cas, tandis que celles des stratégies statistique et exhaustive étaient inférieures respectivement à 84% et 72%. Les résultats ont été obtenus en utilisant les connaissances extraites par la stratégie FIM qui ont permis une meilleure réduction de l'espace de dimensionnalité des jeux de données .

Les résultats ont également révélé que la stratégie statistique et la stratégie FIM convergeaient vers les mêmes valeurs concernant l'utilisation de la mémoire et les performances d'exécution. Cependant, la méthode exhaustive a obtenu les pires résultats en termes de ces deux mesures, ce qui peut s'expliquer par le fait que la stratégie exhaustive a énuméré toutes les combinaisons possibles sans booster le processus de recherche. Par contre, les deux autres stratégies ont stimulé l'exploration de l'espace des solutions en utilisant les informations statistiques dans la stratégie statistique et les modèles fréquents dans la stratégie FIM. A partir de là, la stratégie FIM a été choisie pour les expériences restantes.

5.4.3 Comparaison de DMOM avec des système de l'état de l'art

L'expérience suivante a été réalisée pour comparer notre système DMOM en utilisant la stratégie FIM, avec les algorithmes d'état de l'art (EIFPS [Niu et al., 2012], et le RiMOM [Shao et al., 2016]) sur la base DBPedia. RIMOM est basé sur la réduction du nombre d'instances candidates, tandis que EIFPS est basé sur les algorithmes d'apprentissage, par l'exploration fréquente de modèles dans le processus de liage. Les deux algorithmes montrent leur supériorité par rapport aux algorithmes de correspondance existants [Mohammadi et al., 2019].

La figure 5.3, montre le temps d'exécution des trois approches au pourcentage de propriétés de 20% à 100%, en considérant toutes les instances. Lorsque le nombre d'appariements varie de 100 à $1,000,000$, DMOM a surpassé les deux autres approches. De plus, le temps d'exécution de DMOM s'est stabilisé à un nombre élevé de propriétés de données, là où les deux autres approches prenaient beaucoup de temps pour un grand nombre d'instances et un grand nombre de correspondances. Ainsi, les

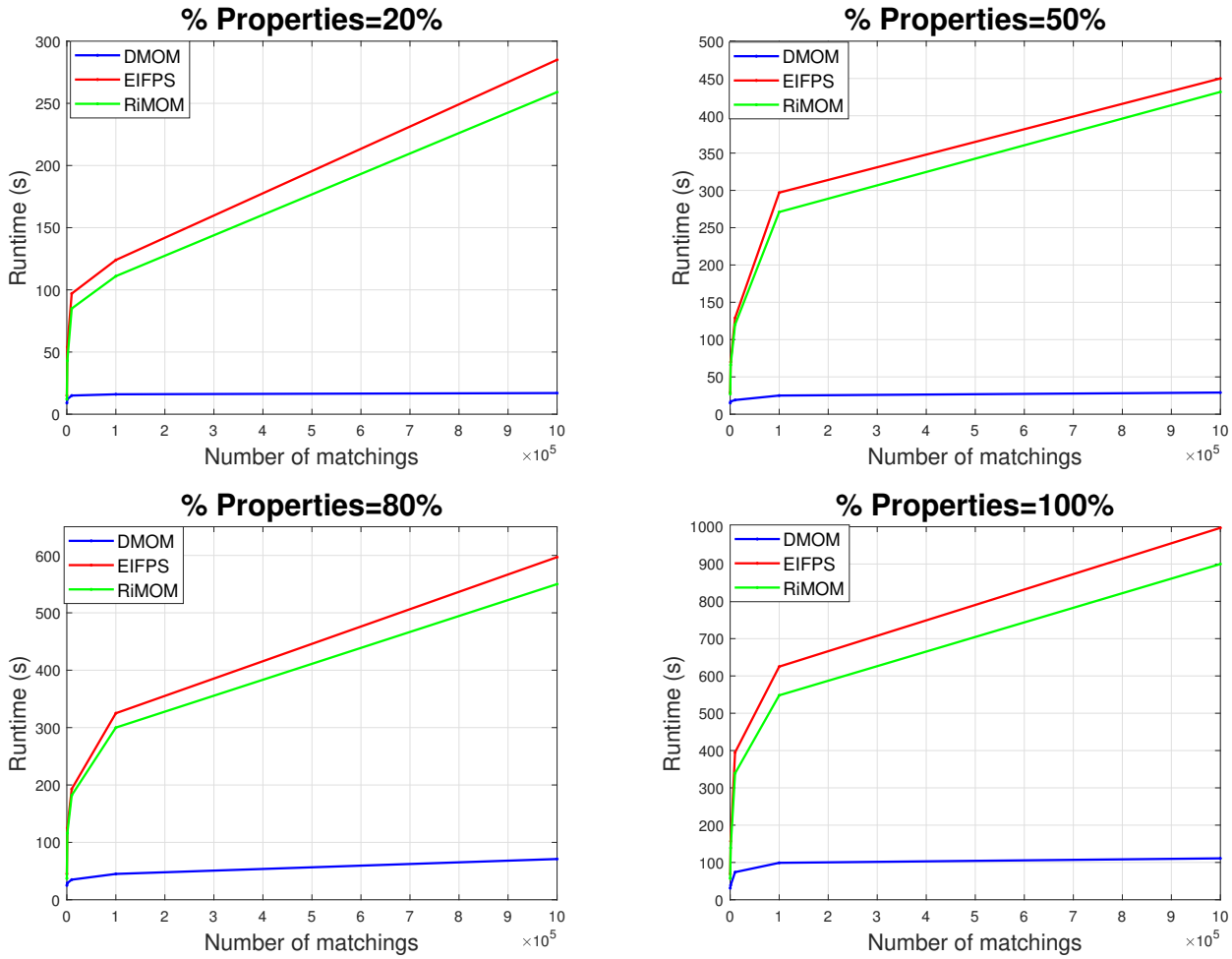


FIGURE 5.3 : Comparaison des performances d'exécution (en secondes) du DMOM, de l'EIFPS et du RiMOM à l'aide de DBpedia en faisant varier le pourcentage des propriétés de données (% P) de 20% à 100%.

deux approches (l'EIFPS et le RiMOM) ont nécessité plus de 900 secondes pour traiter 1,000,000 de correspondances dans l'ensemble de données DBpedia, alors que cela n'a pris que 111 secondes pour le DMOM pour la même tâche. Ces résultats ont été obtenus grâce à l'étape de prétraitement, où seules les propriétés les plus pertinentes ont été sélectionnées en utilisant l'approche d'exploration fréquente d'itemset. En effet, le processus d'appariement dans notre approche n'utilise que les propriétés pertinentes dans les deux jeux de données, alors que les deux autres approches considèrent l'ensemble des propriétés des deux jeux de données.

% I	%P	DMOM			EIFPS			RiMOM		
		Rec.	Prec.	Fmeas.	Rec.	Prec.	Fmeas.	Rec.	Prec.	Fmeas.
20	20	0.97	0.95	0.96	0.97	0.94	0.95	**0.98**	**0.95**	**0.96**
	50	**0.97**	**0.95**	**0.96**	0.92	0.92	0.92	0.95	0.93	0.94
	80	**0.97**	**0.95**	**0.96**	0.90	0.91	0.90	0.93	0.92	0.92
	100	**0.97**	**0.95**	**0.96**	0.88	0.90	0.89	0.89	0.90	0.89
	20	**0.96**	**0.94**	**0.95**	0.93	0.92	0.92	0.95	0.92	0.93
	50	**0.96**	**0.94**	**0.95**	0.89	0.87	0.88	0.91	0.89	0.90
	80	**0.96**	**0.94**	**0.95**	0.87	0.84	0.85	0.89	0.86	0.87
	100	**0.96**	**0.94**	**0.95**	0.85	0.82	0.83	0.87	0.83	0.85
80	20	**0.95**	**0.92**	**0.93**	0.90	0.89	0.89	0.91	0.90	0.90
	50	**0.95**	**0.92**	**0.93**	0.88	0.86	0.87	0.89	0.88	0.88
	80	**0.95**	**0.92**	**0.93**	0.82	0.80	0.81	0.83	0.81	0.82
	100	**0.95**	**0.92**	**0.93**	0.78	0.75	0.76	0.80	0.79	0.79
100	20	**0.94**	**0.90**	**0.92**	0.85	0.82	0.83	0.87	0.86	0.86
	50	**0.94**	**0.90**	**0.92**	0.83	0.81	0.82	0.84	0.82	0.83
	80	**0.94**	**0.90**	**0.92**	0.78	0.75	0.76	0.80	0.77	0.78
	100	**0.94**	**0.90**	**0.92**	0.74	0.70	0.72	0.73	0.72	0.72

TABLE 5.7 : Comparaison du Rappel, de la Précision et de la Fmesure du DMOM, de l'EIFPS et du RiMOM à l'aide de DBpedia en faisant varier à la fois le pourcentage d'instances (% I) et le pourcentage des propriétés de données (% P) de 20% à 100%.

