Chapitre 5

Application de mesures de distance pour la détection de problèmes de qualité de données

5.1. Introduction

Avec la multiplication des sources d'informations disponibles et l'accroissement des volumes et flux de données potentiellement accessibles, la qualité des données et, au sens large, la qualité des informations n'ont cessé de prendre une place de premier plan tant au niveau académique qu'au sein des entreprises.

Si l'analyse des données, l'extraction de connaissances à partir des données et la prise de décision peuvent être réalisées sur des données inexactes, incomplètes, ambiguës et de qualité médiocre, on peut alors s'interroger sur le sens à donner aux résultats de ces analyses et remettre en cause, à juste titre, la qualité des connaissances ainsi « élaborées », tout comme le bien-fondé des décisions prises.

Aujourd'hui, il n'est donc plus question de négliger les données mais, bien au contraire, d'évaluer et de contrôler leur qualité dans les systèmes d'information, les bases et les entrepôts de données.

Ainsi, ont été proposées de nombreuses mesures objectives, des méthodes et tout un outillage technique pour mener une expertise critique de la qualité des données dans ces systèmes, permettant aux utilisateurs de relativiser la confiance qu'ils pourraient accorder aux données et de leur permettre de mieux en adapter leur usage.

Chapitre rédigé par Melanie HERSCHEL et Laure BERTI-EQUILLE.

L'impact et les coûts de la non-qualité des données (tout comme sa méconnaissance) retentissent à chaque étape d'un processus de traitement des données et de nombreuses techniques peuvent être combinées pour consolider et améliorer la qualité de ces données.

L'objet de ce chapitre est de faire un tour d'horizon des méthodes et des techniques employées pour détecter deux des principaux problèmes de qualité des données que sont les doublons et les données aberrantes, en se concentrant sur les méthodes basées sur des mesures de distance. Nous passerons d'abord en revue les principales sources de problèmes de qualité des données ainsi que les solutions mises en œuvre communément dans la pratique. Ensuite, nous nous consacrerons à la définition des problèmes de détection de doublons et de détection de valeurs aberrantes et nous présenterons les mesures de distances pouvant leur être appliquées. Les approches de détection de doublons et de détection de valeurs aberrantes utilisant ces mesures sont présentées par la suite et elles seront illustrées par des exemples d'application réels.

5.2. Les problèmes de qualité des données et leurs solutions en pratique

5.2.1. *Les principales sources de problèmes*

Les causes des problèmes de qualité des données sont d'origines très diverses selon les différents stades de traitement des données considérés. Les tableaux 5.1 et 5.2 récapitulent à titre indicatif quelques-uns des principaux problèmes de qualité des données pouvant survenir à chacune des étapes de leur traitement. Ils présentent également les solutions mises en œuvre pour y remédier.

Ces solutions peuvent, pour la plupart, être classées selon les grands types d'approches complémentaires suivantes : les approches préventives, les approches diagnostiques, et les approches correctives. Par la suite, nous allons aborder des solutions de diagnostique souvent utilisées dans la pratique et qui font appel à des mesures de distance.

5.2.2. *Des solutions en pratique utilisant des mesures de distance*

Parmi les techniques de détection et de correction des problèmes de qualité des données, celles les plus communément employées dans la pratique sont : 1) la vérification d'après une vérité-terrain ou bien d'après une source de données de référence jugée plus fiable, 2) l'audit des données et, enfin 3) le nettoyage des données.

Par la suite, nous verrons que, dans la majorité des cas de diagnostique, il est nécessaire d'employer des mesures de distance pour détecter les anomalies dans les jeux de données. En particulier, il s'agit d'identifier les données qui, soit sont redondantes et dupliquées, soit s'écartent d'un modèle attendus ou des données prises en référence.

Sources de problèmes de qualité des données	**Solutions potentielles**
Etape de la création des données	
Entrée manuelle : absence de vérifications systématiques des formulaires de saisie Entrée automatique : problèmes de capture OCR, de reconnaissance de la parole Incomplétude, absence de normalisation ou inadéquation de la modélisation conceptuelle des données Entrée de doublons, erreurs de mesure Approximations Contraintes matérielles ou logicielles Corruption des données : faille de sécurité physique et logique des données	**Approche préventive** : Architecture pour la gestion de processus et *workflows* : (audits, intendance des données – « *data stewards* ») **Approche corrective** : Nettoyage de données : élimination des doublons, « *merge/purge* », appariement des noms et adresses, jointure approximative **Approche diagnostique** : détection automatique des erreurs et exceptions
Etape de la collecte ou de l'import des données	
Destruction ou mutilation d'information par des prétraitements inappropriés Perte de données : *buffer overflows*, problèmes de transmission Absence de vérification dans les procédures d'import massif Introduction d'erreurs par les programmes de conversion de données Contraintes matérielles ou logicielles	Utiliser des techniques de fouille de données pour vérifier que toutes les données ont été correctement transmises Vérifier la transmission, l'intégrité des données, leur format Suivi de données Edition de données (*data publishing*) Agrégation de données et construction de résumé de données (*data squashing)*
Etape du stockage des données	
Absence de métadonnées Absence de mise à jour et de rafraîchissement des données obsolètes ou répliquées Modèles et structures de données inappropriés, spécifications incomplètes ou évolution des besoins dans l'analyse et conception du système Modifications *ad hoc*	Métadonnées Planification et personnalisation par domaine Profilage des données, moniteur de navigation dans les données

Etape de l'intégration des données	
Problèmes d'intégration de multiples sources de données ayant des niveaux de qualité et d'agrégation divers Problèmes de synchronisation temporelle Systèmes de données non conventionnels Facteurs sociologiques conduisant à des problèmes d'interprétations et d'intégration des données Jointures *ad hoc* Appariements aléatoires Heuristiques d'appariements des données inappropriées	Exiger des estampilles temporelles précises pour assurer la cohérence logique et temporelle des jeux de données, lignage des données Outils commerciaux pour : la migration des données, le nettoyage de données, le profilage de données Outils académiques pour faire les appariements et les jointures entre jeux de données
Etape de la recherche et de l'analyse des données	
Erreur humaine Contraintes liées à la complexité de calcul Contraintes logicielles, incompatibilité Problèmes de passage à l'échelle, de performances et de confiance dans les résultats Approximations dues aux techniques de réduction des grandes dimensions Utilisation de boîtes noires pour l'analyse Attachement à une famille de modèles statistiques Expertise insuffisante d'un domaine Manque de familiarité avec les données	Planification : évaluer le problème par rapport à l'équipement et *vice versa* Fouille de données exploratoire (*Exploratory Data Mining – EDM*) Plus grande responsabilité des analystes Analyse en continu plutôt que ponctuel Echantillonnage plutôt qu'analyse complète Boucle de rétroaction

Tableau 5.1. *Les principales sources des problèmes de qualité de données et leurs solutions*

5.2.2.1. *La vérification d'après une vérité-terrain ou par comparaison avec des données de référence*

La première technique consiste à comparer les valeurs de données avec leur contrepartie dans le monde réel (vérification d'après la vérité-terrain). Cette méthode est très coûteuse en temps et en moyens et, selon les domaines d'application, elle s'avère difficilement réalisable du fait que la contrepartie réelle est parfois inaccessible ou trop complexe.

Classiquement employé dans le domaine des systèmes d'informations géographiques (SIG), la comparaison par rapport à la vérité-terrain (appelé terrain nominal) permet de réaliser des matrices de confusion entre les données de la base à inspecter et les jeux de données de contrôle [DEV 06]. Des mesures de distance sont donc calculées entre les données et leur contrepartie dans le terrain nominal. De nombreux standards (ISO 19113, ISO 19138) dans ce domaine préconisent des éléments de mesures objectives pour évaluer, en particulier, la cohérence logique, la précision thématique, temporelle, positionnelle et sémantique de la base de données géospatiales [DEV 06], en mesurant les distances entre les objets de la vérité-terrain et les données représentant ces objets.

Une seconde approche appelée consolidation, met en œuvre la comparaison de deux bases de données (ou plus). Les données pertinentes de la base à inspecter sont comparées à leur contrepartie dans l'autre base : les données identiques sont considérées correctes, celles qui ne le sont pas sont signalées pour investigation et correction éventuelle. Dans ce dernier cas, la difficulté réside dans la détermination de la valeur correcte (l'une et l'autre donnée pouvant être fausses). La méthode principalement utilisée pour la correction est le remplacement par imputation : la valeur incorrecte sera remplacée par une valeur jugée « plus correcte ». D'autres méthodes dites de fusion [BLE 08] peuvent être appliquées pour résoudre le problème des conflits de valeurs.

L'inconvénient majeur de l'approche de consolidation multi-source dans la détection des erreurs et leur correction demeure : il n'y a aucune garantie que les données identiques des différentes bases soient correctes. Les données utilisées par comparaison pour détecter les données erronées dans la base à inspecter peuvent être fausses, rendant la recherche d'erreurs difficile ; aussi, cette méthode n'empêche en rien l'introduction de nouvelles erreurs dans les données.

