Thèse de doctorat de l'Université Paris 13
dans le cadre d'une Cifre avec l'entreprise Talend

Spécialité Informatique

Par :

Aïcha BEN SALEM

Thèse présentée pour l'obtention du diplôme de Docteur
de l'Université Paris 13 Sorbonne Paris Cité

Qualité contextuelle des données : Détection et nettoyage guidés par la sémantique des données

Soutenue le .. mars 2015 devant le jury composé de :

Rapporteurs :		
Mme	Anne LAURENT	Professeur à l'Université Montpellier 2
Mme	Henda HAJJAMI BEN GHEZALA	Professeur à l'Université de la Manouba, Tunis
M.	Mohamed QUAFAFOU	Professeur à l'Université d'Aix-Marseille
Examinateurs :		
M.	Younès BENNANI	Professeur à l'Université Paris 13 (Président du jury)
M.	Faouzi BOUFARES	Maître de Conférences (HDR) à l'Université Paris 13 (Directeur de thèse)
M.	Sebastiao CORREIA	Chef de projet - Qualité De données- Entreprise Talend (Encadreur de thèse)

Thèse numéro :

Résumé

De nos jours, les applications complexes telles que l'extraction de connaissances, la fouille de données, le E-learning ou les applications web utilisent des données hétérogènes et distribuées. Dans ce contexte, la qualité de toute décision dépend de la qualité des données utilisées. En effet, avec l'absence de données riches, précises et fiables, une organisation peut prendre potentiellement de mauvaises décisions.

L'objectif de cette thèse consiste à assister l'utilisateur dans sa démarche qualité. Il s'agit de mieux extraire, mélanger, interpréter et réutiliser les données. Pour cela, il faut rattacher aux données leurs sens sémantiques, leurs types, leurs contraintes et leurs commentaires.

La première partie s'intéresse à la reconnaissance sémantique du schéma d'une source de données. Elle permet d'extraire la sémantique des données à partir de toutes les informations disponibles, incluant les données et les métadonnées. Elle consiste, d'une part, à classifier les données en leur attribuant une catégorie et éventuellement une sous-catégorie, et d'autre part, à établir des relations inter colonnes et de découvrir éventuellement la sémantique de la source de données manipulée. Ces liens inter colonnes une fois détectés offrent une meilleure compréhension de la source ainsi que des alternatives de correction des données. En effet, cette approche permet de détecter de manière automatique un grand nombre d'anomalies syntaxiques et sémantiques.

La deuxième partie consiste à nettoyer les données en utilisant les rapports d'anomalies fournis par la première partie. Elle permet une correction intra colonne (homogénéisation des données), inter colonnes (dépendances sémantique) et inter lignes (élimination des doublons et similaire). Tout au long de ce processus, des recommandations ainsi que des analyses sont proposées à l'utilisateur.

Mots-clés : Qualité des données, Sémantique des données, Reconnaissance de schéma, Profilage sémantique des données, Rapprochement de schéma, Homogénéisation, Déduplication, Doublons, Similaires.

Abstract

Nowadays, complex applications such as knowledge extraction, data mining, e-learning or web applications use heterogeneous and distributed data. The quality of any decision depends on the quality of the used data. The absence of rich, accurate and reliable data can potentially lead an organization to make bad decisions.

The subject covered in this thesis aims at assisting the user in its quality approach. The goal is to better extract, mix, interpret and reuse data. For this, the data must be related to its semantic meaning, data types, constraints and comments.

The first part deals with the semantic schema recognition of a data source. This enables the extraction of data semantics from all the available information, inculding the data and the metadata. Firstly, it consists of categorizing the data by assigning it to a category and possibly a sub-category, and secondly, of establishing relations between columns and possibly discovering the semantics of the manipulated data source. These links detected between columns offer a better understanding of the source and the alternatives for correcting data. This approach allows automatic detection of a large number of syntactic and semantic anomalies.

The second part is the data cleansing using the reports on anomalies returned by the first part. It allows corrections to be made within a column itself (data homogenization), between columns (semantic dependencies), and between lines (eliminating duplicates and similar data). Throughout all this process, recommendations and analyses are provided to the user.

Keywords : Data Quality, Data Semantics, Schema Recognition, Semantic Data Profiling, Schema Matching, Homogenization, Deduplication, Similar Data, Duplicates.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

1.1 Introduction

Le phénomène du BigData est généralement défini selon quatre dimensions (Vs) : Volume, Vélocité, Variété et Véracité. Le *volume* désigne la gestion de gros volumes de données. La *vélocité* (vitesse) est le temps nécessaire pour collecter et traiter les données. La *variété* traite des données structurées, semi-structurées et non structurées. Enfin, la ***véracité*** qui permet de garantir la qualité et la fiabilité des données. Dans cette thèse, nous nous intéressons à cette dernière V.

De nos jours, les données représentent une richesse pour les entreprises et les administrations et contribuent à leur développement. La qualité de ces données représente un enjeu important. Le coût de la non-qualité peut en effet s'avérer très élevé : prendre une décision à partir de mauvaises informations peut nuire à l'organisation, à ses clients ou ses partenaires. La gouvernance des données est un sujet qui prend de l'importance dans les entreprises et les administrations. Elle permet l'amélioration des interactions entre les différents collaborateurs d'une ou plusieurs

organisations concernées. De plus en plus d'entreprises tentent de capitaliser sur leurs données métier les plus importantes en construisant des référentiels de type MDM (Master Data Management) offrant une vue centrale et unique de ces dernières (figure 1.1). La qualité des données est un prérequis essentiel pour ce type de projets, plus encore que pour les projets BI (Business Intelligence).

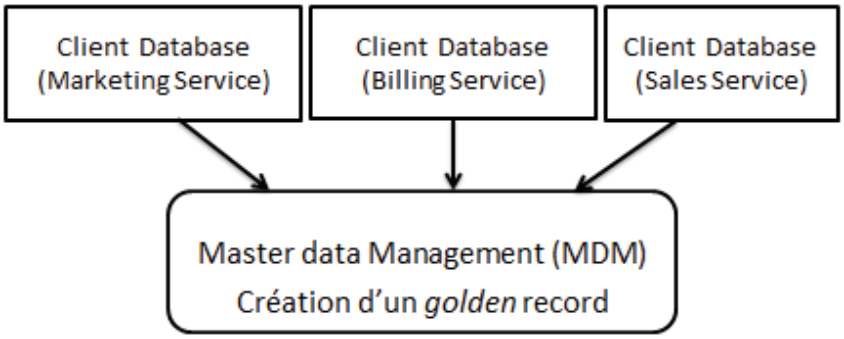


FIGURE 1.1 – Création du Master Data Management

De nombreuses méthodes pour identifier, mesurer et résoudre certains problèmes de qualité des données existent. Les outils proposés ne répondent pas encore à tous les problèmes soulevés. Ils se focalisent le plus souvent uniquement sur la donnée brute et non sur la signification de celle-ci. Or la donnée, pour être utile, doit être interprétée dans son contexte d'utilisation. Actuellement, peu d'entreprises ont mis en place un programme de gestion de la qualité des données tant au niveau des bases de données (BD) gérées qu'au niveau des entrepôts (ED) construits à partir de ces dernières. Les projets d'intégration n'appuient probablement pas assez sur l'importance de la qualité des données.

Alors que les ETL (Extract [1], Transform [2], Load [3]) ont atteint, de nos jours, un grand degré de maturité et offrent de très nombreux composants, les résultats d'intégration des données contiennent beaucoup trop d'anomalies et n'inspirent pas confiance afin d'aider à la prise de décision.

1.2 Contexte

Cette thèse se déroule entre le laboratoire d'Informatique de Paris Nord (LIPN) de l'université Paris 13, et l'entreprise Talend, un des leaders dans le marché des ETL Open Source.

1. Extraire des données depuis les sources
2. Transformer éventuellement les données pour diverses raisons
3. Charger les données dans l'entrepôt

Entreprise Talend

Talend est une jeune société, fondée en 2006. Elle contient 400 employés dans sept pays et deux sièges Los Altos, en Californie et Paris, en France.

Talend permet aux entreprises de déverrouiller toutes leurs données, qu'elles soient historiques, temps réel ou émergentes. Via le support natif des plateformes modernes Big Data, la solution sans empreinte de Talend simplifie l'intégration et fournit aux équipes informatiques les outils pour répondre plus rapidement aux demandes du marché, à un coût prévisible.

Elle propose des solutions (open source et entreprise) d'intégration évolutives pour le Big Data, l'intégration de données et d'applications, la qualité des données, le MDM et BPM (Business Process Management) (figure 1.2). Elle est classée Leader Visionnaire par Gartner et Forrester sur le marché de l'intégration.

FIGURE 1.2 – Talend Plateform

Talend permet aux équipes informatiques de fournir des données selon les besoins métiers. Elle fournit une solution open source supportée par une large communauté et des services de niveau entreprise (figure 1.3).

FIGURE 1.3 – Clients de Talend

Talend offre une solution complète de qualité des données (Talend Data Quality), incluant des fonctionnalités de profilage, de nettoyage, de mise en correspondance et de "monitoring" pour répondre à tous besoins de qualité et de gouvernance de données. Les fonctionnalités de qualité des données peuvent évoluer afin de gérer toutes les données, du fichier plat aux données d'entreprise dans Hadoop. Talend permet de tirer parti des meilleures fonctionnalités de la plateforme pour fournir une qualité des données continue à travers différents types de données et quel que soit le volume de données.

Les produits d'intégration de données de Talend (Talend Data Integration) permettent d'accéder, de transformer et d'intégrer des données de tout système en temps réel ou par lots afin de répondre aux besoins d'intégration de données opérationnelles et analytiques. Avec plus de 800 composants, Talend intègre presque toutes les sources possibles de données.

Les divers scénarios d'utilisation gérés comprennent l'intégration de masse (Big-Data/NoSQL), l'ETL pour le décisionnel, le MDM le data warehousing, la synchronisation, la migration, le partage, la qualité et les services de données.

Cependant, ces différents outils manquent de sémantique. Notre travail sera d'enrichir ces outils avec les aspects sémantiques afin d'aider les utilisateurs dans leurs démarches telle que l'intégration des données et la gestion de la qualité des données.

1.3 Qualité des données

Notre principal objectif dans cette thèse est de garantir une meilleure qualité dans les sources de données (SD). Présentons dans ce qui suit la notion de qualité des données, les coûts de la non qualité ainsi que les dimensions.

1.3.1 Définition

La **qualité des données** est un terme générique décrivant à la fois les caractéristiques de données : complètes, fiables, pertinentes et à jour, 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. Les données sont dites de qualité si elles satisfont aux exigences de leurs utilisateurs. En d'autres termes, la qualité des données dépend autant de leur utilisation que de leur état. Pour satisfaire à l'utilisation prévue, les données doivent être exactes, opportunes et pertinentes, complètes, compréhensibles et dignes de confiance (Toulemonde, 2008).

1.3.2 Coût de la non qualité

L'impact et donc le coût d'une donnée de mauvaise qualité n'est pas le même selon le type de population (dans un CRM (Customer Relationship Management), grand compte ou PME (petites et moyennes entreprises)) mais aussi selon l'utilisation qui en y faite (données bancaires, données médicales, données militaires sensibles ou données CRM). L'estimation des "coûts de la non-qualité" n'est pas aisée. Ajoutons que s'il est relativement aisé d'évaluer combien coûte la mise en oeuvre d'une procédure d'amélioration, les bénéfices escomptés sont plus difficiles à chiffrer en raison des aspects non mesurables, mais néanmoins cruciaux, qui accompagnent l'amélioration de la qualité d'un système informatique, tels que la crédibilité ou la fiabilité de l'information.

A titre indicatif, plusieurs études menées aux États-Unis dans des secteurs divers tels que banques, assurances ou agences de voyage font état d'un taux d'erreur de 5 % à 30 % dans les BDs (ce taux étant, par exemple, évalué sur la base du rapport entre le nombre d'enregistrements contenant au moins une erreur logique et le nombre total d'enregistrements d'une BD). En termes financiers, les coûts de la "non-qualité" sont évalués à une perte d'environ 5 à 10 % du revenu des entreprises examinées. Citons par exemple les coûts en contrôles, correction et maintenance de données de qualité douteuse, les coûts liés au traitement des plaintes des clients non satisfaits ou encore à la réparation des préjudices (Boydens, 1998).

Une étude aux États-Unis, a estimé le coût de la mauvaise qualité des données à plus de 600 milliards de dollars pour les entreprises chaque année (Toulemonde, 2008).

1.3.3 Dimensions de la qualité des données

Différents travaux dans la littérature (Berti-Équille, 2006, Toulemonde, 2008, Wang and Strong, 1996), ont défini un ensemble de dimensions afin de mesurer la qualité des données. Parmi ces dimensions, nous avons détaillé celles que nous allons utiliser dans notre processus afin de garantir aux utilisateurs des données ainsi que des décisions pertinentes.

— Interprétabilité : une donnée doit être représentée sous un format cohérent et sans ambiguïté.
— Standards : les valeurs sont correctes par rapport à un intervalle de réparation ou à un domaine. Par manque de standards de codification, l'organisation "Hôpital Farhat Hached" peut apparaître comme "CHU Farhat Hached", "C.H.U Farhat Hached" ou "CHU FH".
— Intégrité : toutes les données nécessaires sont disponibles pour le besoin métier. Il est impossible d'effectuer une campagne d'e-mailing avec une BD clients ne contenant pas l'adresse mail.
— 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.

Les données doivent avoir la qualité nécessaire pour supporter le type d'utilisation. En d'autres termes, la demande de qualité est aussi importante sur les données nécessaires à l'évaluation d'un risque que sur celles utilisées dans une opération de marketing de masse.

1.3.4 Indicateurs et mesures de la qualité des données

Il faut préciser aussi que chaque organisme doit créer ses propres définitions opérationnelles en fonction des objectifs et priorités, afin de définir des indicateurs pour chacune des dimensions, et vérifier par des mesures régulières leur évolution dans le temps. Pour cela, nous avons défini des indicateurs permettant de mesurer ces dimensions (table 1.1).

TABLE 1.1 – Indicateurs de qualité

Dimensions	Caractéristiques	Indicateurs
Interprétabilité	Les données sont-elles compréhensibles par les utilisateurs ?	Nature/type/langue des données.
Intégrité	Les données sont-elles toutes disponibles ?	Indicateur de valeurs nulles.
Standardisation	Les données sont-elles écrites dans un format standard ?	Indicateur table de fréquence. Indicateur motif de fréquence des données
Duplication	Les données sont-elles répétées ?	Indicateur de valeurs distinctes. Indicateurs de valeurs en doubles

Chaque dimension peut être mesurée soit d'une manière subjective en recueillant la perception des utilisateurs, soit de manière objective au travers de suivis automatiques des indicateurs spécifiques. Dans notre cas, nous allons utiliser la deuxième méthode.

1.4 Problématique

Notre problématique est "Comment rétablir la confiance des décideurs dans les données" manipulées et rassemblées dans une base ou un entrepôt de données. Celles-ci sont volumineuses, totalement hétérogènes, distribuées et de différents niveaux de qualité. Plusieurs questions se posent à propos, d'une part de la validité ou de l'invalidité des données et d'autre part, à propos de ce qu'il y a à corriger et comment le faire ?

En effet, rassembler les données, afin d'en faire par exemple un entrepôt, est une tâche primordiale dans le processus de prise de décision. La prise en compte du concept QdD est notre objectif principal dans le processus d'intégration de ces dernières. Cependant, il faut remarquer que la qualité de connaissances découle de la QdD. Pour répondre à ce besoin de maîtrise de l'information (une donnée qui a un sens) et de sa qualité, des outils de reconstruction de la sémantique à partir des données et de leurs métadonnées sont nécessaires. Ainsi nous contribuons à la création de nouveaux outils ETL sémantiquement riches.

Comprendre les données est une tâche importante dans le processus d'intégration de ces dernières. L'objectif est de mieux extraire, mélanger, interpréter et réutiliser

ces dernières. Pour cela, il faudrait rattacher aux données leurs structures, leurs types et leurs faits implicites existants sur le web. Les données vont être utilisées dans différentes applications et dans des contextes variés (figure 1.1). D'où les problèmes de qualité vont augmenter sur le web et le besoin d'une spécification sémantique sera une urgence.

Signalons cependant que les transformations, même assistées par des composants très ergonomiques dans les différents ETL, sont manuelles et à la charge de l'utilisateur. Ce dernier ne dispose d'aucune aide sémantique qui le guiderait dans sa démarche pour soit (i) empêcher l'intégrer soit (ii) définir par exemple le domaine cible, éventuellement la transformation adéquate et la correction de certaines anomalies.

Les exemples ci-dessus, permettent de mettre l'accent sur la grande variété des types d'anomalies qui peuvent exister dans les données. Celles-ci peuvent provenir des sources ou sont le résultat du processus d'intégration. Plusieurs classements des anomalies ont été faits dans la littérature (Oliveira et al., 2005), (Berti-Équille, 2012). On peut cependant, signaler la pauvreté de l'approche sémantique. On peut classer les problèmes relatifs à la QdD en deux niveaux :

— Les descriptions des données : (i) des conflits entre les noms des objets manipulés tels que les attributs peuvent exister (Homonymes, Synonymes) ; (ii) des imprécisions sur la définition exacte des objets (leurs domaines) peuvent engendrer de nouvelles anomalies. Le manque de la sémantique pourrait en être la cause.

— Les données elles-mêmes : plusieurs catégories d'anomalies ont été recensées dans la littérature. Citons par exemple, les doublons, les données similaires, les valeurs aberrantes, les valeurs nulles et les données obsolètes (Berti-Équille, 2012).

Différentes approches ont été proposées pour remédier à ces problèmes, telles que les quatre approches proposées dans (Berti-Équille, 2007) : (i) l'approche corrective qui permet le nettoyage des données en utilisant des outils d'extraction et de transformation des données tels que les ETL ; (ii) l'approche adaptative pour adapter des traitements, des vérifications, des modifications sur les données lors d'intégration de données ; (iii) l'approche préventive qui évalue la qualité des modèles, la qualité des processus employés pour le traitement des données et (iv) l'approche diagnostique qui permet la détection des problèmes dans les données en appliquant des analyses et des méthodes statistiques.

1.5 Objectifs

Afin de garantir une meilleure gestion de la qualité de leurs données, les entreprises doivent mettre en place un MDM intelligent. Celui-ci devrait contenir des structures riches en sémantique telles que les ontologies.

En résumé, la qualité des connaissances découle de la QdD, mais pas seulement. Notre objectif dans cette thèse est dans un premier temps d'étudier la qualité de l'information à partir de toutes les informations disponibles sur les données (données, métadonnées, résultats d'analyses et informations fournies par l'utilisateur). Pour ce faire, il faudra modéliser la sémantique des données et le contexte de leur utilisation. Dans un deuxième temps, corriger et nettoyer les données.

Nous devrons élaborer des procédures permettant d'extraire et de mettre à jour ces méta-informations sur les données. Puis, il s'agira d'exploiter la sémantique des données reconstruite dans leur contexte pour aider à la correction des problèmes de qualité lors de processus d'intégration des données hétérogènes. Notamment, des règles de qualité pourront être appliquées pour l'homogénéisation des données ou la fusion des données dans le cas de traitement des doublons. D'autres règles permettront de corriger les données en fonction du contexte d'application.

L'enrichissent des données par des informations sémantiques permet un meilleur nettoyage de ces dernières. Le nettoyage des données se fait en deux étapes : la première étape est la détection des anomalies et la deuxième est la correction de ces dernières. Quand la sémantique de ces données existe et bien présentée, une meilleure et plus simple détection est réalisée et de même pour le nettoyage (figure 1.4).

FIGURE 1.4 – Les objectifs de l'outil sémantique DQM

Construire et développer des ETL capables de gérer la qualité de nos données et contenant de la sémantique afin de garantir une meilleure qualité.

Afin de mettre en oeuvre notre démarche qualité, nous définissons les principes d'un outil de gestion de la qualité des données dans les sources de données, appelé DQM (Data Quality Management) (approche sémantique). Cet outil permet à la fois la reconnaissance des métadonnées, le nettoyage et l'enrichissement des données (figure 1.5).

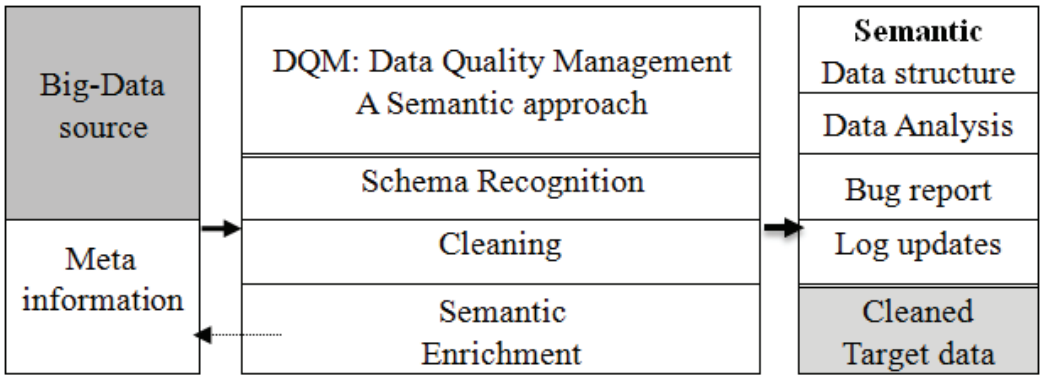


FIGURE 1.5 – L'outil DQM (Approhce sémantique)

Les principales parties de l'outil DQM, présentées dans la figure (1.5), contiennent chacune un ensemble de tâches :

- Reconnaissance sémantique du schéma des données (Schema Recognition)
  - Profilage sémantique des données pour une analyse colonne par colonne.
  - Alignement sémantique du schéma afin de détecter les relations entre colonnes.
- Recommandation sémantique des analyses
- Enrichissement sémantique (Semantic Enrichment)
- Nettoyage de données à base de sémantique (Cleaning)
  - Homogénéisation des données
  - Déduplication des données

Nous traitons dans cette thèse les problèmes qu'une source de données peut contenir. Ces problèmes de QdD présentent un grand enjeu aux utilisateurs lors des opérations de profilage de données, de prise de décision ou n'importe quelle autre opération.

1.6 Plan du document

Ce document est composé d'une introduction générale, un état d'art, de deux grandes parties contenant chacune trois chapitres et d'une conclusion générale.

Le chapitre 2 présente un état de l'art sur la reconnaissance sémantique des schémas ainsi que sur les travaux de correction et nettoyage de données. Dans ce chapitre, nous commençons par présenter les différentes anomalies pouvant exister dans une source de données (le schéma ainsi que les données elles mêmes), les causes de ces anomalies. Nous avons présenté après, les travaux existants dans la littérature proposant des solutions pour ces anomalies en particulier : les travaux sur la recon-

naissance sémantique des schémas de données et ensuite les travaux de correction de données tels que l'élimination des doublons et similaires.

Une première grande partie concerna la reconnaissance sémantique du schéma d'une source de données. Cette reconnaissance consiste en deux étapes. La première étape (chapitre 3) permet la reconnaissance des attributs du schéma (profilage des données) en exploitant la sémantique existante dans les données. La deuxième étape (chapitre 4) permet de reconnaître les relations inter colonnes ainsi que le concept décrit par ces attributs. Un dernier chapitre (chapitre 5) permet l'expérimentation de cette première approche sur un exemple concret en présentant les différents outils utilisés.

La deuxième grande partie est le nettoyage des données. Une fois les données sont reconnues sémantiquement, une information supplémentaire facilitent la détection et la correction des données. Le nettoyage de données consiste aussi en deux étapes : Homogénéisation et élimination des doublons et similaires.

Le chapitre 6 présente la partie homogénéisation des données. Les rapports d'anomalies renvoyés par le profilage de données (chapitre 3) détectent la présence de plus d'un format, code, unité de mesure ou langue dans une même colonne. On propose alors une homogénéisation et unification des données de la source.

Le chapitre 7 présente la deuxième grande étape de nettoyage qui est la déduplication des données. Nous présentons trois algorithmes séquentiels et un algorithme parallèle permettant la détection et la correction des données redondantes. Le chapitre 8 présente l'expérimentation de la deuxième partie.

Le dernier chapitre (chapitre 8) résume les différents apports intégrés aux outils de Talend. D'une part, la reconnaissance sémantique de schéma est appelée lors de plusieurs opérations telles que la jointure, l'union ou la conversion d'une source d'un type vers un autre (d'un fichier csv vers une table d'un SGBD). D'autre part, les différentes actions de homogénéisation sont proposées à l'utilisateur après chaque profilage des données et les algorithmes d'élimination des doublons et similaires sont utilisés dans les outils Talend (TDQ, MDM).

Chapitre 2

Etat de l'art

Sommaire

2.1 Introduction

Les préoccupations stratégiques pour de nombreuses entreprises tournent de nos jour autour des données (bases et entrepôts de données, BigData). Ces dernières sont devenues incontournables pour une meilleure gestion, pour aider au développement d'outils d'aide à la décision et de prédiction. Les données sont très volumineuses, hétérogènes, distribuées et de qualité variable. Plusieurs types d'anomalies (notamment des doublons, des données similaires, des données aberrantes, des données obsolètes et des valeurs nulles) ont été relevés dans les données rassemblées. En conséquence, la qualité aussi bien de la gestion que de l'extraction des connaissances à partir de ces données peuvent s'avérer mauvaise. De surcroît, les coûts financiers qu'engendre la non-qualité des données, ou leur qualité médiocre, sur les prises de décision risquent d'être considérables (Toulemonde, 2008).

On peut facilement constater que les outils de manipulations de données tels que les Systèmes de Gestion de Bases de Données (SGBD) et les outils d'intégration (ETL) d'aujourd'hui sont très manuels. A la charge de utilisateur de définir le

modèle de sortie (la structure détaillée des données résultats). Il devra ainsi être un expert ou une personne du domaine afin d'une part, de comprendre la définition des données sources, et d'autre part, de décider des différentes conversions et transformations à réaliser sur ces dernières. L'utilisateur ne bénéfice d'aucune aide **sémantique** pour effectuer cette tâche primordiale et ne bénéfice pas non plus de recommandations. Non seulement, les données sources hétérogènes sont généralement mal décrites, mais aussi elles peuvent être imparfaites et polluées par la présence de plusieurs types d'anomalies. Ceci complique énormément toute tâche de manipulation de données à savoir l'intégration des données, les calculs sur celles-ci et l'extraction des connaissances. Cette dernière, mal faite, pourrait contribuer à injecter dans le résultat de nouvelles anomalies.

L'opération de rassemblement des données afin de construire de nouvelles bases et entrepôts de données (Hamdoun and Boufarès, 2010), (Badri et al., 2009) ainsi que des outils d'aides à la décision issus de ces très grosses masses de données hétérogènes et distribuées nécessite le développement de nouveaux ETL (**?**). Ces derniers doivent, d'une part, prendre en compte l'hétérogénéité des données et leurs contraintes, et d'autre part, assurer la qualité du nouvel ensemble construit tout en redonnant un *sens* et une *crédibilité aux données rassemblées*.

Dans ce chapitre, nous commencerons par donner une liste non exhaustive des anomalies dans les métadonnées et dans les données elles-même. Dans un deuxième temps on abordera les solutions présentées dans la littérature afin de remédier à ces anomalies. Nous mettrons l'accent sur la nécessité d'une approche sémantique, très peu abordée jusque là, ce qui prouve de l'originalité de notre contribution.

Nous utilisons dans la suite les deux mots "colonne" et "attribut" pour désigner la même chose.

Avant d'entamer la partie état de l'art, annonçons quelques notions qui seront utilisées tout au long du manuscrit.

Définition 2.1 : Types de données

Dans un système de types, les types sont construits à partir des types de base (prédéfinis) BASE et de constructeurs de types CONST TYPES* (Khalfallah, 2006). Un constructeur de type prend une séquence de types et construit un nouveau type.

*TYPE : := BASE | CONST TYPES** ■

Exemple 2.1 : Types de données

BASE : := string | char | date| integer | number | real | boolean | CLOB | BLOB | ...

CONST : := set | list | tuple | ... ■

Propriété de confluence de type

Soient dt_1, dt_2, dt_3 et dt_4 des types de données. La notation $dt_1 \rightarrow dt_2$ pour dire que x : dt_1 (x est de type dt_1) peut être vue comme x : dt_2 (x est de type dt_2).

si $dt_1 \rightarrow^* dt_3$ et $dt_2 \rightarrow^* dt_3$ alors il existe dt_4 tels que $dt_2 \rightarrow^* dt_4$ et $dt_3 \rightarrow^* dt_4$, avec $\rightarrow^*$ est la fermeture transitive de $\rightarrow$ (Khalfallah, 2006). ■

Définition 2.2 : Compatibilité des types de données

Deux types dt_1 et dt_2 sont compatibles s'il existe un dt_3 tel que $dt_1 \rightarrow^* dt_3$ et $dt_2 \rightarrow^* dt_3$ (Khalfallah, 2006).

Dans un système satisfaisant la propriété de confluence, la compatibilité est une relation d'équivalence. ■

Définition 2.3 : Domaine de données

On note D le nom attribué à un domaine.

Soit [[D]] l'ensemble de valeurs concrètes d'un type de données tels que les chaînes de caractères alphanumériques (string, char), les numériques (integer, number, real), les dates, les bouléens (boolean) ou encore des listes et des intervalles de valeurs. ■

Les valeurs de type string peuvent être : Adamssss, Pariiiiis.

Exemple 2.1 : Définition d'un domaine de données

DomainName10 est le nom donné au type string de dix caractères. [[DomainName10]] est l'ensemble de chaînes de caractères de longueur dix. ■

Il faut remarquer que dans cet ensemble, certaines chaînes peuvent être non significatives et devraient être qualifiées sémantiquement incorrectes.

Par abus de langage, on considérera que le mot colonne signifie attribut et le nom de domaine n'est pas précisé.

Soit $C = \{C_1, C_2, ..., C_n\}$ l'ensemble des noms des colonnes d'une source de données.

Définition 2.4 : Une source de données

Une source S est interprétée par l'ensemble [[T]] de tous les $\{C_1 \ldots C_n\}$-tuples définis sur $\prod_{i=1}^{n}[[D_n]]$.

[[T]] est l'ensemble des fonctions avec $t : \{C_1 \ldots C_n\} \rightarrow \prod_{i=1}^{n}[[D_n]]$ tel que t(C_i) noté t.C_i est un élément de $[[D_i]]$.

Chaque élément t de [[T]] est un tuple (enregistrement, record, ligne) de la source S et chaque t.C_i est une valeur de la colonne dans t, que l'on notera aussi v. ■

Définition 2.5 : Schéma d'une source de données

Un schéma est une définition de la structure complète d'une source de données (Menard, 2008).

Un schéma décrit toutes les propriétés d'une source : les noms des colonnes (attributs), les domaines syntaxiques de chacune (types de donnée) et éventuellement des contraintes et des commentaires sur certaines d'entre elles.

Les contraintes (Constraint) définies sur les colonnes expriment des règles sémantiques. Elles peuvent être de plusieurs types : clé primaire (Primary key), unique (Unique), clé étrangère (Foreign key), appartenance à un intervalle ou à une liste (Check), vérification de certaines règles de gestion et des déclencheurs (Triggers).

Soit $Dom = \{D_1, D_2, ..., D_n\}$ l'ensemble des domaines des $C_i, i = 1; n$. D_i présente le domaine syntaxique (type de données).

Soit $K = \{K_1, K_2, ..., K_n\}$ l'ensemble des contraintes sur les $C_i, i = 1; n$. K présente le domaine sémantique d'une colonne.

Soit $Com = \{Com_1, Com_2, ..., Com_n\}$ l'ensemble des commentaires sur les $C_i, i = 1; n$.

On note un schéma
$S\{C_1 : \{D_1, K_1, Com_1\}, C_2 : \{D_2, K_2, Com_2\}, ..., C_n : \{D_n, K_n, Com_n\}\}$.

Par abus de langage on utilise la notation $S\{C_1, C_2, ..., C_n\}$ sachant que les domaines, les contraintes et les commentaires sont implicites, ou $S(C)$.

Une colonne C_i (un attribut) de S sera désigné par $S.C_i$. ■

Exemple 2.2 : Exemples de schémas

S1(FName, DateBirth, Contry). S1 est une source de données définie avec seulement les noms des attributs. C'est le format le plus souvent utilisé lors de la définition d'une source.

S2(FName, BirthDate : {date, ">01/01/1980"', //Date of birth}, Country). S2 est définie avec des noms de colonnes significatifs et une contrainte et un commentaire pour la colonne *BirthDate*.

S3(ABS1 : {string, clé primaire}, ABS2 : {string}, ABS3 :{integer}, ABS4 : {string}). S3 est une source avec des noms de colonnes ambiguës (meaningless). Cependant, le type de données est présent pour chaque colonne et une contrainte est définie sur le premier attribut. ■

Notons que deux catégories de sources de données existent. **Les sources qui sont accompagnées du schéma descriptif**. Celui-ci peut être facilement compréhensible ou encore prêter à confusion.

Exemple 2.3 : Exemple d'une source où les noms des colonnes ne veulent rien dire

TABLE 2.1 – Exemple de source S avec schéma

S				
	ABS1	ABS2	ABS3	ABS4
	string	string	integer	string
	clé primaire	? ?	? ?	? ?
	//	//	//	//
t1	LeBon Adam	0653545577	1000	
t2	LeBon A.	0653545555	2000	
t3	Lui Clémence	0607080911	1000	
t4	LeBon Adam	adam@yahoo.fr		Royaume-Uni
t5	LeBon A.	-		RoyaumeUni
t6	Lui Clémence	clui@gmail.com		France
t7	Traifor Eve	traifor@up13.fr		Chine

Le symbole " ? ?" est utilisé pour montrer l'absence de types de données ou de contraintes sur les colonnes. De même le symbole "//" montre l'absence de commentaires. ■

Les sources de données (sous forme d'un fichier plat (texte, csv)), peuvent ne pas être accompagnées d'un schéma descriptif (meaningless schema) (**?**).

Exemple 2.4 : Union des données sans schéma descriptif

S (table 2.2) est le résultat de l'intégration de deux sources de données (française et anglaise).

La source S peut être vue comme un ensemble de chaînes de caractères. Chaque tuple correspond à une ligne. Cette dernière contient plusieurs colonnes séparées par un point virgule.

TABLE 2.2 – Exemple de source S sans schéma

S1 : Format CSV origine France
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; Londres ; Royaume-Uni ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; A. ; 0653545555 ; - ; M ; M. ; London ; Royaume-Uni ; - ; 2000 ; 31/03/2012 ;0123435432
Londres ;- ; - ; - ; F ; Mme ; Londres ; Royaume-Uni ; - ; 1000 ; 1-3-2002 ; 0411223344
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; - ; Mme ; Epinay \ seine ; France ; - ; 02 mars 2014 ; 0120000022
Adamsss ; LeBon ; 0653545555 ; - ; - ; - ; - ; - ;- ; 31/3/2012 ;-
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; F ; - ; Epinay sur seine ; France ; - ;2/3/2014 ;0120000022
Saint ; R. ; 0708091122 ; www.saint.fr ; M ; M. ; Epinay Villetaneuse ; Frence ; - ;3000 ;-
Tunsi ; Rahma ; - ;- ; - ; Mme ; Epinay sur seine ; France ; - ;1000 ; 31-11-2014 ; 0965321876
Riche ; Emir ; - ;- ; - ; M. ; Qatar ;- ; - ;200000000 ; 11/2/1955 ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; F ; Mme ; Pékin ; Chine ; - ;1000 ; 30-2-2014 ; 0123987654
LeBon ; Adem ; - ; ada@obsolete.uk ; M ; Londre ; - ; - ;- ;-
S2 : Format CSV origine Angleterre
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; Europe ;1000 ; 31/03/2012 ; www.lebon.fr
LeBon ; A. ; 0653545555 ; - ; Male ; Mr ; London ; United-Kingdom ; Europe ;2000 ; 31/03/2012
London ; - ; 0766554425 ; - ; M ; Mr ; London ; United-Kingdom ; - ; - ; 11-12-1998
LeBon ; - ; 0653545577 ; adam@yahoo.fr ; 1 ; M. ; London ; UK ; - ;- ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; Female ; Mrs ; Beijing ; China ; Asia ; 1000 ; 02/29/2014 ; 0123987654
Lebon ; Adel ; 0653545599 ; alebon@up13.fr ; M ;- ; Paris ; France ; Europe ; - ;45
Paris ; H. ; 0607080911 ; paris@live.com ; 0 ; Mrs ; LA ; USA ; America ;10000 ; 23-10-1992 ; 0987654321
Correia ; Sebastiao ; - ; scorreia@talend.com ; M ; - ; Suresnes ; - ; - ;- ; 9/30/2007 ; -

2.2 Les différentes types des anomalies

Plusieurs anomalies ont été définies dans le monde professionnel et dans la littérature (Dong et al., 2009), (Oliveira et al., 2005), (Berti-Équille, 2006), (Bleiholder and Naumann, 2006), (Boufarès and Ben-Salem, 2012). Deux types d'anomalies sont définies : les anomalies dans les métadonnées (descriptifs des données) et celles dans les données. Force est de constater que les métadonnées sont absentes dans la majorités des sources.

2.2.1 Anomalies dans les métadonnées

Les anomalies dans les métadonnées sont diverses et sont causées par différents facteurs. On présente ci-dessous une liste non exhaustive de ces anomalies ainsi que leurs causes.

Rappelons que les métadonnées constituent tout ce qui est descriptif de données à savoir les noms des objets, leurs domaines de définition syntaxique (type de données : string, char, date, integer, number, real, boolean), leurs domaines de définition sémantiques (les contraintes), les commentaires et certaines analyses.

Les métadonnées sont souvent négligées, pourtant ces informations sont indispensables à une bonne compréhension du contenu sémantique des données et leurs contextes d'utilisation.

Parmi les anomalies constatées, on peut citer par exemple :

— Les noms des objets sont souvent non significatifs. Les noms des sources de données (table ou fichier) ne reflètent pas forcement leur contenu. Les noms utilisés dans le schéma descriptif, s'ils existent, de la source sont insignifiants tels que NP, Pnum ou Id. Ces derniers posent le problème de synonymie et d'homonymie.

— Les domaines de définition syntaxique des données ne sont pas suffisamment précis. Les types de données (string, char, date, integer, number, real, boolean) ne suffisent pas pour comprendre la sémantique des données. Par exemple, les noms des personnes (définis sur des chaînes de caractères) ne sont pas comparables sémantiquement à des chaînes de caractères qui représentent les noms des entreprises.

— Les contraintes ne sont pas définies ou sont désactivées. Elles sont désactivées souvent pour des problèmes de performance ou parce que temporairement on a besoin de ne pas respecter ces contraintes. Par exemple, en l'absence du type intervalle, des valeurs numériques peuvent être aberrantes.

— Les commentaires ne sont pas maintenus à jour ou ne sont pas renseignés à cause de la fainéantise des utilisateurs ou bien parce que l'outil n'impose pas de les mettre.

Certaines entreprises définissent les bonnes pratiques, cependant elles ne sont pas souvent respectées à cause de diverses raisons. D'une part, les contraintes système imposées par les outils limitent à 30 caractères la taille des noms des objets, des noms courts sont requis. D'autre part, les règles de nommage imposées par l'entreprise ne permettent pas des noms significatifs.

A l'aire du Big-Data, les sources de données n'ont pas forcément un descriptif (table 2.2). Ce qui complique d'avantage la compréhension du contenu et à fortiori toute manipulation telle que le processus d'intégration. Les anomalies dans les métadonnées, peuvent impacter donc toute opération sur les schémas des données et par conséquent les données elles-mêmes.

Il faudrait : (i) comprendre le sens de chaque colonne et de la source elle-même ; (ii) détecter les anomalies éventuelles au sein de chaque colonne et entre certaines d'entre elles ; (iii) et enfin corriger certaines de ces anomalies. L'exemple ci-dessous explicite ces problèmes.

Par exemple, lors d'une union, la compatibilité entre schémas de données ne peut être définie comme suit :

"*deux schémas sont compatibles si et seulement si ils ont le même nombre de colonnes et toutes les colonnes sont définies deux à deux sur le même domaine syntaxique*" (Codd, 1970).

Ainsi les opérations ensemblistes telles que l'union, l'intersection ou la différence devraient être basées sur une compatibilité plus riche sémantiquement. Les colonnes devraient être définies aussi sur le même domaine sémantique. Il en est de même des opérations binaires telles que la jointure et ses dérivés.

Force est de constater que les outils SGBD (Système de Gestion des Bases de Données) et ETL ne respectent pas cette compatibilité sémantique et n'assistent pas les utilisateurs.

Les exemples ci-dessous, traduisent certains conflits d'intégration de données dûs aux anomalies dans les schémas descriptifs (métadonnées).

Les définitions du nom de la donnée destination ainsi que de son type sont données manuellement. Les outils ETL ne font pas de recommandation et supposent donc une certaine expertise de l'utilisateur.

Exemple 2.4 : Intégration des données (Fusion-Union) (Contrainte non définie)

Etant donné deux sources de données S1 et S2. S1 regroupe la liste des clients. Elle est gérée par un SGBD. La source S2 représente la liste des patients. Elle est donnée sous forme d'un fichier plat (de format texte ou csv).

Les schémas des deux sources sont :

S1(NomP : {String(20)}, Tel : {String(10)}, CA : {Integer})

S2(NP : {String}, Tel : {String}, Pays : {String})

Le résultat de la "Fusion-Union" est : R1= Fusion-Union(S1, S2).

R1(NomP : {String}, Téléphone : {String}, CA : {Integer}, Pays : {String})

L'intégration de ces deux sources, peut engendrer différentes anomalies. La structure de la fusion est décrite par l'utilisateur. Le nom et le type de chaque colonne résultat sont renseignés. *R1.NomP* sera construit à partir de *Client.NomP* et de *Patient.NP*. Le type de cette colonne est celui qui doit couvrir *String(20)* et *String*. Il faut remarquer que les données peuvent être tronquées si le type résultat n'est pas adéquat.

Il en est de même pour la colonne *R1.Téléphone* (respectivement *R1.CA* et *R1.Pays*) qui résulte de *Client.Tel* et de *Patient.Tel* (respectivement *Client.CA* et *Patient.Pays*).

Des valeurs nulles seront générées à cause du fait que l'information sur les données *Pays* n'existe pas dans la première source et la donnée sur le chiffre d'affaires *CA* non plus. ■

TABLE 2.3 – Intégration (Fusion-Union) des deux sources Client et Patient avec un ETL

S1 : Client			
	NomP	Tel	CA
	String(20)	String(10)	Integer
	? ?	? ?	? ?
	//	//	//
t1	LeBon Adam	0653545577	1000
t2	LeBon A.	0653545555	2000
t3	Lui Clémence	0607080911	1000

S2 : Patient			
	NP	Tel	Pays
	String	String	String
	? ?	? ?	? ?
	//	//	//
t1'	LeBon Adam	adam@yahoo.fr	Royaume-Uni
t2'	LeBon A.	-	RoyaumeUni
t3'	Lui Clémence	clui@gmail.com	France
t4'	Traifor Eve	traifor@up13.fr	Chine

R1 : Clients (Résultat de Fusion-Union de Client et Patient				
	NomP	Téléphone	CA	Pays
	String	String	Integer	String
	? ?	? ?	? ?	? ?
	//	//	//	//
t1	LeBon Adam	0653545577	1000	
t2	LeBon A.	0653545555	2000	
t3	Lui Clémence	0607080911	1000	
t4	LeBon Adam	adam@yahoo.fr		Royaume-Uni
t5	LeBon A.	-		RoyaumeUni
t6	Lui Clémence	clui@gmail.com		France
t7	Traifor Eve	traifor@up13.fr		Chine

Il est clair que les choix d'intégration, basés sur la synonymie/homonymie des colonnes de noms *S1.Tel* et *S2.Tel*, introduit des anomalies dans le résultat. Des

adresses mails sont présentes dans la colonne nouvellement baptisée *R1.Téléphone*. L'utilisateur n'est pas assisté dans le processus d'intégration. Plusieurs questions se posent telles que :

— "Fallait-il comprendre les noms des colonnes *S1.Tel* et *S2.Tel* comme étant un téléphone ou un mail ?". D'une part, les valeurs n'étant pas affichées à l'écran, et d'autre part, aucune information n'est renseignée sur la sémantique des données. Le type *String*, à lui tout seul, ne permet pas d'assurer la cohérence sémantique des deux colonnes.

— "Quelle est la structure du résultat : s'agit-il d'une union ou d'une jointure et quel type de jointure (gauche, droite, externe, interne) ?".

Les outils devraient reconnaître la sémantique des deux colonnes avant la réalisation de l'opération.

La sémantique de la première colonne *S1.Tel* devrait être des *numéros de téléphone*, alors que celle de la deuxième colonne *S2.Tel* devrait être des *adresses mail*.

Les outils devraient alerter cette incompatibilité sémantique et interdire même l'opération, ce qui n'est pas toujours le cas de nos jours.

Exemple 2.5 : Intégration des données (Fusion-Jointure-Gauche)

La jointure de ces deux sources de données produit des résultats différents. Par exemple, une jointure à gauche (Codd, 1970) renvoie le résultat présenté dans la table 2.4. Cette jointure donne les personnes existantes dans les deux sources en enrichissant avec l'attribut *Pays*. Cependant, l'information sur le mail est perdue.

R2= Fusion-Left-Join(S1, S2, S1.Tel=S2.Tel).

R2(NomP : {String}, Tel : {String}, CA : {Integer}, Pays : {String})

TABLE 2.4 – Intégration (Fusion-Jointure-Gauche) des deux sources Client et Patient avec un ETL

R2 : Clients (Résultat de la Fusion-Jointure-Gauche de Client et Patient)				
	NomP	Tel	CA	Pays
	String	String	Integer	String
	? ?	? ?	? ?	? ?
	//	//	//	//
t1	LeBon Adam	0653545577	1000	Royaume-Uni
t2	LeBon A.	0653545555	2000	RoyaumeUni
t3	Lui Clémence	0607080911	1000	France

La jointure n'aurait pas dû être acceptée. L'utilisateur devrait être alerté lors du choix des colonnes de jointure. ■

Pour une meilleure intégration sémantique de données, la condition de jointure devrait porter sur des colonnes **sémantiquement équivalentes**.

Exemple 2.6 : Intégration des données (Fusion-Union) (Contrainte définie)

R est le résultat de l'intégration de deux sources de données (table 2.5).

S1 regroupe des données du mois de janvier sur la propreté et la température de la ville de Paris. S2 contient celles du mois de février. S1 est une version francophone et S2 est version anglo-saxonne.

S1(Note : {[0..20}, Température : {[-40..50], °C})

La colonne *S1.Note* représente l'ensemble des notes de satisfaction à propos de la propreté de la ville. Ces notes sont données sur une échelle de 0 à 20. Alors que *S1.Température* représente la température moyenne, en degré Celsius (°C), de la ville de Paris pour certains jours du mois de janvier.

La colonne *S2.Note* pourrait représenter la même information que *S1.Note*. Elle est mal définie (de 0 à 10 ou de 0 à 100). La valeur *100* est une donnée douteuse. *S2.Temp* indique certaines moyennes de la température du mois de février de la ville de Paris, elles sont probablement en Fahrenheit (°F) et l'intervalle de définition n'est pas précisée.

S2(Note, Temp).

TABLE 2.5 – Intégration des deux sources de données avec un ETL

S1		S2	
Note	Température	Note	Temp
? ?	? ?	? ?	? ?
[0..20]	[-40..50]	? ?	? ?
//	°C	//	//
12 ; -5 °C 10 ; -1 °C 8 ; 0 °C 18 ; 6 °C		*100* ; 30 °F 9 ;40 °F 5 ;40 °F 3 ;20 °F	

a) R =Fusion-Union(S1,S2)	
Note	Température
Integer	Integer
? ?	? ?
//	//
12 ;-5 °C	
10 ; -1 °C	
8 ;0 °C	
18 ;6 °C	
3 ; 30 °F	
9 ; 40 °F	
5 ; 40 °F	
7 ; 20 °F	

b) R=Fusion-Union(S1,Transformation(S2))	
Note	Température
Integer	Integer
[0..20]	[-40..50]
//	°C
12 ; -5 °C	
10 ; -1 °C	
8 ; 0 °C	
18 ; 6 °C	
6 ; **-1,11 °C**	
18 ; **4,44 °C**	
10 ; **4,44 °C**	
14 ; **-6,66 °C**	

Dans le premier cas de figure (cas (a) table 2.5), le résultat de l'intégration (Fusion-Union) est erroné. Les colonnes *S1.Température* et *S2.Temp* ne sont pas homogènes car elles ne sont pas sémantiquement équivalentes. Les outils devraient détecter que la sémantique de la donnée est une température dans les deux cas, alors que l'unité de mesure n'est pas la même.

R =Fusion-Union(S1,S2)

R(Note : {Integer}, Température : {Integer})

Plusieurs questions se posent telles que : "Quelles transformations faut-il appliquer et sur quelles colonnes ?". C'est le deuxième cas (cas (b) table 2.5).

R=Fusion-Union(S1,Transformation(S2))

R(Note : {Integer, [0..20}, Température : {Integer, [-40..50], °C})

Dans la mesure où *S1.Température* et *S2.Temp* ont le même sens, leur intégration devrait homogénéiser l'unité de mesure (°C, °F) (cas (b) table 2.5), sachant que 50 degrés Fahrenheit (50°F) = 10 degrés Celsius (10°C).

Les valeurs de la colonne *S2.Note* nécessite une transformation dans le domaine sémantique de la source S1 [0..20] (cas (b) table 2.5). ■

Plusieurs questions se posent telles que : "Quelle est la transformation nécessaire et y-a-t-il des valeurs aberrantes ?".

Il faut remarquer que des analyses sur les données devraient enrichir les définitions syntaxiques et sémantiques des données avant de réaliser toute manipulation de données.

Les données étant hétérogènes et mal renseignées dès le départ, le processus d'intégration peut causer la dégradation des données. L'absence de référentiels et le manque de la sémantique peuvent introduire des valeurs nulles, des incohérences dans les données, voire des contradictions.

Une étude comparative sur les différents ETL commerciaux existants tels que Talend Data Integration [1] (TDI) et Pentaho Data Integration [2] (PDI), nous a permis de mettre l'accent sur ces difficultés d'intégration des sources de données hétérogènes. L'utilisateur ne bénéfice d'aucune recommandation ni assistance. Il en est de même des outils de gestion de la qualité (Talend Data Quality, DataCleaner, Datiris), des SGBD (Oracle [3]) et des outils d'aide à la conception de BD (PowerAMC [4]) .

En effet, lors de l'intégration des données en provenance de plusieurs sources hétérogènes, un grand nombre de types de conflits structurels peut survenir.

Un même nom peut être donné à deux objets différents dans chacune des sources (homonymes). Des appellations différentes d'un même objet peuvent exister (synonymes).

De surcroit, les contraintes définies sur les données peuvent ne pas exister. En effet, chaque donnée (colonne) définie syntaxiquement par un type (string, char, date, integer, number, real, boolean), devrait être définie par un domaine sémantique exprimé généralement par une contrainte telles que :

— L'appartenance à un intervalle de valeurs ou un ensemble de valeurs énumérées,
— l'unicité de valeurs pour une clé primaire (Primary Key) ou la contrainte unique (Unique).
— le sens sémantique de la donnée (une chaîne de caratères qui définit le nom d'un client est différente de celle qui désigne le nom d'une entreprise).
— les dépendances entre les données (le sex dépend de la civilité).

Signalons par ailleurs qu'aucune vision globale entre les colonnes n'est prise en compte. Les dépendances éventuelles entre les colonnes devront être étudiées afin d'éviter certaines incohérences. Il est en est de même concernant le rapprochement des lignes résultats et le traitement en général des redondances. Ce qui constitue l'objet du paragraphe suivant qui est les anomalies dans les données.

1. https ://fr.talend.com/products/data-integration
2. http ://www.pentaho.com/product/data-integration
3. http ://www.oracle.com/index.html
4. http ://www.sap.com/france/pc/tech/database/software/model-driven-architecture/index.html

2.2.2 Anomalies dans les données

Les anomalies dans les données sont très variées et sont causées par plusieurs facteurs.

Les données étant de plusieurs types (string, char, date, integer, number, real, boolean), elles peuvent être représentées dans plusieurs formats plus au moins équivalents.

Les données de type chaîne de caractères (string) peuvent représenter dans la réalité plusieurs sous-catégories de données telles que les données alphabétiques, numériques ou dates. Il convient donc de différencier les **catégories** et les **sous catégories**, non seulement dans les types chaînes de caractères, mais aussi les autres types.

Les anomalies dans les données peuvent être perçues selon la classification ci-dessous. La double validité syntaxico-sémantique doit être prise en compte.

Le format de données de type date devrait être validé par des expressions régulières jj/mm/aaaa, mm/jj/aaaa, aaaa/mm/jj. Le contenu ne devrait pas être aberrant tel que la date de naissance d'un client est le 14 janvier 3011 par rapport à la date système (15 décembre 2014).

Le format de données de type numérique devrait être précisé moyennant la taille de la partie entière et celle de la partie décimale.

Les chaînes de caractères telles que -40 °C, 60 °F, 100 Kg, 1 TO devront faire partie des données numériques. Elles sont présentées dans un format (unité de mesure). Une conversion *StringToNumber* et des *expressions régulières* sont nécessaires pour définir et contrôler au mieux les données.

Les chaînes de caractères doivent respecter les normes et standards existants tels que les normes ISO (International Standard Organisation), les codes pays, les codes postaux, le formatage des adresses, les dates, le numéro IBAN (International Bank Account Number), le numéro de SIREN (Système d'Identification du Répertoire des Entreprises).

Les chaînes de caractères "Epinay sur seine" et "Epinay/seine" doivent être unifiées.

La codification des données de type caractère selon le sens de celles-ci doit être également définie moyennant des expressions régulières. Par exemple, l'énumération des données se fait par une liste (sexe : {Homme, Femme, Male, Female, M, F, 0, 1} ; size : {S, M, L, XL, XXL} ; Blood group : {A+, B+, AB}).

L'encodage des caractères peut donner un sens sémantique à la donnée. Par exemple, les caractères spéciaux peuvent être interdits pour les noms des objets ("Fuentes HernÃ©ndez", EID^2).

Les valeurs aberrantes (Outliers) (Knox and Ng, 1998) qui présentent des écarts par rapport à la moyenne ou au modèle. Le problème consiste à définir un seuil à partir duquel les mesures peuvent être rejetées. La majorité des notes de propreté de la ville de Paris appartiennent à l'intervalle de définition sauf exception, la donnée "100", ce qui peut correspondre à une valeur aberrante.

Les valeurs nulles (Null Values) (Boufarès, 1983), (Berti-Équille, 2012) constituent une des causes principales des problèmes dans les rapports décisionnels. Plusieurs interprétations sont possibles pour celles-ci :

— valeurs applicables et non renseignées : la "ville de naissance" de Charlemagne ?
— valeurs non applicables (non signifiantes) : "nom de jeune fille" pour un homme !

Les valeurs par défaut (Default Values) proches des valeurs manquantes, sont plus difficiles à détecter. Par exemple, étant donnée la liste des villes pour lesquelles on affiche la température de saison, les valeurs dans le tableau ci-dessous (table 2.6) donnent la température pour les villes de Paris et Londres. Une valeur par défaut, donnée à la ville de Londres, peut être mal interprétée. En effet, le 0 est-il la valeur par défaut (température non renseignée) ou la valeur après mise à jour (température = 0) ?.

S(NomVille : {String}, Température : {Integer, 0 par défaut, °C})

Exemple 2.7 : Les valeurs par défaut

TABLE 2.6 – Les valeurs par défaut

S	
NomVille	Température en °C
String	Integer (0 par défaut)
? ?	? ?
//	//
Paris	5,4
Londres	0

L'obsolescence des données (Peralta, 2006) : qui peut être mesurée par deux facteurs. Le premier facteur de résistance des données par rapport aux changements

à partir de la date d'extraction à la date de livraison. Le deuxième facteur exprime l'ancienneté des données : c'est l'écart entre la création et la mise à jour des données par rapport à la livraison sans se soucier de la date de création.

Il faut remarquer que les anomalies citées ci-dessus ne portent que sur une colonne à la fois. Il existe cependant, d'autres anomalies portant sur plusieurs colonnes en même temps à savoir les liens entre colonnes pour exprimer les dépendances entre elles et les lignes en doubles ou similaires.

Les dépendances (Simonenko and Novelli, 2012), (Dallachiesa et al., 2013) entre les colonnes telles que les dépendances fonctionnelles constituent un autre volet d'anomalies. Non seulement il faut les découvrir mais aussi les vérifier. Les contraintes définies sur le schéma conceptuel ne suffissent pas pour éviter les données erronées dans la base de données. La détection des redondances inutiles n'est qu'une partie du problème. La question sémantique reste totalement ouverte pour savoir quelles sont les lignes à éliminer.

Les redondances (doubles et similaires) entre les lignes doivent être détectées et corrigées. Ce problème est appelé aussi "Record Linkage", "Duplicates Data" (Hernandez and Stolfo, 1995), (Sarawagi and Bhamidipaty, 2002), (Koudas et al., 2006), (Benjalloun et al., 2009), (Boufarès et al., 2012a), (Boufarès et al., 2012b), (Boufarès et al., 2013). La similarité entre les données nécessite un calcul de similarité entre les données. Différentes méthodes de mesure existent dans la littérature (Bilenko and Mooney, 2003), (Hernandez and Stolfo, 2004), (Cohen and Richman, 2004), (Koudas et al., 2006), (Winkler, 2006), (Benjalloun et al., 2009), (Berti-Équille, 2007). Les chaînes de caractères "'Le Bon Adam" et "Le Bon Adan" sont proches lexicographiquement. Elles sont donc équivalentes syntaxiquement et sémantiquement. Alors que les chaînes "London" et "Londres" ainsi que les chaînes "Pékin" et "Beijing" sont éloignées lexicographiquement. Elles sont syntaxiquement différentes et sémantiquement équivalentes. Ces quatre dernières chaînes de caractères appartiennent à la **catégorie sémantique** *City*.

Notons que les méthodes de mesure de similarité actuelles sont pauvres en sémantique.

Les **principales causes** des anomalies dans les données résident principalement lors de la saisie manuelle ou de la génération automatique et au cours des transformations et agrégations. Les utilisateurs ne sont pas vraiment assistés dans cette tâche. En effet, d'une part, les formulaires de saisie sont très mal conçus : peu ou pas d'utilisation de référentiels afin d'éviter par exemple les erreurs typographiques, les abréviations et les mélange entre colonnes. La définition syntaxique des données (type de données), n'est pas toujours accompagnée des contraintes (constraints) ni des déclencheurs (triggers) permettant de filtrer les données.

En effet, les formulaires Web, source d'anomalies, contiennent d'une part les noms des colonnes à remplir et d'autre part, les données renseignées par l'utilisateur.

Les noms de colonnes peuvent être la source du problème dans la mesure où ils prêtent à confusion ou ils sont mal compris.

Exemple 2.8 : Les anomalies dans les formulaires Web français

TABLE 2.7 – Formulaire Web source d'anomalies

Nom du champ	Donnée-Saisie	Nom du champ	Donnée-Saisie
Civilité	**Mme**	Email	**adem.lebon**
Prénom	**Adem**	Adresse	**18 rue Carnot**
Nom	**LeBon**	Adresse2	
Sexe	**Masculin**	Ville	**Paris**
Téléphone1	**0155555599**	Code Postal	**92150**
Téléphone2	**0653545599**	Pays	**France**
Fax	**0148904890**		

Les champs *Téléphone1* et *Téléphone2* prêtent à confusion. Il est en de même des champs *Adresse* et *Adresse2*.

Les seules valeurs admises dans civilité devront appartenir à (M., Mme). Le champ sexe est dépendant de la civilité. Il devrait être déduit automatiquement. Le champ pays doit être renseigné avant la ville. Celui-ci devrait servir pour établir automatiquement les contrôles nécessaires du téléphone et fax. ■

De nombreuses questions relatives à la qualité de données doivent être posées, tout au long du cycle de vie de la donnée et surtout au niveau du processus d'intégration, sur :

1. les contraintes sur les données,
2. la compatibilité des types syntaxiques de données,
3. la compatibilité de types sémantiques des données,
4. la transformation, la correction et la validation des données,
5. la déduplication des données et l'élimination des redondances.
6. comment le savoir ?.

Pour aider à résoudre ces problèmes, l'ajout de la sémantique aux métadonnées peut être une solution afin de comprendre les données, détecter et localiser les anomalies existantes. Ainsi, des actions automatiques de correction et de nettoyage peuvent être proposées.