La dernière expérience (Table 5.7) a été réalisée sur la base DBpedia pour comparer la qualité de l'appariement du système DMOM avec les algorithmes EIFPS et le RiMOM. En faisant varier le pourcentage de propriétés des données et le pourcentage d'instances de 20% à 100%, DMOM a surpassé les deux autres algorithmes en ce qui concerne la qualité (rappel, précision et F_mesure) dans tous les cas, sauf dans le premier cas qui contenait 20% de propriétés et d'instances. De plus, les résultats ont montré que la qualité de DMOM n'était pas affectée par l'augmentation du nombre de propriétés de données et du nombre d'instances. Ainsi, la qualité de DMOM atteignait 92%, alors que celles de EIFS et de RiMOM étaient inférieures à 70% et 72%,

respectivement. Ces résultats ont été obtenus en utilisant la technique de sélection des propriétés les plus pertinentes de jeu de données.

5.5 Conclusion

Ce chapitre présente un système de liage des données basé instances en étudiant trois stratégies de sélection de fonctionnalités.

La première, appelée stratégie exhaustive, explore l'arbre de recherche d'une façon aléatoire en générant un sous-ensemble d'attributs d'entités à chaque itération, où chaque nœud est évalué en exécutant le processus de correspondance sur ses attributs sélectionnés.

La deuxième, appelée stratégie statistique, utilise les valeurs statistiques pour sélectionner les propriétés les plus pertinentes.

La troisième, appelée stratégie FIM, explore une corrélation différente entre les propriétés des données et sélectionne les propriétés les plus fréquentes décrivant les instances globales du jeu de données.

Pour évaluer notre système DMOM, des expériences intensives ont été menées sur les bases de données OAEI et DBpedia. Les résultats ont montré que la stratégie FIM, a surpassé les deux autres stratégies (la stratégie exhaustive et la stratégie statistique) en ce qui concerne la qualité du résultat du processus d'appariement. Les résultats ont également révélé que le DMOM a surpassé les méthodes de base (l'EIFPS et RiMOM) en termes de temps d'exécution et de qualité de la solution.

Compte tenu des résultats obtenus, l'objectif visé, à savoir réduire les propriétés dans la phase de pré-traitement de données s'avère pertinente et intéressante. De plus, l'idée d'exploiter la corrélation entre les données via leurs propriétés en sélectionnant les propriétés les plus fréquentes s'avère prometteuse. Comme dans la phase de liage de données, les techniques classiques ont été utilisées. A notre sens, on ne peut pas aller au delà du pré-traitement avec les techniques présentées dans le chapitre 2. Il serait peut être intéressant d'explorer d'autres techniques comme celle de fouille de données. C'est ce que nous avons fait dans le chapitre 6.

Chapitre 6

Le système Clustering for Ontology Matching-based Instances (COMI)

6.1 Introduction

Ce chapitre est consacré au processus d'alignement en se basant sur une technique de fouille de données en l'occurrence le clustering, nommé COMI (Clustering for Ontology Matching-based Instances). COMI est utilisé pour découvrir les connaissances pertinentes en examinant la corrélation entre les instances basées sur des approches de clustering. COMI regroupe d'abord les instances hautement corrélées de chaque jeu de données dans des clusters similaires à l'aide de l'algorithme k-means. L'idée clé de cette méthode est qu'elle regroupe les instances de chaque jeu de données, puis fait le liage entre les clusters des deux jeux de données et les instances correspondantes dans les clusters.

Pour démontrer l'utilité et la précision du COMI, plusieurs expériences sur les bases de données de DBpedia et ceux de la compagne d'évaluation OAEI ont été menées. Les résultats montrent que COMI surpasse des approches de liage des données en termes de temps d'exécution sans perdre la qualité pendant le processus de correspondance.

6.2 L'architecture du système COMI

Le but de notre système COMI (Clustering for Ontology Matching-based Instances) est de diviser l'ensemble des instances de chaque jeu de données en plusieurs clusters dépendants. Chaque cluster contient alors des instances hautement corrélées. Comme expliqué dans la Figure 6.1, le système COMI explore les instances pour les grouper

dans des clusters avec des fonctionnalités communes. Le système comprend principalement le regroupement et le processus de mise en correspondance (ie ; le processus de liage de données). Dans le processus de clustering, les instances entières sont divisées en plusieurs collections de sous-instances (clusters) à l'aide de techniques de fouille des données. Cette étape est considérée comme un pré-traitement des données. L'ensemble du jeu de données est ensuite regroupé en différents clusters avec un petit nombre d'instances. Chaque cluster d'instances partage le nombre maximal de propriétés communes. Les instances d'un cluster sont donc fortement corrélées. Pendant le processus d'appariement, le système COMI explore les instances des clusters pour trouver l'alignement. Au lieu d'effectuer l'opération d'alignement entre les instances des jeux de données une par une, l'alignement est établi entre les instances des deux jeux de données et leurs clusters représentatifs.

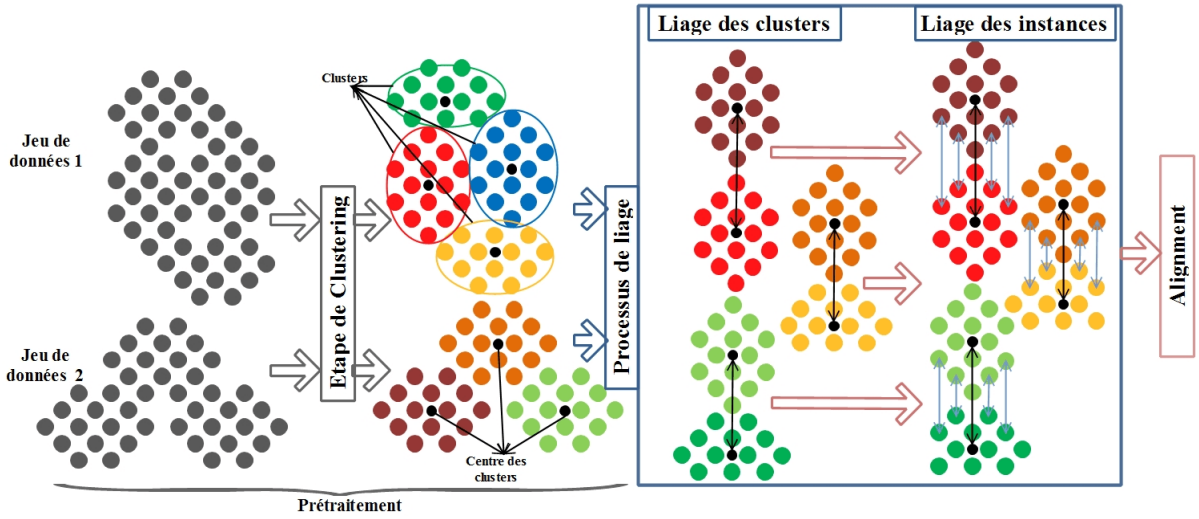


FIGURE 6.1 : Architecture du système COMI proposé.

La complexité de l'algorithme COMI dépend du nombre d'instances n, du nombre de propriétés m, du nombre de clusters k et du nombre de correspondants r . L'étape de décomposition nécessite $O(n \times m \times k)$. Ce processus n'est effectué qu'une seule fois pour chaque jeu de données. Quelque soit le nombre de correspondances à utiliser, pendant le processus de correspondance, seuls les clusters similaires sont utilisés. Cela nécessite $O(\frac{n \times m}{k})$. Le coût total de COMI pour effectuer une correspondance r est $O(n \times m \times k + r \times \frac{n \times m}{k})$, ce qui est trop faible par rapport aux solutions de base qui nécessitent $O(n \times m \times r)$.

6.3 Algorithme COMI

L'algorithme 10 présente le pseudo-code de COMI. Il considère comme entré l'ensemble des instances $\mathcal{I}$, et l'alignement de référence $\mathcal{A}$. L'ensemble des clusters est représenté par $\mathcal{G}$, et l'ensemble des centres de gravité est alors indiqué par g.

La première étape consiste à initialiser aléatoirement les centres de gravité en utilisant la fonction *InitializeCenters()*. La première boucle est effectuée à partir des lignes 5 à 17, qui parcourt l'ensemble des instances I. La fonction $\mathcal{D}_{instance}(e, g_1)$ calcule la distance entre l'instance et le premier centroïde g_1. Soit e ={(Nom, Omar), (âge, 26), (type, homme) }. Le centroïde est défini par g_1 = (26, homme, ALG). $\mathcal{D}_{instance}(e, g_1)$ calcule l'intersection des valeurs qui est fixée à 2. La boucle des lignes 9 à 13 consiste à trouver la plus petite distance entre l'instance e et tous les centroïdes de g, en conservant la plage r. La ligne 16 affecte l'instance e à la liste du cluster r qui représente la distance minimale en utilisant la fonction *AddElement()*.