Dans les deux méthodes mentionnées, des mesures de distance sont typiquement utilisées lors de la comparaison des données à la vérité-terrain ou à d'autres données.

5.2.2.2. *L'audit des données*

L'audit des données met en œuvre des programmes chargés de vérifier si les valeurs des données satisfont différents types de contraintes. L'avantage de l'audit des données est sa simplicité de mise en œuvre par rapport aux méthodes précédentes de comparaison. Elle peut se concevoir en même temps que le modèle conceptuel des données et peut utiliser différents outils diagnostiques d'analyse des données. Cependant, elle ne permet pas d'améliorations de la qualité des données. L'audit et l'édition de données visent l'intégrité et la cohérence de celles-ci, c'est-à-dire la conformité à des règles et des contraintes préalablement définies, mais elles ne garantissent en rien l'exactitude des données.

Généralement, l'audit des données s'articule autour des étapes suivantes :

– la définition du périmètre de l'audit dans la base de données, selon les dimensions de qualité à considérer en priorité et pour des utilisateurs-clés identifiés ;

– l'identification des segments de données à analyser (par exemple, les données client, les grands comptes, les PME, etc.) ;

– le choix d'un ensemble représentatif de données (par exemple, par zone géographique) ;

– l'analyse du dictionnaire de données (par exemple, la sélection des attributs, des types de données, des domaines de valeurs, de leur taux de remplissage, etc.) ;

– l'énumération des contraintes : par exemple, l'unicité des clés pour les enregistrements d'une table, le respect des contraintes d'intégrité, le respect de règles syntaxiques pour les valeurs de certains attributs (tels que le numéro de sécurité sociale), le respect du zonage géographique (défini, par exemple, par une règle de cohérence entre la ville et le code postal ou la base d'un dictionnaire de données), etc. ;

– le profilage des données : par exemple, le comptage du taux d'informations non renseignées, du taux d'anomalies de zonage, du taux de données ne respectant pas chacune des contraintes définies, la détection des doublons, la vérification de la cohérence entre la civilité et le prénom, la normalisation des adresses, le taux de NPAI (c'est-à-dire *n'habite pas à l'adresse indiquée*), la vérification syntaxique des numéros de téléphone, etc. ;

– des calculs croisés : par exemple, le calcul du taux d'individus avec le même email, le même nom, la même adresse, le même téléphone, etc. ;

– l'usage de référentiels : par exemple, un dictionnaire des prénoms, la base SIRET ou du référentiel RNVP (restructuration, normalisation et validation postale).

La détection des valeurs aberrantes fait partie notamment de l'étape de profilage, utilisant des mesures de distance pour identifier des valeurs atypiques du domaine.

5.2.2.3. *Le nettoyage des données, leur normalisation et standardisation*

Le nettoyage de données fait partie des stratégies d'amélioration semi-automatique de la qualité des données et il se décompose en trois étapes principales : (1) auditer les données afin de détecter les incohérences et les anomalies, (2) choisir les transformations pour résoudre les problèmes de qualité de données, et enfin, (3) appliquer les transformations choisies au jeu de données selon des règles de priorité.

Le processus de nettoyage des données repose sur un ensemble de transformations qui visent à normaliser les formats de données et à détecter les paires d'enregistrements qui se rapportent le plus probablement au même objet (ou entité du monde réel). Cette étape d'élimination des doublons est appliquée si des données approximativement similaires et donc redondantes sont trouvées. En particulier, un appariement multitable calcule des jointures approximatives entre des données distinctes mais similaires ce qui permet leur consolidation.

Nous observons que des mesures de distance sont appliquées lors du profilage et lors de la détection de doublons. Ces opérations sont mises en œuvre par une multitude d'outils de nettoyage de données, dont le tableau 5.2 ne donne pas une liste exhaustive mais limitée aux outils en open source et aux prototypes de recherche, notamment ceux à l'origine du nettoyage des données dans les années 2000.

Nom de l'outil	Fonctionnalités
Potter's wheel [RAM 01]	S, V, N, D
Ajax [GAL 01]	S, V, N, D
Intelliclean [LOW 01]	D
Bellman [DAS 02]	D, E, P
Febrl [CHR 08] http://sourceforge.net/projects/febrl	S,V, N, D,P, A
D-Dupe [KAN 08] http://www.cs.umd.edu/projects/linqs/ddupe	V, D
XClean [WEI 07]	D, R
Talend Open Profiler and Studio http://www.talend.com	S,V, N, P, A, E
DataCleaner http://datacleaner.eobjects.org/	P, A, E
Pentaho Data Integration http://kettle.pentaho.com/	S, V, N, P, D

Tableau 5.2. *Prototypes de recherche et* open sources

Le tableau 5.2 récapitule les principales forces et fonctionnalités de ces outils qui, pour les *open sources*, sont modulaires et peuvent être étendus : la standardisation (S), la vérification de contraintes (V), le profilage (P), le nettoyage par des opérateurs de transformation (N), l'élimination des doublons (D), la détection et la résolution de conflits (R), l'enrichissement par des métadonnées (E), et l'analyse exploratoire des données (A).

5.3. Approches de détection basées sur des distances

Les méthodes présentées dans cette partie ont en commun l'utilisation d'une mesure de distance ou d'une mesure de similarité[1], quantifiant ainsi jusqu'à quel degré certaines données sont en accord ou en désaccord avec d'autres données. Nous nous intéressons particulièrement aux approches liées aux problèmes de la détection de doublons (*duplicate detection*) et à la détection de valeurs aberrantes (*outlier detection*), relevant tous deux du problème général du diagnostique.

Nous débutons notre discussion sur les méthodes basées sur des distances avec une définition des différents types de problèmes visés. Ensuite, nous décrivons les méthodes ayant été développées pour résoudre ces problèmes. Une présentation de l'outillage technique fait suite à cet aperçu et nous concluons cette partie en donnant des exemples pratiques et en discutant les limitations des approches présentées.

5.3.1. *Types de problèmes visés*

Dans cette section, nous définissons plus formellement les problèmes à résoudre.

5.3.1.1. *Les doublons*

Soient A et B deux ensembles de représentations d'objets (par exemple, tuples relationnels ou éléments XML), $dist(a,b)$ une mesure de distance et θ une valeur de seuil dans le domaine des valeurs possibles pour la distance $dist(\cdot)$. Le but de la détection de doublons est l'identification de chaque paire de représentations d'objets (a,b) dont la distance est inférieure au seuil θ prédéfini c'est-à-dire tel que $a \in A$, $b \in B$, $dist(a,b) \leq \theta$. Chaque paire (a,b) satisfaisant ce critère est un doublon, toute autre paire est qualifiée de non-doublon. Ainsi, le problème de détection de doublons se rapporte à un problème de classification tel que :

$$classification(a,b) = \begin{cases} si\ dist(a,b) \leq \theta \text{ alors } (a,b)\ sont\ des\ doublons \\ sinon (a,b)\ sont\ des\ non\text{-}doublons \end{cases} \quad [5.1]$$

La valeur du seuil θ doit être spécifiée au préalable lors de la configuration par un utilisateur. En pratique, il est très commun d'avoir plus de deux classes, en

1. Dans ce chapitre, on appelle « mesure de distance », une application d qui, à deux points associe un nombre : la distance entre les deux points, qui vérifie les trois propriétés suivantes : (*i*) pour tout x et y, $d(x,y) = d(y,x)$; (*ii*) $d(x,y) = 0$ si et seulement si Identité(x,y) ; (*iii*) pour tout x, y, z, $d(x,z) \leq d(x,y) + d(y,z)$.
On appelle « mesure de similarité », une valeur comprise entre 0 et 1 caractérisant un degré de proximité entre deux objets : plus des objets sont proches, plus leur mesure de similarité tend vers 1 ; plus des objets sont éloignés, plus leur mesure de similarité tend vers 0. La similarité entre deux objets est simplement une relation réflexive et symétrique.

distinguant les doublons certains, les doublons possibles et les non-doublons. Dans ces cas, plusieurs seuils, voire plusieurs mesures de distance, sont utilisés.

5.3.1.2. *Les valeurs atypiques, isolées ou aberrantes*

Les valeurs atypiques ou isolées (*outliers*) ont été étudiées depuis plus d'un siècle en statistiques. Aussi exceptionnelles soient-elles, certaines de ces valeurs sont légitimes et d'autres sont aberrantes et peuvent être considérées comme des anomalies ou des erreurs. Malgré toute l'artillerie des méthodes statistiques disponibles, il demeure difficile, sans connaissance additionnelle, de discerner entre ces deux cas. Une valeur aberrante est définie comme étant *une observation qui semble dévier de façon marquée par rapport à l'ensemble des autres membres de l'échantillon* [GRU 69].