Dans ce qui suit, un parcours des différents travaux existants va nous permettre d'établir un bilan sur les démarches proposées et prouver l'originalité de notre contribution.

Rappelons qu'à l'aire du BigData, les sources de données à manipuler n'ont pas forcément un descriptif (meaningless schema). Ainsi, une étape supplémentaire de reconstruction du schéma à partir de l'analyse des données nous semble déterminante pour aborder l'approche de nettoyage de données. C'est ce qui caractérise notre approche.

2.3 Traitement des anomalies

Le présent paragraphe exposera l'essentiel des travaux présentés dans la littérature sur sur le rapprochement de schémas, la détection et la correction des anomalies.

La manipulation des sources de données dans le but de les explorer et d'en extraire des connaissances, nécessite donc de reconnaître le sens de chaque colonne (reconstruction sémantique) et éventuellement le lien entre elles afin de remédier aux anomalies. Le lien entre les colonnes pourrait donner une idée sur le contenu globale de la source (le concept).

2.3.1 Rapprochement de schémas

Le rapprochement de schémas (Schema Matching) est le processus d'identification des objets sémantiquement similaires (figure **??**) (Rahm and Bernstein, 2001). Le principe consiste à identifier et caractériser les correspondances entre deux schémas qui existent. Ils se basent essentiellement sur la comparaison lexicographique des noms des colonnes.

Le rapprochement de schéma permet de faciliter le processus d'intégration et la création d'un entrepôt de données à cause de l'hétérogénéité des sources.

Différentes méthodes existent dans la littérature (Rahm and Bernstein, 2001), (Sais, 2007), (Do and Rahm, 2002), (Madhavan et al., 2001), (Castano et al., 2001), (Li and Clifton, 2000) (figure 2.1).

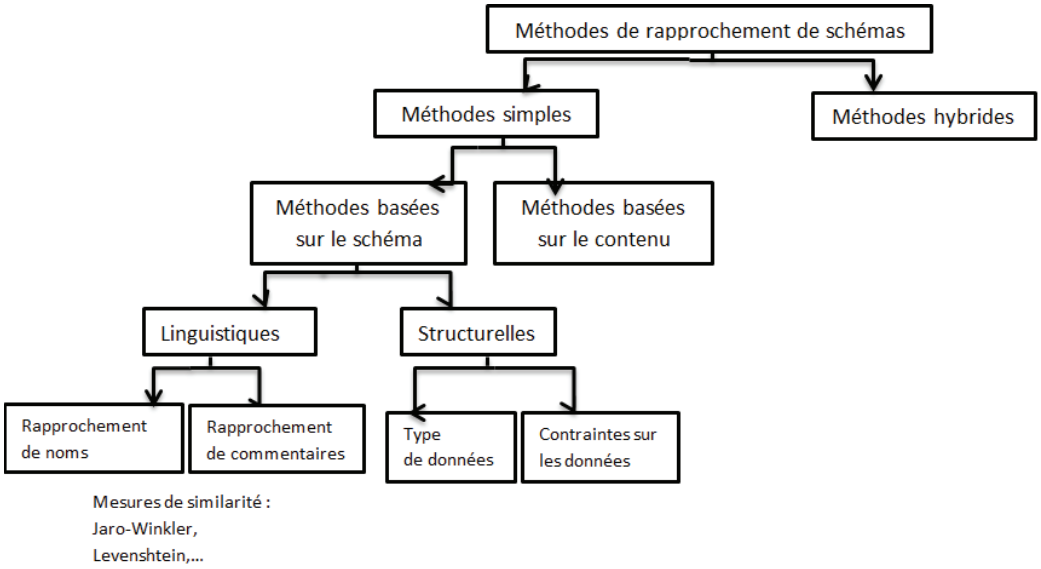


FIGURE 2.1 – Approches de rapprochement de schéma

Parmi ces méthodes, on peut citer :

Approche linguistique (non basée sur la définition syntaxique et sémantique de la colonne) (Glenn and Sethi, 2001) : elle utilise les noms des colonnes et leurs commentaires pour trouver des éléments sémantiquement similaires. Deux niveaux sont définis :

— rapprochement de noms : calculer la similarité entre eux (en prenant en compte les synonymes, les homonymes [5], les hyperonymes [6] existants dans des dictionnaires de données tel que WordNet [7]) avec les mesures de distance d'édition telles que Levenshtein, Jaro-winkler et la mesure phonétique Soundex (Bilenko and Mooney, 2003).

— rapprochement des commentaires : exploiter la sémantique des commentaires associés à chaque colonne, afin de trouver une similarité entre les colonnes des deux schémas.

Exemple 2.9 : Rapprochement des commentaires

Soit S1 et S2 deux sources de données. Des commentaires ont été renseignés sur les deux colonnes *S1.BirthD* et *S2.DateB*. On rapproche les deux commentaires (figure 2.2) afin de détecter que ces deux colonnes sont similaires.

5. A est homonyme à B si A et B s'écrivent de la même façon mais ils ont un sens entièrement différent
6. A est hyperonyme de B si B est inclu dans A
7. wordnet.princeton.edu

FIGURE 2.2 – Rapprochement de commentaires de deux attributs

Approche structurelle basée sur la définition des colonnes (type et contraintes) (Larson et al., 1989) : la similarité est calculée à la base de la correspondance entre :

— les types de données : la notion de compatibilité est utilisée entre les types pour le rapprochement. Par exemple pour une chaîne de caractères, les types existants dans la table (2.8) sont équivalents.

TABLE 2.8 – Compatibilité des types de données

Type de données compatibles
String, varchar, char
Number, integer, real
Date
Boolean

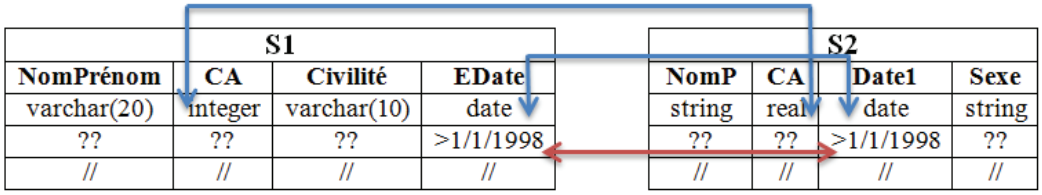
— les contraintes sur les données : rapprocher les contraintes définies sur chaque colonne. Par exemple, rapprocher la contrainte *clé primaire* à la contrainte *Unique*.

Exemple 2.10 : Exemple de rapprochement en utilisant les contraintes ainsi que le type des données

Soit deux schémas S1 et S2 avec (figure 2.3) :

S1(NomPrénom : {varchar(20)}, CA : {integer}, Civilité : {varchar(10)}, EDate : {date, >01/01/1998}).

S2(NomP : {string}, CA : {real}, Date1 : {date, >01/01/1998}, Sexe : {string}, CI : {string})



S1			
NomPrénom	**CA**	**Civilité**	**EDate**
varchar(20)	integer	varchar(10)	date
??	??	??	>1/1/1998
//	//	//	//

S2			
NomP	**CA**	**Date1**	**Sexe**
string	real	date	string
??	??	>1/1/1998	??
//	//	//	//

FIGURE 2.3 – Rapprochement structurelle

S1.CA et S2.CA sont similaires puisqu'ils ont le même type de données (integer).

S1.EDate et S2.Date1 ont le même type "date" et vérifient la même contrainte (>01/01/1998), donc ils sont proches.■

Notons aussi que souvent les descriptifs et les contraintes sur les données sont peu ou mal définis. Peu d'utilisateurs remplissent les commentaires ou spécifient des contraintes sur les données.

Approche hybride (Doan et al., 2001) : celles-ci combinent les deux approches précédentes. En effet, ni la reconnaissance lexicographie du nom de la colonne ni les contraintes ne peut suffire à elles seules de rapprocher les schémas. L'approche linguistique combinée avec l'approche basée sur les contraintes permet une meilleure correspondance entre les deux schémas.

Exemple 2.11 : Approche hybride

Dans l'exemple précédent (exemple 2.10), les colonnes S1.NomPrénom et S1.Civilité et S2.NomP, S2.Sexe et S2.CI ont le même type (string). Ici le type ne suffit pas pour décider de la correspondance. Un rapprochement en fonction des noms peut être appliqué pour rapprocher S1.NomPrénom à S2.NomP. Par contre, les autres colonnes S1.Civilité, S2.Sexe et S2.CI ne sont pas rapprochables.

Le rapprochement linguistique ne prend pas en compte la langue des colonnes. Tous les noms des colonnes sont des chaînes de caractères, sans aucune sémantique, à rapprocher. ■

L'originalité de notre approche est que nous construisions un schéma sémantique. Les colonnes du schéma sont bien définies. Chaque colonne a une catégorie et une sous-catégorie (telle que la langue). Donc c'est peu envisageable de rencontrer un nom colonne tel que *CI*.

La détection des relations entre colonnes pourra faciliter la détection des colonnes similaires. Dans l'exemple 2.10, il existe une dépendance entre la colonne S1.Civilité et S2.Sexe.

Approche basée sur le contenu en détectant les doublons d'une source (Bilke and Naumann, 2005), (Bilke et al., 2005). L'approche consiste à comparer le contenu de deux enregistrements et si les valeurs de deux colonnes sont similaires alors ces deux derniers sont rapprochables. Une matrice de similarité est calculée pour chaque paire d'enregistrements (table 2.9).

TABLE 2.9 – Matrice de similarité pour une paire d'enregistrements

		FirstName	**Phone**	**Sex**	**Civility**	**City**
		Jean	0666554426	M	Mister	Paris
FName	Paris	0	0	0	0	1
Ph	0666554425	0	0,96	0	0	0
S	M	0	0	1	0	0
Adr	Paris	0	0	0	0	1

Une nouvelle matrice est proposée contenant la moyenne des similarité des doublons, pour les différentes colonnes (table 2.10). A partir de cette matrice, le rapprochement de schémas est déduit.

TABLE 2.10 – Matrice des moyennes de similarités

	FirstName	**Phone**	**Sex**	**Civility**	**City**
FName	0	0	0	0	1
Ph	0	0,96	0	0	0
S	0	0	1	0	0
Adr	0	0	0	0	1

Notons que l'on peut rapprocher des colonnes différentes mais avec un contenu similaire. Par exemple, on rapproche les colonnes *FName* à *City* vue que la valeur "Paris" représente d'une part le nom d'une personne et d'autre part, le nom de la capitale de France.

Deux colonnes sont similaires à la même colonne. La valeur "Paris" apparaît dans les deux attributs.

Alignement des ontologies On retrouve le rapprochement de schémas dans le domaine des ontologies. Le concept "alignement des ontologies" est utilisé pour le rapprochement des ontologies.

Un état d'art sur les ontologies et le langage OWL est présenté dans l'annexe (section A.1).

Les travaux existants (Euzenat and Valtchev, 2006), (Shvaiko and Euzenat, 2013) définissent les mêmes méthodes de rapprochement de schémas (linguistique, structurelle, hybrides). Ils ajoutent cependant, des méthodes qui permettent d'exploiter la particularité des ontologies qui est les relations entre entités.

Le mot "entité" est utilisée dans le monde ontologie pour désigner les colonnes et les noms des sources.

Les liens entre entités (équivalence, transitivité, symétrie, disjonction) sont utilisés pour rapprocher les entités entre elles (Giunchiglia et al., 2005).

En conclusion, l'alignement en fonction des relations et des instances est très peu utilisé. La comparaison terminologique et structurelle des noms des objets prend le dessus sur les différentes approches (Shvaiko and Euzenat, 2013). Cependant, ces noms peuvent être peu significatifs ou contiennent peu de sémantique.

2.3.2 Détection des anomalies

Le **profilage** de données représente une étape primordiale pour la détection des anomalies dans les données sur une colonne. Le profilage est une collection de statistiques. Il consiste en une analyse exploratoire des données sur trois niveaux : (i) **analyse** des colonnes indicateurs de fréquence, les valeurs nulles pour détecter des données douteuses comme les valeurs aberrantes, ; (ii) analyse des dépendances (les dépendances fonctionnelles par exemple) ; (iii) analyses des doublons et des similaires (Johnson and Dasu, 2001), (Berti-Équille, 2006).

Le profilage est très utilisé dans le monde industriel. Force est de constater que c'est une tâche très manuelle. Les utilisateurs ne bénéficient d'aucune assistance ni aide. L'utilisateur est censé connaitre la sémantique des colonnes et avoir une idée sur contenu de la source manupilée !

Une étude comparatif des trois outils open source existants : "Talend Data Quality [8] (TDQ)", "DataCleaner [9] (DC)" et "DatirisProfiler [10] (DP)", nous permet d'avoir un aperçu sur les fonctionnalités proposées (table 2.12).

Les statistiques simples comportent des indicateurs tels que le nombre de valeurs total, nombre de valeurs nulles, nombre de valeurs en doubles et autres statistiques appliquées sur tout type de données.

8. www.talend.com/data_quality
9. http ://datacleaner.org/
10. http ://www.datiris.com/

TABLE 2.11 – Tableau comparatif des différents outils de profilage (analyse de colonnes)

Analyses de colonnes (une à une)	**Indicateurs**	**TDQ**	**DC**	**DP**
Statistiques simples	Nombre total des valeurs	X	X	X
	Nombre de valeurs nulles	X	X	X
	Nombre de valeurs distinctes	X	X	X
	Nombre de valeurs doubles	X	X	X
	Table de fréquence	X		
Statistiques sur les chaînes	Longueur maximale des chaînes	X	X	X
	Longueur minimale des chaînes	X	X	X
	Longueur moyenne des chaînes	X	X	X
Statistiques sur les numériques	Valeur maximale des numériques	X	X	
	Valeur minimale des numériques	X	X	
	Moyenne des numériques	X	X	
	Écart type des numériques	X	X	
Statistiques sur les dates	Fréquence des années	X	X	
	Motif de fréquence de Date	X		
Format des chaînes	Motif de fréquence	X		
Nature des chaînes				
Langue des chaînes				
Nature de la langue des chaînes (Latin, Arabe)			X	
Nombre de chaînes valides syntaxique				
Nombre de chaînes valides sémantique				

TABLE 2.12 – Tableau comparatif des différents outils de profilage (analyse de source)

Analyses de la source		TDQ	DC	DP
Détection des doublons et similarités	Choix des attributs de déduplication	X		
	Choix des algorithmes de similarité	X		
	Choix des seuils de similarité	X		
Dépendances fonctionnelles	Dépendances simples	X		
	Dépendances multiples			
	Détection manuelle	X		
	Détection automatique			
Détection des anomalies syntaxiques				
Détection des anomalies sémantique				
Assistance utilisateur				
Recommandation				

Les outils de profilage existants sur le marché fournissent des résumés statistiques sur les données. Ces statistiques concernent exclusivement la syntaxe des données. Elles ne sont pas facilement interprétables et ne permettent pas de détecter les différentes anomalies dans une source de données.

Des règles métiers sont à définir en fonction des besoins utilisateurs telles que :

— Une personne de sexe masculin, n'a pas de nom marital.
— La date de recrutement doit être inférieure à la date de démission.
— Deux attributs sont considérés comme dépendants l'un de l'autre si et seulement si les valeurs de deux colonnes vérifient la dépendance fonctionnelle à plus de 80%.

Des seuils sont aussi à définir par les utilisateurs sur les indicateurs de l'analyse. Par exemple, on n'accepte pas plus de 40% de valeurs nulles dans une colonne donnée.

Ces outils n'assistent pas donc l'utilisateur dans sa démarche, qui ne bénéfice d'aucune recommandation. C'est à lui seul d'analyser les résultats de profilage et d'en déduire les actions de nettoyage sur les données.

Les données ne sont pas catégorisées. Aucun indicateur n'existe pour déterminer la nature de la colonne traitée.

Les outils offrent des analyses statistiques sur les données pour chaque colonne et pour chaque type de donnée. Par contre, aucune analyse sémantique sur les données n'est effectuée.

Cependant il reste à se poser beaucoup de questions telles que : "Est ce que les résultats donnés par les analyses permettent de spécifier le type de traitement à proposer pour l'utilisateur ? ", "Est ce que les résultats d'analyses sont facilement interprétables ? ", "Comment peut-on déduire les attributs clés pour une élimination des doublons ? ".

Les contraintes définies éventuellement sur les données ne sont pas prise en compte. Il faudrait ajouter des vérifications sur ces contraintes tels que les *check* de SQL ou encore les *triggers*.

2.3.3 Correction des anomalies

Afin de remédier à certaines anomalies, différents travaux ont été proposés dans littérature. On s'intéresse dans ce qui suit aux deux actions de nettoyage : l'homogénéisation des données et la déduplication.

Une étude bibliographique sur la problématique d'élimination des doublons et similaires dans une source a été menée en profondeur.

L'élimination des similaires est le processus de comparaison de lignes d'une source de données afin de déterminer celles qui présentent le même objet du monde réel. Il est appelé encore le processus de dé-doublonnage (déduplication), de résolution d'entité (Entity resolution), de rapprochement de données (Entity Matching), de couplage d'enregistrements (record linkage) (Hernandez and Stolfo, 1995), (Sarawagi and Bhamidipaty, 2002), (Koudas et al., 2006), (Benjalloun et al., 2009), (Boufarès et al., 2012a), (Boufarès et al., 2012b), (Boufarès et al., 2013). Il consiste en deux étapes : la comparaison (Match) et la fusion (Merge).

Nous résumons les différentes étapes de l'algorithme d'élimination des doublons et similaires en cinq étapes (algo 1) :

Algorithm 1 Deduplication

```
Input : Data Source S
Output : Cleaned Target Data SC
Begin
    Choose key attributes (deduplication attributes)
    Choose similarity distance
    Choose Match method
    Choose Merge approach
    Evaluate the deduplication rate
return SC
End Algorithm Deduplication
```

2.3.3.1 Choix des attributs de dédoublonnage

Peu de travaux portent sur le choix des attributs clés. Parmi ces travaux, citons celui (Fan et al., 2009) qui propose de choisir les attributs candidats pour le dédoublonnage en utilisant les relations de dépendances entre les attributs.

Les outils de profilage ne permettent pas de bénéficier des résultats des analyses, à savoir de ne pas utiliser les colonnes contenant plusieurs valeurs nulles comme attribut clé.

2.3.3.2 Choix d'un algorithme de similarité

La similarité des données est définie comme étant la distance entre deux valeurs, de même type, de deux enregistrements (comme illustré dans la figure 2.4).

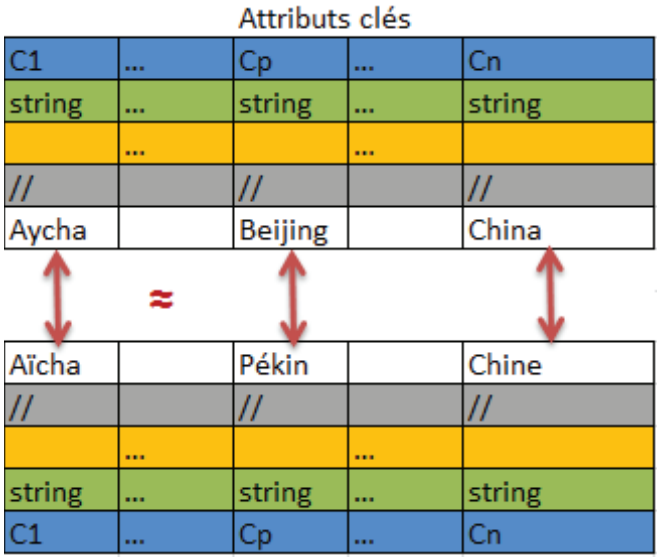


FIGURE 2.4 – Similarité entre les données

Plusieurs questions ont besoin d'une réponse justifiée : quelle mesure utiliser pour calculer la similarité entre les noms des personnes, les noms des villes, les adresses mail, les bibliographies ? Est-ce que c'est la même mesure utilisée pour tous ? Après quel seuil à fixer pour chaque donnée et pour chaque mesure ?.

Pour répondre à une partie de ces questions, plusieurs méthodes de comparaison des attributs d'un enregistrement existent dans la littérature (Bilenko and Mooney, 2003), (Hernandez and Stolfo, 2004), (Cohen and Richman, 2004), (Koudas et al., 2006), (Winkler, 2006), (Benjalloun et al., 2009), (Berti-Équille, 2007). Nous avons classé ces différentes méthodes, qui traitent des chaînes de caractères, en deux groupes : se prononce comme et s'écrit comme. Une liste non exhaustive, présentée

ci-après, des méthodes de calcul de distance de similarité classées en fonction du type des données.

Dans cette étude, nous n'allons-nous intéresser qu'à certaines méthodes. Le but étant de montrer l'apport d'une sémantique supplémentaire sur les données pour de meilleures comparaisons.

1. Mesures pour les chaînes de caractères : Il existe deux types d'algorithmes lexicographiques et phonétiques

   Mesures lexicographiques : les chaînes de caractère s'écrivent d'une manière similaire (Elmagarmid et al., 2007) :

   — La distance Levenshtein (Levenshtein, 1966) calcule la distance de similarité en fonction du nombre des opérations d'édition (insertion, suppression et remplacement).

   — La distance de Q-gram (Ukkonen, 1992) divise les chaînes en q-caractères. Deux chaînes sont similaires si elles partagent le plus grand nombre de Q-gram.

   — La distance de Jaro-Winkler (Winkler, 2006), (Cohen, 1998) est souvent utilisée pour les noms et prénoms. Elle identifie les caractères en commun entre deux chaînes et le nombre de transposition.

   Mesures phonétiques : les chaînes peuvent être similaires phonétiquement sans être similaires en fonction des caractères (Elmagarmid et al., 2007).

   — Soundex (Cohen, 1998) regroupe des consonnes phonétiquement identiques.

   — Double Metaphone (Philips, 2000) sont une alternative de Soundex, appliquées à d'autres langues autre que l'anglais.

   — NYSIIS (New York State Identification and Intelligence System) [11] est un algorithme phonétique conçu en 1970. Il permet d'améliorer la précision de 2,7% par rapport à l'algorithme Soundex.

   Une mesure lexicographique, telle que Jaro-Winkler ou Levenshtein, est appliquée sur le code généré par ces mesures phonétiques. Nous utilisons la notation suivante : mesure phonétique + combinée + avec la mesure lexicographique.

   — SoundexLevenshtein pour la combinaison des deux mesures Soundex et Jaro-Winkler

   — SoundexJaroWinkler : Soundex est combinée à Jaro-Winkler

   — de même pour NYSIISLevenstein, NYSIISJaroWinkler, DoubleMetaphoneLevenshtein et DoubleMetaphoneJaroWinkler

11. http ://en.wikipedia.org/wiki/New_York_State_Identification_and_Intelligence_System

Dans le papier (Elmagarmid et al., 2007), on conclut que la mesure Jaro-Winkler est mieux utilisée pour les noms.

Exemple 2.14 : Mesures de similarité sur différentes chaînes de caractères

Nous présentons dans le tableau suivant (table 2.13), des distances de similarité (en %) pour différentes chaînes selon les mesures de similarité présentées ci-dessus.

On commence avec les distances textuelles avec Jaro-Winkler (JW), Jaccard (Jac), Levenshtein (Lv) et Q-gram (Qg). Ensuite, les mesures phonétiques telles que Soundex (Sdx), Double Metaphone (DM) et Nysiis (Ny). Ces mesures ont été combinées avec les distances Jaro-Winkler et Levenshtein.

TABLE 2.13 – Mesures de similarité sur différentes chaînes de caractères

Chaine1	Chaine2	JW	Jac	Lv	Qg	SdxLv	SdxJW	DMJW	NyLv	NyJW
Aïcha	Aicha	82,20	0,00	66,60	50,00	100,00	100,00	100,00	66,60	61,10
Lebon	Le Bon	79,00	0,00	57,10	50,00	100,00	100,00	100,00	100,00	100,00
Jérémy	Jérémi	84,90	0,00	71,40	80,00	100,00	100,00	100,00	83,30	96,60
Pékin	Beijing	54,90	0,00	25,00	25,00	50,00	66,60	75,00	33,30	57,70
Londres	London	71,40	0,00	42,80	60,00	75,00	88,30	88,30	66,60	89,30
adam@yahoo.fr	adam@yaho.fr	98,40	66,60	92,30	100,00	100,00	100,00	100,00	100,00	100,00
lebon@yahoo.fr	lbon@yahoo.fr	97,80	66,60	92,80	91,60	100,00	100,00	100,00	66,60	89,90
0123435345	0123435355	96,00	0,00	90,00	87,50	0,00	100,00	100,00	0,00	100,00
0678912346	0178912346	87,30	0,00	90,00	77,70	0,00	100,00	100,00	0,00	100,00
35°C	-35°C	93,30	75,00	80,00	100,00	100,00	100,00	100,00	100,00	100,00
104°F	104°FF	96,60	50,00	83,30	100,00	100,00	100,00	100,00	100,00	100,00

Les chaînes “Aïcha” et “Aicha” (avec une différence au début de la chaîne) sont proches selon Jaro-Winkler à 82,20%. Elles se prononcent de la même façon : la similarité est égale à 100% avec les mesures phonétiques.

Toutes les mesures textuelles ainsi que phonétiques renvoient que les deux chaînes “Pékin” et “Beijing” ne sont pas assez proches. Par exemple, avec la distance Jaro-Winkler (qui est recommandée dans la littérature pour les noms (Elmagarmid et al., 2007)), elles sont similaires à 54,90% seulement. Cependant, ces deux chaînes présentent le nom d’une même ville en deux langues différentes (français et anglais). Ces deux chaînes sont dites égales sémantiquement. De même pour les chaînes “Londres” et “London”.

2. Mesures pour les numériques : les numériques peuvent être comparés comme des chaînes de caractères (Elmagarmid et al., 2007).

   Cependant si on compare les deux numériques “123,0000001” et “123” en les transformant en chaînes de caractères, la distance de similarité sera très grande et donc ils seront différents.

   Par contre, un calcul de différence entre ces deux nombres renvoie un score très faible et donc les deux numériques sont proches.

3. Mesures pour les dates : un calcul de différence entre les jours, les mois et les années.

Cependant, ce qui manque à ces mesures est la prise en compte de la sémantique des données dans le choix des méthodes et des seuils adéquats. Le problème devient plus complexe puisque de nos jours avec la notion du BigData, souvent les métadonnées des sources sont peu significatives ou encore inexistante.

Jusqu'à ce jour le choix de la mesure de similarité repose seulement sur quelques critères telles que la longueur de la chaîne à comparer (Sais, 2007) ou bien la distance Jaro-Winkler est mieux utilisée avec les noms (Cohen and Richman, 2004). Souvent aussi ces mesures sont choisies en fonction de l'expérience utilisateur.

L'originalité de notre travail est de recommander ces mesures en fonction de la nature/catégorie et la langue des données. Donc un de nos objectifs alors est de reconnaître la catégorie et/ou la sous-catégorie sémantique des données afin de proposer les meilleures méthodes.

2.3.3.3 Choix d'une approche de correspondance (Fonction Match)

Différentes approches existent pour la comparaison d'enregistrements (Köpcke and Rahm, 2009) : les approches probabilistes et celles déterministes. Nous ne nous sommes intéressés, dans cette étude, qu'aux approches déterministes. Parmi ces dernières, citons les techniques les plus utilisées :

Techniques basées sur les distances :

Le principe est de calculer la distance pondérée entre les données d'un enregistrement (Guha et al., 2004). La formule 2.1 présente la somme pondérée des similarités calculées (d_a) pour les attributs clés et un poids (p_a) est donné à chacun d'eux. Cette somme devrait être inférieure à un seuil global (ε) (Dey et al., 1998). Les poids et le seuil sont définis par un expert. Aucune assistance ou recommandation pour l'utilisateur dans le choix des méthodes de similarité ainsi que les seuils.

$$\sum_{a=1}^{n} p_a.d_a \leq \varepsilon \tag{2.1}$$

Techniques basées sur les règles :

Ces techniques sont un cas particulier du cas précédent tel que la distance entre deux enregistrements est comprise entre 0 et 1. Elles ont besoin de l'expertise utilisateur pour décider des règles de correspondance (Matching Rules). Parmi les travaux utilisant cette approche, il existe (Wang and Madnick, 1989), (Lim et al., 1993), (Hernandez and Stolfo, 1995), (Hernandez and Stolfo, 1998), (R. Ananthakrishna and Ganti, 2002) et (Galhardas et al., 2000). Ils utilisent une base de règles de

connaissances afin de mettre en oeuvre une équation théorique. Cette dernière est l'ensemble de règles logiques qui définissent si deux enregistrements sont similaires en fonction du calcul de distance de similarité.

Vu le nombre important des méthodes de mesure de similarité, le choix de la meilleure mesure ainsi que du meilleur seuil reste toujours un casse-tête pour les utilisateurs. Ce choix devient plus compliqué quand les attributs clés, utilisés pour le dédoublonnage, n'ont pas de noms compréhensibles. Une information sémantique sur les métadonnées pourrait faciliter ce choix et lui donner un sens.

Techniques basées sur les relations entre données :

Parmi ces travaux, citons l'approche proposée dans (**?**) qui s'intéresse aux relations hiérarchiques existantes entre les colonnes. Celles-ci peuvent être exploiter pour rapprocher les contenu de ces colonnes.

Par exemple, il existe une relation hiérarchique entre Ville et Pays. Si pour deux valeurs distinctes de Pays, on a un ensemble important des noms de villes en commun, alors les deux pays sont identiques.

Dans le même contexte, le travail de (Kalashnikov and Mehrotra, 2006) propose d'utiliser les relations d'une BD relationnelle afin de rapprocher les données. La problématique est de gérer l'ambiguïté des références. Chercher à associer à chaque référence son auteur. Par exemple, pour réconcilier les noms d'auteur "D. White" avec "Don White" ou "Dave White", ils proposent de comparer le contenu des autres colonnes, en relation avec la colonne *Auteur*, telles que *Affiliation* ou *Co-auteur* afin de reconnaître la "vraie" valeur du "D. White".

Le travail dans la thèse (Sais, 2007) et (Saïs and R.Thomopoulos, 2014), propose aussi une méthode de réconciliation des références décrites relativement à un même schéma. Elle permet de déterminer celles qui réfèrent la même entité du monde réel et celles qui réfèrent des entités différentes du monde réel. Cette approche consiste à exploiter en plus des informations syntaxiques présentes dans les données, les connaissances du domaine déclarées explicitement dans une ontologie. Dans cette approche, l'exploitation du contenu de l'ontologie a deux rôles : (i) homogénéisation des données hétérogènes en les décrivant relativement au contenu d'une même ontologie en les enrichissant par les concepts, les termes et les relations du domaine représentés dans l'ontologie et (ii) renforcer la tâche de réconciliation par des connaissances du domaine.
Cette approche propose une première méthode logique N2R (Réconciliation Numérique de références) qui repose sur les relations existantes entre les données (telles que l'équivalence, la transitivité, la symétrie et la disjonction). Ces relations sont traduites en règles de réconciliation. Un algorithme d'inférence est appliqué à ces règles afin d'en déduire la correspondance. Une deuxième méthode numérique L2R

(Réconciliation Logique de références) calcule un score de similarité entre les enregistrements. L'utilité de cette approche est la combinaison des deux méthodes N2R et L2R pour décider de la réconciliation ou non des données. Exemple, si une donnée X appartient à une classe C1 et une donnée Y appartient à une classe C2, et une relation de disjonction existe entre les deux classes C1 et C2, alors X est différent de Y même si une similarité existe entre eux.
Cette approche repose sur la présence de relations entre les données, cependant peu d'utilisateurs définissent ces relations. Proposer des analyses en fonction de la sémantique des métadonnées permettra la reconnaissance de certaines relations entre données.

Toutes ces approches sont plutôt manuelles. Elles se basent sur le fait de connaître non seulement le sens des colonnes mais aussi les relations qui peuvent exister entre elles notamment les clés primaires et étrangères et les dépendances fonctionnelles.

Par ailleurs, ces différentes approches manquent de sémantique. Aucune de ces dernières ne peut dire à la fois que les deux chaînes de caractères "London" et "Londres" sont d'un côté similaires et d'un autre coté différentes. Les deux chaînes sont similaires si et seulement si elles appartiennent à une même catégorie sémantique "City". Elles sont différentes si le sens de ces deux chaînes était "Name" ou "FirstName" (figure 2.5).

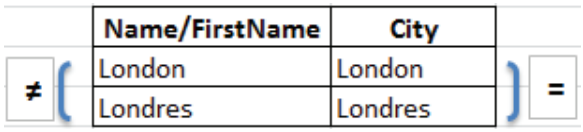


FIGURE 2.5 – Similarité sémantique entre les données

Une fois deux enregistrements sont définis comme similaires, la fusion de ces deux derniers est proposée afin de créer un enregistrement résultat plus riche.

2.3.3.4 Choix de la stratégie de fusion des tuples similaires (Fonction Merge)

La fonction Merge correspond la fusion des enregistrements. La fusion permet de créer un nouvel enregistrement contenant plus d'informations. Merge 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. Cette étape demande la définition d'expression ou de règles de fusion. Quelques travaux abordent cette problématique.

Les travaux (Hernandez and Stolfo, 1995), (Hernandez and Stolfo, 1998) fusionnent deux chaînes de caractères en gardant la plus longue. Cette proposition devrait respectée certaines règles de cohérence telle que une expression régulière. Ceci permettrait d'éviter d'introduire des incohérence dans le résultat, par exemple, si "Adamsss LeBon" ne devrait pas remplacer "Adam Lebon".

Qu'est ce qu'il on est des données de type date ou de type numérique ?

Ou aussi que faire pour les attributs qui n'ont pas servi pour le dédoublonnage ? Comment devront-ils être fusionnés ? Des règles pour leur fusion devront être définies. Des réponses restent à apporter dans les sections suivantes.

2.3.3.5 Évaluation du taux d'élimination des similaires et des doublons

Cependant, la problématique dans la fonction Match reste les **vrai/faux doublons** : comment évaluer les résultats fournis par les mesures de distance de similarité et les règles de décision ? La problématique des vrais/faux doublons présente un vrai enjeu dans le processus d'élimination des données similaires.

Une clé de rapprochement (Matching key) est l'ensemble des attributs clés, défini par l'utilisateur et utilisé pour le dédoublonnage.

Les faux négatifs apparaissent quand la clé de rapprochement étant trop discriminante. C'est le cas où deux enregistrements représentent le même objet, mais ne sont pas identifiés comme similaires (Stricker, 2000).

Les faux positifs apparaissent quand la clé de rapprochement est non suffisamment discriminante (Stricker, 2000). C'est le cas où deux enregistrements sont identifiés similaires alors qu'ils ne représentent pas le même objet. Exemple, "Dupont" et "Dupond" sont identifiés similaires alors qu'ils représentent deux personnes différentes.

Afin de traiter cette problématique, on propose dans la littérature un ensemble de mesures telles que la précision, le rappel, F-mesure. Le rappel=$\frac{n1}{n1+n3}$ et la précision=$\frac{n1}{n1+n2}$ (Stricker, 2000) (table **??**).

TABLE 2.14 – Le rappel et la précision

	Enregistrements Similaires	Enregistrements Non Similaires
Nombre d'enregistrements Supprimés	n1	n2
Nombre d'enregistrements Non Supprimés	n3	n4
Nombre Total	n5	n6

F-mesure est la mesure qui allie la précision et le rappel. C'est la moyenne harmonique de la précision et du rappel.
F-mesure= $\frac{2.precision.rappel}{precision+rappel}$.

Notons que la comparaison des enregistrements entre eux se fait de deux manières différentes : séquentielle ou par blocs.

2.3.3.6 Les méthodes de comparaison des tuples

Deux types de comparaison sont définis : (i) la comparaison de tous les enregistrements entre eux (la production d'un produit cartésien) et (ii) la comparaison par blocs d'enregistrements.

La première approche présentée dans (Benjalloun et al., 2009) présente trois algorithmes comparant chacun tous les enregistrements n existants dans une source. La complexité de cette approche est de l'ordre O(n^2).

Quant à la deuxième approche, le principe de regroupement des données (Köpcke and Rahm, 2009) est de partitionner la source de données en des blocs selon une clé générée à partir des attributs d'un enregistrement. La clé de regroupement (Blocking key) peut être un seul attribut ou construite à partir de plusieurs attributs. Chaque méthode de regroupement définit sa propre clé. Les enregistrements sont triés selon la clé. Ils sont comparés entre eux au sein d'un même bloc. La complexité est de l'ordre de O(n^2/b) pour n enregistrements et b blocs. Chaque bloc est de taille n/b en moyenne.

D'autres travaux proposent ...

Outils d'élimination des doublons et similaires Différents outils existent dans la littérature telle que (Elmagarmid et al., 2007) :

SERF (Stanford Entity Resolution Framework) (Benjalloun et al., 2009) : les auteurs considèrent les fonctions Match et Merge sous forme d'une boite noire. Différents algorithmes sont proposés afin de réduire le nombre d'appel aux boites noires en gardant trace de la dernière comparaison. Différents comparateurs sont combinés dans une disjonction des règles de correspondance.

FEBRL (Christen, 2008) est un outil open source permettant le rapprochement des enregistrements biomédicaux. Il permet la détection des doublons en utilisant des différentes mesures de distance de similarité telles que Jaro, Levenshtein, Q-gram et favorise les mesures phonétiques pour les noms. Il utilise des approches numériques et celles basées sur règles pour la décision de correspondance.

De même l'outil **TAILOR** (Elfeky et al., 2002) repose sur les mêmes décision que FEBRL (des approches numériques et celles basées sur les règles) . Il utilise cinq

mesures de distance : distance de Hamming, Levenshtein, Jaro, q-gram et soundex. Il fournit aussi des mesures statistiques telle que l'exactitude et la complétude afin d'évaluer la qualité des données étudiées.

WHIRL (Elmagarmid et al., 2007) est aussi un outil open source. Il combine les mesures cosine similarity avec TF-IDF afin de calculer la similarité entre deux listes. Toute valeur d'attribut est décomposée en un ensemble de sous chaînes. L'approche consiste à calculer une distance de similarité entre les sous chaînes. Calculer après une moyenne de ces distances et enfin comparer cette moyenne à un seuil de réconciliation.

Les différents outils existants traitent la problématique de dédoublonnage sans tenir compte de la sémantique des données.

Le choix des attributs clés, des algorithmes de similarité ainsi que les seuils de correspondance, repose sur l'expérience utilisateur. Celui-ci n'est pas assisté dans sa démarche.

2.4 Conclusion

Actuellement, un grand nombre d'applications utilisent des données hétérogènes et distribuées de qualité variable. Le besoin de comprendre ces données (donner un sens sémantique), de les intégrer et d'évaluer leur qualité se fait de plus en plus ressentir.

Nous avons essayer tout au long de ce chapitre de lister les différentes anomalies, liées au schéma d'une source ou aux données elles-mêmes. Lister encore les causes de ces anomalies, quelques approches de détection qui restent trop manuelle et demande une certaine expertise utilisateur. Cette dernière demande un minimum de sémantique dans la source.

Enfin, nous avons présenté les travaux cités dans la littérature pour le traitement et la correction de ces données. De même, la présence de la sémantique permettra une meilleure qualité dans le nettoyage de données telle que l'homogénéisation, l'unification et la recommandation des attributs clés, des mesures de similarité ainsi que les seuils d'acceptation.

Afin de redonner d'avantage de sens dans les données rassemblées, nous abordons dans un premier temps la problématique de la reconnaissance du schéma de données. Ensuite, nous abordons la problématique de nettoyage de données en traitant l'homogénéisation des données et en particulier l'élimination des doublons et similaires.

Nous allons dans notre étude ajouter de la sémantique sur les différentes étapes du processus de dédoublonnage. Les fonctions *Match* et *Merge* seront définies en fonction de la sémantique (nature, type et langue) des attributs de dédoublonnage.

L'évaluation du taux de dédoublonnage sera mesurée sur des gros volumes de données (BigData).

Première partie

Reconnaissance sémantique des schémas des données

Introduction

La qualité des données et les aspects sémantiques sont rarement couplés dans la littérature (Becker et al., 2008), (Madnick and Zhu, 2005), (X. Wang and Bither, 2005). Notre défi est donc d'utiliser la sémantique pour améliorer la qualité des données (figure 2.6) (Ben-Salem, 2012). En effet, l'incompréhension du schéma de données est un obstacle pour définir une bonne stratégie pour corriger les anomalies dans les données. Très souvent, les métadonnées si elles existent, ne sont pas assez nombreuses pour comprendre la signification des données et encore moins pour les corriger.

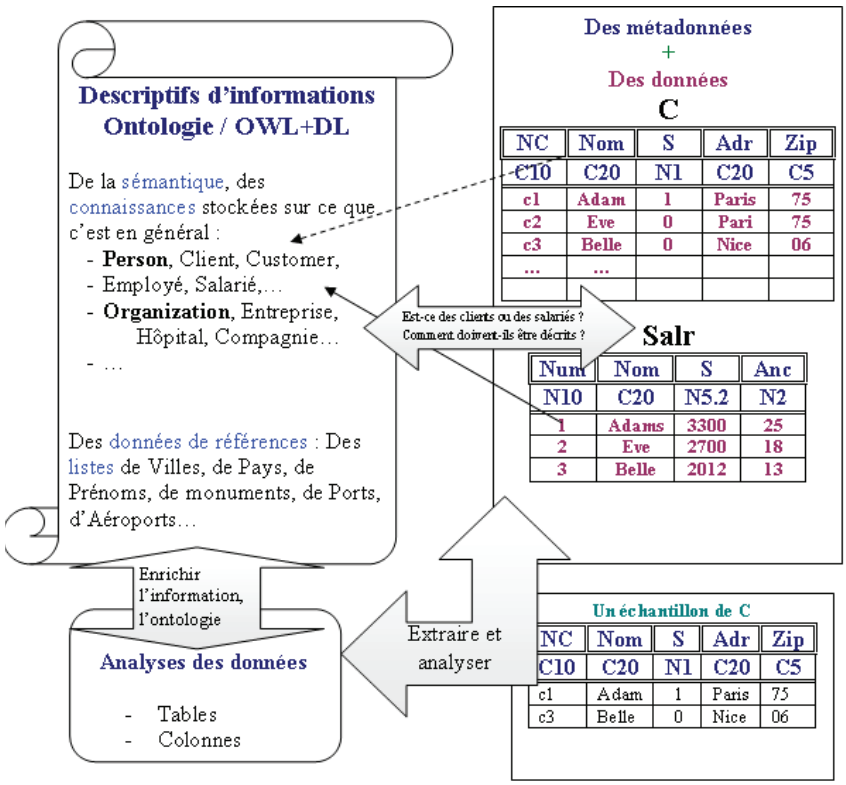


FIGURE 2.6 – Approche sémantique pour la gestion de la qualité des données (Boufarès, 2012)

De nos jours, il n'existe pas d'outils (Köpcke and Rahm, 2009), (Bilenko and Mooney, 2003), (Koudas et al., 2006), (Cohen and Richman, 2004) qui peuvent rapprocher les chaînes de caractères "Pékin" à "Beijing" ou même "Londres" à "London". Des informations sémantiques supplémentaires sont nécessaires pour savoir que ces chaînes représentent la même catégorie et sous-catégorie. De même, il est important de reconnaître la signification sémantique de la chaîne "61°F" qui est une température de type numérique et ne pas la considérer simplement comme une chaîne de caractères.

Nous proposons alors une approche pour la reconnaissance sémantique des schémas de données pour une meilleure compréhension de la définition des données et afin d'améliorer la détection et la correction des anomalies. L'originalité de notre approche réside dans le fait de découvrir non seulement les anomalies dans chacune des colonnes mais aussi celles entre colonnes.

Notre objectif dans cette partie est de définir un schéma sémantique d'une source de données qu'elle ait ou pas une structure de données.

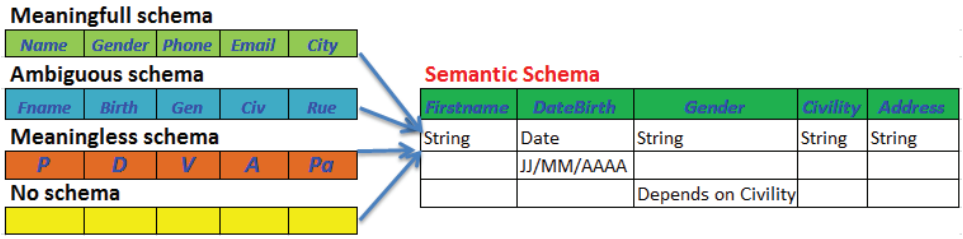


FIGURE 2.7 – Objectif de l'approche sémantique

Cette reconnaissance consiste à utiliser dans un premier temps la sémantique existante dans les données avec un profilage sémantique de ces données (étape 1 de la figure 2.8). Fournir une nouvelle structure sémantique et un ensemble de rapports d'anomalies. On reconnaît alors la sémantique de chaque colonne.

Dans un deuxième temps, nous proposons une reconnaissance des relations inter colonnes ainsi que du concept décrit par ce schéma sémantique (étape 1 de la figure 2.8). C'est l'étape de l'alignement sémantique.

Une fois les métadonnées reconnues, on recommande des analyses spécifiques aux données en utilisant le résultat de l'alignement (étape 2 de la figure 2.8). D'un autre côté, les résultats d'analyse pourront être utiles pour enrichir les données avec de nouvelles connaissances comme le domaine de définition par exemple. L'enrichissement sémantique concerne les données et les métadonnées (étape 3 de la figure 2.8).

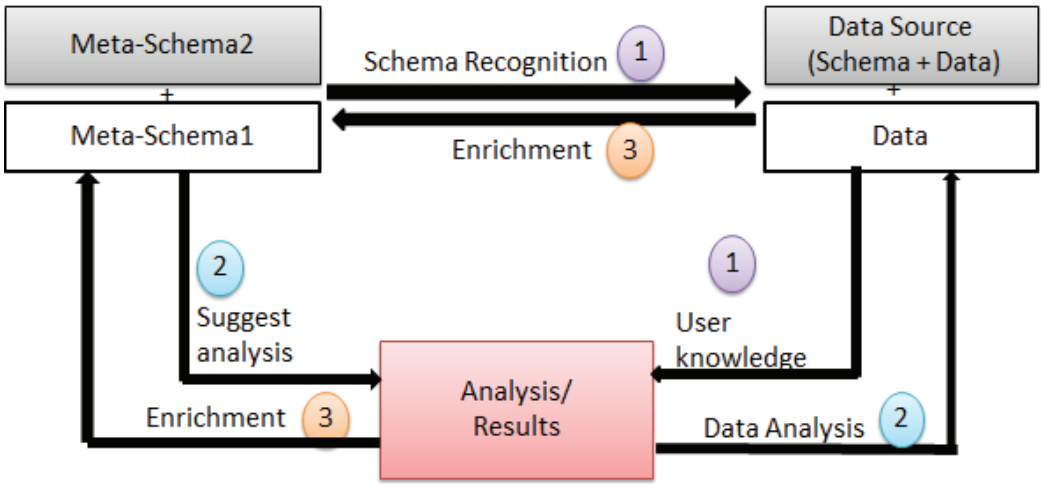


FIGURE 2.8 – Reconnaissance sémantique du schéma des données

Dans cette première partie du manuscrit, nous allons détailler chacune des ces trois étapes : le profilage sémantique de données et l'alignement sémantique des schémas avec des référentiels d'un côté et d'autre côté la recommandation des analyses et l'enrichissement des métadonnées.

Chapitre 3

Profilage des attributs

Sommaire

3.1 Introduction

Rappelons que notre objectif dans cette partie est de reconnaître la sémantique du schéma d'une source de données (figure 3.1). Cette reconnaissance se fait en deux étapes : la reconnaissance des attributs un à un et la reconnaissance du concept décrit par ses attributs ainsi que les relations entre eux.

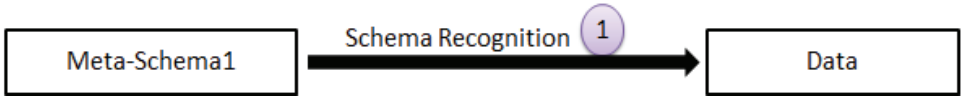


FIGURE 3.1 – Reconnaissance des métadonnées en exploitant la sémantique des données

La première approche consiste à proposer un profilage sémantique des données. Cette étape exploite la sémantique existante dans les données afin de reconnaître la nature, le type de données ainsi que la langue de chaque colonne et les anomalies existantes dans chacune d'elles.

Exemple 3.1 : Source de données sans schéma

On reprend la source de données S, résultat d'intégration des deux sources de données, présentées dans l'état de l'art (table 2.2). La source de données S n'a pas de schéma défini et contient des incohérences dans ses données.

TABLE 3.1 – Un échantillon de la source de données S (fichier CSV)

S
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; Londres ; Royaume-Uni ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; A. ; 0653545555 ; - ; M ; M. ; London ; Royaume-Uni ; - ; 2000 ; 31/03/2012 ;0123435432
Londres ;- ; - ; - ; F ; Mme ; Londres ; Royaume-Uni ; - ; 1000 ; 1-3-2002 ; 0411223344
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; - ; Mme ; Epinay \ seine ; France ; - ; 02 mars 2014 ; 0120000022
Adamsss ; LeBon ; 0653545555 ; - ; - ; - ; - ; - ;- ; 31/3/2012 ;-
Lui ; Clémence ; 0607080911 ; clui@gmail.com ; F ; - ; Epinay sur seine ; France ; - ;2/3/2014 ;0120000022
Saint ; R. ; 0708091122 ; www.saint.fr ; M ; M. ; Epinay Villetaneuse ; Frence ; - ;3000 ;-
Tunsi ; Rahma ; - ;- ; - ; Mme ; Epinay sur seine ; France ; - ;1000 ; 31-11-2014 ; 0965321876
Riche ; Emir ; - ;- ; - ; M. ; Qatar ;- ; - ;200000000 ; 11/2/1955 ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; F ; Mme ; Pékin ; Chine ; - ;1000 ; 30-2-2014 ; 0123987654
LeBon ; Adem ; - ; ada@obsolete.uk ; M ; Londre ; - ; - ;- ;-
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; - ;1000 ; 31/03/2012 ; 0123435433
LeBon ; Adam ; 0653545577 ; adam@yahoo.fr ; M ; Mr ; London ; United-Kingdom ; Europe ;1000 ; 31/03/2012 ; www.lebon.fr
LeBon ; A. ; 0653545555 ; - ; Male ; Mr ; London ; United-Kingdom ; Europe ;2000 ; 31/03/2012
London ; - ; 0766554425 ; - ; M ; Mr ; London ; United-Kingdom ; - ; - ; 11-12-1998
LeBon ; - ; 0653545577 ; adam@yahoo.fr ; 1 ; M. ; London ; UK ; - ;- ;-
Traifor ; Eve ; 0666622223 ; traifor@up13.fr ; Female ; Mrs ; Beijing ; China ; Asia ; 1000 ; 02/29/2014 ; 0123987654
Lebon ; Adel ; 0653545599 ; alebon@up13.fr ; M ;- ; Paris ; France ; Europe ; - ;45
Paris ; H. ; 0607080911 ; paris@live.com ; 0 ; Mrs ; LA ; USA ; America ;10000 ; 23-10-1992 ; 0987654321
Correia ; Sebastiao ; - ; scorreia@talend.com ; M ; - ; Suresnes ; - ; - ;- ; 9/30/2007 ; -

Notons que la sixième colonne (Column6) ne doit contenir que des villes sous leurs noms anglais. "London" et "Beijing" sont syntaxiquement et sémantiquement valides. Alors que, "Pékin" et "Londres" sont syntaxiquement corrects et sémantiquement invalides en supposant que la langue (sous-catégorie) dominante est l'anglais. "Londre" est syntaxiquement invalide.

La colonne dix (Column10) contient principalement des dates dont le format est à homogénéiser. Par conséquent, la valeur de “45” sera considérée comme sémantiquement invalide.

Aussi la quatrième colonne contient en même temps des mails et des adresses Web : la sémantique doit être précisée. D’où, le besoin de plus de sémantique pour comprendre et corriger les données.

Nous proposons un profilage sémantique des données différent du profilage défini dans l’état de l’art (section 2.3.2). Ce profilage sémantique nous permettra une meilleure compréhension de la définition des données (nature des données, type, langue) (figure 3.2) afin d’améliorer la détection et la correction des anomalies (partie 2 du manuscrit).

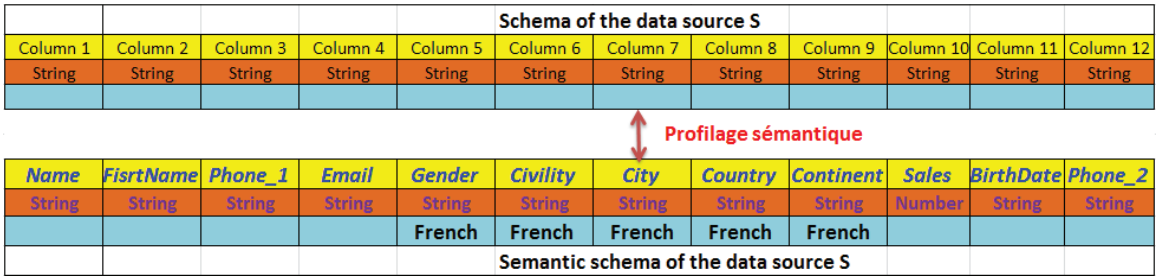


FIGURE 3.2 – Objectif du profilage des attributs : Nouvelle structure sémantique

TABLE 3.2 – Un échantillon de la source de données S (vue tabulaire)

Column1	Column2	Column3	Column4	Column5	Column6
String	String	String	String	String	String
LeBon	Adam	0653545577	adam@yahoo.fr	M	Mr
LeBon	A.	0653545555	-	M	M.
Londres	-	-	-	F	Mme
Lui	Clémence	0607080911	clui@gmail.com	-	Mme
Adamsss	LeBon	0653545555	-	-	-
Lui	Clémence	0607080911	clui@gmail.com	F	-
Saint	R.	0708091122	www.saint.fr	M	M.
Tunsi	Rahma	-	-	-	Mme
Riche	Emir	-	-	-	M.
Traifor	Eve	0666622223	traifor@up13.fr	F	Mme
LeBon	Adem	-	ada@obsolete.uk	M	-
LeBon	Adam	0653545577	adam@yahoo.fr	M	Mr
LeBon	Adam	0653545577	adam@yahoo.fr	M	Mr
LeBon	A.	0653545555	-	Male	Mr
London	-	0766554425	-	M	Mr
LeBon	-	0653545577	adam@yahoo.fr	1	M.
Traifor	Eve	0666622223	traifor@up13.fr	Female	Mrs
Lebon	Adel	0653545599	alebon@up13.fr	M	-
Paris	H.	0607080911	paris@live.com	0	Mrs
Correia	Sebastiao	-	scorreia@talend.com	M	-

Column7	Column8	Column9	Column10	Column11	Column12
String	String	String	String	String	String
Londres	Royaume-Uni	-	1000	31/03/2012	0123435433
London	Royaume-Uni	-	2000	31/03/2012	0123435432
Londres	Royaume-Uni	-	1000	1-3-2002	0411223344
Epinay \ seine	France	-	-	02 mars 2014	0120000022
-	-	-	-	2/3/2014	
Epinay sur seine	France	-	-	01-12-2000	0120000022
Epinay Villetaneuse	Frence	-	3000	-	
Epinay sur seine	France	-	1000	31-11-2014	0965321876
Qatar	-	-	2000000	11/2/1955	
Pékin	Chine	-	1000	30-2-2014	0123987654
Londre	-	-	-	-	
London	United-Kingdom	-	1000	31/03/2012	0123435433
London	United-Kingdom	Europe	1000	31/03/2012	
London	United-Kingdom	Europe	2000	31/3/2012	
London	United-Kingdom	-	-	11-12-1998	
London	UK	-	-	-	
Beijing	China	Asia	1000	02/29/2014	0123987654
Paris	France	Europe	-	45	
LA	USA	America	10000	23-10-1992	0987654321
Suresnes	-	-	-	9/30/2007	-

3.2 Présentation du processus du profilage sémantique

Le profilage prend en entrée à la fois une source de données avec ou sans schéma et un méta-schéma, et renvoie une nouvelle structure sémantique à cette source et des rapports d'anomalies. L'objectif est de proposer un nom sémantique (une catégorie) à chaque colonne, des sous catégories et un type de données (Ben-Salem et al., 2014).

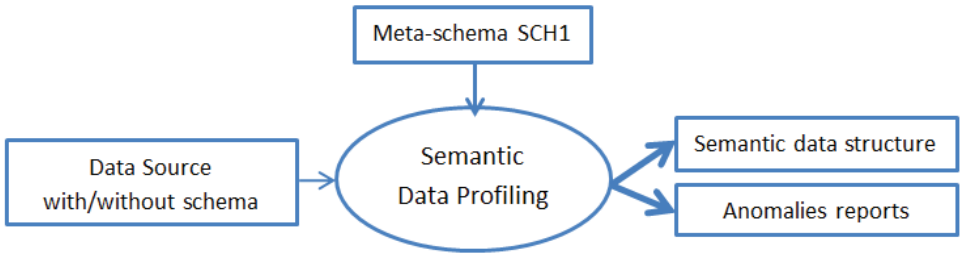


FIGURE 3.3 – Principe du profilage sémantique des données

Nous définissons le profilage sémantique comme étant une analyse qui permet de catégoriser les données d'une source de données, c'est-à-dire donner un sens, une catégorie et éventuellement une sous catégorie à chaque valeur.

Catégoriser est l'organisation d'un ensemble de données en groupes contrastés (Cornuéjols, 2014). Comprimer les données en y découvrant une structure afin de comprendre les données et recommander des actions sur les données telles que des corrections ou des mises à jour.

La catégorisation consiste à attribuer une catégorie aux données à partir d'un ensemble prédéfini de catégories (Forest and Meunier, 2004), (Cornuéjols, 2014). Quelques exemples de données avec leurs catégories sont présentés dans la table (3.3) . Une donnée peut appartenir à une ou plusieurs catégories. Par exemple, la valeur "France" appartient à deux catégories différentes : *Country* et *FirstName*.

TABLE 3.3 – Catégorisation de données

Donnée	Catégorie
Beijing Pékin Paris	City
France Tunisie	Country
France Adam Eve	FirstName
17°C	Temperature

Par abus de langage, nous allons utiliser le mot "source de données" pour désigner dans un premier temps seulement un échantillon de données afin de réaliser le profilage et dans un deuxième temps la source globale pour réaliser les mises à jour.

On définit une catégorie de données et une sous catégorie avec :

Définition 3.1 : Catégorie des données
Soit Cat l'ensemble des catégories : $Cat = \{Cat_i, i = 1; n\}$ avec Cat_i peut être : {FirstName, Country, City, Hospital, University, Animal, Month, Civility, Gender, Email, WebSite, Phone, Address}. avec n est le nombre de catégories. ■

Définition 3.2 : Sous-Catégories
Pour chaque catégorie Cat_i, un ensemble de sous-catégories $SubCat_i = \{SubCat_{ij}, i = 1; n, j = 1; m\}$ peut être défini. Notons que m est le nombre de sous-catégories.

SubCat peut être la langue pour certaines catégories telles que City, Country, Continent. Elle peut aussi être le *Gender* pour la catégorie FirstName : masculin, féminin ou mixte. ■

Exemple 3.2 : Catégories avec la sous-catégorie "Langue"

$[[Cat]] = \{Continent, Country, City, Civility, Gender, Date, Phone\}$.
$[[City/Languages]] = \{English, French, German, Italian, Portuguese, Spanish\}$.
■

Commençons tout d'abord par présenter quelques définitions **sémantiques** utiles lors du processus de profilage.

3.2.1 Définitions sémantiques

Rappelons que le symbole ≈ signifie une recherche approximative entre les données.

La recherche approximative entre les données est calculée en utilisant les mesures de distance de similarité telles que Jaro-Winkler, Levenshtein présentées dans la section 2.3.3 de l'état de l'art.

Définition 3.4 : Validité syntaxique d'une valeur v
Une valeur v est syntaxiquement valide si et seulement si (ssi) :

- $v \in$ RE signifie que v vérifie au moins une expression régulière,
  ou
- $v \in$ KW signifie que v existe dans le dictionnaire de mots clés (recherche exacte)
  ou
- $v \approx w \in$ DD signifie que dans le dictionnaire de données, il existe la valeur v ou une valeur w similaire à v en fonction d'un algorithme de similarité et un seuil ε . ■

Définition 3.5 : Catégorie dominante
Les valeurs d'une colonne C_i peuvent appartenir à une ou plusieurs catégories Cat_j. Soit α_{ij} le nombre de valeurs syntaxiquement correctes pour une colonne C_i et appartenant à une catégorie Cat_j.
Une catégorie Cat_j est dominante pour une colonne C_i si et seulement $\alpha_{ij} > \alpha_{ij'} \forall j \neq j'$.
Le nombre de catégories détecté est défini par l'indicateur "nombre de catégorie ($Isem_1$)". ■

Définition 3.6 : Validité sémantique d'une valeur v
Une valeur v, syntaxiquement valide, est sémantiquement valide si et seulement $v \in Cat_i$, celle-ci est la catégorie dominante. ■

3.2.2 Les éléments du profilage sémantique

Le profilage sémantique des données repose sur un ensemble de méta-données, appelé méta-schéma (SCH1). Il produit en sortie un ensemble de rapports dont la nouvelle structure sémantique.