A partir des lignes de 18 à 24, nous mettons à jour les centres et les conservons dans l'ensemble g'. Si g_{new} est égal au centre précédent dans g, le processus de clustering est alors terminé ; sinon, le même processus est répété jusqu'à ce que g_{new} et g deviennent identiques.

Les résultats finaux du clustering sont ensuite conservés dans une structure matricielle, appelée $\mathcal{MIC}$. Chaque élément $\mathcal{MIC}[i][j]$ correspond à la distance entre le centre de gravité g_j et la $i^{ème}$ instance du $j^{ème}$ cluster, qui est notée : $\mathcal{G}_j^i$ (lignes 25 à 29).

Depuis les lignes 35 à 45, l'algorithme parcourt l'ensemble des centroïdes G^i, G^j des deux jeux de données $\mathcal{D}^i$ et $\mathcal{D}^j$. La distance minimale entre deux centroïdes avec la fonction $\mathcal{D}_{centroïde}$ $(g_{l_1}^i, g_{l_2}^j)$ est déterminée. L'algorithme sélectionne ensuite la distance minimale et ajoute les deux clusters à la liste des clusters d'alignement *list* en utilisant la fonction $AddClusters()$.

Des lignes 48 à 58, l'algorithme parcourt toutes les instances des deux clusters alignés. Soient p et q représentés comme deux clusters sélectionnés, et la boucle des lignes 50 à 56 scanne toutes les instances e_1 et e_2 pour les deux clusters p et q. La distance minimale peut être calculée en utilisant la formule $\mathcal{D}_{instance}$. Nous ajoutons ensuite les résultats d'alignement des clusters p et q, et nous les désignons par $\mathcal{A}_{p,q}$, pour l'ensemble des $\mathcal{A}$ alignés. Ce processus est répété pour tous les clusters dans *list*.

Dans ce qui suit, nous allons détailler les étapes de décomposition et de correspondance.

Algorithm 10 COMI : Clustering for Ontology Matching

Entrée : $\mathcal{I}^i = \{\mathcal{I}^i_1, \mathcal{I}^i_2 ... \mathcal{I}^i_{n_i}\}$: L'ensemble de n instances du jeu de données $\mathcal{D}^i$.
Output : $\mathcal{A}$: L'ensemble d'alignement.
********** **initialisation des centroïdes** **********
InitializeCenters(g^i)
for chaque instance e $\in \mathcal{I}^i$ **do**
    dis $\leftarrow \mathcal{D}_{instance}$(e,$g^i_1$) {Voir EQ.6.1}
    r $\leftarrow$ 1
    for j=2 to k **do**
        d $\leftarrow \mathcal{D}_{instance}$(e,$g^i_j$) {Voir EQ.6.1}
        if d < dis **then**
            dis $\leftarrow$ d
            r $\leftarrow$ j
        end if
    end for
    AddElement(e,c^i_r)
end for
repeat
    change $\leftarrow$ false
    $g^i_{new} \leftarrow$ UpdateCenter (g,$\mathcal{G}^i$)
    if g^i != g^i_{new} **then**
        change $\leftarrow$ true
    end if
until *change* == $false$
******** $\mathcal{MIC}^i(n_i \times k_i)$ **: Matrice des distances entre** $\mathcal{I}^i$ **and** g^i ********
for i = 1 to n **do**
    for j=1 to k **do**
        $MIC^i[i,j] \leftarrow \mathcal{D}_{instance}$ ($\mathcal{G}^i_{ij}$,g_j) {$\mathcal{G}^i_{ij}$ est la $i^{ème}$ instances du cluster $\mathcal{G}^i_j$}
    end for
end for
return $\mathcal{MIC}^i$
********** **Processus de liage** **********
list $\leftarrow \emptyset$
for p = 1 to k_i **do**
    min $\leftarrow \mathcal{D}_{centroid}$ (g^i_p, g^j_1) {Voir EQ.6.3}
    indice $\leftarrow$ 1
    for q = 2 to k_j **do**
        d $\leftarrow \mathcal{D}_{centroid}$($g^i_p$, g^j_q) {Voir EQ.6.3}
        if d < min **then**
            min $\leftarrow$ d
            r $\leftarrow$ j
        end if
    end for
    list $\leftarrow$ list $\cup$ AddClusters(c^i_p, c^j_r)
end for
$\mathcal{A} \leftarrow \emptyset$
for chaque (p, q) $\in$ list **do**
    min $\leftarrow n_i \times n_j$ { initialiser la distance minimum}
    for chaque instance $(e_1, e_2) \in (\mathcal{G}^i_p, \mathcal{G}^j_q)$ **do**
        d $\leftarrow \mathcal{D}_{matching}(e_1, e_2)$ {Voir EQ.6.4}
        if d $\leq$ min **then**
            min $\leftarrow$ d
            $\mathcal{A}_{p,q} \leftarrow (e_1, e_2)$
        end if
    end for
    $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_{p,q}$
end for
return $\mathcal{A}$

6.3.1 Décomposition

Le problème de liage des données traite généralement un grand nombre d'instances, ce qui est une tâche non triviale, en particulier avec des jeux de données à une très

grande échelle. Ainsi, il est nécessaire de décomposer les énormes données en un petit nombre de clusters, ce qui réduit la difficulté du processus d'appariement. Dans cette section, nous étudions l'approche basée sur le partitionnement et utilisons le k-means [MacQueen et al., 1967] algorithme pour le problème de correspondance. La distance et le calcul du centre de gravité sont définis comme suit.

Definition 6.3.1 (Distance entre instances)**.** On note p^i_{jl}, la valeur de la propriété $\mathcal{P}^i_j$ dans l'instance $\mathcal{I}^i_l$ du jeu de données $\mathcal{D}_i$. Nous définission la distance $\mathcal{D}_{instance}$ entre deux instances $\mathcal{I}^i_{l_1}$ et $\mathcal{I}^i_{l_2}$ comme suit :

$$\mathcal{D}_{instance}(\mathcal{I}^i_{l_1}, \mathcal{I}^j_{l_2}) = n_i - (|\bigcup_{j=1}^{n_i} p^i_{jl_1} \cap p^i_{jl_2}|) \tag{6.1}$$

Pour calculer les centroïdes, on considère l'ensemble des instances du cluster $\mathcal{G}_s = \{\mathcal{I}^{(s)}_1, \mathcal{I}^{(s)}_2, ..., \mathcal{I}^{(s)}_{|G_s|}\}$. Le but est de trouver un centre de gravité de cet ensemble qui est également une instance.

Inspiré de la formule du centroïde développée dans [Djenouri et al., 2017b], nous calculons le centroïde μ_s. La fréquence de chaque valeur est calculée pour toutes les instances du cluster G_s.

Les valeurs des instances dans G_s sont triées en fonction de leur fréquence, et seule la valeur fréquente n_i est affectée à μ_s, comme

$$\mu_s = \{j | j \in \mathcal{F}_{n_i}\} \tag{6.2}$$

où $\mathcal{F}_{n_i}$ désigne l'ensemble des éléments fréquents n_i du cluster G_s.

k-means est un algorithme de clustering basé sur le partitionnement bien connu. L'idée principale est de définir les k clusters , et de diviser l'ensemble des instances de chaque jeu de données en k sous-ensembles en prenant en compte la corrélation entre les instances du même cluster. Le processus k-means commence par initialiser les k clusters. Les k instances des jeux de données peuvent être sélectionnées au hasard. Ensuite, il analyse chaque instance de l'ensemble complet, calcule la distance entre cette instance et tous les centres de gravité, et l'assigne au cluster avec le centre de gravité le plus proche. Une fois toutes les instances examinées, le centre de gravité de chaque cluster est mis à jour. Ce processus est répété jusqu'à ce que le centre de gravité du cluster devienne stable.

6.3.2 Processus de liage

Cette étape bénéficie de l'étape de clustering en définissant une nouvelle stratégie de liage, au lieu de calculer la similitude entre deux paires d'instances des jeux de données. D'abord, les mesures de similitude entre les centres de gravité des clusters sont calculés, ensuite la similarité entres les instances sont déterminées.

Deux distances sont définies, la première distance vise à déterminer la similitude entre deux centroïdes de deux jeux différents. La seconde représente la distance entre deux instances dans deux jeux de données différents. L'idée principale du processus d'appariement est de trouver deux clusters fortement corrélés parmi les jeux de données en considérant leurs distances minimales. Après cela, nous vérifions les instances parmi les clusters et essayons de trouver les instances approximatives.