Ainsi le problème de détection d'une valeur aberrante notée a, décrivant un objet o telle que $val(o)= a$, peut se rapporter au calcul d'une distance entre celle-ci et un modèle prédéfini, noté $M(o)$, auquel devrait se conformer la représentation uni ou multivariée de l'objet o, tel que :

$$a \in A,\ M(o) \in A, \text{ si } dist(a,M(o)) \geq \theta \text{ alors } a \text{ est une donnée aberrante}$$

Une multitude de techniques de détection ont été proposées pour définir le modèle de conformité et détecter la déviance des données en anomalie par rapport à ce modèle. Ces techniques sont basées respectivement sur :

– un modèle mathématique (par exemple, la régression linéaire) ;

– une comparaison des caractéristiques des distributions de données avec d'autres échantillons ;

– des méthodes géométriques de mesure de distances de la donnée aberrante au reste des données [KNO 98] ;

– la distribution (ou la densité) des données avec une notion d'exceptions locales (*local outliers*) [BRE 00]. Dans ce dernier cas, il est intéressant de remarquer que certaines méthodes permettront d'identifier des valeurs isolées ou aberrantes de façon globale, sur l'ensemble du jeu de données, alors que d'autres méthodes prendront en compte un autre niveau de granularité plus fin et pourront détecter des données isolées sur des sous-ensembles du jeu de données dans la mesure où ces sous-ensembles n'auront pas la même densité de population.

Des tests de *goodness-of-fit* tels que celui du *Chi2* permettent de vérifier l'indépendance des variables du jeu de données. Dans le cas d'attributs dépendants voire de dépendances fonctionnelles, ces tests permettront d'identifier les combinaisons de valeurs d'attributs qui enfreignent les dépendances attendues. Le test

de Kolmogorov-Smirnov permet de mesurer la distance maximum entre la distribution supposée des données et la distribution empirique calculée à partir des données. Ces tests univariés permettent de valider des techniques d'analyse et des hypothèses sur les modèles employés. D'autres tests multivariés plus complexes tels que le test de Mahalanobis permettant de comparer les distances entre moyennes multivariées, ou encore la mesure Kullback Leibler pour mesurer la distance entre des histogrammes de fréquences, peuvent également être employés. Leur objectif général est de mesurer la différence entre la distribution effective de données et une distribution attendue.

5.3.2. *Mesures de distance pour la détection de doublons*

Dans les deux types de problèmes visés, détection de doublons ou de valeurs aberrantes, la qualité de la détection dépend notamment de la mesure de distance choisie. Cette section se consacre à la présentation des mesures utilisées les plus fréquemment. Ne couvrant pas en détail toutes les mesures existantes, nous recommandons les articles [ELM 07, NAU 10] pour une référence plus complète des mesures proposées pour la détection de doublons.

Avant de passer à la présentation des différentes mesures, il est important de signaler la différence entre une mesure de distance notée *dist(·)*, telle que nous l'utilisons dans la définition de nos deux problèmes, et une mesure de similarité notée *sim(·)*. En effet, tandis que *dist(·)* prend des valeurs plus importantes pour quantifier une plus grande distance entre les éléments comparés, l'inverse est vrai pour *sim(·)*. En admettant que les mesures soient normalisées, ayant un domaine de valeurs entre 0 et 1, une distance $dist(a,b) = 1$ indique la distance maximale (a et b sont complètement différents l'un de l'autre), alors que $sim(a,b) = 1$ indique la similarité maximale (a et b sont considérés identiques). Dans le cas de mesures normalisées, nous pouvons convertir une distance en une similarité telle que :

$$\text{sim}(a,b) = 1 - \text{dist}(a,b)$$

La figure 5.1 catégorise les différentes mesures utilisées lors de la détection de doublons. La majorité de ces mesures considèrent les représentations d'objets comparées comme des chaînes de caractères. Lorsque les représentations sont structurées, nous admettons que ces chaînes de caractères ont été obtenues par concaténation des valeurs individuelles (après les avoir converties en type chaîne de caractères, si nécessaire). Ainsi typiquement, les mesures présentées comparent deux chaînes de caractères *s1* et *s2*.

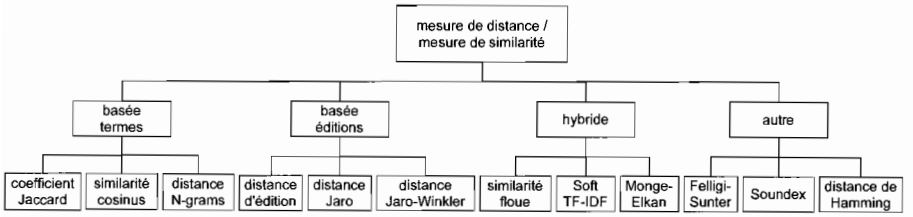

Figure 5.1. *Type de mesures de distance utilisées pour la détection de doublons*

Nous distinguons les principaux types de mesure suivants [NAU 10] :

– *mesures basées sur des termes*, ces mesures comparent deux collections de termes. Ces termes sont obtenus en divisant *s1* et *s2* en termes individuels ;

– *mesures basées édition*, ces mesures comparent *s1* et *s2* et quantifient la distance en nombre d'opérations permettant de transformer *s1* en *s2* ;

– *mesures hybrides*, ces mesures se composent de mesures de types différents, par exemple, des mesures basées sur des termes et des mesures basées édition ;

– *autres mesures*, ces mesures ne peuvent être classées dans l'une des catégories précédentes, par exemple, des distances phonétiques telles que *Soundex*, des distances numériques, dont fait partie la distance de Hamming, ou encore la mesure probabiliste de Fellegi-Sunter.

Les mesures les plus répandues en pratique sont les mesures basées sur des termes, basées édition et les mesures hybrides. Nous limitons la discussion détaillée à des représentants de ces trois types. La définition et les caractéristiques principales des autres mesures sont récapitulées dans le tableau 5.5.

5.3.2.1. *Mesures basées sur des termes*

Parmi les mesures basées sur des termes, le coefficient Jaccard et la similarité cosinus sont présentés car ils sont les plus communément utilisés.

Coefficient de Jaccard : utilisant une fonction *termes(s)* divisant une chaîne de caractères *s* en un ensemble de termes $\{t_1, t_2, ..., t_i, ..., t_n\}$, le coefficient de Jaccard de deux chaînes de caractères *s1* et *s2* est défini par l'équation [5.2] :

$$CoeffJaccard(s1, s2) = \frac{|termes(s1) \cap termes(s2)|}{|termes(s1) \cup termes(s2)|} \qquad [5.2]$$

où $|termes(s)|$ est le cardinal de l'ensemble *termes(s)*.

Similarité cosinus : la similarité cosinus de deux chaînes de caractères *s1* et *s2* est définie par l'équation [5.3] [BIL 03] :

$$SimCosinus(s1,s2) = \frac{V_{s1} \cdot V_{s2}}{\|V_{s1}\| \cdot \|V_{s2}\|} \qquad [5.3]$$

où $\|V_s\| = \sqrt{a^2 + b^2 + c^2 + \cdots}$ pour un vecteur $Vs = [a,b,c,...]$ et le vecteur Vs de dimensionnalité égale au nombre de termes en lesquels la chaîne s peut être décomposée et où le i-ième terme correspond à la i-ième dimension.

Le vecteur Vs contient soit une valeur égale à 0 en i-ième position si s ne contient pas le terme correspondant, soit une valeur correspondant à un poids $w(t_i)$ associé au terme t_i de s.

En pratique, la mesure *TF-IDF* (voir tableau 5.5) est souvent utilisée pour calculer le poids w d'un terme. Afin d'illustrer les mesures basées sur des termes introduites ci-dessus, utilisons une relation décrivant des CDs extraits de *freedb.org* (le 10/08/2011).

CDID	Titre	Artiste	Genre
9e09760c	Britney	Britney	rock
d50bf60e	ngap	britney	misc
9109b10c	Britney	Britney Spears	misc
8d12a51b	Live from Las Vegas	Britney Spears	rock
670dba08	Pink Floyd Live	Pink Floyd	rock
cd0d210f	Try This	Pink	rock
cd0d220f	Try This	Pink	rock

Tableau 5.3. *Exemple extrait de freedb.org*

Dans un premier temps, mesurons le coefficient de Jaccard des deux CDs ayant les identifiants *670dba08* et *cd0d210f*. Nous formons les collections de termes en concaténant les valeurs de l'artiste et du genre avant de diviser la chaîne de

caractères selon les espaces. Ainsi, nous obtenons pour les deux CDs les ensembles de termes suivants :

– *termes(670dba08)* = {Pink, Floyd, rock} ;

– *termes(cd0d210f)* = {Pink, rock}.

Ayant deux termes sur trois en commun, il en résulte un coefficient de Jaccard de 2/3. L'exemple précédent ne prend pas en compte les titres des CDs dans la comparaison, par exemple « Pink Floyd Live » et « Try This » pour les CDs considérés précédemment. En principe, nous obtiendrions les ensembles de termes {Pink, Floyd, Live, rock} et {Try, This, Pink, Rock} où chaque terme n'apparaît qu'une seule fois. En pratique, les termes comparés sont souvent liés à l'attribut d'origine de façon à ce que les ensembles contiennent des paires d'attributs-valeurs. Poursuivant notre exemple, nous obtenons :

– *termes(670dba08)* = {(titre, Pink), (titre, Floyd), (titre, Live), (artiste, Pink), (artiste, Floyd), (genre, rock)} ;

– *termes(cd0d210f)* = {(titre, Try), (titre, This), (artiste, Pink), (genre, rock) }.