3.2.2.1 Méta-schéma (SCH1)

Le méta-schéma SCH1 contient un ensemble d'expressions régulières, un dictionnaire de mots clés, un dictionnaire de données et un ensemble d'indicateurs

(figure 3.4). Il produit en sortie r rapports d'anomalies, des rapports de mises à jour et une nouvelle structure sémantique.

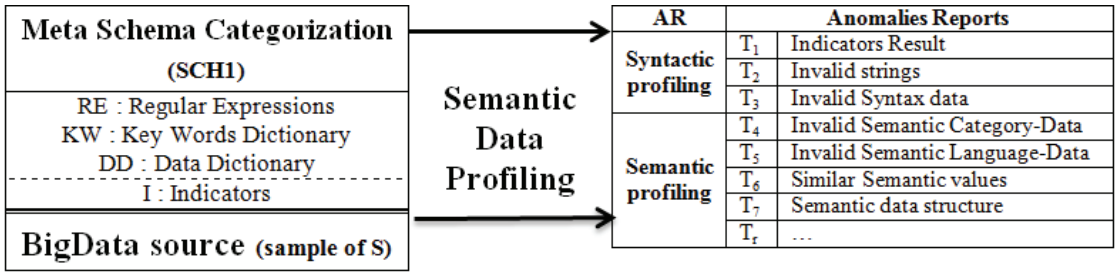


FIGURE 3.4 – Processus du profilage sémantique des données

L'ensemble des expressions régulières (RE) est défini en fonction d'une catégorie, une sous-catégorie et une expression : un triplet {Catégorie, Sous-Catégorie, Expression}

De même le dictionnaire de données (DD) ainsi que le dictionnaire de mots clés (KW) sont aussi définis par un triplet : {Catégorie, Information, Sous-Catégorie}. Ils contiennent des chaînes de caractères valides.

Les indicateurs sont définis en fonction : type de données, nom d'indicateur, description de l'indicateur ainsi que sa requête. On propose des indicateurs statistiques, syntaxiques et sémantiques.

3.2.2.2 Rapports d'anomalies

Les rapports d'anomalies fournissent une description des diverses anomalies existantes dans la source de données telles que la présence de plus d'une catégorie et de plus d'une langue pour la même colonne, différents formats de données, des doublons, des valeurs nulles, des valeurs aberrantes.

Nous définissons r rapports. La structure de ces différents rapports est présentée ci-après :

La première structure contient les résultats des indicateurs. Cette structure est nommée T_1 (Result Indicators). Les indicateurs sont appliqués à chaque colonne de la source et renvoient différents résultats.

Les valeurs mal orthographiées sont automatiquement ajoutées à la structure des valeurs invalides chaînes de caractères, appelée T_2 (Invalid strings). Cette vérification permet d'optimiser la recherche dans le dictionnaire de données. Ce dernier ne contient que des valeurs valides.

La troisième structure T_3 (Invalid Syntax data) contient les valeurs qui ne vérifient aucune expression régulière et n'appartiennent ni au KW ni au DD. On les assigne à une catégorie inconnue (Column1 category). Ces valeurs seront candidates pour l'enrichissement du dictionnaire de données.

Les valeurs qui ne vérifient pas la catégorie dominante sont considérées et stockées dans la table T_4 (Invalid Semantic Category-data) comme étant des valeurs invalides sémantiquement.

Le même processus, utilisé pour détecter la catégorie dominante de chaque colonne d'une source, est appliqué pour reconnaître les sous catégories dominantes. Par exemple pour la sous-catégorie langue, plus d'une langue différente peuvent exister dans une colonne. Nous définissons une langue par colonne, appelée langue dominante. T_5 (Invalid Semantic Language-data) est utilisée pour stocker les valeurs invalides sémantiquement par rapport à la langue. En fin de processus, une langue dominante dans toute la source de données est proposée.

Les données, retrouvées dans le DD en appliquant une recherche approximative, sont stockées dans T_6 (Similar Semantic Values).

La nouvelle structure sémantique, déduite à partir du profilage sémantique des données, est présentée dans T_7.

$T_8..T_r$ pourrait être tout autre type de rapport tels qu'un rapport sur les contraintes ou les dépendances fonctionnelles. Par exemple, en découvrant le type des données pour chaque colonne, les dépendances entre une colonne de type date et une colonne de type string ou encore entre des numériques et des chaînes sont difficilement envisageable.

Le méta-schéma SCH1 permet de sauvegarder des connaissances concernant les données. Il permet de déduire la catégorie, le type de données et la ou les sous-catégories des données.

3.3 Méta-schéma SCH1 (Méta de catégorisation)

Le méta-schéma SCH1 représente l'ensemble prédéfini des catégories (diagramme de classes UML [1] figure 3.5). Il contient un ensemble de connaissances défini par intention avec un ensemble d'expressions régulières et par extension avec un dictionnaire de mots clés et un dictionnaire de données. Un ensemble d'indicateurs (statistiques, syntaxiques et sémantiques) est aussi défini.

1. http ://www.uml.org/

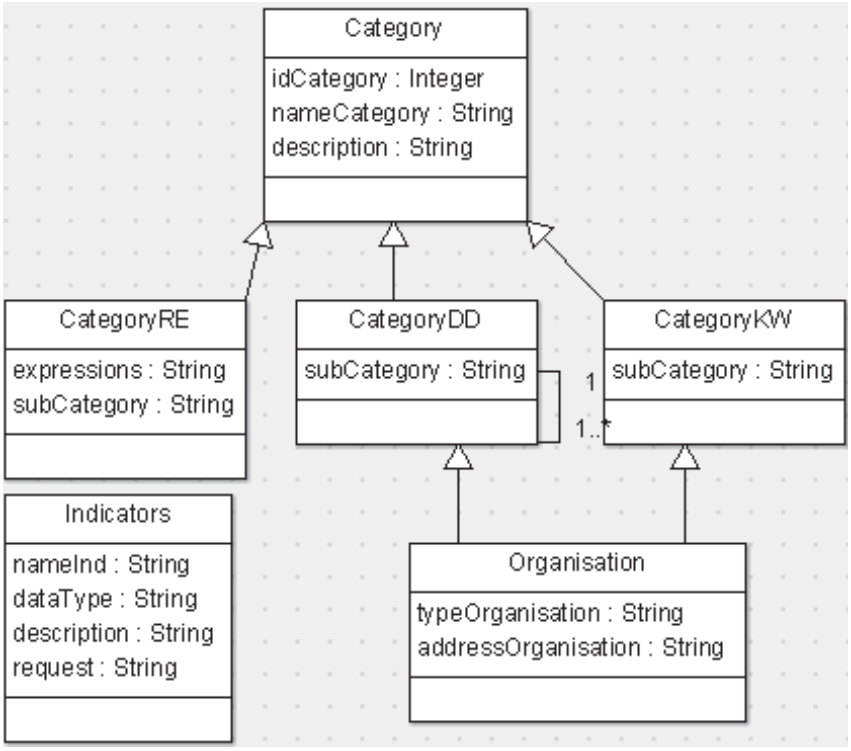


FIGURE 3.5 – Diagramme de classes UML du méta-schéma SCH1 de catégorisation des données

3.3.1 Expressions régulières

Les expressions régulières (RE) sont utilisées pour vérifier la syntaxe des chaînes de caractères et déduire leur catégorie sémantique.

Définition 3.7 : Expressions régulières
Les expressions régulières (RE) peuvent être définies comme un ensemble de triplets : RE= $\{CatRE_i$ / $CatRE_i$ $(Cat_i, SubCat_{ik}, Regex_{ij})$; i=1 ;n, j=1 ;re, k=1 ;m$\}$, avec n le nombre de catégories, re le nombre d'expressions régulières par catégorie et m le nombre de sous catégories. ■

TABLE 3.4 – Exemple d'expressions régulières

idCat	Catégorie	Sous-Catégorie	Expressions régulières
$CatRE_1$	Email		$regex_{11} = \hat{}[a-zA-Z0-9.\_\%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]\{2,4\}\$$
$CatRE_2$	Zip_Code	French	$\hat{}(0[1-9]\|[1-9][0-9])[0-9]\{3\}--\$$
		English	$\hat{}[0-9]\{5\}-\$$
$CatRE_3$	Integer		$\hat{}[[:digit:]]*\$$
$CatRE_4$	Temperature	Celsius	$\hat{}(-?[0-9]\backslash d^{\star}(.\backslash d+)?)?(°C)\$$
		Fahrenheit	$\hat{}(-?[0-9]\backslash d^{\star}(.\backslash d+)?)?(°F)\$$
$CatRE_5$	Date	French	$\hat{}[0-3][0-9](-\mid/\mid\mid.)([0-0][1-9] \mid 10 \mid 11 \mid 12)(-\mid\diagup\mid\mid.)(19 \mid 20)[0-9]2\$$
		English	$\hat{}([0-0][1-9] \mid 10 \mid 11 \mid 12)(-\mid\diagup\mid\mid.)[0-3][0-9](-\mid\diagup\mid\mid.)(19 \mid 20)[0-9]2\$$
$CatRE_6$	WebSite		$\hat{}((http \mid https \mid ftp):\backslash\diagup\backslash\diagup(www\backslash.)? \mid www\backslash.)[a-zA-Z0-9\backslash\_\backslash-]+\backslash.([a-zA-Z]\{2,4\} \mid [a-zA-Z]\{2\}\backslash.[a-zA-Z]\{2\})\$$

Nous définissons des expressions régulières (table 4.11) pour différent types de données : chaîne de caractères, numérique et date.

Notons qu'une valeur peut vérifier plus d'une expression. Exemple, la valeur "657454" vérifie l'expression régulière référant la catégorie sémantique "ZipCode" et la catégorie "Integer".

Cependant, cette liste d'expressions régulières est à enrichir avec de nouvelles expressions afin de couvrir le maximum de catégories sémantiques.

Une liste non exhaustive d'expressions régulières utilisées dans notre processus sera présentée dans le chapitre 5 (table 5.1).

3.3.2 Dictionnaire de données

Le dictionnaire de données contient un ensemble de chaînes de caractères valides. Celles-ci peuvent être structurées en sous-ensembles qui correspondent chacun à une catégorie telle que City, Country ou Continent.

Les données qui appartiennent à une catégorie, peuvent être décrites dans plusieurs sous-catégories (langues).

Des liens sémantiques peuvent exister entre les catégories.

Exemple 3.3 : Liens sémantiques entre catégories

Par exemple, un continent peut contenir plusieurs pays. Chaque pays est rattaché à un continent. Chaque pays contient plusieurs villes. Une ville peut contenir des monuments tels que des universités, des hôpitaux, des musées ou des châteaux (diagramme 3.6).

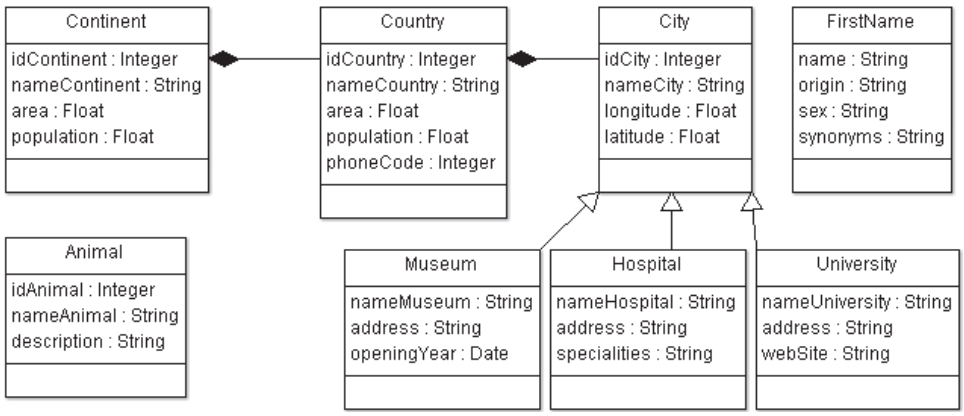


FIGURE 3.6 – Liens sémantiques entre catégories (Diagramme de classes)

Exemple 3.4 : Catégories de données

Nous présentons dans l'exemple suivant un exemple de catégories définies en six langues et avec une clé primaire (PK) et une clé étrangère (FK) (table 3.5). D'autres attributs peuvent être utilisés pour décrire les catégories.

TABLE 3.5 – Catégories des données

Catégorie	Langues						Autres	
	English	French	German	Italian	Portuguese	Spanish	PK	FK
Continent	Europe	Europe	Europa	Europa	Europa	Europa	EUR	
	Africa	Afrique	Afrique	Afrique	Africa	Afrique	AFR	
Country	France	France	Frankreich	Francia	França	Francia	FR	EUR
	Tunisia	Tunisie	Tunesien	Tunisia	Tunísia	Túnez	TN	AFR
City	Paris	Paris	Paris	Paris	Paris	Paris	P1	FR
	Tunis	Tunis	Tunis	Tunisi	Tunis	Túnez	T1	TN
Animal	Dog	Chien	Hund	Cane	Cão	Perro		
	Bear	Ours	Bär	Orso	Urso	Oso		

Définition 3.12 : Dictionnaire de données

Le dictionnaire de données peut être vu comme un ensemble de triplets : CatDD(Catégorie, Information, Sous-Catégorie (Langue)). Information est une chaîne valide (table 3.6).

DD= $\{CatDD_i$ / $CatDD_i$ $(Cat_i, Info_{ij}, SubCat_{ik})$ i=1 ;n, j=1 ;p, k=1 ;6$\}$, avec n le nombre de catégories, p le nombre de valeurs vérifiant une catégorie et six sous-catégories. ■

TABLE 3.6 – Dictionnaire de données

idCat	**Catégorie**	**Information**	**Sous-Catégorie** (Langue)
$CatDD_1$	Continent	$Info_{11}$= Europe $Info_{12}$= Europe	$SubCat_{11}$= English $SubCat_{12}$= French
$CatDD_2$	Country	$Info_{21}$=France $Info_{22}$=France	$SubCat_{21}$= English $SubCat_{22}$=French
$CatDD_3$	City	Paris London Beijing Paris Londres Pékin	English English English French French French
$CatDD_4$	FirstName	Adam Rahma France Marie Paris Aicha	
$CatDD_5$	Civility	Miss Mister Madame Monsieur	English English French French

3.3.3 Dictionnaire de mots clés

Certaines catégories sont découvertes en fonction de mots clés et non pas en utilisant toute la donnée. Une partie de la donnée (un mot par exemple) peut servir à catégoriser les données (table 3.7).

Le dictionnaire de mots clés ainsi que le dictionnaire de données définissent un ensemble de catégories par extension.

Définition 3.8 : Dictionnaire de mots clés
Le dictionnaire de mots clés peut être vu sous forme d'un triplet :
Catkw(Catégorie, Information, Sous-Catégorie (langue)) (table 3.7).
KW= $\{CatKW_i$ / $CatKW_i$ (Cat_i, $Info_{ij}$, $SubCat_{ik}$) i=1 ;n, j=1 ;p, k=1 ;6$\}$, avec n le nombre de catégories et p le nombre de valeurs vérifiant une catégorie. ■

Nous avons choisi comme une première sous-catégorie la langue des données. Nous avons défini k langues différentes. Exemple, pour k=6 : English, French, Ger-

man, Italian, Portuguese et Spanich. La deuxième sous-catégorie peut être toute autre information sémantique utile.

TABLE 3.7 – Dictionnaire de mots clés

idCat	**Catégorie**	**Information** (Langue)	**Sous-Catégorie**
$CatKW_1$	Cat_1=Address	$Info_{11}$=Street	$SubCat_{11}$=English
		$Info_{12}$=St.	$SubCat_{11}$=English
		$Info_{13}$=Avenue	$SubCat_{11}$=English
		$Info_{14}$=Rue	$SubCat_{12}$=French
		$Info_{15}$=R.	$SubCat_{12}$=French
		$Info_{16}$=Avenue	$SubCat_{12}$=French
		$Info_{17}$=Place	$SubCat_{12}$=French
		$Info_{18}$=Pl.	$SubCat_{12}$=French
$CatKW_2$	Study_Organisation	University	English
		School	English
		Universite	French
		Ecole	French
$CatKW_3$	Health_Organisation	Hospital	English
		Hopital	French
$CatKW_4$	Culture_Organisation	Theater	English
		Cinema	English
		Theatre	French
		Cinema	French

Notons que les trois ensembles RE, DD et KW sont disjoints :
RE ∩ KW ∩ DD = ∅. Les catégories dans un ensemble (RE, KW ou DD) n'existent pas dans les autres ensembles.

3.3.4 Indicateurs

Le profilage sémantique des données se base sur un ensemble I de p indicateurs $(I_i, i = 1; p)$, appliqué à la source de données. La plupart des outils existants ne s'intéressent qu'à des résumés quantitatifs sur la source de données. Peu d'outils s'intéressent à l'analyse sémantique. Notre objectif est de proposer des indicateurs sémantiques.

I est composé, d'une part, de trois types d'indicateurs mono-colonnes à savoir q indicateurs statistiques, deux syntaxiques et deux sémantiques.

Indicateurs statistiques $\{Istat_i, i = 1; q\}$ tels que :

- Les indicateurs basiques et appliqués à tout type de données (Istat01..Istat06) :

TABLE 3.8 – Liste des indicateurs basiques

Indicateur	Description	Requête SQL
Istat01	Nom de la source	
Istat02	Nombre total de valeurs	select count(*) from table
Istat03	Nombre de valeurs distinctes	select count(DISTINCT column) from table
Istat04	Nombre de valeurs en doubles	select count(*) from (select column, count(*) from table group by column HAVING count(*)>1)
Istat05	Nombre de valeurs nulles	select count(column) from table where column IS NULL
Istat06	Nombre de valeurs uniques	select count(*) from (select column, count(*) from table group by column HAVING count(*)=1))

- Les indicateurs pour les chaînes de caractères (Istat11..Istat16) :
  — Indicateurs de calcul de longueur d'une chaîne. Ce calcul permettra de reconnaître une certaine sémantique.

TABLE 3.9 – Liste des indicateurs basiques sur les chaînes de caractères

Indicateur	Description	Requête SQL
Istat11	Longueur maximale	select MAX(length(column)) from table
Istat12	Longueur minimale	select MIN(length(column)) from table
Istat13	Longueur moyenne	select AVG(length(column)) from table

  — Table de fréquence des chaînes (Frequency Table) : détermine la liste des valeurs utilisées dans une colonne. Cette liste peut servir comme domaine de définition pour l'attribut analysé. Exemple, l'analyse de la colonne "Grade employée" renvoie trois valeurs : "Employée, Cadre, Chef de projet". Ces valeurs sont utilisées comme domaine de définition de cette colonne.
  — Motif de fréquence de date (Pattern Date Frequency) : indicateur appliqué aux données de type string. Il permet de détecter si la colonne contient des dates ainsi que le format utilisé.
- Les indicateurs pour les numériques (Istat21..Istat24)

  - Minimum et maximum des valeurs numériques : cet indicateur permet de déduire le domaine de définition d'une colonne en fournissant la valeur minimale et la valeur maximale.
  - Moyenne et écart type des numériques.
- Les indicateurs pour les dates (Istat31..Istat33)
  - Indicateur de fréquence de date renvoie les dates utilisées par ordre croisant.
  - Indicateur de fréquence des années renvoie les années existantes dans la source par ordre de fréquence.
  - De même pour l'indicateur de fréquence des mois, qui renvoie les mois existants.

Indicateurs syntaxiques $\{Isyn_i, i = 1; 2\}$ tels que :

- Isyn1 : nombre de valeurs valides syntaxiquement. C'est l'ensemble de valeurs qui vérifient au moins une expression régulière ou existent dans l'un des deux dictionnaires (KW ou DD).
- Isyn2 : nombre de valeurs invalides syntaxiquement. C'est l'ensemble de valeurs qui appartient à la catégorie inconnue. Ces valeurs ne vérifient aucune expression régulière et n'existent ni dans le le KW ni dans le DD.

Indicateurs sémantiques $\{Isem_i, i = 1; 2\}$ tels que : :

- Isem1 : nombre de catégories. Il calcule le nombre de catégories présentes dans une colonne.
- Isem2 : nombre de sous-catégorie. Il calcule le nombre de sous-catégories (langues) utilisées dans une colonne. Cet indicateur facilite le processus d'homogénéisation des données dans un même format et une même langue.

Règles déduites des indicateurs Un ensemble de règles est déduit à partir des différents indicateurs, appliqués à une colonne C_i, présentés ci-dessus (table 3.10). Nous définissons trois types de règles : règles servants pour le choix de la catégorie (respectivement sous-catégorie) dominante (Cat), règles pour les attributs numériques (Num) et règles pour les attributs de dédoublonnage (Key).

TABLE 3.10 – Règles déduites des indicateurs

Type	**Règles**	**Description**
Cat	R1 ($\beta \geqslant 0,7$)	Nombre importants de valeurs valides. Si $\frac{Isyn1}{Istat02-Istat05} \geqslant \beta$ alors la catégorie appartient au DD, KW ou RE.
	R2 ($\beta \geqslant 0,5$)	Nombre importants de valeurs invalides. Si $\frac{Isyn2}{Istat02-Istat05} \geqslant \beta$ lors la détection de la sémantique de la colonne dépend d'autres informations.
	R3 ($\beta \geqslant 0,7$)	Plusieurs catégories. Si $\frac{Isem1}{Isyn1} \geqslant \beta$ alors la détection de la sémantique de la colonne dépend d'autres informations.
	R4	Chaînes de taille fixe.Si $Istat11 = Istat12$ alors la colonne peut appartenir à l'ensemble de catégories : {Date, Téléphone, Numéro de SIRET, Numéro de Sécurité social, ...} .
Num	R5	Domaine de définition des numériques. Les valeurs minimales et maximales des valeurs numériques dans une colonne permettent de définir un domaine de définition pour cette dernière.
Key	R6	Unicité. Si Istat02=Istat03=Istat06 alors la colonne peut être considérée comme une clé primaire.
	R7	Approximativité. Si $Istat02 \approx Istat03$ alors la colonne ne peut pas être considérée comme une clé primaire.
	R8	Pas une clé primaire. Si $Istat03 \neq Istat06$ alors la colonne ne peut pas être considérée comme une clé primaire.
	R9	Ensemble fini de valeurs. Si $Istat03 = Istat04$ alors la colonne est définie sur l'ensemble de valeurs (Select distinct column from table).
	R10	Ensemble fini de valeurs. Si $\frac{Istat03}{Istat02} \approx 0$ alors la colonne est définie sur l'ensemble de valeurs (Select distinct column from table).
	R11 ($\beta \geqslant 0,5$)	Pourcentage de valeurs nulles. Si $\frac{Istat05}{Istat02} \geqslant 0,5$ alors cette colonne n'est pas candidate pour le processus de dédoublonnage. Elle ne peut pas être choisie comme un attribut clé.
	R12 ($\beta \geqslant 3$)	Chaînes de taille petite. Si $Istat11 \geqslant \beta$ ou $Istat13 \geqslant \beta$ alors cette dernière n'est pas recommandée pour un attribut de dédoublonnage.

Après avoir listé les différentes données en entrée pour le profilage de données sémantique, présentons à présent l'algorithme de profilage sémantique.

3.4 Algorithme de profilage sémantique des données

Le méta-schéma SCH1 constitue l'entrée du processus du profilage sémantique des données avec la source de données S en question. Ce processus consiste en trois parties présentées dans la figure (3.7) :

— Analyse préliminaire avec l'application des différents indicateurs statistiques.
— Analyse syntaxique en cherchant le nombre de valeurs valides et invalides syntaxiquement.
— Analyse sémantique afin de déterminer le nombre de catégories et de langues défini dans chaque colonne.

Définition 3.13 : Analyse de données
Une analyse de données est un ensemble d'indicateurs (statistiques, syntaxiques, sémantiques) appliqués aux colonnes d'une source.

FIGURE 3.7 – Processus du profilage sémantique

Le processus du profilage sémantique des données consiste à vérifier si une valeur v appartient à une instance du méta-schéma. Si c'est le cas alors v est définie par une catégorie et peut avoir une sous-catégorie. Le diagramme d'activité suivant (figure 3.8) permet de tracer le déroulement du processus de profilage sémantique.

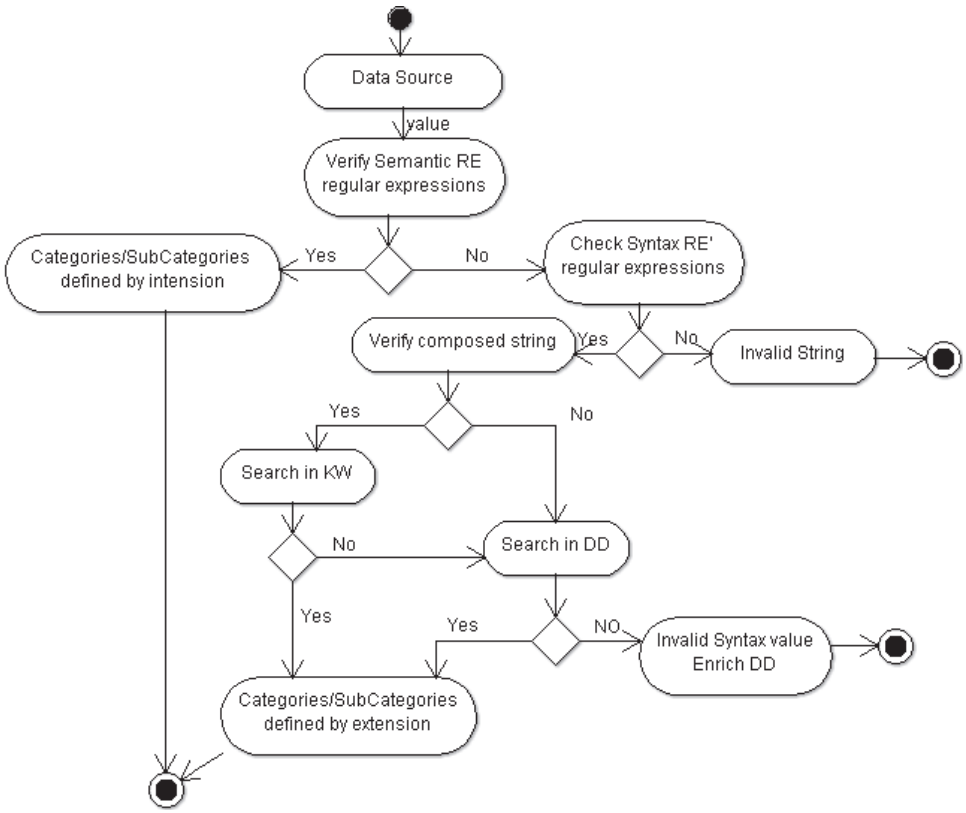


FIGURE 3.8 – Diagramme d'activité du profilage sémantique des données

Nous présentons le processus du profilage sémantique des données sous la forme d'un algorithme avec en entrée : la source de données S et une instance du méta-schéma [[SCH1]]. Les rapports produits par l'algorithme (algo 2) permettent de sauvegarder les résultats des indicateurs statistiques, les valeurs valides et invalides syntaxiquement, les valeurs invalides sémantiquement et la nouvelle structure sémantique pour la source S.

Le principe de l'algorithme (algo 2) se résume en sept points :

1. Créer un échantillon par colonne de la source S sans prendre en compte les valeurs nulles afin de maximiser la probabilité de reconnaître la sémantique de la colonne.
2. Chercher la catégorie (respectivement la sous-catégorie) dominante pour chaque colonne.
3. Refaire les étapes une et deux tant que la nouvelle catégorie dominante est différente de la précédente au maximum h fois.
4. Calculer les indicateurs statistiques qui pourront valider le choix de la catégorie dominante.
5. Remplir la nouvelle structure sémantique T_7 avec les nouvelles informations obtenues pour chaque colonne.

6. Calculer la sous-catégorie dominante pour toute la source.
7. Sauvegarder dans les rapports T_4 et T_5, les valeurs invalides sémantiquement par rapport à la catégorie (respectivement la sous-catégorie) dominante.

Algorithm 2 Algorithm Semantic data Profiling

Input :
S data source
I_{stat} set of Statistical Indicators
nbofIterations (integer)
Output : T_r , r=1 ;7 (Reports)
Begin
Cat_{dom} = null
for each C_i from S (i=1 ;n) **do**
while (Cat_{dom}= ''Unkonown" or h < nbofIterations) **do**
$S'_c \leftarrow$ createSample(C_i) //$S'_c \subseteq$ S. Creates a data sample from column C_i
// Gets the dominant category and subcategory
$\{Cat_{dom}, SubCat_{dom}\} \leftarrow$ semanticCategories(S'_c , C_i)
h++
end while
// Applies Statisticals indicators that can be used with the initial type of the data.
$T_1 \leftarrow$ StatisticalIndicators(S , C_i, I_{stat})
// Return a new semantic data structure
$T_7 \leftarrow$ addCategoryToSemanticStructure(C_i, Cat_{dom}, $SubCat_{dom}$)
end for
if $SubCat_{dom}$ is a language **then**
semanticLanguage(T_7) // Detects the dominant language for the data source S
end if
$T_4, T_5 \leftarrow$ getInvalidSemanticData(S, T_7) //Invalid semantic data relative to the dominant category and sub-Category.
End Algorithm Semantic data Profiling

L'algorithme du profilage sémantique des données est composé de quatre fonctions celles-ci sont détaillées dans les paragraphes ci-dessous.

3.4.1 Création d'échantillon à partir d'une source de données

La première fonction **createSample()** consiste à extraire un échantillon de la source de données (algo 3).

Algorithm 3 Function createSample

```
Input :
    S data source
    C Column from S
    p (size of the sample)
Output : Data source S' (Sample of S)
Begin
j ← 0
while j ≠ p do
  S' ← getValues(j, C, S)
  j++
end while
End function createSample
```

L'échantillonnage est fait en utilisant le principe de "Réservoir"[2]. La méthode *getValues()* choisit aléatoirement p valeurs, différentes de nul, dans la colonne C de la source S.

L'intérêt de cet algorithme est d'avoir un échantillon de taille donnée p sans connaître la taille globale de la source.

3.4.2 Catégorisation des données

Deux fonctions sont appliquées à chaque colonne C_i de l'échantillon S'. La première fonction **semanticCategories** a pour rôle de trouver la catégorie associée à chaque donnée (la valeur v) (algo 4).

La première étape consiste à savoir si la valeur v vérifie au moins une expression régulière.

Le cas échéant la recherche s'effectue dans le dictionnaire de mots clés. La recherche dans le KW consiste à vérifier si la valeur, composée d'au moins de deux mots, existe dans le dictionnaire de mots clés. Une recherche approximative est réalisée avec un seuil ε très faible (une recherche presque exacte).

Définition 3.9 : Recherche approximative $\approx$
$v \in$ KW si et seulement la distance de similarité entre v et une autre valeur v' en utilisant une mesure de similarité est inférieure à un seuil ε : $\mathrm{d}(v, v') < \varepsilon$.

2. http ://en.wikipedia.org/wiki/Reservoir_sampling

Algorithm 4 Function semanticCategories

```
Input :
    C Column from S'
    RE Set of regular expressions, KW Key word dictionary, DD Data dictionary
Output : Cat_dom, SubCat_dom, Reports anomalies
Begin
for each v_j from C (j=1 ;m // m number of records) do
  Cat_j = Null, SubCat_j = Null
  if v_j ∈ RE then
    Cat_j, SubCat_j ← getCategory(v_j, RE) ; Isyn_1++
  else
    //if v_j is a valid string then search in dictionaries.
    if checkValidString(v_j) then
      if v_j is composed then
        //if v_j is composed from more than two words
        if searchInKW(v_j, C) then
          Cat_j, SubCat_j ← getCategory(v_j, KW) ; Isyn_1++
        else
          if searchInDD(v_j, C) then
            Cat_j, SubCat_j ← getCategory(v_j, DD) ; Isyn_1++
          end if
        end if
      else
        if searchInDD(v_j, C) then
          Cat_j, SubCat_j ← getCategory(v_j, DD) ; Isyn_1++
        else
          UnknownCategory(v_j) //v_j belongs to an Unknown Category
          T_3 ← add(v_j, C) //v_j is a candidate to enrich DD and added to T_3 Isyn_2++
        end if
      end if
    else
      // v_j is an invalid string so add v_j to T_2
      T_2 ← add(v_j, C)
    end if
  end if
end for
if Cat_j ≠ Null then
  Isem_1++
end if
if SubCat_j ≠ Null then
  Isem_2++
end if
getIndicatorsResults(Isyn_1, Isyn_2, Isem_1, Isem_1)
Cat_dom ← Max(%Cat_j)
SubCat_dom ← Max(%SubCat_j)
End function semanticCategories
```

Le calcul de la catégorie dominante Cat_{dom} et la sous-catégorie dominante $SubCat_{dom}$ est fait selon la définition (3.5). La catégorie dont le nombre de valeurs valides syntaxiquement est plus élevé est la catégorie dominante. Nous allons justifier formellement ce choix dans la section (3.4.3).

La recherche dans les deux dictionnaires, dans la fonction **semanticCategories**, est effectuée selon deux méthodes : *searchInKW()* et *searchInDD()*.

Les valeurs retrouvées, avec une recherche approximative, dans le DD ou le KW sont ajoutées au rapport T_6 "Similar Semantic Values" pour une éventuelle standardisation par aux dictionnaires.

Algorithm 5 Function searchInKW

```
Input : v a value from C
Output : boolean
Begin
for each word_jk from v (k=1 ;u //u is the number of words in a value) do
  // word_jk exists exactly in the KW
  if word_jk ∈ KW then
    return true
  else if word_jk ≈ w ∈ KW then
    //w a value from KW
    return true
    T_6 ← add(word_jk, C)
  else
    return false
  end if
end for
End function searchInKW
```

Exemple 3.5 : Recherche approximative dans le KW
Pour la recherche approximative, on utilise un des algorithmes de similarité tels que Jaro-Winkler ou Levenshtein (Winkler and Thibaudeau, 1991) avec $\varepsilon = 0.9$. C'est presque une recherche exacte dans le dictionnaire de mots clés. ■

Algorithm 6 Function searchInDD

```
Input : v a value from C
Output : boolean
Begin
// v exists exactly in the DD
if v ∈ DD then
   return true
else if word_jk ≈ w' ∈ KW then
   // w' a value from DD
   return true
   T_6 ← add(v, C)
else
   return false
end if
End function searchInDD
```

Exemple 3.6 : Recherche approximative dans le DD
Pour la recherche approximative, on utilise un des algorithmes de similarité telles que Jaro-Winkler ou Levenshtein (Winkler and Thibaudeau, 1991) avec un seuil égal à 0.8. ■

Dans la mesure où la valeur $v \notin$ RE et $v \notin$ KW, la recherche s'effectue dans le dictionnaire de données DD. On réalise une recherche exacte et rapprochée de la valeur en entrée. Si ces données existent dans le DD, nous assignons une ou plusieurs catégories (respectivement une ou plusieurs sous-catégories) sinon, cette valeur appartient à une catégorie inconnue.

Exemple 3.7 : Valeurs de la catégorie inconnue
La valeur $v =$ *"Epinay Villetaneuse"* est invalide syntaxiquement puisque $v \notin$ KW et $v \notin$ DD. Cette valeur est sauvegardée dans la table T_3 . ■

Les valeurs qui n'existent ni dans le KW ni dans le DD (les valeurs qui appartiennent à la catégorie inconnue) sont ajoutées dans la table T_3 "Invalid Syntax data". Elles sont candidates pour enrichir le DD ou le KW.

Pour chaque colonne, le nombre de valeurs valides syntaxiquement ($Isyn_1$) et le nombre de valeurs invalides syntaxiques ($Isyn_2$) sont calculées.

Aussi les indicateurs sémantiques permettent de calculer le nombre des différentes catégories ($Isem_1$) et le nombre de sous-catégories (langues) ($Isem_2$) détectées au sein d'une même colonne. Le résultat de ces indicateurs est rajouté à la table T_1 "Indicators Result".

Algorithm 7 Function getIndicatorsResults

Input : $Isyn_1$, $Isyn_2$, $Isem_1$, $Isem_1$ (Indicators results)
Output : T_1 (Indicators Result table)
Begin
add($Isyn_1$(C), T_1) // adds the number of valid syntax values to T_1
add($Isyn_2$(C), T_1) // adds the number of invalid syntax values to T_1
add($Isem_1$(C), T_1) // adds the number of used categories to the table T_1
add($Isem_2$(C), T_1) // adds the number of detected subcategories to T_1
End function getIndicatorsResults

Exemple 3.8 : Indicateurs syntaxiques et sémantiques de la source S
La colonne six de la source S contient 78% de valeurs valides syntaxiquement et 11% valeurs invalides. Elle contient aussi deux catégories (*City* et *Country*) et deux langues (table 3.11). ■

Le nombre de valeurs valides et invalides est calculé par rapport au nombre de valeurs non nulles.

TABLE 3.11 – Résultats des indicateurs syntaxiques et sémantiques

ColNum	**Indicator**	**Definition**	**Value(%)**
Column6	$Isyn_1$	Nombre de valeurs valides syntaxiquement	78
Column6	$Isyn_2$	Nombre de valeurs invalides syntaxiquement	11
Column6	$Isem_1$	Nombre de catégories détectées	2
Column6	$Isem_2$	Nombre de sous-catégories	2

Définition 3.10 : Valeur v bien orthographiée
Une valeur v est dite bien orthographiée si et seulement $v \in$ RE', avec RE' est un ensemble d'expressions régulières vérifiant la cohérence des chaînes de caractères (table 3.12).
v est une chaîne qui ne peut pas contenir des caractères spéciaux, ne tolère pas la succession de plus de deux fois la même lettre et ne contient pas que des consonnes.

TABLE 3.12 – L'ensemble des expressions régulières RE'

Nom	Expression régulière	Définition
Alpha_String	$\hat{}([A-Za-z]\{2,\}\|\_\|\backslash-\|)+\$$	Interdire des caractères spéciaux et la succession de plus de deux caractères dans une chaîne (Adamssss et Adam ! ? seront refusés)
	$[zrtypqsdfghjklmwxcvbnZRTYPQ SDFGHJKLMWXCVBN]\{4,\}\$$	Interdire une chaîne construite qu'à partir des consonnes

Définition 3.11 : Valeur v candidate
Une valeur v est dite candidate si et seulement $v \notin$ KW ou $v \notin$ DD mais elle est bien orthographiée. Elle peut être utilisée pour enrichir le DD ou le KW. ■

La table T_3 est extensible dans le temps. Les données sauvegardées serviront pour d'autres analyses futures.

La question qui se pose maintenant est comment choisir théoriquement la catégorie (respectivement sous-catégorie) dominante parmi les différentes **catégories définies par extension** (respectivement sous-catégorie) détectées lors du profilage ?

3.4.3 Choix de la catégorie et la sous-catégorie dominantes

Une colonne de la source de données peut contenir plus d'une catégorie. Notre objectif est de définir une par colonne, appelée catégorie dominante.

Pour chaque colonne, on calcule le nombre de valeurs vérifiant une catégorie. La catégorie dominante est celle vérifiée par le maximum de valeurs. Cependant, nous avons voulu formalisé ce choix et nous proposons de calculer une probabilité pour chaque catégorie en utilisant le naïve bayésienne pour la classification de textes (Naive Bayes (NB) text classification [3]) (Bennani, 2006).

Dans la classification de texte, on cherche à trouver la meilleure classe c pour un document d.

La classification de textes en utilisant NB consiste dans le calcul de la probabilité conditionnelle (formule A.1) :

$$P(c|d) = \frac{P(d|c).P(c)}{p(d)} \tag{3.1}$$

3. http ://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html#laplace

avec d (document) représente une colonne dans notre cas (d=$\{v_i\}$). c (classe) est une catégorie. Les classes doivent être disjointes. Cependant, les valeurs peuvent exister dans plusieurs catégories $v_i \in c_j$ and $v_i \in c'_j$.

Dans notre cas, on cherche à trouver la catégorie dominante pour une colonne. La meilleure classe dans la classification NB est la plus probable ou le maximum a posteriori (MAP) c_{MAP}. Il revient à maximiser les probabilités suivantes :

$$C_{MAP} = \underset{c \in C}{argmax} P(d|c) \cdot P(c)$$

P(c) est la probabilité d'une catégorie par rapport au nombre total de catégories.

$$C_{MAP} = \underset{c \in C}{argmax} P(\{v_1, v_2, ...v_n\}|c) \cdot P(c)$$

or

$$P(\{v_1, v_2, ...v_n\}|c) = p(v_1|c) \cdot p(v_2|c) \cdot p(v_3|c) \cdot ... \cdot p(v_n|c)$$

donc

$$C_{NB} = \underset{c \in C}{argmax} P(c_j) \prod_{v \in V} P(v|c)$$

avec V est l'ensemble de valeurs différentes (vocabulaires) existant dans les différentes catégories, n le nombre de valeurs par colonne (document) et N le nombre de catégories.

Exemple 3.9 : Choix de la catégorie dominante

Soit trois catégories Country, City et FirstName (table 3.13) :

TABLE 3.13 – Ensemble de catégories

Country	**City**	**FirstName**
France	Paris	John
Tunisie	London	Marie
	Tunis	Paris
		Aicha
		Sebastiao
		Faouzi

La colonne dont on cherche la catégorie dominante (table 3.14) :

TABLE 3.14 – Colonne inconnue

Column X
Fred
Paris
John
Julie

$$P(c_j) = \frac{catcount(C = c_j)}{N}$$

Alors pour calculer la probabilité de c_j = Country on aura :

$$P(c_j = Country) = \frac{1}{N} = \frac{1}{3}$$

de même pour

$$P(c_j = City) = \frac{1}{3}$$
$$P(c_j = FirstName) = \frac{1}{3}$$

Pour calculer la probabilité d'appartenance d'une valeur v_i à une catégorie c_j, on utilise la probabilité de maximum de vraisemblance :

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

pour éviter les zéros, on peut utiliser soit la méthode *add-one* ou *Laplace smoothing*[4] est appliqué en ajoutant un 1 (ou bien un paramètre α) aux deux nombres.

$$\hat{P}(w_i|c_j) = \frac{n_k + \alpha}{n + \alpha|V|}$$

avec n_k est le nombre d'occurrences d'une valeur dans la catégorie.

On calcule le nombre des valeurs $v_i \in c_j$, diviser par le nombre de valeurs dans chaque catégorie c_j avec α multiplié par la cardinalité du vocabulaire.

|V| = 10 (nombre de valeurs).

4. http ://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html#laplace

TABLE 3.15 – Calcul de probabilité pour la colonne X

Column XProbabilité	**Country**	**City**	**FirstName**
Fred	$\frac{\alpha}{3+10\alpha}$	$\frac{\alpha}{4+10\alpha}$	$\frac{\alpha}{7+10\alpha}$
Paris	$\frac{1+\alpha}{3+10\alpha}$	$\frac{\alpha}{4+10\alpha}$	$\frac{1+\alpha}{7+10\alpha}$
John	$\frac{\alpha}{3+10\alpha}$	$\frac{\alpha}{4+10\alpha}$	$\frac{1+\alpha}{7+10\alpha}$
Juilie	$\frac{\alpha}{3+10\alpha}$	$\frac{\alpha}{4+10\alpha}$	$\frac{1+\alpha}{7+10\alpha}$

Pour $\alpha = 1$, on aura :

$$P(d|Country) = \frac{1}{3}.\frac{\alpha^4}{(3+10\alpha)^4} = 3,50.10^{-5}$$
$$P(d|City) = \frac{1}{3}.\frac{\alpha^3(\alpha+1)}{(4+10\alpha)^4} = 5,2.10^{-5}$$
$$P(d|FirstName) = \frac{1}{3}.\frac{\alpha(1+\alpha)^3}{(7+10\alpha)^4} = 9,58.10^{-5}$$

La **catégorie dominante** est alors **FirstName**.

Le même processus est appliqué pour la choix de la sous-catégorie dominante pour chaque colonne (annexe A.2).

Schéma sémantique proposé La première partie de l'algorithme spécifie une catégorie (respectivement sous-catégorie) par colonne. On construit alors une structure sémantique pour la source de données avec un nom sémantique pour colonne (SemanticColName), un nouveau type de données (NewDataType) et une sous-catégorie.

Exemple 3.10 : Nouvelle structure sémantique de la source S
On propose dans cet exemple, une structure sémantique pour la source S (table 3.16). Il faut remarquer que S n'avait pas de schéma au départ (un fichier csv).

TABLE 3.16 – "Semantic data structure"

Columns of Initial Schema		Columns of Semantic Schema		
ColNumber	**DateType**	**SemanticColName**	**NewDataType**	**SubCategory**
Column1	String	Column1	String	
Column2	String	FirstName	String	
Column3	String	Phone_1	String	
Column4	String	Email	String	
Column5	String	Gender	String	French
Column6	String	Civility	String	French
Column7	String	City	String	French
Column8	String	Country	String	French
Column9	String	Continent	String	French
Column10	String	Column10	Number	Integer
Column11	String	Column11	Date	French
Column12	String	Phone_2	String	

Remarque : Notons que certaines colonnes restent non reconnues par le processus de profilage telles que Column1, Column10 et Column11 (seulement leur type de données a été reconnu). Nous proposons alors pour les sources avec un schéma, de garder l'ancien nom de la colonne.

Cependant, les règles définies sur ces indicateurs statistiques (table 3.10) permettent de valider certaines catégories reconnues.

Indicateurs statistiques et catégories dominantes Les catégories dominantes sont consolidées par les règles définies dans (table 3.17). Ces règles sont calculées en fonction des résultats des indicateurs statistiques appliqués à la source.

La fonction **statisticalIndicators** (algo 8) consiste à appliquer, sur une colonne, un ensemble d'indicateurs statistiques pour un résumé quantitatif (tels que nombre total de valeurs, nombre de valeurs nulles, motif de fréquence) ou selon le type de données. Par exemple, pour les chaînes de caractères "longueur maximale", "longueur minimale", pour les nombres "la valeur maximale", "la valeur minimale", "la moyenne".

Algorithm 8 Function statisticalIndicators

Input :
C Column from S
I_{stat} Statistical indicators
Output : Statistical Indicators results
Begin
for each I_d from I_{stat} (d=1 ;e //e the number of indicators) **do**
$T_1 \leftarrow$ add(I_d(C)) //Statistical Indicators : total number of values, number of null values...
end for
End function statisticalIndicators

Nous allons nous intéresser aux règles concernant le choix de la catégorie dominante.

- La règle R1 calcule le pourcentage des valeurs valides syntaxiquement par rapport aux valeurs de la colonne.
- R2 est le pourcentage des valeurs invalides syntaxiquement par rapport au nombre total de valeurs de la colonne.
- R3 calcule le pourcentage des catégories détectées par rapport au nombre total de valeurs de la colonne.
- R4 vérifie l'égalité de la taille maximale et minimale d'une chaîne de caractères.

Nous avons défini un algorithme (algo 9) pour la déduction de nouvelles connaissances à partir de ces règles :

Algorithm 9 RulesDeduction

Algorithm RulesDeduction
Input :
Cat Column Categoy
Rules (R1, R2, R3, R4)
Output : Cat_{dom}
calculate(θ_{R1}, θ_{R2}, θ_{R3}, θ_{R4})
if $\theta_{R1} \geqslant 0,5$ and $\theta_{R2} < 0,5$ and $\theta_{R3} < 0,7$ and $\theta_{R4} = True$ **then**
$Cat_{dom} \leftarrow Cat$
end if
END Algorithm RulesDeduction

Exemple 3.11 : Règles et catégories des colonnes de la source S

TABLE 3.17 – Règles et catégories des colonnes de la source S

Columns	Indicators	**Rules**			
Name	Result	R1	R2	R3	R4
Column2 (FirstName)	Istat02=100 Istat03=13 Istat04=13 Istat05=15 Istat06=0 Istat11=9 Istat12=2 Isyn1=65 Isyn2=20 Isem1=2	0,76	0,24	0,03	X
Column3 (Phone)	Istat02=100 Istat03=8 Istat04=8 Istat05=25 Istat06=0 Istat11=10 Istat12=10 Isyn1=75 Isyn2=0 Isem1=3	1	0	0,04	✓
Column4 (Email)	Istat02=100 Istat03=9 Istat04=9 Istat05=35 Istat06=0 Istat11=19 Istat12=12 Isyn1=60 Isyn2=5 Isem1=2	0,92	0,08	0,03	X

Prenons par exemple la colonne 3, le processus de profilage détecte trois catégories : {Phone, Integer, Currency}. Cependant, d'après la règle R5, vu que la taille de la chaîne maximale est égale à la taille minimale alors la colonne contient probablement des numéros de téléphone. De plus la règle R1 est bien vérifiée $\beta_2 = 1$. Les valeurs sont toutes valides syntaxiquement. La catégorie dominante pour la colonne 3 est bien *Phone*.

Colonnes en double Par ailleurs, l'approche permet aussi la détection de colonnes similaires ou en double. Par exemple, les colonnes Phone_1 et Phone_2 représentent des catégories similaires.

Une source des données peut contenir des colonnes "semblables", notée $Col_1 \leq Col_2$.

Dans le cas où $Col_1 = Col_2$, les deux colonnes ont la même catégorie sémantique et le même contenu. Alors l'une d'eux est supprimée afin d'éviter d'avoir des données en doubles.

Nous proposons un algorithme pour la détection des colonnes en doubles (algo 10). Il consiste à comparer les deux colonnes ligne par ligne. Si nous avons les mêmes valeurs alors ces deux colonnes sont égales.

Algorithm 10 DetectEqualColumns

```
Algorithm DetectEqualColumns
Input :
    C1, C2 (Two columns from S)
    n (number of tuples)
Output : equalColumn (boolean)
Begin
equalColumn ← True
i ← 1
while (i ⩽ n and equalColumn = True) do
  v1i ← ti.C1
  v2i ← ti.C2
  if v1i = v2i then
    i++
  else
    equalColumn ← False
  end if
end while
return equalColumn
END DetectEqualColumns
```

Dans le cas où $Col_1 < Col_2$, les deux colonnes ont la même catégorie sémantique, mais quelques différences dans leur contenu (equalColumn ← False). Des noms similaires sont proposés pour les deux colonnes tel est le cas pour la source de données S1 (table 3.16).

Sachant qu'avec notre processus de profilage sémantique, différentes sous-catégories peuvent exister au sein d'une même source. Nous allons dans la suite calculer une seule sous-catégorie pour toute la source de données.

Sous-catégories dominantes pour la source de données Avec la méthode présentée ci-dessus, nous avons pu reconnaître des sous-catégories pour chaque colonne. A la fin de ce processus, on propose, de détecter la sous-catégorie (par exemple langue) dominante pour toute la source de données.

La fonction **semanticLanguage** (algo 11) permet de calculer la dominance d'une langue parmi les langues déduites pour chaque colonne.

Algorithm 11 Function semanticLanguage

Input : T_7 semantic structure
Output : Dominant language
Begin
for each $Language_i$ from T_7 (i=1 ;n) **do**
$n_i \leftarrow$ Count the number of occurrences ($Language_i$)
end for
DominantLanguage $\leftarrow$ LanguagewhereMax(n_i)
update(T_7, DominantLanguage) //updates semantic structure with the new dominant language
End function semanticLanguage

Une mise à jour de la structure sémantique est réalisée à la fin du processus.

Valeurs invalides sémantiquement Une fois la catégorie ainsi que la sous-catégorie dominantes pour toute la source sont reconnues, des rapport sur l'invalidité sémantiquement sont renvoyés (algo 12). Un premier rapport concerne les valeurs invalides sémantiquement par rapport à la catégorie dominante sont stockées dans la table T_4 "Invalid Semantic Category-Data". Un deuxième rapport concerne les valeurs invalides par rapport à la sous-catégorie dominante (table T_5 "Invalid Semantic Language-Data").

Algorithm 12 Function getInvalidSemanticData

Input :
S data source
T_7 semantic data structure
Output :
T_4 Invalid semantic Category-data
T_5 Invalid semantic Language-data
Begin
for each C_i from S (i=1 ;n) **do**
for each v_{ij} from C_i (j=1 ;m) **do**
if $v_{ij} \notin Cat_{domi}$ **then**
$T_4 \leftarrow \text{add}(v_{ij}, C_i)$
end if
if $v_{ij} \notin SubCat_{domi}$ **then**
$T_5 \leftarrow \text{add}(v_{ij}, C_i)$
end if
end for
end for
return T_4, T_5
End function getInvalidSemanticData

Exemple 3.12 : Valeurs invalides sémantiquement par rapport à la catégorie de la source des données S
La valeur "Qatar", existante dans la colonne six de la source S, représente le nom d'un pays et non pas le nom d'une ville.

TABLE 3.18 – "Invalid Semantic Category-Data"

Old Schema			**New Semantic Schema**
ColNum	Value	Category	SemanticColName
Column4	*www.saint.fr*	WebSite	**Email**
Column7	*Qatar*	Country	**City**
Column11	*45*	Numeric	**Date**

"Qatar" n'appartient pas alors à la catégorie dominante *City*. Elle est donc considérée comme étant une valeur invalide sémantiquement (table 3.18).■

On ajoute à la table T_5 "Invalid Semantic Language-Data" les valeurs invalides par rapport à la langue dominante de toute la source de données.

Exemple 3.13 : Valeurs invalides sémantiquement par rapport à la langue de la source des données S

La langue (sous-catégorie) dominante de la source S est le français. La colonne six contient deux langues : l'anglais et le français (d'après l'indicateur "nombre de langues utilisées"). Donc les valeurs en anglais sont toutes des valeurs invalides sémantiquement (table 3.19).

TABLE 3.19 – "Invalid Semantic Language-Data"

Old Schema			**New Semantic Schema**	
ColNum	Value	Language	SemanticColName	DominantLang
Column7	*London*	English	City	**French**
Column7	*Beijing*	English	City	**French**
Column8	*United-Kingdom*	English	Country	**French**
Column8	*China*	English	Country	**French**

Un dernier point lors de la reconnaissance des attributs d'une source est la détection de contraintes pour chaque colonne en utilisant les résultats des indicateurs statistiques.

3.4.4 Contraintes sur les données

Les analyses de données permettent de déduire des contraintes sur les attributs. Ces informations servent à enrichir le schéma sémantique.

A partir des indicateurs statistiques, un ensemble de **contraintes** peut être défini sur les **colonnes**. Citons les types de contraintes utilisés dans ce manuscrit.

- Clé primaire (Primary Key PK) : est une contrainte d'unicité. Elle permet d'identifier de manière unique un enregistrement dans une source.
- Clé étrangère (Foreign key FK) : permet de gérer les relations entre plusieurs colonnes de plusieurs sources.
- Unique (UN) : aucune valeur en double n'est autorisée sur cette colonne.
- Majuscule (Capital Letter CL) : les valeurs dans la colonne doivent être en majuscule.
- Not Null (NN)= mandatory : permet de spécifier qu'un champ est obligatoire.
- Optionnel (Optional OP) : ce champ peut être vide.
- Check (CK) (liste) : une valeur doit appartenir à une liste de valeurs.
- Check (CK) (intervalle) : une valeur numérique appartient à un intervalle de valeurs.
- Dépendance fonctionnelle (Functional dependency DF)

La fonction **statisticalIndicators** (algo 8) renvoie des informations pouvant servir par exemple dans la définition d'un domaine de définition pour une colonne ou une contrainte d'unicité.

Exemple 3.14 : Résultats des indicateurs appliqués à la source S sans schéma
Les indicateurs statistiques sont appliqués à la colonne six (Column6) de l'échantillon de la source S présentée dans l'introduction (table 3.1). La table (3.20) représente un ensemble d'indicateurs basiques et des indicateurs en fonction du type de données.

Remarque : Quand on n'a pas de schéma, on applique à ce stade uniquement les indicateurs basiques et les indicateurs sur les chaînes de caractères vu que la source S (échantillon de la source) est un fichier csv.

TABLE 3.20 – Table "Result Indicators"

ColNum	**DT**	**Indicator**	**Definition**	**Value(%)**
	X	$Istat_1$	Échantillon de la source	S'
Column7	X	$Istat_2$	Nombre total de valeurs	19
Column7	X	$Istat_3$	Nombre de valeurs distinctes	14
Column7	X	$Istat_4$	Nombre de valeurs en doubles	2
Column7	X	$Istat_5$	Nombre de valeurs nulles	1
Column7	C	$Istat_6$	Taille maximale	19
Column7	C	$Istat_7$	Taille minimale	2
Column7	C	$Istat_8$	Taille moyenne	7
Column5	C	$Istat_9$	Table de fréquence des chaines	{F, M}

DT (Data Type) est utilisé pour désigner le type des données analysées.

Exemple 3.15 : Résultats des indicateurs appliqués à la source S avec schéma (type de données reconnu)
Dans ce deuxième cas, on considère que la source contient un schéma avec des données de types différents. On vérifie avec les analyses les contraintes définies si elles existent et on définit des nouvelles.

TABLE 3.21 – Table "Result Indicators"

ColNum	DT	Indicator	Definition	Value(%)
	X	$Istat_1$	Échantillon de la source	S'
Column7	X	$Istat_2$	Nombre total de valeurs	19
Column7	X	$Istat_3$	Nombre de valeurs distinctes	14
Column7	X	$Istat_4$	Nombre de valeurs en doubles	2
Column7	X	$Istat_5$	Nombre de valeurs nulles	0
Column6	C	$Istat_9$	Table de fréquence des chaines	{Mme, M.}
Column10	N	$Istat_{12}$	Valeur minimale	1000
Column10	N	$Istat_{13}$	Valeur maximale	10000
Column11	D	$Istat_{16}$	Haute fréquence des dates	20120331
Column11	D	$Istat_{17}$	Haute fréquence des années	2012
Column11	D	$Istat_{18}$	Haute fréquence des mois	201203

Rappelons que le DT=X est utilisé pour désigner les indicateurs appliqués à tout type de données, DT=C pour les chaînes de caractères, DT=N pour les numériques et DT=D pour les dates.

Ces indicateurs statistiques permettent de détecter les domaines de définition des attributs. Par exemple, pour la colonne neuf (column10), le domaine de définition est l'intervalle [1000..10000]. On peut considérer que toute valeur n'appartenant pas à cet intervalle est une valeur invalide.

Ces informations servent à définir des contraintes explicites sur chaque colonne (table 3.22).

TABLE 3.22 – Schéma sémantique avec des contraintes définies sur les colonnes

ColNum	SemanticColName	NewDataTYPE	SubCategory	Constraint
Column1	Column1	String		
Column2	FirstName	String		
Column3	Phone_1	String		
Column4	Email	String		
Column5	Gender	String	French	CK {F, M}
Column6	Civility	String	French	CK {Mme, M.}
Column7	City	String	French	
Column8	Country	String	French	
Column9	Continent	String	French	
Column10	Column10	Number	Integer	CK [1000..10000]
Column12	Column11	Date	French	JJ/MM/AAAA
Column12	Phone_2	String		

La dernière étape du processus de reconnaissance des colonnes est l'enrichissement du méta-schéma SCH1 (RE, KW et DD).

3.5 Enrichissement des dictionnaires des données (DD, KW)

Lors du profilage, un rapport des valeurs invalides syntaxique est établi (table 3.24). D'après la définition 3.5, ces valeurs ne vérifient aucune expression régulière et n'existent pas (= ou ≈) dans un des deux dictionnaires. Par exemple, la valeur "Londre" dans la colonne 6 de la source S, n'est pas une valeur invalide syntaxiquement puisqu'il existe dans le DD, une valeur très proche qui est "Londres".

Exemple 3.16 : Valeurs invalides syntaxiquement

TABLE 3.23 – "Invalid Syntax Data"

Old Schema		**New Semantic Schema**	
ColNum	Value	SemanticColName	DominantLang
Column2	*Emir*	FirstName	
Column2	*Sebastiao*	FirstName	
Column7	*Epinay Villetaneuse*	City	**French**
Column7	*Suresnes*	City	**French**

La similarité est définie en fonction d'un seuil ε_i. Donc à partir d'un seuil ε_j avec $i \neq j$, on peut considérer que la similarité n'est plus vérifiée. Notons aussi que ces valeurs sont des chaînes valides (vérifient la définition 3.9). Alors, elles peuvent être candidates pour enrichir le DD ainsi que le KW.