Definition 6.3.2 (Distance entre centroïdes)**.** Considérons g et g' comme deux centroïdes de deux jeux de données. Nous définitions la distance $\mathcal{D}_{matching}$ entre deux centroïdes g et g' comme suit :

$$\mathcal{D}_{centroid}(g, g') = |g| + |g'| - |g \cap g'| \tag{6.3}$$

Sachant que $|g|$, $|g'|$, $|g \cap g'|$, sont respectivement le nombre de propriétés des centres de gravité g, g', et leur intersection.

Definition 6.3.3 (Liage des instances)**.** Nous définissions la distance $\mathcal{D}_{matching}$ entre deux instances $\mathcal{I}^i_{l_1}$ et $\mathcal{I}^j_{l_2}$ comme la somme des distances entre chaque instance et son centroïde et la distance entre les deux centroïdes de ces instances :

$$\mathcal{D}_{matching}(\mathcal{I}^i_{l_1}, \mathcal{I}^j_{l_2}) = \mathcal{D}_1(g_i, g_j) + \mathcal{D}_2(\mathcal{I}^i_{l_1}, g_i) + \mathcal{D}_2(\mathcal{I}^j_{l_2}, g_j) \tag{6.4}$$

Où :
$\mathcal{D}_1$ est $\mathcal{D}_{centroid}$, et $\mathcal{D}_2$ est $\mathcal{D}_{instance}$.

6.4 Études expérimentales

Pour valider la qualité du système COMI proposé, des expériences ont été menées sur des bases de données bien connues. L'algorithme a été implémenté dans le langage de programmation Java et les expériences ont été exécutées sur une machine de bureau équipée d'un processeur Intel *I7* et d'une mémoire de 16*Go*. (pour plus de détails sur les données revoir la section 5.4)

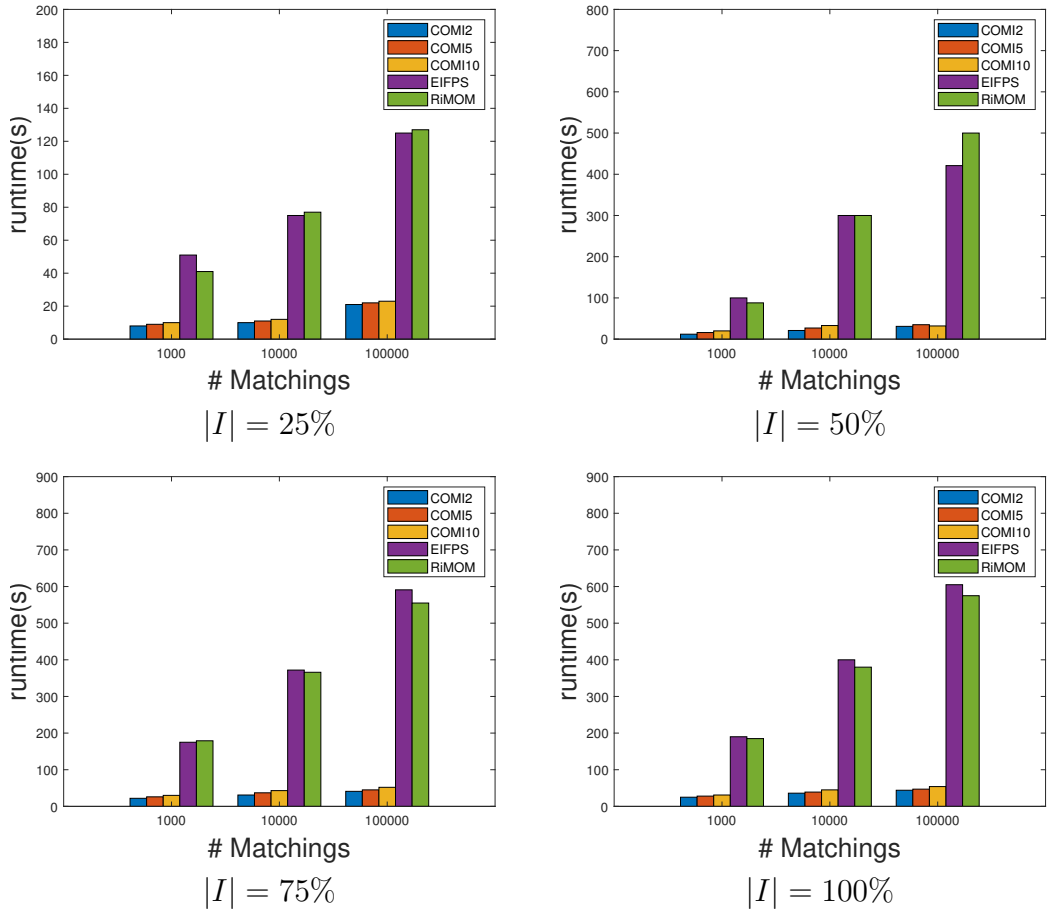


FIGURE 6.2 : Une comparaison de temps d'exécution COMI, EIFPS et RiMOM utilisant DBpedia variant de 25% à 100% et le nombre d'appariements variant de $1,000$ à $100,000$.

6.4.1 Performances d'exécution

La première série d'expériences a été réalisée pour comparer le temps d'exécution de COMI avec des approches de l'état d'art sous divers clusters. $COMI|X|$, où $|X|$ est le nombre de clusters, a été utilisé dans l'approche COMI. Le temps d'exécution correspond au temps de l'ensemble du processus COMI, y compris les étapes de décomposition et de liage. La figure 6.2 montre le temps d'exécution des cinq approches (COMI2, COMI5, COMI10, EIFPS et RiMOM), où les pourcentages d'instances variaient de 25% à 100%. Lorsque le nombre d'appariements est passé de $1,000$ à $100,000$, COMI a surpassé les deux autres approches. De plus, le temps d'exécution de COMI est resté stable, tandis que les approches de base nécessitaient un temps de calcul supplémentaire pour un grand nombre d'instances et de nombreuses correspondances. Ainsi, les deux approches comparées (EIFPS et RiMOM) ont nécessité plus

de 600 secondes pour gérer les correspondances de 100, 000 avec les données DBpedia, contre 54 secondes pour COMI10 conçu (COMI avec dix clusters). Ces résultats s'expliquent par le fait que notre approche ne considère que les instances hautement corrélées dans le processus de mise en correspondance en développant une stratégie efficace pour explorer les informations fournies dans chaque cluster d'instances. Les résultats montrent également qu'en augmentant le nombre de clusters de 2 à 10, une légère différence en termes de temps d'exécution pourrait être obtenue. Le processus de mise en cluster n'a été adopté que lors de l'étape de pré-traitement.

