N° d'ordre :

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES - SIDI BEL ABBES
FACULTE DES SCIENCES EXACTES
DEPARTEMENT D'INFORMATIQUE

THESE DE DOCTORAT EN SCIENCE

Présentée par
OUHAB Abdelkrim

Spécialité : *Informatique*
Option : *Systèmes d'information et de connaissances (SIC)*

Intitulée

Qualité de données pour l'intégration de données

Soutenue le ……
devant le jury composé de :

Mr. TOUMOUH Adil	M.C.A à l'université de sidi bel abbès	Président
Mr. BENSLIMANE Sidi Mohamed	Professeur à l'ESI de sidi bel abbès	Examinateur
Mr. AMAR BENSABER Djamel	M.C.A à l'ESI de sidi bel abbès	Examinateur
Mr. BOUKLI HACENE Sofiane	M.C.A à l'université de sidi bel abbès	Examinateur
Mr. ALI CHERIF Moussa	M.C.A à l'université de sidi bel abbès	Examinateur

Directeur de thèse : Mr MALKI Mimoun, Professeur à l'ESI de sidi bel abbès

Année universitaire : 2018-2019

Dédicaces

A ma grande et petite famille

Remerciements

Cette thèse ne serait pas possible sans l'encouragement constant de plusieurs personnes. Il m'est alors très agréable d'exprimer ma reconnaissance auprès de toutes ces personnes, dont l'intervention au cours de ces années de thèse, a favorisé son aboutissement.

Je remercie tout particulièrement mon directeur de thèse Monsieur MALKI Mimoun, Professeur à l'école supérieure en Informatique de Sidi Bel Abbes pour sa disponibilité, ses critiques et ses conseils toujours opportuns ainsi que pour sa grande gentillesse et de m'avoir fait confiance et d'avoir cru à mes compétences.

Mes vifs remerciements s'adressent aussi aux membres du jury Monsieur TOUMOUH Adil, Maître de conférences A à l'université de sidi bel abbès, Monsieur BENSLIMANE Sidi Mohamed, Professeur à l'ESI de sidi bel abbès, Monsieur AMAR BENSABER Djamel, Maître de conférences A à l'ESI de sidi bel abbès, Monsieur BOUKLI HACENE Sofiane, Maître de conférences A à l'université de sidi bel abbès et Monsieur ALI CHERIF Moussa, Maître de conférences A à l'université de sidi bel abbès, pour leur disponibilité et acceptation d'examiner et de rapporter mon travail.

Je remercie aussi Monsieur BERRABAH Djamel, Maître de conférences A au département informatique, faculté des sciences exactes pour son soutien incessant et ses conseils toujours opportuns ainsi que pour sa grande gentillesse.

Je remercie aussi Monsieur BOUFARES Faouzi, Maître de Conférences HdR à l'Université Sorbonne Paris cité Paris 13 pour son soutien incessant et ses conseils toujours opportuns ainsi que pour sa grande gentillesse.

Enfin, je remercie toutes les personnes qui m'ont aidé durant mon processus de recherche et d'écriture de cette thèse.

Résumé

L'intégration de données consiste à combiner les données de plusieurs sources de données hétérogènes pour fournir une vue unifiée des données disponibles à une application ou à un utilisateur final. Cependant, la qualité des données intégrées peut être dégradée en raison de la présence de doublons avec des fautes d'orthographe, des abréviations, des valeurs contradictoires, etc. La résolution d'entité est une étape importante dans l'intégration de données et le nettoyage de données. Elle permet d'améliorer la qualité des données en identifiant les enregistrements qui représentent la même entité du monde réel.

Dans cette thèse, nous proposons un système de résolution d'entité entièrement automatique qui prend en charge les sources de données en anglais et en arabe, et indépendant au domaine dans lequel s'effectue la résolution d'entité. Pour l'automatisation, le système génère automatiquement un ensemble de données d'apprentissage, qui est ensuite utilisé pour apprendre un modèle de classification. Pour prendre en charge la langue arabe, le système utilise le système Unicode. Pour l'indépendance au domaine, le système n'utilise aucune connaissance préalable du domaine et il est évalué sur trois cas de test en arabe et quatre cas de test en anglais.

Mots clés : qualité de données, intégration des données, apprentissage automatique, classification, résolution d'entité, détection des doublons.

Abstract

Data integration involves combining data from multiple heterogeneous data sources to provide a unified view of the data available to an application or end user. However, the quality of the integrated data may be degraded due to the presence of duplicates with spelling errors, abbreviations, conflicting values, etc. Entity resolution is an important step in data integration and data cleansing. It improves data quality by identifying records that represent the same real-world entity.

In this thesis, we propose a fully automatic entity resolution system that supports data sources in English and Arabic, and independent of the domain in which entity resolution takes place. For automation, the system automatically generates a training set, which is then used to learn a classification model. To support the Arabic language, the system uses the Unicode system. For domain independence, the system does not use any prior knowledge of the domain and is evaluated on three test cases in Arabic and four test cases in English.

Keywords: data quality, data integration, machine learning, classification, entity resolution, duplicate detection.

ملخص

ينطوي إدماج المعطيات على جمع المعطيات من مصادر معطيات متعددة غير متجانسة لتوفير رؤية موحدة للمعطيات المتاحة لتطبيق ما أو للمستخدم النهائي .لكن، قد تتدهور جودة المعطيات المدمجة بسبب وجود نسخ مكررة مع الأخطاء الإملائية والاختصارات والقيم المتضاربة ، الخ . يعد كشف المعطيات المكررة خطوة مهمة في إدماج المعطيات وتنظيف المعطيات والذي يؤدي إلى تحسين جودة المعطيات من خلال تحديد المعطيات التي تمثل نفس الكيان الحقيقي.

في هذه الأطروحة، نقترح نظامًا أوتوماتيكيا لكشف المعطيات المكررة و ذلك لمصادر المعطيات الإنجليزية والعربية، و مستقلًا عن المجال الذي يتم فيه كشف المعطيات المكررة. بالنسبة للتشغيل الأوتوماتيكي، يقوم النظام بإنشاء مجموعة من معطيات التعلم أوتوماتيكيا، والتي يتم استخدامها بعد ذلك لتعلم نموذج التصنيف. لدعم اللغة العربية، يستعمل النظام المقترح نظام Unicode. بالنسبة للاستقلالية عن المجال ، لا يستخدم النظام أي معرفة مسبقة عن المجال، كما تم تقييم النظام المقترح في ثلاث حالات اختبار باللغة العربية وأربعة حالات اختبار باللغة الإنجليزية.

الكلمات المفتاحية: جودة المعطيات ، إدماج المعطيات ، االتعلم الأوتوماتيكي ، التصنيف ، الكشف عن المعطيات المكررة.

Sommaire

Liste des Figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Contexte

Le monde a connu une explosion du volume de données ces dernières années. Cela a ouvert des opportunités pour l'émergence de plusieurs nouvelles applications telles que l'extraction de connaissances, la fouille de données, l'apprentissage automatique, le big data, le e-commerce, etc. Ces applications intègrent des données provenant de plusieurs sources de données hétérogènes pour fournir aux utilisateurs une vue unifiée des données ou pour prendre des décisions au niveau de l'entreprise. Cependant, la qualité des données intégrées peut être dégradée en raison de la présence de doublons avec des fautes d'orthographe, des abréviations et des valeurs contradictoires. La qualité des données intégrées a un impact direct sur la qualité des résultats produits par ces applications et peut nuire gravement à l'efficacité des organisations et des entreprises (Batini & Scannapieco, 2006; Herzog, Scheuren & Winkler, 2007).

En général, Les sources de données à intégrer sont hétérogènes au niveau schéma et au niveau entité (Rahm & Do, 2000). L'hétérogénéité des schémas provient du fait que les sources de données peuvent utiliser un nombre différent d'attributs ou des noms différents pour le même attribut. L'hétérogénéité des entités est causée par : l'utilisation de valeurs différentes pour le même attribut de la même entité, l'inexistence d'identifiant d'entité (clé) global ou l'inexistence d'identifiants d'entité au niveau des sources de données à intégrer.

L'intégration de données provenant de plusieurs sources de données autonomes et hétérogènes comprend trois étapes principales (Doan, Halevy & Ives, 2012; Dong & Srivastava, 2015). La première étape est *l'alignement de schéma* qui permet d'identifier les éléments des schémas équivalents et en particulier les attributs équivalents. La deuxième étape est *la résolution d'entité* qui permet d'identifier les enregistrements qui représentent la même entité du monde réel. La troisième tâche est *la fusion de données*, elle consiste à fusionner les enregistrements représentant la même entité en un seul enregistrement complet et consistant.

La résolution d'entité est une étape importante dans *l'intégration des données* (Doan et al., 2012; Dong & Srivastava, 2015; Lenzerini, 2002), *le nettoyage de données* (Rahm & Do, 2000; Elmagarmid, Ipeirotis & Verykios, 2007) et *l'interconnexion des données liées* (Ngomo & Lyko, 2013; Kejriwal & Miranker, 2015). Elle permet d'améliorer la qualité des données intégrées en identifiant les enregistrements qui représentent la même entité du monde réel (Christen, 2012a). *Dans cette thèse, nous nous intéressons à la résolution d'entité dans le contexte d'intégration et de nettoyage des données.*

La résolution d'entité est appelée encore couplage d'enregistrements (record linkage) (Elfeky, Verykios, & Elmagarmid, 2002), détection des doublons (duplicate detection) (Elmagarmid et al., 2007; Naumann & Herschel, 2010), rapprochement de données (data matching) (Christen, 2012a), link discovery (Ngomo & Lyko, 2013), instance matching (Kejriwal & Miranker, 2015), etc. Pour le reste de cette thèse, nous utiliserons le terme *résolution d'entité* (ER).

1.2 Problématique

Une entité du monde réel peut être une personne, un produit, une entreprise ou tout autre objet. Des représentations multiples de la même entité du monde réel dans plusieurs sources de données sont particulièrement problématiques. Des exemples de représentations multiples (ce que l'on appelle des doublons) comprennent

plusieurs représentations du même client dans plusieurs sources de données, différentes représentations du même produit dans un catalogue en ligne, et plusieurs références de la même publication dans différents référentiel du Web. Un défi majeur dans la résolution d'entité est le manque d'identifiants d'entité uniques (clés) à cause de l'autonomie des sources de données à matcher. Par conséquent, la résolution d'entité doit être effectuée en comparant les attributs communs des sources de données à matcher, telles que des noms, des adresses ou des dates de naissance. Les valeurs de ces attributs sont souvent de qualité médiocre, car elles peuvent contenir des erreurs typographiques et des variations ou peuvent changer à travers le temps. Par conséquent, des mesures de similarités approximatives (telles qu'edit distance, Jaccard, Jaro, Jaro-Winkler, etc.) sont généralement utilisées pour comparer les paires d'enregistrements (Christen, 2012a; Elmagarmid et al., 2007). Le résultat de la comparaison de chaque paire d'enregistrements est un vecteur de comparaison qui contient les résultats des mesures de similarité. Enfin, les vecteurs de comparaison représentant les paires d'enregistrements sont classés en matches (où ils sont supposés correspondre à la même entité du monde réel) et non-matches (où ils sont supposés correspondre à des entités différentes) en utilisant un modèle de classification.

Il existe deux approches principales pour l'ER: les approches basées sur les règles et celles basées sur l'apprentissage (Christen, 2012a; Elmagarmid et al., 2007). Les approches basées sur les règles utilisent des règles de matching pour matcher les enregistrements. Ces règles sont élaborées par un expert humain et sont limitées à des domaines bien précis. Les approches basées sur les règles sont manuelles et ne peuvent être utilisées dans tous les domaines. Les approches basées sur l'apprentissage utilisent des données d'apprentissage (ensemble de vecteurs de comparaison étiquetés comme matches ou non matches) pour apprendre un modèle de classification (règles de matching ou fonction de combinaison numérique) qui est utilisé par la suite à prédire le statu des paires d'enregistrement à comparer. Dans

des situations réelles, les données d'apprentissage ne sont pas disponibles ou doivent être élaborées par un expert.

Compte tenu du coût de l'expertise du domaine, un système d'ER devrait présenter un degré élevé d'automatisation, ce qui signifie qu'il ne nécessite pas l'existence de données d'apprentissage ou des règles de matching.

Plusieurs systèmes d'ER ont été développés pour les sources de données en latin et en particulier l'anglais comme Febrl (Christen, 2008) et TAILOR (Elfeky et al., 2002). La majorité des systèmes proposés ne reconnaissent pas les caractères non latins et en particulier les caractères arabes parce qu'ils n'utilisent pas le système Unicode (Higazy, El Tobely, Yousef & Sarhan, 2013). D'un autre coté, les approches d'ER développées pour prendre en charge la langue arabe comme (Gueddah, Yousfi & Belkasmi, 2012) et (Ghafour, El-Bastawissy & Heggazy, 2011) se sont concentrées sur le matching des noms en arabe, ce qui signifie qu'ils sont limitées à un seul domaine et ne peuvent pas être utilisées dans plusieurs domaines de la langue arabe.

Compte tenu de l'existence de plusieurs domaines en langue arabe, un système d'ER qui prend en charge la langue arabe devrait être indépendant du domaine, plutôt que d'être conçu pour répondre aux besoins spécifiques d'un domaine particulier tels que les noms. Un système dépendant d'un domaine d *s'appuie sur un nombre important de connaissances préalables sur le domaine* d *et ne peut pas être utile dans un autre domaine.*

1.3 Objectifs du mémoire

L'objectif de cette thèse est de proposer *un système de résolution d'entité qui remplit simultanément trois exigences: l'indépendance au domaine, l'automatisation et la prise en charge des sources de données en langue arabe ainsi que celles en langue anglaise.* L'automatisation signifie que le système proposé devrait effectuer la résolution d'entité sans l'intervention d'un expert pour l'élaboration de données

d’apprentissage ou de règles de matching. L'indépendance au domaine signifie que le système proposé ne s'appuie pas sur des connaissances préalables du domaine dans lequel s’effectue la résolution d’entité.

1.4 Contribution

Dans cette thèse, nous proposons un système entièrement non supervisée (automatique) pour la résolution d'entité qui prend en charge les sources de données en anglais ainsi que celles en arabe. Le premier avantage du système proposé est qu'il puisse fonctionner sur n'importe quelles sources de données dans n'importe quel domaine. Le deuxième avantage du système proposé est sa capacité d’effectuer la résolution d’entité sur des sources de données en anglais ainsi que des sources de données en arabe. Le troisième avantage du système proposé est qu’il est entièrement automatique, ce qui signifie qu’il ne nécessite pas l’intervention d’un expert du domaine pour l’élaboration de données d’apprentissage ou de règles de matching.

Le système proposé suit plusieurs étapes. La première est une étape de prétraitement qui permet de prendre en charge les sources de données en anglais ainsi que celles en arabe.

Dans la deuxième étape, le système proposé génère ses propres données d’apprentissage d’une façon automatique. Comme contribution de cette thèse, nous proposons un algorithme non supervisé appelé *générateur de données d’apprentissage (TSG)*, qui utilise une heuristique pour générer les données d’apprentissage. Pour générer les exemples d’apprentissage (vecteurs de comparaison étiquetés), au lieu d’utiliser une seule mesure de similarité entre les valeurs d’attributs d’une paire d’enregistrements, nous utilisons pour chaque attribut plusieurs mesures de similarité afin de prendre en charge plusieurs types d’erreurs qui peuvent exister dans les sources de données en anglais et dans les sources de données en arabe.

Dans la troisième étape, les données d'apprentissage générées par le TSG sont utilisées par un classificateur (par exemple SVM ou arbres de décision) pour apprendre un modèle de classification. Enfin, le modèle de classification appris est utilisé pour classer les nouveaux vecteurs de comparaison non étiquetées en matches et non-matches.

1.5 Organisation du mémoire

Le reste de ce mémoire est organisé comme suit. Le chapitre 2 fournit une introduction aux concepts et technologies utilisés dans tous les chapitres restants. Il présente le concept de qualité des données, la procédure d'intégration des données ainsi que les étapes de résolution d'entité. Le chapitre 3 analyse les travaux connexes et les compare en se basant sur un ensemble de critères. Le chapitre 4 présente l'approche proposée, il décrit l'architecture du système proposé et ses différents modules en détails. Le chapitre 5 sera consacré à la présentation des données de test, l'évaluation du système proposé et sa comparaison avec d'autres approches d'ER. Enfin, le chapitre 6 résume le mémoire et recense quelques perspectives pour étendre ce travail.

Chapitre 2

Préliminaires

2.1 Introduction

Dans ce chapitre, nous présentons les connaissances de base nécessaires pour comprendre le problème de recherche abordé dans cette thèse. Nous fournissons un aperçu sur la qualité de données dans la section 2.2. Ensuite, la procédure d'intégration des données sera abordée dans la section 2.3. Avant de conclure, nous présentons les étapes de la résolution d'entité dans la section 2.4.

2.2 Qualité de données

La qualité des données est un terme générique décrivant à la fois les caractéristiques de données : complètes, fiables, pertinentes, à jour et cohérentes, mais aussi l'ensemble du processus qui permet de garantir ses caractéristiques. Le but est d'obtenir des données sans doublons, sans fautes d'orthographes, sans omission, sans variation superflue et conforme à la structure définie (Boufarès & Ben Salem, 2012).

Des données de mauvaise qualité peuvent entraîner des résultats erronés au moment de l'interrogation ou au moment de la prise de décision. Des statistiques récentes révèlent que les entreprises estiment généralement des taux d'erreur de données d'environ 1% à 5% (Redman, 1998) et que les données de mauvaise qualité coûtent aux entreprises US environ 600 milliards de dollars par an (Eckerson, 2002).

La qualité des données peut être mesurée par des indicateurs (mesures) tels que le nombre de valeurs distinctes, ou encore le nombre de valeurs en doubles. Ces indicateurs permettent d'évaluer des dimensions de la qualité des données telles que l'intégrité, la standardisation ou la déduplication.

Une dimension de qualité des données est une caractéristique utilisée pour classer les besoins de données. En fait, une dimension offre un moyen pour mesurer et gérer la qualité des données (McGilvray, 2008).

Il y'a un certain nombre de divergences dans la définition de la plupart des dimensions en raison de la nature contextuelle de la qualité des données (Batini & Scannapieco, 2006). Plusieurs dimensions ont été identifiées pour la qualité de données (Sidi et al., 2012) telles que l'exactitude (accuracy), la complétude (completeness), la consistance (consistency). Toutefois, il n'existe aucun accord général soit sur quel ensemble de dimensions définit la qualité des données, ou sur le sens exact de chacune des dimensions. Nous rappelons brièvement, ci-dessous, les définitions des dimensions essentielles.

Consistance

La consistance se réfère à la violation des règles sémantiques définies sur l'ensemble des données (Batini, Cappiello, Francalanci & Maurino, 2009). Dans le modèle relationnel, les contraintes d'intégrités sont des instanciations de ces règles sémantiques.

Complétude

La capacité d'un système d'information à représenter chaque état significatif du monde réel représenté (peu de valeurs absentes ou inconnues) (Batini et al., 2009).

Exactitude (Accuracy)

Les données sont exactes lorsque les valeurs des données stockées dans les sources de données correspondent à celles du monde réel (Batini et al., 2009).

Duplication

Les données sont répétées. L'entité est gérée par plusieurs systèmes d'informations sous des identifiants différents et donc sa vue n'est pas unifiée.

2.3 Intégration de données

L'intégration de données consiste à combiner les données de plusieurs sources de données hétérogènes pour fournir une vue unifiée des données disponibles à une application ou à un utilisateur final (Doan et al., 2012; Dong & Srivastava, 2015; Lenzerini, 2002). L'un des avantages des systèmes d'intégration de données est que l'utilisateur d'un tel système obtient une vue d'ensemble complète et concise de toutes les données existantes sans devoir accéder à toutes les sources de données séparément. Elle est complète car aucune entité ou donnée ne manque dans le résultat. Elle est concise car aucune entité n'est représentée deux fois et les données présentées à l'utilisateur sont sans contradiction (Bleiholder & Naumann, 2010). La figure 2.1 montre un exemple simple d'intégration de données qui contient des enregistrements de deux sources de données bibliographiques, à savoir, ACM et DBLP. Un système d'intégration de données doit en premier lieu identifier les paires d'enregistrements faisant référence aux mêmes publications, qui sont dans ce cas (564753, conf/sigmod/BumbulisB02) et (872806, conf/sigmod/LometT03). Ensuite, les fusionner pour avoir seulement trois enregistrements au lieu de cinq.

Dans les environnements distribués, les sources de données sont généralement caractérisées par divers types d'hétérogénéités qui peuvent généralement être classées en (i) hétérogénéités des schémas et (ii) hétérogénéités au niveau des instances.

Un système d'intégration de données complet est complexe et nécessite une collaboration entre plusieurs domaines de recherche. La figure 2.2 illustre les trois composants principaux d'un système d'intégration de données (Dong & Srivastava, 2015; Bleiholder & Naumann, 2006). L'alignement de schémas permet de résoudre

les hétérogénéités des schémas en détectant les éléments équivalents de schémas dans des sources de données différentes et en particulier les attributs équivalents. La résolution d'entité et la fusion de données permettent de résoudre les hétérogénéités au niveau des instances. La résolution d'entité permet la détection des descriptions d'entités qui font référence aux mêmes entités du monde réel dans des situations où les identifiants d'entité ne sont pas disponibles. Enfin, la fusion de données permet de résoudre le problème des valeurs d'attributs contradictoires et de représenter les descriptions d'entité équivalentes en une seule description.

ACM

ID	Title	Authors	Venue	Year
564753	A compact B-tree	Peter Bumbulis, Ivan T. Bowman	International Conference on Management of Data	2002
872806	A theory of redo recovery	David Lomet, Mark Tuttle	International Conference on Management of Data	2003

DBLP

ID	Title	Authors	Venue	Year
conf/sigmod/BumbulisB02	A compact B-tree	Ivan T. Bowman, Peter Bumbulis	SIGMOD Conference	2002
conf/sigmod/LometT03	A Theory of Redo Recovery	Mark R. Tuttle, David B. Lomet	SIGMOD Conference	2003
conf/sigmod/DraperHW01	The Nimble Integration Engine	Daniel S. Weld, Alon Y. Halevy, Denise Draper	SIGMOD Conference	2001

Figure 2.1: *Un exemple d'intégration de données dans le domaine bibliographique.*

2.3.1 Alignement de schémas

Les systèmes d'intégration de données doivent généralement faire face aux schémas hétérogènes des sources de données à intégrer. Afin de présenter les résultats d'une requête à l'utilisateur dans un schéma unique, les hétérogénéités schématiques doivent être gérées. Pour faire face à l'hétérogénéité et parvenir à l'interopérabilité, deux méthodes sont utilisées le matching des schémas (Rahm & Bernstein, 2001) et le mapping des schémas (Miller, Haas & Hernández, 2000).

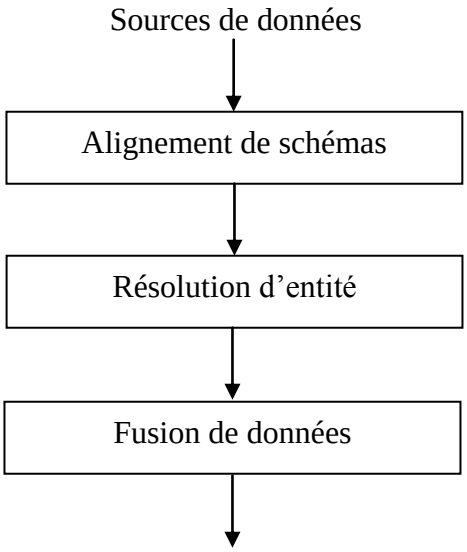

Figure 2.2: *composants principaux d'un système d'intégration de données.*

Comme le montre la figure 2.3, l'alignement des schémas comprend en général trois étapes (Dong & Srivastava, 2015): création d'un schéma de médiation, matching des schémas et mapping des schémas.