Dans cet exemple, deux paires d'attributs-valeurs sont communes aux deux ensembles dont l'union en contient huit, le coefficient Jaccard est donc égal à 2/8.

Dans un deuxième temps, consacrons-nous au calcul de la similarité cosinus des deux CDs 8d12a51b et 670dba08. En admettant qu'uniquement les titres et les genres soient utilisés lors de la comparaison, nous obtenons :

– *termes(8d12a51b)* = {Live, from, Las, Vegas, rock};

– *termes(670dba08)* = {Pink, Floyd, Live, rock}.

Calculons à présent le poids de chaque terme, en utilisant la mesure *TF-IDF* (voir tableau 5.5). Admettons que la relation décrivant les CDs corresponde à la relation intégrale, c'est-à-dire, sept tuples au total d'après le tableau 5.3. Nous observons d'une part que chaque terme n'apparaît qu'une seule fois par tuple, d'où $tf = 1$ dans tous les cas. D'autre part, chaque terme apparaît dans un seul des sept tuples, excepté *Live* apparaissant deux fois et *rock*, qui apparaît dans cinq tuples. Ainsi, nous obtenons les scores *TF-IDF* suivants :

tfidf(Live) = log(1 + 1) * log(7/2) = 0,134

tfidf(from) = log(1 + 1) * log(7/1) = 0,253

tfidf(Las) = log(1 + 1) * log(7/1) = 0,253

tfidf(Vegas) = log(1 + 1) * log(7/1) = 0,253

tfidf(Pink) = log(1 + 1) * log(7/1) = 0,253

tfidf(Floyd) = log(1 + 1) * log(7/1) = 0,253

tfidf(rock) = log(1 + 1) * log(7/5) = 0,044

Dans le tableau 5.4, la première colonne montre les termes du domaine des valeurs dont les termes considérés sont issus. Les deux colonnes suivantes reprennent les vecteurs utilisés pour le calcul de similarité cosinus représentant les deux CDs considérés. Ignorons pour l'instant les colonnes restantes.

Terme du domaine	Vecteur de 8d12a51b	Vecteur de 670dba08	Vecteur de 9e09760c	Vecteur de 9109b10c
Britney	0	0	0,134	0,134
ngap	0	0	0	0
Live	0,134	0,134	0	0
from	0,253	0	0	0
Las	0,253	0	0	0
Vegas	0,253	0	0	0
Pink	0	0,253	0	0
Floyd	0	0,253	0	0
Try	0	0	0	0
This	0	0	0	0
misc	0	0	0	0,134
rock	0,044	0,044	0,044	0

Tableau 5.4. *Exemple des vecteurs représentant les CDs*

Utilisant les vecteurs des CDs *8d12a51b* et *670dba08*, nous obtenons la similarité cosinus suivante :

$$SimCosinus\ (s1,s2) = \frac{0{,}134^2 \cdot 0{,}044^2}{\sqrt{0{,}134^2 + 0{,}253^2 + 0{,}253^2 + 0{,}253^2 + 0{,}044^2} \cdot \sqrt{0{,}253^2 + 0{,}253^2 + 0{,}044^2}}$$
$$= 0{,}00021$$

Nous observons que cette valeur est très proche de 0, indiquant que les deux ensembles de termes sont très différents l'un de l'autre, ce qui est en effet le cas. Notons que le coefficient de Jaccard, dans cet exemple égale à $2/7 \approx 0{,}29$.

Le tableau 5.4 montre également les vecteurs des deux CDs *9e09760c* et *9109b10c*. Leur similarité cosinus est égale à :

$$SimCosinus(s1,s2) = \frac{0{,}134^2}{\sqrt{0{,}134^2 + 0{,}134^2} \cdot \sqrt{0{,}134^2 + 0{,}044^2}}$$
$$= 0{,}672$$

tandis que le coefficient de Jaccard est quant à lui égal à $1/3 \approx 0{,}33$ seulement.

Nous observons dans les deux exemples que le coefficient de Jaccard obtient un résultat similaire aux environs de 0,3, tandis que la similarité cosinus distingue nettement les ensembles très différents dans le premier exemple, et les ensembles similaires dans le second. Ceci est principalement dû au fait que chaque terme est associé à un poids reflétant la force d'identification d'un terme dans un domaine. En effet, tandis que le terme *rock*, commun à beaucoup de tuples dans la relation considérée, n'a pas beaucoup d'impact sur la similarité, des termes plus spécifiques à un CD particulier, tels que les noms d'artistes, ont plus d'influence sur la similarité totale.

Jusqu'à présent, nous avons divisé des chaînes de caractères en ensembles de termes. Il y a cependant d'autres possibilités, notamment la formation de *q-grams* qui est très populaire car elle permet de compenser de petites erreurs typographiques. Etant donnée une chaîne de caractères s, nous obtenons l'ensemble des q-grams en faisant glisser une fenêtre de taille q au-dessus de la chaîne de caractères s, de telle façon que chaque contenu de la fenêtre corresponde à un terme de q caractères. Au début et à la fin de s sont introduits q-1 caractères de complétion (notés #) pour garantir que chaque terme a q caractères.

En générant par exemple des q-grams de longueur 3, c'est-à-dire des trigrammes, nous obtenons pour les valeurs d'artistes « britney » et « Britney » :

- *termes(*britney*)* = {##b, #br, bri, rit, itn, tne, ney, ey#, y#} ;
- *termes(*Britney*)* = {##B, #Br, Bri, rit, itn, tne, ney, ey#, y#}.

Utilisant ces deux ensembles, on obtient : *CoeffJaccard(britney,Britney) = 6/12.* Sans l'utilisation de q-grams, cette similarité aurait été de 0, montrant ainsi que l'utilisation de q-grams peut compenser de petites erreurs telles que la capitalisation différente des deux valeurs comparées.

5.3.2.2. *Mesures de distance d'édition*

Consacrons-nous maintenant aux mesures basées *édition*, notamment la distance de *Levenshtein*, la similarité de *Jaro* et son extension, similarité de *Jaro-Winkler*. D'autres mesures basées édition existent, par exemple la distance de *Smith-Waterman* ou l'utilisation d'espaces affines [NAV 03].

5.3.2.2.1. Distance de Levenshtein et distance d'édition

Soient *s1* et *s2* deux chaînes de caractères à comparer, la distance de Levenshtein notée *DistLevenshtein(s1,s2)* est égale au nombre minimal d'opérations nécessaires à la transformation de *s1* en *s2*, les opérations étant l'ajout, la suppression ou le remplacement d'un caractère.

En général, une distance d'édition est définie par des opérations d'édition ayant chacune un coût associé. La distance de Levenshtein en est un cas particulier, pour lequel chaque opération a un coût égal à 1.

Afin d'obtenir un résultat compris entre 0 et 1, la distance de Levenshtein peut être divisée par la longueur maximale des deux chaînes de caractères, car, au pire, il faut remplacer tous les caractères de la chaîne la plus courte puis la compléter afin d'obtenir la chaîne la plus longue.

La distance de Levenshtein est utile lorsque les chaînes de caractères comparées sont plutôt courtes et ne se distinguent que par quelques erreurs typographiques (oubli d'un caractère, faux caractère, addition d'un caractère supplémentaire, par exemple). En revanche, elle pénalise démesurément les transpositions de caractères ou encore des erreurs en bloc affectant plusieurs caractères, par exemple l'insertion d'un suffixe. Les mesures suivantes ont comme but d'atténuer l'impact des transpositions (similarité de Jaro) et des suffixes non-communs aux deux chaînes de caractères (similarité de Jaro-Winkler).

5.3.2.2.2. Similarités de Jaro et Jaro-Winkler

La similarité de Jaro-Winkler [WIN 91] entre deux chaînes de caractères *s1* et *s2* ayant un préfix commun dénoté ρ se calcule par :

$$SimJaroWinkler(s1, s2) = SimJaro(s1, s2) + |\rho| \cdot f \cdot (1 - SimJaro(s1, s2)) \quad [5.4]$$

Dans cette équation, *f* est un facteur corrigeant la similarité calculée par la similarité de Jaro [JAR 89], notée *SimJaro(s1,s2),* en considérant le préfixe commun ρ entre *s1* et *s2*.

SimJaro(s1, s2) est définie par l'équation [5.5] :

$$SimJaro(s1, s2) = \frac{1}{3} \cdot \left(\frac{|\sigma|}{|s1|} + \frac{|\sigma|}{|s2|} + \frac{|\sigma| - 0.5t}{|\sigma|} \right) \quad [5.5]$$

où σ est l'ensemble des caractères communs entre *s1* et *s2* et *t* est le nombre de transpositions de caractères communs.