6.4.2 Qualité de la solution

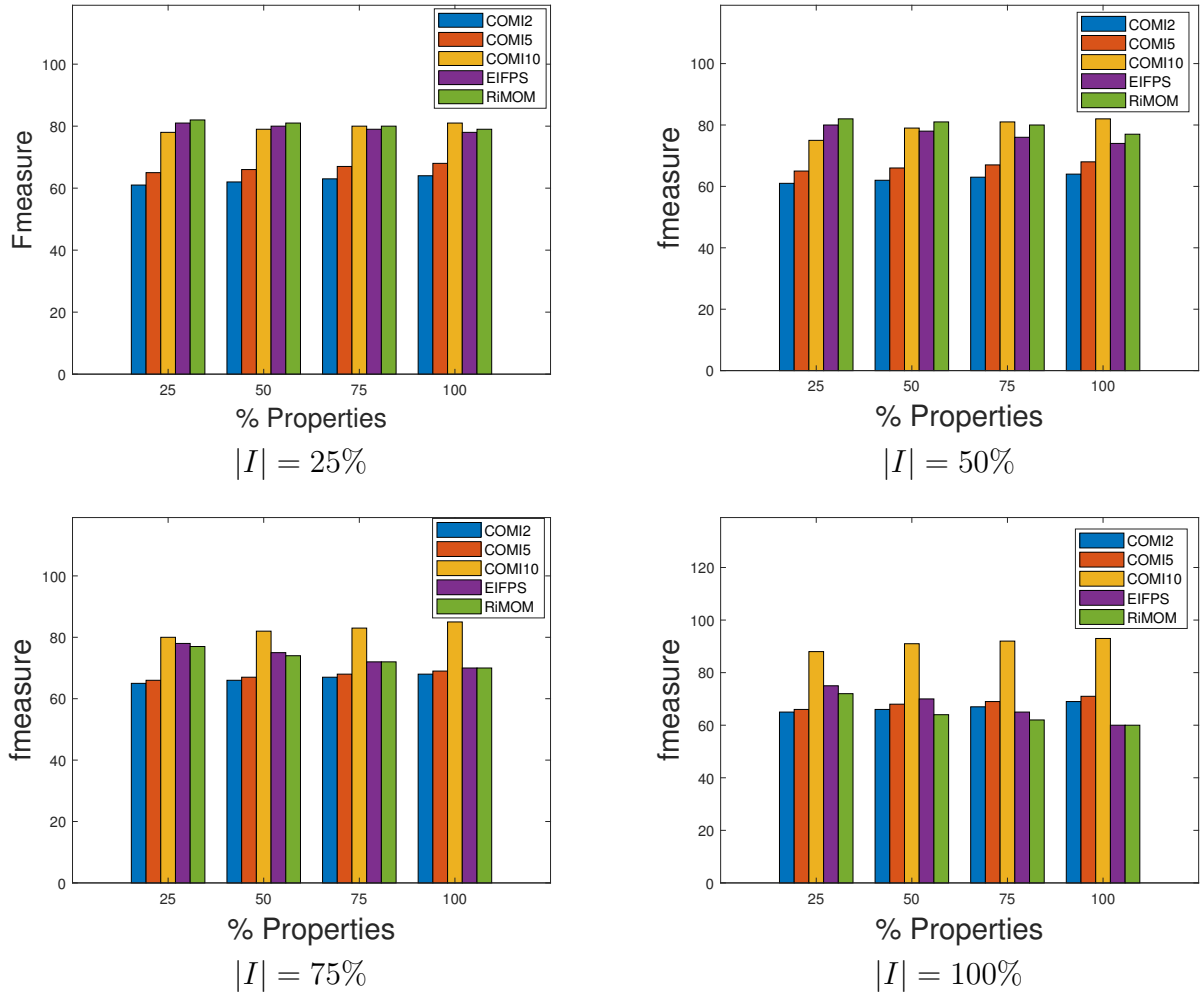


FIGURE 6.3 : Les résultats comparatifs de COMI, EIFPS et RIMOM mesurent les performances avec DBpedia en faisant varier le pourcentage d'instances et le pourcentage de propriétés de 25% à 100%.

Un deuxième ensemble d'expériences a été réalisé pour comparer la qualité des solutions par COMI avec les algorithmes EIFPS et RiMOM en utilisant la base de données DBPedia. La figure 6.3 montre les résultats des cinq approches (COMI2, COMI5, COMI10, EIFPS et RiMOM), où les pourcentages des propriétés varient toujours de 25% à 100%, respectivement. Les résultats révèlent que les approches COMI10, EIFPS et RiMOM avaient une qualité similaire, tandis que COMI5 et COMI2 offraient moins de qualité que les premières. Ainsi, si plus de clusters sont générés, alors le système COMI ainsi conçu peut obtenir de meilleurs résultats. Comme c'est le cas pour, dix clusters avec les données DBPedia. De plus, COMI10 avait de meilleures performances que les algorithmes EIFPS et RiMOM avec des données de grande dimension. Par exemple, lorsque le pourcentage de propriétés et d'instances était défini sur 25%, la F_mesure d'EIFPS et de RiMOM était respectivement de 81% et de 82%, tandis que COMI10 n'atteignait pas 80%. Cependant, pour 100% des données, la F_mesure de COMI était de 93%, tandis que celles des deux autres approches étaient d'environ 60%. Nous expliquons ce problème par le fait que la qualité du clustering avec $k = 10$ était meilleure que $k = 2$ et $k = 5$. Un grand nombre de similarité partageant un grand nombre de propriétés a été obtenu avec $k = 10$, par contre les propriétés étaient hétérogènes avec des propriétés différentes qui ont été déterminées en explorant deux et cinq clusters. Seuls 2, 5 et 10 clusters ont été étudiés dans cette expérience, car la qualité du clustering était réduite lors de la définition du nombre de clusters au-dessus de 10.

On peut conclure de ces résultats que COMI a obtenu les meilleurs résultats en termes d'exécution par rapport aux algorithmes de liage existants, en particulier pour les grands jeux de données comme DBPedia. De plus, ce problème ne dégrade pas la qualité de la solution si le nombre approprié de clusters est choisi.

6.5 Comparaison entre DMOM et COMI

Dans cette expérience, l'évolutivité des système COMI et DMOM a été évaluée. Plusieurs critères, tels que la qualité des solutions, le coût de calcul (c'est-à-dire le temps d'exécution) et l'utilisation de la mémoire, ont été évalués sur les bases de données OAEI. L'API Java standard a été utilisée dans les expérimentation, et pour l'utilisation de la mémoire. Les résultats du tableau (6.1) présentent la F_mesure, le temps CPU et l'utilisation de la mémoire de DMOM et COMI sous diverses bases de données et stratégies (c.-à-d. énumère de manière exhaustive toutes les correspondances possibles de deux jeux de données). Comme indiqué, DMOM a obtenu les meilleurs

Data	Exhaustive			COMI			DMOM		
	F-meas.	CPU Time	Mem.	F-meas.	CPU Time	Mem.	F-meas.	CPU Time	Mem.
Dis_1	0.76	3.25	115	0.84	**2.12**	**95**	**0.93**	2.61	102
Dis_2	0.77	4.26	158	0.88	**3.91**	**112**	**0.95**	3.99	115
Rec_1	0.79	5.21	136	0.93	**4.96**	**118**	**0.97**	4.99	119
Rec_2	0.75	7.12	159	0.91	**6.02**	**124**	**0.96**	6.15	126
ID_1	0.79	6.15	171	0.92	6.36	**105**	**0.98**	**6.21**	101
ID_2	0.80	10.23	166	0.91	9.12	**99**	**0.95**	**9.02**	103
Sim_1	0.74	12.98	147	0.91	10.12	**113**	**0.95**	**9.85**	117
Sim_2	0.72	14.13	151	0.90	**10.18**	**88**	**0.94**	11.02	92
O_{101}	0.74	11.02	110	0.92	11.00	101	**0.93**	**10.02**	**97**
O_{104}	0.73	14.15	187	**0.94**	**12.25**	102	0.92	13.02	**95**
O_{202}	0.81	18.69	**84**	0.92	17.65	111	**0.93**	**14.23**	88
O_{230}	0.82	19.36	82	**0.95**	21.02	78	0.92	**18.26**	**67**
B_{000}	0.89	**8.26**	**83**	0.95	9.26	87	**0.95**	10.29	93
B_{104}	0.81	12.25	112	**0.96**	**4.12**	92	0.95	5.26	**89**
P_{11}	0.83	5.26	129	0.97	4.26	**81**	**1.0**	**3.77**	85
P_{12}	0.85	12.25	127	0.97	10.25	83	**1.0**	**9.36**	**81**
P_{21}	0.84	11.29	124	0.98	12.03	80	**1.0**	**8.76**	**78**
P_{22}	0.87	15.23	119	0.99	**12.36**	**75**	**1.0**	13.62	77

TABLE 6.1 : Une comparaison du F_mesure, du processeur (en sec.) et l'utilisation de la mémoire (Mo) des trois approches (Exhaustive, COMI et DMOM).

résultats par rapport aux deux autres stratégies en termes de F_mesure pour 15 et 18 cas. La qualité de DMOM dans tous les cas était jusqu'à 92%, tandis que la qualité du COMI et exhaustif était inférieure à 84% et72%. Ces résultats ont été obtenus grâce aux connaissances découvertes par DMOM, qui ont permis de mieux réduire l'espace dimensionnel des bases de données. Les résultats ont également montré que l'utilisation de la mémoire et les performances d'exécution de COMI et DMOM convergeaient vers les mêmes valeurs. L'approche exhaustive a cependant obtenu les pires résultats des deux mesures, ce qui peut être attribué au fait que la stratégie exhaustive a répertorié toutes les combinaisons sans augmenter le processus de recherche. Les deux autres stratégies ont amélioré l'exploration de l'espace de solution en utilisant les clusters et les modèles de fouille de données.

6.6 Cas d'étude sur la modélisation sémantique des villes intelligentes

Une série d'expériences visait à montrer la capacité des algorithmes COMI et DMOM à gérer la modélisation sémantique dans les environnements de villes intelligentes. Alors que de nombreuses propositions ont été faites concernant les données des villes intelligentes, la modélisation sémantique à partir de ces données est un sujet de recherche ouvert dans la communauté des villes intelligentes. Dans cette étude, nous traitons ce problème difficile en appliquant le processus de correspondance aux données des villes intelligentes décrites dans [1]. Le tableau 6.2 montre les résultats des trois approches (DMOM, COMI et RiMOM), où les pourcentages des instances et des propriétés variaient de 20% à 100%. Les résultats ont révélé que le COMI et le DMOM ont surpassé RiMOM en termes de temps d'exécution et de qualité de la solution. Ces résultats confirment à nouveau l'utilité de COMI et DMOM pour résoudre le problème d'appariement des données et leur capacité à traiter des données hétérogènes à grande échelle. De nos nombreuses expériences portant sur les données des villes intelligentes, certaines perspectives restent à étudier :

1. Détection des valeurs aberrantes : de nombreuses valeurs aberrantes ont été trouvées dans les expériences. Ces valeurs aberrantes ont réduit les performances globales du processus de correspondance d'ontologie. Il serait avantageux de les supprimer lors de l'étape de prétraitement. Une solution consiste à appliquer les algorithmes de détection des valeurs aberrantes existants, tels que le facteur de valeur aberrante locale et k les plus proches voisins. Une distance d'accessibilité locale entre les propriétés et les instances doit être développée pour adapter ces algorithmes à une ontologie.