TABLE 3.24 – Valeurs candidates pour enrichir le DD

Category	**SubCategory**	**Value**	**UT**
FirstName		*Emir*	5
FirstName		*Sebastiao*	3
City	Language	*Epinay Villetaneuse*	1
City	Language	*Suresnes*	10

avec UT (usedTimes) : est le nombre d'apparition d'une valeur dans la source S.

On calcule $\rho = \frac{UT}{Istat2}$. Si ρ est assez important alors la valeur correspondante est candidate afin d'enrichir le DD.

Si la catégorie de la valeur contient une sous-catégorie alors avant de l'ajouter dans le DD, on essaye de déduire sa sous-catégorie pour un meilleur enrichissement. Pour la sous-catégorie langue, nous avons testé différentes applications java (Labs, 2010) pour la détection la langue d'une chaîne de caractères. Cependant, le résultat n'est pas très performant pour les petites chaînes.

Notons aussi que les utilisateurs peuvent également enrichir les méta-schémas avec de nouvelles expressions régulières.

3.6 Conclusion

Le profilage sémantique des données permet de catégoriser les données en leur attribuant une catégorie et une sous-catégorie. Dans notre cas, nous nous sommes intéressés en particulier à la langue comme étant une sous-catégorie existante dans un ensemble de catégories (Ben-Salem et al., 2014).

Cette catégorisation de données permet de vérifier les dimensions *Interprétabilté, Standardisation et duplication*, présentées dans l'introduction (table1.1).

La sémantique donnée aux structures de données, afin de mieux comprendre leurs contenus, permet de vérifier la dimension *Interprétabilité*.

Le rapport sur les anomalies, existantes dans une source, permet de mettre en oeuvre les données non standards et les doublons.

Nous avons pensé à une étape de consolidation ou de vérification du résultat de profilage. Cette étape consistera à l'application du processus du profilage sémantique sur d'autres échantillons de la source. Soit des échantillon par colonne (comme appliquer auparavant), soit des échantillons sur toutes les colonnes de la source.

Le profilage sémantique nous a permis de découvrir le sens des colonnes. Cependant, il reste à reconnaître les liens existants entre les colonnes. Ces derniers feront l'objet d'une étude approfondie dans le chapitre suivant.

Chapitre 4

Reconnaissance sémantique des relations inter colonnes

Sommaire

4.1 Introduction

Dans le précédent chapitre, nous avons présenté une méthode de reconnaissance sémantique de chaque colonne d'une source de données. Cette approche consistait à exploiter la sémantique existante dans les données de chaque colonne.

La prochaine étape est de reconnaître les liens existants entre ces colonnes ainsi que la sémantique du nom de la source et celle de la source elle-même.

Définition 4.1 : Concept d'une source de données

On définit un *concept* pour désigner le nom d'une source (une table, un fichier). L'ensemble des attributs définit un concept.

Exemple 4.1 : Attributs, concept

Les attributs *FirstName*, *Name*, *DateBirth*, *Gender*, *Civility*, *Address* définissent le concept *Person*.

Ces attributs sont les catégories définies dans le méta-schéma SCH1 (présenté dans le chapitre précédent section 3.3).

Nous allons répondre dans ce chapitre à différentes questions (figure **??**) :

1. Comment reconnaître le concept décrit par le schéma sémantique ainsi que les liens inter colonnes ?
2. Comment recommander de nouvelles connaissances (analyses, domaines de définition) aux sources ainsi qu'à chaque colonne ?
3. Comment enrichir le référentiel avec ces nouvelles connaissances ?

L'alignement du schéma sémantique, résultat du profilage, avec un référentiel (ensemble de connaissances) permettra de répondre à ces trois questions (étape 1 de la figure 2.8). D'abord, l'alignement permet de déterminer le nombre d'attributs du schéma reconnus dans le référentiel. En fonction de ces derniers, on essaye de découvrir le concept décrit par ces attributs.

La recommandation des analyses de colonnes se fait en utilisant les connaissances stockées dans le référentiel concernant les attributs rapprochés (synonymes). Des analyses sur la source sont recommandées en fonction des relations inter colonnes telles que une analyse de dépendance fonctionnelle.

Enfin, le référentiel est enrichi avec des nouveaux concepts et des nouveaux attributs (non reconnus dans le référentiel). Les résultats des analyses de source et des colonnes (les seuils, les domaines de définitions) peuvent être aussi ajoutés au référentiel.

Définissons dans ce qui suit, le méta-schéma SCH2, méta modèle du référentiel (base d'alignement).

4.2 Définition du méta-Schéma SCH2

Une source de données peut être décrite de plusieurs manières différentes. La différence est essentiellement au niveau des noms des concepts, des attributs qui les définissent et leur nombre.

L'idée est de stocker toutes descriptions équivalentes d'une source dans une méta-structure, méta-Schéma ($SCH2$). $SCH2$ contient les noms des sources (concepts), des synonymes à ces noms, des noms des colonnes (attributs) de la source ainsi que leurs synonymes, des contraintes et des commentaires (tags). Le diagramme de classes UML suivant (figure 4.1) définit ce modèle.

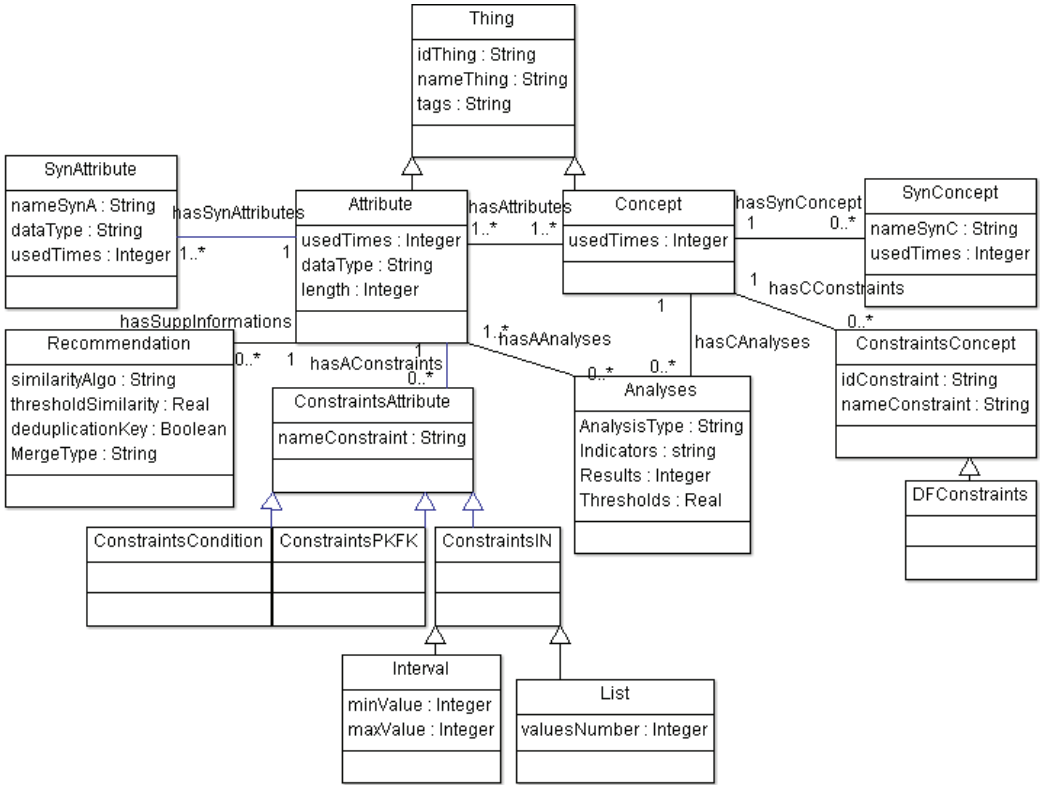


FIGURE 4.1 – Diagramme de classes du méta-Schéma SCH2

Détaillons les différents éléments du diagramme :

Définition 4.2 : Classe *Thing*

Un objet peut être un concept ou un attribut. Il est défini par un identifiant (*idThing*) et un nom (*nameThing*).

Notons aussi qu'un concept ou un attribut peuvent être définis par un ensemble de commentaires (*Tags*). Ces derniers peuvent contenir de la sémantique sur ces connaissances. ■

Définition 4.3 : Classe *Concept*

La classe *Concept* représente les noms des différentes sources de données. soit Ct l'ensemble des concepts.

$Ct = \{Ct_i, i = 1; n\}$ avec n le nombre de concepts. ■

Définition 4.4 : Classe *SynConcept*

Un concept peut avoir un ou plusieurs synonymes. Ces synonymes sont sauvegardés dans la classe *SynConcept*.

On note $SynCt_i$, l'ensemble des synonymes à un concept Ct_i.

$SynCt_i = \{SynCt_{ij}$, j=1 ;m$\}$ avec m le nombre de synonymes concept Ct_i. ■

On définit la synonymie comme un rapport de similarité sémantique entre des objets de même langue ou de langues différentes. La similarité sémantique [1] signifie que ces objets ont des significations très semblables et sont, par conséquent, définis par les mêmes attributs.

Définition 4.5 : Classe *Attribute*

Notons aussi, qu'un concept Ct_i peut être décrit par un ou plusieurs attributs (représentés par la classe *Attribute*). Soit C, l'ensemble des attributs (colonnes) décrivant un concept Ct_i.

C = $\{C_p$, p=1 ;d$\}$ avec d le nombre d'attributs du concept Ct_i. ■

Définition 4.6 : Classe *SynAttribute*

Un attribut C_p peut aussi avoir un ou plusieurs synonymes. La liste des synonymes est stockée dans la classe *SynAttribute*. Soit $SynC_p$, l'ensemble des attributs synonymes à C_p.

$SynC_p$ =$\{$ $SynC_{pk}$, k=1 ;a$\}$ avec a le nombre de synonymes de l'attribut C_p. ■

Remarque : Les ensembles *Concept* et *SynConcept* sont disjoints avec les ensembles *Attribute* et *SynAttribute*.

$(Ct \cup SynCt_i) \cap (C \cup SynC_p) = \emptyset$

Exemple 4.2 : Concept, synonyme concept, attribut et synonyme attribut

Une *organisation* peut être une *entreprise*, une *société* ou une *compagnie*.

Elle peut être définie par un *identifiant*, un *nom*, un *statut*, une *adresse*, un *numéro de téléphone*, un *numéro de fax* et un *site web*.

L'attribut *Nom* peut avoir différents synonymes tels que *Name*, *Nome* (table 4.1).

TABLE 4.1 – Concept, Synonyme concept et attributs

Organisation	**Entreprise**	**Société**	**Compagnie**
Identifier	Identifiant	Numéro	Identifiant
Name	Nom	Nome	Name
Status	Statut	Statut	Status
Address	Adresse	Adresse	Address
Phone	Téléphone	telefono	Telefon
Fax	Fax	Fax	Fax
Website	SiteWeb	SiteWeb	SitoWeb

Définition 4.7 : Classe *ConstraintsConcept*

Un concept Ct_i peut être lié à une ou plusieurs contraintes. On note KCt, l'ensemble

1. http ://fr.wikipedia.org/wiki/Synonymie

des contraintes appliquées à un concept Ct_i.
$KCt_i = \{KCt_{il}$, l=1 ;e$\}$ avec e le nombre de contraintes par concept Ct_i. ■

Exemple 4.3 : Contraintes sur un concept
Le tableau suivant (table 4.2) présente quelques exemples de dépendances sémantiques dans une source S. On peut citer comme dépendance sémantique les dépendances fonctionnelles.

Formellement, une dépendance fonctionnelle sur une source S est définie comme S (X → Y), où X et Y sont des ensembles d'attributs. Une instance [[S]] satisfait la DF si pour tout t1 et t2 deux tuples dans [[S]], t1.Y = t2.Y lorsque t1.X = t2.X, c'est à dire les attributs de X d'un tuple déterminent uniquement ses attributs Y.

TABLE 4.2 – Exemple de liens entre colonnes (DF X → Y)

idConstraint	Concept	X	Y	Satisfiable
KC_{11}	Person	Country	Continent	T
KC_{12}	Person	City	Country	T
KC_{13}	Person	ZipCode	City	T
KC_{14}	Person	Civility	Gender	T

avec T pour dire True (cette dépendance est vérifiable) et F pour dire False.

Définition 4.8 : Classe *ConstraintsAttribute*
Un attribut C_p peut vérifier une ou plusieurs contraintes. Une classe contrainte (*Constraints*) contient différents types de contraintes telles que les contraintes de cardinalité et les contraintes d'intégrité. Soit KC, l'ensemble des contraintes appliquées à un attribut C_p.
$KC_p = \{KC_{ph}, h = 1; f\}$ avec f le nombre de contraintes par attribut C_p. ■

Ces attributs doivent vérifier différentes contraintes (table 4.5, 4.6).

Définition 4.9 : Classe *Analyses*
Un ensemble d'analyses peut être appliqué sur un concept ou sur des attributs. Les analyses permettent d'avoir de nouvelles informations. La classe *Analyses* contient toutes les informations utilisées lors d'une analyse telles que les indicateurs utilisés, les résultats de ces indicateurs ainsi que leurs seuils.
Soit ACt l'ensemble des analyses appliquées à un concept Ct_i. $ACt_i = \{ACt_{it}, t = 1; u\}$ avec u le nombre d'analyses par concept Ct_i.
Soit AC l'ensemble des analyses appliquées à un attribut C_p. $AC_p = \{AC_{pr}, r = 1; o\}$ avec o le nombre d'analyses par attribut C_p. ■

Définition 4.10 : Classe *Recommendation*
Nous avons défini une classe *Recommendation*, pour sauvegarder toutes informa-

tions supplémentaires sur les attributs pour une éventuelle recommandation. Ces informations peuvent être les attributs qui ont servi pour le dédoublonnage " Deduplication keys", les mesures de similarité utilisées, les seuils ainsi que le type de fusion (concaténation ou autre). ■

Exemple 4.4 : Modèle relationnel

On présente (table 4.3, 4.4) le modèle relationnel du méta-schéma SCH2. Différentes informations existent pour chaque classe du modèle. Par exemple, un attribut est défini par son type de données, sa taille maximale, minimale et moyenne et des contraintes.

TABLE 4.3 – Modèle relationnel du $SCH2$ (une instance de $SCH2$) (Concept)

Concept		
IdCt	NameCt	UT
Ct1	Person	120
Ct2	Customer	
Ct3	Company	
Ct4	Invoice	
Ct5	Order	

SynConcept					
IdSynCt	NameSynCt	Language	UT	IdCt	Tags
Ct1000	Person	English	120	Ct1	A person can have different names.
Ct1001	Man	English	11	Ct1	
Ct1002	Woman	English	20	Ct1	
Ct2000	Customer	English		Ct2	
Ct2001	Client	English	22	Ct2	
Ct2002	Guest	English		Ct2	
Ct2003	Patient	English		Ct2	
Ct3000	Organisation	English		Ct3	An organization may have a unique name, must have a phone.
Ct3001	Enterprise	English		Ct3	
Ct3002	Compagnie	French		Ct3	
Ct4000	Invoice	English		Ct4	
Ct4001	Statement	English		Ct4	
Ct4002	Facture	French		Ct4	

TABLE 4.4 – Modèle relationnel du $SCH2$ (une instance de $SCH2$) (Attribut)

Attribute								
IdA	NameA	IdCt	Type	Length	MinL	MaxL	AvL	Constraints
Ct1-C1001	Identifier	Ct1	String	50	20	80	50	
Ct1-C1002	Name	Ct1	String	30	30	30		
Ct1-C1003	Married_name	Ct1	String	30	30	30		
Ct1-C1004	Family_name	Ct1	String	30	30	30		
Ct1-C1005	FirstName	Ct1	String	30	30	30		
Ct1-C1006	Gender	Ct1	String	1	1	1	1	
Ct1-C1007	Civility	Ct1	String	20	20	20		
Ct1-C1008	Marital_status	Ct1	String	20	20	20		
Ct1-C1009	DateBirth	Ct1	Date	8				yyyy-mm-dd
Ct1-C1010	Age	Ct1	Number					Years
Ct1-C1011	Weight	Ct1	Number					Kilogrammes

SynAttribute						
IdSynA	NameSynA	Type	Language	IdAttribute	UT	Tags
Ct1-C100100	Identifier	String	English	Ct1-C1001	120	
Ct1-C100101	Identifiant	String	French	Ct1-C1001		
Ct1-C100200	Name	String	English	Ct1-C1002	100	
Ct1-C100201	Nom	String	French	Ct1-C1002		
Ct1-C100202	Nome	String	Portuguese	Ct1-C1002		
Ct1-C100300	Married name	String	English	Ct1-C1003		
Ct1-C100301	Nom matrimonial	String	French	Ct1-C1003		
Ct1-C100302	Nome de casada	String	Portuguese	Ct1-C1003		
Ct1-C100500	FirstName	String	English	Ct1-C1005		
Ct1-C100501	Surname	String	English	Ct1-C1005		
Ct1-C100502	Given name	String	English	Ct1-C1005		
Ct1-C100503	Fore name	String	English	Ct1-C1005		
Ct1-C100504	Prenom	String	French	Ct1-C1005		

avec :

- IdCt : IdConcept
- IdA : IdAttribute
- IdSynA : IdSynAttribute
- UT (UsedTimes) indique le nombre de fois qu'un objet est utilisé.
- MinL : Minimum length, MaxL : Maximum length et AvL : Average length

TABLE 4.5 – Ensemble de connaissances stockées dans [[$SCH2$]] (Concept *Person*)

Concept	Attribut	TypeAtt	ContraintesAtt	TagsAtt
Person Client Employee Customer Guest Man Woman Ouvrier Cliente Personne Personnel Homme Femme Patient Malade	**Identifier** Identifiant Numero	Number, String	PK (Primary Key) NN (Not Null)	
	Name Nom Nome FullName	String	CL (Capital Letter)	
	FirstName Prenom Surname	String		
	Birthdate	Date	MM/DD/YYYY	
	Age	Number	CK (Check) [0..150]	Interval of values
	Gender	String	CK {F, M}	Set of two values. F for Female and M for Male
	Civility	String	CK {Mrs, Mr} DF Gender.	Set of 2 values.
	Email	String		
	Phone	String	Optional	May contain the country code.
	MobilePhone	String	Optional	May contain the country code.
	Fax	String	Optional	May contain the country code.
	Statut	String	CK {Q1, Q2, Q3, Q4}	Set of 4 values
	Address	String		Composed of the number, type and name of the street.
	ZipCode	String	DF City	
	City	String		
	Country	String	DF Continent	
	Continent	String		

TABLE 4.6 – Ensemble de connaissances stockées dans [[$SCH2$]] (Concepts *Organisation, Invoice, Order*)

Concept	Attribut	TypeAtt	ContraintesAtt	TagsAtt
Organisation Entreprise Company Hospital University School Theater Cinema	**Identifier**	Number, String	PK	Identifies an organisation.
	Name	String	Unique	
	SIRET	Number	Unique	
	Type	String		
	Status	String	CK{SC, EURL, SARL, SAS, SA}	
	Phone Telephone Telefono	String	Mandatory	May contain the country code.
	Fax	String	Optional	May contain the country code.
	Address	String		Composed of the number, type and name of the street.
	ZipCode	String	DF City	
	City	String		
	Country	String	DF Continent	
	Continent	String		
Invoice Facture	**Identifier** Numero	Number, String Number, String	PK	
	Date	Date	JJ/MM/AAAA	
	IdClient	Number, String	FK	
Order Commande	**Identifier** Numero	Number, String Number, String	PK	
	Date	Date	JJ/MM/AAAA	
	IdClient	Number, String	FK	

Le modèle relationnel présente les contraintes et les dépendances sur une même colonne. Afin d'enrichir le référentiel (instance de $SCH2$) avec des informations sémantiques sur les dépendances entre attributs, on définit quelques dépendances sémantiques inter colonnes.

Exemple de dépendances sémantiques pour les différents concepts :

TABLE 4.7 – Dépendances sémantiques entre colonnes (DF X → Y)

Concept	**X**	**Y**	Satisfiable
Person	Country	Continent	T
Person	City	Country	T
Person	ZipCode	City	T
Person	Civility	Gender	T
Person	FirstName	Gender	F
Person	Identifier, Email	Name	T
Person	Name	FirstName	F
Organisation	Name	Statut	F
Organisation	ZipCode	City	T
Invoice	Date	Identifier	F

On définit aussi d'autre type de relations : les relations qui aident à reconnaître un concept (table 4.8). Par exemple, l'attribut *Gender* définit une personne tandis que l'attribut numéro de *SIRET* décrit une organisation. L'attribut *Name* définit à la fois une personne et une organisation ou d'autres concepts.

TABLE 4.8 – Relations de reconnaissance des concepts

Concept	**Attribut**
Person	Gender, Civility Name, FirstName Social Security Number Phone, Email
Organisation	SIRET, Name, Fax, Phone, Email

SCH2 est un ensemble de connaissances qui peut être géré en utilisant les ontologies grâce à leur meilleure expressivité. Nous présentons ci-après une instance du SCH2 en utilisant les ontologies.

Exemple 4.5 : Échantillon du référentiel [[SCH2]] (figure 4.2)

Le Concept *Person* peut avoir différents synonymes comme *Client*, *Employee*, *Customer*, *Guest*, *Cliente*, *Man*, *Woman*, *Homme*, *Femme*, *Personnel*, *Personne*.

Le concept *Person* peut aussi être décrit par un ou plusieurs attributs tels que *Name*, *MarriedName*, *FirstName*, *Birthdate*, *Age*, *Gender*, *Civility*, *Email*, *Phone*, *Fax*, *Address*, *ZipCode*, *City*, *Country*.

L'attribut *FirstName* a différents synonymes tels que *Prénom*, *Surname*.

Remarque : La notion de synonymie utilisée dans notre approche signifie que les informations ont le même sens et dans différentes langues.

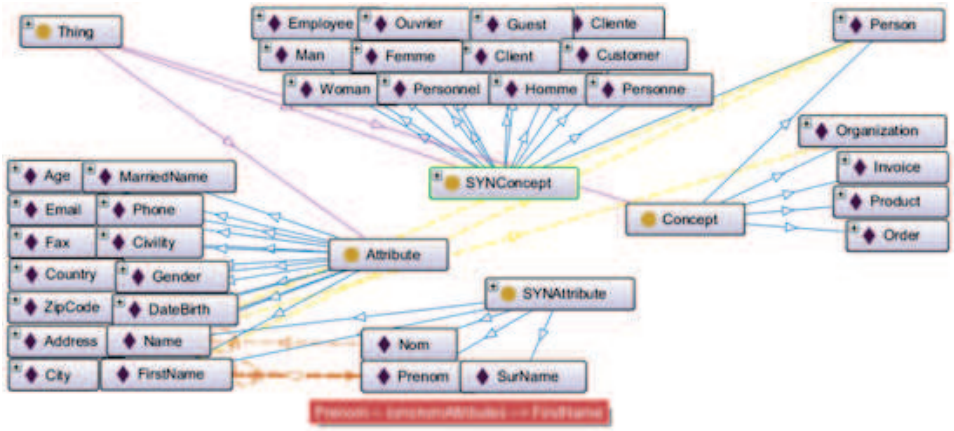


FIGURE 4.2 – Instance du Méta-Schéma SCH2

Plusieurs instances du SCH2 plus complètes sont présentées dans le chapitre 5.

La démarche de rapprochement du schéma sémantique SCHS aux instances du référentiel SCH2 est présentée ci-dessous.

4.3 Reconnaissance du concept

L'originalité de notre approche consiste à non seulement utiliser les noms de colonnes tel que mentionnés (s'ils existent) mais on vérifie leur sémantique et on propose un nom sémantique à chaque colonne (chapitre précédent).

Toutefois, le nom sémantique de certaines colonnes ne peuvent pas être reconnues. Seuls leurs types de données ont été reconnu telles que les colonnes une, dix et onze (structure sémantique table 3.16). Ces attributs peuvent fausser le rapprochement alors soit on essaye de les reconnaître en utilisant la sémantique existante dans le nom déjà attribué ou dans les commentaires d'une source avec un schéma au départ, soit on ne les prend pas en compte lors du rapprochement.

Premier cas : la source S est avec un schéma

La première alternative consiste à exploiter la sémantique du schéma de la source s'il existe. Dans ce cas, on réexamine le SCHS proposé lors du profilage et les attributs non reconnus sont comparés aux noms définis dans le schéma initial de la source S.

Le schéma de la source S est rapproché, d'une part, aux attributs du référentiel [[SCH2]] (classes *Attribute* et *SYNAttribute*), et d'autre part, aux deux bases Word-

Net [2] et WOLF [3] (Wordnet Libre du Français) afin de vérifier leur sémantique. Si ces attributs sont retrouvés dans le référentiel ou dans une des deux bases alors on met à jour le SCHS.

La deuxième alternative consiste à exploiter la sémantique existante dans les commentaires d'une source pour les différentes colonnes. Une information sémantique peut être déduite à partir de ces commentaires.

L'algorithme *getSemanticFromTags* (algo 13) est appliqué sur le commentaire d'une colonne donnée. Il calcule la présence (exacte ou rapprochée) du nom d'une colonne dans un commentaire. Le nom d'une colonne appartient à l'ensemble des noms des attributs et leurs synonymes (classe *Attribute* et *SYNAttribute*).

Algorithm 13 Algorithm getSemanticFromTags

```
Input :
    tag (Tag attribute)
    listRefAttributes (List of attributes and their synonyms)
    algoDistance, threshold similarity (ε)
Output : attributeName (String)
Begin
attributeName ← Null
while (attributeName = null) do
  for each C_i_name from listRefAttributes //i=1 ;n do
    for  each word from tag do
      algoDistance(C_i_name, word) ⩽ ε
      attributeName ← C_i_name
    end for
  end for
end while
return attributeName
End Algorithm getSemanticFromTags()
```

La colonne est reconnu, le SCHS est rapproché à tous les attributs du référentiel (attributs et leurs synonymes)

Deuxième cas : la source S est sans un schéma

Aucune nouvelle information ne peut être rajoutée au schéma sémantique. Le rapprochement final est effectué alors entre le SCHS et seulement les attributs des concepts (classe *Attribute*). Aucun rapprochement avec les attributs synonymes n'est proposé vu que les catégories du profilage présentent les attributs du référentiel.

2. http ://wordnet.princeton.edu/
3. http ://alpage.inria.fr/ sagot/wolf-en.html

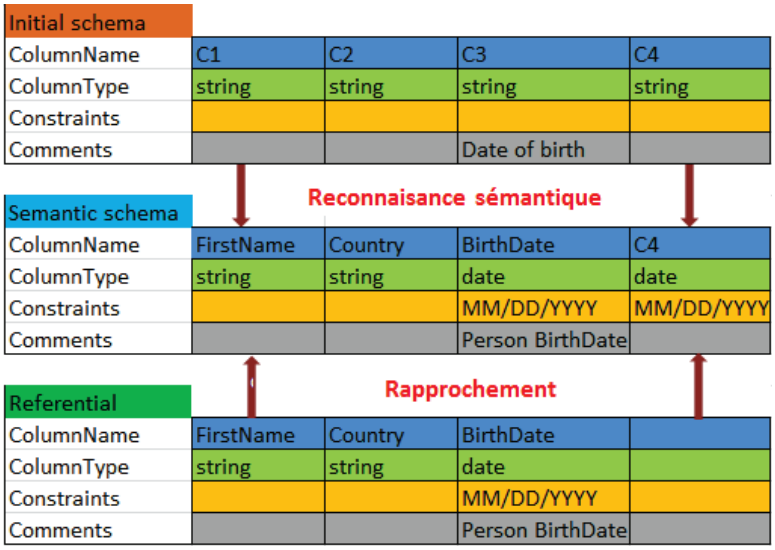


FIGURE 4.3 – Rapprochement du schéma sémantique au référentiel

La sémantique de la colonne "C3" n'a pas pu être découverte à partir de la première étape de reconnaissance (profilage de données). Le rapprochement du commentaire existant dans la source, pour la colonne en question, a permis de lui attribuer un nom sémantique. Une confirmation est ajoutée en rapprochement le nom, le type de données ainsi que la contrainte de la colonne par rapport au référentiel.
La contrainte sur la colonne "C3" est donnée par l'ensemble des analyses statistiques appliqué à la source (section 3.4.4, chapitre 3.7). Cependant, aucune information sémantique existante pourra détecter le nom sémantique de la colonne "C4" reste inconnu. Seul le type de données a été reconnu.

La première étape de reconnaissance, nous a permis d'attribuer à chaque colonne de la source un nom sémantique, un type de données et des contraintes. La prochaine étape est de reconnaître le concept décrit en rapprochant le schéma sémantique (résultat de profilage) par rapport à un référentiel. Différentes approches sont proposées.

Étant donné que le référentiel, instance de SCH2, est modélisé en utilisant des ontologies, une première réflexion était d'utiliser l'alignement d'ontologies. Nous avons alors dû transformer le schéma sémantique en une ontologie.

4.3.1 Première approche : Alignement des ontologies

Rappelons d'abord que la méthode de rapprochement, que nous proposons, diffère de celle présentée dans les travaux de la littérature (section 2.3.1) du fait que le schéma d'entrée est valide sémantiquement (figure 4.3).

Plusieurs travaux dans la littérature ont abordé cette problématique d'alignement d'ontologies telles que (Benslimane et al., 2006), (Euzenat and Valtchev, 2006), (Cerbah, 2008), (Shvaiko and Euzenat, 2013). Cependant leur objectif est l'automatisation de l'utilisation des quantités de données existantes dans les sources de données dans le Web sémantique qui diffère de notre objectif.
Plus de détails sur ces travaux seront présentés en annexe (section A.3.1).

Pour la transformation du SCHS en une ontologie, nous avons défini certaines règles de transformation (schéma → ontologie). Le tableau suivant (table 4.9) résume ces règles.

TABLE 4.9 – Définition de correspondance entre le schéma d'une source S et le SCH2

Éléments d'une source de données	**Éléments du SCH2**
Data source name	Concept
Columns name	Attributes
Constraints on the data source	ConstraintsDF
Constraints on attributes	ConstraintsPKFK
	ConstraintsIN
	ConstraintsCondition
Comments	Tags

Parmi les outils d'alignements existants dans la littérature, nous avons utilisé l'outil AlignApi (Euzenat, 2012). Plus de détails sur les différents outils et approches sont présentés dans l'annexe (section A.3.2).

AlignApi se base sur le calcul de similarité entre les noms des entités (attributs et concepts) en utilisant des distances de similarité telles que SMOA(Stoilos et al., 2005). Cependant, dans notre cas, on différencie bien les deux niveaux attributs et concepts. On ne rapproche pas le nom d'un concept à un nom d'un attribut. De surcroît, on vise un rapprochement plus riche et non seulement en fonction des noms des entités. Un rapprochement en fonction des autres informations sémantiques (type de données, contraintes et commentaires) existantes dans le SCHS et le référentiel est recommandé.

Évaluation de la première approche

On transforme le SCHS en une ontologie et on l'aligne avec le référentiel. On évalue le fichier de l'alignement (fichier rdf[4]) avec un fichier de référence. Ce dernier est construit à partir des différentes correspondances possibles pour un domaine donné. On calcule alors les mesures d'évaluation : la précision, le rappel et F-Mesure.

4. http ://www.w3.org/RDF/

L'outil AlignApi propose une classe qui calcule ces mesures [5] .

Exemple 4.6 : Alignement des ontologies (SCHS et [[SCH2]]
Soit le schéma sémantique suivant (figure 4.4). SCHS représente le schéma d'une source de données décrivant une personne. On aligne ce schéma avec le référentiel (instance du SCH2).

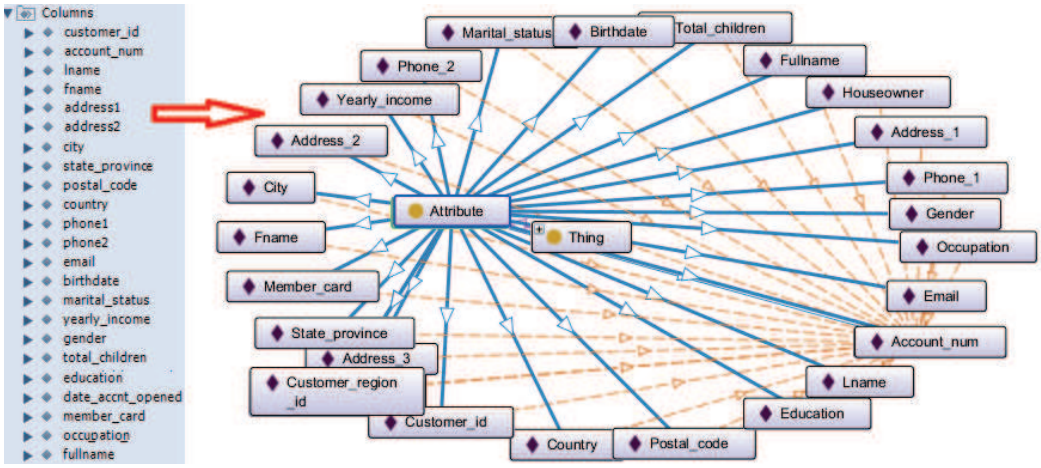


FIGURE 4.4 – Schéma sémantique d'une source S

On obtient en sortie, un fichier rdf contenant les différentes correspondances entre les attributs du SCHS et les attributs (instances des classes *Attribute* et *SynAttribute*) et les concepts (instances des classes *Concept* et *SynConcept*). Ci-après un aperçu de ce fichier (figure 4.5).

```
<map>
  <Cell>
    <entity1 rdf:resource='file:/C:/workspaceSADL2/JenaSemProject/data/Organization'/>
    <entity2 rdf:resource='file:/c:/onto2/Occupation'/>
    <relation>=</relation>
    <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>0.5954545454545455</measure>
  </Cell>
</map>
```

FIGURE 4.5 – Extrait du fichier de sortie de l'outil AlignApi

On remarque que l'outil AlignApi ne prend pas en compte la structure hiérarchique adoptée dans le SCH2 (la différence entre les concepts et attributs). Il rapproche le concept *Organisation* à l'attribut *Occupation*. Cette méthode n'est donc pas adaptée à notre besoin. Il compare juste des entités entre eux. Il ne permet pas de reconnaître le concept correspondant à un ensemble d'attributs.

5. http ://alignapi.gforge.inria.fr/eval.html

Pour évaluer le résultat d'alignement, il fallait construire manuellement un fichier, appelé "correct alignment [6]". C'est un fichier de référence par rapport aux rapprochements souhaités. Lors du calcul des différentes mesures d'évaluation, il sera comparé avec le résultat d'alignement obtenu.

Nous avons créée un fichier contenant le "correct alignment", pour le concept *Person* et nous avons calculé les trois mesures : Précision, Rappel et FMeasure (table4.10).

TABLE 4.10 – Mesures d'évaluation de l'alignement proposé par l'outil AlignApi

	Précision	**Rappel**	**FMeasure**
SCHS	0.148	0.102	0.121

Ces résultats trop faibles dépendent de la taille du référentiel ainsi que du fichier "correct alignment". Nous avons alors défini une nouvelle méthode de rapprochement de schémas.

4.3.2 Deuxième approche : Alignement des éléments du schéma

Le principe de rapprochement dans la deuxième méthode consiste à rapprocher, d'un côté, le nom de la source s'il existe avec les instances des deux classes du référentiel *Concept* et *SynConcept* et d'un autre coté, rapprocher les attributs entre eux (attributs du SCHS avec les attributs des deux classes *Attribute* et *SynAttribute* de chaque concept du référentiel). Ce rapprochement se fait en fonction de trois critères :

- lexicographique en fonction des noms des attributs ainsi que le type de données,
- rapprochement des contraintes,
- rapprochement des commentaires.

Différents cas se présentent lors de l'alignement du SCHS avec les concepts du référentiel. Plusieurs cas existent. Nous étudions les sept cas suivants.

Définition 4.11 : Attributs d'un concept
Soit C_i l'ensemble des attributs du concept Ct_i. $C_i \neq \emptyset$.
$C_i = \{C_{i1}, C_{i2}, ...C_{ip}\}$
$|C_i|$ est le nombre d'attributs de l'ensemble C_i.

6. http ://alignapi.gforge.inria.fr/eval.html

C est l'ensemble d'attribut de SCHS.
C_{ref} est l'ensemble de tous les attributs du référentiel. ■

Cas 1 : Aucun des attributs du schéma SCHS n'existe dans le référentiel (figure 4.6)
$C \cap C_{ref} = \emptyset$

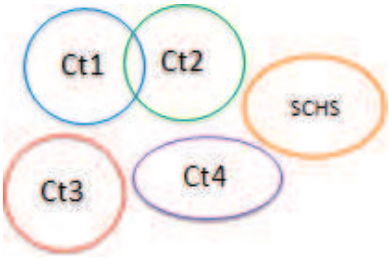


FIGURE 4.6 – Alignement du SCHS avec les concepts du référentiel : Cas 1

Cas 2 : Tous les attributs du SCHS appartiennent à un même et seul concept Ct (figure 4.7)
$\exists$ i tel que :
$C \cap C_i = C$ et $C \subseteq C_i$

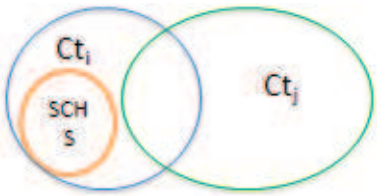


FIGURE 4.7 – Alignement du SCHS avec les concepts du référentiel : Cas 2

Cas 3 : Certains attributs du schéma existent parmi les attributs d'un seul concept Ct du référentiel (figure 4.8)
$\exists$ i tel que :
$C \cap C_i \neq \emptyset$
et $C \subseteq C_i$.

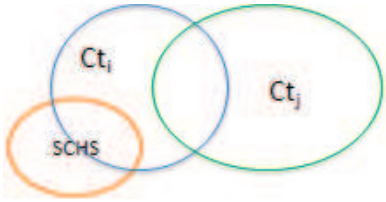


FIGURE 4.8 – Alignement du SCHS avec les concepts du référentiel : Cas 3

Cas 4 : Les attributs du SCHS appartiennent à l'intersection des deux concepts (figure 4.9)
∃ i,j tel que :
$C_i \cap C_j = C$, $C_i \cap C = C$ et $C_j \cap C = C$.

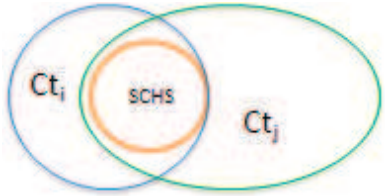


FIGURE 4.9 – Alignement du SCHS avec les concepts du référentiel : Cas 4

Cas 5 : SCHS vérifient le concept Ct_j, tandis qu'une partie est commune avec un autre concept Ct_i (figure 4.10)
∃ i,j tel que :
$C_i \cap C_j \neq \emptyset$, $C_i \cap C \neq \emptyset$, $C_j \cap C \neq \emptyset$, $C \subsetneq C_i$ et $C \subsetneq C_j$.

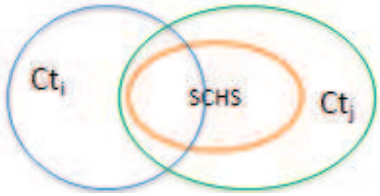


FIGURE 4.10 – Alignement du SCHS avec les concepts du référentiel : Cas 5

Cas 6 : Une partie des attributs du SCHS appartient à l'intersection des deux concepts (Ct_i et Ct_j) et une partie n'appartient à aucun des concepts (figure 4.11)
∃ i,j tel que :
$C_i \cap C_j \neq \emptyset$, $C_i \cap C \neq \emptyset$, $C_j \cap C \neq \emptyset$, $C_i \cap C_j \cap C \neq \emptyset$, $C \subsetneq C_i$ et $C \subsetneq C_j$.

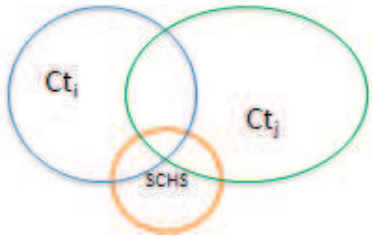


FIGURE 4.11 – Alignement du SCHS avec les concepts du référentiel : Cas 6

Cas 7 : Une partie du SCHS appartient indépendamment aux deux concepts (C_i et C_j) et une partie des attributs n'appartient à aucun des concepts (figure 4.12)
∃ i,j tel que :
$C_i \cap C_j = \emptyset$, $C_i \cap C \neq \emptyset$, $C_j \cap C \neq \emptyset$, $C \subsetneq C_i$ et $C \subsetneq C_j$.

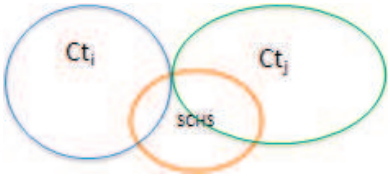


FIGURE 4.12 – Alignement du SCHS avec les concepts du référentiel : Cas 7

Afin de décider que faire dans ces différents cas et rapprocher le SCHS avec les concepts du référentiel, on propose deux méthodes. Rappelons que l'objectif est :

1. Identifier le SCHS avec un concept existant.
2. Identifier le SCHS comme un nouveau concept n'existant pas encore dans le référentiel (avec une possibilité d'enrichir le référentiel).
3. Le SCHS représente une partie d'un concept existant dans le référentiel.

Méthode 1 : Calcul du score de similarité entre SCHS et les concepts du [[SCH2]] Les attributs du SCHS sont rapprochés aux attributs de chaque concept Ct_i du référentiel (instance de SCH2). Un score est renvoyé pour chaque Ct_i.

Un schéma est similaire à un concept ssi le score est proche de un (score ≈ 1).

Si le score est égale à zéro alors le SCHS est un nouveau concept par rapport au référentiel sinon on choisit le concept dont le score est le plus élevé. On reconnaît alors le concept décrit par le SCHS.

Présentons ci-après l'algorithme du processus d'alignement sémantique de la première méthode (algo 15) :

Algorithm 14 Algorithm Concept recognition

```
Input :
    T7 (Semantic data structure)
    [[SCH2]] (Referential)
Output :
    Ctm (Matched Concept)
Begin
//loop over each concept of the referential
for each Cti from [[SCH2]] (i=1 ;n) do
  score ← matchConcept(Cti, T7)
end for
if score=0 then
  Ctm ← ∅ //Case 1 (figure 4.6)
else
  //only one concept match
  if Ct ← getMaxScore(score) then
    Ctm ← Ct
  end if
end if
End Concept recognition
```

La méthode *matchConcept* permet de calculer un score entre chaque concept Ct_i du référentiel et le SCHS (T_7).

Les attributs de chaque Ct_i sont triés et sauvegardés sous la forme d'une chaîne de caractères. Cette dernière est composée des noms des attributs, du type de données et les contraintes.

De même, le schéma SCHS avec ses différentes composantes (nom de colonnes, type de données, contraintes) est représenté dans une chaîne de caractères. Cette dernière est triée dans l'ordre alphabétique.

Exemple 4.7 : Concepts du référentiel et le schéma sémantique d'une source sans schéma

Nous présentons dans l'exemple suivant un schéma sémantique que l'on va essayer d'aligner avec des concepts du référentiel.

TABLE 4.11 – Concepts du référentiel et le schéma sémantique

[[SCH2]]	Person(Name : {string}, MarriedName : {string}, FirstName : {string}, BirthDate : {date, //date of birth}, Gender : {string, CK (F, M)}, Civility : {string, CK (Mr, Mme)}, Country : {string}, City : {string}, Phone : {string}, Email : {string}). Organization(Identifer : {integer}, Name : {string}, Phone : {string}, Fax : {string}, DateCreation : {date}, Country : {string}, City : {string}, Status : {string}, Sales : {real}). Product(Identifier : {integer}, Name : {string}, Quality_Product : {string}, Color : {string}, BrandName : {string}).
SCHS	S(FirstName : {string}, Country : {string}, City : {string}, Column3 : {string}, Temperature : {real, CK [37..40] °C}, DateBirth : {date}).

Un algorithme de distance de similarité est utilisé afin de calculer la similarité entre les deux chaînes.

Algorithm 15 Algorithm matchConcept

Input :
  $SCHS$ (Semantic data structure)
  Ct (A concept from the referential)
Output :
  score (score of each concept)
Begin
stCtRef ← null
stSCHS ← null
//get attributes name, data type and constraints of the concept Ct and sort them

stCtRef ← sort(getAttributesFrom(Ct))
//loop on semantic schema attributes and get their name, data type, constraints and tags
for each C_p from $SCHS$ (p=1 ;l) **do**
  // build and sort the semantic schema string
  stSCHS ← sort(addAttribute(C_p))
end for
score ← algoDistance(stCtRef, stSCHS)
return score
End matchConcept

Dans le rapprochement des contraintes, on vérifie que le nom des contraintes et pas leur valeur.

Le résultat de rapprochement selon différentes mesures de similarité (table 4.12) :

TABLE 4.12 – Scores de similarité entre les différents concepts du référentiel et le SCHS

		JW	Lv	QGram
Ct_1	SCHS	0,672	0,403	0,74
Ct_2	SCHS	0,854	0,517	0,74
Ct_3	SCHS	0,725	0,514	0,462

Les distances obtenues sont un peu ambiguës. Pour s'aider dans la prise de décision pour le concept correspondant, nous pouvons tenir en compte des attributs permettant de reconnaître un concept particulier (table 4.8). Dans notre cas, l'attribut *FirstName*, permet de reconnaître le concept Ct_1 comme étant le concept le plus proche du SCHS.

Pour cela, nous présentons dans ce qui suit, une nouvelle méthode plus précise dans le choix du concept correspondant.

Méthode 2 : Approche ensembliste (Vérifier l'appartenance de chaque attribut du SCHS à un concept du référentiel) La deuxième méthode consiste à rapprocher chaque attribut du SCHS (nom, type de données et contraintes) par rapport aux attributs des différents concepts du référentiel.

Pour cela, nous calculons, comme pour la déduction de la catégorie dominante dans le chapitre précédent (section 3.4.3), une probabilité d'appartenance du SCHS à un concept du référentiel, en utilisant le naïve bayésienne (NB) pour la classification de textes.

La classification de textes en utilisant NB consiste à calculer la probabilité conditionnelle (formule A.1) :

$$P(ct\backslash d) = \frac{P(d\backslash ct).P(ct)}{p(d)} \tag{4.1}$$

avec d (document) représente le schéma sémantique, l'ensemble d'attributs décrivant une source de données.

Dans notre cas (d=$\{C_i\}$). ct est un concept. Les concepts doivent être disjoints. Cependant, les attributs peuvent décrire différents concepts $C_i \in Ct_j$ and $C_i \in Ct'_j$.

On cherche à trouver le meilleur concept décrivant l'ensemble d'attributs du SCHS. Il faut maximiser alors les probabilités suivantes :

$$Ct_{MAP} = \underset{ct \in Ct}{argmax} P(d \backslash ct) \cdot P(ct)$$

P(ct) est la probabilité d'un concept dans le référentiel.

$$Ct_{MAP} = \underset{ct \in Ct}{argmax} P(\{C_1, C_2, ...C_n\} \backslash ct) \cdot P(ct)$$

or

$$P(\{C_1, C_2, ...C_n\} \backslash ct) = p(C_1 \backslash ct) \cdot p(C_2 \backslash ct) \cdot p(C_3 \backslash ct) \cdot ... \cdot p(C_n \backslash ct)$$

donc

$$Ct_{NB} = \underset{ct \in Ct}{argmax} P(ct_j) \prod_{C_i \in C} P(C_i \backslash ct)$$

avec C est l'ensemble des colonnes, n nombre d'attributs par concept (document) et N le nombre de concepts.

Exemple 4.8 : Choix du concept correspond au SCHS

Soit 3 concepts Person, Organisation et Invoice du référentiel (table 4.13) (extrait des deux tables **??**) :

TABLE 4.13 – Ensemble de concepts du référentiel

Concept	Attribut	Type de données	Contrainte
Person	Name	String	CL
	FirstName	String	
	Gender	String	CK
	Address	String	
	City	String	
	Country	String	
Organisation	Name	String	CL
	SIRET	Number	UN
	Phone	String	OP
	Address	String	
	City	String	
	Country	String	
Invoice	Identifier	Number	PK, UN
	Date	Date	
	IdClient	String	FK

Exemple de schéma sémantique SCHS dont on cherche le concept correspondant (table 4.14) :

TABLE 4.14 – Schéma sémantique SCHS

SCHS				
Name	FirstName	Gender	City	ZipCode
String	String	String	String	String
CL		CK		

$$P(ct_j) = \frac{cptcount(Ct = ct_j)}{N_{concept}}$$

La probabilité qu'un concept $ct_j = Person$ est égale :

$$P(ct_j = Person) = \frac{1}{N} = \frac{1}{3}$$

de même pour

$$P(ct_j = Organisation) = \frac{1}{3}$$
$$P(ct_j = Invoice) = \frac{1}{3}$$

Pour calculer la probabilité d'appartenance d'un attribut C_i à un concept ct_j, on utilise la probabilité de Maximum Likelihood :

$$\hat{P}(C_i \backslash ct_j) = \frac{count(C_i, ct_j)}{\sum_{c \in C} count(c, ct_j)}$$

Cependant, le rapprochement qu'on propose ne se limite pas aux noms des attributs mais aussi aux types de données des attributs ainsi que les contraintes. Donc la probabilité :

$$\hat{P}(C_i \backslash ct_j) = \hat{P}(w_i \backslash ct_j).\hat{P}(dt_i \backslash ct_j).\hat{P}(const_i \backslash ct_j)$$

avec w_i est le nom de l'attribut C_i, dt_i est le type de données de C_i et $const_i$ est le type de la contrainte sur C_i.

On calcule alors les trois probabilité de la même manière :

$$\hat{P}(w_i \backslash ct_j) = \frac{n_k + \alpha}{n + \alpha|C|}$$

avec n_k est le nombre d'occurrence des attributs dans un concept.

On calcule le nombre d'attribut $C_i \in ct_j$, diviser par le nombre d'attributs dans chaque concept c_j avec α multiplié par la cardinalité du vocabulaire.

|C| = 11 (nombre d'attributs existant dans les différents concepts).

TABLE 4.15 – Calcul de probabilité pour le SCHS (nom des attributs)

SCHSProbabilité	**Person**	**Organisation**	**Invoice**
Name	$\frac{1+\alpha}{7+11\alpha}$	$\frac{1+\alpha}{7+11\alpha}$	$\frac{\alpha}{4+11\alpha}$
FirstName	$\frac{1+\alpha}{7+11\alpha}$	$\frac{\alpha}{7+11\alpha}$	$\frac{\alpha}{4+11\alpha}$
Gender	$\frac{1+\alpha}{7+11\alpha}$	$\frac{\alpha}{7+11\alpha}$	$\frac{\alpha}{4+11\alpha}$
City	$\frac{1+\alpha}{7+11\alpha}$	$\frac{1+\alpha}{7+11\alpha}$	$\frac{\alpha}{4+11\alpha}$
ZipCode	$\frac{\alpha}{7+11\alpha}$	$\frac{\alpha}{7+11\alpha}$	$\frac{\alpha}{4+11\alpha}$

|dataType| = 4 est le nombre de types de données existant dans les différents concepts. On calcule la probabilité que pour le type *String* vu que les attributs du SCHS ont tous le même type.

TABLE 4.16 – Calcul de probabilité pour le SCHS (type de données des attributs)

SCHSProbabilité	**Person**	**Organisation**	**Invoice**
String	$\frac{6+\alpha}{7+4\alpha}$	$\frac{5+\alpha}{7+4\alpha}$	$\frac{1+\alpha}{4+4\alpha}$

|Constraint| = 7 est le nombre de type de contraintes existant dans les différents concepts. De même on calcule la probabilité pour les deux types de contraintes utilisés dans le SCHS.

TABLE 4.17 – Calcul de probabilité pour le SCHS (type des contraintes des attributs)

SCHSProbabilité	**Person**	**Organisation**	**Invoice**
CL	$\frac{1+\alpha}{7+7\alpha}$	$\frac{1+\alpha}{7+7\alpha}$	$\frac{\alpha}{4+7\alpha}$
CK	$\frac{1+\alpha}{7+7\alpha}$	$\frac{\alpha}{7+7\alpha}$	$\frac{\alpha}{4+7\alpha}$

Pour $\alpha = 1$, on aura :

$$P(d\backslash Person) = \frac{\alpha(1+\alpha)^4}{(7+11\alpha)^5}.\frac{(6+\alpha)^5}{(7+4\alpha)^5}.\frac{\alpha^3(1+\alpha)^2}{(7+7\alpha)^5} = 1,82.10^{-8}$$

$$P(d\backslash Organisation) = \frac{\alpha^3(\alpha+1)^2}{(7+11\alpha)^5}.\frac{(5+\alpha)^5}{(7+4\alpha)^5}.\frac{\alpha^4(1+\alpha)}{(7+7\alpha)^5} = 3,8.10^{-13}$$

$$P(d\backslash Invoice) = \frac{\alpha^5}{(4+11\alpha)^5}.\frac{(1+\alpha)^5}{(4+4\alpha)^5}.\frac{\alpha^5}{(4+7\alpha)^5} = 7,98.10^{-15}$$

Donc, le concept **Person** est le plus proche pour décrire le schéma sémantique SCHS.

Consolidation de l'approche de rapprochement de schéma

Le calcul de probabilité d'appartenance du SCHS à un concept du référentiel [[SCH2]], semble être la plus efficace dans le traitement des différents cas définis ci-dessus. Concernant la première méthode, l'absence de certains éléments sur les attributs tels que les contraintes peut fausser la similarité.

Le traitement des commentaires n'est pas bien pris en compte dans les méthodes définies ci-dessus. La méthode *matchTags(conceptName, tags)* permet d'extraire de la sémantique à partir des commentaires (tags) existant sur une source. On calcule la présence du nom du concept Ct_m, reconnu par l'approche probabiliste, dans les tags correspondants (algo 16).

Algorithm 16 Function matchTags(conceptName, tag)

```
Input :
    tag (Tag concept)
    conceptName (string)
    algoDistance, threshold similarity (ε)
Output : match Boolean
Begin
match ← False
while (match = False) do
  for each word from tag do
    if algoDistance(word, conceptName) ⩽ ε then
      match ← True
    end if
  end for
end while
return match
End Function matchTags()
```

Cette dernière étape permettra de valider certains choix entre des concepts assez proches du SCHS.

L'alignement sémantique permet autre que la reconnaissance des relations inter attributs et la sémantique du nom de la source, la recommandation de nouvelles connaissances telles que des analyses de colonnes et des analyses de source et les attributs de dédoublonnage ainsi l'enrichissement du référentiel.

4.4 Recommandation sémantique des analyses

Rappelons que notre objectif est de guider l'utilisateur dans sa démarche qualité. Pour cela, nous proposons un ensemble d'analyses sémantiques pour la source de données afin d'aider l'utilisateur dans la compréhension de ses données.

Les analyses sont recommandées pour un schéma sémantique (figure 2.8). Le référentiel contient des informations déduites à partir des connaissances utilisateur.

On recommande alors pour chaque colonne un ensemble d'indicateurs adéquats aux données (en fonction de la catégorie et du type de données). Pour cela, on a fait une étude bibliographique sur les approches et outils de recommandation existants dans la littérature et sur le marché afin de proposer des indicateurs pertinents.

4.4.1 Étude de l'existant

Comme l'indique le nom de cette section, nous nous sommes intéressés dans notre recherche bibliographique aux systèmes de recommandation [7].

Ces systèmes permettent de filtrer des informations tels que les films, la musique, les livres, les news, les images ou les pages Web qui sont susceptibles d'intéresser l'utilisateur. Le système de recommandation permet de comparer le profil d'un utilisateur à certaines informations de référence, et cherche à prédire l'avis que donnerait un utilisateur.

Deux grands types de recommandation (Gong, 2010) :

- **recommandation basée sur le contenu** (Dis-moi quelle musique je veux écouter maintenant) : rapprocher les objets par rapport aux préférences et notes des utilisateurs. Il n'est pas nécessaire de connaître les caractéristiques des utilisateurs ou des objets. On doit connaître juste le contenu d'un objet, par exemple le genre d'un film et les préférences de l'utilisateur (romantique et comédie).
- **recommandation collaborative (RC)** rapproche les objets par rapport à un ensemble de personnes qui partage les mêmes envies. RC se résume dans la phrase suivante : "Le client qui achète cet objet, achète ..." et se base sur deux approches.
  - La première approche est l'approche basée sur l'utilisateur. Elle compare les notes d'un utilisateur cible avec les notes des autres utilisateurs (aime et n'aime pas) afin de construire un groupe de personnes similaires (chercher les utilisateurs qui partagent le maximum d'objets). Ensuite, les objets les plus annotés dans le groupe sont proposés à l'utilisateur.

7. http ://fr.wikipedia.org/wiki/Système_de_recommandation

— Deuxième approche basée sur l'objet. Elle consiste à proposer les objets similaires en fonction des notes des utilisateurs (les plus notés par les utilisateurs). Le principe de cette approche consiste à : (i) classer les objets, notés de la même façon par les utilisateurs dans un même groupe ; (ii) calculer la similarité entre objets avec la distance *Pearson*[8] ; (iii) recommander alors les objets similaires à l'utilisateur ; et enfin (iv) calculer le poids d'un objet par rapport à un utilisateur.

Notre recommandation ne repose pas sur les notes ou avis utilisateurs mais plutôt sur :

1. La reconnaissance sémantique des attributs et
2. la reconnaissance des relations sémantique entre les attributs telles que la relation de synonymie ou de dépendance.

La relation de synonymie entre l'Attribut1 et l'Attribut11 (déduit de l'alignement), nous permet de recommander les mêmes indicateurs (Indicateur1 et Indicateur4) de l'attribut1 à l'attribut11 (table 4.18).

TABLE 4.18 – Recommandation des indicateurs en fonction de la synonymie entre attributs

AttributsIndicateurs	Ind1	Ind2	Ind3	Ind4	Ind5	Ind6
Attribut1	X			X		
Attribut2		X			X	
Attribut11	X			X		
Attribut3	X					X

4.4.2 Recommandation des analyses

Nous recommandons deux types d'analyses à savoir l'analyse des attributs et celle de la source.

4.4.2.1 Analyse des attributs

Comme indiqué dans l'exemple appliqué à l'approche de recommandation basée utilisateur (table 4.18), nous allons recommander pour chaque attribut du schéma sémantique, un ensemble d'indicateurs selon trois critères :

1. Proposer des indicateurs basiques ($Istat_{ij}$) tels que nombre total de valeurs, nombre de valeurs nulles.

8. http ://gedas.bizhat.com/dist.htm

2. Proposer des indicateurs en fonction de type des données. Par exemple, pour les chaînes de caractères, proposer les indicateurs ($Istat_{kj}$) taille maximale, minimale et moyenne des chaînes.
3. Recommander des expressions régulières (table 4.11) en rapprochant leurs noms (catégories) aux noms et types des attributs. Par exemple, pour l'attribut *Email*, on pourra le rapprocher à l'expression régulière "Address_Email", existante dans l'outil Talend. Pour le type de données Date, on pourra rapprocher l'attribut aux expressions régulières "French Date" ou "English Date". Le rapprochement est fait en utilisant un algorithme de calcul de similarité et en fonction d'un seuil ε_1.
4. Utiliser la relation de synonymie. Dans cette partie, on essaye d'apprendre des connaissances déjà existantes dans le référentiel ontologique. On recommande les mêmes connaissances pour deux attributs similaires. Cette similarité est une relation définie par l'alignement sémantique entre le schéma sémantique de la source et le référentiel.

   Par exemple, un attribut A existant dans le référentiel est similaire à un attribut B du schéma sémantique (A≈ B selon l'alignement). Si A est définit en fonction d'un ensemble d'indicateurs IA alors l'ensemble IA peut être appliqué à l'attribut B.

Les indicateurs recommandés sont stockés dans le référentiel avec leurs noms, leurs résultats et les seuils utilisés. Ils sont aussi attachés aux attributs analysés (figure 4.13).

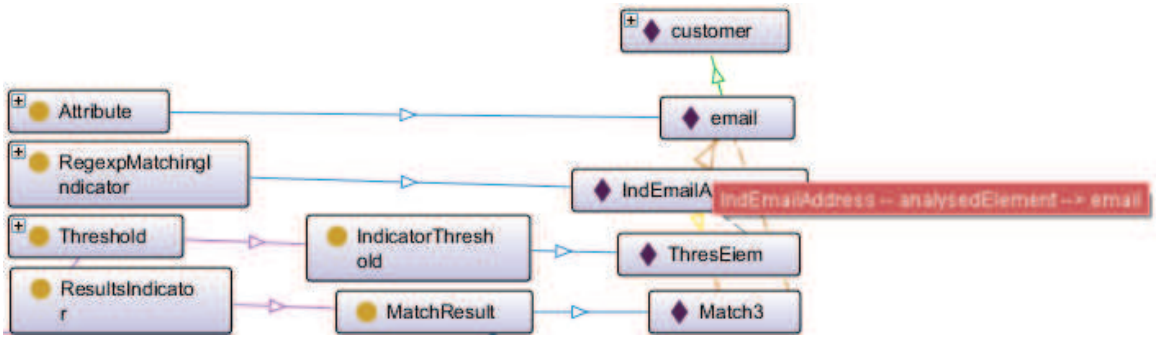


FIGURE 4.13 – Sauvegarder les indicateurs dans le référentiel ontologique

Un enrichissement du référentiel avec ces nouvelles connaissances déduites à partir des analyses sur les données.