Plus formellement, un caractère *c* fait parti de σ si *c* fait partie de *s1* en position *i*, *c* fait partie de *s2* en position *j* et $|i - j| \leq [0.5 \times maximum(|s1|,|s2|)] - 1$. Une transposition existe alors lorsque, en traversant *s1* et *s2*, le *i*-ième caractère commun de *s1* est différent du *i*-ième caractère de *s2*.

L'illustration des mesures basées éditions est fondée sur les mêmes données du domaine des CDs que les exemples des mesures basées sur des termes, notamment sur les données présentées dans le tableau 5.3.

Consacrons-nous tout d'abord à la distance de Levenshtein. Afin d'illustrer toutes les opérations d'édition (ajout, suppression, et remplacement de caractères), nous admettons que nous avons deux valeurs erronées de l'artiste nommée Britney Spears, par exemple *s1* = « britney Spear » et *s2* = « Brit Spears ». Dans ce cas, la distance de Levenshtein mesure 5, car pour transformer *s1* en *s2*, nous devons remplacer *b* par *B*, supprimer trois caractères (*n, e, y*), et ajouter un *s*. La distance normalisée est de $5/maximum(13,11) \approx 0,83$, ce qui peut également être traduit en une similarité 1 – 0,38 = 0,62. La distance de Levenshtein entre « Spears » et « Spaers » mesure 2 et peut être transformée comme décrit précédemment en une similarité égale à 0,67. La transposition de caractères étant une erreur fréquente, un tel impact est souvent démesuré. La distance de Jaro égale à 0,93, dont nous illustrons le calcul ci-après, permet de réduire cet effet.

En comparant s1 = « Spears » et s2 = « Spaers », tous les caractères font partie de l'ensemble de caractères communs, car pour *S, p, r,* et *s,* les positions sont identiques tandis que pour *e* et *a*, les positions varient de 1, ce qui est inférieur à la limite $0.5 \times 6 - 1$. Ainsi, $\sigma = \{S,p,e,a,r,s\}$. En traversant *s1* et *s2*, nous observons que le premier caractère commun (*S*) est également le premier caractère commun de

s2, le second caractère commun de *s1* (*p*) est également le second caractère commun de *s2*, mais le troisième caractère commun de *s1* (*e*) n'est pas le troisième caractère en commun de *s2*. Ceci signifie une première transposition. Une seconde transposition est observée pour le quatrième caractère de *s1* (*a*), les positions restantes sont identiques. Nous avons donc au total un nombre de transpositions $t = 2$. Finalement, nous obtenons :

$$SimJaro(s1, s2) = \frac{1}{3} \cdot \left(\frac{|5|}{|5|} + \frac{|5|}{|5|} + \frac{|5| - 0.5 \cdot 2}{|5|} \right) = 0.93$$

Comparons à présent les CDs ayant les identifiants 9e09760c et 9109b10c en utilisant les noms d'artistes uniquement, c'est-à-dire s1 = « Britney » et s2 = « Britney Spears ».

Dans ce cas, la similarité de Jaro mesure $1/3 \times (7/7 + 7/14 + 1) = 0{,}83$.

La similarité de Jaro-Winkler mesure, pour un choix $f = 0.1$, SimJaroWinkler = $0.83 + 8 \times 0.1 \times (1\text{-}0.83) = 0{,}97$, dû au fait que le préfixe commun est valorisé.

5.3.2.3. *Mesures hybrides*

Les mesures hybrides réutilisent des concepts de plusieurs mesures individuelles, par exemple des mesures basées édition ou des mesures basées sur des termes. Des exemples de mesures hybrides sont la mesure de similarité floue [CHA 03], la similarité Monge-Elkan [MON 96] ou encore la mesure présentée dans [BIL 03]. Dans cette section, nous décrivons brièvement la similarité floue, les autres mesures sont définies dans le tableau 5.5.

5.3.2.3.1. Similarité floue

La similarité floue (*fuzzy match similarity*) divise tout d'abord les deux chaînes de caractères comparées en termes, nous obtenons donc, comme pour les mesures basées sur des termes, deux ensembles de termes termes(s1) et termes(s2). A chaque terme t est associé un coût w(t) égal à sa valeur IDF (voir la définition de la mesure TF-IDF dans le tableau 5.5). La similarité floue correspond alors au coût minimal de transformation de l'ensemble termes(s1) en l'ensemble termes(s2), rappelant une mesure basée édition. Les opérations d'édition sont adaptées aux ensembles de termes et le coût de chaque opération diffère (contrairement à la distance de Levenshtein). Plus précisément :

– le remplacement d'un terme *t1* de *s1* par un terme *t2* de *s2* est associé :à un coût égal à *DistLevenshtein(t1,t2)* × *w(t1)* ;

– l'ajout d'un terme *t* coûte $c_{ins} \times w(t)$ (où c_{ins} est une constante définie au préalable) ;

– l'effacement d'un terme *t* coûte *w(t)* ;

– le coût total de la transformation de *termes(s1)* en *termes(s2)* est noté *tc(s1,s2)*.

En notant *W(s)* la somme des coûts de tous les termes faisant partie de *termes(s)*, la similarité floue est définie par l'équation suivante :

$$FuzzyMatch\ Sim(s1, s2) = 1 - \min\left(\frac{tc(s1, s2)}{W(s1)}, 1\right) \quad [5.6]$$

Appliquée aux deux CDs *d50bf60e* et *9109b10c*, la similarité floue est égale à 0,32 ; ce résultat est obtenu en suivant les étapes de calcul suivantes :

– *termes(d50bf60e)* = {ngap, britney, misc} ;

– *termes(9109b10c)* = {Britney, Britney, Spears, misc}.

En utilisant *idf* pour mesurer le coût d'un terme, nous obtenons :

– *W(d50bf60e)* = *idf*(ngap) + *idf*(britney) + *idf*(misc) ;

– = log(7/1) + log(7/1) + log(7/2) = 2,23.

Le coût total pour transformer *termes(d50bf60e)* en *termes(9109b10c)* correspond à la somme des coûts des opérations suivantes : remplacement de *ngap* par *Britney*, remplacement de *britney* par *Britney* et ajout de *Spears*. Avec $c_{ins} = 1$, nous obtenons :

tc(d50bf60e, 9109b10c) = 1 × idf(ngap) + 1/7 × idf(britney) + idf(Spears)

= log(7/1) + 1/7 log(7/1) + log(7/2) = 1,51

FuzzyMatchSim(d50bf60e, 9109b10c) = 1 – min(1,51/2,23, 1) = 0,32

Calcul de similarité	Définition et principales caractéristiques
Coefficient de Jaccard	Soient deux ensembles de termes S et T : $CoeffJacca\ rd(S,T) = \lvert S \cap T \rvert / \lvert S \cup T \rvert$
Distance de Hamming	Applicable à des champs numériques fixes (n° Sécu, CP) sans prendre en compte les ajout/suppression de caractères.
Distance d'édition	Soient *s1* et *s2* deux chaînes de caractères à apparier, le calcul du coût minimal de conversion de *s1* en *s2* en cumulant le coût unitaire des opérations d'ajout (A), suppression (S) ou remplacement de caractères (R) est tel que : $Edit(s1,s2) = \min\left(\sum A(s1,s2) + S(s1,s2) + R(s1,s2)\right)$
Distance de Jaro	Soient *s1* et *s2*, deux chaînes de caractères de longueur respective *L1* et *L2*, ayant C caractères communs et T transpositions de caractères (utilisé pour les chaînes de caractères courtes) : $Jaro(s1, s2)=(C/L1 + C/L2 + (2C-T)/2C)/3$
Distance de Jaro-Winkler	Soit P la longueur du plus long préfixe commun entre *s1* et *s2* : $Jaro\text{-}Winkler(s1,s2)=Jaro(s1,s2) + max(P,4).(1-Jaro(s1,s2))/10$
Distance N-grams	Somme du nombre de caractères communs sur toutes les sous-chaînes de caractères x de longueur N présents dans les chaînes a et b : $Ngram\ (a,b) = \sqrt{\sum_{\forall x} \lvert f_a(x) - f_b(x) \rvert}$
Soundex	Première lettre du mot puis encodage des consommes sur 3 caractères tel que : B, F, P, V -> 1 ; C, G, J, K, Q, S, X, Z ->2 ; D, T -> 3 ; L -> 4 ; M,N -> 5 ;R -> 6 ; Exemple : « John » et « Jan » sont encodé J500 ; « Dupontel » est encodé D134.
Mesure *TF-IDF*	Soit un terme *s1* et un document d dans un ensemble de documents D, *tf* le nombre d'occurrences du terme *s1* dans le document d et *idf* la fraction du nombre de documents dans D sur le nombre de documents contenant s1 : $Tfidf(s1, d, D)=log(tf(s1,d) + 1)*log(idf(s1, D))$
Mesure probabiliste *IDF* de Fellegi-Sunter	Soient $P_{A\cap B}(s)$ la probabilité que la chaîne de caractère s se retrouve à la fois dans A et dans B (et soit donc identifiée comme doublon) et $P_A(s).P_B(s)$ la probabilité qu'elle ne le soit pas (avec $P_A(s) = P_B(s) = P_{A\cap B}(s)$) $Fellegi\text{-}Sunter\text{-}IDF(s) = \log(P_{A\cap B}(s) / (P_A(s).P_B(s)) = \log(1 / P_A(s))$