2. Crowdsourcing : les solutions de correspondance d'ontologies pourraient identifier différents alignements à partir des mêmes données. Le problème est de savoir comment décider quels alignements sont utiles aux urbanistes. Une approche de crowdsourcing peut être appliquée pour améliorer l'utilité de l'alignement détecté, où différentes approches d'appariement d'ontologies devraient travailler ensemble pour identifier les meilleurs alignements fournis aux urbanistes. Les agents représentés par les approches et les programmes pourraient

[1] http ://www.noaa.org/

trouver localement les alignements et les envoyer aux urbanistes. Ensuite, les urbanistes pourraient utiliser des environnements de crowdsourcing pour trouver le meilleur alignement pour la modélisation sémantique de la ville intelligente.

3. Manque de vérité terrain : Le manque de vérité terrain est un problème courant dans l'évaluation des algorithmes de correspondance d'ontologie, en particulier, pour des scénarios réels, tels que la modélisation sémantique de ville intelligente. En tant que défis pour la recherche future concernant l'évaluation de la qualité des résultats de correspondance d'ontologie, les problèmes et questions de recherche suivants restent à résoudre :

   - La définition de données de référence utiles et accessibles au public sur les villes intelligentes pour les problèmes de modélisation sémantique est bénéfique pour l'analyse des algorithmes de correspondance d'ontologie.
   - Il serait très utile d'identifier les critères significatifs pour une évaluation interne de l'appariement d'ontologie. Une façon de résoudre ce problème difficile consiste à fournir des scores de fonction de classement unifiés pour classer les alignements. Ces fonctions doivent être indépendantes de l'ensemble du processus d'identification des meilleurs alignements.

6.7 Conclusion

Ce chapitre présente un système de liage des données basé instances en utilisant la technique du clustering dans la partie du processus de liage.

Il se compose principalement de deux étapes. La première étape vise à regrouper les instances de chaque jeu de données dans des clusters similaires en utilisant l'algorithme k-means. Après cela, les groupes extraits sont alors utilisés pour trouver la correspondance entre les instances entre les deux jeux de données.

Pour évaluer notre système, des expériences intensives ont été menées sur les bases de données OAEI et DBpedia. Les résultats ont révélé que le COMI a surpassé les méthodes de base (l'EIFPS et RiMOM) en termes de temps d'exécution et de qualité de la solution. Dans ce chapitre, nous avons comparer nos deux travaux le système DMOM et COMI, on constate que les deux stratégies ont améliorées l'exploration de l'espace de solution en utilisant les clusters et la fouille de données. Pour clôturer notre contribution , nous avons ajouter un petit cas d'étude sur la modélisation sémantiques villes intelligentes, pour voir l'application de notre travail dans un cas réel.

% I	%P	DMOM		COMI		RiMOM	
		F-measure	CPU (sec.)	F-measure	CPU (sec.)	F-measure	CPU (sec.)
20	20	0.97	0.95	0.96	0.97	0.94	0.95
	50	**0.97**	**0.95**	**0.96**	0.92	0.92	0.92
	80	**0.97**	**0.95**	**0.96**	0.90	0.91	0.90
	100	**0.97**	**0.95**	**0.96**	0.88	0.90	0.89
50	20	**0.96**	**0.94**	**0.95**	0.93	0.92	0.92
	50	**0.96**	**0.94**	**0.95**	0.89	0.87	0.88
	80	**0.96**	**0.94**	**0.95**	0.87	0.84	0.85
	100	**0.96**	**0.94**	**0.95**	0.85	0.82	0.83
80	20	**0.95**	**0.92**	**0.93**	0.90	0.89	0.89
	50	**0.95**	**0.92**	**0.93**	0.88	0.86	0.87
	80	**0.95**	**0.92**	**0.93**	0.82	0.80	0.81
	100	**0.95**	**0.92**	**0.93**	0.78	0.75	0.76
100	20	**0.94**	**0.90**	**0.92**	0.85	0.82	0.83
	50	**0.94**	**0.90**	**0.92**	0.83	0.81	0.82
	80	**0.94**	**0.90**	**0.92**	0.78	0.75	0.76
	100	**0.94**	**0.90**	**0.92**	0.74	0.70	0.72

TABLE 6.2 : Une comparaison de F_mesure et du CPU de DMOM, COMI et RiMOM à l'aide des données d'une ville intelligente en faisant varier à la fois le pourcentage d'instances (% I) et le pourcentage des propriétés (%P) de 20% à 100%.

Conclusion Générale et Perspectives

Bilan

Dans cette thèse, nous avons traité le problème de liage de données de masse. Dans un premier lieu, nous avons fait un état de l'art sur le développement du web de données et de ces technologies. Nous avons vu que l'approche classique de résolution du problème ne convenait pas, à cause de l'augmentation sans cesse des données publiées selon les technologies du web sémantique.

Pour y remédier et afin de réaliser nos systèmes, nous avons adopté un processus en deux phases : i) La phase de pré-traitement des données qui permettra la sélection des attributs pertinents. ii) La phase d'alignement des données qui permettra de lier des instances par une relation d'équivalence.

Dans un premier temps, nous avons appliqué l'algorithme génétique ainsi que les règles d'association dans la phase de pré-traitement. Deux systèmes ont donc été réalisés. Le premier système GFSOM (Genetic Feature Selection for Ontology Matching) utilise une méta-heuristique en l'occurrence l'algorithme génétique tandis ce que le second, le système DMOM (Data Mining for Ontology Matching based instances), est basé sur une technique de fouille de données en l'occurrence les items sets fréquents (FIM).

Dans un second temps, nous avons testé la technique de clustering dans le processus global de liage de données. Le système COMI (Clustering for Ontology Matching-based Instances) a été développé. Il regroupe les instances hautement corrélées dans des clusters à l'aide de l'algorithme K-means.

Afin de valider les approches proposées, des expérimentations approfondies ont été menées à l'aide de jeux de données bien connues, ceux de la campagne OAEI (Ontology Alignment Evaluation Initiative) où le nombre d'instances peux varier de 47 à 29645, et celui de DBpedia qui est un hub de données avec au maximum $4,233,000$ instances

et 2, 795 propriétés différentes. Les évaluations ont été faites en termes de précision, temps d'exécution et parfois d'utilisation de la mémoire.

La proposition que nous avons faite sur la base de données DBpedia par nos différents système à conduit aux résultats suivants : La qualité de GFSOM atteignait 90% tandis que celles de EIFPS et de RiMOM était inférieure à 72%. La qualité de DMOM atteignait 92%, alors que celles de EIFS et de RiMOM étaient inférieures à 70% et 72%, respectivement. On déduit, que les systèmes GFSOM et DMOM n'étaient pas affectés par l'augmentation du nombre de propriétés et d'instances.

La F_mesure d'EIFPS et de RiMOM était respectivement de 81% et de 82%, tandis que COMI10 n'atteignait pas 80%. Cependant, pour 100% des données, la F_mesure de COMI était de 93%, tandis que celles des deux autres approches étaient d'environ 60%. Ce problème ne dégrade pas la qualité de la solution si le nombre approprié de clusters est choisi convenablement.

Par contre, dans la comparaison entre nos systèmes, la qualité de DMOM dans tous les cas était jusqu'à 92%, tandis que celle de COMI était inférieure respectivement à 84% et72%. Ces résultats ont été obtenus grâce aux connaissances découvertes par DMOM, qui ont permis de mieux réduire l'espace dimensionnel des bases de données. Les résultats ont également montré que l'utilisation de la mémoire et les performances d'exécution de COMI et DMOM convergeaient vers les mêmes valeurs.

Perspectives

A l'issue de ce travail de thèse, nous pouvons dire que la direction que nous avons suivi pour le problème de liage de données de masse trouvait une solution dans certaines techniques de fouille de données et de méta-heuristiques. Cependant, plusieurs points restent à éclaircir et d'autres problèmes restent encore à développer. Les perspectives envisageables pour ce travail de thèse sont :

De comparer les algorithmes proposés pour la "sélection des attributs" et estimer la meilleure solution, étant donné que "le processus d'alignement" est le même pour les deux systèmes GFSOM et DMOM. Aussi, il serait intéressant de comparer les résultats de COMI avec le meilleur système qu'on aura estimé précédemment.

D'appliquer les algorithmes proposés dans les domaines qui utilisent de grande quantités de données, à titre d'exemple des réseaux sociaux (Twitter, Facebook, ...).