En premier lieu, un schéma médiateur est créé par un expert du domaine pour avoir une vue unifiée et virtuelle des sources de données à intégrer. Le schéma médiateur permet à un utilisateur final ou à une application de poser des requêtes au système d'intégration de données sans connaitre les schémas des sources de données à intégrer.

La deuxième étape dans l'alignement des schémas est le matching des schémas. Elle consiste à matcher les attributs de chaque schéma source avec les attributs correspondants du schéma médiateur. Dans de nombreux cas, la correspondance d'attribut est un à un; toutefois, un attribut du schéma médiateur peut parfois correspondre à la combinaison de plusieurs attributs d'un schéma source et inversement. Plusieurs techniques ont été proposées pour le matching d'attributs, en explorant la similarité entre les noms d'attributs, les types, les valeurs et les relations de voisinage entre les attributs. Ces techniques ont été revues par (Rahm & Bernstein, 2001), (Bellahsene, Bonifati & Rahm, 2011) et (Euzenat & Shvaiko, 2007).

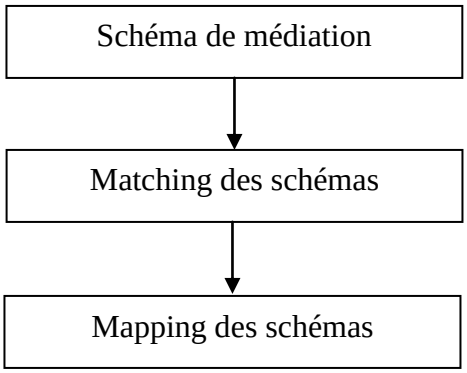

Figure 2.3: *Etapes d'alignement des schémas.*

Enfin, l'étape mapping des schémas utilise les correspondances d'attributs générées par l'étape matching des schémas pour construire un mapping des schémas entre chaque schéma source et le schéma médiateur. De tels mappings spécifient les relations sémantiques entre les contenus des différentes sources de données et seraient utilisés pour reformuler une requête utilisateur sur le schéma médiateur en un ensemble de requêtes sur les sources de données sous-jacentes. Il existe trois types de mapping de schémas: GAV (global as-view), LAV (local-as-view) et GLAV (global-local-as-view). GAV spécifie comment obtenir des données dans le schéma médiateur en interrogeant les données dans les schémas sources; En d'autres termes, les données du schéma médiateur peuvent être considérées comme une vue

de base de données des données sources. LAV spécifie les données source en tant que vue des données du schéma médiateur; Cette approche facilite l'ajout d'une nouvelle source de données avec un nouveau schéma. Enfin, GLAV spécifie à la fois les données du schéma médiateur et les données des sources de données sous forme de vues de données d'un schéma virtuel. Clio (Fagin et al., 2009) est un exemple d'outils qui permet de construire de manière semi-automatique des mapping de schémas en utilisant les résultats du matching d'attributs. La figure 2.4 montre un exemple d'alignement de schémas.

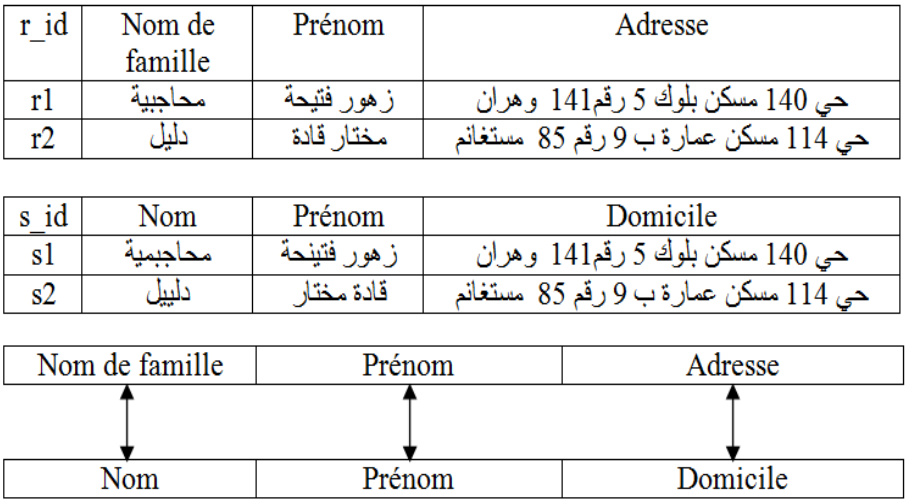

Figure 2.4: *Exemple d'alignement de schémas.*

2.3.2 Résolution d'entité

La seconde étape dans le processus d'intégration des données est l'étape de résolution d'entité (également appelée record linkage (Elfeky et al., 2002), détection des doublons (Elmagarmid et al., 2007; Naumann & Herschel, 2010), data matching (Christen, 2012a) et bien d'autres). L'objectif de cette étape est d'identifier les enregistrements qui représentent la même entité du monde réel. Si les enregistrements ont des identifiants uniques et sans erreur, tels que des numéros de sécurité sociale, la résolution d'entité devient un processus simple pouvant être

facilement effectué par une simple opération de jointure de base de données standard. Cependant, dans de nombreux scénarios pratiques, un tel identifiant unique n'existe pas. La figure 2.1 montre un exemple dans le domaine bibliographique dans lequel les sources de données ACM et DBLP utilisent des identifiants différents. Dans ce cas, le processus de résolution d'entité doit être effectué en utilisant des comparaisons approximatives des attributs correspondants des enregistrements. Il est également intéressant de noter que les mêmes données peuvent être représentées de différentes manières dans différentes sources de données en raison de facteurs tels que des conventions différentes, des erreurs typographiques, des valeurs manquantes ou encore aberrantes et obsolètes.

A première vue, la résolution d'entité est simple: comparer chaque paire d'enregistrements en utilisant une mesure de similarité. Si la similarité est supérieure ou égale à un seuil donné, les deux enregistrements sont déclarés comme étant des doublons. En réalité, il y a deux difficultés principales à résoudre: l'efficacité et la mise en échelle. L'efficacité est principalement influencée par la qualité de la mesure de similarité et le choix d'un bon seuil global de similarité. La mise en échelle est un problème dû au volume des sources de données à matcher. Si les sources de données sont très volumineuses, le calcul de la similarité de tous les pairs d'enregistrements peut devenir un obstacle. La section 2.4 détaille beaucoup plus cette étape de résolution d'entité.

2.3.3 Fusion de données

L'étape finale dans le processus d'intégration de données est la fusion de données (Bleiholder & Naumann, 2009). Elle consiste à consolider et à fusionner les paires ou les groupes d'enregistrements faisant référence aux mêmes entités en une seule représentation cohérente et complète pour chaque entité. La fusion de données permet l'enrichissement des enregistrements : renseigner les valeurs nulles, enrichir les données (une personne peut avoir deux numéros de téléphone différents ou plus), corriger les formats des données. Le principal défi de la fusion des données est la

résolution des conflits au niveau des données lorsque les enregistrements correspondant à une entité contiennent des valeurs d'attribut différentes (par exemple contradictoires). La résolution des conflits consiste à choisir une (ou plusieurs) valeur parmi un ensemble de valeurs contradictoires.

Les approches de résolution de conflits sont classées en deux grandes catégories (Bleiholder & Naumann, 2009) : *les approches basées sur les instances* et *les approches basées sur les métadonnées.* Les approches basées sur les instances considèrent uniquement les valeurs de données contradictoires pour choisir la valeur correcte comme le choix de la valeur la plus commune, le calcul d'une moyenne, le calcul du minimum ou le calcul du maximum parmi les valeurs contradictoires . Les approches basées sur les métadonnées choisissent les valeurs en se basant sur les métadonnées telles que la fraîcheur des données (la valeur la plus récente) ou la fiabilité d'une source par rapport à une autre.

2.4. Résolution d'entité

La résolution d'entité est le processus consistant à déterminer les enregistrements qui font référence à la même entité du monde réel. Une entité peut être tout objet existant dans le monde réel, comme une personne, un produit, une entreprise ou une publication. Les entités à résoudre peuvent résider dans des sources de données distribuées, généralement hétérogènes ou peuvent être stockées dans une source de données unique. Elles peuvent être matérialisés physiquement ou être le résultat d'une requête.

La résolution d'entité est une tâche importante dans *l'intégration de données* (Doan et al., 2012; Dong & Srivastava, 2015; Lenzerini, 2002), *le nettoyage de données* (Rahm & Do, 2000; Elmagarmid et al., 2007) et *l'interconnexion des données liées* (Ngomo & Lyko, 2013; Kejriwal & Miranker, 2015). Dans *l'intégration et le nettoyage des données*, la résolution d'entité est utilisée pour détecter les données dupliquées, erronées et obsolètes. Dans *l'interconnexion de*

données liées, la résolution d'entité est utilisée pour lier les instances dans le Web des données liées et en particulier la découverte des liens *owl:sameAs* entre les instances dans le Web des données liées. Dans *ce mémoire, nous nous intéressons à la résolution d'entité dans le contexte d'intégration et de nettoyage des données.*

Dans la littérature, le problème de résolution d'entité (Christen, 2012a) a été étudié sous différents noms: couplage d'enregistrements (record linkage) (Elfeky et al., 2002; Fellegi & Sunter, 1969; Winkler, 1994), détection des doublons (duplicate detection) (Elmagarmid et al., 2007; Naumann & Herschel, 2010; Bilenko & Mooney, 2003a), rapprochement de données (data matching) (Christen, 2012a), merge/purge (Hernandez & Stolfo, 1998), instance identification (Chua, Chiang & Lim, 2003), deduplication (Sarawagi & Bhamidipaty, 2002), entity matching (Köpcke & Rahm, 2010), link discovery (Ngomo & Lyko, 2013), instance matching (Kejriwal & Miranker, 2015), etc. Pour le reste de cette thèse, nous utiliserons le terme *résolution d'entité* (ER).

Divers chercheurs ont défini le problème d'ER. Newcombe, Kennedy, Axford and James (1959) ont défini l'ER comme « bringing together two or more separately recorded pieces of information concerning a particular individual or family ». Fellegi and Sunter (1969) ont décrit l'ER comme « the problem of recognizing those records in two files which represent identical persons, objects, or events ». L'ER a également été définie par Winkler (1999) comme « the methodology of bringing together corresponding records from two or more files or finding duplicates within files », et par Elmagarmid et al. (2007) comme « to identify records in the same or different data sets that refer to the same real-world entity, even if the records are not identical ».

La résolution d'entité fait partie du processus d'intégration des données. Elle peut donc avoir lieu chaque fois que différentes représentations de la même entité du monde réel (doublons) sont intégrées et doivent être combinées. La résolution d'entité a été utilisée de manière intensive dans divers domaines, notamment la

santé, les recensements nationaux, la prévention de la criminalité et de la fraude, et les élections nationales.

Système d'information de santé: dans les services de santé, les informations personnelles et médicales sont collectées chaque fois qu'une personne entre en contact avec des services de santé. Si ces informations sont matchées entre différents services de santé, elles peuvent être analysées pour améliorer le système de santé.

Agences nationales de statistiques: les agences nationales de statistiques sont chargées de collecter et de publier des données relatives à divers domaines tels que la population, l'économie, la santé, l'éducation, la culture ou la politique. Les données statistiques générées par ces agences sont fournies aux gouvernements et aux organisations pour améliorer les procédures de prise de décision. Les agences nationales de statistiques ont toujours utilisé la résolution d'entité lors de la réalisation d'enquêtes statistiques pour améliorer la qualité des données collectées (Winkler, 1995). Le bureau de recensement aux Etats Unis a été l'un des premiers à adopter des techniques de résolution d'entité et a également joué un rôle clé dans la recherche sur l'ER (Winkler, 2005).

Prévention de la criminalité et de la fraude: avec la croissance rapide des technologies modernes, la fraude augmente considérablement, entraînant la perte de milliards de dollars dans le monde (Bolton & Hand, 2002). De nos jours, divers systèmes d'information, applications statistiques et techniques d'exploration de données sont utilisés pour détecter la fraude dans de nombreuses entreprises et organisations (Chen et al., 2004; Phua, Lee, Smith & Gayler, 2010). Les techniques d'ER sont utilisées pour améliorer la qualité des données utilisées par ses systèmes.

Autres applications: l'ER a été appliquée dans divers autres domaines. Les moteurs de recherche utilisent les techniques d'ER pour identifier les documents qui couvrent des sujets similaires (Hajishirzi, Yih & Kolcz, 2010). De nombreux sites de comparaison utilisent également l'ER pour identifier des produits similaires afin de pouvoir fournir des comparaisons de prix (Bilenko, Basil, & Sahami, 2005; Köpcke, Thor, Thomas & Rahm, 2012). Les bibliothèques numériques, quant à elles,

identifient des articles similaires pour améliorer les possibilités de recherche et pour dédupliquer leurs ensembles de données (Yan, Lee, Kan & Giles, 2007). La déduplication peut également être utilisée dans les entreprises pour supprimer les doublons des listes de diffusion de leurs clients, ce qui réduit les coûts d'envoi des courriers publicitaires aux clients (Naumann & Herschel, 2010).

Le processus de résolution d'entité est composé de plusieurs étapes (Christen, 2012a; Elmagarmid et al., 2007) comme illustré dans la figure 2.5. Dans l'étape prétraitement, les données sont unifiées, normalisées et standardisées en une forme bien définie. La deuxième étape, l'indexation, vise à réduire le nombre de comparaisons nécessaires pour effectuer l'ER en regroupant les enregistrements en blocs (groupes). L'ER est alors limitée aux enregistrements du même bloc (paires d'enregistrements candidates). Dans l'étape comparaison, les paires d'enregistrements candidates sont comparées en utilisant des mesures de similarité. Cette étape retourne pour chaque paire d'enregistrement à comparer, un vecteur de comparaison contenant les résultats des mesures de similarité (valeurs comprises entre 0 et 1). Enfin, Les vecteurs de comparaison générés sont classés en matches et non matches dans l'étape classification en utilisant un modèle de classification (ensemble de règles ou une fonction de combinaison numérique). Les différentes étapes du processus d'ER sont décrites plus en détail dans les sous sections suivantes.

2.4.1. Prétraitement

La plupart des données collectées dans le monde réel sont de mauvaise qualité et peuvent inclure des erreurs, des valeurs manquantes ou des valeurs incohérentes (Batini & Scannapieco, 2006; Herzog et al., 2007). De plus, les sources de données à matcher peuvent avoir des structures et des formats différents. Le problème des structures différentes (ou hétérogénéités structurelles) est résolu dans l'étape alignement de schémas qui est la première étape de l'intégration de données, alors que le problème des formats différents de données est résolu dans l'étape nettoyage

de données. La qualité des données est vitale pour le processus d'ER et affecte de manière significative ses résultats (Herzog et al., 2007; Christen, 2012a; Elmagarmid et al., 2007). L'étape de prétraitement est effectuée en utilisant des techniques de nettoyage et de normalisation de données qui reposent fortement sur des tables de consultation (look-up Tables). Ces tables contiennent, par exemple, des noms personnels et leurs variantes, ainsi que des fautes d'orthographe courantes, des noms de banlieue, de ville ou d'état et des codes postaux d'un pays donné. D'autres techniques utilisent des règles de standardisation élaborées manuellement (Christen, 2012a). Les tables et les règles sont dépendantes du domaine et de la langue des sources de données à matcher (Yousef, 2015).

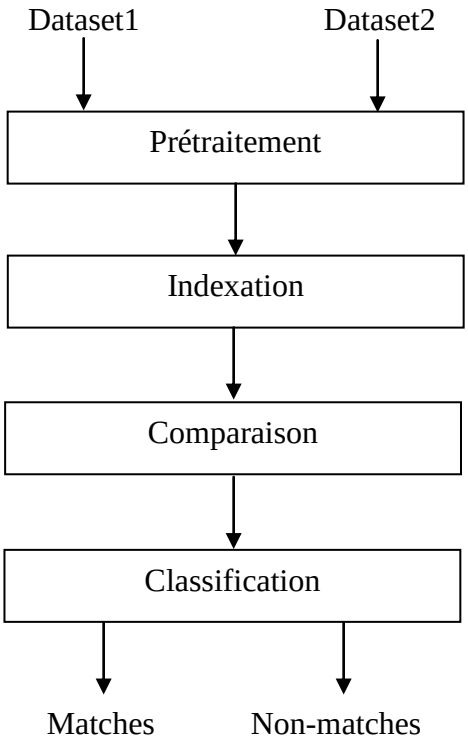

Figure 2.5: *composants principaux d'un système de résolution d'entité.*

Les principales approches couramment utilisées pour nettoyer et standardiser les données lors de la première étape du processus d'ER (Elmagarmid et al., 2007) sont les suivantes:

Standardisation: Avant que les valeurs d'attributs ne soient utilisées dans le processus d'ER, les attributs doivent être convertis en un format standard. La standardisation consiste généralement à supprimer les caractères indésirables, à corriger les fautes d'orthographe, à élargir les abréviations et à s'assurer que les mêmes systèmes de codage et unités de mesure sont utilisés dans les sources de données à matcher (Christen, 2012a; Rahm & Do, 2000; Winkler, 1995).

Parsing: L'analyse syntaxique vise à identifier et à isoler les composants individuels dans les valeurs d'attributs en composants cohérents et bien définis (Christen, 2012a; Elmagarmid et al., 2007; Winkler, 1995). Un exemple de parsing consiste à diviser le numéro de téléphone en un code pays, un indicatif régional et un numéro de téléphone.

Transformation de données: Dans cette étape, les valeurs des attributs sont converties dans leur forme correcte (Christen, 2012a; Elmagarmid et al., 2007; Rahm & Do, 2000). Les conversions courantes dans cette étape incluent le décodage des valeurs d'attribut codées dans leur forme d'origine, la vérification de la plage et la vérification des dépendances. Un exemple de vérification de dépendance consiste à valider le nom d'une ville et son code postal. Si les deux valeurs ne sont pas cohérentes, cela correspond généralement à une erreur de saisie de données qui doit être corrigé en modifiant une des deux valeurs d'attribut pour qu'elle soit cohérente.

La quantité de recherches dans le domaine du prétraitement des données (nettoyage et normalisation) est étonnamment faible. La plupart des systèmes d'ER ignorent cette étape. Une des raisons à cela pourrait être que le prétraitement des données est une tâche très spécifique à un domaine qui implique des quantités importantes d'expertise du domaine et de personnalisation et d'intervention manuelle (Christen, 2012a).

2.4.2 Indexation

Étant donné deux sources de données R et S, représentées de manière générique sous la forme d'ensembles d'enregistrements, un système d'ER naïf évaluerait toutes les paires d'enregistrements possibles. Chaque enregistrement de R est comparé en détails avec tous les enregistrements de S en utilisant des mesures de similarité qui sont coûteuses. En supposant un coût constant par évaluation, le temps d'exécution serait O ($|R||S|$). Pour des sources de données volumineuses, il est impossible de comparer toutes les paires d'enregistrements possibles (Christen & Goiser, 2007). L'étape d'indexation a pour but de réduire le nombre important de comparaisons potentielles en supprimant autant que possible les paires d'enregistrements qui correspondent aux non-matches et en regroupant les paires d'enregistrements qui partagent certaines caractéristiques en blocks. L'ER est alors limitée aux enregistrements du même bloc. Le résultat de l'indexation est un ensemble de paires d'enregistrements appelées *paires d'enregistrements candidates* qui seront comparées en détails dans l'étape comparaison. Un exemple d'indexation simple est de regrouper tous les enregistrements qui partagent les trois premières lettres du nom en blocs.

L'indexation utilise une fonction appelée clé de blocage (blocking key) pour regrouper les entités similaires en blocs. Étant donné une source de données R représentée comme un ensemble d'enregistrements, une clé de blocage K est une fonction qui prend un enregistrement de R comme entrée et renvoie un ensemble non vide de valeurs, appelé valeurs de clé de blocage (BKVs) de l'enregistrement. Sans perte de généralité, les valeurs renvoyées par la clé de blocage K sont supposés être des chaînes de caractères. La clé de blocage est généralement déterminée manuellement.

Pour la plupart des techniques d'indexation, un index inversé (Witten, Moffat & Bell, 1999) est utilisé. Les valeurs de clé de blocage deviendront les clés de l'index inversé, et les identificateurs de tous les enregistrements ayant la même valeur de clé de blocage seront insérés dans la même liste d'indexation inversée.

Lors de l'ER de deux sources de données, une structure de données d'index distincte est créée pour chaque source ou une structure de données unique avec des valeurs de clé communes est générée. Dans le second cas, chaque identificateur d'enregistrement doit inclure un indicateur indiquant la provenance de l'enregistrement.

Dans ce qui suit, nous présentons les principales approches d'indexation (Christen, 2012b; Draisbach & Naumann, 2009; Papadakis, Svirsky, Gal & Palpanas, 2016).

Blocage standard

Le blocage standard ou traditionnel (Fellegi & Sunter, 1969) représente chaque enregistrement par une valeur de clé de blocage (BKV). Tous les enregistrements avec la même BKV sont insérés dans le même bloc.

La manière de définition des clés de blocage influencera la qualité et le nombre des paires candidates générées. Un inconvénient majeur est que des clés de blocage dont les valeurs contiennent des erreurs entraineront une diminution de la complétude des paires (les matches sont affectés à des blocs différents et ne seront pas détectés dans la phase de comparaison). Un second inconvénient est que les tailles des blocs générés dépendent de la distribution de fréquence des BKVs, et cela peut entrainer la génération de blocs larges contenant un nombre important de comparaisons.

La figure 2.6 illustre la fonctionnalité du blocage standard pour quatre enregistrements. La clé de blocage utilisé est une concaténation des trois premières lettres du nom avec les trois premières lettres du prénom. Dans cet exemple, deux blocs sont générés et qui sont représentés par les deux BKVs : GEO_CAR et WIL_GEO. Dans l'exemple de la figure 2.6, l'ensemble final des paires d'enregistrements candidates sera {(R1,R3), (R2,R4)}.

ID	Nom	Prénom	BKVs
R1	WILLIAM	GEORG	WIL_GEO
R2	GEORGE	CARTIER	GEO_CAR
R3	WILIAM	GEORGE	WIL_GEO
R4	GEORG	CARTER	GEO_CAR

WIL_GEO
R1 R3

GEO_CAR
R2 R4

Figure 2.6: *Exemple de blocage standard avec BK étant les trois premières lettres du non concaténées avec les trois premières lettres du prénom*

Voisinage trié (Sorted Neighborhood)

Une autre méthode de blocage influente qui a été fondamentalement conçue pour garantir une limite de la taille de l'ensemble des paires candidates est la méthode du voisinage trié en anglais Sorted Neighborhood (SN) (Hernández & Stolfo, 1995). Cette méthode utilise les mêmes BKVs que le blocage standard, mais construit des blocs en se basant sur leur similarité au lieu de leur égalité. Elle trie les BKVs par ordre alphabétique et glisse une fenêtre d'un nombre fixe d'enregistrements w ($w > 1$) sur les valeurs triées. À chaque étape, Les paires d'enregistrements candidates sont générées uniquement à partir des enregistrements de la fenêtre en cours.

La fenêtre glissante a deux implications pour la génération des paires candidates. Tout d'abord, les enregistrements qui n'ont pas la même BKV peuvent être insérés dans le même bloc. Deuxièmement, certains enregistrements avec la même BKV peuvent être insérés dans des blocs différents.