Calcul de similarité	Définition et principales caractéristiques
Mesure du cosinus	Soient *a* et *b* deux attributs, *Da* et *Db* les ensembles de termes de chaque attribut, et les scores *Tf-idf* du terme s respectivement dans *Da* et dans *Db* : $SimCosinus(a, b) = \sum_{t \in Da \cap Db} Tfidf(t,D_a).Tfidf(t,D_b)$
Autre distance hybride	Soient $D_a=\{a_1,a_2,...,a_k\}$ et $D_b=\{b_1,b_2,...,b_p\}$ des ensembles de termes, et *s1* et *s2* deux chaînes de caractères à comparer avec distance de similarité $Sim(a_i,b_j)$: $$Hybrid1\ (D_a, D_b) = \frac{1}{k}\sum_{i=1}^{k} \max_{j=1}^{p}\left(Sim\ (a_i, b_j)\right)$$
Similarité floue	Soient *Da* et *Db* deux ensembles de termes, le coût de transformation de *Da* en *Db* est calculé en utilisant la distance d'édition et la mesure *TF-IDF* tel que : $$Cost(D_a, D_b) = \sum_{S_i \in Da} Tfidf\ (s_i, D_a)\Big|_0^1 + Tfidf\ (s_i, D_a) + Edit\ (s_i, s_j) * Tfidf\ (s_i, D_a)$$ $$FuzzyMatch\quad Sim = 1 - \min\left(\left(Cost\ (D_a, D_b)/\sum_{S_i \in Da} Tfidf\ (s_i, D_a)\right),1\right)$$

Tableau 5.5. *Distances de similarité pour comparer les chaînes de caractères et identifier les doublons potentiels*

5.3.3. *Méthodes appliquées pour la détection de valeurs aberrantes*

Les valeurs aberrantes peuvent être classées en trois catégories : les valeurs aberrantes d'amplitude, les valeurs aberrantes spatiales et les valeurs aberrantes relationnelles :

– les valeurs aberrantes d'amplitude sont considérées comme étant trop élevées ou trop basses comparées à l'intervalle des valeurs prises par la majorité des échantillons ;

– les valeurs aberrantes spatiales sont généralement définies comme des observations qui sont extrêmes par rapport aux valeurs voisines ;

– les valeurs aberrantes relationnelles sont définies comme des observations non conformes aux relations (ou corrélations) qui existent entre les variables.

La figure 5.2 catégorise les différentes mesures utilisées lors de la détection de ces différents types de valeurs aberrantes.

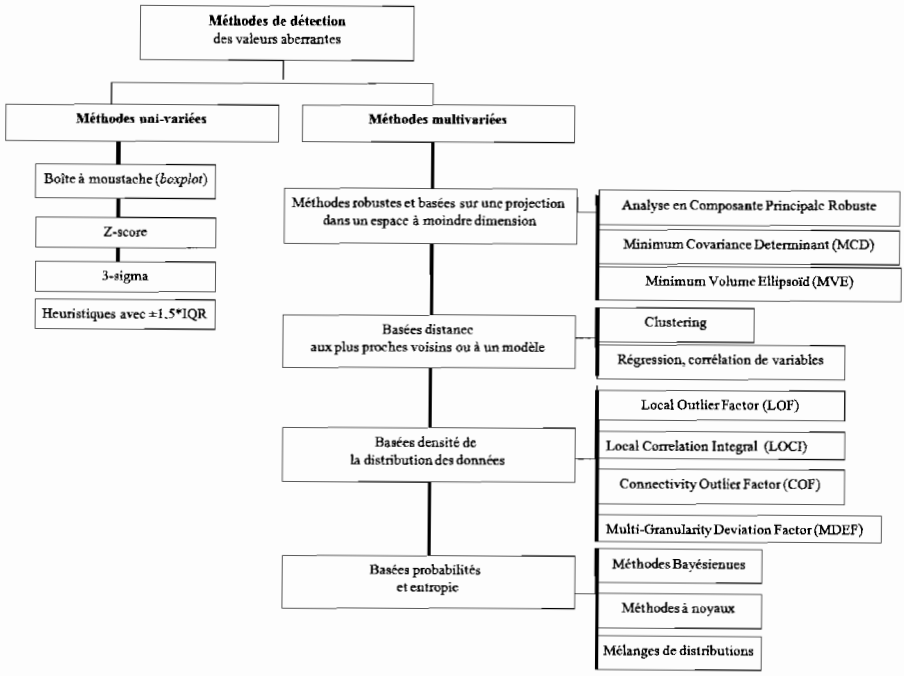

Figure 5.2. *Mesures de distance utilisées lors de la détection de valeurs aberrantes*

5.3.3.1. *Méthodes univariées*

Dans le premier cas, en mode univarié (c'est-à-dire ne considérant les valeurs de qu'un seul attribut), on pourra calculer les indicateurs représentatifs (ou valeurs typiques) telle que la moyenne, la médiane, l'écart-type ; les boîtes à moustaches (*boxplot*) permettront de mettre en évidence les points atypiques par rapport à ces indicateurs. Les extrémités des moustaches sont délimitées par 1,5 fois l'écart interquartile (Q3−Q1) calculé entre les deux quarts inférieurs et supérieurs de la distribution des données par rapport à la médiane. Cette règle permet de déceler l'existence d'un point extrême et elle est plus fiable que la fameuse règle des 3-σqui consiste à isoler les points en-deçà ou au-delà de trois fois l'écart-type autour de la moyenne. En effet, cette règle ne repose pas sur une hypothétique symétrie de la distribution des données et elle utilise des paramètres de localisation (les quartiles) qui, à la différence de la moyenne empirique, sont peu influencés par les points extrêmes. Les principaux inconvénients de ces techniques univariées comme ceux des graphes de contrôle résident d'une part, dans la difficulté à généraliser dans un cadre multidimensionnel pour considérer tous les attributs et d'autre part, à s'affranchir des hypothèses de normalité et de symétrie des jeux de données.

5.3.3.2. *Méthodes multivariées*

Dans le cas multivarié, de nombreuses méthodes non paramétriques ont été proposées. Certaines sont basées sur des projections dans des espaces à moindres dimensions [AGG 01] ou selon des composantes principales comme le montre l'algorithme dans le tableau 5.6. Dans le tableau 5.6, l'algorithme calcule la moyenne et la matrice de variance-covariance du jeu de données et sélectionne les points dont la distance de Mahalanobis à la moyenne est supérieure à la valeur critique de la loi du Chi2 pour le degré de liberté correspondant aux nombre de variables considérées et le seuil de risque choisi.

D'autres méthodes telles que celles proposées par [KNO 98, RAM 00] prennent en compte l'éloignement d'un point à l'ensemble des points de son voisinage selon sa densité ; elles offrent l'avantage de détecter les données isolées localement qui n'auraient pas pu être détectées dans la globalité du jeu de données. A titre d'exemples, [KNO 98] définissent un point O comme *DB(p,d)-outlier* si au moins une fraction p du jeu de données est située à une distance plus grande que la distance d. Selon [RAM 00], les données aberrantes sont les n premiers points dont la distance au k-ième plus proche voisin est la plus grande.

Les méthodes les plus récentes exploitent différents types de corrélations entre les variables et les corrélations spatiales [CHA 05]. Nous invitons le lecteur a consulté le tutorial de Kriegel *et al.* [KRI 10] pour une description détaillée des différentes méthodes disponibles.

Entrée : jeu de données N X D (N lignes, d colonnes)

Sortie : ensemble des valeurs aberrantes candidates noté O

- calcul de la moyenne μ et de la matrice variance-covariance Σ
- soit C, le vecteur-colonne composé de la racine carrée de la distance de Mahalanobis à la moyenne μ tel que :

$$(x-\mu)'\Sigma^{-1}(x-\mu)=(x-\mu)'\begin{bmatrix}\sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d}\\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d}\\ \vdots & \vdots & \ddots & \vdots\\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd}\end{bmatrix}^{-1}(x-\mu)$$

- sélectionner les points O de C dont la valeur est supérieure à la valeur critique de la loi du Chi 2 pour le degré de liberté d à la probabilité de 97.5 % :

$$inv\left(\sqrt{\chi_d^2(.975)}\right)$$

Tableau 5.6. *Méthode multivariée pour la détection des données aberrantes basée sur la distance de Mahalanobis*

Comme l'illustre la figure 5.3, toute la difficulté réside dans le choix des méthodes et de leur paramétrage. Les méthodes dans leur grande diversité divergeront dans leurs résultats de classification des données aberrantes : dans le cas d'une analyse univariée sur chacune des deux variables X et Y de la figure 5.3, la zone de rejet sera définie comme devant être inférieure à 2 % ou supérieure à 98 % du jeu de données en se basant sur une contrainte sur l'écart interquartile, délimitant ainsi un rectangle et excluant tous les points au dehors ; en menant une analyse multivariée combinant les deux variables et employant la distance de Mahalanobis comme l'a présenté l'algorithme du tableau 5.6, la zone des valeurs acceptables sera alors définie par une ellipsoïde. Mais, en superposant les deux analyses, nous constaterons que certains points sont considérés aberrants par l'analyse multivariée alors qu'ils ne le sont pas par l'analyse univariée. De plus, certaines méthodes reposent sur de fortes hypothèses de normalité ou de symétrie du jeu de données qui sont rarement vérifiées dans le cas de données réelles.