Pour soutenir l'importance du système réalisé, il serait opportun de l'intégrer dans le cadre de services web sémantiques, pour montrer leur efficacité à grand échelle.

Implémenter les systèmes (DMOM,COMI) proposés avec des processus en parallèle afin d'augmenter le temps d'exécution.

Bibliographie

[Abbas et al., 2019] Abbas, N., David, J., and Napoli, A. (2019). Linkex : A tool for link key discovery based on pattern structures.

[Abubakar et al., 2018] Abubakar, M., Hamdan, H., Mustapha, N., and Aris, T. N. M. (2018). Instance-based ontology matching : A literature review. In *International Conference on Soft Computing and Data Mining*, pages 455–469. Springer.

[Acampora et al., 2012] Acampora, G., Loia, V., Salerno, S., and Vitiello, A. (2012). A hybrid evolutionary approach for solving the ontology alignment problem. *International Journal of Intelligent Systems*, 27(3) :189–216.

[Acampora et al., 2013] Acampora, G., Loia, V., and Vitiello, A. (2013). Enhancing ontology alignment through a memetic aggregation of similarity measures. *Information Sciences*, 250 :1–20.

[Agrawal et al., 1998] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105.

[Agrawal et al., 1993] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM.

[Al-Bakri et al., 2016] Al-Bakri, M., Atencia, M., David, J., Lalande, S., and Rousset, M.-C. (2016). Uncertainty-sensitive reasoning for inferring same as facts in linked data. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 698–706. IOS press.

[Al-Bakri et al., 2015] Al-Bakri, M., Atencia, M., Lalande, S., and Rousset, M.-C. (2015). Inferring same-as facts from linked data : an iterative import-by-query approach. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Alam et al., 2017] Alam, M., Recupero, D. R., Mongiovi, M., Gangemi, A., and Ristoski, P. (2017). Event-based knowledge reconciliation using frame embeddings and frame similarity. *Knowledge-Based Systems*, 135 :192–203.

[Algergawy et al., 2011] Algergawy, A., Massmann, S., and Rahm, E. (2011). A clustering-based approach for large-scale ontology matching. In *East European Conference on Advances in Databases and Information Systems*, pages 415–428. Springer.

[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics : ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2) :49–60.

[Atencia et al., 2014] Atencia, M., David, J., and Euzenat, J. (2014). Data interlinking through robust linkkey extraction. In *ECAI*, pages 15–20.

[Atencia et al., 2020] Atencia, M., David, J., Euzenat, J., Napoli, A., and Vizzini, J. (2020). Link key candidate extraction with relational concept analysis. *Discrete applied mathematics*, 273 :2–20.

[Barron et al., 1998] Barron, A., Rissanen, J., and Yu, B. (1998). The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6) :2743–2760.

[Belhadi et al., 2019] Belhadi, H., Akli-Astouati, K., Djenouri, Y., and Lin, J. C.-W. (2019). Exploring pattern mining for solving the ontology matching problem. In *European Conference on Advances in Databases and Information Systems*, pages 85–93. Springer.

[Belhadi et al., 2020] Belhadi, H., Akli-Astouati, K., Djenouri, Y., and Lin, J. C.-W. (2020). Data mining-based approach for ontology matching problem. *Applied Intelligence*, pages 1–18.

[Belhadi et al., 2018] Belhadi, H., Akli-Astouati, K., Djenouri, Y., Lin, J. C.-W., and Wu, J. M.-T. (2018). Gfsom : Genetic feature selection for ontology matching. In *International Conference on Genetic and Evolutionary Computing*, pages 655–660. Springer.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific american*, 284(5) :34–43.

[Beyer and Schwefel, 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1) :3–52.

[Bilenko et al., 2003] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5) :16–23.

[Bizer et al., 2011] Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data : The story so far. In *Semantic services, interoperability and web applications : emerging concepts*, pages 205–227. IGI Global.

[Blickle and Thiele, 1996] Blickle, T. and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4) :361–394.

[Brin et al., 1997] Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. *Acm Sigmod Record*, 26(2) :255–264.

[Cerón-Figueroa et al., 2017] Cerón-Figueroa, S., López-Yáñez, I., Alhalabi, W., Camacho-Nieto, O., Villuendas-Rey, Y., Aldape-Pérez, M., and Yáñez-Márquez, C. (2017). Instance-based ontology matching for e-learning material using an associative pattern classifier. *Computers in Human Behavior*, 69 :218–225.

[Christen, 2012] Christen, P. (2012). *Data matching : concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

[Dhiman and Kaur, 2018] Dhiman, G. and Kaur, A. (2018). Optimizing the design of airfoil and optical buffer problems using spotted hyena optimizer. *Designs*, 2(3) :28.

[Djenouri et al., 2018] Djenouri, Y., Belhadi, A., Fournier-Viger, P., and Lin, J. C.-W. (2018). Fast and effective cluster-based information retrieval using frequent closed itemsets. *Information Sciences*, 453 :154–167.

[Djenouri et al., 2017a] Djenouri, Y., Comuzzi, M., and Djenouri, D. (2017a). Ss-fim : Single scan for frequent itemsets mining in transactional databases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 644–654. Springer.

[Djenouri et al., 2017b] Djenouri, Y., Djamel, D., and Djenoouri, Z. (2017b). Data-mining-based decomposition for solving maxsat problem : Towards a new approach. *IEEE Intelligent Systems*.

[Djenouri and Zimek, 2018] Djenouri, Y. and Zimek, A. (2018). Outlier detection in urban traffic data. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, page 3. ACM.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, 1(4) :28–39.

[Elmagarmid et al., 2007] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection : A survey. *IEEE Transactions on knowledge and data engineering*, 19(1) :1–16.

[Ergezer et al., 2008] Ergezer, M., Simon, D., and Du, D. (2008). Biogeography-based optimization. In *IEEE Transactions on Evolutionary Computation*. Citeseer.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Euzenat et al., 2008] Euzenat, J., Mocan, A., and Scharffe, F. (2008). Ontology alignments. In *Ontology Management*, pages 177–206. Springer.

[Euzenat et al., 2007] Euzenat, J., Shvaiko, P., et al. (2007). *Ontology matching*, volume 18. Springer.

[Euzenat and Valtchev, 2004] Euzenat, J. and Valtchev, P. (2004). Similarity-based ontology alignment in owl-lite. ecai.

[Fellegi and Sunter, 1969] Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328) :1183–1210.

[Fogel, 1997] Fogel, D. B. (1997). The advantages of evolutionary computation. In *BCEC*, pages 1–11.

[Fuhr, 2000] Fuhr, N. (2000). Probabilistic datalog : Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2) :95–110.

[Gamberger et al., 1999] Gamberger, D., Lavrac, N., and Jovanoski, V. (1999). High confidence association rules for medical diagnosis.

[Ganter and Kuznetsov, 2001] Ganter, B. and Kuznetsov, S. O. (2001). Pattern structures and their projections. In *International conference on conceptual structures*, pages 129–142. Springer.

[Ganter and Wille, 2012] Ganter, B. and Wille, R. (2012). *Formal concept analysis : mathematical foundations*. Springer Science & Business Media.

[Goldberg and Deb, 1991] Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier.

[Goldberg and Holland, 1988] Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning.

[Gouda and Zaki, 2001] Gouda, K. and Zaki, M. J. (2001). Efficiently mining maximal frequent itemsets. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 163–170. IEEE.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2) :199–220.

[Guarino, 1998] Guarino, N. (1998). *Formal ontology in information systems : Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. IOS press.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). *Data mining : concepts and techniques*. Elsevier.

[Han et al., 2000] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM.

[He et al., 2009] He, S., Wu, Q. H., and Saunders, J. (2009). Group search optimizer : an optimization algorithm inspired by animal searching behavior. *IEEE transactions on evolutionary computation*, 13(5) :973–990.

[Heflin and Song, 2016] Heflin, J. and Song, D. (2016). Ontology instance linking : Towards interlinked knowledge graphs. In *AAAI*, pages 4163–4169.

[Hernández and Stolfo, 1995] Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. *ACM Sigmod Record*, 24(2) :127–138.

[Hinneburg and Gabriel, 2007] Hinneburg, A. and Gabriel, H.-H. (2007). Denclue 2.0 : Fast clustering based on kernel density estimation. In *International symposium on intelligent data analysis*, pages 70–80. Springer.

[Hopcroft and Tarjan, 1973] Hopcroft, J. and Tarjan, R. (1973). Algorithm 447 : efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6) :372–378.

[Iwata et al., 2017] Iwata, T., Kanagawa, M., Hirao, T., and Fukumizu, K. (2017). Unsupervised group matching with application to cross-lingual topic matching without alignment information. *Data mining and knowledge discovery*, 31(2) :350–370.

[Jiménez-Ruiz and Grau, 2011] Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap : Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer.