4.4.2.2 Analyse de la source (concept)

La reconnaissance sémantique des liens entre colonnes et du concept décrit par ces colonnes permet de découvrir les connaissances liées à chaque concept. Ces connaissances facilitent la recommandation des analyses sur les concepts (l'ensemble des colonnes).

Par exemple, si avec l'alignement, on a découvert qu'un concept $Ct_1 \approx Ct_2$, avec Ct_1 est un concept du référentiel et Ct_2 est le concept déduit pour un schéma sémantique, alors toute analyse appliquée à Ct_1 peut être appliquée à Ct_2. Le profil de chaque analyse est stocké dans le référentiel.

Exemple d'analyse que nous pourrons recommander à une source est l'analyse de rapprochement de données (Matching analysis). Cette analyse permet de détecter la présence ou non des doublons et similaires dans une source.

Si une analyse de ce type a été déjà appliquée sur le concept Ct_1 , alors la configuration de cette dernière peut être utilisée avec le concept Ct_2. La configuration de cette analyse consiste à définir des attributs de regroupement et des règles de correspondance.

Pour chaque règle, on définit un ou plusieurs attributs de dédoublonnage, un algorithme de distance de similarité et un seuil. Plus d'une règle de correspondance peuvent être configurées.

La recommandation de cette configuration repose sur la connaissance/expérience utilisateur (indicateur, seuil, algorithme de similarité) stockée dans le référentiel.

Notons aussi que les relations inter colonnes reconnues lors d'alignement permettent de suggérer des analyses entre colonnes telles que l'analyse des dépendances fonctionnelles.

Une dernière étape dans le processus de la reconnaissance sémantique des schémas de données est l'enrichissement sémantique.

4.5 Enrichissement sémantique du référentiel [[SCH2]]

Après l'enrichissement sémantique des dictionnaires (DD et KW) dans le chapitre précédent, on propose à ce stade d'enrichir le référentiel (instance SCH2) (figure 4.14).

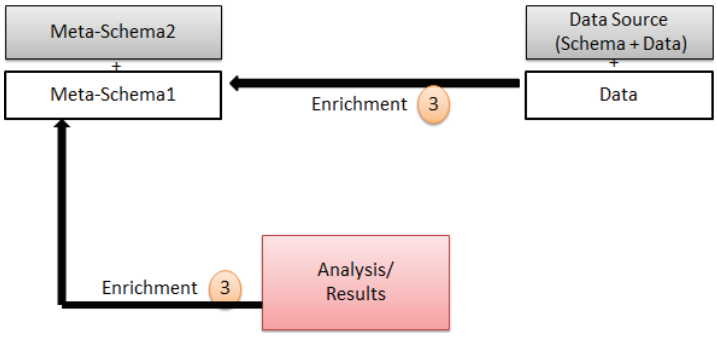


FIGURE 4.14 – Etape 3 Enrichissement sémantique

Différents niveaux du référentiel peuvent être enrichis.

Enrichissement avec des nouveaux attributs Les métadonnées du SCH2 peuvent être enrichies avec des nouveaux attributs et leurs synonymes. Ces attributs peuvent exister dans le schéma initial de la source. Avant de les ajouter, il faut vérifier certaines conditions :

- Le nouvel attribut doit être valide (bien orthographié définition 3.9, chapitre3.7).
- Il doit être différent du nom d'un concept.
- Il doit être différent des attributs existants dans le référentiel [[SCH2]]. La distance de similarité doit être proche de 0.
- Il doit exister dans au moins une des bases lexicales (LDB) telles que WordNet ou WOLF.

L'algorithme suivant présente le détail du processus d'enrichissement en fonction des attributs non reconnus dans le référentiel et existant dans le schéma initial de la source S (algo 17). Ces attributs sont ajoutés à la classe *Attribute* du référentiel.

Algorithm 17 Algorithm SemanticEnrichment

```
// automatic enrichment of the referential.
Input :
    Referential [[SCH2]]
    LDB (Lexical DataBase)
    algoDistance, ε threshold similarity
Output: Enriched referential [[SCH2']]
Begin
for each Ct_i from [[SCH2]] (i=1 ;n) do
  for each C_ij from Ct_i (j=1 ;p) do
    for each C_k from SCHS (k=1 ;m) do
      if (C_k is validString and C_k.Name ≠ Ct_i.Name and
      algoDistance(C_k, C_ij) < ε and C_k ∈ LDB) then
        [[SCH2']].add(C_k, Ct_i)
      end if
    end for
  end for
end for
End Algorithm SemanticEnrichment
```

Enrichissement avec des nouveaux concepts Des nouveaux concepts peuvent être ajoutés au référentiel selon deux cas :

— si le nom d'une source existe alors on peut l'ajouter à la classe *Concept*.

— si un ensemble d'attribut est commun à plus d'un concept, alors on peut créer un nouveau concept avec ces attributs. Cependant, le problème qui reste à gérer est comment attribuer un nom à ce nouveau concept ?

Enrichissement avec les résultats des analyses Les analyses proposées dans la section précédente peuvent être stockées et permettent l'enrichissement du référentiel avec les indicateurs, leurs résultats ainsi que les seuils. Par exemple, on applique l'indicateur *FrequencyTable* sur l'attribut *Gender*. L'indicateur renvoie les deux valeurs "F" et "M". On peut alors ajouter cette information comme étant le domaine de définition de l'attribut *Gender* dans le référentiel.

Un autre exemple, l'indicateur *FrequencyTable* renvoie la valeur "F" comme étant la valeur la plus fréquente pour l'attribut *Gender* (figure 4.15).

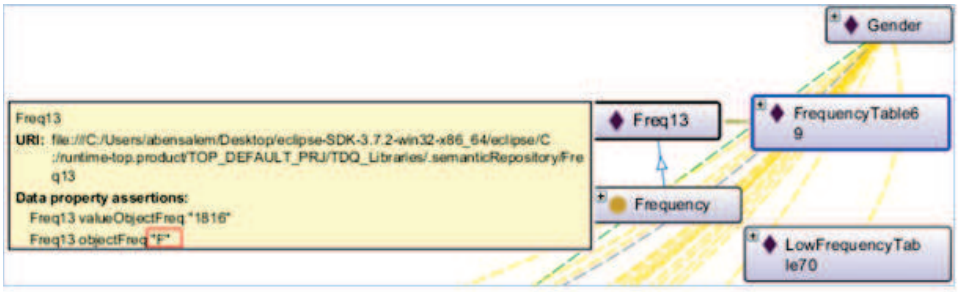


FIGURE 4.15 – Résultat de l'indicateur FrequencyTable sous Protégé

Ces deux informations sont sauvegardées dans le référentiel pour une éventuelle utilisation dans une action de correction telle que l'homogénéisation (plus de détails dans le chapitre 6.

4.6 Conclusion

Le rapprochement et l'alignement de concepts restent toujours des tâches difficiles. L'information sémantique sur les colonnes d'un schéma retrouvée dans le chapitre précédent présente une aide importante lors de l'alignement. Le nom sémantique des attributs, leur type de données ainsi que les contraintes déduites par les analyses permet un meilleur rapprochement.

Ces deux étapes de reconnaissance sémantique du schéma permettent de mieux comprendre les données et leur proposer des actions (Ben-Salem, 2014). Nous proposons alors aux utilisateurs des analyses sémantiques en fonction du schéma de la source reconnu SCHS. Notre objectif est d'aider l'utilisateur à maîtriser ses données et enrichir en même temps le référentiel ontologique.

Les analyses permettent de retrouver des contraintes non spécifiées dans une source telles que les dépendances fonctionnelles, définir un domaine des données.

Les rapports d'anomalies permettent de détecter les anomalies existantes dans les colonnes. Dans la partie suivante, nous allons corriger les données de toute la source de données.

Chapitre 5

Expérimentation : Reconnaissance sémantique du schéma d'une source de données

Sommaire

5.1 Introduction

Nous allons présenter dans ce chapitre, la partie initialisation du méta-schéma SCH1 (l'ensemble des expressions régulières, le dictionnaire de mots clés ainsi que le dictionnaire de données). Ensuite, nous introduisons le prototype que nous avons défini pour le profilage sémantique des données. Nous présentons les outils utilisés ainsi que les différents rapports renvoyée sur la source S.

5.2 Initialisation des méta-schémas

Tout au long de ce processus de reconnaissance de schémas, nous avons utilisé différents méta-schémas. Le premier méta-schéma (SCH1) contenait les modèles en entrée nécessaire pour le profilage des attributs. Le deuxième méta-schéma (SCH2) servait pour la reconnaissance de concept décrit par les attributs de profilage et la détection des relations inter colonnes.

Nous détaillons dans ce qui suit comment nous les avons initier et créer.

5.2.1 Définition des expressions régulières

On a pu utiliser certaines expressions régulières existantes dans l'outil *Talend Data Quality* et nous avons défini d'autres expressions. Le tableau suivant (table 5.1) présente une liste non exhaustive d'expressions autres que celles présenter dans la chapitre 3.7. Nous avons utilisé 29 expressions régulières.

TABLE 5.1 – Exemple d'expressions régulières

Catégorie	**Sous-Catégorie**	**Expressions régulières**
FR_PHONE	French	$regex1 = \hat{}(0033\|\backslash\backslash + 33\|0)[1-689]([-.]?[0-9]\{2\})\{4\}\$$
	French	$regex2 = \hat{}[+][1-7689]([-,]?[0-9]\{2\})\{4\}\$$
US_PHONE	English	$(([2-9]\{1\})([0-9]\{2\})([0-9]\{3\})([0-9]\{4\}))\$$
FR_SSN	French	$\hat{}[12][0-9]\{2\}(0[1-9]\|1[0-2])(2[AB]\|[0-9]\{2\})[0-9]\{6\}([0-9]\{2\})?\$$
FR_ISBN	French	$\hat{}[ISBN]\{4\}[]\{0,1\}[0-9]\{1\}[-]\{1\}[0-9]\{3\}[-]\{1\}[0-9]\{5\}[-]\{1\}[0-9]\{0,1\}\$$
INT.PASSPORT		$\hat{}[A-Z0-9<]\{9\}[0-9]\{1\}[A-Z]\{3\}[0-9]\{7\}[A-Z]\{1\}[0-9]\{7\}[A-Z0-9<]\{14\}[0-9]\{2\}\$$
MASTER CARD NUMBER		$\hat{}5[1-5][0-9]\{14\}--\$$
BLOOD GROUP		$\hat{}(A\|A\backslash+\|A\backslash-\|AB\|B\|B\backslash+\|B\backslash-\|O\backslash+\|O\backslash-)\$$
CURRENCY		$\hat{}\backslash\$?(([1-9],)?([0-9]\{3\},)\{0,3\}[0-9]\{3\}\|[0-9]\{0,16\})(\backslash.[0-9]\{0,3\})?\$$
WEIGHT	kg	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(kg)\$$
	g	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(g)\$$
	t	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(t)\$$
	dg	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(dg)\$$
	mg	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(mg)\$$
DURATION	h	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(h)\$$
	min	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(min)\$$
	s	$\hat{}([0-9]\backslash d^{\star}(.\backslash d+)?)?(s)\$$
URL		$\wedge https? : //[[: alnum :]\_-][\backslash.][[: alnum :]\_-]*] * \backslash.(com \mid edu \mid org \mid net \mid info \mid eu \mid biz \mid mil \mid gov \mid aero \mid travel \mid pro \mid name \mid museum \mid coop \mid asia \mid [a-z][a-z]) \mid +(: [[: digit :]]+)?[: /[[: alnum :]\backslash.\_\#-]*] * /?\$$
IP_ADDRESS		$\wedge([01]?[[: digit :]][[: digit :]]? \mid 2[0-4][[: digit :]] \mid 25[0-5])\backslash.([01]?[[: digit :]][[: digit :]]? \mid 2[0-4][[: digit :]] \mid 25[0-5])\backslash.([01]?[[: digit :]][[: digit :]]? \mid 2[0-4][[: digit :]] \mid 25[0-5])\backslash.([01]?[[: digit :]][[: digit :]]? \mid 2[0-4][[: digit :]] \mid 25[0-5])\$$

5.2.2 Construction du dictionnaire des mots clés et du dictionnaire des données

Nous avons pu récupérer sur le web les données de plusieurs catégories. Nous avons défini 16 catégories en six langues dans le dictionnaire de données : des prénoms, le sexe et la civilité d'une personne, des noms de continents, de pays, de villes, de musées, de châteaux, d'aéroports, des mois, des jours, des noms d'animaux, des médicaments, des spécialités médicales, des tests médicaux et des noms de pharmacies (Hammou et al., 2013), (Souid et al., 2013).
Neuf catégories ont été défini dans le dictionnaires de mots clés en six langues aussi : adresse, organisations des études, organisations de santé, organisations de culture, organisations de sport, organisations de banque, organisations d'hôtellerie, organisations de services publiques et organisations de transport.

Cependant comme toutes informations venant du web, la qualité de l'information n'est pas garantie. Nous avons alors élaboré un processus de nettoyage des données du dictionnaire.

Ce processus consiste en différentes étapes :
— l'élimination des caractères spéciaux en utilisant des expressions régulières
— l'élimination des doublons
Nous avons aussi utilisé les outils *Talend Data Quality* et *Talend Data Integration* pour la correction des données. Pour chaque étape différents jobs ont été créés (exemple de job figure5.1) (Meddah et al., 2014).

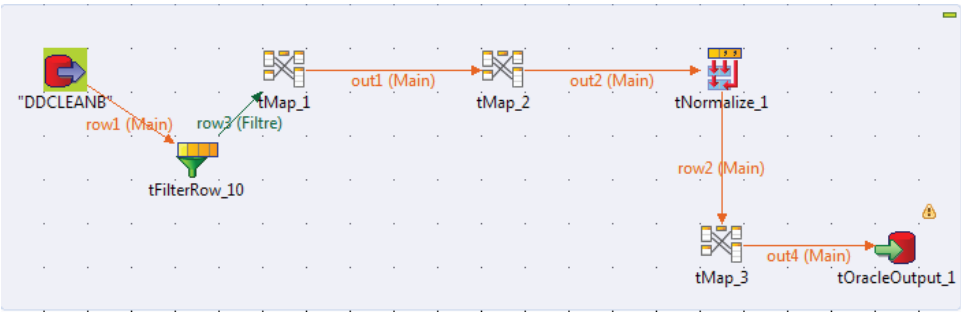


FIGURE 5.1 – Job de création du dictionnaire de données

L'étape correction permet de garantir un dictionnaire de meilleur qualité.

Le dictionnaire de données (table 3.5) contient à peu près 5,5 millions de lignes. Tandis que le dictionnaire de mots clés contient plus de 200 lignes.

5.2.3 Construction automatique du référentiel ontologique

Le référentiel ontologique représente une instance du méta-schéma SCH2. Afin de *construire* différentes instance du référentiel, nous avons fait appel à des ressources/standards existants sur le web (Liegl et al., 2010), (Bedini et al., 2008a). Parmi eux, nous avons choisi d'utiliser quatre standards :

— UBL : Les business modèles défini dans OASIS (Advancing open standards for the information society). OASIS est un consortium à but non lucratif qui entraîne le développement, la convergence et l'adoption de normes ouvertes pour la société mondiale de l'information. Il contient un ensemble de modèle décrivant des ressources en langage UBL (Universal Business Language). UBL est la langue universelle des affaires. C'est le produit d'un effort international visant à définir une bibliothèque libre de droit de documents commerciaux électroniques XML standard, tels que les bons de commande et les factures.

— HR-XML est une version open source du standard HR. Le but est de permettre un échange de données ressources humaines entre la communauté et aussi de simplifier l'intégration entre les fournisseurs de services de technologie HR.

— OAGIS (Open Applications Group Integration Specification) définit des contenus de modèles communs ainsi que des messages de communication entre des applications métiers.

Différentes versions du référentiel ont été générées en fonction de différents domaines (Bedini et al., 2007), (Bedini et al., 2008a), (Bedini et al., 2008b) (table 5.2).

TABLE 5.2 – Les différents standards

Standard	**Secteur d'activité**	**Nombre de mots**
UBL	Facturation, Commande	274
OAGIS	Industrie transversale	704
HR-XML	Ressources humaines	949

On extrait des standards, les principaux composants de SCH2. Nous avons défini pour cela une table de correspondance entre les éléments du standard UBL et les éléments du SCH2 (table 5.3).

TABLE 5.3 – Table de correspondance entre les éléments du standard UBL et les éléments du SCH2

Éléments des standards	Éléments du SCH2
xsd :element	Concept
ccts :Component	Attribute
ccts :DictionaryEntryName	SynAttribute
ccts :DataType	JavaTypeAttribute
ccts :Comments	Tags

Notons que ces ressources contiennent des éléments en commun vu le chevauchement de leur secteur d'activité.

Enrichissement du SCH2 avec des connaissances utilisateurs

Afin *d'enrichir* le référentiel avec le maximum d'informations, nous exploitons l'expérience utilisateur sur les données. Nous ajoutons alors les profils proposées par les utilisateurs sur la source (dans l'outil Talend Data Profiling) ainsi que sur les colonnes dans le référentiel. On sauvegarde la liste des indicateurs appliqués sur chaque attribut ainsi que les seuils (étape 1 du processus de reconnaissance sémantique des métadonnées figure 5.2).

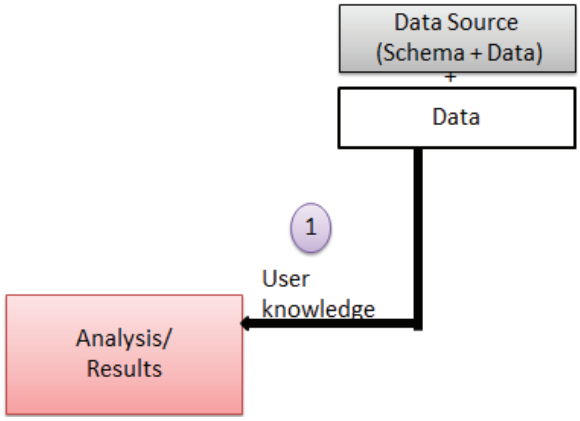


FIGURE 5.2 – Sauvegarder les connaissances utilisateur dans le référentiel

L'indicateur *TableFrequency* permet de définir une liste de valeurs utilisée dans la colonne analysée. Cette liste définit le domaine de cette dernière.

Notons aussi que les seuils (minimum et maximum) fixés par l'utilisateur sur chaque indicateurs, ramènent de l'information sur les attributs. Cette connaissance est ajoutée au référentiel sous forme d'un domaine de définition de l'attribut en

question. Une mise à jour de ces connaissances est réalisée au fur et à mesure de l'expérience utilisateur.

Ces connaissances permettent l'enrichissement de la partie contraintes et domaines de définition des attributs (table 5.4).

TABLE 5.4 – Enrichissement du référentiel avec les connaissances utilisateur

Éléments d'une analyse	**Éléments du SCH2**
Analysis	Analysis
Indicator	Indicator
Analysed element	Attribute
Indicator TableFrequency Thresholds Indicator	ConstraintIN
Results Indicator	Thresholds

5.3 Expérimentation

Afin de vérifier et valider notre approche, nous avons développé un prototype, prenant en entrée une source de données et les méta-schémas (SCH1 et SCH2). Les attributs du SCH2 construit les classes de SCH1. Un ensemble d'expressions régulières ainsi qu'un ensemble d'indicateurs sont aussi fournis en entrée. En sortie, n rapports dont la nouvelle structure sémantique, sont proposés.

La figure suivante (figure 5.3) présente aussi la partie "Nettoyage de données" avec ses deux étapes que nous allons traiter plus tard dans ce manuscrit. En sortie, une nouvelle source de données nettoyée est fournie.

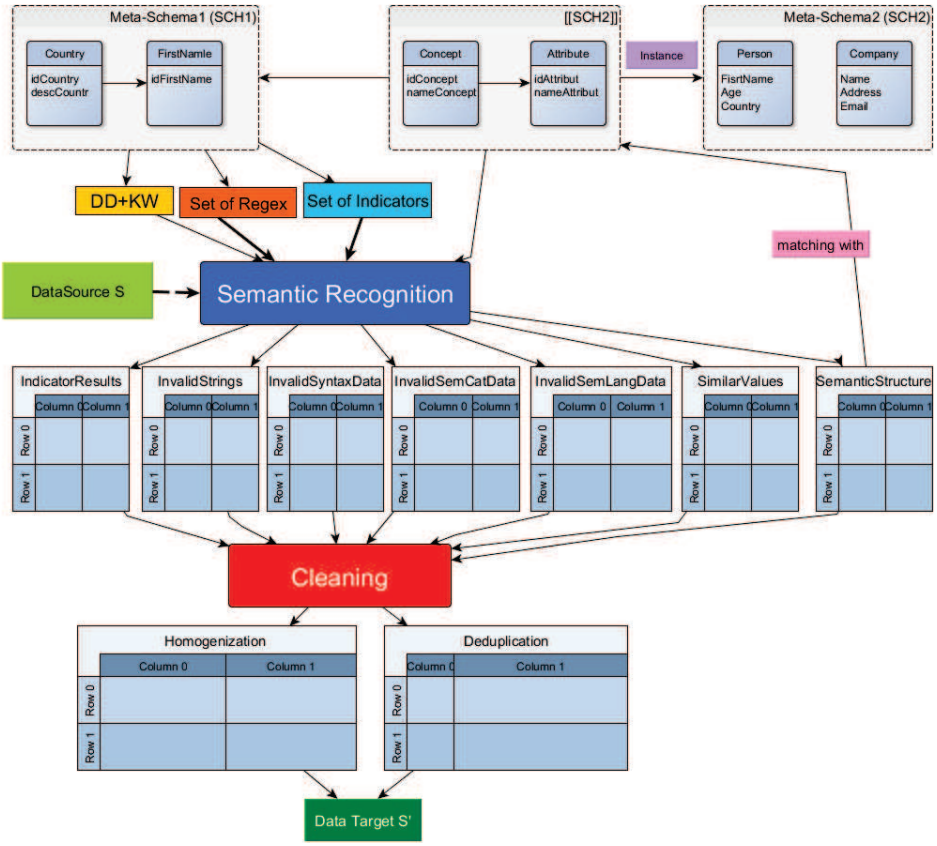


FIGURE 5.3 – Processus du profilage sémantique

Différents outils et langages sont utilisés dans cette expérimentation.

5.3.1 Outils et langages

Pour l'implémentation d'un prototype de notre approche, nous avons utilisé le SGBD Oracle 10g pour la création de la méta-schéma SCH1 (les expressions régulières, le dictionnaire des mots clés et le dictionnaire de données) et le langage PLSQL pour la définition des différentes fonctionnalités..

Oracle SGBD (ora, 2011) Oracle Database est un système de gestion de base de données relationnel (SGBDR). Il est aussi qualifié de système de gestion de base de données relationnel-objet (SGBDRO) depuis l'introduction du support du modèle objet dans sa version 8. Il est fourni par Oracle Corporation, il a été développé par Larry Ellison.

Il est parmi les SGBD les plus recommandés pour la gestion de gros volumes de données.

PL/SQL (Procedural Language / Structured Query Language) PLSQL[1] est un langage, conçu aux paradigmes procédural et structuré. Il est propriétaire, créé par Oracle et utilisé dans le cadre de bases de données relationnelles. Sa syntaxe générale ressemble à celle des langages Pascal et Ada. PL/SQL est disponible dans Oracle Database (depuis la version 7).

Il permet de combiner des requêtes SQL et des instructions procédurales (boucles ou conditions), dans le but de créer des traitements complexes destinés à être stockés sur le serveur de BD (objets serveur), comme des procédures stockées ou des déclencheurs.

Les dernières évolutions proposées par Oracle reposent sur un moteur permettant de créer et gérer des objets contenant des méthodes et des propriétés.

Pour la partie alignement de schéma sémantique avec le référentiel, nous avons utilisé :

— les index Lucène pour créer le référentiel (instance du méta-schéma).
— Jena pour la création et la manipulation des modèles ontologiques ainsi que l'enrichissement du référentiel.
— Protégé pour la visualisation des ontologies.

Jena Jena[2] est un framework Java pour créer des applications Web sémantique. Il offre une vaste bibliothèque Java pour aider les développeurs à développer du code qui gère RDF, RDFS, OWL et SPARQL en respectant les W3C recommandations. Jena comprend un moteur d'inférence à base de règles pour effectuer le raisonnement basé sur OWL et RDFS ontologies. Il contient une variété de stratégies de stockage pour stocker des triplets RDF en mémoire ou sur disque.

Historiquement Jena a été initialement développé par des chercheurs de HP Labs, à Bristol, Royaume-Uni, en 2000. Jena a été toujours un projet open-source, et a été largement utilisé dans une grande variété d'applications du web sémantique. En Novembre 2010, Jena a été adopté par l'Apache Software Foundation et en avril 2012, il a été considéré comme un projet de haut niveau.

Protégé[3] est outil open source pour la conception des ontologies. Développé par Stanford Medical Informatics Group à l'université Stanford. Il est un des plateformes les plus utilisées dans la conception et développement des ontologies. Il peut de lire et sauvegarder des ontologies dans la plupart des formats d'ontologies : RDF, RDFS, OWL, etc.

1. http ://fr.wikipedia.org/wiki/PL/SQL
2. https ://jena.apache.org/about_jena/about.html
3. http ://protege.stanford.edu/

Protégé est développé en Java. Il est gratuit et son code source est publié sous une licence libre (la Mozilla Public License). Il a une architecture extensible qui permet d'ajouter d'autres plugins.

Nous utilisons Protégé pour la validation des fichiers OWL générés par Jena et la visualisation des ontologies crées.

Index Lucène Lucène [4] est une API open source écrite en Java qui permet l'indexation et la recherche du texte. Il est utilisé dans certains moteurs de recherche.

C'est un projet de la fondation Apache mis à disposition sous licence Apache. Lucène est de haute performance et évolutif.

Index Lucène [5]

Les concepts fondamentaux de Lucène sont index, document, champ et terme :

— Un index contient une séquence de documents.
— Un document est une séquence de trames.
— Un champ est une séquence nommée de termes.
— Un terme est une chaîne.

Nous avons créé un index pour les catégories et un index pour la langue. La structure des index est : Catégorie/Valeurs. De même pour la langue : Langue/Valeurs.

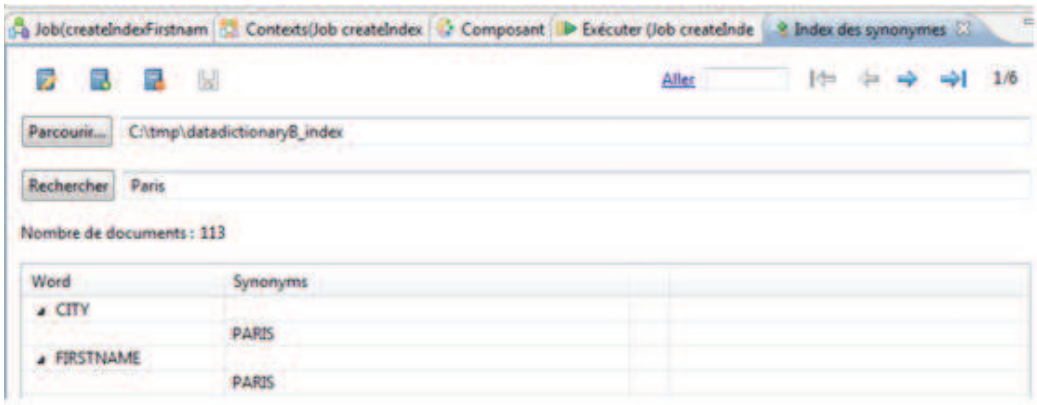


FIGURE 5.4 – Index Catégorie (recherche du mot “Paris")

La recherche du mot “Paris” dans l'index Lucène Catégorie renvoie 113 documents (figure 5.4) qui correspondent à ce mot. Talend offre un moteur de recherche basé sur les index Lucène.

4. http ://lucene.apache.org/
5. http ://lucene.apache.org/core/3_5_0/fileformats.html#Definitions

Le résultat de recherche montre que le mot paris peut appartenir à plus d'une catégorie tel le cas ici : Paris est le nom d'une ville (*City*) et le prénom (*FirstName*) d'une personne.

Un autre exemple de recherche mais cette fois dans l'index Langue (figure 5.5) renvoie que Paris s'écrit de la même manière dans les six langues proposées par l'index (English, French, German, Italian, Portuguese, Spanich).

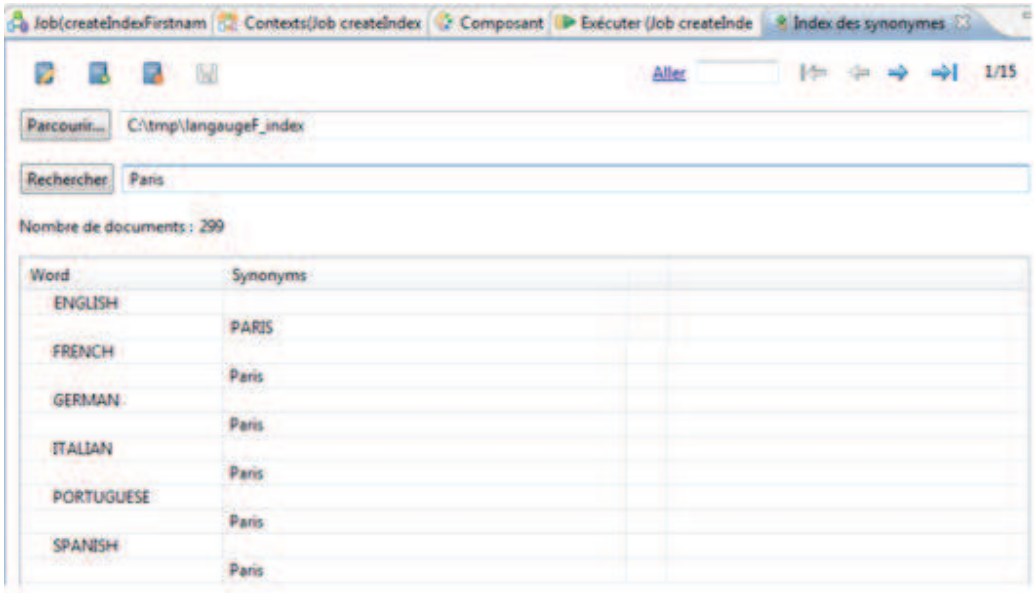
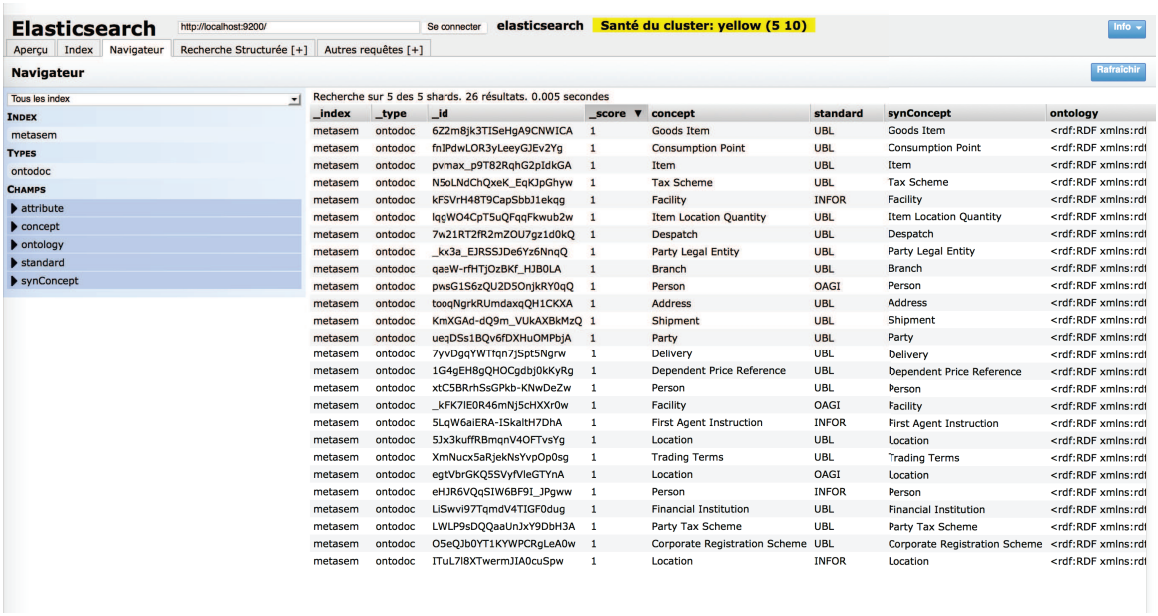


FIGURE 5.5 – Index Langue (recherche du mot "Paris")

Afin de sauvegarder ces différents index et faire un recherche rapide, nous avons utilisé *Elasticsearch*[6] (figure 5.6). Il permet de gérer les index Lucène et effectuer une recherche rapide dans les différents index.

6. http ://www.elasticsearch.org



_index	_type	_id	_score	concept	standard	synConcept	ontology
metasem	ontodoc	6Z2m8jk3TISeHgA9CNWICA	1	Goods Item	UBL	Goods Item	<rdf:RDF xmlns:rdf
metasem	ontodoc	fnlPdwLOR3yLeeyGJEv2Yg	1	Consumption Point	UBL	Consumption Point	<rdf:RDF xmlns:rdf
metasem	ontodoc	pvmax_p9TB2RqhG2pIdkGA	1	Item	UBL	Item	<rdf:RDF xmlns:rdf
metasem	ontodoc	N5oLNdChQxeK_EqKJpGhyw	1	Tax Scheme	UBL	Tax Scheme	<rdf:RDF xmlns:rdf
metasem	ontodoc	kFSVrH48T9CapSbbJ1ekqg	1	Facility	INFOR	Facility	<rdf:RDF xmlns:rdf
metasem	ontodoc	lqcWO4CpT5uQFqqFkwub2w	1	Item Location Quantity	UBL	Item Location Quantity	<rdf:RDF xmlns:rdf
metasem	ontodoc	7w21RT2fR2mZOU7gz1d0kQ	1	Despatch	UBL	Despatch	<rdf:RDF xmlns:rdf
metasem	ontodoc	_kx3a_EJRSSJDe6Yz6NnqQ	1	Party Legal Entity	UBL	Party Legal Entity	<rdf:RDF xmlns:rdf
metasem	ontodoc	qaaW-rfHTjOzBKf_HJB0LA	1	Branch	UBL	Branch	<rdf:RDF xmlns:rdf
metasem	ontodoc	pwsG1S6zQU2D5OnjkRY0qQ	1	Person	OAGI	Person	<rdf:RDF xmlns:rdf
metasem	ontodoc	tooqNgrkRUmdaxqQH1CKXA	1	Address	UBL	Address	<rdf:RDF xmlns:rdf
metasem	ontodoc	KmXGAd-dQ9m_VUkAXBkMzQ	1	Shipment	UBL	Shipment	<rdf:RDF xmlns:rdf
metasem	ontodoc	ueqDSs1BQv6fDXHuOMPbjA	1	Party	UBL	Party	<rdf:RDF xmlns:rdf
metasem	ontodoc	7yvDgqYWTtqn7jSpt5Ngrw	1	Delivery	UBL	Delivery	<rdf:RDF xmlns:rdf
metasem	ontodoc	1G4gEH8gQHOCgdbj0kKyRg	1	Dependent Price Reference	UBL	Dependent Price Reference	<rdf:RDF xmlns:rdf
metasem	ontodoc	xtC5BRrhSsGPkb-KNwDeZw	1	Person	UBL	Person	<rdf:RDF xmlns:rdf
metasem	ontodoc	_kFK7lE0R46mNj5cHXXr0w	1	Facility	OAGI	Facility	<rdf:RDF xmlns:rdf
metasem	ontodoc	5LqW6aiERA-ISkaltH7DhA	1	First Agent Instruction	INFOR	First Agent Instruction	<rdf:RDF xmlns:rdf
metasem	ontodoc	5Jx3kuffRBmqnV4OFTvsYg	1	Location	UBL	Location	<rdf:RDF xmlns:rdf
metasem	ontodoc	XmNucx5aRjekNsYvpOp0sg	1	Trading Terms	UBL	Trading Terms	<rdf:RDF xmlns:rdf
metasem	ontodoc	egtVbrGKQ5SVyfVleGTYnA	1	Location	OAGI	Location	<rdf:RDF xmlns:rdf
metasem	ontodoc	eHJR6VQqSIW6BF9I_JPgww	1	Person	INFOR	Person	<rdf:RDF xmlns:rdf
metasem	ontodoc	LiSwvi97TqmdV4TIGF0dug	1	Financial Institution	UBL	Financial Institution	<rdf:RDF xmlns:rdf
metasem	ontodoc	LWLP9sDQQaaUnJxY9DbH3A	1	Party Tax Scheme	UBL	Party Tax Scheme	<rdf:RDF xmlns:rdf
metasem	ontodoc	O5eQJb0YT1KYWPCRgLeA0w	1	Corporate Registration Scheme	UBL	Corporate Registration Scheme	<rdf:RDF xmlns:rdf
metasem	ontodoc	ITuL7I8XTwermJIA0cuSpw	1	Location	INFOR	Location	<rdf:RDF xmlns:rdf

FIGURE 5.6 – Visualisation des index avec Elasticsearch

Un document est créé pour chaque concept avec l'ensemble de ses attributs et synonymes attributs.

5.3.2 Les rapports du profilage

Nous avons appliqué le profilage sémantique de données sur la source présentée dans l'introduction 3.1 . Plusieurs structures de données ont été créé afin stocker les différents rapports.

5.3.2.1 Résultats d'indicateurs (Indicators results)

La première structure contient les résultats des différents indicateurs. Pour chaque colonne, nous avons des résumés statistiques (par exemple pourcentage de valeurs nulles, pourcentage des valeurs uniques), des résumés syntaxiques (le nombre de valeurs syntaxiquement invalides et le nombre de valeurs syntaxiquement valides) et des résumés sémantiques (le nombre de catégories détectées et le nombre de langues utilisées).

Un extrait du rapport est présenté dans le tableau suivant (table 5.6) :

TABLE 5.5 – Résultats des indicateurs appliqués à la source S (a)

ColumnNum	Indicator	Definition	Value/Percentage
	Indicator 01	Profiling Date	23/12/2014
	Indicator 02	Data Source	ExempleThese
	$Istat_{01}$	Sample DS	ExempleThese100
Column2	$Istat_{02}$	Total Values	100
Column2	$Istat_{03}$	Distinct Values	13
Column2	$Istat_{04}$	Duplicate values	13
Column2	$Istat_{05}$	Unique Values	0
Column2	$Istat_{06}$	Null Values	15
Column2	$Istat_{11}$	Maximum Length	9
Column2	$Istat_{12}$	Minimum Length	2
Column2	$Istat_{13}$	Average Length	4,29
Column2	$Istat_{14}$	Frequency Table	{Adam, A., Eve}
Column2	$Isyn_1$	Valid Syntaxe Values	65
Column2	$Isyn_2$	Invalid Syntaxe Values	20
Column2	$Isem_1$	Number of categories	2
Column2	$Isem_2$	Number of languages	6
Column3	$Istat_{02}$	Total Values	100
Column3	$Istat_{03}$	Distinct Values	8
Column3	$Istat_{04}$	Duplicate values	8
Column3	$Istat_{05}$	Unique Values	0
Column3	$Istat_{06}$	Null Values	25
Column3	$Istat_{11}$	Maximum Length	10
Column3	$Istat_{12}$	Minimum Length	10
Column3	$Istat_{13}$	Average Length	10
Column3	$Istat_{14}$	Frequency Table	{0653545577}
Column3	$Isyn_1$	Valid Syntaxe Values	75
Column3	$Isyn_2$	Invalid Syntaxe Values	0
Column3	$Isem_1$	Number of categories	2
Column3	$Isem_2$	Number of languages	0

TABLE 5.6 – Résultats des indicateurs appliqués à la source S (b)

ColumnNum	Indicator	Definition	Value/Percentage
Column5	$Istat_{02}$	Total Values	100
Column5	$Istat_{03}$	Distinct Values	7
Column5	$Istat_{04}$	Duplicate values	7
Column5	$Istat_{05}$	Unique Values	0
Column5	$Istat_{06}$	Null Values	20
Column5	$Istat_{11}$	Maximum Length	6
Column5	$Istat_{12}$	Minimum Length	1
Column5	$Istat_{13}$	Average Length	1,5
Column5	$Istat_{14}$	Frequency Table	{M, F}
Column5	$Isyn_1$	Valid Syntaxe Values	75
Column5	$Isyn_2$	Invalid Syntaxe Values	0
Column5	$Isem_1$	Number of categories	2
Column5	$Isem_2$	Number of languages	5
Column11	$Istat_{01}$	Total Values	100
Column11	$Istat_{04}$	Distinct Values	16
Column11	$Istat_{05}$	Duplicate values	16
Column11	$Istat_{03}$	Unique Values	0
Column11	$Istat_{02}$	Null Values	15
Column11	$Istat_{11}$	Maximum Length	10
Column11	$Istat_{12}$	Minimum Length	2
Column11	$Istat_{13}$	Average Length	9
Column11	$Istat_{14}$	Frequency Table	{30/03/2012}
Column11	$Isyn_1$	Valid Syntaxe Values	75
Column11	$Isyn_2$	Invalid Syntaxe Values	0
Column11	$Isem_1$	Number of categories	4
Column11	$Isem_2$	Number of languages	0

5.3.2.2 Structure sémantique (Semantic data structure)

La catégorie et la sous-catégorie (langue dans notre cas) dominantes sont utilisées pour définir un nouveau schéma sémantique pour la source de données. Ainsi, ce dernier rapport enregistre la correspondance entre la source de données initiale (nom initial de la colonne, type de données initial) et la nouvelle structure sémantique définie (nom de colonne sémantique, nouveau type de données et la langue).

La recherche approximative dans le dictionnaire de données est faite, en utilisant les index Lucène, avec par exemple l'algorithme de similarité Levenshtein et un seuil de 0.8.

Une instance de la structure sémantique de la source de données S est donnée dans la table 5.7.

TABLE 5.7 – Structure sémantique de la source S

Columns of Initial Schema		Columns of Semantic Schema		
ColumnNum	**DateType**	**SemanticColName**	**NewTYPE**	**Language**
Column1	String	Column1	String	
Column2	String	FirstName	String	
Column3	String	Phone_1	String	
Column4	String	Email	String	
Column5	String	Gender	String	French
Column6	String	Civility	String	French
Column7	String	City	String	French
Column8	String	Country	String	French
Column9	String	Continent	String	French
Column10	String	Integer	Number	
Column11	String	Column11	Date	
Column12	String	Phone_2	String	

5.3.2.3 Chaînes invalides (Invalid Strings)

Les valeurs avec des fautes d'orthographe sont automatiquement ajoutées à la structure des chaînes invalides (table 5.8). Ces valeurs sont vérifiées en utilisant les expressions de l'ensemble RE'.

TABLE 5.8 – Chaînes invalides

RowNumber	**ColumnNum**	**SemanticColName**	**InvalidValue**
2	Column2	FirstName	A.
4	Column7	City	Epinay / Seine
5	Column2	FirstName	Adamsss
7	Column2	FirstName	R.
14	Column2	FirstName	A.
16	Column8	Country	UK
19	Column7	City	LA

Rappelons que avons défini une chaîne valide ssi sa taille minimale est supérieur à trois caractères, sauf les chaînes reconnues à travers des expressions régulières (RE) telles que les colonnes *Gender* et *Civility*.

5.3.2.4 Valeurs invalides syntaxiquement (Invalid Syntax data)

La troisième structure (table 5.9) contient les valeurs invalides syntaxiquement et qui appartiennent à la catégorie inconnue. Par exemple, les valeurs "Toto", "Titi" sont inconnues puisqu'elles ne vérifient aucune des expressions régulières et n'existent ni dans le dictionnaire des mots clés, ni dans le dictionnaire de données.

TABLE 5.9 – Valeurs invalides syntaxiquement

RowNumber	ColumnNum	SemanticColName	Value
7	Column7	City	Epinay Villetaneuse
9	Column2	FirstName	Emir

5.3.2.5 Valeurs invalides sémantiquement par rapport à la catégorie (Invalid Semantic Category-Data)

Pour chaque colonne de la source de données, on peut reconnaître plus d'une catégorie. Donc, pour valider la catégorie dominante, nous choisissons celle qui a le plus grand pourcentage. Le pourcentage est calculé en fonction du nombre de valeurs qui appartiennent à une catégorie. Si nous avons deux catégories avec un même pourcentage ou un pourcentage très proche, on choisit un autre échantillon et on ré applique le profilage.

Les valeurs qui n'appartiennent pas à la catégorie dominante sont stockées dans la table T_4 comme des valeurs invalides sémantiquement par rapport à la catégorie (table 5.10).

TABLE 5.10 – Valeurs invalides sémantiquement par rapport à la catégorie

RowNumber	ColumnNum	SemanticColName	Value	Category
3	Column2	FirstName	Londres	City
7	Column4	Email	www.saint.fr	WebSite
13	Column12	Phone_2	www.lebon.fr	WebSite
15	Column2	FirstName	London	City
18	Column11	Column11	45	Integer

5.3.2.6 Valeurs invalides sémantiquement par rapport à la langue (Invalid Semantic Language-Data)

Le même processus est utilisé pour reconnaître la langue dominante dans une source de données. On commence par reconnaitre la ou les langues utilisées dans une

colonne, après on généralise pour toute la source. On stocke dans T_5 (table 5.11) les valeurs invalides sémantiquement par rapport à la langue dominante pour faciliter l'homogénéisation des valeurs au sein d'une même source.

TABLE 5.11 – Valeurs invalides sémantiquement par rapport à la langue

ColumnNum	SemanticColName	DominantLang	Value	Language
Column7	City	French	London	English
Column7	City	French	Beijing	English
Column8	Country	French	United-Kingdom	English
Column8	Country	French	China	English
Column9	Continent	French	Asia	English
Column9	Continent	French	America	English

5.3.2.7 Valeurs similaires (Similar Semantic values)

Les données rapprochées au DD d'une manière approximative sont stockées dans T_6 (table 5.12) afin de les corriger par rapport au dictionnaire de données dans la phase de nettoyage et proposer la valeur valide.

TABLE 5.12 – Valeurs similaires

ColumnNum	SemanticColName	Value	ValueInDD	LangaugeVDD
Column1	FirstName	Adamsss	Adam	
Column1	FirstName	Adem	Adam	
Column7	City	Pari	Paris	French
Column7	City	Londre	Londres	French
Column8	Country	Frence	France	English

5.3.3 Alignement sémantique du schéma

Afin de reconnaitre le schéma traité, nous allons aligner les attributs reconnus sémantiquement par le profilage avec un référentiel ontologique. Ce dernier peut couvrir différents domaines. Nous avons construit à chaque domaine une ontologie extraite des modèles métiers existants dans les standards et ressources retrouvés dans le web.

Nous avons généré différents référentiels selon le domaine d'activité. Nous avons supposé qu'un concept est créé si et seulement si il est décrit par au moins deux attributs (figure 5.13).

TABLE 5.13 – Nombre de concepts générés avec chaque standard

Standards	Nombre de concepts générés
UBL	220
OAGIS	118
HR-XML	370

Pour la rapprochement lexicographique, on utilise Elasticsearch qui permet de rapprocher le schéma sémantique par rapport aux attributs des différents concepts existants dans le référentiel.

```
## Catalog: crm
       ## Tables
                    ## Columns
- [Iron Man] loaded [], sites []
--------runMultiQuery----------------------------------------
>>> Search for [contract] in field [concept]...
>>> Search for [number,offer_name,duration,Tp_cd,owner,beneficiary,Begin_dt,End_dt] in field [attribute]...
Total hits: 117
0,244897   Contract Extension       UBL      [Options Description, Maximum Number Numeric, Option Validity Period, Minimum Number Numeric, Renewal]
0,174729   Contract Execution RequirementUBL      [Name, Description, Execution Requirement Code]
0,156020   Contract                 UBL      [Validity Period, Note, Contract Document Reference, ID, Contractual Delivery, Version ID, Nomination Time,
0,105919   Period                   UBL      [End Date, End Time, Description, Start Time, Duration Measure, Description Code, Start Date]
0,097513   Contract Price           OAGI     [Location, Base Currency Amount, Target Base UOM Quantity, Contract Report Amount, Cost Report Amount, Cont
0,080421   Account Information      INFOR    [Name On Account, Number, Type]
0,079057   Sales Task               INFOR    [Assignee Type, Actual Effort Duration, Opportunity Reference, Parent Task ID, Repeating Frequency Code, As
0,075575   Account Information      OAGI     [Name On Account, Type, Number, Issuer Name]
0,065405   Time Tolerance           INFOR    [Over Duration, Under Duration]
0,065405   Time Tolerence           OAGI     [Under Duration, Over Duration]
0,057811   Event                    INFOR    [Description, Duration]
0,046248   Down Time                OAGI     [Unit Cost, Percent, Duration]
0,046248   Resource Cost Rule       OAGI     [Duration, Burden, Rate]
0,046248   Tool Actual              INFOR    [Tool, Down Time Duration]
0,045955   Meter                    UBL      [Meter Property, Meter Constant Code, Meter Constant, Meter Reading, Total Delivered Quantity, Meter Number
0,044080   Customer Account ReferenceOAGI      [Type, Number]
0,040878   Coverage By Date         OAGI     [Duration, Expire Date Time, Start Date Time, Renew Threshhold Duration]
0,037851   Card Account             UBL      [Validity Start Date, Issue Number ID, Chip Application ID, Network ID, Card Type Code, Expiry Date, Holder
0,035264   Lot Serial               OAGI     [Lot IDs, Serial Number]
0,034823   Asset Hospitality        OAGI     [Open Bay Indicator, Room Communication, Living Room Type, Primary Guest Middle Name, Room Area Measure, Ro
0,034686   Labour Actual            INFOR    [Work Time Period, Break Time Duration, Labour]
0,034686   Machine Actual           INFOR    [Down Time Period, Down Time Duration, Machine]
0,033061   Marketing Program        INFOR    [Offer Description, ID, Opportunity Type, Opportunity Method Code, Discount]
0,031170   Serialized Lot           INFOR    [Serial Number, Disposition, Item Quantity, Lot, Parent Serial Number]
0,030725   Address                  UBL      [Postal Zone, Building Number, Country Subentity Code, Address Type Code, City Subdivision Name, ID, Depart
0,028905   Coverage By Usage        OAGI     [Start Usage, Warranty Usage Duration, Expire Usage, Renew Threshhold Usage]
0,028615   Notification Requirement UBL      [Notification Period, Notification Type Code, Notify Party, Pre Event Notification Duration Measure, Post E
0,028469   Transportation Service   UBL      [Name, Transport Event, Transport Equipment, Tariff Class Code, Sequence Numeric, Commodity Classification,
0,026448   Lot Serial               INFOR    [Quantity, Serial Number, Lot IDs, Discrepancy Code]
0,026448   M S D S                  OAGI     [Reference URL, Publish Date, Sheet Number ID]
```

FIGURE 5.7 – Recherche dans les index avec Elasticsearch

5.4 Conclusion

Le prototype défini prend en entrée l'ensemble des données du méta-schéma et une source de données. Il produit en sortie une nouvelle structure sémantique pour la source ainsi qu'un ensemble de rapports d'anomalies. Ces derniers présentent aux utilisateurs un aperçu sur la qualité de leurs sources. Ils permettront une meilleure détection et correction des différentes anomalies.

Deuxième partie

Nettoyage des données

Introduction

Force est de constater que l'on assiste aujourd'hui à une véritable explosion de la masse de données à traiter et la grande difficulté consiste à bien appréhender leur diversité. Ces données constituent un gisement de connaissances, qui bien traité et analysé, peut s'avérer un puissant atout de performance dans une entreprise ou une organisation ; mais qui, à l'inverse, mal appréhendé, peut avoir des répercussions très négatives.

Pour la démarche qualité de données, après avoir détecté l'existence des anomalies, il est intéressant de proposer leur correction. Cette étape permet d'améliorer la qualité des données. Elle consiste à appliquer des transformations sur les données d'une source pour résoudre des problèmes de format, de codage et d'incohérence.

La reconnaissance de schéma facilite la détection et la correction des différentes anomalies pouvant exister au sein d'une source de données. Les résultats du profilage sémantique des données permet de détecter les différentes anomalies existantes dans les données et faciliter le choix des attributs clés pour l'élimination des doublons et similaires en offrant une structure sémantique des données. Parmi les actions de correction, nous avons défini deux grandes actions de correction des données : l'homogénéisation et l'élimination des doublons et similaires.

Pour l'homogénéisation, les rapports d'anomalies fournis par le profilage sémantique des données renvoient différent types d'anomalies :

— plusieurs formats, codes et langues sont utilisés au sein d'une même colonne (catégorie).
— les données reconnues avec une recherche approximative dans le dictionnaire de données lors du profilage sont à standardiser avec des valeurs valides.
— l'étude des dépendances fonctionnelles permet le traitement de valeurs nulles présentes dans la source.

Actuellement, un grand nombre d'applications utilisent des données hétérogènes et distribuées de qualité variable. Le besoin d'intégration de données et d'évaluation de la qualité des données se fait de plus en plus ressentir. Afin de redonner d'avantage de sens aux données rassemblées, nous abordons la problématique complexe de l'élimination de similaires dans les sources de données. En effet, le développement

de nouveaux outils pour traiter les données similaires nécessite l'approfondissement, d'une part, des deux concepts de base *Match* (comparer) et *Merge* (fusionner), et d'autre part, le développement d'autres algorithmes pour parcourir et comparer les données dans des environnements séquentiels et parallèles.

L'élimination des doublons et similaires permet de nettoyer la source, des enregistrements égaux ou proches.

Cette problématique consiste à rapprocher les données de deux enregistrements et fusionner les données similaires afin de construire un nouvel enregistrement unique et éventuellement plus complet.

Dans la suite, le mot "tuple" sera utilisé comme une alternative au mot "enregistrement".

Chapitre 6

Homogénéisation de données

Sommaire

6.1 Introduction

Une fois la structure sémantique d'une source et les rapports d'anomalies produits, nous procédons aux différentes actions de nettoyage.

Les rapports d'anomalies fournis par le profilage montrent différents problèmes d'homogénéisation telles que l'utilisation de plus d'un format, code ou langue dans une même colonne. Ils présentent aussi les valeurs similaires (T_6) aux valeurs existantes dans le dictionnaire de données. Ces données ont besoin d'une certaine standardisation.

La première étape de nettoyage (mono-colonne), consiste à corriger les données colonne par colonne. Les relations sémantiques inter colonnes, constituent le deuxième volet de notre processus de nettoyage. En effet, certaines relations d'interdépendances stockées dans le schéma SCH2, devront être validées après l'homogénéisation des colonnes. Des liens de dépendance pourront ainsi être confirmés.

6.2 Homogénéisation des données

La première étape dans l'homogénéisation est la standardisation (correction syntaxique et codification). Ensuite, pour les données déjà corrigées syntaxiquement, on propose une unification dans la sous-catégorie dominante ; enfin, une conversion des données vers le nouveau type de données. Ce dernier a été détecté par le profilage sémantique des données.

Ces types de correction sont appliqués colonne par colonne, cependant, notre objectif est de détecter aussi les relations entre colonnes et corriger les données ligne par ligne. Un traitement des dépendances fonctionnelles entre attributs est alors proposé.

6.2.1 Correction syntaxique des données

La standardisation des données se fait en rapprochant les valeurs invalides et similaires par rapport au dictionnaire de données (DD). Nous proposons des corrections pour les valeurs mal orthographiées telles que *Adamssss* et celles proches du DD telles que *Londre, Pari* existants dans la source de données. Les résultats de profilage sémantique des données, facilitent cette tâche étant donné que nous avons l'information sémantique sur la nature de la colonne en question (catégorie et sous-catégorie). La standardisation se fait selon l'algorithme suivant (algo 18) :

Algorithm 18 Algorithme de correction syntaxique

```
Algorithm SyntaxDataCorrection
//Standardize the invalid data.
Input :
    S Data source
    SCHS (semantic structure) : Cat (Data category), Lang (Data language)
    Ddd Dictionaries (DD, KW)
    param (invalidValue or oldValue)
Output :
    Cleaned data source SC
    Invalid data source SI
Begin
for each Ci from S (i=1 :n) do
  for each vj from Ci (j=1 ;m) do
    if vj' ← f(vj, Cati, Ddd) then
      update(vi, vj', SC)
    else
      update(vi, param, SC)
      add(vi, SI)
    end if
  end for
end for
End Algorithm SyntaxDataCorrection
```

La fonction f prend en entrée trois paramètres : la valeur invalide, sa catégorie sémantique ainsi que les dictionnaires de données. f consiste à chercher la valeur la plus proche de v_j dans le DD afin de la corriger. En fonction de la catégorie Cat_i reconnue de v_j, la recherche dans le dictionnaire est optimisée.

Si la valeur n'existe pas dans les dictionnaires, l'utilisateur a le choix entre deux possibilités, représentées par la variable *param*. Soit la valeur est remplacée par la chaîne "invalidValue", soit la même valeur ("oldValue") est conservée.

Exemple 6.1 : Correction syntaxique

Par exemple, la recherche approximative dans le Ddd se fait en utilisant des mesures de similarité telles que Jaro-Winkler avec un seuil de 0.8 (table 6.7).

TABLE 6.1 – Correction syntaxique des données

Old Schema		**New Semantic Schema**	
ColNum	**OldValue**	**SemanticColName**	**NewValue**
Column2	Adamsss	FirstName	Adam
Column6	Pari	City	Paris
Column6	Londre	City	Londres
Column7	Frence	Country	France

6.2.2 Codification des données

Différents codes et formats peuvent être utilisés dans une même source. Nous proposons une unification de la codification des valeurs. La reconnaissance sémantique de la colonne permet de valider que le format le plus utilisé est bien un format correct. Une transformation vers le format dominant est proposée.

Le processus consiste en deux étapes présentées dans l'algorithme (algo 9.14). La fonction g permet de transformer une valeur invalide v_j vers une valeur valide v'_j selon le format dominant. Pour certaines colonnes, le format dominant est spécifié dans leurs contraintes. Pour d'autres c'est en fonction des analyses que le format dominant est détecté. Par exemple, l'indicateur "High Table Frequency" renvoie les valeurs les plus fréquentes dans une colonne. On ajoute la nouvelle valeur valide v'_j dans la nouvelle source SC.

Algorithm 19 Algorithme de codification des données

```
Algorithm DataCodification
// Put all data in the same format or code.
Input :
    S Data source
    SCHS (semantic structure) : Cat (Data category)
    Format (data format from a Cat in the Lang)
    param (invalidValue or oldValue)
Output :
    Cleaned data source SC
    Invalid data source SI
Begin
for each C_i from S (i=1 :n) do
  for each v_j from C_i (j=1 ;m) do
    if v_j' ← g(v_j, Cat_i, Format) then
      update(v_i, v_j', SC)
    else
      update(v_i, param, SC)
      add(v_i, SI)
    end if
  end for
end for
END Algorithm DataCodification
```

La reconnaissance de catégorie sémantique permet de reconnaître les colonnes qui ont besoin d'une homogénéisation de format. On a créé un index "FormatHomogenization+Cat_i" pour chaque catégorie contenant le format dominant comme clé et les valeurs correspondantes comme synonymes.

La structure de l'index "FormatHomogenizationGender" est la suivante :

TABLE 6.2 – Index "FormatHomogenizationGender"

Format	**Synonymes**
F	Female, Féminin, Femminile, Mujeres Woman, Femme, Frauen, Donna, Mulheres, Mujeres, 0
M	Male, Masculin, Maschio, Masculino Man, Homme, Man, Uomo, Man, Man, 1

Exemple 6.2 : Codification des données

Pour la catégorie *Gender* le format choisi est F, M. F pour désigner le féminin et M pour le masculin (table 6.3).

TABLE 6.3 – Codification des données

Old Schema		**New Semantic Schema**	
ColNum	**OldValue**	**SemanticColName**	**NewValue**
Column4	Female	Gender	F
Column4	Male	Gender	M
Column4	0	Gender	F
Column4	1	Gender	M
Column5	Mrs	Civility	Mme
Column5	Mr	Civility	M.