L'algorithme SN est sensible à la taille de la fenêtre w, les petites fenêtres entraînent un rappel faible, mais les grandes entrainent de nombreuses comparaisons et une précision faible. Plusieurs variations de cette technique existent (Christen, 2012b). Les principales différences entre ces versions et la version d'origine sont les différentes méthodes pour régler le paramètre de la fenêtre glissante (e.g., adaptatif ou constant) pour des performances maximales (Yan et al., 2007). Une autre extension de l'algorithme SN est l'approche Multi-Pass (Hernández & Stolfo, 1998)

qui vise à surmonter le problème de clé de blocage unique sensible aux erreurs. Elle consiste à exécuter l'algorithme plusieurs fois avec plusieurs clés de blocage différentes.

La figure 2.7 illustre la fonctionnalité de l'indexation voisinage trié pour les quatre enregistrements présentés dans la figure 2.6. La clé de blocage utilisée est le prénom et la taille de la fenêtre *w*=3. L'intervalle de la fenêtre représente les paires d'enregistrements sélectionnés comme candidats pour chaque glissement de la fenêtre. Dans l'exemple de la figure 2.7, l'ensemble final des paires d'enregistrements candidates sera {(R4,R2), (R4,R1), (R2,R1), (R2,R3), (R1,R3)}.

Position fenêtre	BKVs (Prénom)	ID
1	CARTER	R4
2	CARTIER	R2
3	GEORG	R1
4	GEORGE	R3

Intervalle fenêtre	Paires candidates
1-3	(R4,R2), (R4,R1), (R2,R1)
2-4	(R2,R1), (R2,R3), (R1,R3)

Figure 2.7: *Exemple d'indexation voisinage trié avec BK étant prénom à partir de la figure 2.6 et taille de la fenêtre w=3.*

Indexation Q-grams

L'indexation Q-grams (Gravano et al., 2001) s'appuie sur les BKVs du blocage standard, mais les transforme dans un format garantissant une plus grande résilience au bruit (erreurs des valeurs d'attributs). Au lieu d'utiliser les BKVs entières, elle considère leurs q-grams (c'est-à-dire des sous-chaines de *q* caractères) et crée des blocs sur leur égalité. Cette méthode présente deux inconvénients. Elle génère un nombre élevé de blocs et chaque bloc contient un nombre élevé de comparaisons.

La version étendue de l'indexation Q-grams (Christen, 2012b) permet de produire des BKVs plus discriminantes. Elle vise à réduire le nombre de comparaisons dans les blocs résultants. Plus précisément, au lieu des q-grams individuels, cette version utilise les combinaisons issues de la concaténation d'au moins L q-grams. L=max(1, [k .T]), où k est le nombre de q-grams dans la BKV d'origine et T est un seuil entre 0 et 1 défini par l'utilisateur.

L'avantage de cette méthode est qu'elle permet de surmonter les erreurs et les variations des BKVs, ce qui entraîne une complétude des paires plus élevée et donc une qualité de matching améliorée. Son inconvénient est que la génération récursive d'un grand nombre de sous-chaines est coûteuse en calculs, en particulier pour les valeurs de BKVs longues.

Canopy Clustering

L'indexation Canopy Clustering (McCallum, Nigam & Ungar, 2000) utilise les mêmes BKVs de l'indexation traditionnel ou l'indexation Q-grams, mais crée des blocs en se basant sur leur similarité, au lieu de leur égalité. Les mesures de similarité couramment utilisées pour calculer les similarités entre les BKVs sont Jaccard et TF-IDF (pour plus de détails sur ces deux mesures voir l'annexe). Dans cette méthode, les blocs (clusters ou canopies) sont générés en utilisant les étapes suivantes: (1) Initialement, tous les enregistrements sont insérés dans un pool de candidats P, (2) Un enregistrement p est choisi de manière aléatoire et ensuite supprimé du pool P. Cet enregistrement devient le centre d'un nouveau cluster, (3) Les BKVs de p sont comparées avec celles de tous les autres enregistrements de P. Tous les enregistrements ayant une valeur de similarité supérieure à un seuil $w1$ sont insérés dans le même bloc de p, (4) Tous les enregistrements ayant une valeur de similarité supérieure à un seuil de similarité $w2$ (avec $w2 \geq w1$) sont supprimés du pool P. Les étapes 2 à 4 sont répétées jusqu'à ce que le pool des candidats P soit vide.

Une autre variante de Canopy Clustering a été proposée dans (Christen, 2012b), elle est basée sur le plus proche voisin. Au lieu de deux paramètres de seuil globaux, deux paramètres des plus proches voisins sont utilisés: *n1* et *n2*. *n1* est le nombre d'enregistrements les plus proches du centre *p* choisi aléatoirement (selon les similarités des BKVs) qui seront insérés dans le bloc de *p*, et *n2* est le nombre d'enregistrements les plus proches qui seront retirés du pool d'objets candidats. Cette approche permet d'avoir des blocs de tailles similaires, avec la taille maximale connue auparavant qui est *n1*.

Apprentissage des clés de blocage

En général, Les sources de données à matcher sont de mauvaise qualité et contiennent des erreurs et des variations pouvant affecter le processus de sélection des clés de blocage. De plus, la sélection des clés optimales peut dépendre du domaine, et plusieurs clés de blocage potentielles peuvent être définies en se basant sur plusieurs attributs (Christen, 2012b). Cela rend la sélection manuelle des clés de blocage problématique. La plupart des techniques d'indexation existantes (telles que celles décrites précédemment) nécessitent une sélection manuelle des clés de blocage par un expert. Une alternative serait d'apprendre automatiquement ces clés de blocage.

Plusieurs techniques automatiques utilisent des données d'apprentissage pour apprendre les clés de blocage optimales. Les données d'apprentissage sont constituées de vecteurs de comparaison (représentant des paires d'enregistrements) étiquetées comme des matches ou des non-matches. Les vecteurs de comparaison étiquetés comme matches (non-matches) sont aussi appelés exemples positifs (exemples négatifs).

Bilenko, Kamath and Mooney (2006) ont étudié l'utilisation d'une forme normale disjonctive (Aizenstein & Pitt, 1995) pour créer des blocs. Le but était d'éliminer les clés de blocage qui couvrent de nombreuses paires négatives (classées comme des vraies non-matches dans l'ensemble de données d'apprentissage) et de

sélectionner les clés qui couvrent les paires positives les plus possibles (classées comme vraies matches dans l'ensemble de données d'apprentissage).

Michelson and Knoblock (2006) ont proposé une approche pour l'apprentissage de *schéma de blocage*. Un schéma de blocage est représenté par une conjonction d'un attribut et d'une méthode d'indexation ({attribut, méthode}). L'objectif de l'approche est de maximiser le rapport de réduction (le rapport entre les paires d'enregistrements candidats générés et toutes les paires d'enregistrements possibles). Leur algorithme ajoute automatiquement des paires {attribut, méthode} aux conjonctions jusqu'à ce qu'un rapport de réduction optimal est atteint.

Une autre approche supervisée a été récemment proposée par Vogel and Naumann (2012). Les auteurs utilisent les combinaisons des uni-grams des valeurs d'attribut (c'est-à-dire une combinaison de caractères uniques provenant d'attributs différents) en tant que clés de blocage. Ensuite, la précision et l'efficacité des blocs générés sont utilisées pour apprendre l'ensemble des clés de blocage optimales.

Indexation automatique (non supervisée)

Les approches automatiques présentées précédemment nécessitent l'existence de données d'apprentissage. Cependant, ces données étiquetées ne sont pas toujours disponibles et leur élaboration est généralement coûteuse. Pour surmonter ce problème, plusieurs techniques non supervisées de sélection de clés de blocage ont été développées.

Kejriwal and Miranker (2013) ont proposé un algorithme non supervisé pour l'apprentissage des clés de blocage. Dans sa première phase, l'algorithme génère un ensemble de données d'apprentissage en utilisant un schéma de pondération TF-IDF. Chaque enregistrement est considéré comme un document, deux enregistrements sont considérés comme matches si leur TF-IDF dépasse un certain seuil *s1* et non-matches si leur TF-IDF est inferieur à un certain seuil *s2* (avec $s2<s1$). Dans la deuxième phase, l'algorithme utilise les données d'apprentissage générées pour apprendre les clés de blocage optimales en utilisant un critère de discrimination de

Fisher (Gu, Li & Han, 2012). Le score de Fisher est utilisé pour sélectionner la clé de blocage optimale (dont le score Fisher est le plus élevé). L'inconvénient de cette approche est qu'elle nécessite la spécification des deux seuils *s1* et *s2*.

Deux autres approches non supervisées ont été récemment proposées par Papadakis, Ioannou, Palpanas, Niederee and Nejdl (2013): *Attribute Clustering* et *Token blocking*. Dans Token blocking, chaque token *t* crée un bloc contenant tous les enregistrements contenant *t* avec la condition que *t* soit partagé par au moins 2 enregistrements, quels que soient les noms d'attributs associés. Token blocking présente deux avantages majeurs en termes de performances: premièrement, elle peut être efficacement mise en œuvre à l'aide d'un index inversé, même dans le cas de sources de données volumineuses. Deuxièmement, elle est robuste au bruit et à l'hétérogénéité, car la probabilité que deux enregistrements identiques ne se partagent aucun token est très faible. En effet, cela ne peut être le cas que lorsque deux enregistrements qui matchent n'ont aucun token en commun, une situation très improbable pour les enregistrements décrivant la même entité du monde réel.

La figure 2.8 montre un exemple d'indexation Token Blocking. Pour chaque token, un bloc est créé. Pour les tokens WILLIAM, WILIAM, CARTIER et CARTER, aucun bloc n'est créé car chacun d'eux est contenu dans un seul enregistrement. Dans l'exemple de la figure 2.8, l'ensemble final des paires d'enregistrements candidates sera {(R2,R3), (R1,R4), (R1,R3), (R2,R4)}.

Attribute Clustering partitionne les noms d'attributs en un ensemble de *K* clusters disjoints en fonction de la similarité de leurs valeurs et applique Token blocking à l'intérieur de chaque cluster. Chaque token *t* des valeurs d'un cluster *k* de *K* crée un bloc avec tous les enregistrements contenant *t* dans un attribut appartenant à *k*. L'avantage de ces deux méthodes d'indexation est qu'elles ne nécessitent pas des données d'apprentissage ou un ensemble de fonctions d'indexation.

ID	Nom	Prénom	Code postal
R1	WILLIAM	GEORG	104
R2	GEORGE	CARTIER	141
R3	WILIAM	GEORGE	104
R4	GEORG	CARTER	141

GEORGE	GEORG	104	141
R2 R3	R1 R4	R1 R3	R2 R4

Figure 2.8: *Exemple d'indexation Token Blocking.*

2.4.3 Comparaison

Bien que l'ensemble des enregistrements candidats soit censé contenir la plupart, sinon la totalité des matches des deux sources de données, il contient également un nombre important de non-matches. Les paires d'enregistrements appartenant au même bloc nécessitent une comparaison plus détaillée pour décider si les enregistrements se réfèrent ou non à la même entité du monde réel.

Étant donné que les données réelles sont de mauvaise qualité et contiennent souvent des erreurs typographiques et des variations, l'utilisation de mesures de comparaison exactes entraîne probablement une mauvaise qualité de matching. Par conséquent, des fonctions de comparaison approximatives doivent être utilisées pour résoudre le problème d'ER. Cette étape de comparaison est généralement basée sur la comparaison des valeurs de plusieurs attributs en utilisant des mesures de similarité qui renvoient pour chaque paire d'enregistrements comparée *un vecteur de comparaison* contenant les résultats des comparaisons (valeurs comprises entre 0 et 1).

Les chaînes de caractères sont le type de données le plus important dans de nombreux problèmes, car de nombreux attributs sont représentés par des chaînes de caractères, tels que les noms, les adresses, les descriptions, etc. Une chaîne de

caractères peut être décomposée en une séquence de tokens, de caractères ou de n-grams. Les mesures de similarité de chaîne de caractères existantes utilisent les tokens, les caractères, ou les n-grams pour calculer la similarité entre deux chaines de caractères. Les q-grams d'une chaîne *s* sont les sous-chaines de *q* caractères de *s*.

Plusieurs mesures de similarité ont été proposées dans la littérature (en particulier pour les chaines de caractères) et ont été classées en plusieurs catégories (Elmagarmid et al., 2007; Cohen, Ravikumar & Fienberg, 2003). Chaque catégorie est adaptée à un type d'erreur. Par exemple, les mesures de similarité basée sur les caractères (comme Levenshtein, Jaro et Jaro-Winkler) gèrent les erreurs typographiques, alors que les mesures de similarité basée sur les tokens (comme Jaccard, q-grams et TF-IDF) gèrent correctement les erreurs de réarrangement des mots.

2.4.4 Classification

Dans l'étape de classification, les vecteurs de comparaison (représentant les paires d'enregistrements candidats) générés lors de l'étape de comparaison sont classés en général en matches et non-matches (dans certain cas, et en matches potentielles). Plusieurs approches ont été proposées pour l'étape de classification. Ces approches peuvent être classées en deux grandes catégories (Christen, 2012a; Elmagarmid et al., 2007) : les approches basées sur l'apprentissage et les approches manuelles (les approches basées sur les règles ou les approches numériques).

a- Approches numériques

Les approches numériques combinent les valeurs de similarité des paires d'enregistrements (*r*, *s*) déterminées par les différentes mesures de similarité en utilisant une fonction de combinaison numérique *f*. La valeur numérique *v* renvoyée par la fonction *f* est mappée à une décision de match si *v* dépasse un certain *seuil* et à une décision de non-match dans le cas contraire. Les fonctions de combinaison typiques sont la moyenne et la moyenne pondérée (*MP*) définie comme suit :

$$MP = \frac{\sum_{i=1}^{n} x_i \times p_i}{n} \quad (2.1)$$

P = $\{p_i\}$ est un vecteur de poids et X = $\{x_i\}$ est un vecteur de comparaison. Lorsque P est le vecteur des uns (1,….,1), la fonction de combinaison numérique devient la moyenne normale. L'introduction des poids dans le calcul de la moyenne permet de favoriser les attributs les plus discriminatifs par rapport aux autres. Les approches numériques nécessitent la spécification d'une fonction de combinaison numérique par un expert. Alternativement, les techniques d'apprentissage automatique (Han, Pei & Kamber, 2011) peuvent être utilisées pour apprendre la fonction de combinaison numérique optimale à partir des données d'apprentissage si elles existent.

b- Approches basées sur les règles

Les approches basées sur les règles utilisent des langages déclaratifs pour spécifier un ensemble de règles nommées *règles de matching* qui sont utilisées pour décider si deux enregistrements sont des matches ou non-matches. Ces règles sont liées aux similarités attribuées à chaque valeur d'attribut dans les vecteurs de comparaison des enregistrements comparés (qui sont générés à l'étape de comparaison). Une règle de matching simple *R* consiste en la conjonction logique de *n* conditions de matching: $R = \bigwedge_{i=1}^{n} C_i$ tel que C_i est une condition de matching, elle peut être une condition qui stipule que la valeur d'une seule mesure de similarité *sim* soit supérieure à un *seuil* bien défini : $C = sim_j\ (r[a_i], s[a_i]) > seuil$. Par exemple, deux livres peuvent être considérés comme identiques si les similarités de leurs attributs titres et auteur dépassent toutes les deux certains seuils. De telles règles de matching simples composées uniquement de conditions de seuil pour le matching de valeur d'attributs sont connues sous le nom de *similarity joins* (Arasu, Ganti & Kaushik, 2006). D'autres règles de matching complexes peuvent être aussi définies en combinant

plusieurs règles de matching simples, par exemple, par disjonction : $\bigvee_{i=1}^{n} R_i$ .De nombreuses approches comme (Benjelloun et al., 2009) et (Boufares, Salem, Rehab & Correia, 2013) ont utilisé ce principe qui nécessite l'intervention d'un expert humain pour spécifier de telles règles de matching d'une façon déclarative. Alternativement, les techniques d'apprentissage automatique (Han at al., 2011) peuvent être utilisées pour apprendre l'ensemble optimal de règles de matching à partir des données d'apprentissage si elles existent.

c- Approches basées sur l'apprentissage

Les approches de combinaison numérique ainsi que celles basées sur les règles nécessitent l'intervention d'un humain pour la spécification d'une configuration qui comprend les mesures de similarité, les poids pour les attributs ainsi que les seuils à utiliser. La configuration choisie peut avoir un impact important sur la qualité globale du matching. Cela rend ces approches dépendantes du domaine d'intérêt (non adaptatives) et même les experts trouveront difficile et fastidieux de déterminer un bon choix.

Depuis le début des années 2000, l'apprentissage automatique a été activement utilisé pour l'ER (Elmagarmid et al., 2007). Un système d'ER basé sur l'apprentissage apprend de manière adaptative de bonnes règles de matching ou une fonction de combinaison numérique à partir de données d'apprentissage (ensemble de vecteurs de comparaison préalablement étiquetés comme étant matches ou non-matches) pour les approches supervisées et semi-supervisées. Les données d'apprentissage sont utilisées pour trainer un classificateur (par exemple les arbres de décision ou SVM) et apprendre un modèle de classification (règles de matching ou fonction de combinaison numérique). Enfin, le modèle de classification appris dans l'étape d'apprentissage est utilisé pour classer les vecteurs de comparaison non étiquetés. Plusieurs approches d'ER ont été proposées (Kopcke, Thor & Rahm,

2010). L'inconvénient de ces approches est qu'elles nécessitent l'existence de données d'apprentissage qui ne sont pas disponibles dans des situations réelles.

2.5 Conclusion

De nos jours, l'intégration de données est utilisée dans plusieurs domaines pour présenter à l'utilisateur final ou à une application une vue unifiée de plusieurs sources de données distribuées et hétérogènes. La résolution d'entité est une étape importante dans l'intégration et le nettoyage des données, elle permet d'améliorer la qualité des données en détectant les descriptions qui représentent la même entité du monde réel. Dans ce chapitre, nous avons présenté les connaissances de base et la terminologie nécessaires pour comprendre le problème de recherche abordé dans cette thèse qui est la résolution d'entité. Une étude sur les différents travaux de recherche sur la résolution d'entité est présentée dans le chapitre suivant.

Chapitre 3

Etat de l’art

3.1 Introduction

Le problème de résolution d’entité a été défini à l'origine par Newcombe et al. (1959) et a été formalisé par Fellegi and Sunter (1969) dix ans après. Depuis les années 1990, le développement de l’informatique a contribué à la promotion de la recherche sur l’ER (Winkler, 1990). Depuis lors, ce problème a été considéré au sein de différentes communautés, y compris la communauté d’intelligence artificielle et la communauté de base de données. Plusieurs approches et frameworks ont été proposés pour l'ER. La plupart des approches se focalisent sur l’étape de classification.

3.2 Approches à base de règles

SERF (Stanford Entity Resolution Framework) (Benjelloun et al., 2009) est un framework générique d’ER qui vise à améliorer la mise en échelle. Les mesures de similarité sont considérées comme des boites noires. Différents algorithmes sont proposés pour minimiser le nombre d'appels à ces boîtes noires potentiellement coûteuses en gardant une trace des valeurs comparées précédemment, évitant ainsi les comparaisons redondantes. Plusieurs mesures de similarité peuvent être combinées par une disjonction de règles de matching simples définies manuellement.

MOMA (mapping-based object matching) (Thor & Rahm, 2007) est un framework d'ER qui fournit une bibliothèque extensible de mesures de similarité. Le matching est basé à la fois sur les valeurs d'attributs et le contexte des entités. Le matching basé sur le contexte utilise les relations sémantiques entre différentes entités, par exemple deux auteurs sont susceptibles d'être le même individu s'ils ont des co-auteurs similaires. Différentes fonctions de combinaison (moyenne, min, max, pondérées) peuvent être utilisées pour combiner les valeurs des mesures de similarité individuelles.

DuDe (duplicate detection toolkit) (Draisbach & Naumann, 2010) est boîte à outils pour l'ER qui offre trois benchmarks (pour la déduplication) et plusieurs composants avec des interfaces pouvant être desservies avec un code individuel. L'outil propose deux méthodes d'indexation: Sorted-Neighborhood et une variante de cette méthode qui modifie de manière dynamique la taille de la fenêtre en fonction du fait que la dernière paire renvoyée a été interprétée comme match ou non. Plusieurs mesures de similarité peuvent être combinées par ce que l'on appelle des comparateurs multiples et qui permettent de calculer la valeur minimale ou maximale, la moyenne pondérée et la moyenne harmonique.

FRIL (Finegrained Record Integration and Linkage Tool) (Jurczyk, Lu, Xiong, Cragan, & Correa, 2008) est un outil développé en Java. Il fournit un ensemble de paramètres ajustables par l'utilisateur, complétés par des outils de visualisation graphiques, pour aider les utilisateurs à comprendre les effets des choix des paramètres. Sorted-Neighborhood est utilisé comme méthode d'indexation. Les mesures de similarité disponibles sont: Éditer Distance, Soundex, Q-grams et Egalité. Une somme pondérée normalisée est fournie pour combiner les valeurs des mesures de similarité en une valeur *v*. Deux seuils *t1* et *t2* (*t1<t2*) doivent être définis par l'utilisateur. Si *v>t2* alors les deux enregistrements sont considérés comme matches, si *v<t1* alors les deux enregistrements sont considérés comme non-matches et si *v* est comprise entre *t1* et *t2* alors les deux enregistrements sont considérés comme matches probables et doivent être vérifiés par un expert.

BN (Bayesian network) (Leitão, Calado & Weis, 2007) est un framework pour l'ER des entités représentées en XML sur la base d'un modèle de réseau bayésien. Les réseaux bayésiens fournissent un formalisme basé sur des graphes pour représenter explicitement les dépendances entre les entités d'un domaine. Ce modèle est dérivé de la structure des entités XML à matcher. L'approche combine numériquement la similarité des valeurs d'attributs des deux entités XML ainsi que celles des entités XML descendantes. L'utilisateur doit spécifier un seuil de probabilité de matching au-dessus duquel les entités sont considérées comme des matches.

Les approches à base de règles sont des approches manuelles qui nécessitent la spécification d'un modèle de classification (ensemble de règles ou fonction de combinaison numérique) par un expert du domaine. Par conséquent, ces systèmes manuels deviennent difficiles à utiliser pour tous les utilisateurs et dans tous les domaines.

3.3 Approches à base d'apprentissage supervisé

Les approches basées sur l'apprentissage supervisé apprennent de bonnes règles de matching à partir de données d'apprentissage (ensemble de vecteurs de comparaison préalablement étiquetés comme étant matches ou non-matches).

MARLIN (Multiply Adaptive Record Linkage with INduction) (Bilenko & Mooney, 2003a) est un framework basé sur l'apprentissage qui utilise SVM à deux niveaux. Au premier niveau, SVM est utilisé pour apprendre pour chaque attribut la mesure de similarité à utiliser. Au deuxième niveau, SVM est utilisé pour déterminer une combinaison de mesures de similarité. MARLIN utilise la méthode d'indexation Canopy Clustering utilisant la mesure de similarité Jaccard. Pour la sélection des données d'apprentissage, deux méthodes semi-automatiques sont utilisées: la *sélection statique active* et la *sélection négative faiblement étiquetée* (Bilenko & Mooney, 2003b). Dans la sélection statique active, les paires

d’enregistrements les plus similaires et les plus dissimilaires en utilisant la mesure de similarité TF-IDF sont sélectionnées et présentées à l’utilisateur pour être étiquetées manuellement et utilisées comme données d’apprentissage. La sélection négative faiblement étiquetée est une méthode non supervisée, elle est utilisée si les exemples positifs sont disponibles et l’objectif est de sélectionner les exemples négatifs. Dans ce cas, un ensemble de paires d’enregistrements parmi ceux qui ne partageant pas plus de 20% de tokens sont sélectionnées d’une façon aléatoire et considérées comme des exemples négatifs.