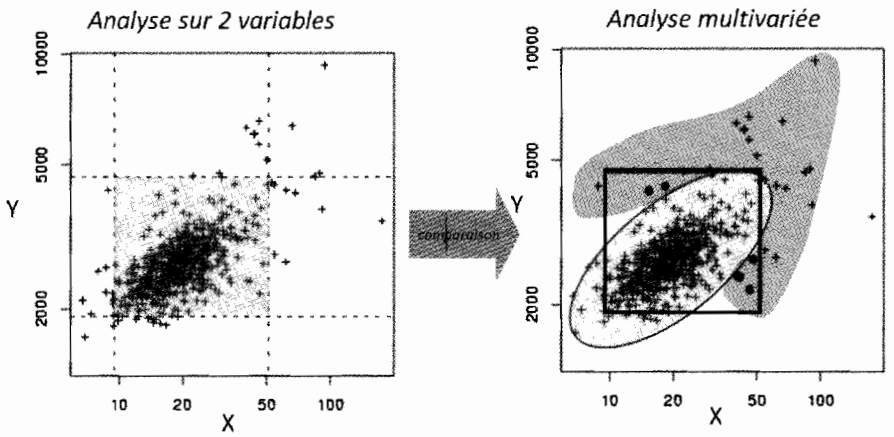

Figure 5.3. *Contradictions entre les méthodes de détection univariées et multivariées*

Les méthodes basées uniquement sur la proximité du voisinage sans considérer la densité de celui-ci n'auront pas la même capacité à détecter les valeurs aberrantes comme l'illustre l'exemple de la figure 5.4. Les points *O1* et *O2* sont à des distances respectives *d1* et *d2* de leurs plus proches voisins respectifs avec $d2 \geq d1$. Une méthode basée sur les plus proches voisins considèrera seulement *O2* comme étant une donnée isolée ou *outlier* (et non *O1)* alors qu'une méthode combinant la distance aux plus proches voisins et la densité de ceux-ci considérera à l'opposé *O1* comme un *outlier* et *O2* n'en sera pas un.

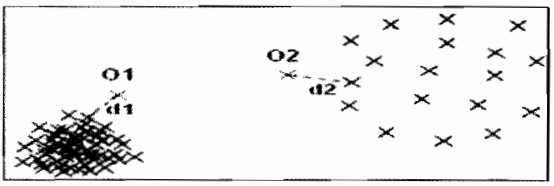

Figure 5.4. *Contradictions soulevées par les méthodes de détection*

Finalement, considérant la diversité des méthodes disponibles et leur pouvoir de détection variable, une approche récente consiste à employer un panel de méthodes à la fois univariées et multivariées, et faire corroborer leurs résultats de détection pour obtenir un consensus sur les valeurs qui apparaissent comme étant aberrantes pour la majorité des méthodes.

5.3.4. *Mise en œuvre des mesures de distance dans des méthodes de détection de doublons*

Ayant discuté les mesures de distances utilisées lors de la détection de doublons et lors la détection de valeurs aberrantes, voyons comment ces mesures sont utilisées et mises en œuvre par différentes méthodes, en particulier, dans le cas de la détection des doublons. Ne pouvant dans ce chapitre détailler toutes les méthodes ayant été proposées, nous nous concentrons sur les méthodes les plus utilisées en pratique pour la détection des doublons (notamment dû à leur applicabilité à de larges volumes de données) et nous invitons nos lecteurs à se reporter aux ouvrages de synthèse [ELM 07, NAU 10] pour un panorama plus complet.

Rappelons que le but de la détection de doublons est de détecter chaque paire de représentations d'objets *(a, b)*, avec $a \in A$ et $b \in B$, dont la distance entre objet est inférieure à un seuil fixé. Cela nécessite la comparaison de toutes les paires issues du produit Cartésien A × B, engendrant une complexité quadratique dans le nombre de représentations d'objets. En pratique, une telle complexité n'est pas admissible pour de larges volumes de données, d'où l'intérêt commun à toutes les méthodes de détection de doublons présentées ci-après à limiter cette complexité en réduisant le nombre de comparaisons nécessaires. Lors de cette démarche, il est cependant indispensable de ne pas réduire la qualité du résultat outre mesure. En effet, tout en omettant certaines comparaisons, il faut minimiser le risque que celles-ci soient en réalité des comparaisons identifiant des doublons, car ceux-ci resteraient alors dans les données après le nettoyage.

Dans le contexte de la réduction de comparaisons de paires de représentations d'objets, deux approches sont souvent citées : les approches basées sur la formation de partitions (*blocking*) [ANA 02, BAX 03, BIL 06] et les méthodes basées sur des fenêtres glissantes (*windowing*) [HER 95, MON 96].

Les méthodes dites *blocking* forment des partitions des ensembles A et B. Le critère de partitionnement appliqué aux ensembles A et B peut être spécifique au domaine considéré (par exemple, des représentations d'objets ayant le même code postal font partie de la même partition) ou basé sur des critères indépendants du domaine. Suite à cette division de A et B en blocs idéalement plus petits que les ensembles mêmes, les représentations d'objets issues des blocs correspondants sur A et B sont comparées (par exemple, les blocs ayant le même code postal).

Plus précisément, étant données l'ensemble des partitions de A noté $\{P_1^A, P_2^A, ..., P_k^A\}$ et l'ensemble des partitions de B noté $\{P_1^B, P_2^B, ...P_k^B\}$, uniquement les paires faisant partie de $(P_1^A \times P_1^B) \cup (P_2^A \times P_2^B) \cup ... \cup (P_k^A \times P_k^B)$ sont comparées.

Le choix du critère de partitionnement est essentiel à l'obtention d'une réduction du nombre de comparaisons satisfaisante, tout en maintenant une haute qualité du résultat. D'une part, il est préférable de choisir un critère qui génère de petites partitions de tailles comparables afin de réduire le nombre de comparaisons. D'autre part, le même critère doit idéalement assurer que des doublons tombent dans des partitions correspondantes.

Concernant les méthodes basées sur des fenêtres glissantes, l'idée générale est de former l'union $D = A \cup B$, de trier D en utilisant une clé de tri adaptée (spécifique au domaine ou générique), de faire glisser une fenêtre d'une certaine taille (fixe ou adaptable) par dessus la séquence triée et de comparer uniquement des représentations d'objets situées dans la même fenêtre. En considérant une taille de fenêtre constante notée w telle que $2 \leq w \leq |D|$, le nombre de comparaisons est réduit de $O(|D|^2)$ à $O(w|N|)$ avec N le nombre d'enregistrements Bien sûr, il faut également prendre en compte la complexité des étapes précédant l'étape de comparaison, dont l'étape de tri qui est la plus complexe en $O(|D| \ log \ |D|)$. Dès lors que w est choisi suffisamment petit, ce qui est le cas en pratique (souvent, des valeurs entre 5 et 30 sont suffisantes), la complexité des méthodes à fenêtres glissantes est en $O(|D| \log |D|)$.

En choisissant une petite valeur de w, le nombre de comparaisons est réduit de manière significative. Mais, afin de détecter les doublons, il faut s'assurer que ceux-ci se trouvent dans la même fenêtre. Le choix de la clé de tri joue un rôle très

important dans ce contexte, car, idéalement, elle permet de trier les doublons proches les uns des autres.

Nous observons que les deux types de méthodes présentés ci-dessus ont chacun un paramètre de configuration dont dépend la qualité du résultat (le critère de partitionnement et la clé de tri, respectivement). En pratique, il est difficile, voir impossible de définir ces paramètres de manière optimale. Pour entraver cet effet, une solution est d'élargir les ensembles de représentations d'objets comparés, par exemple en comparant des représentations de partitions adjacentes ou en augmentant la taille de la fenêtre glissante. Mais cela engendre un plus grand nombre de comparaisons, ce que les méthodes proposées essaient justement de réduire. Une solution souvent choisie en pratique est le choix de plusieurs configurations alternatives (par exemple, une configuration qui délimite les partitions en fonction du code postal, puis une autre utilisant la première lettre du nom de la ville). Ces différentes configurations sont appliquées une à une et les résultats respectifs sont unifiés en formant l'enveloppe transitive de toutes les paires de doublons détectées.

Nous illustrons les deux approches en utilisant des données décrivant des personnes et leurs lieux de résidence, représentées dans le tableau 5.7. Comme précédemment, nous simplifions l'exemple en choisissant $A = B$, nous désirons donc identifier des doublons dans une seule relation. Dans ce cas, le nombre de comparaisons sans l'utilisation des approches présentées est de 21 (en utilisant une mesure de distance symétrique). Admettons que les doublons à détecter soient les paires aux identifiants (7,4), (6,3) et (1,5).