6.2.3 Unification en une même sous-catégorie

Plusieurs sous-catégories peuvent être utilisées au sein d'une même source. Nous traitons dans ce qui suit l'unification des données dans une même sous-catégorie *langue*.

Le profilage sémantique permet la détection de la langue dominante dans la source. Une traduction vers la langue dominante est proposée afin d'homogénéiser les données au sein d'une même source. Le processus de traduction se résume dans l'algorithme (algo 20).

Algorithm 20 Algorithme de traduction en langue dominante

```
Algorithm TranslateToDominantLanguage
// Translate values to the dominant language.
Input :
    S Data source
    SCHS (semantic structure) : Cat (Data category), Lang (Data language)
    DD data dictionary
    param (invalidValue or oldValue)
Output :
    Cleaned data source SC
    Invalid data source SI
Begin
for each C_i from S (i=1 :n) do
  for each v_j from C_i (j=1 ;m) do
    if v_j' ← h(v_j, Cat_i, DD) then
      update(v_i, Lang_j', SC)
    else
      update(v_i, param, SC)
      add(v_i, SI)
    end if
  end for
end for
END Algorithm TranslateToDominantLanguage
```

La fonction h permet de traduire chaque valeur invalide v_j dans la langue dominante $Lang_i$ (reconnue à partir de la structure sémantique proposée à S) en utilisant le dictionnaire de données.

Le principe consiste à chercher v_j dans le DD et choisir la valeur correspondante dans la langue dominante $Lang_j$. On ajoute la nouvelle valeur valide v'_j dans la source en sortie.

Exemple 6.3 : Traduction des données dans la langue dominante
La source S contient des valeurs en français et autre en anglais. Exemple, la colonne sémantique *City* contient des valeurs en anglais telles que “London” et “Beijing”. Ces valeurs ont besoin d’être traduit dans la langue dominante qui est le français dans notre cas (table 7.6).

TABLE 6.4 – Traduction dans la langue dominante

Old Schema		**New Semantic Schema**		
ColNum	**OldValue**	**SemanticColName**	**DomLanguage**	**NewValue**
Column6	London	City	French	Londres
Column6	Beijing	City	French	Pékin
Column7	United-Kingdom	Country	French	Royaume-Uni

Notons que l'unification dans d'autres sous-catégories est possible telle que la conversion dans une même unité de mesure pour les catégories *Temperature, Weight, Duration* ou *DeviceSize*. Exemple, convertir la température de degré Celsius en degré Fahrenheit.

6.2.4 Conversion des données vers un nouveau type de données

Le profilage sémantique des données permet de reconnaître non seulement le nom sémantique des colonnes mais aussi le type des données. Une conversion vers le nouveau type est proposée. Le tableau ci-après (table 6.5) définit les différents types de conversions proposées.

TABLE 6.5 – Conversion de type de données

	String	Date	Numérique
String	x	x	x
Date	x	x	▬
Numérique	x	▬	x

Le principe de l'algorithme est défini ci-après (algo 21) :

Algorithm 21 Conversion du type des données

Algorithm DataTypeConversion
// Convert data to the correct data type.
Input :
S Data source
T_7 (semantic structure) : Cat (Data category), newType (discovred data type)

Output :
Cleaned data source SC
Invalid data source SI
Begin
for each C_i from S (i=1 :n) **do**
for each v_j from C_i (j=1 ;m) **do**
if v_j' ← k(v_j, Cat_i, newType) **then**
update(v_i, v_j', SC)
end if
end for
end for
END Algorithm DataTypeConversion

La fonction k permet de convertir les données d'un type vers un autre type de données. Différents types de conversions sont possibles et donc différentes versions de la fonction k (*to-numeric*, *to-date* ou *to-char*) sont développées.

Exemple 6.4 : Conversion du type de données
Dans la source de données S (source 3.2), les données sont toutes vues en tant que chaînes de caractères. La reconnaissance du schéma sémantique de la source, permet de reconnaître des dates et des numériques données sous le type *string* des contrôles sémantiques peuvent se faire. Par exemple, le format de date le plus utilisé dans la source est *jj/mm/aaaa*. Une homogénéisation dans ce format est proposée pour la colonne de catégorie *Date* (table 6.6).

TABLE 6.6 – Conversion des données en fonction du nouveau type

Old Schema			**New Semantic Schema**		
ColNum	**OldValue**	**OldType**	**SemanticColName**	**NewType**	**NewValue**
Column11	02 mars 2014	String	Column11_Date	Date	02/03/2014
Column11	30-2-2014	String	Column11_Date	Date	invalidValue
Column10	1000	String	Column10	Numeric	12

Les données invalides Comme présenté dans l'exemple précédent, lors de la conversion de type de la colonne 11 de string vers date, renvoie la valeur "invalidValue" pour la valeur "30-02-2014". 30 n'existe pour le mois de Février.

Autre forme d'invalidité est quand la valeur à standardiser, ou à codifier ou à traduire n'existe pas dans les deux dictionnaires KW ou DD (table 6.7).

TABLE 6.7 – Souce SI des valeurs invalides

RowNumber	ColumnName	SemanticColumnName	InvalidValue
6	Column6	City	Epinay sur seine
8	Column11	Column11_Date	31/11/2014
9	Column2	FirstName	Emir

L'utilisateur pourra corriger certaines valeurs invalides, stockées dans SI à la fin du processus d'homogénéisation.

En sortie de ces différentes étapes, les valeurs déjà corrigées, stockées dans SC, seront fusionnées avec les correction de l'utilisateur pour les valeurs invalides afin d'avoir une source homogène pour les prochaines étapes de nettoyage des données.

6.2.5 Parallélisation du processus d'homogénéisation des données

Afin de traiter les différentes étapes d'homogénéisation des données, nous avons pensé à les paralléliser afin de garantir une certaine performance.

Les sources sont souvent volumineuse et ce processus de parallélisation pourra accélérer le traitement de l'homogénéisation. Pour cela, nous avons utilisé la méthode MapReduce [1] (Dean and Ghemawat, 2004), (Zhang), (Rehab-Adjout and Boufarès, 2014).

La source de données est divisé en différents blocs. Pour chaque bloc, on homogénéise les données selon les quatre fonctions définies ci-dessus (correction syntaxique, codification, unification selon la sous-catégorie, conversion de type de données). Les deux dictionnaires (DD et KW) sont distribués sur chaque serveur.

Une fois les données sont corrigées, une nouvelle source homogénéisée est produite en sortie.

1. http ://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

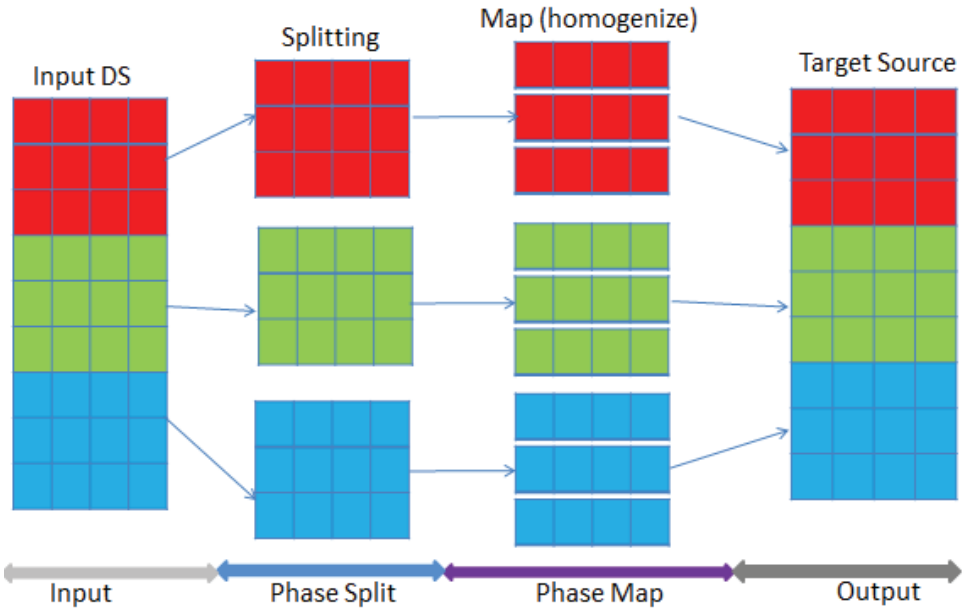


FIGURE 6.1 – Processus MapReduce pour l'homogénéisation des données

L'algorithme suivant (algo 22, 23) permet de décrire le processus d'homogénéisation.

Algorithm 22 Main Function of the MapReduce Algorithm

Main Function
Input : Data source S
Output : Data source S_j // j=1..nbBlocks
Begin
nbBlocks ← Size(S)/BlockSize
S_j ← BlockDivide(S, BlockSize)
Emit(j, S_j)
END Main Function

Algorithm 23 Mapper Function of the MapReduce Algorithm

```
Mapper Function
Input :
    S_j (Data source) // j=1... nbBlocks
    Ddd (Dictionaries DD and KW)
Output : S_j_updated (Updated data source)
Begin
for each j=1 to nbBlocks do
  for each i=1 to nbAttributes do
    v_ji ← get(S_j, j, i)
    v'_ji ← homogenize(v_ji, Ddd)
    S_j_updated ← set(j, i, v'_ji)
    Emit(j, S_j_updated)
  end for
end for
END Mapper Function
```

La fonction *homogenize()* permet d'homogénéiser les valeurs d'une ligne selon les quatre fonctions (correction syntaxique, codification, unification selon la sous-catégorie, conversion de type de données).

Algorithm 24 Fonction homogenize()

```
Homogenize Function
Input :
    Ddd (Dictionaries DD and KW)
    tuple (Tuple from the block)
Output : Cleaned Data List CListC
Begin
for each v_j from tuple //j=1..n do
  correctSyntaxValues(v_j, Ddd)
  codificationValues(v_j, Ddd)
  unificationValues(v_j, Ddd)
  conversionDataType(v_j, Ddd)
end for
return tuple
End Homogenize Function
```

Cette technologie permettra de corriger plus rapidement les données dans une même source.

Un autre type de traitement des anomalies des données est la correction d'un ensemble de colonnes en utilisant les liens déduits inter colonnes. Nous vérifions dans la section suivante l'ensemble des dépendances sémantiques définies entre attributs.

Une fois les dépendances sont vérifiées, ces dernières pourront servir à corriger certaines anomalies telles que les valeurs nulles.

6.2.6 Traitement des dépendances sémantiques inter colonnes

Les dépendances sémantiques entre les colonnes (telles que les dépendances fonctionnelles (DF)) représentent un autre type de contraintes à exploiter dans les sources de données. L'exploitation consiste à vérifier les DF, recommander les DF probables, corriger les données en fonction des DF et aussi recommander les attributs clés de dédoublonnage. Par exemple, certaines valeurs inconnues au sein d'une seule colonne, peuvent être déduites à partir de ces dépendances.

Afin de détecter et traiter certaines dépendances, nous proposons l'algorithme suivant (algo 25).

Les actions d'homogénéisation réalisées auparavant, peuvent faciliter la détection des dépendances ; ainsi l'ordre de le traitement des différentes anomalies est important.

Algorithm 25 Extraction des dépendances sémantiques

```
Algorithm Semantic Dependency
Input : Data Source S
Output : Cleaned Target Data SC
Begin
checkDF(S) //check dependencies
hundleDF(S) //hundle valid and not valid dependencies
return SC
End Algorithm Semantic Dependency
```

A partir des dépendances sémantiques définies pour chaque concept (chapitre 4, section 4.2, table 4.7), on vérifie ces dépendances afin de proposer celles valides et non valides pour une source.

Le principe de l'algorithme de vérification des DF (X $\rightarrow$ Y) est comme suit (Hammou et al., 2013) :

- Pour chaque DF, on calcule le nombre d'occurrence de l'attribut X dans la source.
- X et Y ne peuvent pas être égaux et ils devront être différents de nul.
- En fonction de l'occurrence on peut décider que la dépendance déclarée est vérifiée ou pas.

Algorithm 26 Vérification des dépendances sémantiques (DF)

Algorithm checkFD
Input :
    Data Source S
    X, Y (attributes)
Output :
    validDF (table to save valid DF)
    notValidDF (table to save not valid DF)
Begin
//calculate the occurence of X (left part of the DF) with removing some cases such as X and Y are not null values and X is not equal to Y
if (X != null or Y !=null and X !=Y) **then**
  $n \leftarrow$ countOccurrence(X, S)
end if
if $n = 1$ **then**
  validDF $\leftarrow$ getValidDF() //Gets values that satisfy (X $\rightarrow$ Y).
end if
if $n > 1$ **then**
  notValidDF $\leftarrow$ getNotValidDF() //Gets values which do not satisfy (X $\rightarrow$ Y).
end if
End Algorithm checkFD

A la fin de cette étape, une vue est proposée à l'utilisateur qui montre les dépendances fonctionnelles possibles et peu probables (table 6.8).

TABLE 6.8 – Les dépendances sémantiques possibles et peu probables entre les colonnes de S

		1	2	3	4	5	6	7	8	9	10	11
1	FirstName											
2	Phone_1	X										
3	Email				X							
4	Gender	X				✓						
5	Civility	X										
6	City					X		✓				
7	Country					X			✓			
8	Continent											
8	Column10	X					X	X			X	
10	Column11	X								X		
11	Phone_2											

Un attribut qui est défini sur un ensemble de valeurs ($\frac{Istat04}{Istat02} \approx 0$) ne pourra pas déterminer un attribut défini avec un très grand nombre de valeurs $\frac{Istat04}{Istat02} \approx 1$. Cet attribut ne peut pas être utilisé comme étant partie gauche de la dépendance.

Exemple 6.5 : Dépendances peu probables
L'attribut *FirstName* ne pas peut dépendre de l'attribut *Gender*. Ce dernier est défini sur un ensemble de deux valeurs {F, M} tandis que *FirstName* peut contenir un ensemble plus grand de valeurs (table 6.9).

TABLE 6.9 – Non dépendances entre deux attributs de domaines différents

Gender	$\nrightarrow$	**FirstName**
F	$\nrightarrow$	Dominique
M	$\nrightarrow$	Dominique
F	$\nrightarrow$	Aïcha
M	$\nrightarrow$	Adam

En fonction du bilan proposé, certaines dépendances permettent de corriger différentes anomalies entre colonnes. Nous allons nous intéresser dans cette thèse à la problématique des valeurs nulles.

Afin de traiter les valeurs nulles deux possibilités se posent : soit les correspondances définies sont vérifiées (X et Y existant bien dans le DD ou KW), soit on utilise les données stockées dans "Invalid data source". Ces données n'existent pas dans les dictionnaires mais bien orthographiées donc elles peuvent servir à enrichir des valeurs nulles. Tant que une dépendance existe entre colonne, on devrait vérifier toutes les données. On remplace alors dans la partie gauche de dépendance (Y), toute valeur par la donnée correspondante renvoyée par X.

Algorithm 27 Traitement des valeurs nulles en fonction des dépendances

```
Algorithm hundleFD
Input
    Data Source S
    X, Y (DF attributes)
Output: Cleaned Target Data SC
Begin
verifyDetectedDF(X, Y) // checks in the DD the correspondance between X and Y
// loops on Y attribute and corrects data with the corresponding values according to the X value.
for each Yi from S (i=1 :n) do
  replace(Yi, Y(X)) //hundles the different cases : Y=null and X=Y
end for
return SC
End Algorithm hundleFD
```

Exemple 6.5 : Dépendances fonctionnelles et traitement des valeurs nulles

Rappelons qu'après les premières étapes de nettoyage de ce chapitre, les colonnes *Gender* et *Civility* sont homogénéisées. Pour l'attribut *Gender* on garde les valeurs {F, M} : F pour féminin et M pour masculin. Pour l'attribut *Civility*, on unifie selon deux valeurs aussi {Mme, M.} avec Mme→F et M.→M.

Afin de traiter certaines valeurs nulles dans la colonne *Gender*, on utilise la dépendance détectée entre *Civility* et *Gender*.

La correspondance valide renvoyée par les dépendances vérifiées est donnée dans le tableau suivant (table 6.10).

TABLE 6.10 – Dépendance DF valides

X	**Y**	**Occurrence**
Mme	F	1
M.	M	1

Le traitement des valeurs nulles, en fonction de la DF *Civility* → *Gender*, appliqué à la source S renvoie le résultat suivant (table 6.11).

TABLE 6.11 – Traitement des DF de la source S

RowNumber	DF	ValueX	ValueY
4	*Civility* → *Gender*	Mme	F
8	*Civility* → *Gender*	Mme	F
9	*Civility* → *Gender*	M.	M

Une autre version de l'algorithme de détection des DF a été développé en utilisant l'algorithme d'élimination des doublons et similaires. Cette problématique sera traitée dans le chapitre suivant.

6.3 Conclusion

L'homogénéisation des données assure une meilleure qualité de la source de données et valide la dimension *Standardisation* et *Conformité*. Cette dernière veille à la présence, dans une même source, des données homogènes, respectant le même format et le même standard.

Une fois la source est homogénéisée, nous pouvons aborder la deuxième action de nettoyage, qui est l'élimination des doublons et similaires. Nous abordons ce problème différemment grâce à la présence de la sémantique dans les données. Une information sémantique dans le choix des attributs clés pour le dédoublonnage. Une recommandation dans le choix des méthodes de calcul de distance de similarité, le seuil de correspondance ainsi que dans la fusion des enregistrements est proposée en fonction de la catégorie de données détectée. Plus de détails seront présentés dans le chapitre suivant (chapitre 7).

Chapitre 7

Elimination des doublons et similaires

Sommaire

7.1 Introduction

Dans du chapitre précédent (chapitre 6) nous avons traité et corrigé les données colonne par colonne et entre colonnes (DF). Nous allons maintenant nettoyer les données ligne par ligne. Pour cela, nous nous sommes intéressés dans cette thèse à la problématique qui préoccupe de plus en plus les travaux de recherche ainsi que les

organisations, qui est le dédoublonnage des données. Nous avons alors en entrée à ce processus, des données homogènes (standardisées, unifiées dans un même format, code et langue).

L'ordre de ce traitement (homogénéisation suivi de déduplication des données) est justifié par des expérimentations. Soit S une source qui contient α doublons et similaires. On va éliminer les doubles sur deux versions de S : une avant l'homogénéisation et une après homogénéisation. On calcule ensuite le taux d'élimination (table **??**).

TABLE 7.1 – Évaluation de l'élimination des doublons et similaires pour deux versions de S

Source	**Rappel**	**Précision**	**F-Measure**
S avant homogénéisation	0,18	1	0,30
S après homogénéisation	0,45	1	0,62

Nous avons calculé ces mesures sur la source de départ S (table 3.2) avant et après homogénéisation. Nous avons choisi comme attributs clés les colonnes une, cinq, six et sept. La mesure de similarité utilisée est Jaro-Winkler avec un seuil de 0,02.

Les résultats obtenus (table 7.1). confirme bien l'algorithme proposé dans l'introduction pour un meilleur nettoyage de données. Une homogénéisation est recommandé avant une déduplication.

Comme présentée dans l'état d'art (chapitre 2), la déduplication des données repose sur deux fonction *Match* et *Merge*.

La fonction *Match* compare les données de deux enregistrements. Cette comparaison peut être basée sur différents algorithmes de comparaison, comme ceux vus précédemment, tels que Jaro-Winkler, Jaccard et Levenshtein. Les algorithmes de comparaison nécessitent selon le contexte la définition d'un seuil d'acceptation.

La fonction *Merge* prend en paramètre deux enregistrements qui sont supposés similaires et retourne un nouvel enregistrement qui est le résultat de la fusion des deux précédents.

Dans les travaux de la littérature (chapitre 2) aucune ou peu de sémantique est prise en compte lors du processus de dédoublonnage. Nous proposons alors une approche originale qui exploite la sémantique existante dans une source pour une meilleure définition des deux fonctions *Match* et *Merge*. La catégorisation des données fournie par le profilage sémantique des données (chapitre 3.7) permettra la recommandation des attributs de dédoublonnage, les algorithmes de calcul de la

distance de similarité la plus adéquate ainsi que le meilleur seuil d'acceptation. Pour la fonction *Merge*, les règles de fusion pour les attributs clés et les attributs non clés seront recommandées en fonction de la catégorie des données utilisées.

Propriétés des fonctions *Match* et *Merge* Ces deux fonctions génériques *Match* et *Merge* constituent donc la stratégie de base pour le rapprochement des tuples dans notre démarche. Pour cela, ces deux fonctions doivent vérifier les quatre propriétés ICAR (Benjalloun et al., 2009) (*Idempotence, Commutative, Associative*, et *Représentative*) :

- Idempotence (réflexive) : Soit t un tuple dans une source de données, *Match*(t,t)=Vrai, et t=*Merge*(t,t)
- Commutative : Soit t et t' deux tuples, *Match*(t,t')=Vrai ssi *Match*(t',t)=Vrai si *Match*(t,t')=Vrai alors *Merge*(t,t')=*Merge*(t',t).
- Associative : Soit t, t' et t" trois tuples, si *Merge*(t,*Merge*(t',t")) et *Merge*(*Merge*(t,t'),t") existent alors *Merge*(t,*Merge*(t',t"))=*Merge*(*Merge*(t,t'),t").
- Représentative : Soit t, t', t" et t"' quatre tuples, si t" = *Merge*(t,t') alors pour tout t"'
  si *Match*(t,t"')=Vrai alors *Match*(t",t"')=Vrai.
  L'algorithme utilise la notion de dominance (ou représentativité) : un tuple t domine un autre tuple t' s'il est similaire et possède plus d'informations.

Commençons par présenter la définition de la fonction *Match*.

7.2 Fonction *Match*

Comme défini auparavant S est une source de données et C l'ensemble de ses colonnes (attributs), noté S(C). On considère que C=A ∪ B. *A* est l'ensemble des attributs qui servent pour éliminer les tuples similaires dans la source S (attributs clés). B est l'ensemble des attributs qui ne participent pas à l'élimination des doublons. Notons que :

- B=C-A
- A ∩ B=∅
- B peut être vide.

7.2.1 Choix des attributs clés

Généralement les attributs servant au dédoublonnage, appelés attributs clés ou attributs de dédoublonnage sont choisis en fonction d'une expertise humaine, tel est le cas dans les outils Talend.

Notre objectif est d'aider et assister l'utilisateur dans ses différents choix en lui apportant de la sémantique. Cette dernière est un avantage pour comprendre ses données, détecter les anomalies et passer enfin à la correction de ces dernières.

Avec la reconnaissance sémantique du schéma que nous proposons dans la première partie de ce manuscrit, il est possible d'avoir une information sémantique à propos des attributs clés. Le choix de ces derniers se base sur plusieurs critères, concernant une seule colonne, tels que :

— La catégorie sémantique de l'attribut et les informations stockées dans le référentiel [[SCH2]] :
  - Classe *Attribute* : le nombre de fois (*usedTimes*) qu'un attribut a été utilisé. UT définit le poids de l'attribut.
  - Classe *Recommandation* : une information *usedAsDeduplicationKey* peut exister en fonction d'une analyse antérieure. Elle définit si un attribut a été utilisé auparavant comme étant un attribut de dédoublonnage.

— Les règles déduites des indicateurs statistiques :
  - R6 (Règle d'unicité). La colonne est unique alors elle peut être une clé primaire. Par exemple, la colonne numéro de sécurité sociale, elle est unique et donc peut identifier une personne (chapitre 4, table 4.8). Si deux tuples ont le même numéro de sécurité alors il s'agit de la même personne.
  - R7 (Règle d'approximativité). La colonne contient presque des valeurs uniques à quelques valeurs prés. Un rapprochement approximatif devrait être appliqué dans le cas où R12 (taille de chaine petite) est vérifiée. Le rapprochement doit être exact vu que les méthodes de mesure de similarité ne sont pas efficaces avec des chaînes de taille limitée.

Exemple 7.1 : Attributs clés de la source S

TABLE 7.2 – Attributs clés de la source S

	A : Attributs clés		
SemanticColName	**Email**	**City**	**Country**
DType(Istat11)	String(19)	String(18)	String(10)
Constraint		DF Country	
R6			
R7			
R11	15%	5%	20%
R12			
UT	5	10	8
usedAsDKey	X	X	X

Cependant, certains attributs, utilisées seuls, sont *déconseillés* pour le dédoublonnage. C'est le cas lorsque certaines règles définies dans le chapitre 3.7, table 3.10 sont vérifiées :

— Les attributs dont la catégorie n'a pas pu être détectée.

— Les attributs Y de la dépendance X → Y ne peuvent pas être des attributs clés.

— Règles déduites des indicateurs statistiques :

- R9 et R10 (Doublons) sont vérifiées. La colonne est définie sur un ensemble fini de valeurs. Par exemple, *Gender* {F, M}, *Civility* {Mme, M.} ou *Qualification* {Employee, Project Manager, CEO}).
- Si la règle R11 est vérifiée (la colonne contient un pourcentage élevé des valeurs nulles).

Exemple 7.2 : Attributs non clés de la source S

TABLE 7.3 – Attributs non clés de la source S

	B : Attributs non clés	
SemanticColName	**Civility**	**Gender**
DType(Istat11)	String(3)	String(1)
Constraint	DF Gender	
R9/R10	{Mme, M.}	{F, M}
R11	25%	5%
UT	1	2
usedAsDKey		

Les règles sont calculées en appliquant une analyse statistique sur la source.

Remarque : Des colonnes non recommandées pour une utilisation indépendante, pourront être associées avec d'autres colonnes afin de construire un attribut clé.

7.2.2 Distances de similarité

La fonction *Match* compare deux tuples t et t' d'une source de données. *Match*(t,t')=Vrai ssi t≈t'. Cette similarité (≈) est calculée en fonction de différentes méthodes de calcul de distance de similarité.

Avec le profilage sémantique, nous proposons des mesures de similarité en fonction de la catégorie et la sous-catégorie (langue) d'une colonne si elle existe. Nous recommandons aux utilisateurs les mesures et seuils adéquats. Cette tâche présente une aide précieuse à l'utilisateur.

Nous présentons dans le tableau suivant (table 7.4), les distances de similarité adéquates pour certaines catégories de type chaîne de caractère (les numériques et dates sont transformés en chaînes aussi). Nous traitons les deux types d'algorithme de similarité : textuel et phonétique.

Un seuil est donné en fonction de la mesure adéquate pour chaque catégorie. Plus de détails seront présentés dans le chapitre expérimentation (chapitre 8).

TABLE 7.4 – Algorithmes de similarité et seuils recommandés pour les différentes catégories

Catégories	Type Algorithme	Algorithmes recommandés	Seuil recommandé
FirstName	Phonétique	SdJW	0,002
	Textuel	Jaro-Winkler	0,05
Country	Phonétique	SdJW	0,001
	Textuel	Jaro-Winkler	0,04
City	Phonétique	SdJW	0,005
	Textuel	Jaro-Winkler	0,03
Address	Phonétique	Q-Gram	0,21
	Textuel	Jaro-Winkler	0,11
Email	Phonétique	NysiisJW	0,21
	Textuel	Jaro-Winkler	0,11
Phone	Phonétique	DM	0
	Textuel	Jaro-Winkler	0,034
Date	Phonétique	NysiisLV	0
	Textuel	Jaro-Winkler	0,046

avec DM (Double Metaphone) ; SdLv (Soundex combiné avec Levenshtein) ; SdJW (Soundex combiné avec Jaro-Winkler).

Divers algorithmes de calcul de distance de similarité sont utilisés pour les chaînes de caractères tels que Jaro-Winkler, Jaccard, Levenshtein et Qgram pour les algo-

rithmes textuels et Double Metaphone, Soundex et NYSIIS en faisant appel à Jaro-Winkler et Levenshtein pour les algorithmes phonétiques. Les seuils pour les chaînes de caractères sont homogénéisés et compris dans l'intervalle [0,1].

Exemple 7.2 : Recommandation des mesures et seuils pour les attributs clés

TABLE 7.5 – Recommandation des mesures et seuils pour les attributs de l'ensemble A de la source S

	A : Attributs clés de la source S		
ColName	**Email**	**City**	**Country**
DataType	String	String	String
Measure	Jaro-Winkler	Jaro-Winkler	Jaro-Winkler
Threshold	0,11	0,03	0,04

L'étude de similarité porte sur les attributs clés de l'ensemble A selon les règles de similarité.

7.2.3 Règles de décision

Afin de décider de la similarité entre valeurs et ensuite entre tuples, nous avons défini trois similarités pour la fonction *Match* : similarité entre valeurs, règles de similarité et similarité entre tuples.

Définition 7.1 : Similarité entre valeurs (notée ≈)

Deux valeurs v et v' sont similaires, on note $(v \approx v')$, ssi la distance de similarité d, calculée entre ces deux valeurs, vérifie une condition k.

Pour deux tuples t et t', pour un attribut A_i avec (i=1 ;n avec n est le nombre d'attributs), t.$A_i \approx$ t'.A_i ssi la condition k_j est vérifiée (j=1 ;f avec f est le nombre de conditions). La condition k_j se base sur le calcul de la distance d_i de similarité selon le type des données. Plusieurs cas de figures peuvent être envisagés. L'utilisateur peut ainsi fixer (donner) un ou plusieurs seuils.

— k_j : d_i est inférieur à un seuil maximal ; $(d_i < \varepsilon_i)$ avec $\varepsilon_i \in [0,1]$ et $d_i \in [0,1]$.
— k_j : d_i appartient à un ensemble de seuils ; $(d_i \in \{\varepsilon_1, \ldots, \varepsilon_p\})$;
— k_j : d_i appartient à un intervalle de seuils ; $(d_i \in [\varepsilon_1..\varepsilon_2])$

La similarité ($\approx$) vérifie évidemment les propriétés de réflexivité, commutativité et d'associativité. C'est-à-dire $[v \approx v]$, $[(v \approx v') \Leftrightarrow (v' \approx v)]$ et $[v \approx (v' \approx v'') \Leftrightarrow (v \approx v') \approx v'']$. ■

Exemple 7.1 : Similarité entre valeurs

Soit :
Address1 ← t.Address ; Address2 ← t'.Address ;
Email1← t. Email ; Email2 ← t'. Email ;
Name1 ← t. Name ; Name2 ← t'. Name ;
Phone1← t. Phone ; Phone2 ← t'. Phone

Deux adresses sont similaires (Address1 ≈ Address2) si la distance d de similarité, selon une mesure choisie, est inférieure au seuil $\varepsilon = 0,2$ par exemple. Les deux adresses suivantes Address1 = "133 BOULEVARD MARCEL EPINAY/SEINE " et Address2= "133 BOULEVARD MARCEL 93800 EPINAY-SUR-SEINE " sont similaires, selon la méthode de Jaro-Winkler car la distance calculée est de 0,057. ■

Définition 7.2 : Règle de similarité

Une règle r de similarité est une conjonction de similarités qui portent sur des attributs A_i avec (i=1,n) de l'ensemble A de la source S (formule 7.1). x=1 ;g avec g est le nombre de règles de similarité.

$$r_x = \bigwedge_{a=1}^{n} d_a \leq \varepsilon_a \tag{7.1}$$

avec d_a est la distance de similarité entre deux attributs clés et ε_a un seuil pour chaque pair d'attributs.

r = (t.A_1≈t'.A_1) ∧ (t.A_2 ≈ t'.A_2) ∧ (t.A_i≈t'.A_i)...∧(t.A_n≈t'.A_n). ■

Exemple 7.2 : Règles de similarité

Voici un exemple de deux règles de similarités r_1 et r_2 :
r_1 = (Name1≈Name2) ∧ (Email1≈Email2) ∧ (Address1≈Address2),
r_2 = (Email1≈Email2) ∧ (Phone1≈Phone2). ■

Définition 7.3 : Similarité entre tuples

Deux tuples t et t' sont similaires ssi la disjonction des règles de similarité définies sur la source S est vraie. On note t≈t' ssi $r_1 \vee r_2 \vee \ldots \vee r_k \ldots \vee r_q$ est vraie avec q étant le nombre de règles de similarité (formule 7.2).

$$Rules = \bigvee_{x=1}^{q} r_x \quad ■ \tag{7.2}$$

Exemple 7.3 : Similarité entre tuples

t et t' sont similaires ssi $r_1 \vee r_2$:
t≈t' ssi ((Name1≈Name2) ∧ (Email1≈Email2) ∧ (Address1≈Address2))
∨ ((Email1≈Email2) ∧ (Phone1≈Phone2)). ■

Remarque : Ces règles peuvent présenter toutefois des incohérences ou des contradictions qui ne seront pas vérifiées dans cette thèse.

Le choix des seuils dans les travaux précédents nécessitent des connaissances métiers (Hernandez and Stolfo, 1998). Notre apport est de recommander les meilleurs seuils en fonction de la catégorie sémantique des données.

Les algorithmes de déduplication proposés (MFB) comparent plusieurs tuples en même temps et non pas deux à deux comme proposé dans les algorithmes de la littérature (Benjalloun et al., 2009).

7.2.4 Algorithme de la fonction *Match*

L'algorithme de la fonction *Match* (algo 28) dépend de la catégorie sémantique des données. On recommande les méthodes de calcul de distance de similarité ainsi que les seuils adéquats en fonction de la sémantique de la donnée.

Le principe de l'algorithme de *Match* se résume en deux étapes :

— Pour chaque attribut clés A_i, on calcule la similarité entre deux valeurs des deux tuples t et t'. Cette distance devrait respecter un certain seuil (définition 1).

— Des règles de similarité *Rules* sont définies afin de décider si t et t' sont similaires (définition 3).

Algorithm 28 Fonction *Match*

Algorithm *Match*
Input :
Two tuples, t and $t' \in$ S (data source)
AlgoDistance (Similarity Measure), ε (Threshold)
SCHS (semantic structure) : DataType (DataType of data), Lang (language of data)
Output : Result ← True if ($t \approx t'$)
Begin
Result ← False
for (all $Rules_j$ j from 1 to q) **do**
$Rules_j$ ← True ; i← 1
while ($Rules_j$ and i ≤ n) **do**
$v \leftarrow t.A_i$; $v' \leftarrow t'.A_i$
if (v or v' = NULL) **then**
Result ← False
else
switch DataType(A_i)
CASE Date : //To compare two dates, calculates the diffrence between days, months and years or transforms them into strings and calculate the similarity.
Result ← matchValues(v, v')
CASE Numeric : //For the numeric, same thing as date (difference or similarity between strings)
Result ←handleNumericValues(v, v')
CASE String categories language dependent : //like City, Country or Continent
Result ← handleStringValues(v, v')
Default : // By default, transform values into strings and calculate the similarity.
Result ←matchValues(v, v')
end switch
end if
$Rules_j$ ← $Rules_j$ and Result ; i++
end while
Result ← Result or $Rules_j$
end for
END Algorithm *Match*

La fonction **handleNumericValues** gère le rapprochement des valeurs numériques.

Algorithm 29 Fonction *handleNumericValues*

Algorithm *handleNumericValues*
Input :
    Two values, v and v'
    AlgoDistance (Similarity Measure), ε (Threshold)
Output : Result (boolean)
Begin
if (diff(v, v')=0) **then**
  Result ← True
else
  Result ←matchValues(v, v')
end if
CASE String categories language independent : //like FirstName or Email
if (AlgoDistance(v, v') $\leq \varepsilon$) **then**
  Result ← True
end if
return Result
END Algorithm *handleNumericValues*

La fonction **handleStringValues** gère la comparaison des valeurs de type chaînes de caractères.

Algorithm 30 Fonction *handleStringValues*

Algorithm *handleStringValues*
Input :
    Two values, v and v'
    AlgoDistance (Similarity Measure), ε (Threshold)
Output : Result (boolean)
Begin
if (Lang(v) = Lang(v')) **then**
  if (AlgoDistance(v,v') $\leq \varepsilon$) **then**
    Result ← True
  end if
else
  if (AlgoDistance(v, translate(v', $Lang_1$)) $\leq \varepsilon$) **then**
    Result ← True
  end if
end if
return Result
END Algorithm *handleStringValues*

La fonction **matchValues** permet de calculer la similarité entre deux valeurs d'autre type que chaînes de caractères.

Algorithm 31 Fonction *matchValues*

Algorithm *matchValues*
Input :
    Two values, v and v'
    AlgoDistance (Similarity Measure), ε (Threshold)
Output : match (boolean)
Begin
match ← False
ch_1 ← String.parseString(v)
ch_2 ← String.parseString(v')
if (AlgoDistance(ch_1, ch_2) $\leq \varepsilon$) **then**
  return match ← True
end if
END Algorithm *matchValues*

7.3 Fonction *Merge*

La fonction *Merge* traite deux tuples supposés similaires, tels que $Match(t, t') = Vrai$. Elle retourne un nouveau tuple qui est le résultat de la fusion des deux précédents : $t'' = Merge(t, t')$.

7.3.1 Attributs de fusion

La fonction *Merge* retourne un tuple qui apporte "plus" d'information. Un enrichissement des données peut être réalisé à la demande de l'utilisateur sur tous les attributs de la source (les attributs A et B). Cet enrichissement dépend de la catégorie des données. Une aide sémantique est aussi proposée afin de recommander la meilleure règle de fusion.

Dans la fonction Match seuls les attributs clés A sont utilisés cependant avec la fonction Merge tous les attributs sont exploités. Les attributs non clés B sont aussi importants lors de la fusion et nécessitent des règles de fusion.

7.3.2 Règles de fusion

D'une manière plus générale, les règles de fusion se résument en ces différentes actions :

— Concaténer (concat) : on fusionne le contenu d'attribut du premier enregistrement avec celui du deuxième enregistrement - par exemple, "0621314151" et "0142526782" seront fusionnés en "0621314151, 0142526782" . On pourra

spécifier un séparateur à utiliser entre les valeurs (par exemple ","). Ou bien en fonction de la catégorie sémantique, on aura une information sur cet attribut s'il peut supporter plus d'une valeur tels que *Phone* ou *Email*.

— Le plus fréquent (mostFrequent) : on choisit la valeur la plus fréquente dans les tuples similaires.

— Les plus récents ou les plus anciens (pour les dates) : on choisit la date la plus ancienne ou la date la plus récente dans les tuples similaires. L'homogénéisation des formats et la vérification de la validité des dates est nécessaire (tâche réalisée précédemment dans la première partie de nettoyage) pour un meilleur choix.

— La plus longue (longest) ou la plus courte (shortest) : on choisit la valeur la plus longue ou la valeur la plus courte dans les tuples similaires.

— Plus grand ou plus petit ou la moyenne (min max, avg) : on choisit la plus grande valeur numérique ou la plus petite ou la moyenne dans les tuples similaires. Des transformations peuvent être réalisées telles que des conversions dues à des changements d'unité. Les valeurs nulles sont considérées égales à 0.

— La fusion d'une chaîne quelconque avec une valeur nulle (NULL) renvoie la chaîne elle-même.

Exemple 7.3 : Règles de fusion pour les attributs de la source S

TABLE 7.6 – Règles de fusion pour les attributs de la source S

ColName	**Email**	**City**	**Country**	**Civility**	**Gender**
DataType	String	String	String	String	String
Constraint		DF Country		DF Gender	
MergeRule	Concat	mostFrequent	mostFrequent	mostFrequent	mostFrequent

7.3.3 Algorithme de la fonction *Merge*

Le principe général de l'algorithme de la fusion de deux tuples est donné ci-dessous (algo 32). La fonction fusion permet de combiner deux valeurs selon le type et la catégorie de données de l'attribut en question :

Nous résumons les différentes règles présentées ci-dessus dans l'algorithme suivant (algo 32).

Algorithm 32 Fonction *Merge*

Algorithm *Merge*
// Merges a set of tuples according to some rules such as regular expressions or functions defined by the user.
Input :
Set of tuples, $t_1...t_m \in$ S (m similar tuples)
μ (Merge rule)
SCHS (Semantic Structure) : DataType (DataType of data)
Output : A tuple t
Begin
$t_1 \approx t_2 \approx ... \approx t_m$
$v \leftarrow null$
for (all Attribute C_i in C $\in$ S from 1 to n) **do**
$v_1 \leftarrow t_1.C_i$; $v_2 \leftarrow t_2.C_i$; ... ; $v_m \leftarrow t_m.C_i$
// we enrich the nulls values
if ($v_1 = NULL$ or $v_2 = NULL$ or $...v_m = NULL$) **then**
$v \leftarrow \mu(v! = NULL)$
else if ($v_1 = v_2 = ... = v_m$) **then**
$v \leftarrow v_1$
else
switch DataType(C_i)
CASE Date : //μ can be the recent date.
$v \leftarrow \mu(v_1, v_2, ..., v_m)$
CASE Numeric : // μ can be the min, max or avg of the values
$v \leftarrow \mu(v_1, v_2, ..., v_m)$
CASE String : //Concatenate more than one value for some categories like Email or Phone
$v \leftarrow \mu(\text{getDifferentValues}(v_1, v_2, ..., v_m))$ //Concatenate a set of values
Default :
$v \leftarrow \mu(v_1, v_2, ..., v_m)$//$\mu$ can be the mostFrequent
end switch
$t.C_i \leftarrow v$
end if
end for
END Algorithm *Merge*

Présentons à présent les différents algorithmes utilisés et développés afin de résoudre la déduplication des données.

7.4 Les algorithmes de déduplication

Plusieurs travaux dans la littérature ont abordé la problématique de déduplication. Nous avons retenu les algorithmes SERF (Benjalloun et al., 2009). Les fonctions

Match et Merge sont réalisées simultanément sur une paire de tuples. La notion de dominance entre tuple assure une meilleure détection des doublons et similaires.

Cependant, en étudiant de plus près ces algorithmes, nous avons pu constater la limite du traitement des tuples deux à deux. Afin d'y remédier, nous proposons trois nouveaux algorithmes MFB(1-3).

7.4.1 Algorithmes SERF

Benjalloun et all [BMG+2009] ont développé trois algorithmes séquentiels (G, R et F) qui traitent le problème d'élimination des doublons et similaires. Les fonctions *Match* et *Merge* sont appelées sous forme d'une boite noire. Le principe de fonctionnement de l'algorithme R est comme suit :

— Chaque tuple de Tdep (source de départ) est comparé avec tous les tuples (n*n) existants dans Tarr (source arrivée).
— Deux tuples similaires sont fusionnés et le nouveau tuple (résultat de fusion) est ajouté à Tdep.
— Le processus est ré appliqué jusqu'à que Tdep soit vide.

La complexité de cet algorithme est de l'ordre de O(n^2) vu qu'on compare les n tuples de départ avec les n tuples en sortie.

Plus de détails dans l'algorithme ci-après (algo 33).

Algorithm 33 L'algorithme R

```
Algorithm R
    Input : Tdep a set of tuples from a data source (may contain duplicates)
    Output : Tarr a set of tuples (without duplicates)
Begin
Tarr ← ∅
while (Tdep ≠ ∅ ) do
  t ← a tuple from Tdep
  remove t from Tdep
  buddy ← null
  if Tarr ≠ ∅ then
    for (all tuples t' in Tarr) do
      if Match(currentTuple , t') then
        buddy ← t'
        EXITFOR
      end if
    end for
  end if
  if (buddy = null) then
    add currentTuple to Tarr
  else
    t" ← Merge(currentTuple, buddy)
    remove buddy from Tarr
    add t" to Tdep
  end if
end while
return Tarr
END Algorithm R
```

La gestion de la mémoire présente une limite pour les algorithmes G, R et F lors de traitement de gros volumes de données. Nous proposons alors de nouveaux algorithmes (Boufarès et al., 2012b), (Boufarès et al., 2012a), (Boufarès et al., 2013).

7.4.2 Algorithme MFB1

MFB1 propose une solution pour une meilleure gestion de la mémoire en utilisant des structures intermédiaires. Le traitement de Big-data (Ben-Salem, 2013) est alors possible.

Le principe de fonctionnement de MFB est modélisé par la figure suivante (figure7.1).

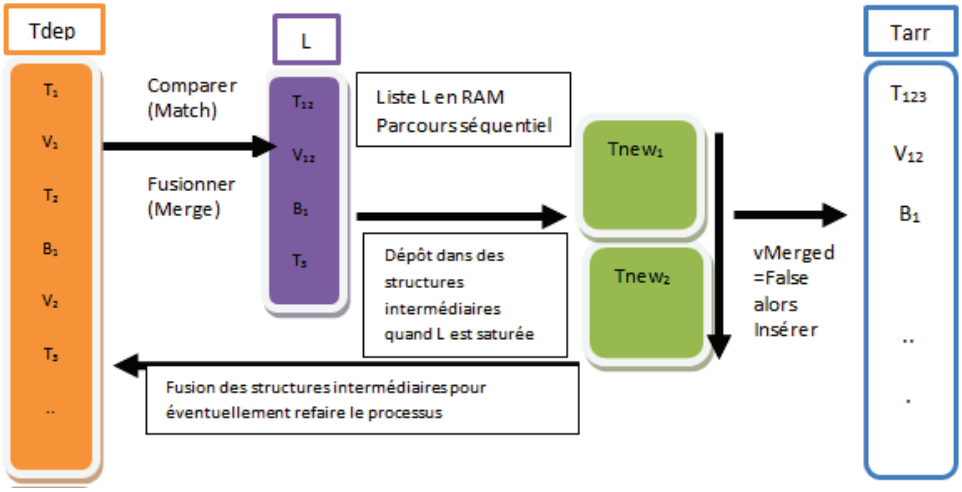


FIGURE 7.1 – Principe de l'algorithme MFB1

Il se résume en cinq points :

— Chaque tuple de Tdep (source de départ) est comparé avec les tuples existants dans la mémoire (liste L).

— Les tuples similaires (tuple de Tdep et tuple de L) sont fusionnés directement dans L.

— En cas de saturation de mémoire, L est vidé dans des structures intermédiaires.

— Le processus est relancé h fois (h étant le nombre d'itérations) avec une nouvelle source en entrée avec moins de doublons.

La complexité de MFB1 est aussi de l'ordre de $O(n^2)$ étant donné que nous comparons tous les enregistrements entre eux. Le nombre de comparaisons est ainsi réduit si les similaires se suivent dans la source en entrée (moins de comparaisons avec les tuples de la liste L).

Algorithm 34 L'algorithme MFB1

```
Algorithm MFB1
Input :
    Tdep a set of tuples from a data source (may contain duplicates)
    h number of iteration given by the user
Output : Tarr a set of tuples (without duplicates)
Begin
vMerged ← False // A variable that indicates whether there was a merge of two tuples at least.
// Tnew(i) corresponds to a block of Tdep. Each block of Tdep may contain duplicates or not (i=1 ; n blocks).
a : tuple
a ← retrieve a tuple from Tdep
L.add (a) // add a to the temporary list of tuples L
lengthL ← 1
while (h ≠ 0) do
  for (each tuple t from Tdep) do
    m ← Match(a, t)
    if (m== True) then
      a ← Merge(a, t)
      i, vMerged ← addToTheListL(a, L, Tnew(i))
    else
      i, vMerged ← addToTheListL(t, L, Tnew(i))
    end if
  end for
  if ( (vMerged = True and i>=1)) then
    Tdep ← createNewDTable(Tnew(i), i, Tdep)
    // Treatment is the same on the new table Tdep Result of mixture Blocks
    if ( (L.notEmpty())) then
      add(L, Tdep)
    end if
    //L n'est pas saturée. Ajouter son contenu dans la source d'arrivée.
  else
    Tarr← L
  end if
  Tarr ← Tdep
  h ← h -1
end while
return Tarr
END Algorithm MFB1
```

La fonction **addToTheListL()** permet d'ajouter un tuple dans L. Elle gère aussi la création des intermédiaires $Tnew_i$ au fur et à mesure que L est saturée.

Algorithm 35 Fonction « addToTheListL »

```
Function addToTheListL
//Allows to add vMerged tupled to a list L or to tables Tnew1 until Tnewn.
Input : tuple t, List L, Tables Tnew(i)
Output : vMerged, i
Begin
for (all int j from 0 to lengthL Do) do
  if ((Match(t, L[j])) then
    L[j] ← Merge(t, L[j])
    vMerged ← True
  end if
  if lengthL ≤ lmax then
    L.add (t)
    lengthL++
  else
    Tnew(i) ← L // if size of L exceeds a fixed length so add L to Tnew(i) and
    reset L.
    i++
    L ← ∅
    L.add (t)
  end if
end for
return vMerged
END function addToTheListL
```

La méthode **createNewSourceS()** permet de reconstruire une nouvelle source T_{New} (moins les fusions déjà réalisées) en mélangeant les tables intermédiaires (formule7.3) autrement afin de croiser les tuples qui n'ont pas encore été rapprochés (algo 36). Le principe est :

— prendre un bloc de taille p de chaque table intermédiaire T_{newi}

— les insérer dans la nouvelle table de départ T_{New}.

— on reproduit le processus jusqu'à vider les T_{newi}.

$$T_{New} = \cup_{i=1}^{m} T_{newi} \tag{7.3}$$

Algorithm 36 Function "createNewSourceS"

```
Function createNewSourceS
//Create a new depart table which contains the tuples of all tables Tnew(i) in
some order.
Input : Tables Tnew(i), int i
Output : Table Tdep
Begin
a : tuple of S
p : number of tuples of Tnew(i)
for (all int j from 0 to p) do
  for (all int k from 0 to i) do
    a ← Tnew(k)[j]
    add(a, S)
  end for
end for
END Function createNewSourceS
```

L'algorithme MFB1 est utilisé dans l'outil Master Data Management (MDM) de Talend afin de contrôler la construction du MDM. Ils utilisent le principe de partitionnement selon une clé définie par l'utilisateur comme alternative au traitement par bloc que nous proposons en fonction de la taille de mémoire (Boufarès et al., 2013).

Les mesures de performance sont nettement améliorées par rapport aux algorithmes SERF. Le graphique ci-après (figure 7.2) illustre cette évolution.

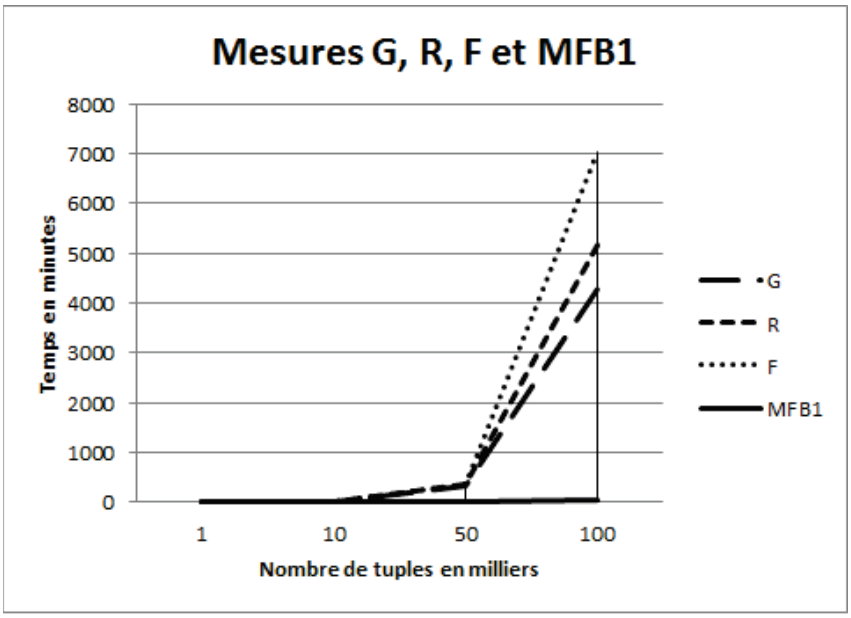


FIGURE 7.2 – Mesure des performances des algorithmes G, R, F et MFB1 avec 10% de doublons (en min)

7.4.3 Algorithme MFB2

Une nouvelle version pour MFB1, optimisant la recherche des doublons. Une recherche arborescente remplace le parcours séquentiel des enregistrements dans la liste L(figure 7.3). Une amélioration dans le temps d'exécution est constatée.

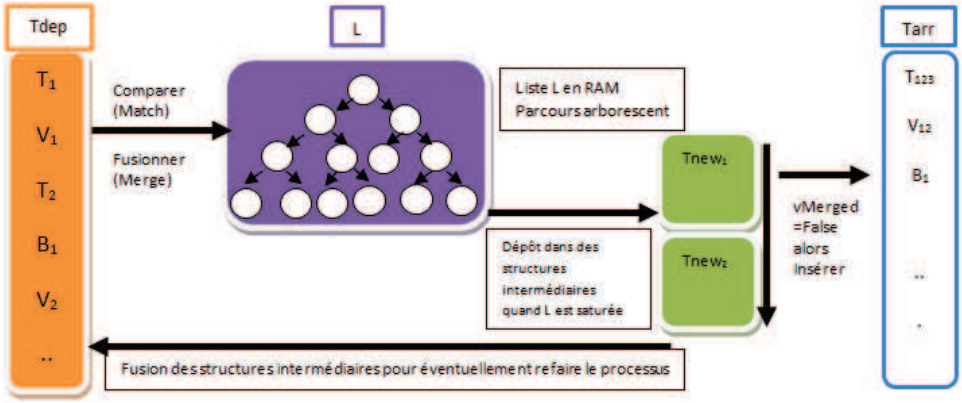


FIGURE 7.3 – Principe de l'algorithme MFB2

La structure **TreeMap** [1] est utilisé pour gérer la création d'un arbre ordonné et navigable. Elle est composée d'un ensemble de paires (Noeuds) = <Clé, Valeur>. Plusieurs couples pourront être définis. Dans notre cas, nous avons choisi comme :

1. http ://fr.wikipedia.org/wiki/Treemap

— Clé : la taille du tuple
— Valeur : la liste des tuples qui ont la même taille

Les tuples sont triés en fonction de la clé. Le principe de la recherche arborescente appliqué à MFB1 est résumé dans les quatre points suivants :

— On prend un tuple t de la table de départ Tdep et on calcule sa taille.
— On vérifie dans l'arbre L s'il y a un noeud qui a la même taille (valeur).
— Si L n'est pas saturée alors on ajoute t à L tout en vérifiant s'il existe un tuple similaire. S'il existe un t' $\approx$ t alors on les fusionne et on ajoute t" (résultat de fusion) à L.
— S'il n'y a pas de noeud qui a la même taille, on crée un nouveau noeud et on insère t.

La recherche des tuples similaires dans L passe d'une recherche séquentielle avec MFB1 à une recherche arborescente dans MFB2.

La complexité de MFB2 est de l'ordre de O($nLogn$) (tri plus le parcours arborescent).

Ce traitement accélère le temps de parcours des enregistrements (figure 7.4) (Hammou et al., 2013), cependant, il reste assez long pour des Big-Data.

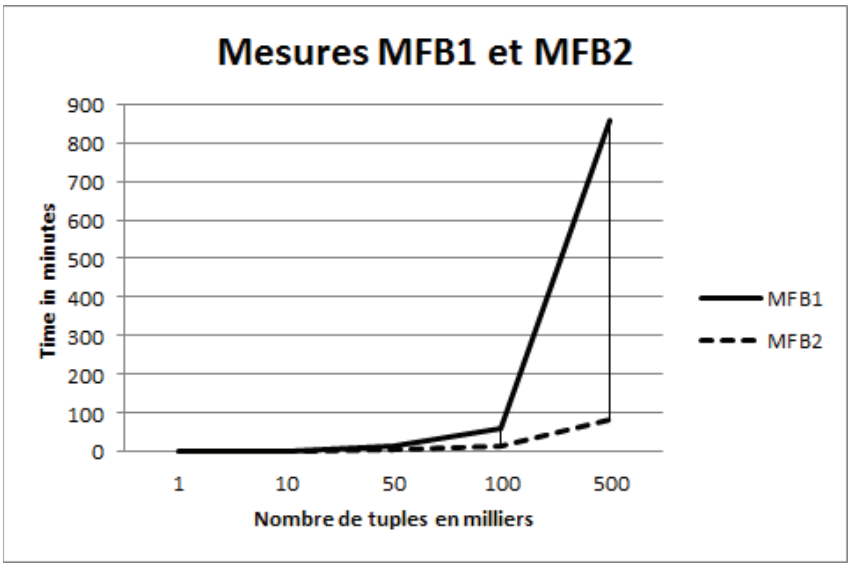


FIGURE 7.4 – Mesure des performances des algorithmes MFB1 et MFB2 avec 10% de doublons (en min)

7.4.4 Algorithme MFB3

Nous avons essayé d'étudier les causes et les parties dans l'algorithme qui pénalisent le temps d'exécution. Nous avons conclu que la recherche des similaires est très longue vu qu'un enregistrement peut être similaire à un enregistrement qui se

trouve à la queue de la liste. Ce parcours prend énormément de temps et pénalise le temps d'exécution de l'algorithme.

Nous avons pensé alors à une approche qui essaye de ramener les tuples similaires un au-dessous de l'autre. L'avantage avec cette nouvelle approche est que nous réalisons la comparaison et la fusion d'un ensemble de tuples et plus deux à deux. Cette nouvelle perspective va améliorer nettement le temps de réponse.

La solution alors était de trier la source de départ en fonction des **attributs clés**, que nous avons choisi pour le dédoublonnage. Le principe de MFB3 est représenté par la figure 7.5.

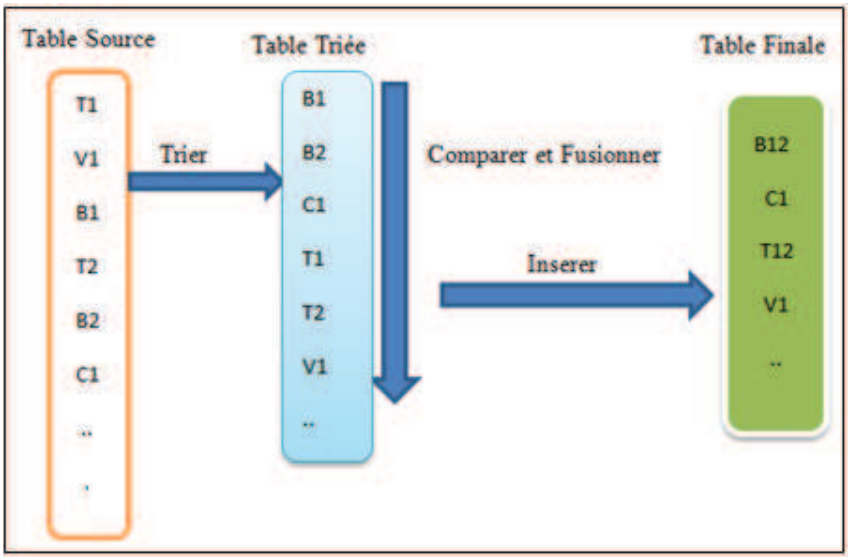


FIGURE 7.5 – Principe de l'algorithme MFB3 (version1)

Le principe de MFB3-1 consiste à :

— La source de départ est triée (QuickSort) et contenue en mémoire.
— On compare les enregistrements entre eux et on fusionne les similaires au fur et à mesure.
— Un enregistrement différent est rencontré, on ajoute alors le précédent à la table de sortie.
— On refait le même processus avec le reste des enregistrements.

Algorithm 37 Algorithme MFB3

```
Algorithm MFB3
Input :
      Tdep
      KeyAttributes
Output : Tarr
Begin
n ← number of tuples in Tdep
QuickSort(Tdep, KeyAttributes) ;
i ← 1
repeat
   tupleRes ← Tdepi
   v ← true
   while ( (v and i≤n)) do
      i ← i+1
      v ← Duplicate(tuplesRes, Tdepi)
   end while
   insert (tupleRes, Tarr)
until i>n
END Algorithm MFB3
```

L'algorithme MFB3 est une version plus performante des deux précédents algorithmes MFB1 et MFB2 grâce au tri rapide de la table source avant de démarrer le dé-doublonnage. Pour le calcul de la nouvelle complexité : le tri est de l'ordre O($nLogn$) et la comparaison est de l'ordre O(n) . Or $nLogn + n \approx nLogn$ donc la complexité de MFB3 revient à O($nLogn$).

Le tri rapide (QuickSort [2]) est un algorithme de tri inventé par Tony Hoare en 1960 et fondé sur la méthode de conception diviser pour régner. Quicksort divise d'abord un large tableau en deux sous-tableaux : les plus petits et les plus grands. Quicksort trie ensuite de manière récursive les deux sous-tableaux.

Deux versions de cet algorithme existent selon la taille de la source. La figure ci-dessous (figure 7.6) permet d'expliciter la différence entre les deux version de l'algorithme MFB3.