Le système de classificateur multiple (Zhao & Ram, 2005) utilise une variété de classificateur pour apprendre à combiner les mesures de similarité, notamment les arbres de décision, Naïve Bayes, la régression linéaire et logistique, les réseaux de neurones et les k-voisins les plus proches. En outre, les approches de méta-combinaison permettant de combiner plusieurs apprenants supervisés sont prises en charge, à savoir le bagging, le boosting et le stacking (voir annexe). Tandis que le bagging et le boosting combinent plusieurs classificateurs supervisés du même type, le stacking est utilisé pour combiner des classificateurs supervisés de types différents (par exemple, les arbres de décision et la régression logistique). Le support d’indexation n'est pas explicitement mentionné, mais l'évaluation suggère qu'une méthode d’indexation a été utilisée.

Chaudhuri, Chen, Ganti and Kaushik (2007) spécifient l’ER par une arborescence d'opérateurs qui correspond à l'union (disjonction) de *similarity joins*. Des échantillons d'apprentissage étiquetés manuellement sont utilisés pour construire les arbres d'opérateurs selon une stratégie récursive de division et de conquête. Le nombre maximal de *similarity joins* dans une arborescence d'opérateurs et le nombre maximal de fonctions de similarité par *similarity joins* peuvent être limités par l'utilisateur. L’indexation n'est pas explicitement prise en charge. Cependant, les auteurs indiquent que la méthode d’indexation Canopy Clustering utilisant la mesure de similarité Jaccard est utilisée dans l'évaluation.

Dans (Reyes-Galaviz, Pedrycz, He & Pizzi, 2017), les auteurs ont proposé un algorithme d'apprentissage supervisé basé sur les gradients issu d'une certaine topologie d'un réseau de neurones artificiels. Pour l'agrégation des scores des mesures de similarité, une somme pondérée normalisée est utilisée. L'algorithme proposé permet d'apprendre les poids des attributs et deux seuils. Les paires dont le score est au-dessus d'un seuil supérieur sont considérées comme des matches et les paires dont le score est au-dessous d'un seuil inférieur sont considérées comme des non-matches. Les paires avec un score compris entre les deux seuils sont considérées comme des matches potentiels et sont examinés par un expert. Pour l'indexation, le blocage standard (traditionnel) est utilisé.

Les approches basées sur l'apprentissage supervisé se traduisent généralement par une qualité de matching plus meilleure car elles sont adaptatives (Christen, 2012a), mais nécessitent un grand nombre d'exemples d'apprentissage (vecteurs de comparaisons étiquetés comme matches ou non-matches) qui n'est pas disponible dans de nombreuses situations réelles.

3.4 Approches à base d'apprentissage semi supervisé

Ces dernières années, des approches semi-supervisées ont été proposées pour alléger le problème des approches supervisées en utilisant un ensemble de données d'apprentissage réduit. Parmi les approches semi-supervisées, les approches basées sur l'apprentissage actif procèdent d'une façon itérative et commencent par un ensemble réduit d'exemples d'apprentissage. À chaque itération, les vecteurs de comparaison considérés comme représentatifs sont identifiés, étiquetés manuellement par l'utilisateur et ajoutés à l'ensemble des données d'apprentissage pour retrainer le classificateur.

Active Atlas (Tejada, Knoblock & Minton, 2002) est un système qui permet d'apprendre un ensemble de règles de matching en utilisant l'apprentissage actif. Plusieurs apprenants (arbre de décision) sont utilisés. Les vecteurs de comparaison

non étiquetés qui présentent plus de désagrément entre un comité de classificateurs sont considérés comme représentatifs et présentés à l'utilisateur pour étiquetage. Pour la comparaison des valeurs d'attributs, une variante de TF-IDF est utilisée. Pour l'indexation, une stratégie de blocage disjointe basée sur le hachage est prise en charge.

ALIAS (Sarawagi & Bhamidipaty, 2002) est un système similaire à Active Atlas qui prend en charge trois apprenants comme classificateur de base (arbre de décision, SVM et naive Bayes). Pour la comparaison des valeurs d'attributs, plusieurs mesures de similarité sont supportées comme 3-grams et Levenshtein. Pour l'indexation, Sorted Neighborhood et le blocage standard sont supportés.

T3S (Dal Bianco, Galante, Goncalves, Canuto & Heuser, 2015) est une approche qui vise à minimiser le nombre des vecteurs de comparaison à étiqueter par l'utilisateur. En premier lieu, les paires d'enregistrements candidates sont partitionnées en plusieurs groupes selon leurs scores de similarité. Ensuite, un ensemble de paires est sélectionné d'une façon aléatoire à partir de chaque groupe. Dans la deuxième étape, les paires représentatives sont sélectionnées d'une façon itérative en utilisant la méthode SSAR (Selective Sampling using Association Rules) (Silva, Gonçalves & Veloso, 2014). La mesure de similarité utilisée est Jaccard des n-grams. L'indexation est basée sur le filtrage PPJoin+ (Xiao, Wang, Lin, Yu & Wang, 2011) (avec la mesure de similarité jaccard) et le seuil est déterminé automatiquement.

Dans (Wang, Vatsalan & Christen, 2015), les auteurs ont présenté une approche qui permet de prendre en compte le nombre alloué des vecteurs de comparaison à étiqueter par l'utilisateur. Dans chaque itération, l'algorithme de clusturing farthest-first est utilisé pour sélectionner les vecteurs de comparaison les plus distants entre eux. Ces vecteurs sont étiquetés par l'utilisateur, enlevés de l'ensemble des vecteurs non étiquetés et ajoutés aux données d'apprentissage. Ensuite, les vecteurs non étiquetés sont fragmentés en deux groupes en utilisant le classificateur et sont traités dans la prochaine itération. L'algorithme se termine

lorsque le nombre alloué des vecteurs de comparaison à étiqueter par l'utilisateur est atteint ou tous les vecteurs non étiquetés sont traités.

SILK (Volz, Bizer, Gaedke & Kobilarov, 2009) est un framework hybride pour l'ER des entités représentées en RDF. Il fournit un langage déclaratif permettant à l'utilisateur de spécifier les mapping entre propriétés, les mesures de similarité (jaro, jaroWinkler, q-grams, egalité, etc.), la méthode d'agrégation des scores des similarités (moyenne pondérée, max, min, produit pondéré), les seuils et la méthode d'indexation à utiliser. Pour l'indexation, SILK permet la spécification (manuelle) de plusieurs clés de blocage, c'est-à-dire que seules les instances partageant une des clés de blocage doivent être comparées les unes aux autres. SILK prend en charge aussi l'apprentissage supervisé utilisant les algorithmes génétiques grâce à son algorithme GenLink (Isele & Bizer, 2012) et l'apprentissage actif utilisant les algorithmes génétiques grâce à son algorithme Active-GenLink (Isele & Bizer, 2013).

Une deuxième catégorie d'approches semi-supervisées utilisée dans (Kejriwal & Miranker, 2015) repose sur l'auto apprentissage. L'idée est de commencer avec un ensemble réduit de données d'apprentissage. Ensuite, à chaque itération, les vecteurs de comparaison pour lesquels le classificateur est le plus sûr sont ajoutés à l'ensemble de données d'apprentissage pour retrainer le classificateur. Ce processus itère jusqu'à ce que tous les vecteurs de comparaison non étiquetés aient été classés par le classificateur final. 28 mesures de similarité sont utilisées : 2 numériques, 8 basées sur les caractères et les tokens, et 18 phonétiques. Pour l'indexation, *Attribute Clustering* (Papadakis et al., 2013) est utilisé. Pour la classification, les auteurs utilisent l'apprentissage d'ensemble (Dietterich, 2000) et en particulier le boosting (voir annexe) qui consiste à trainer plusieurs classificateurs de base afin de construire un classificateur général. Le classificateur d'ensemble utilise le vote à la majorité pondérée pour prendre une décision.

Les approches semi supervisées nécessitent un ensemble réduit de données d'apprentissage, elles sont itératives et peuvent aussi nécessiter l'intervention continue d'un humain pour l'étiquetage.

3.5 Approches non supervisées

Les approches semi-supervisé nécessite l'existence d'un ensemble réduit de données d'apprentissage et / ou l'intervention d'un expert pour l'étiquetage. Pour faire face à ce problème, plusieurs approches non supervisées (automatiques) ont été proposées.

TAILOR (Elfeky et al., 2002) est un framework hybride qui prend en charge des approches de combinaison numériques et des approches basées sur l'apprentissage utilisant les arbres de décision. Cinq mesures de similarité peuvent être utilisées (distance de Hamming, Levenshtein, Jaro, q-grams et soundex). TAILOR supporte le blocage standard et Sorted Neighborhood comme méthodes d'indexation. Pour l'automatisation, deux approches sont proposées. La première approche non supervisée utilise l'algorithme de clusturing k-means. La deuxième approche est hybride, elle utilise l'algorithme k-means pour classer un sous ensemble de vecteurs de comparaison en matches et non matches. Les vecteurs classés par k-means sont ensuite utilisés comme données d'apprentissage par les arbres de décision. Les auteurs ont constaté dans l'expérimentation que l'approche hybride est plus performante que l'approche non supervisée utilisant k-means.

FEBRL (Freely Extensible Biomedical Record Linkage) (Christen, 2008) est un framework hybride qui prend en charge des approches basées sur l'apprentissage utilisant SVM ainsi que des approches basées sur les règles (fonction de combinaison numérique). 26 mesures de similarité différentes sont disponibles pour la comparaison des valeurs d'attributs et 6 méthodes d'indexation sont supportées. Pour l'automatisation, l'auteur propose deux méthodes pour la génération automatique des données d'apprentissage (Christen, 2007) : *threshold based* et *nearest based.* Dans la méthode threshold based, les vecteurs de comparions qui ont

dans tous leurs éléments une valeur inferieure (supérieure) à un seuil *s1* (*s2*) sont considérés comme non-matches (matches). Dans la méthode nearest based, les vecteurs de comparaisons les plus proches du vecteur 1(1,1,…,1) (du vecteur 0(0,0,…,0)) en utilisant par exemple la distance euclidienne sont considérés comme matches (non- matches). Pour cette approche, la sélection des vecteurs de comparaison est basée sur le score d'une seule mesure de similarité (Jaro-Winkler), qui est une mesure de similarité basée sur les caractères et qui ne prend pas en charge les erreurs de réarrangement de mots, alors que notre générateur de données d'apprentissage utilise plusieurs mesures de similarité pour chaque attribut pour prendre en charge plusieurs types d'erreurs (typographiques et réarrangements de tokens). Le travail de (Christen, 2007) sera comparé avec notre générateur de données d'apprentissage dans l'expérimentation.

Une deuxième catégorie d'approches non supervisées repose sur l'optimisation de la valeur d'une fonction objective appelée pseudo F-mesure (PFM) sur des données non étiquetées. La PFM est une heuristique visant à donner une approximation de la F-mesure réelle, elle est formulée en supposant que deux enregistrements de sources de données différentes représentent la même entité, tandis que deux enregistrements de la même source de données représentent des entités distinctes. L'objectif de ces approches est de trouver des règles de matching qui maximisent la valeur de la PFM.

KnoFuss (Nikolov, Uren & Motta, 2007) est un framework hybride pour l'ER des entités représentées en RDF qui prend en charge la spécification manuelle des règles de matching (combinaison numérique) ainsi qu'un algorithme non déterministe (KnoFuss+GA) qui utilise la PFM comme fonction de fitness dans les algorithmes génétiques (Nikolov, d'Aquin & Motta, 2012).

LIMES (Ngomo & Auer, 2011) est un framework pour le matching des entités représentées en RDF qui est similaire à SILK. Il fournit un langage déclaratif permettant à l'utilisateur de spécifier les mapping entre propriétés, les mesures de similarité, la méthode d'agrégation des scores des similarités et les seuils. LIMES

applique un filtrage en exploitant l'inégalité triangulaire pour optimiser l'exécution. LIMES prend en charge aussi l'apprentissage supervisé, l'apprentissage actif utilisant les algorithmes génétiques grâce à son algorithme EAGLE (Ngomo & Lyko, 2012), l'apprentissage actif utilisant son algorithme déterministe RAVEN (Ngomo, Lehmann, Auer & Höffner, 2011) qui effectue une recherche hiérarchique pour trouver un classificateur linéaire et un classificateur booléen, et enfin l'apprentissage non supervisé en utilisant un algorithme déterministe (EUCLID) et un autre non déterministe (EAGLE-UN) (Ngomo & Lyko, 2013).

EUCLID (Ngomo & Lyko, 2013) est un algorithme déterministe qui effectue une recherche hiérarchique pour trouver un classificateur linéaire et un classificateur booléen en optimisant la PFM. Les auteurs ont présenté aussi un algorithme non déterministe qui utilise la PFM comme fonction de fitness dans les algorithmes génétique. Dans une étude récente, les auteurs dans (Ngomo & Lyko, 2013) ont constaté que, sur des données réelles, la PFM n'est pas bien corrélée avec la vraie F-mesure. Avec l'approche génétique, les sources de données doivent être analysées en plusieurs itérations et les résultats ne sont pas déterministes. Contrairement aux approches non supervisées basées sur la PFM, l'approche proposée dans cette thèse évite la supervision en générant automatiquement ses propres données d'apprentissage. EUCLID (Ngomo & Lyko, 2013) sera comparé avec notre approche dans l'expérimentation.

3.6 Approches pour langue arabe

La plupart des frameworks ont été développés pour effectuer la résolution d'entité dans les sources de données en latin et en particulier l'anglais comme Febrl (Christen, 2008) et TAILOR (Elfeky et al., 2002). Ces frameworks ne reconnaissent même pas les caractères non latins et en particulier l'arabe parce qu'ils n'utilisent pas le système Unicode (Higazy et al., 2013). Pour cette raison, plusieurs travaux ont été développés pour prendre en charge la langue arabe.

Gueddah et al. (2012) ont proposé une variante de la distance de Levenshtein. Pour calculer la similarité, le cout des opérations d'édition dépend de trois matrices de fréquences (M_insertion, M_suppression, M_substitution) qui ont été calculées à partir d'un test effectué par un ensemble d'opérateurs professionnels. Cette approche nécessite l'existence d'un corpus afin d'évaluer et d'estimer les matrices de fréquence des erreurs d'édition.

Une autre approche pour améliorer l'algorithme de Levenshtein est présentée dans (Ghafour et al., 2011). Les auteurs utilisent trois matrices de similarité pour calculer le cout d'opérations d'éditions (ajout, suppression et modification) : la matrice de similarité de forme de caractère, la matrice de similarité phonétique et la matrice de proximité des lettres dans le clavier (en fonction de la distance euclidienne).

El-Shishtawy (2013) a proposé une extension de la distance de Levenshtein pour les noms arabes composés de plusieurs tokens. L'idée est de traiter chaque token comme un caractère. Le cout des opérations d'édition (insertion, suppression et modification) est calculé par une fonction qui calcule la position du token dans le nom et sa fréquence dans un corpus de noms.

D'autres travaux se sont concentrés sur les mesures de similarités phonétiques pour la langue arabe. Aqeel, Beitzel, Jensen, Grossman and Frieder (2006) ont proposé la version arabe de l'algorithme Soundex. Yousef (2014) a proposé une mesure phonétique pour les noms arabes composés de plusieurs tokens. L'idée est de calculer le code soundex de chaque token et ensuite concaténer les codes résultants pour avoir le code soundex du nom composé.

Les approches citées précédemment se sont concentrées sur la résolution d'entité dans les sources de données composées d'un seul attribut (en général les noms en arabe). En plus, ces approches nécessitent des données d'apprentissage élaborées manuellement par un expert et/ou la définition d'un seuil pour décider si deux enregistrements (noms) matchent ou non.

Higazy et al. (2013) ont proposé un framework d'ER qui prend en charge les sources de données en anglais ainsi que celles en arabe. Le taux du rappel été amélioré lorsqu'une extension pour la langue arabe a été ajoutée à l'étape de prétraitement. Le framework présenté dans (Higazy et al., 2013) nécessite l'intervention d'un expert pour l'élaboration des règles de matching ou l'existence des données d'apprentissage.

3.7 Etude comparative des approches

Dans cette section, nous présentons une comparaison entre les différentes approches de résolution d'entité selon les critères suivants : (1) méthode d'indexation utilisée, (2) mesures de similarité. Pour cette colonne deux cas se présentent : la comparaison se fait en utilisant les valeurs d'attributs ou selon le contexte en utilisant les relations sémantiques entre les différentes entités, par exemple deux auteurs sont susceptibles d'être le même individu s'ils ont des co-auteurs similaires, (3) modèle de classification : ensemble de règles ou fonction numérique dans le cas des approches manuelles, le classificateur utilisé dans le cas des approches à base d'apprentissage et (4) L'automatisation indique si l'indexation ou la classification ou les deux sont automatisées ou non.

Concernant le format de données accepté par les approches traitées dans ce mémoire, BN (Leitão et al., 2007) se focalise sur les données XML. SERF (Benjelloun et al., 2009), MOMA (Thor & Rahm, 2007), DuDe (Draisbach & Naumann, 2010), FRIL (Jurczyk et al., 2008), MARLIN (Bilenko & Mooney, 2003a), classificateur multiple (Zhao & Ram, 2005), (Chaudhuri et al., 2007), (Reyes-Galaviz et al., 2017), Active Atlas (Tejada et al., 2002), ALIAS (Sarawagi & Bhamidipaty, 2002), (Dal Bianco et al., 2015), (Wang et al., 2015), TAILOR (Elfeky et al., 2002) et FEBRL (Christen, 2008) se focalisent sur les données relationnelles. SILK (Volz et al., 2009), GenLink (Isele & Bizer, 2012), Active-GenLink (Isele & Bizer, 2013), (Kejriwal & Miranker, 2015), KnoFuss (Nikolov et

al., 2007), KnoFuss+GA (Nikolov et al., 2012), LIMES (Ngomo & Auer, 2011), EAGLE (Ngomo & Lyko, 2012), RAVEN (Ngomo et al., 2011), EAGLE-UN (Ngomo & Lyko, 2013) et EUCLID (Ngomo & Lyko, 2013) se focalisent sur les données RDF.

Le tableau 3.1 compare les approches basées sur les règles (qui n'utilisent pas des données d'apprentissage), alors que le tableau 3.2 compare les approches basées sur les données d'apprentissage.

Tableau 3.1 : *Systèmes basés sur les règles (- : non supporté)*

Système	**Indexation**	**Mesures de similarité**	**Modèle de classification**	**Automatisation** ***Indexation*** ***Classification***
SERF (Benjelloun et al., 2009)	-	Valeur d'attribut	Règles	- Manuelle
MOMA (Thor & Rahm, 2007)	-	Valeur d'attribut, contexte	Fonction numérique	- Manuelle
DuDe (Draisbach & Naumann, 2010)	Sorted-Neighborhood	Valeur d'attribut	Fonction numérique	manuelle manuelle
FRIL (Jurczyk et al., 2008)	Sorted-Neighborhood	Valeur d'attribut	Fonction numérique	manuelle manuelle
BN (Leitão et al., 2007)	-	Valeur d'attribut, contexte	Fonction numérique	- Manuelle
SILK (Volz et al., 2009)	standard (multi dimensionnel)	Valeur d'attribut	Fonction numérique	manuelle manuelle
KnoFuss	-	Valeur	Fonction	-

(Nikolov et al., 2007)		d'attribut	numérique	Manuelle
LIMES (Ngomo & Auer, 2011)	-	Valeur d'attribut	Règles	- Manuelle
(Gueddah et al., 2012)	-	Valeur d'attribut	Règles	- Manuelle
(Ghafour et al., 2011)	-	Valeur d'attribut	Règles	- Manuelle
(El-Shishtawy, 2013)	-	Valeur d'attribut	Règles	- Manuelle
(Aqeel et al., 2006)	-	Valeur d'attribut	Règles	- Manuelle
(Yousef, 2014)	-	Valeur d'attribut	Règles	- Manuelle
(Higazy et al., 2013)	Imbriquée (standard + Sorted-Neighborhood)	Valeur d'attribut	Règles	manuelle manuelle

Indexation

La plupart des systèmes d'ER basés sur l'apprentissage fournissent un support explicite pour l'indexation ou un filtrage pour optimiser l'exécution, alors que les approches basées sur les règles n'utilisent pas l'indexation à l'exception de DuDe (Draisbach & Naumann, 2010), FRIL (Jurczyk et al., 2008), SILK (Volz et al., 2009) et (Higazy et al., 2013). Cela semble être influencé par le fait que les classificateurs comme SVM ou les arbres de décision, sont couteux en termes de calcul, de sorte que l'indexation est obligatoire pour réduire l'espace de recherche. Les méthodes d'indexation les plus utilisées sont Sorted Neighborhood et le blocage standard. A l'exception des deux systèmes présentés dans (Dal Bianco et

al., 2015) et (Kejriwal & Miranker, 2015), tous les systèmes nécessitent la spécification manuelle de la clé de blocage par un expert. Ceci constitue une limitation sérieuse concernant l'automatisation de l'indexation.

Comparaison

Pour l'étape de comparaison, tous les systèmes utilisent les valeurs d'attributs pour comparer les enregistrements. Seulement deux systèmes MOMA (Thor & Rahm, 2007) et BN (Leitão et al., 2007) utilisent les informations du contexte (relations sémantiques entre les entités). Cela est dû au fait que ces relations ne sont pas toujours disponibles et sont en général présentes dans des domaines spécifiques comme le domaine bibliographique.

Classification

Pour la classification, les approches basées sur les règles utilisent un ensemble de règles de matching ou une fonction de combinaison numérique (comme la moyenne pondérée) développées manuellement par un expert. L'inconvénient de ces systèmes manuels est qu'ils ne peuvent pas être utilisés par tous les utilisateurs et dans tous les domaines et cela constitue une limitation sérieuse concernant l'automatisation.

Les systèmes basés sur l'apprentissage supervisé: MARLIN (Bilenko & Mooney, 2003a), classificateur multiple (Zhao & Ram, 2005), (Chaudhuri et al., 2007), (Reyes-Galaviz et al., 2017) et GenLink (Isele & Bizer, 2012) sont adaptatifs du fait qu'ils apprennent un modèle de classification (ensemble de règles de matching ou une fonction de combinaison numérique) à partir de données d'apprentissage. Les classificateurs les plus utilisés sont SVM pour apprendre une fonction de combinaison numérique et les arbres de décision pour apprendre un ensemble de règles de matching. L'inconvénient de ces approches est qu'ils nécessitent l'existence d'un nombre important d'exemples d'apprentissage qui ne sont pas disponible dans des situations réelles.

Les systèmes basés sur l'apprentissage semi supervisé: Active Atlas (Tejada et al., 2002), ALIAS (Sarawagi & Bhamidipaty, 2002), T3S (Dal Bianco et al., 2015), (Wang et al., 2015), Active-GenLink (Isele & Bizer, 2013), EAGLE (Ngomo & Lyko, 2012), RAVEN (Ngomo et al., 2011) et (Kejriwal & Miranker, 2015) utilisent un ensemble réduit d'exemples d'apprentissage et procèdent d'une façon itérative. Dans chaque itération, l'objectif de ces systèmes est d'identifier les vecteurs de comparaison représentatifs qui seront présentés à l'utilisateur pour étiquetage et ensuite ajoutés aux données d'apprentissage pour retraiter le classificateur (dans le cas de l'auto apprentissage (Kejriwal & Miranker, 2015), les vecteurs sont étiquetés par le système). Les vecteurs de comparaison représentatifs sont sélectionnés en se basant sur le désagrément entre un comité de classificateurs (Active Atlas (Tejada et al., 2002), ALIAS (Sarawagi & Bhamidipaty, 2002), Active-GenLink (Isele & Bizer, 2013) et (Ngomo & Lyko, 2012)), l'agrément entre un comité de classificateurs (Kejriwal & Miranker, 2015), les règles d'association T3S (Dal Bianco et al., 2015), les plus loin entre eux en utilisant l'algorithme de clusturing farthest-first (Wang et al., 2015) et les plus proches de la frontière de séparation RAVEN (Ngomo et al., 2011). Les approches semi supervisées réduisent considérablement le nombre d'exemples d'apprentissage mais nécessitent l'intervention continue d'un expert pour l'étiquetage. Un autre inconvénient de ces approches est qu'elles sont itératives parce qu'elles trainent un classificateur (ou un ensemble de classificateurs) plusieurs fois pour aboutir un modèle de classification optimal.