[Karaboga, 2005] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, engineering faculty, computer ....

[Kaveh and Talatahari, 2010] Kaveh, A. and Talatahari, S. (2010). A novel heuristic optimization method : charged system search. *Acta Mechanica*, 213(3-4) :267–289.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE.

[Klein, 2001] Klein, M. (2001). Combining and relating ontologies : an analysis of problems and solutions. In *OIS@ IJCAI*.

[Kuo et al., 2011] Kuo, R. J., Chao, C. M., and Chiu, Y. (2011). Application of particle swarm optimization to association rule mining. *Applied Soft Computing*, 11(1) :326–336.

[Lacoste-Julien et al., 2013] Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., and Ghahramani, Z. (2013). Sigma : Simple greedy matching for

aligning large knowledge bases. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 572–580.

[Lassila, 1999] Lassila, O. (1999). Resource description framework (rdf) model and syntax specification, w3c recommendation. *http ://www. w3. org/TR/PR-rdf-syntax*.

[Lauriere, 1987] Lauriere, J.-L. (1987). *Intelligence artificielle : résolution de problèmes par l'homme et la machine*. Eyrolles Paris.

[Li et al., 2013] Li, J., Wang, Z., Zhang, X., and Tang, J. (2013). Large scale instance matching via multiple indexes and candidate selection. *Knowledge-Based Systems*, 50 :112–120.

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Martinez-Gil et al., 2008] Martinez-Gil, J., Alba, E., and Aldana-Montes, J. F. (2008). Optimizing ontology alignments by using genetic algorithms. In *Proceedings of the workshop on nature based reasoning for the semantic Web. Karlsruhe, Germany*.

[Martinez-Gil and Aldana-Montes, 2011] Martinez-Gil, J. and Aldana-Montes, J. F. (2011). Evaluation of two heuristic approaches to solve the ontology meta-matching problem. *Knowledge and Information Systems*, 26(2) :225–247.

[Mata et al., 2001] Mata, J., Alvarez, J., and Riquelme, J. (2001). Mining numeric association rules with genetic algorithms. In *Artificial neural nets and genetic algorithms*, pages 264–267. Springer.

[Mata et al., 2002] Mata, J., Alvarez, J.-L., and Riquelme, J.-C. (2002). An evolutionary algorithm to discover numeric association rules. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 590–594.

[Mohammadi et al., 2018] Mohammadi, M., Atashin, A. A., Hofman, W., and Tan, Y. (2018). Comparison of ontology alignment systems across single matching task via the mcnemar's test. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4) :51.

[Mohammadi et al., 2019] Mohammadi, M., Hofman, W., and Tan, Y.-H. (2019). A comparative study of ontology matching systems via inferential statistics. *IEEE Transactions on Knowledge and Data Engineering*, 31(4) :615–628.

[Moore, 1959] Moore, E. F. (1959). The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, pages 285–292.

[Mosteghanemi, 2010] Mosteghanemi, H. (2010). *Les essaims d'abeilles pour la recherche d'information à grande échelle*. PhD thesis.

[Nentwig et al., 2017] Nentwig, M., Hartung, M., Ngonga Ngomo, A.-C., and Rahm, E. (2017). A survey of current link discovery frameworks. *Semantic Web*, 8(3) :419–436.

[Neshat et al., 2014] Neshat, M., Sepidnam, G., Sargolzaei, M., and Toosi, A. N. (2014). Artificial fish swarm algorithm : a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial intelligence review*, 42(4) :965–997.

[Niu et al., 2012] Niu, X., Rong, S., Wang, H., and Yu, Y. (2012). An effective rule miner for instance matching in a web of data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1085–1094. ACM.

[Otero-Cerdeira et al., 2015] Otero-Cerdeira, L., Rodríguez-Martínez, F. J., and Gómez-Rodríguez, A. (2015). Ontology matching : A literature review. *Expert Systems with Applications*, 42(2) :949–971.

[Park et al., 1995] Park, J. S., Chen, M.-S., and Yu, P. S. (1995). An effective hash-based algorithm for mining association rules. *Acm sigmod record*, 24(2) :175–186.

[Parsopoulos, 2009] Parsopoulos, K. E. (2009). Cooperative micro-differential evolution for high-dimensional problems. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 531–538.

[Pei et al., 2000] Pei, J., Han, J., Mao, R., et al. (2000). Closet : An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, pages 21–30.

[Rashedi et al., 2009] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa : a gravitational search algorithm. *Information sciences*, 179(13) :2232–2248.

[Rosaci, 2007] Rosaci, D. (2007). Cilios : Connectionist inductive learning and inter-ontology similarities for recommending information agents. *Information systems*, 32(6) :793–825.

[Rosaci, 2015] Rosaci, D. (2015). Finding semantic associations in hierarchically structured groups of web data. *Formal Aspects of Computing*, 27(5-6) :867–884.

[Savasere et al., 1995] Savasere, A., Omiecinski, E. R., and Navathe, S. B. (1995). An efficient algorithm for mining association rules in large databases. Technical report, Georgia Institute of Technology.

[Scharffe et al., 2011] Scharffe, F., Ferrara, A., and Nikolov, A. (2011). Data linking for the semantic web. *International Journal on Semantic Web and Information Systems*, 7(3) :46–76.

[Shah-Hosseini, 2011] Shah-Hosseini, H. (2011). Principal components analysis by the galaxy-based search algorithm : a novel metaheuristic for continuous optimisation. *International Journal of Computational Science and Engineering*, 6(1-2) :132–140.

[Shao et al., 2016] Shao, C., Hu, L.-M., Li, J.-Z., Wang, Z.-C., Chung, T., and Xia, J.-B. (2016). Rimom-im : a novel iterative framework for instance matching. *Journal of computer science and technology*, 31(1) :185–197.

[Shaw et al., 2001] Shaw, M. J., Subramaniam, C., Tan, G. W., and Welge, M. E. (2001). Knowledge management and data mining for marketing. *Decision support systems*, 31(1) :127–137.

[She Yang, 2010] She Yang, X. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bioinspired computation*, 2(2) :78–84.

[Shvaiko and Euzenat, 2013] Shvaiko, P. and Euzenat, J. (2013). Ontology matching : state of the art and future challenges. *IEEE Transactions on knowledge and data engineering*, 25(1) :158–176.

[Sitas and Kapidakis, 2008] Sitas, A. and Kapidakis, S. (2008). Duplicate detection algorithms of bibliographic descriptions. *Library hi tech*.

[Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., et al. (2007). The obo foundry : coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11) :1251.

[Storn and Price, 1995] Storn, R. and Price, K. (1995). Differential evolution-a simple efficient adaptive scheme for global optimization. Technical report, Technical Report over continuous spaces. International Computer Science ....

[Tran et al., 2011] Tran, Q.-V., Ichise, R., and Ho, B.-Q. (2011). Cluster-based similarity aggregation for ontology matching. *Ontology Matching*, 814.

[Tversky, 1977] Tversky, A. (1977). Features of similarity. *Psychological review*, 84(4) :327.

[Wang et al., 2006] Wang, J., Ding, Z., and Jiang, C. (2006). Gaom : Genetic algorithm based ontology matching. In *Services Computing, 2006. APSCC'06. IEEE Asia-Pacific Conference on*, pages 617–620. IEEE.

[Wang et al., 2011] Wang, M., Zou, Q., and Liu, C. (2011). Multi-dimension association rule mining based on adaptive genetic algorithm. In *2011 International Conference on Uncertainty Reasoning and Knowledge Engineering*, volume 2, pages 150–153. IEEE.

[Wang et al., 1997] Wang, W., Yang, J., Muntz, R., et al. (1997). Sting : A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195.

[Wang et al., 2013] Wang, Z., Li, J., Zhao, Y., Setchi, R., and Tang, J. (2013). A unified approach to matching semantic data on the web. *Knowledge-Based Systems*, 39 :173–184.

[Wu et al., 2008] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1) :1–37.

[Xue et al., 2018] Xue, X., Chen, J., Chen, J., and Chen, D. (2018). Using compact coevolutionary algorithm for matching biomedical ontologies. *Computational intelligence and neuroscience*, 2018.

[Xue and Liu, 2017] Xue, X. and Liu, J. (2017). A compact hybrid evolutionary algorithm for large scale instance matching in linked open data cloud. *International Journal on Artificial Intelligence Tools*, 26(04) :1750013.

[Yan et al., 2009] Yan, X., Zhang, C., and Zhang, S. (2009). Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems with Applications*, 36(2) :3066–3076.

[Yang, 2010] Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer.

[Yang, 2019] Yang, X.-S. (2019). *Introduction to Algorithms for Data Mining and Machine Learning*. Academic Press.

[Yang and Deb, 2009] Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, pages 210–214. IEEE.

[Zobel and Moffat, 2006] Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM computing surveys (CSUR)*, 38(2) :6–es.