2. http ://www.personal.kent.edu/ rmuhamma/Algorithms/MyAlgorithms/Sorting/quickSort.htm

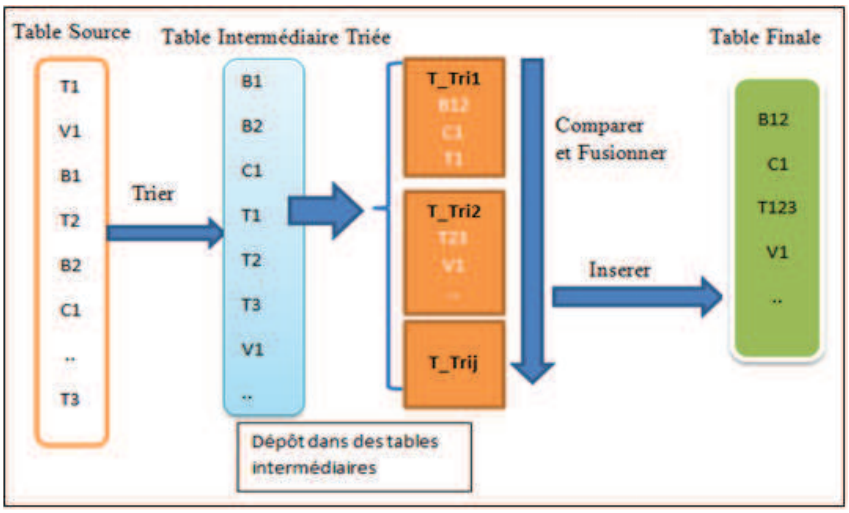


FIGURE 7.6 – Principe de l'algorithme MFB3 (version2)

Le principe de fonctionnement de MFB3-2 (algo 37) est résumé selon quatre étapes :

— Le même processus que MFB3-1 est appliqué sauf que la mémoire se sature pour un certain nombre d'enregistrements. On les trie alors par bloc.

— Chaque bloc de tuples est trié, comparé, fusionné et sauvegardé dans une structure intermédiaire.

— On garde de chaque bloc le dernier tuple afin qu'il soit comparé avec les enregistrements du bloc suivant.

— A la fin du processus, la source de sortie est créée en triant et fusionnant les différentes structures intermédiaires.

Principe de tri et fusion des tables intermédiaires $Tnew_i$ (figure 7.7) :

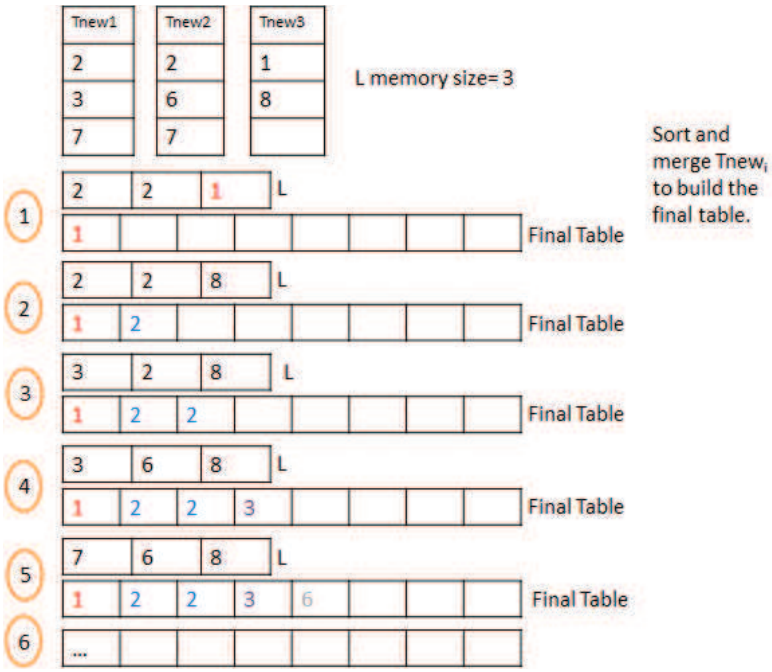


FIGURE 7.7 – Processus de tri et fusion utilisé pour la création de Tarr à partir des $Tnew_i$

Remarque : L'ordre des attributs clés lors du tri rapide est important. Afin de garantir un meilleur tri, nous organisons les attributs selon l'indicateur "nombre des valeurs nulles" existantes dans chaque attribut clé. L'attribut qui contient le moins des valeurs nulles est prioritaire dans l'ordre des attributs.

MFB3 est nettement meilleur que les autres algorithmes définis précédemment en temps d'exécution (figure 7.8).

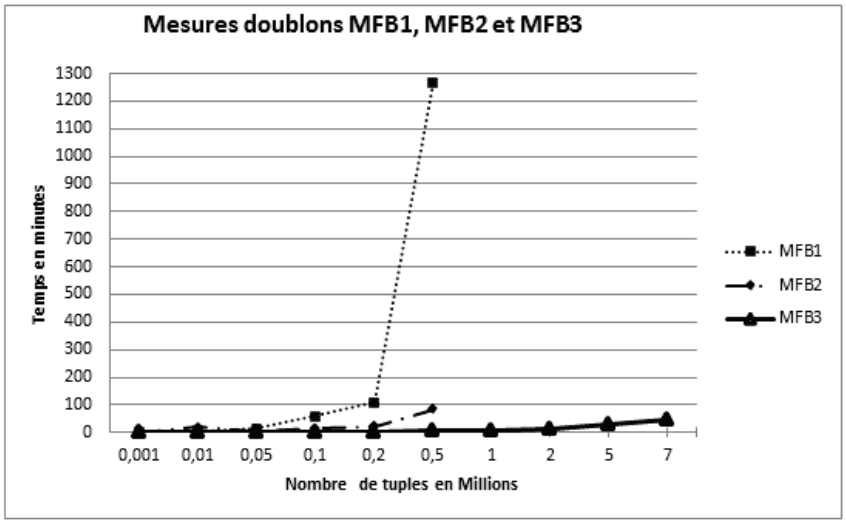


FIGURE 7.8 – Mesure des performances des MFB algorithmes avec 10% de doublons (en min)

Taux d'élimination (Mesures d'évaluation) Afin de calculer et comparer le taux d'élimination des trois versions de l'algorithme MFB, nous avons calculé les mesures d'évaluation : table , Précision et F-mesure (présentées dans l'état de l'art (chapitre2, table 2.14).

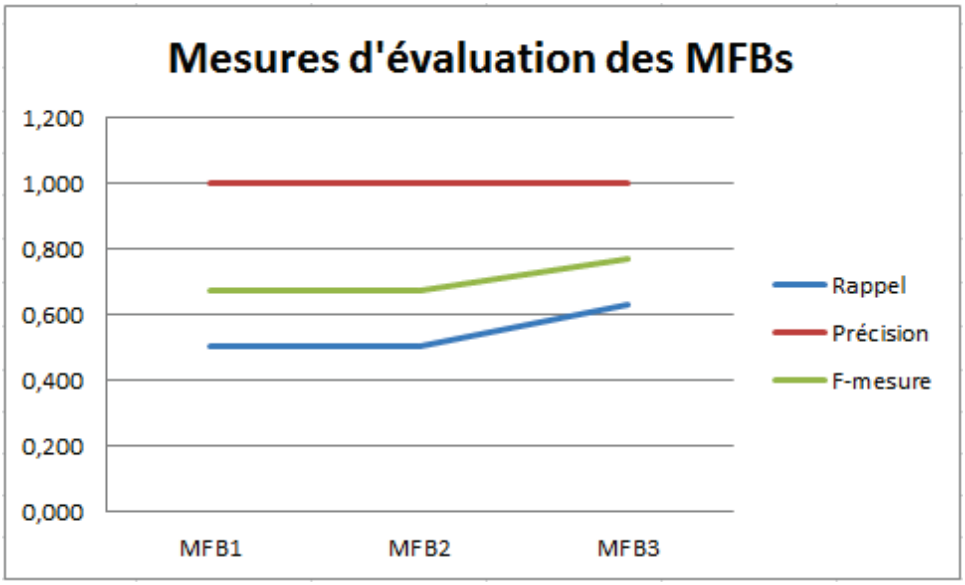


FIGURE 7.9 – Comparaison de l'efficacité des algorithmes MFBs

7.4.5 Algorithme MFB4 (Parallélisation des MFB)

MFB3 est testé avec la technologie Map-Reduce (Rehab-Adjout and Boufarès, 2014), (Zayen et al., 2014) afin d'attaquer des gros volumes de données (BigData). Le nouveau algorithme est appelé MFB4. Comme un premier travail, cette technologie a été testé seulement avec des doublons exacts.

La figure suivante (figure 7.10) détaille le principe de l'algorithme MFB4.

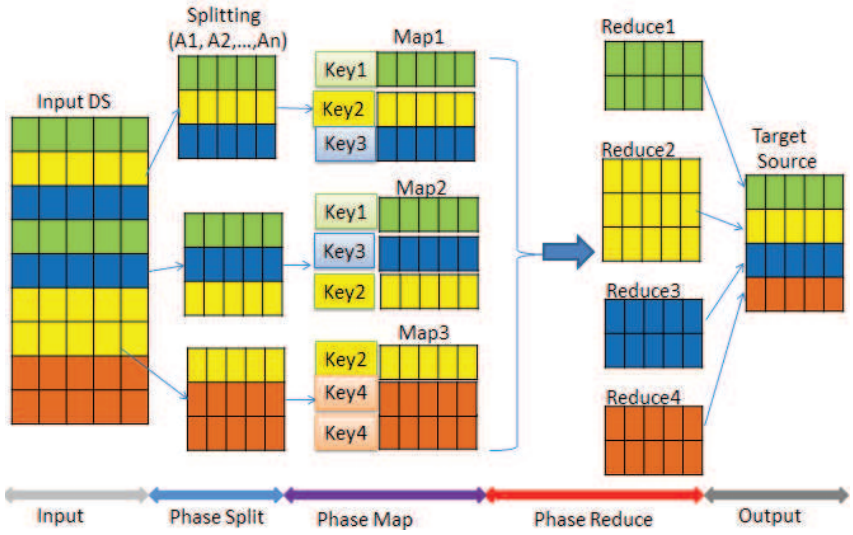


FIGURE 7.10 – Principe de l'algorithme MFB4 (technique MapReduce)

La fonction Map aura comme clé, la liste des attributs clés pour le dédoublonnage. La fonction Reduce va faire le regroupement des résultats de la fonction Map ayant la même clé et garde un seul parmi les doublons.

Algorithm 38 Main Function of the MapReduce Algorithm

Main Function
Input : S Data source
Output : List Key_Attributes
Begin
List Key_Attributes $\leftarrow \{S.A_1, S.A_2, ..., S.A_n\}$//List of key attributes
END Main Function

Algorithm 39 Mapper Function of the MapReduce Algorithm

Mapper Function
Input :
    List Key_Attributes ,
    $Tuple_i$ // i=1... nbtuples
Output :
    Text Key_i
    $Tuple_i$ // i=1...nbtuples
Begin
(Text Key_i, $Tuple_i$) = Map($Tuple_i$)
for each A_i in Key_Attributes **do**
  if Key_i = null **then**
    $Key_i \leftarrow A_i.v$
  else
    $Key_i \leftarrow Key_i$ + ";" + $A_i.v$
  end if
end for
Emit(Key_i, $Tuple_i$)
END Mapper Function

La fonction Reducer(), regroupe les enregistrements qui ont la même clé. Ces tuples sont des doublons donc le Recucer() renvoi un parmi eux (*getTupleFrom()*).

Algorithm 40 Reducer Function of the MapReduce Algorithm

Reducer Function
Input :
    Text Key_i
    List of tuples [$tuple_1$,..,$tuple_i$]// i=1... nbtuples
Output : tuple
Begin
tuple $\leftarrow$ getTupleFrom([$tuple_1$,..,$tuple_i$])
Emit(Key_i, tuple) //takes just one tuple for each key.
END Reducer Function

7.5 Conclusion

Nous avons durant ce chapitre exposé différentes versions des algorithmes de déduplication. L'originalité de ces algorithmes réside dans le fait de procéder à l'élimination des données redondantes au sens strict et au sens large. Les fonctions *Match* et *Merge* sont appliquées simultanément. On traite de données de types caractère, numérique et date. La concordance des données se fait selon des combinaisons de

règles proposées par l'utilisateur, celles-ci devront être dynamiques et réutilisables. La sémantique des données traitées est prise en compte dans les fonctions *Match* et *Merge*.

En éliminant les doublons et les similaires dans une source de données, nous veillons à valider la dimension *Duplication* de la qualité de données (table 1.1). Cette dimension assure la présence de données non répétées dans la source et donc une meilleure qualité.

La dimension *Complétude* est assurée avec la fonction *Merge* que nous proposons. Elle permet l'enrichissement des données et ainsi traiter les valeurs nulles.

Comme perspective, nous allons essayer d'améliorer l'algorithme MFB4 afin qu'il gère aussi l'élimination des similaires.

Chapitre 8

Expérimentation : Nettoyage de données

Sommaire

8.1 Introduction

Nous allons dans ce chapitre détailler l'expérimentation faite sur les différents algorithmes d'élimination des doublons et similaires. Une application Java a été implémentée pour faciliter l'utilisation des algorithmes, recommander les mesures de similarité et les seuils adéquats et enfin gérer le taux d'élimination.

8.2 Application DQM

Afin de tester nos algorithmes et fournir aux utilisateurs une facilité d'utilisation de ces algorithmes, nous avons implémenté une application Java prenant en entrée une source de données (une table d'une BD sous Oracle, MySQL ou autre SGBD ou un fichier) (figure 8.1) contenant des doublons (doubles exacts) ou des similaires (figure 8.2). Elle renvoie une nouvelle source nettoyée (sans doublons).

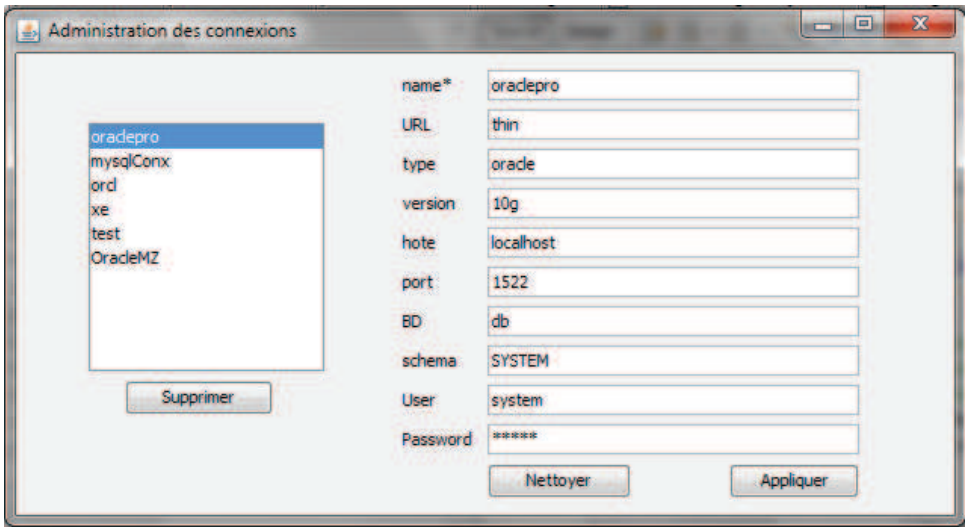


FIGURE 8.1 – Administration des connexions

Cette application propose d'autres actions en plus du dédoublonnage telles que le traitement des DF et la création de doublons. Cette dernière action permet de gérer le nombre de doublons et similaires dans une source et d'évaluer le taux d'élimination.

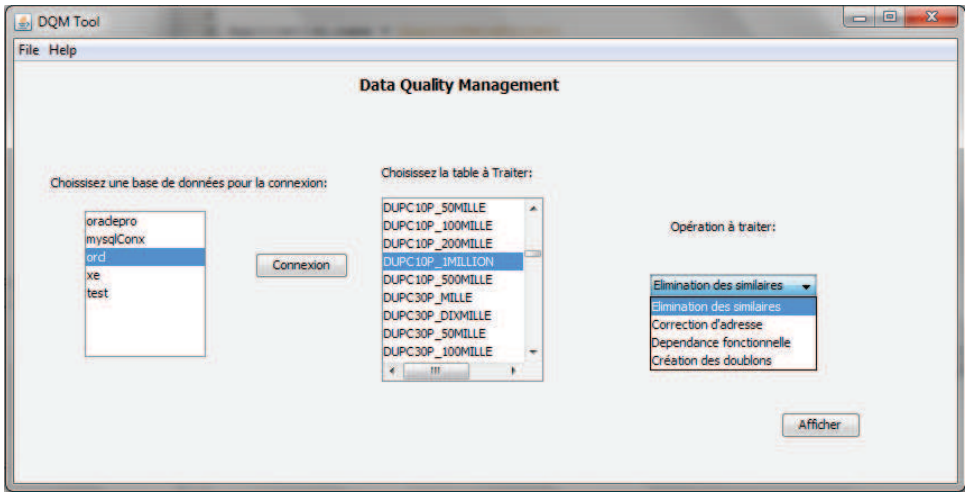


FIGURE 8.2 – Interface d'accueil de l'outil DQM

8.2.1 Outils et langages utilisés

Nous avons utilisé le langage de développement Java pour l'implémentation de notre application.

Langage Java[1] est un langage de programmation informatique orienté objet. Il reprend en grande partie la syntaxe du langage C++.

1. http ://fr.wikipedia.org/wiki/Java_(langage)

Java est un langage facilement portable sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java. Le langage Java reprend en grande partie la syntaxe du langage C++. Java permet de développer des applications client-serveur.

Les SGBD **Oracle 10g** et **MySQL** [2] ont été utilisés pour la création et la gestion des bases de données lors des tests.

MySQL est un SGBD open source. Il permet une création rapide et efficace des bases de données. MySQL fournit un shell interactif pour créer des tables, insérer et mettre à jour des données. La version utilisée lors de l'expérimentation (MySQL 5.2), fournit une interface utilisateur dynamique et simple d'utilisation.

8.2.2 Interfaces déduplication de l'outil QDM

Les différentes étapes de l'algorithme de déduplication sont modélisées en différentes interfaces de l'outil DQM.

Plusieurs interfaces sont créées en fonction des différentes étapes du processus d'élimination des doublons et similaires :

1. **Première interface :** Choix et configuration des attributs clés (Fonction Match).

   La fonction *Match* repose sur la sémantique des données. Pour le choix des attributs clés, nous recommandons les attributs dont le nombre d'utilisation (UT) est élevé ou ils ont été utilisés, dans une analyse ultérieure, comme étant des attributs de déduplication ($usedAsDeduplcationKey$).

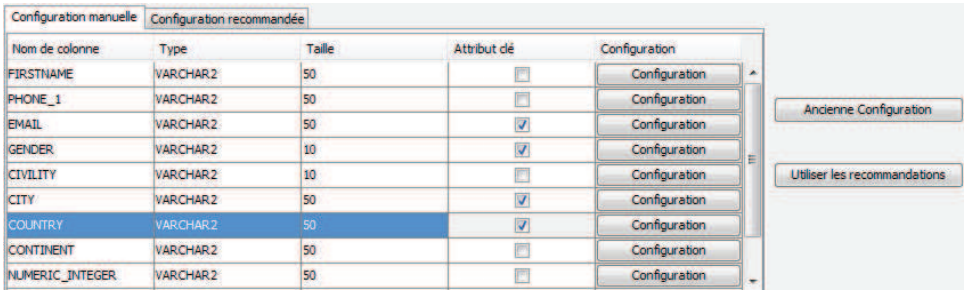


FIGURE 8.3 – Choix des attributs clés

Pour le choix des méthodes de calcul de similarité, nous avons établi une étude sur sur la meilleure méthode de recommandation en fonction de la

2. http ://www.mysql.com/

catégorie de l'attribut clé et en fonction de la sous-catégorie langue si elle existe. Nous allons détailler le choix des algorithmes de calcul de similarité pour les données de type chaîne de caractères.

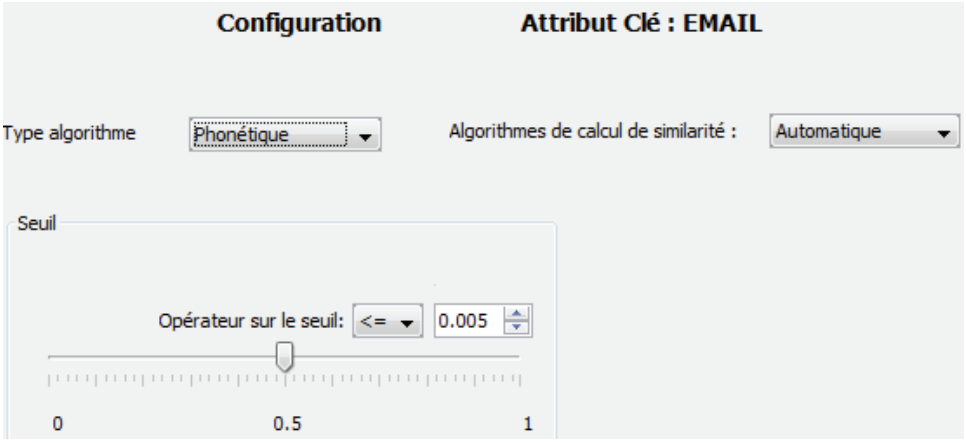


FIGURE 8.4 – Configuration des attributs clés

Outil de calcul de distance de similarité

Nous avons testé les algorithmes (Jaro-Winkler, Levenshtein, Jaccard, Soundex, DoubleMetaphone et NYSIIS) sur plusieurs milliers (10^5) de chaînes de caractères de différentes catégories (telles que FirstName, City, Country et Address) de différentes longueurs (entre 1 et 188).

Pour cela, nous avons développé un outil de calcul de mesure de distance de similarité (en Java) (figure 8.5) permettant de créer différents chaînes à partir d'une même en insérant ou supprimant ou remplaçant un ou plusieurs caractères dans différents emplacements de la chaîne. L'utilisateur pourra préciser l'emplacement de la modification ou bien c'est l'outil qui le gère aléatoirement. Une fois les chaînes sont crées, le calcul de distance de similarité est lancé en utilisant neuf mesures. Nous avons voulu tester l'impact d'une suppression, une insertion ou une modification dans une chaîne sur les méthodes de calcul de similarité.

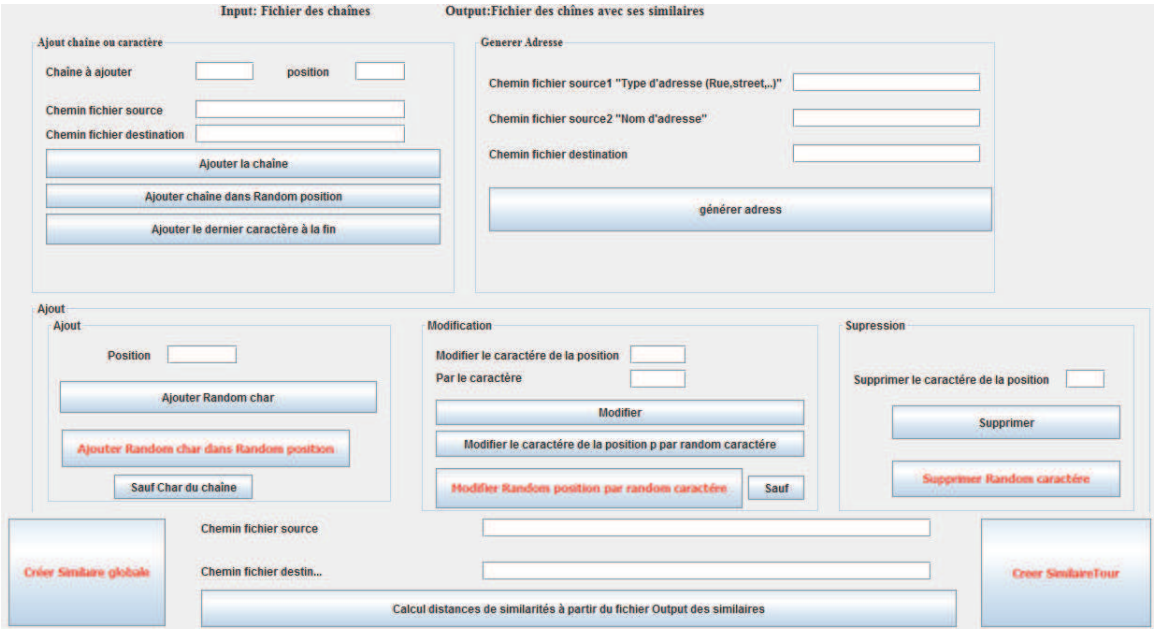


FIGURE 8.5 – Outil de calcul de distance de similarité

Le tableau ci-dessous (table 8.1) présente les moyennes en pourcentage des calculs des similarités sur des milliers de chaînes de caractères pour quelques catégories, renvoyées par l'outil présenté ci-dessus.

TABLE 8.1 – Moyennes de similarité pour différentes catégories

	JW	Jac	Lv	Qg	SdxLv	SdxJW	DMJW	NyLv	NyJW
FirstName	94,12	5,14	81,00	76,26	78,12	90,60	87,89	78,32	90,56
Country	95,97	25,33	86,73	83,13	83,34	93,24	92,79	83,62	94,09
City	96,34	25,48	86,32	83,84	84,49	93,59	93,86	84,62	94,44
Address	95,27	63,74	86,17	85,68	73,75	84,39	84,70	69,65	84,21
Email	96,98	59,44	93,80	92,48	91,28	96,68	96,85	91,15	96,90
PhoneFR	96,60	64,67	93,13	93,16	0,00	100,00	100,00	0,00	100,00
PhoneUK	96,60	58,22	92,43	92,04	0,00	100,00	100,00	0,00	100,00
Days	94,52	4,82	81,36	75,92	77,55	90,70	89,01	78,40	92,39
Date	95,32	86,67	89,17	91,79	0,00	100,00	100,00	0,00	100,00

avec JW (Jaro-Winkler), Jac (Jaccard), Lv (Levenshtein), SdxLv (Soundex combiné avec Levenshtein), SdxJW (Soundex combiné avec Jaro-Winkler), DMJW (DoubleMetaphone combiné avec Jaro-Winkler).

A travers les comparaisons effectuées, le tableau ci-dessus (table 7.4) récapitule les résultats de quelques catégories ainsi que la recommandation d'un algorithme selon la catégorie et le seuil par lettre de différence pour les chaînes de caractères.

Nous proposons alors une recommandation automatique pour les différentes catégories, les algorithmes de calcul de distance de similarité adéquats ainsi que les seuils (figure 8.4).

2. **Deuxième interface :** Configuration des attributs de fusion (Fonction Match). On propose les règles de fusion en fonction du type de données (figure 8.6, 8.7).

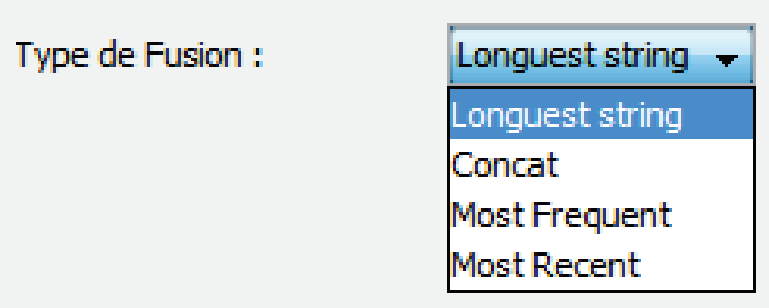


FIGURE 8.6 – Règles de fusion pour les chaînes de caractères

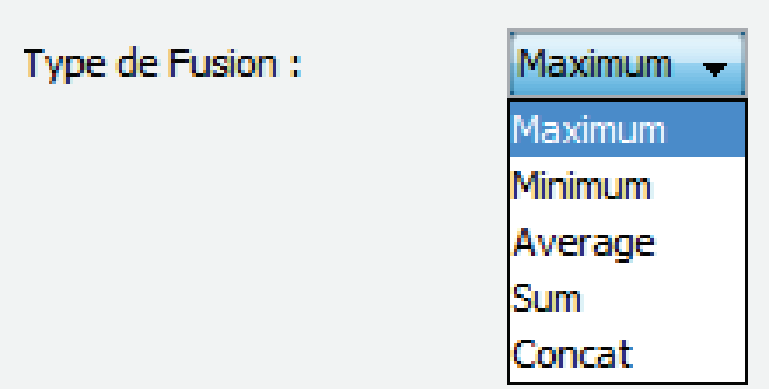


FIGURE 8.7 – Règles de fusion pour les numériques

3. **Troisième interface :** Configuration des règles de similarité (figure 8.8)

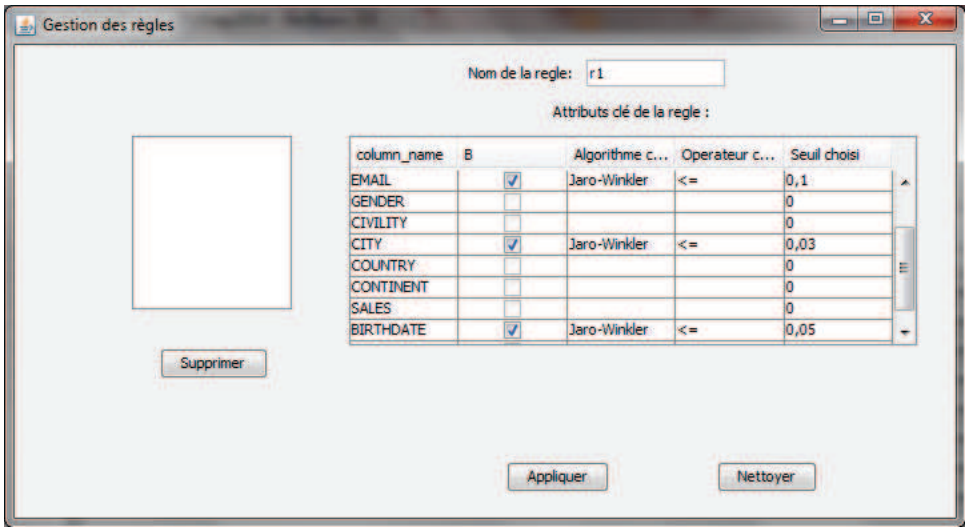


FIGURE 8.8 – Gestion des règles de similarité

Seuls les attributs clés peuvent être sélectionnés pour définir une règle. La sélection des autres attributs est grisée.

4. **Quatrième interface :** Choix de l'algorithme d'élimination de doublons et similaires (figure 8.9).

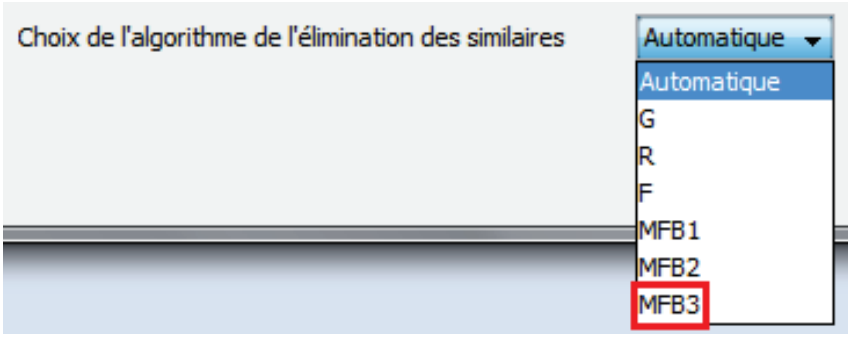


FIGURE 8.9 – Choix de l'algorithme d'élimination des doublons et similaires

Cette application nous a permis de faire plus facilement nos différentes mesures de performance.

8.3 Mesures de performance

Les données sont générées aléatoirement en utilisant notre générateur de gros volumes de données **GenLD** (Generator Large Data).

8.3.1 Generator Large Data

GenLD est un outil permettant de générer des grosses volumétries de données. L'action proposé dans le menu d'accueil de DQM (figure 1.4) fait appel à cet outil.

Il permet de contrôler le taux des doublons ou similaires dans une source générée (figure 8.10). Pour les doublons, on insère à la fin de la source le pourcentage des données en doubles.

Pour les similaires, on effectue différentes opérations d'ajout, de modification et d'insertion d'un ou plusieurs caractères. Le jeu de données similaire est aussi ajouté en queue de la source générée.

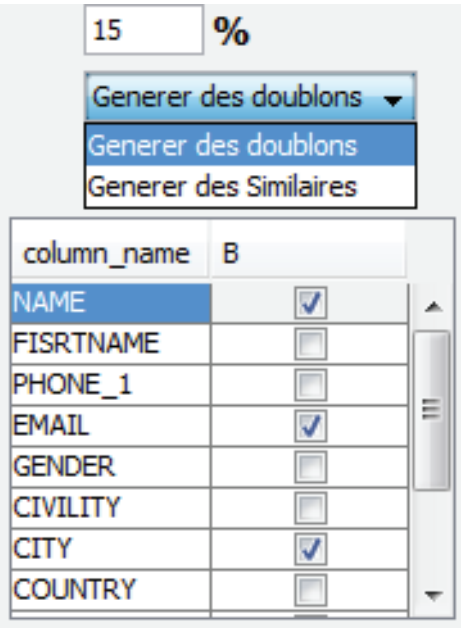


FIGURE 8.10 – Génération des doublons et similaires

Nous avons effectué les tests sur la table "Customer" dont la description SQL est la suivante :

CREATE TABLE CUSTOMER (FIRSTNAME varchar2(100), PHONE_1 varchar2(10), EMAIL varchar2(100), GENDER varchar2(20), CIVILITY varchar2(20), CITY varchar2(100), COUNTRY varchar2(100), CONTINENT varchar2(50), NUMERIC_INTEGER number, Column10_DATE date, PHONE_2 varchar2(100)) ;

Les attributs *NAME, CITY, EMAIL* et *DATEBIRTH* ont été recommandés pour l'élimination des similaires en fonction de leur UT (nombre d'utilisation) par rapport aux autres attributs. Les algorithmes de distance de similarité ainsi que les seuils ont été choisis en fonction de la catégorie des attributs.

Nous recommandons ainsi :
La mesure Soundex combinée avec Jaro-Winkler pour la catégorie *NAME* (cette catégorie a été rapprochée à la catégorie *FIRSTNAME*). Le seuil ε_1= 0,01.
Jaro-Winkler a été recommandée pour les catégories *City*, *Email* et *DATEBIRTH* avec :

— le seuil ε_2= 0,03 pour la catégorie *CITY*.
— le seuil ε_3= 0,1 pour la catégorie *EMAIL*.
— le seuil ε_4= 0,05 pour la catégorie *DATEBIRTH*.

L'utilisateur peut demander l'enrichissement de ces données afin d'éliminer, d'une part, d'éventuelles valeurs nulles, et d'autre part regrouper ou corriger certaines informations. Par exemple, dans la mesure où l'on veut avoir plusieurs numéros de téléphone pour un même client, la définition de l'attribut concerné (résultat de la fusion) devrait être suffisamment élargie, selon une règle métier, pour mettre x valeurs. Exemple, la taille du champ *PHONE_1* passe à varchar2(100) au lieu de *PHONE_1* varchar2(10). On multiplie la taille initiale 10 par x et dans ce cas x=10. Deux règles de similarité entre tuples ont été utilisées pour réaliser la fusion (Merge) :

$R_{ules} = r_1 \vee r_2$ avec :
$r_1 : (d_1 \leq \varepsilon_1) \wedge (d_2 \leq \varepsilon_2) \wedge (d_3 \leq \varepsilon_3)$;
$r_2 : (d_2 \leq \varepsilon_2) \wedge (d_4 \leq \varepsilon_4)$

La mesure de performance est effectuée sur trois étapes : une première comparaison des algorithmes G, R, F et MFB1 ; ensuite une deuxième expérimentation concernant MFB1 et MFB2 et enfin, une expérimentation sur les trois versions de MFB.

Remarque : Nos mesures sont exécutées sur un processeur 2.70GHz CPU Intel i7.

8.4 Conclusion

Nous offrons une application dynamique et facile à utiliser. La configuration des attributs clés et autres attributs est bien guidée. Une recommandation des mesures de similarité ainsi que les seuils facilitent la tâche aux utilisateurs.

Nous avons pu nettement améliorer le temps de réponse depuis la première version de l'algorithme jusqu'à la dernière version. L'approche de parallélisme pourra encore améliorer le temps de performance.

Chapitre 9

Apports dans Talend

Sommaire

9.1 Introduction

Cette thèse Cifre se déroule au sein de l'entreprise Talend [1]. Nous avons essayé d'intégrer nos différentes approches dans les outils Talend et en particulier l'outil Talend Data Quality, l'outil Talend Data Integration et l'outil MDM (Master Data Management).

9.2 Reconnaissance sémantique du schéma des données

L'union ou la jointure des données nécessite une certaine connaissance et compréhension des données. Cependant le méta-schéma d'une source peut ne pas exister avec l'air du BigData (les sources sont des fichiers csv, Json) ou il peut exister mais qu'il soit incompréhensible (absence de sémantique).

1. http ://talend.com/

Présentons ci-après (figure 9.1) un exemple de jointure pour deux sources avec chacune un schéma contenant peu de sémantique :

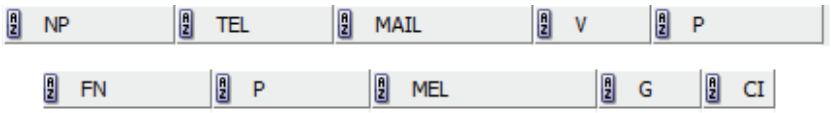


FIGURE 9.1 – Schémas de deux sources de données

Dans la plupart des ETL, la jointure est faite naturellement par l'utilisateur sur les colonnes de même nom S1.P et S2.P (figure 9.2) :

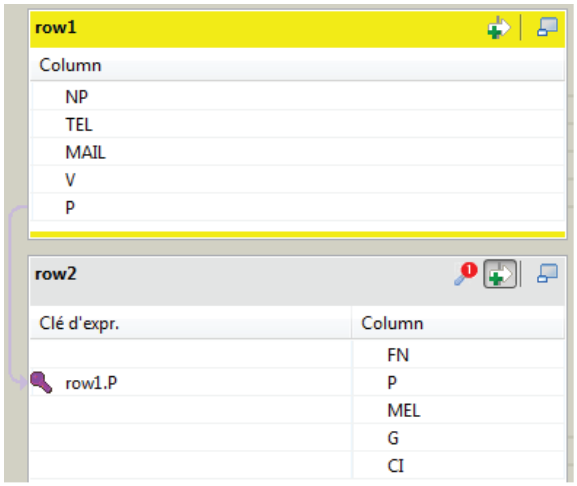


FIGURE 9.2 – Clé de jointure entre les deux sources

Or cette jointure n'est pas correcte à cause du contenu des deux sources de données (figure 9.3 et 9.4). Dans la première, S1.P contient les noms des pays et dans la deuxième source S2.P contient des numéros de téléphones (Phone) :

NP	TEL	MAIL	V	P
Le Bon Adam	0654321298	adam@yahoo.fr	Londres	Royaume-Uni
Lui Clémence	-	lui@yahoo.fr	Paris	France
Le Bon Adam	-	adam@yahoo.fr	Londre	RU
Adam Le Bon	0654321298	adam@yahoo.fr	London	Angleterre
Traifor Eve	0666321222	eve@yahoo.fr	Pékin	Chine
Traifor Eve	-	eve@yahoo.fr	Beijing	China

FIGURE 9.3 – Source de données S1

FN	P	MEL	G	CI
Le Bon Adam	0654321298	adam@yahoo.fr	M	Mr
Lui Clémence	-	lui@yahoo.fr	Female	
Le Bon Adam	-	adam@yahoo.fr	-	Mr
Adam Le Bon	0654321298	adam@yahoo.fr	M	Mr
Paris H	0132435462	parisH@yahoo.fr	F	Mme
H.Paris	0632435462	paris@live.fr	1	Mme

FIGURE 9.4 – Source de données S2

On peut conclure que nous avons besoin de profiler les sources de données avant leur intégration afin de donner un schéma sémantique aux données.

Profilage sémantique dans Talend

Pour l'intégration du profilage sémantique dans Talend, nous avons choisi l'utilisation des indexes Lucène pour la recherche dans les dictionnaires (DD et KW) au lieu des tables Oracle pour être indépendant des SGBD et pour des questions de performance.

En utilisant les indexes Lucène définis dans la section expérimentation de la première partie (section 5.3.1), nous avons créé deux indicateurs UDI (User Define Indicator) Talend en java : *Category Indicator* et *Language Indicator*. Nous avons choisi d'appliquer comme un premier travail la sous-catégorie langue. Ces indicateurs sont appliqués à chaque colonne de la source afin de déduire la nature et la langue de chaque valeur d'une colonne (C_i). La recherche d'une valeur v se fait dans l'indexes **Category** pour le premier indicateur et dans l'indexes **Language** pour le deuxième indicateur.

Chaque indicateur renvoie la liste des catégories (respectivement langues) détectées dans chaque colonne.

Exemple 9.1 : Application des deux indicateurs sémantique sur un échantillon d'une colonne

Soit l'échantillon d'une colonne C_i (figure 9.5), le résultat des deux indicateurs *Category Indicator* et *Language Indicator* est présenté dans les figures respectives (9.6, 9.7).

12 rue Carnot
2 avenue champs
r Carnot
square Carnot
112 boulevard saint germain
cinema gaumont
hotel ibis
115 rue Puteaux
1 boulevard Huassman
laboratoire lipn
3 Allée du 8 mai 1945

FIGURE 9.5 – Échantillon d'une colonne C_i

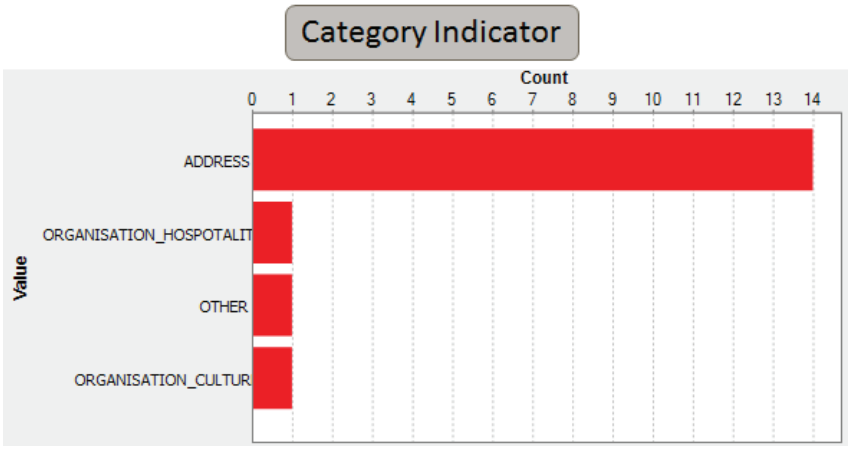


FIGURE 9.6 – Indicateur pour la détection de catégorie

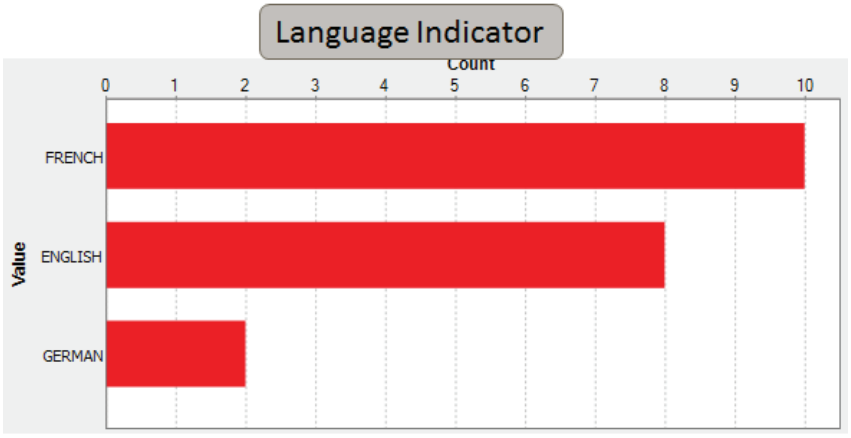


FIGURE 9.7 – Indicateur pour la détection de la langue

Dans l'ETL Talend, on propose une structure sémantique pour chaque source donnée en entrée d'une opération de jointure ou union. On recommande aussi les correspondances sémantiques existantes entre les colonnes des deux sources. Dans le cas (figure 9.3 et 9.4), on proposera la structure (Name, Phone, Email, City, Country) pour la source 1 et la structure (Name, Phone, Email, Gender, Civility) pour la source 2.

On aligne les deux schémas des deux sources (figure 9.8) et on présentera les correspondances entre les trois premières colonnes sous forme d'une boite de dialogue ("Warning") aux utilisateurs.

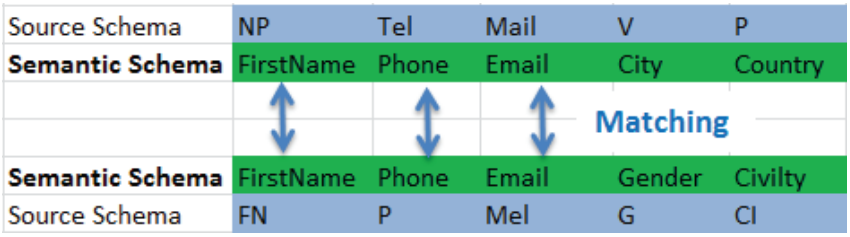


FIGURE 9.8 – Schéma sémantique et rapprochement de schémas

Cette aide sémantique déconseille la jointure sélectionnée par l'utilisateur (S1.P et S2.P) et fournit à l'utilisateur une liste de clés de jointure possible. C'est à l'utilisateur de confirmer les recommandations proposées.

Le profilage sémantique de données renvoie différents rapports d'anomalies ainsi qu'une nouvelle structure sémantique de données (figure 9.9). Ces résultats contiennent de la sémantique et pourront être utilisés dans la partie nettoyage.

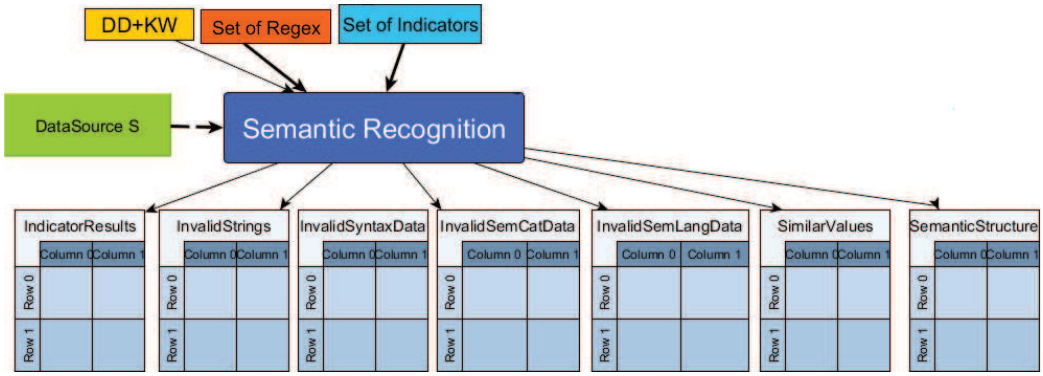


FIGURE 9.9 – Rapports des anomalies du profilage sémantique des données

La nouvelle structure sémantique est proposée lors de la conversion des sources d'un type à un autre. Par exemple, lors de la conversion d'un fichier csv en une table

Oracle, on propose un schéma sémantique pour la nouvelle table au lieu des noms in-significatifs proposés par défaut (Column 1, ..., Column n).

Autre objectif de ces travaux sémantiques est le rapprochement du schéma sémantique par rapport à un référentiel (instance du SCH2) afin de reconnaître les relations entre colonnes et recommander des analyses sur des sources de données.

L'outil TDQ fournit une liste d'indicateurs applicables sur différentes sources de données. Mais pour une première utilisation de l'outil, nous voulons assister l'utilisateur dans son analyse et lui recommander les indicateurs adéquats.

9.3 Alignement et recommandation sémantiques

Nous proposons dans cette partie une action « Suggest Semantic Analysis » sur chaque source de données que ce soit une table dans une BD ou un fichier (figure 9.10).

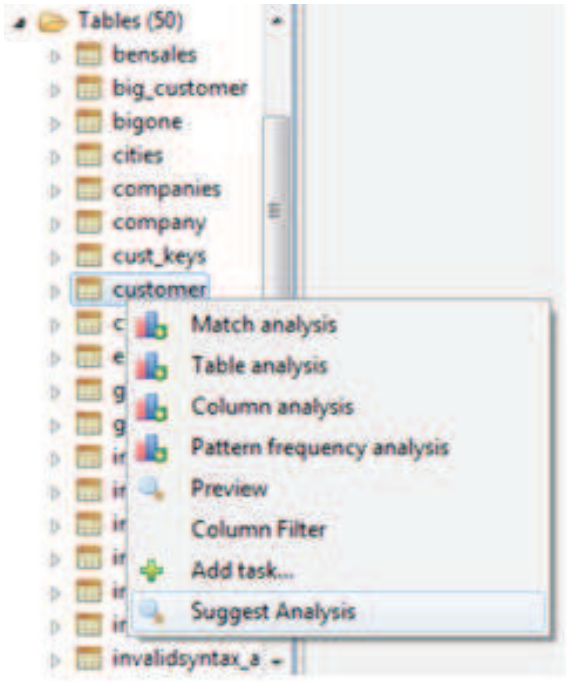


FIGURE 9.10 – Proposer une analyse sémantique

Cette action permet de proposer une liste d'indicateurs adéquate en fonction du schéma de la source. Cette action est proposée après avoir reconnu la structure sémantique de la source.

La recommandation repose sur trois aspects :

— recommandation des indicateurs basiques tels que nombre des valeurs nulles, nombres de valeurs distinctes, nombre de valeurs en doubles.
— aligner les noms des indicateurs, en particulier les expressions régulières, avec les noms sémantique des attributs et recommander les indicateurs proches.
— recommander les indicateurs qui étaient utilisé par des attributs synonymes aux attributs de la source.

Pour chaque source de données, on fournit un profil d'une analyse sémantique. Cette dernière se basant sur la reconnaissance du schéma et le rapprochement entre les données.

TDQ propose également une analyse de rapprochement (Matching Analysis). Cette analyse détecte la présence ou non des doublons et similaires dans une source. Elle nécessite une certaine connaissance/expertise de l'utilisateur afin de configurer les différentes parties de l'analyse telles que :

— choisir les attributs de partitionnement et les configurer en fonction des algorithmes de traitement.
— choisir les attributs de dédoublonnage (attributs clés) et les configurer en fonction des algorithmes de rapprochement (les distances de similarité, les seuils de confiance).

Sachant que le référentiel [[SCH2]] sauvegarde un ensemble de connaissances, l'alignement du schéma sémantique de la source avec le référentiel permet de reconnaître les attributs déjà utilisés dans des analyses similaires et proposer la même configuration à la source (figure 9.11).

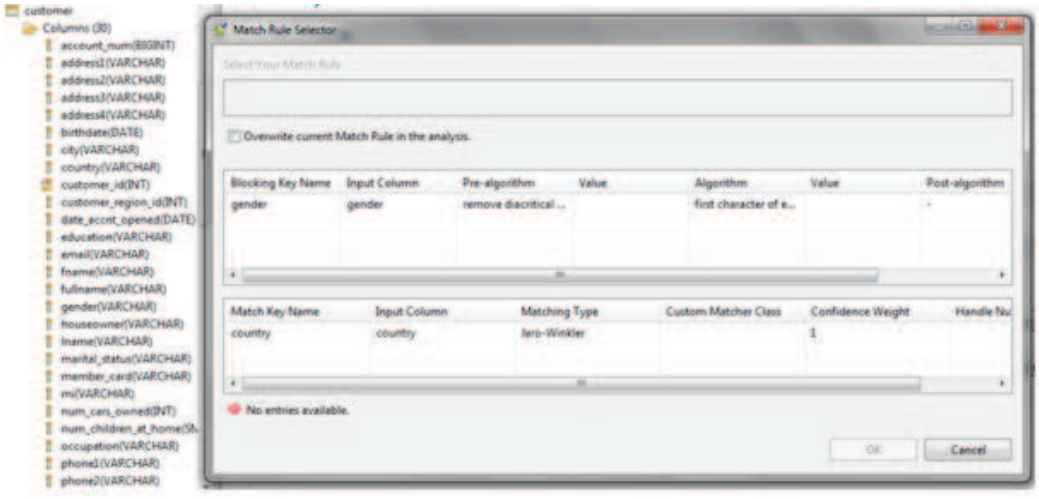


FIGURE 9.11 – Proposer une configuration pour l'analyse de correspondance

Dans le référentiel, l'attribut *Gender* a été utilisé comme un attribut de partitionnement avec le pré-algorithme "remove diacritical marks" et l'algorithme "first

character of each word". L'attribut *Country* a été utilisé comme un attribut clé avec l'algorithme de similarité Jaro-Winkler et un seuil de confidence 1.

9.4 Enrichissement du référentiel

Le référentiel ontologique doit être enrichi au fur et à mesure en fonction de nouvelles informations et connaissances. Les analyses des données permettent d'ajouter des informations sur les données analysées. Par exemple l'union de ces deux échantillons des sources de données (figure 9.12) renvoie un résultat faux à cause de l'absence du domaine de définition des deux colonnes « Note » et « Grade ».

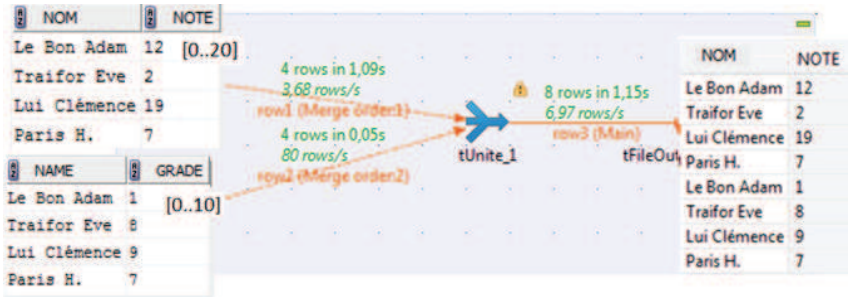


FIGURE 9.12 – Union des deux sources

La présence d'une information sémantique sur les domaines des deux attributs « Note » et « Grade » permet une meilleure union. Ces deux colonnes présentent la note des élèves dans une matière, notée par deux enseignants différents.

L'analyse sémantique de la colonne "Note" renvoie un domaine [0..20] de définition. Pour la colonne "Grade", le domaine défini est [0..10]. Une transformation d'une de ces colonnes est nécessaire pour une meilleure union (union sémantique) de ces deux sources. Un message d'avertissement avec ces informations sémantiques sera envoyé aux utilisateurs de Talend.

La dernière étape est le nettoyage de données. Les données requises auparavant facilitent la tâche de correction et nettoyage de données.

9.5 Nettoyage de données

Le résultat de profilage (structure sémantique) ainsi que les rapports d'anomalies sont utilisés afin de proposer différentes actions de nettoyage (figure 9.13).

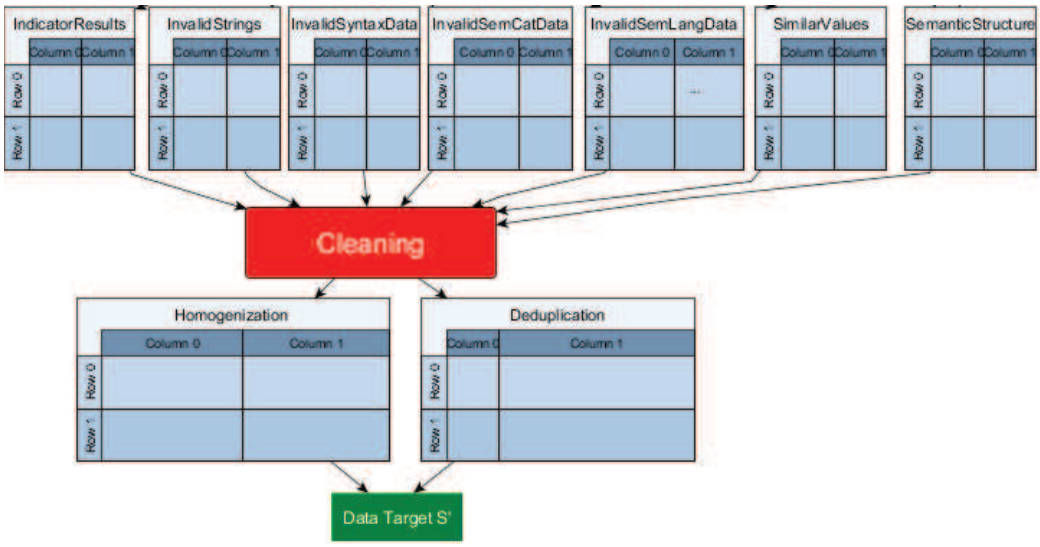


FIGURE 9.13 – Nettoyage de données en utilisant les rapports du profilage sémantique

Pour faciliter leur exploitation ils seront affichés dans une interface graphique dynamique pour une éventuelle communication avec les utilisateurs. Cette interface est en cours de construction par l'équipe "Data Quality" de Talend.

9.5.1 Homogénéisation

L'homogénéisation des données dépend des rapports d'anomalies, que ce soit une homogénéisation dans le même format, code ou langue (la langue a été choisie comme une sous-catégorie).

Des jobs Talend seront proposées à la fin de chaque rapport afin de fournir aux utilisateurs la possibilité de corriger leurs données :

— des jobs de correction syntaxique des valeurs retrouvées avec une recherche approximative dans le DD. Ces valeurs sont stockées dans le rapport des valeurs sémantiquement similaires, renvoyé par le profilage (chapitre 3.7).

— des jobs de codification des données selon un code ou format particulier. Un exemple de job est présenté dans la figure suivante (figure 9.14).

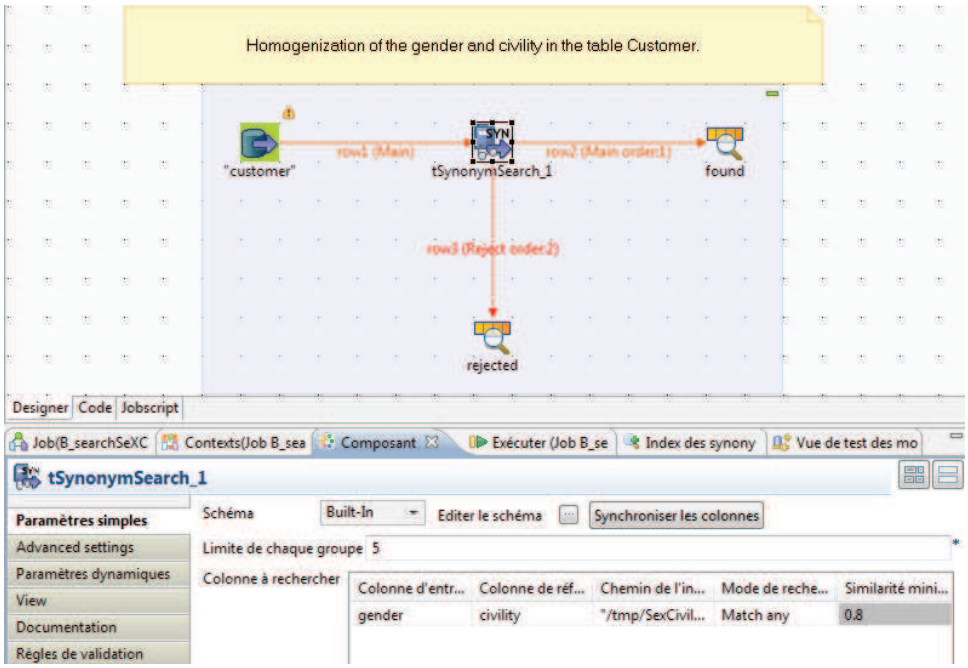


FIGURE 9.14 – Job d'homogénéisation de la codification des données *Gender/Civility*

Le format {F, M} est utilisé pour homogénéiser la colonne *Gender*. L'ensemble de valeurs {Miss, Madam, Mister} unifie la colonne *Civility* (figure9.15). Une dépendance entre ces deux colonnes est aussi respectée.

F	MISS
M	MISTER
F	MADAM
F	MISS

FIGURE 9.15 – Résultat d'homogénéisation *Gender/Civility*

— des jobs de traduction dans la langue dominante pour le rapport des valeurs invalides sémantiquement par rapport à la langue (figure 9.16). Chaque valeur de la source "Client1" n'étant pas dans la langue dominante est traduite en fonction d'un indexe Lucène ("Traduction"). Une nouvelle table, mise à jour, est proposée en sortie en attendant la confirmation de l'utilisateur.