Pour surmonter le problème de supervision, les approches non supervisées: TAILOR (Elfeky et al., 2002), FEBRL (Christen, 2008), KnoFuss+GA (Nikolov et al., 2012), EUCLID (Ngomo & Lyko, 2013) et EAGLE-UN (Ngomo & Lyko, 2013) visent à effectuer la résolution d'entité d'une façon automatique (non supervisée). EAGLE-UN (Ngomo & Lyko, 2013) et KnoFuss+GA (Nikolov et al., 2012) sont des algorithmes non déterministes qui utilisent la pseudo F-mesure (PFM) comme fonction de fitness dans les algorithmes génétique alors qu'EUCLID (Ngomo &

Lyko, 2013) est un algorithme déterministe qui effectue une recherche hiérarchique pour trouver un classificateur linéaire et un classificateur booléen en optimisant la PFM. L'inconvénient des approches génétiques (Ngomo & Lyko, 2013; Nikolov et al., 2012) est la nécessité d'analyser les sources de données en plusieurs itérations. Un deuxième inconvénient des approches génétiques est qu'elles sont non déterministes et peuvent ne pas converger. Un autre inconvénient qui concerne l'approche déterministe et non déterministe (Ngomo & Lyko, 2013) est que la PFM n'est pas bien corrélée avec la vraie F-mesure. Contrairement aux approches non supervisées basées sur la PFM, le système proposé dans cette thèse évite la supervision en générant automatiquement ses propres données d'apprentissage. EUCLID (Ngomo & Lyko, 2013) sera utilisé comme ligne de base dans l'expérimentation.

Les deux autres approches pour automatiser l'ER sont TAILOR (Elfeky et al., 2002) et FEBRL (Christen, 2008). La première approche (Elfeky et al., 2002) consiste à utiliser l'algorithme de clusturing K-means pour classer les vecteurs de comparaison alors que la deuxième approche (Christen, 2007) consiste à générer les données d'apprentissage d'une façon automatique en utilisant une heuristique. Les auteurs dans (Elfeky et al., 2002) et (Christen, 2007) ont constaté dans l'expérimentation que l'approche basée sur la génération automatique des données d'apprentissage est plus performante que l'approche non supervisée utilisant k-means. Dans le travail de (Christen, 2007), la sélection des vecteurs de comparaison est basée sur le score d'une seule mesure de similarité (Jaro-Winkler), alors que notre générateur de données d'apprentissage utilise plusieurs mesures de similarité pour chaque attribut. Le travail de (Christen, 2007) sera utilisé comme ligne de base dans l'expérimentation.

Tableau 3.2 : *Systèmes basés sur l'apprentissage (- : non supporté, ?: supporté mais non spécifié)*

Système	**Indexation**	**Mesures de similarité**	**Modèle de classification**	**Automatisation** ***Indexation Classification***
MARLIN (Bilenko & Mooney, 2003a)	Canopy Clustering	Valeur d'attribut	SVM	manuelle semi supervisée
classificateur multiple (Zhao & Ram, 2005)	-	Valeur d'attribut	7 (SVM, arbres de décision, etc.)	- Supervisée
(Chaudhuri et al., 2007)	Canopy Clustering	Valeur d'attribut	Algorithme arborescence d'opérateurs, SVM	manuelle supervisée
(Reyes-Galaviz et al., 2017)	blocage standard	Valeur d'attribut	Algorithme basé sur les gradients	manuelle supervisée
GenLink (Isele & Bizer, 2012)	standard (multi dimensionnel)	Valeur d'attribut	Algorithmes génétique	manuelle supervisée
Active Atlas (Tejada et al., 2002)	Hachage	Valeur d'attribut	Comité d'arbres de décision	manuelle semi supervisée
ALIAS (Sarawagi & Bhamidipaty, 2002)	Sorted Neighborhood, blocage standard	Valeur d'attribut	Comité (arbre de décision, SVM et naive Bayes)	manuelle semi supervisée
T3S (Dal Bianco et al., 2015)	PPJoin+ (Xiao et al., 2011) (jaccard)	Valeur d'attribut	SVM	automatique semi supervisée

(Wang et al., 2015)	?	Valeur d'attribut	arbre de décision	? semi supervisée
Active-GenLink (Isele & Bizer, 2013)	standard (multi dimensionnel)	Valeur d'attribut	Algorithmes génétique	manuelle semi supervisée
EAGLE (Ngomo & Lyko, 2012)	-	Valeur d'attribut	Algorithmes génétique	- semi supervisée
RAVEN (Ngomo et al., 2011)	-	Valeur d'attribut	Algorithme recherche hiérarchique	- semi supervisée
(Kejriwal & Miranker, 2015)	Attribute Clustering (Papadakis et al., 2013)	Valeur d'attribut	Boosting (Random forests, multilayer perceptrons)	automatique semi supervisée
TAILOR (Elfeky et al., 2002)	blocage standard, Sorted Neighborhood	Valeur d'attribut	arbres de décision, k-means	Manuelle Automatique
FEBRL (Christen, 2008)	6 (blocage standard, Sorted Neighborhood, etc.)	Valeur d'attribut	SVM, k-means, FarthestFirst	Manuelle Automatique
KnoFuss+GA (Nikolov et al., 2012)	-	Valeur d'attribut	Algorithmes génétique	- Automatique
EUCLID (Ngomo & Lyko, 2013)	-	Valeur d'attribut	Algorithme recherche hiérarchique	- Automatique
EAGLE-UN (Ngomo & Lyko, 2013)	-	Valeur d'attribut	Algorithmes génétique	- Automatique

Automatisation

Pour l'automatisation, deux étapes de la procédure d'ER sont à automatiser. Pour qu'un système d'ER soit complètement automatisé, l'étape d'indexation et l'étape de classification doivent être automatisées. La colonne Automatisation du tableau 3.1 et du tableau 3.2 montre le degré d'automatisation pour les différents systèmes d'ER.

Pour les approches basées sur les règles, la classification n'est pas automatisée puisque les règles doivent être spécifiées par un expert du domaine. Pour l'indexation, ces approches ne l'utilisent pas à l'exception des systèmes: DuDe (Draisbach & Naumann, 2010), FRIL (Jurczyk et al., 2008), SILK (Volz et al., 2009) et (Higazy et al., 2013) qui utilisent une indexation manuelle (standard multi dimensionnel et Sorted-Neighborhood) nécessitant la spécification de clé de blocage.

Les systèmes basés sur l'apprentissage supervisé: MARLIN (Bilenko & Mooney, 2003a), classificateur multiple (Zhao & Ram, 2005), (Chaudhuri et al., 2007), (Reyes-Galaviz et al., 2017) et GenLink (Isele & Bizer, 2012) dérivent un modèle de classification à partir des données d'apprentissage. Ils sont adaptatifs mais la classification n'est pas automatique du fait que les données d'apprentissage doivent être élaborées par un expert. Une exception à ces systèmes est MARLIN (Bilenko & Mooney, 2003a) dont la classification est semi automatique. Tous les systèmes supervisés utilisent une indexation manuelle.

Pour tous les systèmes basés sur l'apprentissage semi supervisé: Active Atlas (Tejada et al., 2002), ALIAS (Sarawagi & Bhamidipaty, 2002), T3S (Dal Bianco et al., 2015), (Wang et al., 2015), Active-GenLink (Isele & Bizer, 2013), EAGLE (Ngomo & Lyko, 2012), RAVEN (Ngomo et al., 2011) et (Kejriwal & Miranker, 2015), la classification est semi automatique alors que l'indexation est manuelle à l'exception du système présenté dans (Kejriwal & Miranker, 2015) qui utilise Attribute Clustering (Papadakis et al., 2013) qui est une indexation automatique ne nécessitant pas la définition de clé de blocage et des deux approches EAGLE

(Ngomo & Lyko, 2012) et RAVEN (Ngomo et al., 2011) qui n'utilisent pas explicitement l'indexation.

Les systèmes automatiques: TAILOR (Elfeky et al., 2002), FEBRL (Christen, 2008), KnoFuss+GA (Nikolov et al.,2012), EUCLID (Ngomo & Lyko, 2013) et EAGLE-UN (Ngomo & Lyko, 2013) ont tendance à automatiser l'étape de classification. TAILOR (Elfeky et al., 2002) et FEBRL (Christen, 2008) utilisent une indexation manuelle alors que KnoFuss+GA (Nikolov et al.,2012), EUCLID (Ngomo & Lyko, 2013) et EAGLE-UN (Ngomo & Lyko, 2013) ne l'utilisent pas. En comparaison avec ces systèmes automatiques, nous proposons un système d'ER entièrement automatique (non supervisé) qui automatise l'étape de classification et l'étape d'indexation.

3.8 Conclusion

Les systèmes de résolution d'entité permettent l'utilisation combinée d'algorithmes d'indexation et de multiples algorithmes de classification pour résoudre efficacement diverses tâches de matching. Ces systèmes doivent répondre à plusieurs exigences difficiles et en partie contradictoires. Ces exigences concernent l'efficacité élevée, la mise en échelle, la généricité et l'automatisation. Pour évaluer l'état de l'art actuel des systèmes d'ER, nous avons comparé quatorze systèmes basés sur les règles et dix huit systèmes basés sur l'apprentissage. La comparaison a été basée sur un ensemble de critères qui incluent : les méthodes d'indexation supportées, les méthodes de comparaison utilisées, le modèle de classification ainsi que l'automatisation (de l'indexation et de la classification).

Notre étude indique que les recherches dans ce domaine ont tendance à utiliser des approches basées sur l'apprentissage pour semi automatiser ou automatiser les étapes d'ER et en particulier l'étape de classification. Cela est dû au fait que les systèmes basés sur les règles sont difficiles à utiliser dans tous les domaines ou dans les domaines inexpérimentés.

Les systèmes basés sur l'apprentissage supervisé présentent l'inconvénient de nécessiter un nombre important d'exemples d'apprentissage qui ne sont pas disponibles dans des situations réelles. Une autre tendance pour alléger le problème d'inexistence des données d'apprentissage est d'utiliser l'apprentissage semi supervisé. Bien que ces dernières approches commencent avec un nombre réduit d'exemples d'apprentissage, elles sont itératives et nécessitent l'intervention continue d'un expert pour l'étiquetage.

Pour matcher des sources de données dépourvues de données d'apprentissage ou appartenant à un domaine inexpérimenté, il est difficile de définir avec précision un modèle de classification efficace. L'approche automatique (non supervisée) est donc très importante à étudier.

Les approches automatiques basées sur l'optimisation de la PFM: KnoFuss+GA (Nikolov et al., 2012), EUCLID et EAGLE-UN (Ngomo & Lyko, 2013) nécessitent plusieurs itérations pour converger vers un modèle de classification efficace. En plus, les approches génétiques KnoFuss+GA et EAGLE-UN sont non déterministes et nécessitent la spécification de plusieurs paramètres. Contrairement aux approches non supervisées basées sur la PFM, notre système est non itératif. Il évite la supervision en générant automatiquement ses propres données d'apprentissage en utilisant un algorithme non supervisé appelé générateur de données d'apprentissage (TSG).

La deuxième catégorie d'approches non supervisées est basée sue la génération de données d'apprentissage: TAILOR (Elfeky et al., 2002) et FEBRL (Christen, 2008). TAILOR (Elfeky et al., 2002) utilise K-means alors que FEBRL (Christen, 2008) utilise une heuristique basée sur une seule mesure de similarité qui ne peut pas prendre en charge plusieurs types d'erreurs. En comparaison avec ces deux dernières approches, notre générateur de données d'apprentissage utilise plusieurs mesures de similarité pour chaque attribut ce qui le rend plus performant.

Une autre limitation de la plupart des systèmes d'ER comme TAILOR (Elfeky et al., 2002) et FEBRL (Christen, 2008) est qu'ils ont été développés pour les

sources de données en latin et en particulier la langue anglaise et ne prennent pas en charge les caractères non latins et en particulier la langue arabe parce qu'ils n'utilisent pas le système Unicode (Higazy et al., 2013). D'une autre part, les systèmes d'ER développées pour prendre en charge la langue arabe sont manuels et se sont concentrés sur le matching des noms en arabe, ce qui signifie qu'ils sont limités à un seul domaine et ne peuvent pas être utilisées dans plusieurs domaines de la langue arabe.

Sur la base de ces limitations, nous proposons dans cette thèse un framework d'ER entièrement automatique et non itératif qui peut effectuer l'ER dans n'importe quel domaine pour les sources de données en Anglais ainsi que celles en Arabe.

Chapitre 4

Un système automatique de résolution d'entité

4.1 Introduction

Comme mentionné dans le chapitre 3, la plupart des systèmes de résolution d'entité présentent deux limitations. La première concerne la non prise en charge des caractères non latin parce qu'ils n'utilisent pas le système Unicode. Cela signifie que la plupart de ces systèmes ne prennent pas en charge la langue arabe. La seconde concerne l'automatisation. Pour qu'un système d'ER soit utilisable dans n'importe quel domaine, il doit présenter un degré élevé d'automatisation.

Dans ce chapitre, nous proposons un système de résolution d'entité entièrement automatique (OUHAB, MALKI, BERRABAH & BOUFARES, 2017). Le système proposé prend en charge des sources de données en anglais ainsi que des sources de données en arabe et il est indépendant du domaine dans lequel s'effectue la résolution d'entité.

4.2 Présentation de l'approche

La figure 4.1 illustre un schéma de haut niveau du système proposé. Le système prend comme entrées deux sources de données et retourne les deux ensembles matches et non-matches. L'ensemble matches contient les paires d'enregistrements représentant les mêmes entités alors que l'ensemble non-marches contient les paires

d'enregistrements représentant des entités différentes. Par exemple, si les deux sources de données de la figure 4.2 sont à matcher par le système proposé, les résultats attendus d'une exécution idéale sont les deux ensembles *matches* = {(r_1, s_1), (r_2, s_2)} et *non-matches* = {(r_1, s_2), (r_2, s_1)}

Comme le montre la figure 4.1, le système proposé comprend plusieurs modules: prétraitement, indexation, génération des données d'apprentissage, génération du modèle de classification et classification. L'étape de prétraitement permet de prendre en charge les sources de données en anglais ainsi que celles en arabe. La deuxième étape, l'indexation, vise à réduire le nombre de comparaisons nécessaires pour effectuer l'ER en regroupant les enregistrements en blocs (groupes). Seuls les enregistrements qui partagent un bloc sont considérés comme candidats et sont traités dans l'étape suivante.

Comme contribution principale dans le cadre de ce framework, nous proposons un algorithme non supervisé (TSG : Training Set Generator) à base d'heuristique qui génère un ensemble d'exemples d'apprentissage et permettant d'automatiser la procédure de résolution d'entité. Comme indiqué au chapitre 3, répondre aux besoins en automatisation constitue un goulot d'étranglement difficile pour les systèmes d'ER. En particulier, les exemples d'apprentissage sont difficiles à localiser en raison de la faible densité des données. Une fois générés, les exemples d'apprentissage sont utilisés pour apprendre un modèle de classification. Enfin, le modèle de classification appris est utilisé pour classer les vecteurs de comparaison non étiquetés (n'appartenant pas aux données d'apprentissage) en matches et non-matches. Dans ce qui suit, nous présentons le fonctionnement de chaque module du système proposé.

4.3 Prétraitement

L'étape de prétraitement permet au système proposé de prendre en charge les caractères non latin et en particulier les caractères de la langue arabe. Dans cette

étape, les sources de données sont converties au système Unicode pour prendre en charge la langue arabe. Plusieurs travaux (Yousef, 2015; Higazy et al., 2013; El-Shishtawy, 2013; Yousef, 2014) ont utilisé un ensemble de règles de normalisation pour les sources de données arabes. Ces règles consistent à remplacer un ensemble de caractères par leur caractère équivalent. Par exemple, remplacez les caractères إ ، أ ، آ par le caractère ا, le caractère ي par le caractère ى, le caractère ة par le caractère ه et le caractère ؤ par le caractère و. L'utilisation de ces règles peut être très coûteux mais utile si les sources de données à matcher contiennent de telles erreurs typographiques. Dans notre système, l'utilisateur a la possibilité d'utiliser ou non ces règles. Par exemple, pour les quatre enregistrements de la figure 4.2, le mot « حي » sera remplacé par le mot « حى ».

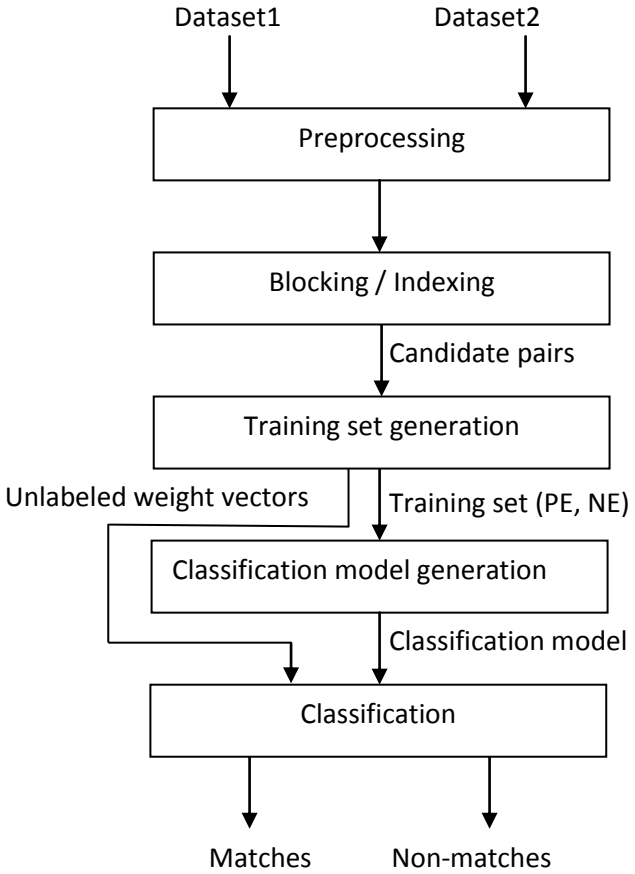

Figure 4.1: *Schéma du système automatique de résolution d'entité présenté dans cette thèse.*

4.4 Indexation

Dans cette étape, les sources de données sont indexées pour réduire l'espace de recherche. Le résultat de cette étape est un ensemble de paires d'enregistrements candidates. La résolution d'entité est alors limitée à ces paires d'enregistrements candidates. Plusieurs méthodes d'indexation ont été développées et le choix de la méthode d'indexation à utiliser dépend des caractéristiques des données et les types d'erreurs qu'elles contiennent (Christen, 2012b). La plupart des approches d'indexation nécessitent la spécification d'un schéma de blocage (clé de blocage) ou la fourniture d'un ensemble de données d'apprentissage pour apprendre un schéma de blocage.

Kejriwal and Miranker (2013) ont récemment proposé un système non supervisé pour apprendre la clé de blocage. Premièrement, un ensemble de données d'apprentissage est généré automatiquement. Dans la deuxième étape, l'apprentissage d'un schéma de blocage est considéré comme un problème de sélection des caractéristiques de Fisher (Gu et al., 2012). En tant qu'alternative à l'approche mentionnée précédemment, *Token blocking* (Papadakis et al., 2013) est une autre approche d'indexation non supervisée qui ne nécessite pas de données d'apprentissage. Dans Token blocking, chaque token distinct *t* crée un bloc contenant tous les enregistrements contenant *t*, quels que soient les noms d'attributs associés. Pour des raisons d'automatisation, notre système utilise Token blocking comme méthode d'indexation.

Un problème connu de Token blocking est qu'il' est très sensible à la fréquence des tokens utilisés pour créer les blocks. Un token très fréquent (contenu dans plusieurs enregistrements) et non discriminatif peut générer un block qui contient un nombre important de comparaisons. La notion de *Block Purging* a été introduite dans (Papadakis et al., 2013) pour éliminer les comparaisons qui correspondent à des non-matches en supprimant les blocs surdimensionnés. Ce sont des blocs qui contiennent un nombre excessivement élevé de comparaisons, même s'il est très

improbable qu'ils contiennent des matches ne possédant pas d'autres blocs plus petits en commun. Cela consiste à ne retenir que les blocks dont le nombre de comparaisons est inferieur ou égal à un seuil nommé *seuil de purging*. Pour le cas de déduplication (résolution d'entité dans une seule source de données), un algorithme a été proposé pour calculer automatiquement le *seuil de purging*, mais nécessite d'effectuer deux passes à travers les blocs générés (Papadakis et al., 2013).

Notre système utilise Token blocking suivi de Block Purging comme méthode d'indexation. Pour déterminer le seuil de purging, nous utilisons une heuristique simple et efficace qui consiste à fixer ce seuil à la taille de la plus petite source à matcher (*seuil de purging* = *min* (| *S* |, | *R* |)) tel que *S* et *R* sont les deux sources à matcher.

r_id	Last name	First name	Address
r1	محاجبية	زهور فتيحة	حي 140 مسكن بلوك 5 رقم141 وهران
r2	دليل	مختار قادة	حي 114 مسكن عمارة ب 9 رقم 85 مستغانم

s_id	Last name	First name	Address
s1	محاجبية	زهور فتيحة	حي 140 مسكن بلوك 5 رقم141 وهران
s2	دليل	قادة مختار	حي 114 مسكن عمارة ب 9 رقم 85 مستغانم

Pair_id	Lev LN	Jac LN	Lev FN	Jac FN	Lev Adr	Jac Adr	Score
r1-s1	0.88	0.00	0.91	0.34	1.0	1.0	2.79
r1-s2	0.14	0.00	0.10	0.00	1.0	1.0	1.24
r2-s1	0.13	0.00	0.18	0.00	1.0	1.0	1.31
r2-s2	0.80	0.00	0.00	1.00	1.0	1.0	2.80

Figure 4.2: *Quatre enregistrements et leur WVscore correspondant. Lev et Jac indiquent respectivement la similarité de Levenshtein et de Jaccard. LN, FN et Adr désignent respectivement le nom, le prénom et l'adresse.*

Par exemple, pour les quatre enregistrements de la figure 4.2, les blocs (r_1, s_1), (r_2, s_2) et (r_1, s_1, r_2, s_2) seront générés par Token blocking. Le bloc (r_1, s_1, r_2, s_2) sera éliminé par Block Purging car le nombre de comparaison impliqué par ce bloc est supérieur au *seuil de purging* qui 2 dans notre exemple.

4.5 Génération des données d'apprentissage

Au lieu d'utiliser des exemples étiquetés manuellement, notre système génère son propre ensemble de données d'apprentissage (exemples d'apprentissage positifs et négatifs). Par rapport au travail présenté dans (Christen, 2007), notre générateur de données d'apprentissage combine plusieurs mesures de similarité (similarité basée sur les caractères et similarité basée sur les tokens) et prend en charge plusieurs types d'erreurs (erreurs typographiques et erreurs de réarrangement de mots). L'ensemble de données d'apprentissage généré est ensuite utilisé pour apprendre un modèle de classification. Le pseudo-code du générateur de données d'apprentissage (TSG) est présenté dans l'algorithme 4.1.