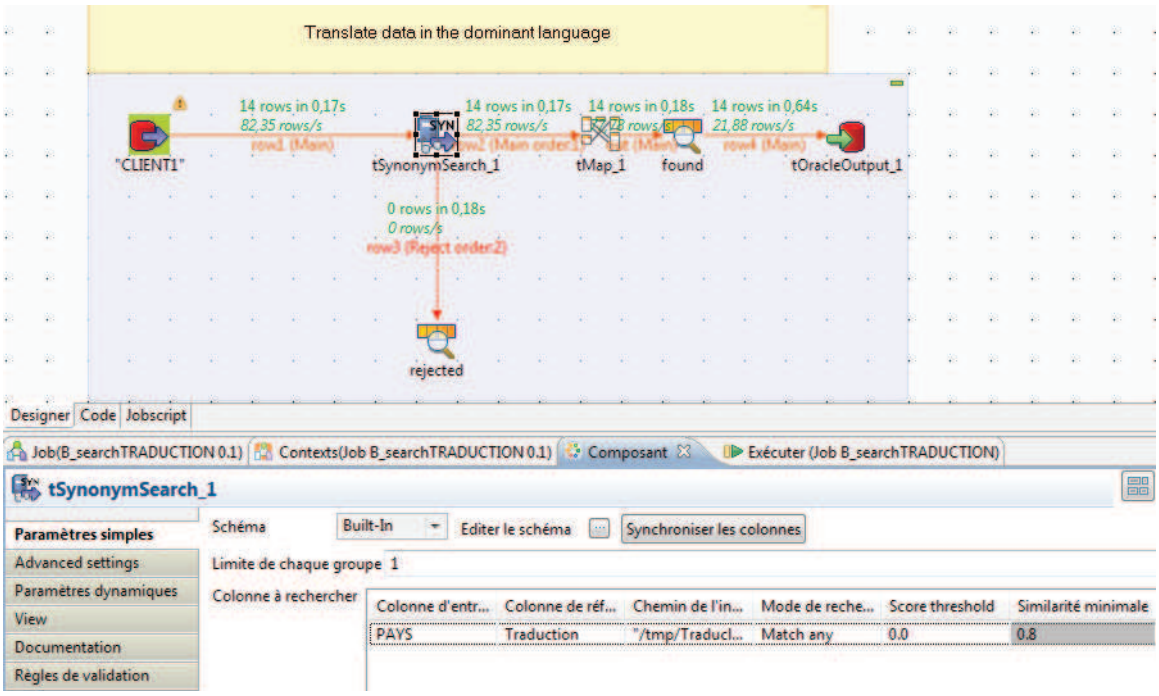


FIGURE 9.16 – Traduction des données en langue dominante

— des jobs pour la conversion de type des données en fonction de la nouvelle structure sémantique proposée.
— des jobs pour la vérification des dépendances fonctionnelles et le traitement des valeurs nulles.

9.5.2 Élimination des doublons et similaires

Une variante de MFB1 a été intégré dans l'outil MDM afin de garantir aux utilisateurs de ne créer et de n'insérer dans la base référentielle que les « golden record », des enregistrements uniques.

MFB1 (appelé dans Talend T-Swoosh) est utilisé dans la définition d'une règle de correspondance/rapprochement dans l'outil MDM (figure 9.17) afin de décider si deux ou plusieurs données correspondent.

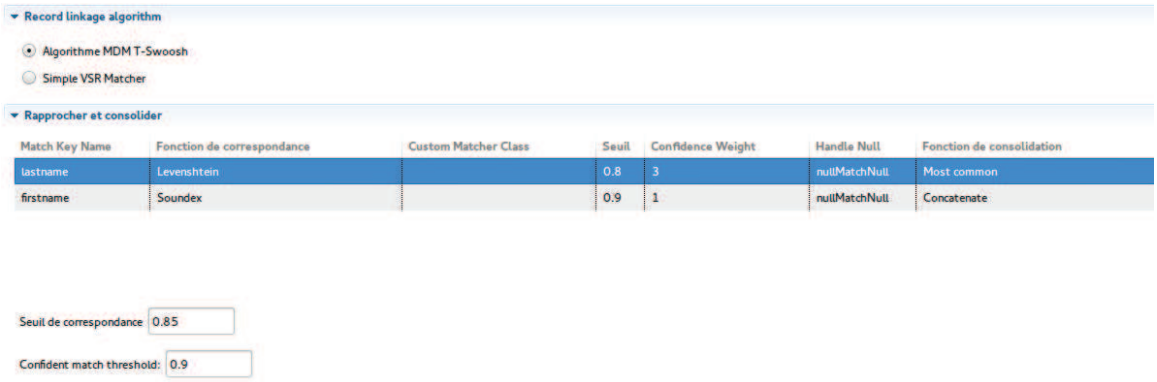


FIGURE 9.17 – Définition d'une règle de rapprochement dans MDM

L'analyse de rapprochement proposée par TDQ, utilise le principe de la fonction *Match*, définit dans MFB1, pour détecter les doublons dans une source de données. On définit aussi une ou plusieurs règles de rapprochement liées à une disjonction pour décider de la correspondance entre les données. Une règle est donc la conjonction d'un ou plusieurs attributs (figure 9.18).

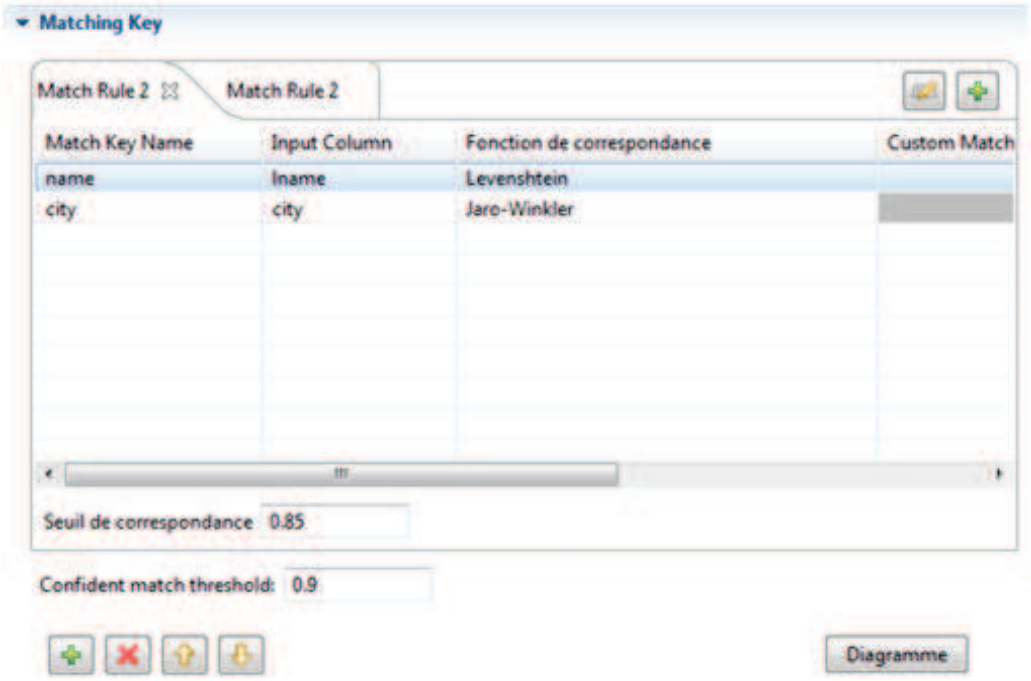


FIGURE 9.18 – Règles de rapprochement dans l'analyse de correspondance

L'algorithme MFB1 ainsi que les fonctions Match et Merge sont utilisés et bien intégrés dans les différents outils Talend.

9.6 Conclusion

Nous avons pu durant cette thèse, ajouter de la sémantique aux différents outils Talend (TDQ, MDM). Ces apports sémantiques ont pour objectif de faciliter l'utilisation de Talend. L'outil recommande aux utilisateurs des analyses sémantiques (en fonction du schéma sémantique défini pour chaque source), les méthodes de calcul de distance de similarité en fonction de la catégorie des données ainsi que le seuil adéquat. Une aide est proposée aux utilisateurs ainsi qu'une assistance dans sa démarche qualité.

Conclusion et Perspectives

Bilan

Cette thèse a pour objectif la détection et le nettoyage de données guidés par leur sémantique. En effet, la gestion de la qualité des données tant au niveau des bases de données gérées qu'au niveau des entrepôts construits à partir de ces dernières, constituent l'objectif principal. Pour cela, il faut rattacher aux données leurs sens sémantiques, leurs types et leurs contraintes. L'objectif est de mieux extraire, mélanger, interpréter et réutiliser les données afin de rétablir la confiance des décideurs dans les données. Avec l'air du BigData, les données gérées sont de plus en plus volumineuses. Les sources sont souvent mal renseignées, les schémas sont incompréhensibles (meaningless) voire inexistants (schemaless).

Un panorama bibliographique présente d'une part, les *approches de reconnaissance sémantique des schémas* de données de sources hétérogènes. Il est souligné que ces approches se basent essentiellement sur les comparaisons terminologique et structurelle des noms des objets. Ces derniers peuvent être peu significatifs ou contiennent peu de sémantique. Et d'autre part, *les approches de correction des données* qui ont abordé essentiellement la problématique d'élimination des doublons et similaires. En effet, les algorithmes présentés dans la littérature ne traitent pas vraiment de gros volumes de données. Ils ne prennent pas non plus en considération la sémantique de ces dernières dans les processus de comparaison et de fusion.

Les nombreux outils d'intégration de données et de gestion de la qualité de ces dernières (Data Integration and Data Quality) proposés ne répondent pas encore à tous les problèmes soulevés. Ils se focalisent le plus souvent uniquement sur la donnée brute et non sur la signification de celle-ci. Notre apport est donc d'initier de nouveaux outils, plus riches sémantiquement, pour assister les utilisateurs dans leurs démarches. Cette thèse aborde d'une manière originale, dans un premier temps, l'extraction de la sémantique des données à partir de toutes les informations disponibles (données, métadonnées, résultats d'analyses statistiques et éventuellement les informations fournies par l'utilisateur). Dans un deuxième temps, les méthodes de correction et nettoyage des données (correction intra colonne, inter colonnes et inter

lignes) sont présentées. Celles-ci profitent pleinement de la sémantique, issue de la première étape, dont elles disposent.

La **première contribution** de cette thèse concerne la reconnaissance de schéma d'une source de données. Cette approche est composée de deux étapes : (i) le *profilage sémantique* des données (reconnaissance en fonction des données) et (ii) *l'alignement sémantique* (reconnaissance du concept décrit par les attributs du profilage et des relations inter colonnes). Nous proposons un ensemble de rapports d'anomalies ainsi qu'une nouvelle structure sémantique des données. Cette reconnaissance facilite la tâche aux utilisateurs afin de comprendre leurs données, détecter les anomalies existantes et les corriger. Une amélioration dans le traitement de données a été aperçue. L'intégration des données est plus sûre.

La **deuxième contribution** est la proposition des actions de correction et nettoyage de données. La première étape consiste à *homogénéiser les données* (les corriger syntaxiquement, les unifier sous le même format, le même code et la même sous-catégorie). Ces modifications intra colonne sont suivies par des corrections inter colonnes, dictées par les liens sémantiques entre colonnes. Quant à la deuxième étape, l'objectif est d'apporter des corrections inter lignes. Elle concerne est la *déduplication des données*. Elle consiste à détecter et éliminer les doublons et similaires dans une source. Elle est fondue sur la sémantique établie.
Nous avons proposé des algorithmes séquentiels se basant sur deux fonctions *Match* et *Merge*. La première fonction *Match* permet d'évaluer et décider de la similarité entre deux enregistrements (tuples) en rapprochant les attributs clés de ces derniers. Cette similarité est calculée en tenant compte de la sémantique de l'attribut traité (catégorie, sous-catégorie et type de données) ainsi que la sous-catégorie utilisée (résultat de profilage sémantique pratiqué auparavant sur la source). La fonction *Merge* permet la fusion des enregistrements similaires afin de construire un nouveau tuple plus riche en information. Nous utilisons la technique de blocage en définissant comme clé de blocage la taille de la mémoire disponible.

Pour des questions de performance et d'optimisation, nous avons fait appel aux nouvelles notions de BigData lors de processus nettoyage des données. Le principe de MapReduce est utilisé tant au niveau de l'homogénéisation que celui de la déduplication.

Perspectives

Nombreuses sont les perspectives de recherche qu'offre cette thèse. En effet, l'hétérogénéité des domaines abordés ouvre différentes pistes.

La **reconnaissance sémantique des données** (*Semantic data schema recognition*) nécessite l'enrichissement des instances des deux méta-schémas : SCH1 (méta-schéma de catégorisation) et SCH2 (méta-schéma d'alignement). En effet, la classification des données en plusieurs catégories, requiert la définition d'un ensemble de critères qui porte sur le même concept sémantique.
L'automatisation de l'enrichissement des catégories (nouvelles instances du SCH1), tant au niveau des expressions régulières qu'au niveau des dictionnaires permettra d'affiner la reconnaissance sémantique des schémas des sources de données.
Quant aux instances du SCH2, l'exploitation des standards existants sur le Web devrait permettre leur enrichissement automatique. La reconnaissance des concepts et de l'ensemble des attributs (ainsi que leurs synonymes) qui les définissent demeure une tâche difficilement automatisable. Un autre type de classification, de ces descriptifs des données, est requis. La qualité des informations récupérées à partir du Web constitue le problème essentiel. Non seulement, la syntaxe des concepts et de leurs attributs descriptifs dépend de la langue, mais aussi des liens sémantiques éventuels entre les colonnes. Les commentaires à propos de ces dernières sont quasi inexistants. Même dans le cas où les commentaires sont renseignés, il faudrait inventer un processus de duplication pour chacun des synonymes.

Nous proposons l'utilisation des algorithmes d'apprentissage ainsi que l'inférence sur les ontologies pour la recommandation d'indicateurs à appliquer sur une colonne. Le résultat de ces derniers pourra enrichir les instances du SCH2.

Le **processus de nettoyage des données** (*Data cleaning process*) fait face à deux problèmes à savoir, d'une part, la sémantique des données considérées valides, et d'autre part, les performances des algorithmes utilisés.
En effet, l'ordre des priorités des actions à mener influence grandement la qualité des données finale. L'homogénéisation, le traitement des valeurs nulles, les liens inter colonnes et inter lignes sont des étapes dépendantes. Une réflexion très approfondie sur l'ordre des actions à entreprendre devrait être menée tant au niveau des performances que celui de la qualité du résultat.

Une étude plus détaillée sur les liens sémantiques tels que les dépendances fonctionnelles, les dépendances fonctionnelles conditionnelles et les dépendances d'inclusion entre colonnes devra être menée afin de garantir une meilleure véracité des données. Les dépendances minimales satisfaites par une source devrait tenir compte non seulement des données dans la source mais aussi de leur sémantique contextuelle afin de réduire la liste des dépendances.
L'exploitation de ces dépendances, par un humain, serait plus facile afin de mieux choisir les attributs de déduplication par exemple.

De point de vue algorithmique, de nouveaux travaux sont nécessaires tant au niveau sémantique qu'au niveau performance. En effet, les méthodes de calcul de distance de similarité devraient prendre mieux en compte la sémantique des chaînes traitées (catégorie et sous-catégorie) et ne pas considérer que la comparaison lexicographique.
Notons aussi que les règles de similarité dans la fonction Match peuvent toutefois présenter des incohérences. Il faut avoir un outil gestionnaire de la cohérence de ces règles.
Afin d'améliorer la performance de l'algorithme de déduplication par exemple, l'utilisation le concept de MapReduce (BigData), semble être une voie à mieux explorer surtout dans le cas de très grosses volumétries (de l'ordre des zéta-octets).

Bibliographie

Oracle Warehouse Builder Data modeling, ETL, and Data Quality Guide. In *Data Profiling*, 2011.

M. Badri, F. Boufarès, S. Hamdoun, V. Heiwy, and K. Lellahi. Construction and Maintenance of Heteregeneous Data WareHouse. In *Chapitre de livre Data WareHousing Design and Advanced Engineering Applications : Methods for Complex Construction*, pages 189–204, Editeur Adavnces in Data Warehousing and lining (ADWM) Book Series, Information Science, ISBN : 1935-2646, IGI Global Book Publishing, July 2009.

J. Barrasa, Ó. Corcho, and A. Gómez-Pérez. R2O, an Extensible and Semantically Based Database to-ontology Mapping Language. In *2nd Workshop on Semantic Web and Databases (SWDB)*, pages 1069–1070, Toronto, Canada, 2004.

S. Bechhofer. Ontologies and Vocabularies. In *the 9th Summer School on Ontology Engineering and the Semantic Web (SSSW'12)*, pages 1–53, Cercedilla, Spain, 2012.

J. Becker, M. Matzner, O. Müller, and A. Winkelmann. Towards a Semantic Data Quality Management -Using Ontologies to Assess Master Data Quality in Retailing. In *the Fourteenth Americas Conference on Information Systems (AMCIS'08)*, pages 1–11, Toronto, ON, Canada, 2008.

I. Bedini, B. Nguyen, and G. Gardarin. Building Reference Ontologies from B2B XML Schema Files. In *(PRiSM Laboratory Technical Report)*, pages 1–19, University of Versailles, 2007.

I. Bedini, B. Nguyen, and G. Gardarin. B2B Automatic Taxonomy Construction. In *(Tenth International Conference on Enterprise Information Systems Volume ISAS-2)*, pages 12–16, Barcelona, Spain, June 2008a.

I. Bedini, B. Nguyen, and G. Gardarin. Janus : AutomaticOntologyBuilderfrom XSD Files. In *Proceedings of the 17th International Conference World Wide Web Conference (WWW'08)*, Beijin, China, April 2008b.

A. Ben-Salem. Semantics in Data quality. In *Poster in the 9th Summer School on Ontology Engineering and the Semantic Web (SSSW'12)*, Cercedilla, Spain, July 2012.

A. Ben-Salem. Qualité des données & Grosses bases de données. In *Poster dans les Journées Big Data & Visualization (JBDMV'14)*, Paris, France, Juin 2013.

A. Ben-Salem. From Data quality to Data information. In *Poster dans l'École de Printemps sur l'Apprentissage arTificiel 2014 (EPAT'14)*, Carry-le-Rouet, France, Juin 2014.

A. Ben-Salem, F. Boufarès, and S. Correia. Semantic recognition of a data structrue in Big Data. In *6th International Conference on Computational Intelligence and software Engineering (CISE2014)*, pages 93–102, Beijing, China, July 2014.

O. Benjalloun, H. Garcia-Molina, D. Menestria, S.E. Whang Q. Su, and J. Widom. Swoosh : A Generic Approach to Entity Resolution. In *The 35 th International Journal on Very Large Data Bases (VLDB '09)*, pages 255–276, New York, USA, 2009.

Y. Bennani. Traitement Numérique des Données. France, 2006. Edition Hermes Science Publications.

S.M. Benslimane, D. Benslimane, M. Malki, Y. Amghar, and F. Gargouri. Construction d'une ontologie à partir d'une base de données relationnelle : approche dirigée par l'analyse des formulaires HTML. In *(INFORSID'06)*, pages 991–1010, Hammamet, Tunisie, 2006.

L. Berti-Équille. Qualité des données. In *Techniques de l'Ingénieur H3700, collection Technologies logicielles Architecture des systèmes*, pages 1–19, 2006.

L. Berti-Équille. Quality Awereness for managing and mining Data. In *HDR*, Rennes, France, 2007.

L. Berti-Équille. Panorama des méthodes de détection et de traitement des anomalies. In *5èmes Journées thématiques d'Apprentissage Artificiel & Fouille de Données (AAFD'12)*, pages 1–56, Villetaneuse, France, 2012.

M. Bilenko and R.J. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *the Ninth ACM SIGKDD International Conference on Knowledge Discovery, and Data Mining*, pages 39–48, Washington DC, USA, 2003.

A. Bilke and F. Naumann. Schema Matching using Duplicates. In *IEEE 21st International Conference on Data Engineering (ICDE'05)*, pages 69–80, Tokyo, Japan, 2005.

A. Bilke, J. Bleiholder, C. Bohm, K. Draba, F. Naumann, and M. Weis. Automatic Data Fusion with HumMer. In *the 31th International Conference on Very Large Databases (VLDB'05))*, pages 1251–1254, Trondheim, Norway, 2005.

C. Bizer. D2R MAP - A Database to RDF Mapping Language. In *Poster in 12th Conference in Wordl Wide Web (WWW03)*, Budapest, Hangary, May 2003.

J. Bleiholder and F. Naumann. Conflict Handling Strategies in an Integrated Information System. In *Humboldt-Universitatzu Berlin Unter den Linden 6 Berlin*, pages 1–6, Germany, 2006.

F. Boufarès. Etude théorique des valeurs nulles et des domaines sémantiques dans les Bases de Données : Application sur les SGBD Pépin. In *Rapport DEA Informatique Fondamentale, Laboratoire ISEM, DEA*, pages 1–87, Université Paris 11, France, septembre 1983.

F. Boufarès. Des Bases de Données aux Entrepôts de Données, Contribution au développement de nouveaux outils de gestion de la Qualité des Données. In *HDR en Informatique- Université Paris 13*, Villetaneuse, France, June 2012.

F. Boufarès and A. Ben-Salem. Heterogeneous data-integration and data quality : Overview of conflicts. In *6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'12), ISBN 978-1-4673-1657-6*, pages 867–874, Sousse, Tunisie, 2012.

F. Boufarès, A. Ben-Salem, and S. Correia. Qualité de données dans les entrepôts de données : élimination des similaires. In *8èmes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA'12), RNTI-B 2012*, pages 32–41, Bordeaux, France, 2012a.

F. Boufarès, A. Ben-Salem, and S.Correia. Un algorithme de déduplication pour les bases et entrepôts de données. In *Congrès INFormatique des ORganisations et Systèmes d ?Information et de Décision (INFORSID'12)*, pages 497–506, Montpellier, France, juin 2012b.

F. Boufarès, A. Ben-Salem, M. Rehab, and S. Correia. Similar Elimination data : MFB Algorithm. In *IEEE - 2013 International Conference on Control, Decision and Information Technologies (CODIT'13)*, pages 289 – 293, Hammamet, Tunisie, mai 2013.

I. Boydens. Evaluer et améliorer les qualités des bases de données. In *Techno 7, Publication technique de la Smals-MvM*, Bruxelles, 1998.

S. Castano, V. De Antonellis, and S. De Capitani Di Vimercati. Global viewing of heterogeneous data sources. In *Journal of Knowledge and Data Engineering, IEEE Transactions on (Volume :13 , Issue : 2)*, pages 277–297, 2001.

F. Cerbah. Learning Highly Structured Semantic Repositories from Relational Databases - RDBtoOnto Tool. In *in the 5th European Semantic Web Conference (ESWC 2008)*, pages 777–781, Tenerife, Spain, June 2008.

P. Christen. Febrl - A Freely Available Record Linkage System with a Graphical User Interface. In *Australasian Workshop on Health Data and Knowledge Management (HDKM'08), volume 80*, pages 17–25, New South Wales, Australia, 2008.

E. F. Codd. A Relational Model of Data for Large Shared Data Banks. In *(Communication of the ACM, Vomule 13, Number 6)*, pages 377–387, June 1970.

W.W. Cohen. Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity. In *1998 ACM SIGMOD International Conference in Management of Data (SIGMOD'98),*, pages 201–212, 1998.

W.W. Cohen and J. Richman. Iterative Record Linkage for Cleaning and Integration. In *9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD'04)*, pages 11–18, Paris, France, 2004.

A. Cornuéjols. Introdution à l'Apprentissage Artificiel. In *Ecole de Printemps sur l'Apprentissage arTificiel (EPAT'14)*, Carry le Rouet, France, 2014.

M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I.F. Ilyas, M. Ouzzani, and Nan Tang. NADEEF : a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of Data (SIGMOD'13)*, pages 541–552, New York, USA, June 2013.

J. Dean and S. Ghemawat. MapReduce : simplified data processing on large clusters. In *The 6th Conference On Symposium On Operating Systems Design And Implementation (OSDI'04)*, pages 137–150, California, USA, December 2004.

D. Dey, S. Sarkar, and P. De. Entity Matching in Heterogeneous Databases : A Distance Based Decision Model. In *31st Ann. Hawaii International Conference in System Sciences (HICSS)*, pages 305–313, 1998.

H.H. Do and E. Rahm. COMA : a system for flexible combination of schema matching approaches. In *the 28th International Conference on Very Large Data Bases (VLDB'02))*, pages 610–621, Hong Kong, China, 2002.

A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources : a machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD'01)*, pages 509–520, California, USA, May 2001.

X.L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data : the role of source dependence. In *the 35th International Conference on Very Large Databases (VLDB'09)*, pages 550–561, Lyon, France, August 2009.

M.G. Elfeky, A.K. Elmagarmid, and V.S. Verykios. TAILOR : A Record Linkage Tool Box. In *18th IEEE International Conference Data Eng. (ICDE'02)*, pages 17–28, 2002.

A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios. Duplicate Record Detection : A Survey. In *IEEE Transactions on Knowledge and Data Engineering, VOL. 19, NO. 1*, pages 1–16, 2007.

J. Euzenat. Ontology matching. In *the 9th Summer School on Ontology Engineering and the Semantic Web (SSSW'12)*, pages 1–9, Cercedilla, Spain, 2012.

J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *IEEE Transactions on Knowledge and Data Engineering, Volume :25, Issue :1*, pages 1–6, 2006.

W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about Record Matching Rules. In *(VLDB'09)*, pages 24–28, Lyon, France, August 2009.

C. Faucher, F. Bertrand, and J-Y.Lafaye. Génération d'ontologie à partir d'un modèle métier UML annoté. In *Revue des Nouvelles Technologies de l'Information RNTI-B*, pages 65–84, France, 2008.

D. Forest and J.G. Meunier. Classification et catégorisation automatiques : application à l'analyse thématique des données textuelles. In *7ème Journées internationales d'Analyse statistique des Données Textuelles (JADT'04)*, pages 433–444, Louvain La Neuve, Belgique, 2004.

H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An Extensible Framework for Data Cleaning. In *International conference on Data Engineering (ICDE*, pages 1–32,, San Diego, California, USA, 2000.

R. Ghawi, N. Cullot, and K. Yétongnon. DB2OWL : A Tool for Automatic Database-to-Ontology Mapping, Symposium on Advanced Database Systems. In *Conference on Advanced Database Systems (ADS)*, pages 491–494, Torre Canne di Fasano (BR), Italy, 2007.

F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic Schema Matching. In *Proceedings 13rd International Conference on Cooperative Information Systems (CoopIS'05)*, pages 347–365, Grenoble, 2005.

B. Glenn and A. Sethi. Matching records in a national medical patient index. In *Communications of the ACM, Volume 44, Issue 9*, pages 83–88, September 2001.

S. Gong. A collaborative filtering recommendation algorithm based on item classification. In *(Journal of software, vol 5, No. 7)*, pages 745–752, july 2010.

S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the Results of Approximate Match Operations. In *30th International Conference in Very Large Databases (VLDB'04)*, pages 636–647, Toronto, Canada, 2004.

S. Hamdoun and F. Boufarès. Un formalisme pour l'intégration de données hétérogènes. In *Revue des Nouvelles Technologies de l'Information, (RNTI), B-6, Entrepôts de Données et l'Analyse en ligne (EDA'10)*, pages 107–119, Djerba, Tunisie, Juin 2010.

M. Hammou, A. Ben-Salem, and F. Boufarès. Gestion de la qualité des données : Aide à la compréhension du schéma de données ; élimination des similaires. In *Mémoire de stage de Master2 Informatique, option Exploration Informatique de données et décisionnel, Université Paris 13*, pages 1–50, 2013.

I. Herman. Semantic Web Activities@W3C. In *9th Summer School on Ontology Engineering and the Semantic Web (SSSW'12)*, pages 1–89, Cercedilla, Spain, 2012.

M. Hernandez and S. Stolfo. Utility-based resolution of data inconsistencies. In *International Workshop on Information Quality in Information Systems (IQIS)*, pages 35–43, Maison de la Chimie, France, 2004.

M.A. Hernandez and S.J. Stolfo. The merge/purge problem for large databases. In *the ACM International Conference on Management of Data (SIGMOD'95)*, pages 127–138, 1995.

M.A. Hernandez and S.J. Stolfo. Real-world Data is Dirty : Data Cleansing and The MergePurge Problem. In *Data Mining and Knowledge Discovery (DMKD'98)*, pages 9–37, New York, USA, 1998.

T. Johnson and T. Dasu. A Data Quality Browser. In *Proceedings of the Sixth International Conference on Information Quality*, pages 233–243, AT&T Labs - Research, 2001.

D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. In *Journal ACM Transactions on Database Systems (TODS) TODS Homepage archive Volume 31 Issue 2*, pages 716–767, 2006.

M. Kamel and N.Aussenac-Gilles. Construction automatique d'ontologies à partir de spécification de bases de données. In *Actes des 20èmes Journées Francophones d'Ingénierie des Connaissances (IC)*, pages 85–96, Hammamet, Tunisia, 2009.

S. Hamdoun Khalfallah. Construction d'entrepôts de données par intégration de sources hétérogènes. In *Thèse de doctorat en Informatique- Université Paris 13*, pages 1–189, Villetaneuse, France, December 2006.

E.M. Knox and R.T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proceedings of the 24th International Conference in Very Large Databases (VLDB'98)*, pages 392–403, New York, USA, 1998.

N. Koudas, S. Sarawagi, and D. Srivastava. Record Linkage : Similarity Measures and Algorithms. In *the ACM International Conference on Management of Data (SIGMOD'06)*, pages 802–803, Chicago, Illinois, USA, 2006.

H. Köpcke and E. Rahm. Frameworks for entity matching : A comparison. In *Data Knowledge Engineering (DKE'09)*, pages 197–210, Leipzig, Germany, 2009.

S. Krivine, J. Nobécourt, L. Soualmia, F. Cerbah, and C. Duclos. Construction automatique d'ontologie à partir de bases de données relationnelles : application au médicament dans le domaine de la pharmacovigilance. In *Actes des journées francophones d'Ingénierie de Connaissances (IC'09)*, pages 73–84, Hammamet, Tunisie, 2009.

Cybozu Labs. Language Detection Library for Java. `https://code.google.com/p/language-detection/`, 2010.

J.A. Larson, S.B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. In *Software Engineering, IEEE Transactions on (Volume :15 , Issue : 4)*, pages 449 – 463, April 1989.

V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Doklady Akademii Nauk SSSR, vol. 163, no. 4*, pages 845–848, 1966.

W. Li and C. Clifton. SEMINT : A tool for identifying attribute correspondences in heterogeneous databases using neural networks. In *Journal of Data and Knowledge Engineering, Volume 33 Issue 1*, pages 49–84, 2000.

P. Liegl, M. Zapleta, C. Pichler, and M. Strommer. State-of-the-art in business document standards. In *8th IEEE International Conference on Industrial Informatics (INDIN)*, pages 234–241, Osaka, Japan, 2010.

E. P. Lim, J. Srivastava, S. Prabhakar, and J. Richardson. Entity Identification in Database Integration. In *Ninth IEEE International Conference in Data Engineering (ICDE)*, pages 294–301, 1993.

J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pages 49–58, Roma, Italy, September 2001.

S. Madnick and H. Zhu. Improving data quality through effective use of data semantics. In *Working paper CISL#2005-08*, pages 1–19, 2005.

B. Meddah, A. Ben-Salem, and F. Boufarès. Qualité de données. In *Mémoire de stage de Master2 Informatique, option Exploration Informatique de données et décisionnel, Université Paris 13*, pages 1–42, 2014.

W. Mefteh, A. Bouju, and J. Malki. Cadre applicatif pour la construction d'ontologie basée sur un modèle conceptuel UML2 et la réutilisation des ontologies. In *Atelier construction d'ontologies GBPOnto*, pages 1–12, France, 2009.

D. Menard. Schémas de bases de données. `http://www.ascodocpsy.org/santepsy/AutoDoc/?filename=fab.schemas#fab.schemas.introduction`, 2008.

N.F.Noy and D.L. McGuinness. Ontology Development 101 : A Guide to Creating Your First Ontology. In *Stan- ford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, pages 1–25, 2001.

C. Nyulas, M.O'Connor, and S. Tu. DataMaster - a Plug-in for Importing Schemas and Data from Relational Databases into Protégé. In *10 th International Protégé Conference*, pages 1–3, Budapest, Hungary, July 2007.

P. Oliveira, F. Rodrigues, P. Henriques, and H. Galhardas. A Taxonomy of Data Quality Problems. In *inproceedings universitaire*, pages 1–15, 2005.

V. Peralta. *Data quality evaluation in data integration systems.* PhD thesis, Versailles, France, 2006.

L. Philips. The Double Metaphone Search Algorithm. In *C/C++ Users J., vol. 18, no. 5*, pages 38–43, 2000.

S. Chaudhuri R. Ananthakrishna and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *the 28th international conference on Very Large Data Bases (VLDB'02)*, pages 586–597, Hong Kong, China, 2002.

E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. In *VLDB Journal : Very Large Data Bases, Vol. 10, No. 4*, pages 334–350, September 2001.

M. Rehab-Adjout and F. Boufarès. A Massively Parallel Processing for the Multiple Linear Regression. In *The 10th International Conference on Signal Image Technology & Internet Based Systems*, Marrakesh - Morocco, November 2014.

F. Sais. *Intégration sémantique de données guide par une ontologie*. PhD thesis, Paris, 2007.

S. Sarawagi and A. Bhamidipaty. Interactive Deduplication using Active Learning. In *Proceedings The 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 269–278, Alberta, Canada, 2002.

F. Saïs and R.Thomopoulos. Ontology-Aware Prediction from Rules : A Reconciliation-Based Approach. In *(International Journal on Knowledge-Based Systems)*, pages 117–130, juin 2014.

P. Shvaiko and J. Euzenat. Ontology Matching : State of the Art and Future Challenges. In *IEEE Transactions on Knowledge and Data Engineering, Volume :25, Issue :1*, pages 158–176, 2013.

E. Simonenko and N. Novelli. Extration de dépendances fonctionelles approximatives : une approche incrémentale. In *Extractions et Gestions des Connaissances, RNTT E.23 (EGC'12)*, pages 95–100, Bordeaux, France, 2012.

A. Souid, A. Ben-Salem, and F. Boufarès. Qualité des données : Aide à la compréhension des schémas ; Profilage des données ; Dépendances fonctionnelles. In *Mémoire de stage de Master2 Informatique, option Programmation et Logiciel Sûrs, Université Paris 13*, pages 1–50, 2013.

G. Stoilos, G. Stamou, and S. Kollias. A String Metric for Ontology Alignment. In *4th International Semantic Web Conference (ISWC'05), LNCS 3729*, pages 624–637, Galway, Ireland, 2005.

M. Stricker. Réseaux de neurones pour le traitement automatique du langage : conception et réalisation de filtres d'informations. In *Thèse de Doctorat de l'Université Pierre et Marie Curie - Paris VI*, Paris, France, 2000.

C. Toulemonde. JEMM research_Informatica : Le capital de votre organisation. In *Un livre blanc de JEMM research - Des données de qualité*, Bordeaux, France, 2008.

E. Ukkonen. Approximate String Matching with q-Grams and Maximal Matches. In *Theoretical Computer Science, vol. 92, no. 1*, pages 191–211, 1992.

R. Wang and D. Strong. Beyong accuracy : what data quality means to data consumers. In *Journal of management information systems*, pages 5–34, 1996.

Y. R. Wang and S.E. Madnick. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In *Proceedings Fifth IEEE International Conference on Data Engineering (ICDE)*, pages 46–55, New York, USA, 1989.

W.E. Winkler. Overview of Record Linkage and Current Research Directions. In *Research Report Series, RRS*, pages 1–44, Washington, USA, 2006.

W.E. Winkler and Y. Thibaudeau. An Application of the FellegiSunter Model of Record Linkage to the 1990 US Decennial Census. In *Technical Report Statistical Research Report Series RR91/09, US Bureau of the Census*, pages 1–22, Washington, USA, 1991.

J-H. Hamilton X. Wang and Y. Bither. An ontology-based approach to data cleaning. In *Technical Report CS-2005-05*, pages 1–10, 2005.

M. Zayen, F. Boufarès, A. Ben-Salem, and M. Rehab-Adjout. La technologie MapReduce (Hadoop/Spark) au service de la qualité des données : Élimination des doubles et des similaires dans les grosses masses de données. In *Mémoire de stage de Master2 Informatique, option Système d'Information et Décision, Université Paris 1*, pages 45–52, 2014.

D. Zhang. Basic MapReduce Algorithm Design. `http://www.dcs.bbk.ac.uk/~dell/teaching/cc/book/ditp/ditp_ch3.pdf`.

Annexe A

Annexe

A.1 Aperçu sur le web sémantique

Présentons dans ce qui suit un aperçu sur les ressources sémantiques et particulièrement les ontologies.

A.1.1 Les ressources sémantiques

Le traitement automatisé des données requiert une représentation de la sémantique compréhensible et échangeable par les machines, d'où le besoin du web sémantique.

Le web sémantique est un vaste espace d'échange de ressources entre humains et machines permettant une meilleure exploitation de masse de données disponibles sur le web.

Le défi du web sémantique est de fournir un langage qui (i) exprime à la fois les données et les règles de raisonnement sur ces données et ; (ii) permet aux règles de n'importe quel système de représentation des connaissances d'être transférées sur le Web (le partage d'information et l'interopérabilité).

Parmi les ressources sémantiques citons quelques-uns (Bechhofer, 2012) :

- Ontologie : est une bibliothèque de termes ou des définitions de concepts, qui décrivent la structure de l'information pour un domaine donné ou une activité particulière. Elle désigne l'ensemble des concepts d'un domaine ainsi que leurs relations. Elle est employée pour raisonner à propos des objets du domaine concerné.
  " L'ontologie est aux données ce que la grammaire est au langage "[] .
  L'architecture d'une ontologie :
  Toute ontologie est composée au moins de ces différents éléments :

— Concepts (classes) : ensemble, collection d'objets du monde réel.
— Relations : interaction entre concepts (relations sémantiques, relations de subsomption (inclusion)).
— Individus : les objets de base,
— Attributs : caractéristiques, spécificités particulières, définir de manière unique un à domaine
— Événements : changements subis par des attributs ou des relations.

Ces éléments peuvent être enrichit par d'autres composants.

- Vocabulaire contrôlé : C'est un ensemble de termes définis par un groupe (une communauté de pratiques) afin de pouvoir labelliser des contenus, écrire un document. La signification des termes n'est pas forcément définie et il n'y a pas nécessairement d'organisations logiques des termes entre eux. Par exemple, le glossaire d'un livre ou encore des catégories dans un système de carnets Web partagés entre différents auteurs.
- Taxonomie : Dans une taxonomie, le vocabulaire contrôlé est organisé sous forme hiérarchique simple. Cette hiérarchisation correspond souvent à une spécialisation. Il existe donc un lien précis entre un terme du vocabulaire et ses enfants. Ce lien donne un sens supplémentaire, une signification. D'un vocabulaire contrôlé, on passe à un vocabulaire organisé. Par exemple, dans une classification animale, nous aurons les vertébrés, invertébrés et puis sous les vertébrés nous auront les mammifères, les ovipares, etc. Tous ces termes nous permettront de classifier les animaux. On pourra donc dire que Les mammifères sont une sous-catégorie (sous-classe) des vertébrés.
- Thésaurus : est une liste organisée de termes contrôlés et normalisés (descripteurs et non descripteurs) représentant les concepts d'un domaine de la connaissance.

  C'est un langage contrôlé utilisé pour l'indexation de documents et la recherche de ressources documentaires dans des applications informatiques spécialisées. Les thésaurus sont donc une catégorie de langages documentaires parmi d'autres. Les termes (dans l'exemple ci-contre : véhicule, navire,...) sont reliés entre eux par des relations de synonymie (terme équivalent), de hiérarchie (terme générique et terme spécifique) et d'association (terme associé) ; chaque terme appartient à une catégorie ou domaine.

  — Les relations entre concepts sont de trois types :
    — Relation hiérarchique : base de la hiérarchie du thésaurus. Elles sont représentées par les sigles TG (terme générique) et TS (terme spécifique).
    — Relation d'association : enrichissant le réseau de relations hiérarchiques selon d'autres axes de type sujets connexes. Ces relations

peuvent être de nature très variée : causalité, localisation, relations de nature temporelle, composition, etc.

— Appartenance à un groupe de concepts : il est courant de sélectionner et regrouper des concepts selon un critère spécifique, tels que leur pertinence à un domaine particulier. Ces regroupements de concepts sont appelés suivant les contextes : thèmes, domaines, champs sémantiques, micro thésaurus (MT).

— Relations entre les termes représentant les concepts, relations d'équivalence Les relations d'équivalence entre termes représentant un même concept permettent de lutter contre la polysémie.

- Topic Maps « est une structure abstraite organisée autour de Topics représentant des sujets que le créateur de la Topic Maps souhaite décrire et pour lesquels des ressources sont disponibles pour fournir de la connaissance sur ces sujets et de relations (ou associations) entre ces Topics » []. Pour définir un topic Maps, il faut définir (i) les topics (concepts) qui représentent les objets d'intérêt, (ii) les associations qui représente les relations entre les topics et (iii) les occurrences qui permet de connecter les topics à des ressources d'information avec des informations pertinentes.

Nous présentons dans le tableau suivant (table A.1), un bilan sur les différentes ressources sémantiques :

TABLE A.1 – Ressources sémantiques (Herman, 2012)

Ressources sémantiques	Définition	Particularité
Vocabulaire contrôlé	Liste définie de termes	Permet de contrôler les termes associés aux sujets
Taxonomie	Liste de termes contrôlés organisés de façon hiérarchique	Facilite la recherche de termes à partir de relations hiérarchiques
Thésaurus	Réseau de termes contrôlés, enrichi par des relations associatives prédéfinies	Facilite la recherche de termes en fonction de différents types de relations (pas seulement hiérarchiques)
Ontologie	Modèle de description des connaissances basé sur des concepts avec types, propriétés et relations	Représentation des connaissances : permet d'identifier les relations sémantiques entre concepts et la nature de ces relations.
Topic Maps	Modèle de description de concepts ?"topics" reliés par des associations libres	Permet d'associer plusieurs termes à un concept (synonymie) et plusieurs concepts à un terme (homonymie) et de définir un contexte d'application pour chaque " topic"

Pourquoi les ontologies ?

Les ontologies jouent un rôle très important dans le Web sémantique proposé par Tim Berners-Lee en permettant de spécifier de manière formelle des vocabulaires pour la description du contenu du Web. Ainsi, les informations du Web peuvent être accessibles et compréhensibles à la fois par les humains et par les machines. Plusieurs langages ont été proposés pour représenter les ontologies avec une expressivité plus ou moins élevée et une complexité de raisonnement plus ou moins grande. Le traitement automatisé des données requiert une représentation de la sémantique compréhensible et échangeable par les machines. Cette représentation peut être faite avec différentes ressources sémantiques avec chacune un niveau d'expressivité (les ontologies, les taxonomies, les thésaurus) (Bechhofer, 2012).

Les ontologies décrivent la structure de l'information pour un domaine donné ou une activité particulière. Elles désignent l'ensemble des concepts d'un domaine ainsi que leurs relations. Elles sont employées pour raisonner à propos des objets du

domaine concerné. " L'ontologie est aux données ce que la grammaire est au langage " .

Elle est composée des concepts, des instances représentant des objets du monde réel et des relations entre les concepts. Les ontologies peuvent être des taxonomies et être lexicalisées. Une taxonomie est un vocabulaire contrôlé, organisé sous forme hiérarchique simple. Cette hiérarchisation correspond souvent à une spécialisation. Il existe donc un lien précis entre un terme du vocabulaire et ses enfants. Tandis que les thésaurus sont des listes organisées de termes contrôlés et normalisés représentant les concepts d'un domaine de la connaissance (N.F.Noy and McGuinness, 2001).

Les ontologies représentent un degré d'expressivité supérieur aux taxonomies et thésaurus. Une ontologie correspond à un langage formel c'est-à-dire une grammaire qui définit la façon dont les termes peuvent être utilisés entre eux. Les ontologies permettent de représenter des instances. Les autres ressources ne permettent pas ce genre de représentation.

A.1.2 Les langages sémantiques des ontologies

Afin de représenter des ontologies, plusieurs langages ont été définis. En premier lieu le langage RDF puis le langage RDFS et enfin le langage OWL (Herman, 2012).

RDF est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions. Il est développé par le consortium W3C. Les informations des ressources sont décrites par un triplé : Sujet représente la ressource à décrire, Prédicat représente une propriété utilisée pour caractériser et décrire une ressource et Objet représentant une donnée ou une autre ressource.

RDFS est un langage extensible. Il représente une collection de classes et les classes organisées en hiérarchie, et offrant une extensibilité grâce à la spécialisation en sous-classes. RDFS fournit des éléments de base (table A.2) pour la définition d'ontologies ou des vocabulaires destinés à structurer des ressources RDF.

TABLE A.2 – Syntax RDFS

```
<rdf :Property rdf :ID='name'>
<rdfs :domain rdf :resource='Person'/>
<rdfs :range rdf :resource='&rdfs ;Literal'/>
<rdfs :label xml :lang='fr'>nom</rdfs :label>
<rdfs :label xml :lang='afr'>nom de famille</rdfs :label>
<rdfs :label xml :lang='en'>name</rdfs :label>
</rdf :Property>
```

Cependant, RDF/RDFS a un pouvoir expressif limité. En effet, il est souvent nécessaire d'exprimer des contraintes supplémentaires sur les classes et sur les propriétés, telles que, des contraintes de disjonction entre classes, des contraintes de fonctionnalité et des contraintes de cardinalité maximale et minimale sur les propriétés. C'est pour cela qu'il est souhaitable d'utiliser d'autres langages plus expressifs pour la représentation d'ontologies, comme le langage OWL.

OWL est construit sur le modèle de données de RDF. Il offre les moyens de définir des ontologies structurées. Il facilite l'interopérabilité en fournissant plus de vocabulaire pour décrire les classes et les propriétés, comme les approches orientées objets. Par exemple : les relations entre les classes, les cardinalités, l'égalité, le typage plus riche des propriétés, les caractéristiques des propriétés, etc.

TABLE A.3 – Syntax OWL

```
<owl :Class rdf :ID="Sportive">
<owl :equivalentClass>
<owl :Restriction>
<owl :onProperty rdf :resource="#hobby" />
<owl :someValuesFrom rdf :resource="#Sport" />
</owl :Restriction>
</owl :equivalentClass>
</owl :Class>
```

Le langage OWL (table A.3) a une expressivité très élevée et une complexité de raisonnement très grande. Afin d'avoir un compromis entre l'expressivité et la complexité de ce langage, OWL fournit trois sous langages de plus en plus expressifs conçus pour l'usage des communautés spécifiques des utilisateurs et des développeurs : OWL Lite, OWL DL et OWL Full (Berti-Équille, 2012).

OWL-Lite est la version la plus simple des trois. Il permet la déclaration d'une hiérarchie de classes, d'une hiérarchie de propriétés et certaines contraintes (les cardinalités 0 ou 1 des propriétés). OWL-DL a un pouvoir d'expression supérieur à celui de OWL-Lite, avec une capacité de raisonnement complète (toutes les conclusions sont calculables) et décidable (les conclusions sont toujours calculables en un temps fini). Enfin, OWL-Full est le sous-langage qui a l'expressivité maximale mais, en revanche, ne garantit pas la décidabilité du raisonnement.

En résumé :

— RDFS permet de définir la structure des données, pas OWL. OWL décrit les relations sémantiques.

— OWL ajoute de la sémantique au schéma. Il permet de spécifier beaucoup plus sur les propriétés et les classes. Il est également exprimé dans des triplets.

— Une autre chose utile OWL ajoute la capacité de dire deux choses sont les mêmes, ce qui est très utile pour joindre des données exprimées dans des schémas différents.

RDF définit la manière d'écrire des objets et OWL définit la façon de quoi écrire.

Pour sa meilleure expressivité, le langage OWL sera utilisé dans notre approche sémantique.

Rappelons que notre objectif en utilisant des ontologies exprimées avec le langage OWL est de suivre et guider les utilisateurs dans leur démarche qualité en ajoutant la sémantique aux métadonnées et aux données.

A.2 Calcul de la sous-catégorie dominante

Pour chaque colonne, on calcule le nombre de valeurs vérifiant une sous-catégorie (la langue dans notre cas). La langue dominante est celle vérifiée par le maximum de valeurs. Nous calculons alors une probabilité pour chaque sous-catégorie en utilisant le naïve bayésienne pour la classification de textes (Naive Bayes (NB) text classification [1]) (Bennani, 2006).

Dans la classification de texte, on cherche à trouver la meilleure classe c pour un document d.

La classification de textes en utilisant NB consiste dans le calcul de la probabilité conditionnelle (formule A.1) :

$$P(c|d) = \frac{P(d|c).P(c)}{p(d)} \tag{A.1}$$

1. http ://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html#laplace

avec d (document) représente une colonne dans notre cas (d=$\{v_i\}$). c (classe) est une sous-catégorie. Les classes doivent être disjointes. Cependant, les valeurs peuvent exister dans plusieurs langues $v_i \in c_j$ and $v_i \in c'_j$.

Dans notre cas, on cherche à trouver la sous-catégorie dominante pour une colonne. La meilleure classe dans la classification NB est la plus probable ou le maximum a posteriori (MAP) c_{MAP}. Il revient à maximiser les probabilités suivantes :

$$C_{MAP} = \underset{c \in C}{argmax} P(d|c) \cdot P(c)$$

P(c) est la probabilité d'une sous-catégorie par rapport au nombre total de sous-catégories.

$$C_{MAP} = \underset{c \in C}{argmax} P(\{v_1, v_2, ...v_n\}|c) \cdot P(c)$$

or

$$P(\{v_1, v_2, ...v_n\}|c) = p(v_1|c) \cdot p(v_2|c) \cdot p(v_3|c) \cdot ... \cdot p(v_n|c)$$

donc

$$C_{NB} = \underset{c \in C}{argmax} P(c_j) \prod_{v \in V} P(v|c)$$

avec V est l'ensemble de valeurs différentes (vocabulaires) existant dans les différentes catégories, n le nombre de valeurs par colonne (document) et N le nombre de sous-catégories.

Exemple 9.1 : Choix de la sous-catégorie dominante

Soit deux langues *French* et *English* (table A.4) :

TABLE A.4 – Ensemble de sous-catégories

French	**English**
Paris	Paris
Tunis	Tunis
Londres	London
Pékin	Beijing

La colonne dont on cherche la langue dominante (table A.5) :

TABLE A.5 – Sous-catégorie inconnue

Colonne X
Londres
Paris
Tunis
Pékin

$$P(c_j) = \frac{catcount(C = c_j)}{N}$$

Alors pour calculer la probabilité de c_j = French on aura :

$$P(c_j = French) = \frac{1}{N} = \frac{1}{2}$$

de même pour

$$P(c_j = English) = \frac{1}{N} = \frac{1}{2}$$

Pour calculer la probabilité d'appartenance d'une valeur v_i à une sous-catégorie c_j, on utilise la probabilité de maximum de vraisemblance :

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

pour éviter les zéros, on peut utiliser soit la méthode *add-one* ou *Laplace smoothing*[2] est appliqué en ajoutant un 1 (ou bien un paramètre α) aux deux nombres.

$$\hat{P}(w_i|c_j) = \frac{n_k + \alpha}{n + \alpha|V|}$$

avec n_k est le nombre d'occurrence d'une valeur dans une langue.

On calcule le nombre des valeurs $v_i \in c_j$, diviser par le nombre de valeurs dans chaque sous-catégorie c_j avec α multiplié par la cardinalité du vocabulaire.

|V| = 6 (nombre de valeurs).

2. http ://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html#laplace

TABLE A.6 – Calcul de probabilité pour la colonne X

Column XProbabilité	**French**	**English**
Londres	$\frac{1+\alpha}{4+6\alpha}$	$\frac{\alpha}{4+6\alpha}$
Paris	$\frac{1+\alpha}{4+6\alpha}$	$\frac{1+\alpha}{4+6\alpha}$
Tunis	$\frac{1+\alpha}{4+6\alpha}$	$\frac{1+\alpha}{4+6\alpha}$
Pékin	$\frac{1+\alpha}{4+6\alpha}$	$\frac{\alpha}{4+6\alpha}$

Pour $\alpha = 1$, on aura :

$$\begin{aligned} P(d|French) = & \frac{1}{2}.\frac{(1+\alpha)^4}{(4+6\alpha)^4} = 8.10^{-4} \\ P(d|English) = & \frac{1}{2}.\frac{\alpha^2(1+\alpha)^2}{(4+6\alpha)^4} = 2.10^{-4} \end{aligned} \tag{A.2}$$

La **langue dominante** est alors **French** pour la colonne X.

A.3 Reconnaissance sémantique du concept

A.3.1 Transformation d'un schéma en une ontologie

Plusieurs travaux abordent la transformation d'un schéma conceptuel d'une BD en une ontologie (Krivine et al., 2009). Ils prennent en entrée un diagramme de classe UML et le transforme en une ontologie. Une première approche dans (Faucher et al., 2008) permet de transformer les modèles métier exprimés en diagramme de classe UML en une ontologie se conformant à ces métas modèles de correspondances (par exemple, une classe du diagramme UML est transformée en un concept d'une ontologie). Deux possibilités sont spécifiées dans ODM pour le passage d'un modèle UML vers un modèle RDFS : soit en utilisant des profils UML, soit par l'écriture des règles de transformation en langage QVT . ODM propose trois métas modèles : RDFS, OWL et topic Maps. Parmi les moteurs de catégorisation, il existe TEMIS, CORESE (permet d'annoter directement des termes exemple : Paris sera annoté "lieu" dans locationconcept).

Un autre cas étudié dans (Mefteh et al., 2009) commence par la création de la partie déclarative : les détails de la construction de l'ontologie (utilisation de commande SQL). Ensuite, le peuplement de l'ontologie : transformation ou correspondance entre BD et ontologie en utilisant les différents outils existants. Une troisième

étape est la vérification de la consistance de la base de connaissances. Enfin, la définition des règles d'inférence. Ils utilisent trois outils : D2R Map (Bizer, 2003), R2O (qui introduit des requêtes SQL directement dans le document de correspondance) et KAON Reverse (correspondance entre colonnes des tables et propriétés des concepts). Ils proposent un transformateur Eclipse, appelé UML2OWL. Une des limites de cet outil est qu'il n'exprime pas les associations de compositions.

Une approche de reverse engineering des BDs vers des ontologies (Benslimane et al., 2006). Cette approche est basée sur l'idée que la sémantique d'une BD peut être déduite sans une analyse explicite de schéma relationnel, des tuples et des requêtes des utilisateurs, mais plutôt par l'analyse des formulaires HTML. Ces formulaires présentent l'interface de communication la plus populaire avec des BDs pour la saisie et l'affichage des données sur le Web. La sémantique est complétée par le schéma relationnel et la connaissance des utilisateurs pour construire une ontologie. Cette approche peut être appliquée à la migration de données des pages Web, qui sont généralement basées sur des BDs, vers une ontologie. L'objectif de cette approche est de rendre exploitables par les machines, les bases de données relationnelles disponibles sur le Web.

Une autre approche similaire à la précédente permet l'extraction d'une ontologie à partir d'une BD en se basant sur les spécifications des BDs XML. Le principe est d'utiliser la sémantique existante dans les balises XML. Afin de définir des règles pour la création des concepts, des relations sémantiques et un noyau d'ontologie. Une dernière étape serait d'enrichir ce noyau en exploitant le texte en langage naturel présent dans le document (Kamel and N.Aussenac-Gilles, 2009). Ces différentes approches sont utilisées dans plusieurs outils tels que DataMaster, KAON2 , RDBToOnto (Cerbah, 2008), DB2OWL (Ghawi et al., 2007), R2O (Barrasa et al., 2004).

— DataMaster [3] est un plug-in de Protégé spécialisé dans l'import des structures et des données de BDs relationnelles (Nyulas et al., 2007). Il propose une méthode d'import des tables de la BD en des concepts OWL. En utilisant DataMaster avec une BD, chaque table est convertie en un concept et chaque ligne de la table est convertie en une instance du concept correspondant. Les valeurs des attributs sont instanciées avec les valeurs des champs correspondants de la table. Cette première démarche considère une table comme un concept et une ligne de la table comme une instance. Cependant, les contraintes d'une BD ne sont pas traitées avec le DataMaster.

— KAON2 [4] est une plateforme de construction d'ontologies développée à l'université de Karlsruhe. Elle permet l'extraction des instances depuis une BD en

3. http ://protegewiki.stanford.edu/wiki/DataMaster
4. http ://kaon2.semanticweb.org/

utilisant les langages D2RQ [5] (transforme une BD en RDF) et R2O (Barrasa et al., 2004) (transformation d'une BD en RDF(S) ou OWL). KAON2 vise à offrir des moyens déclaratifs pour décrire des procédés d'instanciation à partir de bases relationnelles d'ontologies prédéfinies manuellement. Il permet de fournir une « vue » sous forme d'ontologie, alimentée à la volée par des instances extraites d'une base de données.

— DB2OWL permet de générer automatiquement des ontologies en OWL à partir des schémas des BDs (Ghawi et al., 2007). Il détecte les cas particuliers existants dans la BD (clé étrangère, relations) et apparie chaque élément de la BD à son élément adéquat dans l'ontologie. Elle considère qu'une table de la BD est un concept (classe OWL), une relation composée entre deux autres tables est une relation (objectproperty), une table qui contient une clé étrangère est une sous classe de la classe qu'elle référence et les colonnes sont des propriétés (datatypeproperty).

— RDBToOnto (Cerbah, 2008) permet à l'utilisateur un paramétrage du projet de conversion à savoir :
  — la définition interactive de contraintes locales, attachées aux tables d'entrée, pour orienter le processus de transformation.
  — Le choix d'un convertisseur (implémentation d'une méthode de transformation), RDBToOnto permet de réutiliser les connaissances contenues dans une base de données.

— L'outil R2O (Barrasa et al., 2004) permet la transformation d'une BD existante en une ontologie existante RDF ou OWL. Quatre manières pour faire la correspondance entre une BD et une ontologie : (i) la correspondance directe : chaque ligne d'une table représente une instance d'un concept ; (ii) jointure/union : plusieurs tables liées (par une clé étrangère) sont transformées en un concept. Chaque jointure d'enregistrement est transformée en une instance d'un concept ; (iii) la projection : un sous ensemble de colonnes nécessaire pour transformer un concept ; (iv) la sélection : un sous ensemble de lignes est transformé en un concept. On peut faire correspondre une table à un concept ou à plusieurs concepts, mais une seule instance peut être générée.

A.3.2 Principe d'alignement d'ontologies

Deux critères pour faire la correspondance entre deux ontologies (Euzenat, 2012) : (i) le contenu qui considère ce qui est à l'intérieur d'une ontologie comme le nom de l'entité, les contraintes sur les relations, les relations entre entités et (ii) le contexte

5. http ://d2rq.org/

qui est la relation des ontologies avec l'extérieur par exemple les ressources annotées, le web, les ontologies extérieures (dbpedia [6]) et les ressources externes (WordNet).

Parmi les outils existants d'alignement, il existe le plugin Prompt [7] de Protégé. Il contient trois fonctionnalités : compare, transforme et fusionne deux ontologies. Il prend les deux ontologies en question et propose des correspondances à l'utilisateur. Un autre outil LOM (Lexicon-based Ontology Mapping) [L2004] qui dépend aussi de la présence de l'utilisateur mais il réduit cette intervention en utilisant quelques outils intelligents. Il fournit différents types de rapprochement : un rapprochement basé sur la similarité entre les termes, sur un ensemble de mots, sur les mots synonymes existants dans WordNet [Web7].

Pour un rapprochement sémantique, il existe l'outil S-Match (**?**), qui à partir de deux structures de données (ontologies) renvoie les relations sémantiques existantes entre les éléments des ontologies. Exemple de relation : équivalence, plus général, moins général, inadéquation et chevauchement. On propose aussi dans (Euzenat, 2012) un calcul d'alignement en se basant sur la distance sémantique SMOA. On fournit une Api Java AlignApi et un outil en ligne « Alignment server commands ».

Nous avons eu différents résultats en testant ces outils. Pour Prompt, l'intervention de l'utilisateur est assez importante. Prompt propose des possibilités de correspondances et c'est à l'utilisateur de les valider. Du coup il requière une bonne assistance de l'utilisateur. Pour l'outil AlignApi, en calculant un alignement basé sur les noms des entités, on constate qu'il n'y a pas une distinction entre les noms de concepts et les noms des attributs. On trouve dans le résultat d'alignement qu'il essaye de faire correspondre un nom de concept avec celui d'un attribut. Néanmoins, il y a moins d'assistance utilisateur. Nous allons donc utiliser l'outil AlignApi pour l'alignement des ontologies.

6. http ://dbpedia.org/
7. http ://protege.stanford.edu/plugins/prompt/prompt.html