Algorithm 4.1 Générateur des données d'apprentissage

Entrées :

- *Cpairs*: paires d'enregistrements candidates
- *np*: nombre d'exemples d'apprentissage positifs
- *nn*: nombre d'exemples d'apprentissage négatifs

Sorties :

- *PE*: ensemble des exemples d'apprentissage positifs
- *NE*: ensemble des exemples d'apprentissage négatifs

1 Initialiser les ensembles vides *PE* et *NE*
2 Initialiser l'ensemble vide *WV*
3 pour toutes les paires d'enregistrement (*r, s*) de *Cpairs* faire
4 *WVscore* (*r,s*) = *computeScoreWV*(*r,s*)
5 ajouter *WVscore* (*r,s*) à *WV*
6 fin pour

7 Trier *WV* par ordre décroissant en fonction des scores calculés dans l'étape 4
8 Placer dans *PE* les *np* vecteurs de comparaison se trouvant au sommet de WV, tel qu'un enregistrement apparaisse au plus une fois dans une paire de *PE*
9 Permuter les paires de *PE* pour obtenir *nn* paires de *NE,* tel que *PE* ∩ *NE* est vide
10 Retourner *PE* and *NE*

L'algorithme 4.1 prend en entrée *Cpairs* l'ensemble des paires d'enregistrements candidates (résultat de l'étape d'indexation), *np* le nombre d'exemples d'apprentissage positifs et *nn* le nombre d'exemples d'apprentissage négatifs, et retourne l'ensemble des exemples positifs *PE* et l'ensemble des exemples négatifs *NE*. Les paramètres *np* et *nn* peuvent être initialisés à la même valeur pour avoir un problème de classification équilibré. Une autre alternative consiste à choisir *nn* > *np* pour avoir un problème de classification non équilibré. D'autre part, l'ensemble des vecteurs de comparaison et leurs scores *WV* aura un grand nombre de vecteurs de comparaison représentant des non-matches et un nombre réduit de vecteurs de comparaison représentant des matches. Pour cette raison, le nombre d'exemples positifs d'apprentissage *np* doit être inférieur ou égal au nombre d'enregistrements de la plus petite source de données ($np \leq min\ (|S|, |R|)$). Sinon, nous aurons des exemples d'apprentissage négatifs qui seront insérés dans l'ensemble des exemples d'apprentissage positifs *PE*.

Pour chaque paire d'enregistrements *s* et *r* appartenant à l'ensemble des paires d'enregistrements candidates *Cpairs*, l'algorithme 4.1 utilise la fonction *computeScoreWV* qui renvoie *WVscore* (un vecteur de comparaison et son score) (ligne 4). Chaque vecteur de comparaison est stocké avec son score dans l'ensemble *WV* (ligne 5). La figure 4.2 montre un exemple de quatre enregistrements et leurs *WVscore* associés. Les vecteurs de comparaison de l'ensemble *WV* sont triés d'une façon décroissante en fonction du score calculé par la fonction *computeScoreWV*. Ensuite, les *np* vecteurs de comparaison dont le score est le plus élevé sont étiquetés comme positifs et sont placés dans *PE* l'ensemble des exemples d'apprentissage positifs. Enfin, les paires de l'ensemble *PE* sont permutées pour obtenir les paires

représentant les non-matches de l'ensemble *NE*; ces paires sont étiquetées comme négatives. Une contrainte à noter dans la ligne 8 de l'algorithme 4.1 est qu'un enregistrement apparaît une fois au plus dans n'importe quelle paire de l'ensemble *PE*. Cette contrainte permettra de disposer d'un ensemble de données d'apprentissage qui est le plus représentatif que possible. Dans l'exemple de la figure 4.2, si nous initialisons le nombre d'exemples d'apprentissage positifs *np* et le nombre d'exemples d'apprentissage négatifs *nn* sur la même valeur qui est deux, nous aurons $PE = \{(r_1, s_1), (r_2, s_2)\}$ et $NE = \{(r_1, s_2), (r_2, s_1)\}$. Le Tableau 4.1 montre les données d'apprentissage générées par L'algorithme 4.1 et qui seront utilisées dans l'étape *génération du modèle de classification.*

Tableau 4.1: *Données d'apprentissage générées par L'algorithme 4.1.*

Pair_id	Lev LN	Jac LN	Lev FN	Jac FN	Lev Adr	Jac Adr	Classe
r2-s2	0.80	0.00	0.00	1.00	1.0	1.0	positive
r1-s1	0.88	0.00	0.91	0.34	1.0	1.0	positive
r2-s1	0.13	0.00	0.18	0.00	1.0	1.0	negative
r1-s2	0.14	0.00	0.10	0.00	1.0	1.0	negative

La fonction *computeScoreWV* (algorithme 4.2) prend en entrée deux enregistrements *s* et *r*, ainsi qu'un ensemble de mesures de similarité *SIM*, et renvoie *WVscore* (*r, s*), qui est un vecteur de comparaison et son score. La figure 4.2 montre quatre *WVscore* représentant les paires d'enregistrements (r_1, s_1), (r_1, s_2), (r_2, s_1) et (r_2, s_2). En réalité, les types d'erreur dans les sources de données ne sont pas connus à l'avance et une seule mesure de similarité ne peut pas prendre en charge tous les types d'erreur. L'idée de l'algorithme 4.2 est d'utiliser plusieurs mesures de similarité (similarité basée sur les caractères et similarité basée sur les tokens) pour prendre en charge plusieurs types d'erreurs (erreurs typographiques et erreurs de réarrangement de mots) pouvant exister dans les sources de données en anglais et en arabe. Dans l'exemple de la figure 4.2, la mesure de similarité Levenshtein renvoie la valeur 0 pour les deux prénoms (مختار قادة, قادة مختار) des enregistrements r_2 et s_2, alors qu'il

s'agit du même prénom. Cela est dû au fait que la mesure de similarité Levenshtein est une mesure de similarité basée sur les caractères qui prend en charge les erreurs typographiques, mais pas les erreurs de réarrangement des mots.

Pour chaque attribut a_i, la fonction *computeScoreWV* calcule la similarité entre les deux valeurs $r[a_i]$ et $s[a_i]$ ($t[a_i]$ représente la valeur de l'attribut a_i dans l'enregistrement t) en utilisant les mesures de similarité de l'ensemble *SIM* (lignes 3 à 11 de l'algorithme 4.2). Dans l'exemple de la figure 4.2, deux mesures de similarité ont été utilisées pour chaque attribut, une similarité basée sur les caractères (Levenshtein) et une similarité basée sur les tokens (Jaccard). La fonction *computeScoreWV* considère uniquement la valeur maximale (*valmax*) parmi les valeurs renvoyées par les mesures de similarité de l'ensemble *SIM* (ligne 8 de l'algorithme 4.2). Dans l'exemple de la figure 4.2, la fonction *computeScoreWV* calcule la similarité entre les deux prénoms (مختار قادة, قادة مختار) des enregistrements r_2 et s_2 en utilisant les deux mesures de similarité citées ci-dessus et considère la valeur 1 qui est le maximum entre les valeurs renvoyées par les deux mesures de similarité. Pour les deux noms de famille (دليل, دلييل) des enregistrements r_2 et s_2, la valeur 0.80 est prise en compte alors que pour les deux adresses, la valeur 1 est retenue. Enfin, les valeurs maximales sont additionnées pour obtenir le score final du vecteur de comparaison représentant la paire d'enregistrements (r, s). En continuant avec le même exemple, le vecteur de comparaison représentant la paire d'enregistrements (r_2, s_2) aura le score de 2.80.

La complexité de l'algorithme 4.1 dépend de la taille de l'ensemble des paires d'enregistrements candidates (*Cpairs*) et la complexité de la fonction *computeScoreWV* (algorithme 4.2) qui est $O(|SIM|\times|A|)$, où *SIM* représente l'ensemble des mesures de similarité et A représente l'ensemble des attributs. Au total, la complexité de l'algorithme 4.1 est $O(|Cpairs|\times|SIM|\times|A|)$.

Algorithm 4.2 Fonction computeScoreWV

Entrées :

- *s* et *r* : deux enregistrements
- *SIM* : ensemble de mesures de similarité

Sorties :

- *WVscore* (*r,s*): un vecteur de comparaison et son score

1 double *score*: = 0.0
2 String *w*: = *idr*+"::"+*ids*
3 pour chaque attribut a_i
4 *valmax*:=0
5 pour chaque sim_j dans *SIM*
6 *val* := sim_j (*r*[a_i],*s*[a_i])
7 *w* := *w* + ";" + *val*
8 si (*val*>*valmax*) alors *valmax* := *val*
9 fin pour
10 *score*: = *score* + *valmax*
11fin pour
12 *WVscore* (*r,s*) := *w* + *score*
13 Retourner *WVscore* (*r,s*)

4.6 Génération du modèle de classification et classification

Dans l'étape génération du modèle de classification, l'ensemble des données d'apprentissage généré (*PE* et *NE*) à l'étape précédente est utilisé pour trainer un classificateur. En reprenant l'exemple de la section précédente, l'ensemble des exemples positifs $PE = \{(r_1, s_1), (r_2, s_2)\}$ et l'ensemble des exemples négatifs $NE = \{(r_1, s_2), (r_2, s_1)\}$ seront utilisés comme données d'apprentissage par le classificateur. Le résultat de cette étape est un modèle de classification qui sera utilisé lors de l'étape de classification pour classer les vecteurs de comparaison non étiquetés en tant que matches et non-matches.

Plusieurs classificateurs ont été utilisés pour la résolution d'entité comme SVM et les arbres de décision. Des études antérieures ont validé empiriquement que

SVM est plus performant que les arbres de décision (Kopcke et al., 2010) et que SVM avec un noyau fonction de base radiale (RBF) est plus performant que l'apprentissage d'ensemble avec arbres de décision comme classificateur de base (Bilenko & Mooney, 2003a) lorsque l'ensemble des données d'apprentissage est limité. Dans notre système, nous utilisons SVM avec un noyau RBF car il ne nécessite pas l'ajustement de nombreux paramètres tels que le noyau polynomial. De plus, le noyau linéaire (et pour certains paramètres, le noyau sigmoïde) est un cas particulier du noyau RBF (Hsu, Chang & Lin, 2003).

4.7 Conclusion

Dans ce chapitre, nous avons présenté notre approche pour effectuer la résolution d'entité. Comme souligné dans le chapitre 3, pour qu'un système de résolution d'entité soit utilisable dans tous les domaines et par tous les utilisateurs, il doit présenter un degré élevé d'automatisation. Dans ce contexte, nous avons présenté un système de résolution d'entité entièrement automatique (non supervisé) qui permet aussi de prendre en charge les sources de données en anglais ainsi que les sources de données en arabe.

Le système proposé fonctionne en plusieurs étapes. La première étape est une étape de prétraitement qui permet de prendre en charge les sources de données en anglais et en arabe en convertissant les données au système Unicode. La deuxième étape qui est l'indexation, est utilisée pour réduire l'espace de recherche en générant un ensemble de paires d'enregistrements candidates qui sont comparées en détail dans les étapes suivantes. La méthode d'indexation utilisée est automatique et ne nécessite pas la spécification de clé de blocage. Le troisième module du système proposé est un générateur de données d'apprentissage (TSG) qui permet de générer automatiquement un ensemble d'exemples d'apprentissage positifs et négatifs qui sont utilisés pour la prise en charge d'une exécution complètement automatique de la procédure de résolution d'entité. Le TSG utilise les scores calculés par plusieurs

mesures de similarité afin de prendre en charge différents types d'erreurs qui peuvent exister dans les sources de données en anglais ainsi que celles en arabe. Enfin, les données d'apprentissage générées sont utilisées pour trainer un classificateur et apprendre un modèle de classification utilisé pour classer les vecteurs de comparaison non étiquetés.

Le système proposé a été implémenté et testé sur plusieurs sources de données en anglais et d'autres en arabe. Les résultats obtenus sont très encourageants. L'expérimentation et l´évaluation de notre système sont présentées dans le chapitre qui suit.

Chapitre 5

Implémentation et évaluation

5.1 Introduction

Dans le chapitre précédent, nous avons présenté un système de résolution d'entité entièrement automatique qui permet de prendre en charge les sources de données en anglais ainsi que celles en arabe. Le système proposé a été implémenté en Java. L'open source SimMetrics (http://sourceforge.net/projects/simmetrics/) qui contient l'implémentation de plusieurs mesures de similarité a été utilisé pour la comparaison des valeurs d'attributs. Pour la classification, nous avons utilisé LIBSVM (Chang & Lin, 2011).

Ce chapitre a pour objectif d'évaluer et de tester les performances du système proposé. La première série d'expérimentations consiste à évaluer le générateur de données d'apprentissage (TSG) proposé alors que la deuxième série d'expérimentations consiste à évaluer le système dans sa totalité qui est le résultat final de la classification.

Les évaluations présentées dans ce chapitre ont été effectuées sur une machine avec 4 Go de RAM et un processeur Intel i5 3230M à 2,60 GHz.

5.2 Données de test

Pour l'expérimentation, nous avons utilisé trois sources de données en arabe et quatre sources de données en anglais. Les sources de données en langue arabe n'étant pas disponibles, nous avons utilisé une source de données *ds* qui contient

8340 enregistrements avec trois attributs: prénom, nom et adresse. À partir de cette source de données *ds*, nous avons extrait 1000 enregistrements de manière aléatoire pour obtenir la source de données *d*. Ensuite, à partir de la source de données *d*, nous avons généré trois sources de données *d1*, *d2* et *d3* en injectant différents types d'erreur. La source de données *d1* contient des erreurs typographiques (insertion, suppression ou remplacement d'un seul caractère), la source de données *d2* contient des erreurs de réarrangement de mots, tandis que la source de données *d3* contient différents types d'erreurs (typographiques et réarrangements de mots). Enfin, nous avons effectué la résolution d'entité non supervisée sur les cas de test contenant différents types d'erreur (*ds* vs *d1*, *ds* vs *d2* et *ds* vs *d3*).

Pour la langue anglaise, nous avons utilisé quatre sources de données réelles couramment utilisés pour évaluer les approches d'ER: DBLP-ACM, DBLP-Scholar, Abt-Buy et Amazon-GP (Kopcke et al., 2010). ACM-DBLP et DBLP-Scholar couvrent le domaine bibliographique, tandis que les deux autres sources de données couvrent le domaine du commerce électronique.

Comme mentionné dans le chapitre précédent, notre système utilise Token blocking suivi de *Block Purging* (Papadakis et al., 2013) comme méthode d'indexation automatique. Le ratio de réduction (RR) et la complétude des paires d'enregistrements (PC) sont utilisés pour évaluer les méthodes d'indexation (Christen, 2012b). Ces mesures sont calculées par l'équation 5.1 et l'équation 5.2, respectivement.

$$RR = 1 - \frac{|Cpairs|}{|S| \times |R|} \tag{5.1}$$

$$PC = \frac{|Cpairs \cap Mpairs|}{|Mpairs|} \tag{5.2}$$

Cpairs représente l'ensemble des paires d'enregistrements candidates générées par la méthode d'indexation, *S* et *R* les deux sources données et *Mpairs* l'ensemble des paires d'enregistrements qui correspondent réellement à des matches (selon le

mapping parfait). Le tableau 5.1 présente les caractéristiques des sources de données utilisées dans l'expérimentation ainsi que les résultats de la méthode d'indexation.

5.3 Métriques

Pour évaluer le système proposé, différentes mesures sont utilisées. Pour l'étape de classification, nous nous sommes basés sur les métriques utilisées dans le domaine de la recherche d'information et sont aussi utilisées dans le domaine de la résolution d'entité: la précision, le rappel et la F-mesure (Christen, 2012a). Ces mesures sont calculées par les équations suivantes:

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.4}$$

$$F - mesure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.5}$$

TP (true positives) est le nombre des matches détectés (doublons) et qui sont vraiment des matches, *FP* (false positives) est le nombre des matches détectés et qui sont vraiment des non-matches, *FN* (false negatives) est le nombre des non- matches détectés et qui sont vraiment des matches.

La génération des données d'apprentissage est évaluée par la qualité des exemples positifs (*Qpositive*): le pourcentage des vrais matches dans les exemples d'apprentissage positifs (*PE*) et la qualité des exemples négatifs (*Qnegative*): le pourcentage des vrais non-matches dans les exemples d'apprentissage négatifs (*NE*) (Christen, 2007). Ces mesures sont calculées par l'équation 5.6 et l'équation 5.7, respectivement.

$$Qpositive = \frac{\left|\text{true matches in } PE\right|}{\left|PE\right|} \tag{5.6}$$

$$Qnegative = \frac{|\text{true non - matches in } NE|}{|NE|} \quad (5.7)$$

Tableau 5.1: *Caractéristiques des sources de données utilisées dans les expérimentations.*

Source de données	Nombre d'enregistrements	Complétude des paires	Ratio de réduction	Nombre des vrais matches
ds-d1	8340/1000	99.60	97.97	1000
ds-d2	8340/1000	100	97.55	1000
ds-d3	8340/1000	100	97.52	1000
DBLP-ACM	2616/2294	99.82	95.78	2224
DBLP-Scholar	2616/64263	98.65	99.57	5347
Abt-Buy	1081/1092	97.86	89.86	1097
Amazon-GP	1363/3226	99.18	87.21	1300

5.4 Expérimentations et résultats

5.4.1 Evaluation du générateur de données d'apprentissage

L'objectif de cette expérimentation est d'évaluer le générateur de données d'apprentissage proposé (Proposed TSG). Le TSG proposé est comparé avec l'approche présentée dans (Christen, 2007) qui utilise une seule mesure de similarité (JaroWinkler) pour générer les données d'apprentissage (two-step TSG). Pour générer les exemples d'apprentissage, la fonction *computeScoreWV* (algorithme 4.2) utilise deux mesures de similarité: une similarité basée sur les caractères (Levenshtein) et une similarité basée sur les tokens (Jaccard).

Le TSG proposé (voir l'algorithme 4.1) prend en entrée le nombre d'exemples d'apprentissage positifs (*np*) et le nombre d'exemples d'apprentissage négatifs (*nn*). Dans cette expérimentation, *np* a été choisi pour être inférieur au nombre d'enregistrements de la plus petite source de données ($np < min(|S|, |R|)$) afin d'éviter d'introduire des exemples d'apprentissage négatifs dans l'ensemble d'exemples d'apprentissage positifs (*PE*). Les approches de résolution d'entité

supervisées et semi-supervisées utilisent un pourcentage d'exemples d'apprentissage sélectionnés à partir du *mapping parfait* (*np* = *x%* du *mapping parfait*). Etant donné que notre système est non supervisé, nous utilisons un pourcentage d'exemples positifs de la plus petite source de données. Dans la suite de ce chapitre, nous notons *x%* exemples positifs comme utilisant *x%* de la plus petite source de données (*np* = *x%* × *min* (|*S*|, |*R*|). Le nombre d'exemples négatifs *nn* peut être initialisé à la même valeur que *np* (*nn* = *np*) ou à une valeur supérieure à *np* (*nn* = *np* × *n* où *n* est un nombre naturel). En général les vecteurs de comparaison contiennent un nombre important de non-matches et un nombre réduit de matches. Pour cette raison, nous avons utilisé *nn* = *np* × 10 dans notre système.

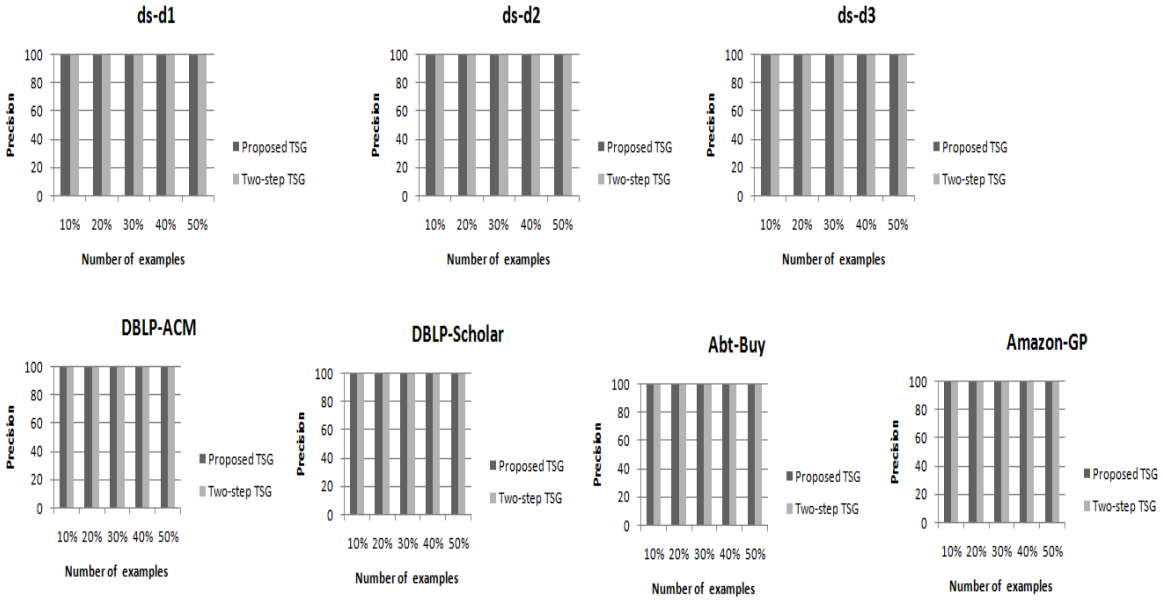

Figure 5.1: *Comparaison entre* Proposed TSG *et* two-step TSG *en termes de qualité des exemples d'apprentissage négatifs générés (Qnegative).*

Comme le montre la figure 5.1, les exemples d'apprentissage négatifs générés par les deux approches sont de très bonne qualité pour toutes les sources de données et pour toutes les valeurs de *nn* (*Qnegative* entre 99,60% et 100%). Cela est dû au

fait que les vecteurs de comparaison contiennent un nombre important de non-matches qui sont faciles à détecter.

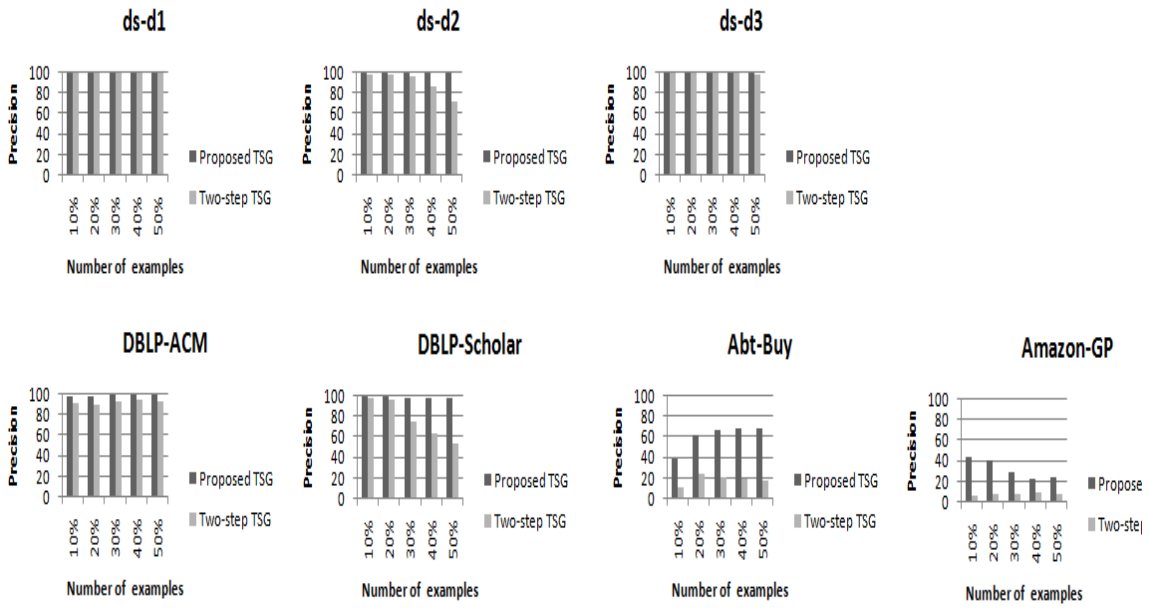

Figure 5.2: *Comparaison entre* Proposed TSG *et* two-step TSG *en termes de qualité des exemples d'apprentissage positifs générés (Qpositive).*

La figure 5.2 montre la qualité des exemples d'apprentissage positifs générés par Proposed TSG (notre système) par rapport à l'approche two-step TSG pour différentes valeurs de *np*. Pour la source de données *ds-d1*, les exemples positifs générés par les deux TSG sont de très bonne qualité. En effet, la source de données *d1* contient des erreurs typographiques qui sont prises en charge par Proposed TSG (utilisant Levenshtein) et two-step TSG (utilisant JaroWinkler). Pour les sources de données *ds-d3*, les performances du two-step TSG atteignent 97% de *Qpositive* pour *np* = 30%, tandis que Proposed TSG atteint 100% de *Qpositif* pour le même nombre d'exemples positifs. En effet, la source de données *d3* contient des erreurs typographiques et des erreurs de réarrangement de tokens. Pour la source de données *ds-d2*, Proposed TSG (100% en moyenne) dépasse largement two-step TSG (89,53% en moyenne) en termes de *Qpositive*. Cela peut s'expliquer par le fait que la

source de données *d2* contient des erreurs de réarrangement de tokens qui ne sont pas prises en charge par JaroWinkler, alors que Proposed TSG utilise plusieurs mesures de similarité et prend en charge plusieurs types d'erreur.

Enfin, pour les sources de données réelles, Proposed TSG dépasse two-step TSG pour toutes les valeurs de *np* et pour toutes les sources de données. Pour *np* = 30%, Proposed TSG dépasse two-step TSG de 6,25% sur *DBLP-ACM*, de 34,3% sur *DBLP-Scholar*, de 45,35% sur *Abt-Buy* et de 20,44% sur *Amazon-GP*. Cela peut s'expliquer par le fait que two-step TSG utilise une seule mesure de similarité et ne prend pas en charge les erreurs de réarrangement de tokens, alors que Proposed TSG utilise plusieurs mesures de similarité et prend en charge plusieurs types d'erreur.

5.4.2 Evaluation de la classification

L'objectif de cette expérimentation est de tester la procédure de résolution d'entité dans sa totalité. Les données d'apprentissage générées par le TSG proposé sont utilisées par un classificateur pour apprendre un modèle de classification qui sera ensuite utilisé pour classer les vecteurs de comparaison non étiquetés (n'appartenant pas à *PE* et *NE*), ce classificateur sera nommé *non supervisé proposé*, car il utilise les données d'apprentissage générées automatiquement par le TSG proposé. Le nombre d'exemples d'apprentissage utilisés pour l'apprentissage doit être suffisamment important pour représenter correctement les caractéristiques des sources données, mais pas trop pour éviter les problèmes de sur-ajustement (*overfitting*). Kejriwal and Miranker (2015) ont constaté que le classificateur avait tendance à sur-ajuster les données lorsque le nombre d'exemples dépasse 30% du *mapping parfait*. Dans cette expérimentation, le classificateur est trainé utilisant *np* exemples positifs ($np = 30\% \times min(|S|, |R|)$ et *nn* exemples négatifs ($nn = np \times 10$).

L'approche non supervisée proposée est comparée avec trois autres approches. La première est une approche supervisée (*supervisée*) qui a accès au *mapping parfait* (exemples d'apprentissage parfaitement étiquetés) et utilise le même nombre d'exemples (sans bruit) que l'approche non supervisée proposée. La deuxième et la

troisième approche ne sont pas supervisées et reposent sur l'optimisation de la valeur d'une fonction objective appelée pseudo F-mesure à travers les données non étiquetées (Ngomo & Lyko, 2013). La deuxième approche apprend un classificateur linéaire (*linéaire*) tandis que la troisième apprend un classificateur booléen (*booléen*). Pour la comparaison des valeurs d'attribut, deux mesures de similarité ont été utilisées: une similarité basée sur les caractères (Levenshtein) et une similarité basée sur les tokens (Jaccard). Comme mentionné précédemment, le classificateur utilisé dans cette expérimentation est SVM avec un noyau RBF (Radial Basis Function).

Tableau 5.2: *Comparaison du système proposé avec d'autres systèmes*

.

Source de données	**Linéaire**	**booléen**	**supervisée**	**non supervisé proposé**
ds-d1	99,20	94,25	99,74	99,05
ds-d2	98,37	98,91	99,95	99,70
ds-d3	91,48	92,03	99,15	98,61
dblp-acm	90,99	91,09	97,13	92,91
DBLP – Scholar	50,08	50,08	76,17	74,23
Abt-Buy	33,22	33,22	41,20	33,10
Amazon-Gp	35,76	33,30	43,73	28,94
Moyenne	**71,30**	**70,41**	**79,58**	**75,22**

Le tableau 5.2 montre les résultats de classification en terme de la F-mesure sur les sept cas tests. L'approche supervisée a donné les meilleurs résultats comme prévu, atteignant 79,58% de F-mesure en moyenne. Cela est dû au fait qu'elle a accès à des données d'apprentissage parfaitement étiquetées. L'approche non supervisée proposée atteint en moyenne 75,22% de F-mesure et dépasse l'approche linéaire de 3,92% et l'approche booléenne de 4,81% sans être itérative.

Les résultats de la F-mesure du tableau 5.2 montrent que le système proposé est compétitif par rapport aux approches supervisées qui ont accès à des exemples d'apprentissage parfaitement étiquetés et dépasse les approches non supervisées proposées récemment en termes de F-mesure. Les résultats obtenus montrent que

notre système peut être utilisé pour effectuer la résolution d'entité pour les sources de données en anglais ainsi que les sources de données en arabe d'une façon automatique et non itérative.

5.5 Conclusion

Dans ce chapitre, nous avons présenté l'expérimentation que nous avons menée dans le cadre de cette thèse. Sur le plan pratique, nous avons implémenté le système de résolution d'entité proposé, ensuite nous avons évalué le système proposé sur sept cas de test (trois en arabe et quatre en anglais) qui couvrent trois domaines (domaine des personnes, domaine bibliographique et domaine du commerce électronique). Les résultats obtenus montrent que notre générateur de données d'apprentissage est plus performant en termes de précision (*Qpositive*) que le générateur de données d'apprentissage présenté dans (Christen, 2007) et cela sur les sources de données en anglais et celle en arabe.

La deuxième expérimentation concerne la procédure de résolution d'entité dans sa totalité. Nous avons comparé notre système avec deux systèmes itératifs non supervisés ainsi qu'une approche supervisée. Les résultats obtenus montrent que notre système est plus performant que les systèmes itératifs non supervisés et compétitif avec l'approche supervisée.

Chapitre 6

Conclusion générale

6.1 Conclusion

La résolution d'entité est définie comme l'identification de différentes descriptions ou enregistrements qui représentent la même entité du monde réel. Ces représentations considérées comme des doublons ou similaires sont dues à des erreurs et à des incohérences dans les données, telles que des fautes de saisie et des fautes d'orthographe, des informations manquantes ou des données obsolètes. La résolution d'entité est une étape importante dans les processus de nettoyage et d'intégration des données qui permet d'améliorer la qualité des données. Dans le chapitre 2, nous avons présenté brièvement les concepts de la qualité des données, la procédure d'intégration des données et les étapes de résolution d'entité.

Dans le chapitre 1 de la thèse, nous avons souligné le fait qu'un système de résolution d'entité doit simultanément satisfaire trois exigences : l'indépendance au domaine, l'automatisation et la prise en charge de la langue arabe en plus de la langue anglaise. L'indépendance au domaine est nécessaire car la résolution d'entité peut se faire dans de nombreux domaines comme le domaine bibliographique, le domaine du commerce électronique, le domaine des personnes, etc. Un système de résolution d'entité ne doit pas être conçu pour répondre aux besoins spécifiques d'un domaine particulier tels que les noms. L'automatisation est nécessaire à cause du coût élevé de l'expertise du domaine et de l'existence de domaines inexpérimentés. Un système de résolution d'entité devrait présenter un degré élevé d'automatisation,

ce qui signifie qu'il ne nécessite pas l'intervention d'un expert du domaine pour élaborer des données d'apprentissage ou des règles de matching. La prise en charge de la langue arabe dans plusieurs domaines est nécessaire car la plupart des systèmes de résolution d'entité ont été conçus pour la langue anglaise alors que les approches de résolution d'entité qui prennent en charge la langue arabe se sont concentrées sur le matching des noms, ce qui signifie qu'elles sont dépendantes du domaine des noms en arabe et ne peuvent pas être utilisées dans d'autres domaines que les noms en arabe.

Le développent d'un système de résolution d'entité entièrement automatique qui prend en charge la langue arabe ainsi que la langue anglaise a été motivé dans le chapitre 3 en présentant un état de l'art qui décrit les différents systèmes existants de résolution d'entité, suivi par une étude comparative des différents travaux abordés selon plusieurs critères.

Partant du résultat de ces études, dans le chapitre 4, nous avons proposé un nouveau système de résolution d'entité entièrement non supervisée (automatique) qui prend en charge les sources de données en anglais ainsi que celles en arabe. Le système proposé peut fonctionner sur n'importe quelles sources de données dans n'importe quel domaine, prend en charge les sources de données en anglais ainsi que les sources de données en arabe et ne nécessite pas l'intervention d'un expert du domaine pour l'élaboration de données d'apprentissage ou de règles de matching.

Le système proposé comprend plusieurs modules à savoir: le prétraitement, l'indexation, la génération des données d'apprentissage, la génération du modèle de classification et la classification. L'étape de prétraitement permet de prendre en charge les sources de données en anglais ainsi que celles en arabe en utilisant le système Unicode. La deuxième étape, l'indexation, utilise Token blocking qui est une méthode d'indexation automatique qui ne nécessite pas la spécification d'une clé de blocage. Le troisième module constitue la contribution principale de cette thèse qui est un algorithme non supervisé appelé générateur de données d'apprentissage (TSG), qui utilise plusieurs mesures de similarité pour générer un

ensemble d'exemples d'apprentissage utilisé pour automatiser la procédure de résolution d'entité. Une fois générés, les exemples d'apprentissage sont utilisés pour apprendre un modèle de classification qui est utilisé pour classer les vecteurs de comparaison non étiquetés en matches et non-matches.

Le système de résolution d'entité proposé a été implémenté pour ensuite être validé. Nous avons mis en valeur l'intérêt de notre approche par des expérimentations appliquées sur sept cas de test (trois en arabe et quatre en anglais) qui couvrent trois domaines (domaine des personnes, domaine bibliographique et domaine du commerce électronique). Les résultats obtenus montrent que notre système est plus performant que les systèmes itératifs non supervisés et compétitif avec les approches supervisées.

6.2 Perspectives

Bien que cette thèse ait apporté un progrès dans la réalisation de l'objectif général de qualité et intégration des données et en particulier la procédure de résolution d'entité, elle soulève également plusieurs possibilités d'amélioration et d'autres questions à traiter dans les travaux futurs. Un des travaux futurs est de tester la précision et la mise en échelle de notre système sur des sources de données volumineuses en anglais et en arabe, dans d'autres domaines.

Une tendance récente dans la mise en œuvre d'application est l'utilisation de MapReduce (MR) qui est un modèle de programmation populaire pour le traitement parallèle sur des infrastructures en nuage comportant jusqu'à plusieurs milliers de nœuds (Dean & Ghemawat, 2008). Un autre travail futur est la parallélisation de notre système et en particulier le générateur de données d'apprentissage en utilisant des distributions MR comme Hadoop.

La résolution d'entité discutée dans cette thèse se focalise sur le matching de deux sources de données. Dans certaines applications, telles que les comparaisons de produits en ligne, des enregistrements provenant de plus de deux sources doivent

être matchés. Un troisième travail futur est d'étendre notre système pour effectuer la résolution d'entité de plusieurs sources de données.

Le paradigme des données liées a évolué pour devenir un moyen de transition du Web orienté document vers le Web sémantique. Parmi les principes qui sont utilisés pour stipuler la manière de publication des données liées, est que les données devraient être liées à des ensembles de données existants (Schmachtenberg, Bizer & Paulheim, 2014). Un quatrième travail futur est d'étendre notre système pour effectuer la résolution d'entité pour les données liées.

Références bibliographiques

Aizenstein, H., & Pitt, L. (1995). On the learnability of disjunctive normal form formulas. Machine Learning, 19(3), 183-208.

Aqeel, S. U., Beitzel, S., Jensen, E., Grossman, D., & Frieder, O. (2006). On the development of name search techniques for Arabic. Journal of the American Society for Information Science and Technology, 57(6), 728-739.

Arasu, A., Ganti, V., & Kaushik, R. (2006, September). Efficient exact set-similarity joins. In Proceedings of the 32nd international conference on Very large data bases (pp. 918-929). VLDB Endowment.

Batini, C., & Scannapieco, M. (2006). Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications. Springer-Verlag, New York, Ine Secaucus, NJ, USA.

Batini, C., Cappiello, C., Francalanci, C., & Maurino, A. (2009). Methodologies for data quality assessment and improvement. ACM computing surveys (CSUR), 41(3), 16.

Bellahsene, Z., Bonifati, A., & Rahm, E. (2011). Schema matching and mapping (Vol. 20). Heidelberg (DE): Springer.

Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S. E., & Widom, J. (2009). Swoosh: a generic approach to entity resolution. The VLDB Journal—The International Journal on Very Large Data Bases, 18(1), 255-276.

Bilenko, M., & Mooney, R. J. (2003a, August). Adaptive duplicate detection using learnable string similarity measures. In Proceedings of the ninth ACM

SIGKDD international conference on Knowledge discovery and data mining (pp. 39-48). ACM.

Bilenko, M., & Mooney, R. J. (2003b, August). On evaluation and training-set construction for duplicate detection. In Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation (pp. 7-12).

Bilenko, M., Basil, S., & Sahami, M. (2005, November). Adaptive product normalization: Using online learning for record linkage in comparison shopping. In Fifth IEEE International Conference on Data Mining (ICDM'05) (pp. 8-pp). IEEE.

Bilenko, M., Kamath, B., & Mooney, R. J. (2006, December). Adaptive blocking: Learning to scale up record linkage. In Sixth International Conference on Data Mining (ICDM'06) (pp. 87-96). IEEE.

Bleiholder, J., & Naumann, F. (2006). Conflict handling strategies in an integrated information system. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik.

Bleiholder, J., & Naumann, F. (2009). Data fusion. ACM Computing Surveys (CSUR), 41(1), 1.

Bleiholder, J., & Naumann, F. (2010). Data fusion and conflict resolution in integrated information systems (Doctoral dissertation, University of Potsdam).

Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical science, 235-249.

Boufares, F., & Salem, A. B. (2012, March). Heterogeneous data-integration and data quality: Overview of conflicts. In 2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT) (pp. 867-874). IEEE.

Boufares, F., Salem, A. B., Rehab, M., & Correia, S. (2013, May). Similar data elimination: MFB algorithm. In 2013 International Conference on Control, Decision and Information Technologies (CoDIT) (pp. 289-293). IEEE.

Bourne, C. P., & Ford, D. F. (1961). A study of methods for systematically abbreviating english words and names. Journal of the ACM (JACM), 8(4), 538-552.

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), 27.

Chaudhuri, S., Chen, B. C., Ganti, V., & Kaushik, R. (2007, September). Example-driven design of efficient record matching queries. In Proceedings of the 33rd international conference on Very large data bases (pp. 327-338). VLDB Endowment.

Chen, H., Chung, W., Xu, J. J., Wang, G., Qin, Y., & Chau, M. (2004). Crime data mining: a general framework and some examples. computer.

Christen, P. (2007, December). A two-step classification approach to unsupervised record linkage. In Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70 (pp. 111-119). Australian Computer Society, Inc..

Christen, P. (2008, January). Febrl: a freely available record linkage system with a graphical user interface. In Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80 (pp. 17-25). Australian Computer Society, Inc..

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

Christen, P. (2012b). A survey of indexing techniques for scalable record linkage and deduplication. IEEE transactions on knowledge and data engineering, 24(9), 1537-1555.

Christen, P., & Goiser, K. (2007). Quality and complexity measures for data linkage and deduplication. In Quality measures in data mining (pp. 127-151). Springer, Berlin, Heidelberg.

Chua, C. E. H., Chiang, R. H., & Lim, E. P. (2003). Instance-based attribute identification in database integration. The VLDB Journal—The International Journal on Very Large Data Bases, 12(3), 228-243.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003, August). A comparison of string metrics for matching names and records. In Kdd workshop on data cleaning and object consolidation (Vol. 3, pp. 73-78).

Dal Bianco, G., Galante, R., Goncalves, M. A., Canuto, S., & Heuser, C. A. (2015). A practical and effective sampling selection strategy for large scale deduplication. IEEE Transactions on Knowledge and Data Engineering, 27(9), 2305-2319.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. Ecology, 26(3), 297-302.

Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In International workshop on multiple classifier systems(pp. 1-15). Springer, Berlin, Heidelberg.

Doan, A., Halevy, A., & Ives, Z. (2012). *Principles of data integration*. Elsevier.

Dong, X. L., & Srivastava, D. (2015). Big data integration. Synthesis Lectures on Data Management, 7(1), 1-198.

Draisbach, U., & Naumann, F. (2009, August). A comparison and generalization of blocking and windowing algorithms for duplicate detection. In Proceedings of the International Workshop on Quality in Databases (QDB) (pp. 51-56).

Draisbach, U., & Naumann, F. (2010). DuDe: The duplicate detection toolkit. In Proceedings of the International Workshop on Quality in Databases (QDB).

Eckerson, W. W. (2002). Data quality and the bottom line: Achieving business success through a commitment to high quality data. The Data Warehousing Institute, 1-36.

Elfeky, M. G., Verykios, V. S., & Elmagarmid, A. K. (2002). TAILOR: A record linkage toolbox. In Proceedings 18th International Conference on Data Engineering (pp. 17-28). IEEE.

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. IEEE Transactions on knowledge and data engineering, 19(1), 1-16.

El-Shishtawy, T. (2013). A hybrid algorithm for matching arabic names. arXiv preprint arXiv:1309.5657.

Euzenat, J., & Shvaiko, P. (2007). Ontology matching (Vol. 18). Heidelberg: Springer.

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Clio: Schema mapping creation and data exchange. In Conceptual modeling: foundations and applications(pp. 198-236). Springer, Berlin, Heidelberg.

Fellegi, I. P., & Sunter, A. B. (1969). A theory for record linkage. Journal of the American Statistical Association, 64(328), 1183-1210.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780), 1612.

Ghafour, H. H. A., El-Bastawissy, A., & Heggazy, A. F. A. (2011, November). AEDA: Arabic edit distance algorithm Towards a new approach for Arabic name matching. In The 2011 International Conference on Computer Engineering & Systems(pp. 307-311). IEEE.

Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan, S., & Srivastava, D. (2001, September). Approximate string joins in a database (almost) for free. In VLDB(Vol. 1, pp. 491-500).

Gu, Q., Li, Z., & Han, J. (2012). Generalized fisher score for feature selection. arXiv preprint arXiv:1202.3725.

Gueddah, H., Yousfi, A., & Belkasmi, M. (2012). Introduction of the weight edition errors in the levenshtein distance. International Journal of Advanced Research in Artificial Intelligence, 1(5), 30-32.

Hajishirzi, H., Yih, W. T., & Kolcz, A. (2010, July). Adaptive near-duplicate detection via similarity learning. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval (pp. 419-426). ACM.

Hall, P. A., & Dowling, G. R. (1980). Approximate string matching. ACM computing surveys (CSUR), 12(4), 381-402.

Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

Hernández, M. A., & Stolfo, S. J. (1995, June). The merge/purge problem for large databases. In ACM Sigmod Record (Vol. 24, No. 2, pp. 127-138). ACM.

Hernández, M. A., & Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. Data mining and knowledge discovery, 2(1), 9-37.

Herzog, T. N., Scheuren, F. J., & Winkler, W. E. (2007). Data quality and record linkage techniques. Springer Science & Business Media.

Higazy, A., El Tobely, T., Yousef, A. H., & Sarhan, A. (2013, November). Web-based Arabic/English duplicate record detection with nested blocking technique. In 2013 8th International Conference on Computer Engineering & Systems (ICCES) (pp. 313-318). IEEE.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

Isele, R., & Bizer, C. (2012). Learning expressive linkage rules using genetic programming. Proceedings of the VLDB Endowment, 5(11), 1638-1649.

Isele, R., & Bizer, C. (2013). Active learning of expressive linkage rules using genetic programming. Web Semantics: Science, Services and Agents on the World Wide Web, 23, 2-15.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaudoise Sci Nat, 37, 547-579.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association, 84(406), 414-420.

Jurczyk, P., Lu, J. J., Xiong, L., Cragan, J. D., & Correa, A. (2008). FRIL: a tool for comparative record linkage. In AMIA annual symposium proceedings (Vol. 2008, p. 440). American Medical Informatics Association.

Kejriwal, M., & Miranker, D. P. (2013, December). An unsupervised algorithm for learning blocking schemes. In 2013 IEEE 13th International Conference on Data Mining (pp. 340-349). IEEE.

Kejriwal, M., & Miranker, D. P. (2015, May). Semi-supervised instance matching using boosted classifiers. In European Semantic Web Conference (pp. 388-402). Springer, Cham.

Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: A comparison. Data & Knowledge Engineering, 69(2), 197-210.

Kopcke, H., Thor, A., & Rahm, E. (2010). Learning-based approaches for matching web data entities. IEEE Internet Computing, 14(4), 23-31.

Köpcke, H., Thor, A., Thomas, S., & Rahm, E. (2012, March). Tailoring entity resolution for matching product offers. In Proceedings of the 15th International Conference on Extending Database Technology (pp. 545-550). ACM.

Leitão, L., Calado, P., & Weis, M. (2007, November). Structure-based inference of XML similarity for fuzzy duplicate detection. In Proceedings of the

sixteenth ACM conference on Conference on information and knowledge management (pp. 293-302). ACM.

Lenzerini, M. (2002, June). Data integration: A theoretical perspective. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems(pp. 233-246). ACM.

Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710).

McCallum, A., Nigam, K., & Ungar, L. H. (2000, August). Efficient clustering of high-dimensional data sets with application to reference matching. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 169-178). ACM.

McGilvray, D. (2008). Executing data quality projects: Ten steps to quality data and trusted information (TM). Elsevier.

Michelson, M., & Knoblock, C. A. (2006, July). Learning blocking schemes for record linkage. In AAAI (Vol. 6, pp. 440-445).

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000, September). Schema mapping as query discovery. In VLDB (Vol. 2000, pp. 77-88).

Monge, A. E., & Elkan, C. (1996, August). The Field Matching Problem: Algorithms and Applications. In KDD (pp. 267-270).

Naumann, F., & Herschel, M. (2010). An introduction to duplicate detection. Synthesis Lectures on Data Management, 2(1), 1-87.

Newcombe, H. B., Kennedy, J. M., Axford, S. J., & James, A. P. (1959). Automatic linkage of vital records. Science, 130(3381), 954-959.

Ngomo, A. C. N., & Auer, S. (2011, June). LIMES—a time-efficient approach for large-scale link discovery on the web of data. In Twenty-Second International Joint Conference on Artificial Intelligence.

Ngomo, A. C. N., & Lyko, K. (2013, October). Unsupervised learning of link specifications: deterministic vs. non-deterministic. In Proceedings of the Ontology Matching Workshop.

Ngomo, A. C. N., & Lyko, K. (2012, May). Eagle: Efficient active learning of link specifications using genetic programming. In Extended Semantic Web Conference (pp. 149-163). Springer, Berlin, Heidelberg.

Ngomo, A. C. N., Lehmann, J., Auer, S., & Höffner, K. (2011). Raven–active learning of link specifications. Ontology Matching, 2011.

Nikolov, A., d'Aquin, M., & Motta, E. (2012, May). Unsupervised learning of link discovery configuration. In Extended Semantic Web Conference (pp. 119-133). Springer, Berlin, Heidelberg.

Nikolov, A., Uren, V., & Motta, E. (2007, October). KnoFuss: A comprehensive architecture for knowledge fusion. In Proceedings of the 4th international conference on Knowledge capture (pp. 185-186). ACM.

OUHAB, A., MALKI, M., BERRABAH, D., & BOUFARES, F. (2017). An Unsupervised Entity Resolution Framework for English and Arabic Datasets. International Journal of Strategic Information Technology and Applications (IJSITA), 8(4), 16-29.

Papadakis, G., Ioannou, E., Palpanas, T., Niederee, C., & Nejdl, W. (2013). A blocking framework for entity resolution in highly heterogeneous information spaces. IEEE Transactions on Knowledge and Data Engineering, 25(12), 2665-2682.

Papadakis, G., Svirsky, J., Gal, A., & Palpanas, T. (2016). Comparative analysis of approximate blocking techniques for entity resolution. Proceedings of the VLDB Endowment, 9(9), 684-695.

Phua, C., Lee, V., Smith, K., & Gayler, R. (2010). A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. the VLDB Journal, 10(4), 334-350.

Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. IEEE Data Eng. Bull., 23(4), 3-13.

Redman, T. C. (1998). The impact of poor data quality on the typical enterprise. Communications of the ACM, 41(2), 79-83.

Reyes-Galaviz, O. F., Pedrycz, W., He, Z., & Pizzi, N. J. (2017). A supervised gradient-based learning algorithm for optimized entity resolution. Data & Knowledge Engineering, 112, 106-129.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5), 513-523.

Sarawagi, S., & Bhamidipaty, A. (2002, July). Interactive deduplication using active learning. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 269-278). ACM.

Schmachtenberg, M., Bizer, C., & Paulheim, H. (2014, October). Adoption of the linked data best practices in different topical domains. In International Semantic Web Conference (pp. 245-260). Springer, Cham.

Sidi, F., Panahy, P. H. S., Affendey, L. S., Jabar, M. A., Ibrahim, H., & Mustapha, A. (2012, March). Data quality: A survey of data quality dimensions. In 2012 International Conference on Information Retrieval & Knowledge Management (pp. 300-304). IEEE.

Silva, R. M., Gonçalves, M. A., & Veloso, A. (2014). A Two stage active learning method for learning to rank. Journal of the Association for Information Science and Technology, 65(1), 109-128.

Sutinen, E., & Tarhio, J. (1995, September). On using q-gram locations in approximate string matching. In European Symposium on Algorithms (pp. 327-340). Springer, Berlin, Heidelberg.

Tejada, S., Knoblock, C. A., & Minton, S. (2002, July). Learning domain-independent string transformation weights for high accuracy object identification. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 350-359). ACM.

Thor, A., & Rahm, E. (2007, January). MOMA-A Mapping-based Object Matching System. In CIDR (pp. 247-258).

Ukkonen, E. (1992). Approximate string-matching with q-grams and maximal matches. Theoretical computer science, 92(1), 191-211.

Vogel, T., & Naumann, F. (2012). Automatic blocking key selection for duplicate detection based on unigram combinations. In Proceedings of the International Workshop on Quality in Databases (QDB).

Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). Silk-a link discovery framework for the web of data. LDOW, 538.

Wang, Q., Vatsalan, D., & Christen, P. (2015, May). Efficient interactive training selection for large-scale entity resolution. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 562-573). Springer, Cham.

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage.

Winkler, W. E. (1994). Advanced methods for record linkage.

Winkler, W. E. (1995). Matching and record linkage. Business survey methods, 1, 355-384.

Winkler, W. E. (1999). The state of record linkage and current research problems. In Statistical Research Division, US Census Bureau.

Winkler, W. E. (2005). Approximate string comparator search strategies for very large administrative lists. Statistics, 2.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). Managing gigabytes: compressing and indexing documents and images. Morgan Kaufmann.

Xiao, C., Wang, W., Lin, X., Yu, J. X., & Wang, G. (2011). Efficient similarity joins for near-duplicate detection. ACM Transactions on Database Systems (TODS), 36(3), 15.

Yan, S., Lee, D., Kan, M. Y., & Giles, L. C. (2007, June). Adaptive sorted neighborhood methods for efficient record linkage. In Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries (pp. 185-194). ACM.

Yousef, A. H. (2014). Cross-language personal name mapping. arXiv preprint arXiv:1405.6293.

Yousef, A. H. (2015). Cross language duplicate record detection in big data. In Big Data in Complex Systems (pp. 147-171). Springer, Cham.

Zhao, H., & Ram, S. (2005). Entity identification for heterogeneous database integration—a multiple classifier system approach and empirical evaluation. Information Systems, 30(2), 119-132.

Annexe

A.1 Mesures de similarité

Mesures de similarité basées sur les caractères

Les mesures basées sur les caractères calculent la similarité des chaînes en considérant les chaînes comme des séquences contiguës de caractères ou de tokens. Ces mesures prennent en charge les erreurs typographiques. Plusieurs mesures de ce type ont été proposées.

La distance d'édition: La distance d'édition entre deux chaînes *a* et *b* est le nombre minimum d'insertions, de suppressions et de remplacements de caractères nécessaires pour transformer la chaîne *a* en *b*. Dans sa forme la plus simple, chaque opération d'édition a un coût de 1. Cette version de la distance d'édition est également appelée distance de Levenshtein (Levenshtein, 1966).

Etant donné deux chaines de caractères *a* et *b* de taille $|a|$ et $|b|$ respectivement, leur similarité Levenshtein est calculé comme suit :

$$\text{Levenshtein}(a,b) = 1 - \frac{\text{dist}(a,b)}{\max(|a|,|b|)} \tag{A.1}$$

Plusieurs variantes de la distance d'édition ont été proposées pour améliorer son efficacité en utilisant différents coûts pour différents types d'éditions. Des extensions de la distance d'édition peuvent être trouvées dans (Christen, 2012a; Elmagarmid et al., 2007; Hall & Dowling,1980).

Jaro: Jaro (1989) a introduit une mesure de similarité qui tient compte de la longueur des deux chaînes, du nombre de caractères communs et des types d'erreurs

dans ces chaînes (insertions, omissions ou transpositions). La transposition signifie que deux caractères adjacents sont échangés dans les deux chaînes.

Etant donné deux chaines de caractères *a* et *b* de taille $|a|$ et $|b|$ respectivement, leur similarité Jaro est calculé comme suit :

$$\text{Jaro}(a,b) = \frac{1}{3} \times \left(\frac{c}{|a|} + \frac{c}{|b|} + \frac{c - t/2}{c} \right) \quad \text{(A.2)}$$

Avec *c* qui représente le nombre de caractères communs et *t* le nombre de transpositions des caractères communs. Les communs sont tous les caractères *a*[i], *b*[j] pour lesquels *a*[i] = *b*[j] et $|\text{i} - \text{j}| \leq \frac{1}{2} \min \{|a|, |b|\}$. Une transposition existe lorsque le caractère commun à la position i de *a* est différent du caractère commun à la position i de *b.*

Jaro-Winkler: Winkler (1990) a amélioré la mesure de similarité Jaro, en attachant plus d'importance aux préfixes (caractères initiaux) des chaînes comparées.

$$\text{Jaro - Winkler}\,(a,b) = \text{Jaro}\,(a,b) + l \times p \times (1 - \text{Jaro}\,(a,b)) \quad \text{(A.3)}$$

Avec *l* qui représente la longueur du préfix commun (en général limité à 4) et *p* est un facteur pour contrôler l'impact du préfix commun. Dans beaucoup de travaux, la valeur par défaut de *p* est 0.1.

Mesures de similarité basées sur les tokens

Les mesures de similarité basées sur les tokens ou les q-grams divisent une chaîne en sous-chaînes (tokens ou q-grams) et les considèrent comme des ensembles. Une simple tokenisation divise une chaîne en tokens en se basant sur les espaces. L'avantage de ces mesures est leurs robustesses vis-à-vis des réarrangements des mots. L'inconvénient est que les erreurs typographiques dans les tokens ne sont pas détectées. Une exception est la mesure de similarité Q-grams qui divise les chaînes en q-grams. Plusieurs mesures de ce type ont été proposées.

Q-Gram: Les mesures basée sur les q-grams (Ukkonen, 1992) décomposent d'abord les chaînes comparées en sous-chaînes de longueur *q*, appelées q-grams. Chaque chaine est représentée par un ensemble de q-grams. La similarité entre deux ensembles de q-grams est calculé en utilisant un coefficient comme le coefficient de Jaccard (Jaccard, 1901), le coefficient Dice (Dice, 1945), ou le coefficient de chevauchement. Une autre version de la similarité q-grams qui considère également la position des q-grams dans la chaîne a été proposée par (Sutinen & Tarhio, 1995).

Etant donné deux chaines de caractères *a* et *b* représentées par leurs ensembles de q-grams *Qa* et *Qb* respectivement, leur similarité Q-Gram est calculé comme suit en utilisant le coefficient Dice :

$$\text{Q - Gram}(a,b) = \frac{2|Qa \cap Qb|}{|Qa| + |Qb|} \tag{A.4}$$

Jaccard: Le coefficient de Jaccard (Jaccard, 1901) mesure la similarité entre des ensembles de tokens, il est défini comme la taille de l'intersection divisée par la taille de l'union des ensembles de tokens.

Etant donné deux chaines de caractères *a* et *b* représentées par leurs ensembles de tokens *Ta* et *Tb* respectivement, leur similarité Jaccard est calculé comme suit:

$$\text{Jaccard}(a,b) = \frac{|Ta \cap Tb|}{|Ta \cup Tb|} \tag{A.5}$$

TF-IDF: La similarité TF-IDF est une mesure bien connue dans le domaine de recherche d'informations. Cette mesure se compose de deux parties: le TF-IDF (Salton & Buckley, 1988) et la similarité cosinus. TF-IDF est un schéma de pondération mesurant l'importance des mots (c'est-à-dire les termes ou les tokens) en fonction de leur fréquence. Dans le contexte de recherche d'informations, l'objectif est de calculer la similarité entre deux documents. Un document est un ensemble de termes et appartenant à un corpus D qui est un ensemble de documents. TF (fréquence du terme) ou tf (*q*, *d*) reflète le nombre de fois que le terme *q* apparaît

dans le document *d*, alors qu'IDF (fréquence de document inverse) ou idf (*q*) représente le nombre de documents contenant *q*. Finalement, le poids TF-IDF s'obtient en multipliant les deux mesures TF.IDF. Dans le contexte d'ER, chaque enregistrement est considéré comme un sac à mot (ensemble de tokens) alors que le corpus est les deux sources de données à matcher. La mesure cosinus est une mesure de la similarité entre deux vecteurs. Elle est définie comme le produit interne des vecteurs divisé par le produit de leurs longueurs. Le TF-IDF de deux enregistrements est calculé comme suit :

$$\log \text{TF-IDF}(r, s) = \sum_{q \in r \cap s} w(r,q) \times w(s,q) \tag{A.6}$$

avec chaque enregistrement *t* et token *q* :

$$w(t, q) = \frac{p(t,q)}{\sqrt{\sum_{q \in t} p(t,q)^2}} \tag{A.7}$$

et avec

$$p(t,q) = \log(tf_{t,q} + 1) \times \log(\frac{|S| + |R|}{idf_q} + 1) \tag{A.8}$$

Les équations assument que *r* et *s* sont deux enregistrements des deux sources de données *S* et *R* respectivement, *w(t,q)* est le poids TF-IDF normalisé du terme *q* dans l'enregistrement *t*, $tf_{t,q}$ est la fréquence de *q* dans *t*, $|R| + |S|$ est le nombre total des enregistrements dans les deux source de données à matcher, et idf_q représente le nombre d'enregistrements contenant *q*.

Autres mesures

Les mesures discutées ci-dessus sont les plus utilisées pour l'ER. Dans cette section, nous résumons les mesures de similarité pour trois autres cas. D'autres mesures de similarité sont décrites dans (Christen, 2012a; Naumann & Herschel, 2010).

Les mesures de similarité hybrides combinent plusieurs mesures de similarité. Par exemple, Soft TF-IDF (Cohen et al., 2003) est une extension de TF-IDF pour prendre en compte la similarité entre les tokens individuels au lieu de leurs égalités. Un autre exemple de mesure hybride est la mesure Monge and Elkan (Monge & Elkan, 1996).

Les mesures de similarité phonétique prennent en compte la prononciation des mots, qui peuvent être très similaires malgré de grandes différences orthographiques. Soundex (Bourne & Ford, 1961) est un système de codage phonétique très utilisé. Soundex attribue des chiffres de code identiques aux groupes de consonnes phonétiquement similaires.

Les approches existantes pour comparer des données numériques sont plutôt primitives. Souvent, les nombres sont traités comme des chaînes de caractères. Cependant, aucune des mesures de similarité des chaînes évoquées précédemment ne donne des résultats satisfaisants lorsqu'elle est appliquée à des données numériques. D'autres approches plus raisonnables utilisent la différence absolue ou relative des nombres.

A.2 Apprentissage automatique

L'apprentissage automatique correspond au domaine se consacrant au développement et à l'analyse d'algorithmes qui permettent à une machine d'apprendre à partir d'un ensemble de données, i.e. d'extraire des concepts et des patrons caractérisant ces données (Witten, Frank, Hall & Pal, 2016). Bien que la motivation originale fût de permettre la mise sur pied de systèmes manifestant une intelligence artificielle, les algorithmes issus de ce domaine sont maintenant

répandus dans bien d'autres domaines, comme la bioinformatique, le commerce électronique, la recherche d'information et la résolution d'entité.

D'une façon plus formelle, Un algorithme d'apprentissage est un algorithme prenant en entrée un ensemble de données *D* et retournant une fonction *f*. *D* représente l'ensemble des données d'apprentissage (nommé aussi exemples d'apprentissage) et *f* représente le modèle appris.

L'ensemble de données *D* caractérise une série d'instances du phénomène à apprendre, nommés *exemples*. Dans le cas de *l'apprentissage supervisé* comme la classification ou la régression, chaque exemple **z** est constitué d'une description **x** et d'une étiquette y. La description $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ de l'exemple s'avère un vecteur de valeurs de dimension *d*, ou *d* est le nombre de *caractéristiques* (*attributs*) de l'exemple. L'´etiquette y de l'exemple détermine sa *classe* d'appartenance dans le cas de la classification. Dans le cas de *l'apprentissage non supervisé* comme le regroupement (*clusturing*), chaque exemple **z** est constitué uniquement de la description **x** et l'étiquette y n'existe pas. Dans le cas de la résolution d'entité, l'objet de cette thèse, la description **x** est le vecteur de comparaison et l'´etiquette y correspond à la classe matche ou non-matche. Les classificateurs les plus utilisés dans le domaine de la résolution d'entité sont les arbres de décision et SVM, alors que pour le clustring, K-means est le plus utilisé.

A.3 Arbre de décision

Les arbres de décision utilisent une approche «diviser pour régner». Les nœuds dans un arbre de décision impliquent de tester un attribut particulier. Habituellement, le test compare la valeur d'un attribut avec une constante. Un nœud feuille donne une classification qui s'applique à toutes les instances qui atteignent cette feuille. Pour classer une instance inconnue, celle-ci est acheminée dans l'arborescence en fonction des valeurs des attributs testés dans les nœuds successifs. Lorsqu'une feuille est atteinte, l'instance est classée en fonction de la classe attribuée à la feuille.

En général, les arbres de décision sont représentés par une disjonction de conjonctions de contraintes sur les valeurs d'attributs des instances. Chaque chemin de la racine de l'arbre à une feuille correspond à une conjonction de tests d'attributs et l'arbre lui-même correspond à une disjonction de ces conjonctions.

La construction d'un arbre de décision se fait d'une manière récursive. Tout d'abord, un attribut est sélectionné comme nœud racine. Ensuite, une branche pour chaque valeur possible de cet attribut est créée. Cela divise les exemples en sous-ensembles, un pour chaque valeur de l'attribut. Le processus est répété de manière récursive pour chaque branche, en utilisant uniquement les instances qui atteignent réellement la branche. Si, toutes les instances d'un nœud ont la même classification, il faut arrêter de développer cette partie de l'arborescence.

Le choix d'un attribut est basé sur le gain d'information, l'attribut qui apporte le plus d'informations est sélectionné. Pour mesurer le gain d'information, une mesure couramment utilisée dans la théorie de l'information, appelée entropie est utilisée. Si un attribut traité peut prendre *c* différentes valeurs, alors l'entropie de S (l'ensemble des exemples d'apprentissage) est définie comme suit:

$$entropie\ (\mathrm{S}) = \sum_{i=1}^{c} - p_i \log_2 p_i \qquad (\mathrm{A.9})$$

où p_i est la proportion de S appartenant à la classe *i*.

A.4 Machine à vecteurs de support (SVM)

La *Machine à vecteurs de support* (*SVM*, de l'anglais *Support Vector Machine*) est un algorithme d'apprentissage qui permet de construire un classificateur linéaire. Dans SVM, l'ensemble des descriptions $\{\mathbf{x}_i\}$ est considéré comme un ensemble de points dans un espace vectoriel de dimension *d* dans lequel chaque observation $\mathbf{x}_i$ est un point dans cet espace. Si les points sont linéairement séparables dans l'espace de caractéristiques, il est toujours possible de construire un hyperplan H qui sépare les exemples des deux catégories en permettant d'assigner l'étiquette 1 ou -1 selon que

le point observé se trouve d'un côté ou l'autre de l'hyperplan. SVM construit l'hyperplan de marge maximale en utilisant les exemples d'apprentissage qui sont les plus difficiles à classifier (appelés les vecteurs de support), et qui seront à la limite de la marge de séparation. L'hyperplan de séparation est défini comme suit:

$$\mathbf{w}\mathbf{x} + \mathrm{b} = 0 \tag{A.10}$$

où $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ est un exemple dans l'espace de caractéristiques de dimension d, $\mathbf{w} = (w_1, w_2, \ldots, w_d)$ est un vecteur de poids et b est un scalaire, aussi appelée le biais.

SVM nécessite la solution du problème d'optimisation suivant:

$$\text{Minimiser}\ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \tag{A.11}$$

où les ξ_i sont des variables ressort qui mésusent l'erreur de classification et C est une constante, appelée le terme de régularisation, qui permet de donner plus ou moins d'importance aux points qui violent la marge de séparation.

Une fois que les paramètres optimaux **w** et b sont trouvés, Le classificateur appris est une fonction f de la forme:

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{i=1}^{d} w_i x_i + b\right) \tag{A.12}$$

Si les données ne sont pas linéairement séparables, SVM mappe les coordonnées sur un espace de dimension supérieure à l'aide d'une fonction de noyau K (ou Kernel) et recherche un hyperplan dans cet espace. Les quatre noyaux de base sont :

Linéaire : $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$

Polynômiale : $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma\mathbf{x}_i^T\mathbf{x}_j + r)^d$

Fonction de base radiale gaussienne(RBF) : $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$

Sigmoide: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\gamma \mathbf{x}_i^T \mathbf{x}_j + r\right)$

r, *d* et γ sont les paramètres du noyau (avec $\gamma > 0$).

A.5 K-means

Les techniques de clustering s'appliquent lorsqu'il n'y a pas de classe à prédire mais que les instances doivent être divisées en groupes. k-means est un algorithme de clusturing qui prend comme paramètre *k* le nombre de clusters recherchés. Ensuite, *k* points sont choisis au hasard en tant que centres de ces clusters. Chaque instance est affectée au centre du cluster le plus proche selon une mesure de similarité telle que la distance euclidienne. Etant données deux descriptions $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ et $\mathbf{y} = (y_1, y_2, \ldots, y_d)$, la distance euclidienne est calculée comme suit :

$$\text{distance euclidienne}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (y_i - x_i)^2} \qquad \text{(A.13)}$$

Ensuite, le centroid ou la moyenne de chaque cluster est calculé. Ces centroids sont considérés comme de nouvelles valeurs centrales pour leurs clusters respectifs. Enfin, l'ensemble du processus est répété avec les nouveaux centroids. L'itération se poursuit jusqu'à aucun changement de centroid ne soit possible.

A.6 Apprentissage d'ensemble

La méthode d'apprentissage d'ensemble (Dietterich, 2000) consiste à apprendre un ensemble de modèles de classification et combiner leurs décisions pour obtenir de meilleures performances que n'importe lequel de ces classificateurs individuels. Les modèles de classification inclus dans un ensemble sont couramment appelés classificateurs de base (BC). Les méthodes d'apprentissage d'ensemble les plus connus sont : le *bagging*, le *boosting* et le *stacking*. Ces techniques sont

généralement appliquées à des tâches de classification et à des problèmes de prédiction numérique.

Le *bagging* (Breiman, 1996) consiste à combiner les décisions de différents modèles du même type (e.g. arbres de décision) en une seule prédiction. La manière la plus simple de procéder dans le cas de la classification est d'utiliser le vote majoritaire. En premier lieu, plusieurs modèles de classification sont appris en utilisant des échantillons différents de données d'apprentissage choisis d'une façon aléatoire. Ensuite, la classe d'une instance est prédite par les différents modèles de classification appris. Enfin, l'instance est attribuée à la classe qui est prédite par la majorité. Une autre alternative à l'échantillonnage est d'utiliser des sous ensembles de données d'apprentissage avec différents *caractéristiques* (*attributs*) comme dans le cas de Random Forest (Breiman, 2001). Avec Random Forest, chaque BC utilise un sous-ensemble différent de *caractéristiques* (*attributs*).

le *boosting* (Freund, Schapire & Abe, 1999) consiste à combiner les décisions de différents modèles du même type (e.g. arbres de décision) en une seule prédiction en utilisant le vote majoritaire comme le *bagging*. Cependant, le *boosting* est itératif. Alors que dans le *bagging*, les modèles de classification individuels sont construits séparément, dans le *boosting* chaque nouveau modèle de classification est influencé par les performances de ceux construits précédemment. Le *boosting* commence par attribuer un poids égal à toutes les instances dans les données d'apprentissage. Il appelle ensuite l'algorithme d'apprentissage pour apprendre un classificateur pour ces données et réattribuer un nouveau poids à chaque instance en fonction du résultat du classificateur. Le poids des instances correctement classées (faciles) est diminué et celui des instances mal classées (difficiles) est augmenté. Lors de chaque itération, un classificateur est créé pour les nouvelles données d'apprentissage, ce qui permet par conséquent de classer correctement les instances difficiles. Ensuite, les poids des instances sont augmentés ou diminués en fonction du résultat de ce nouveau classificateur.

Le *stacking* est utilisé pour combiner des modèles de classification construits par différents algorithmes d'apprentissage. Il introduit le concept du méta-apprenant (un algorithme d'apprentissage), qui remplace la procédure du vote majoritaire. Une instance est d'abord introduite dans les modèles de niveau 0 (BCs), et chacun prédit une classe pour l'instance. Ces prédictions sont introduites au modèle de niveau 1 (méta-apprenant), qui les combine pour obtenir la prédiction finale.