PID	Nom	Code Postal	Ville
1	Pierre	06200	Nice
2	Jean	69001	Lyon
3	Dupont	75002	Paris
4	Didier	75002	Paris
5	Pier	02600	Nice
6	Dupond	75000	Paris 2e
7	Didié	75002	F-Paris

Tableau 5.7. *Exemple d'une liste de personnes et de leurs résidences*

Dans un premier temps, formons des partitions basées sur le code postal. Le résultat est représenté dans le tableau 5.8.

PID	Nom	Code Postal	Ville
1	Pierre	*06200*	Nice
2	Jean	*69001*	Lyon
5	Pier	*02600*	Nice
6	Dupond	*75000*	Paris 2e
7	Didié	*75002*	F-Paris
3	Dupont	*75002*	Paris
4	Didier	*75002*	Paris

Tableau 5.8. *Exemple de liste partionnée selon le code postal*

Nous obtenons cinq partitions, dont une seule contient plus d'un tuple. Lors de la détection de doublons, uniquement les tuples de chaque partition sont comparés entre eux, c'est-à-dire les paires (7,3), (7,4) et (3,4). Le nombre de comparaisons est donc réduit considérablement (3 au lieu de 21). Lors des trois comparaisons exécutées, nous trouvons le doublon (7,4) en utilisant par exemple une mesure de distance d'édition telle que nous l'avons décrite précédemment. Malheureusement, les doublons (6,3) et (1,5) ne sont pas identifiés. Ceci est dû au choix du critère formant les partitions.

Nous pouvons remédier à ce problème en utilisant un second critère de partitionnement, par exemple, en divisant l'ensemble de tuples en fonction de la première lettre de la ville (voir tableau 5.9).

Dans ce cas, nous comparons (1,5), (3,4), (3,6) et (4,6) et trouvons avec ces quatre comparaisons supplémentaires les doublons manquants (mais pas celui trouvé précédemment !).

En unifiant les résultats obtenus de ces deux configurations, nous trouvons tous les doublons avec sept au lieu de 21 comparaisons.

PID	Nom	Code Postal	Ville
1	Pierre	06200	*Nice*
5	Pier	02600	*Nice*
2	Jean	69001	*Lyon*
3	Dupont	75002	*Paris*
4	Didier	75002	*Paris*
6	Dupond	75000	*Paris 2e*
7	Didié	75002	*F-Paris*

Tableau 5.9. *Autre exemple de partionnement de la liste selon la ville de résidence*

Voyons maintenant comment une méthode basée sur une fenêtre glissante se comporte dans cet exemple. Nous devons tout d'abord choisir une clé de tri, par exemple, une clé constituée des deux premiers caractères du nom et des deux premiers chiffres du code postal. Ensuite, les tuples sont triés en ordre croissant de leur clé. Le résultat du tri et les clés y aboutissant sont montrés dans le tableau 5.10.

PID	Nom	Code Postal	Ville	Clé de tri
4	Didier	75002	Paris	*Di75*
7	Didié	75002	F-Paris	*Di75*
3	Dupont	75002	Paris	*Du75*
6	Dupond	75000	Paris 2e	*Du75*
2	Jean	69001	Lyon	*Je69*
5	Pier	02600	Nice	*Pi02*
1	Pierre	06200	Nice	*Pi06*

Tableau 5.10. *Autre exemple de partionnement basé sur une clé de tri*

En choisissant une fenêtre glissante de taille constante égale à 2, nous comparons, dans cet ordre (4,7), (7,3), (3,6), (6,2), (2,5) et (5,1), identifiant les trois paires de doublons avec six comparaisons seulement.

5.3.5. *Exemple d'applications*

Après avoir limité les exemples des sections précédentes à des exemples dédiés à l'illustration des approches, nous donnons ici quelques exemples d'applications réelles. Lors d'un projet de coopération industrielle avec la compagnie SCHUFA Holding AG qui estime la solvabilité des personnes en Allemagne et dont dépend par exemple l'obtention d'un crédit, nous avons détecté des doublons dans leurs données relationnelles décrivant des personnes et des contrats (crédit, téléphone portable, compte bancaire, etc.) Pour cela, des méthodes adaptées au domaine spécifique de SCHUFA ont été développées [WEI 08]. Celles-ci sont cependant basées sur les méthodes discutées dans ce chapitre. Le processus de détection de doublons proposé compare des représentations d'objets (personnes et contrats) en utilisant un profil de comparaison, qui est en principe une séquence des composantes suivantes : filtres utilisant des méthodes *blocking*, classificateurs de doublons basés sur des règles spécifiques au domaine (par exemple, si la date de naissance et le nom de famille sont identiques, et si le prénom a une distance de Levenshtein inférieure à un seuil spécifié, alors les personnes sont des doublons), classificateurs de non-doublons basés sur des règles spécifiques au domaine et une mesure de distance hybride incluant, entre autres, le coefficient de Jaccard et la distance de Levenshtein.

Afin d'éviter l'application de ces composantes à toutes les paires, trop coûteuse étant donné le nombre de tuples à comparer (60 millions de personnes avec les contrats associés), une technique de fenêtre glissante de taille fixe a été utilisée. L'utilisation de la méthode proposée a été évaluée sur une fraction des données SCHUFA mises à notre disposition (dix millions de tuples représentant des personnes et les contrats associés à ces personnes). Le résultat ayant été satisfaisant, les méthodes ont été intégrées dans le système opérationnel de SCHUFA et elles sont utilisées depuis fin 2010.

Nous avons choisi des méthodes similaires pour d'autres applications (par exemple, dans les domaines de données décrivant des CDs ou des films). Dans tous les cas, nous avons observé que des mesures génériques telles que celles discutées ici obtiennent de bons résultats, mais l'inclusion de savoir spécifique au domaine considéré est souvent indispensable à l'obtention d'un résultat de bonne qualité. Cela s'explique d'une part par le meilleur raisonnement classifiant un doublon, mais également par le fait que ce savoir est utilisé pour définir des critères de partitionnement ou des clés de tri.

5.4. Conclusion et perspectives

Ce chapitre a passé en revue les principales méthodes et mesures de distance permettant de détecter les doublons et les données aberrantes. Elles ont été illustrées par de nombreux exemples et leur calcul a été détaillé. De part leur grande diversité et les nombreuses alternatives de paramétrage possible, il n'est pas possible de déclarer qu'une méthode ou une mesure est meilleure qu'une autre ; ce constat dépendra du jeu de données, de ses caractéristiques propres et du domaine d'application. C'est pourquoi il est important d'utiliser et de maîtriser un panel conséquent de méthodes et de mesures afin de corroborer leurs résultats intelligemment. Toutefois, une artillerie de méthodes et de mesures ne sera pas suffisante sans un expert du domaine. Une valeur isolée ou jugée aberrante par un ensemble de méthodes peut toutefois être une exception légitime. Le bénéfice du doute lui sera accordé tant que l'expert ou une vérité-terrain ne la classera pas définitivement comme un problème de qualité de données.

Pour conclure, les multiples problèmes évoqués dans ce chapitre offrent plus que jamais d'intéressantes perspectives de recherche pour les différentes communautés scientifiques travaillant autour de la qualité des données en statistiques, bases de données, ingénierie de la connaissance, gestion de processus. Pour les entreprises et industriels, détenteurs de données en masse, la qualité des données peut demeurer un épineux problème qui se pose de façon récurrente, les guidant ponctuellement vers des choix pragmatiques et souvent à court terme par manque de moyens et d'appuis hiérarchiques. Il est alors clair que pour apporter des solutions concrètes, opérationnelles sur le long terme et théoriquement fondées, des collaborations étroites entre le monde académique et les industriels sont une nécessité.

5.5. Bibliographie

[AGG 01] AGGARWAL C.C., YU P.S., « Outlier detection for high dimensional data », *ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2001)*, 2001.

[ANA 02] ANANTHAKRISHNA R., CHAUDHURI S., GANTI V., « Eliminating fuzzy duplicates in data warehouses », *International Conference on Very Large Databases*, Hong-Kong, Chine, août 2002.

[BAX 03] BAXTER R., CHRISTEN P., CHURCHES T., « A comparison of fast blocking methods for record linkage », *International Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, Washington DC, 2003.

[BIL 03] BILENKO M., MOONEY R.J., COHEN W.W., RAVIKUMAR P.D., FIENBERG S.E., « Adaptive name matching in information integration », *IEEE Intelligent Systems*, vol. 18, n° 5, p. 16-23, 2003.

[BIL 06] BILENKO M., KAMATH B., MOONEY R.J., « Adaptive blocking : Learning to scale up record linkage », *IEEE International Conference on Data Mining*, Las Vegas, Nevada, Etats-Unis, juin 2006.

[BLE 08] BLEIHOLDER J., NAUMANN F., « Data fusion », *ACM Computing Surveys*, vol. 41, n° 1, p. 1:1 – 1:41, 2008.

[BRE 00] BREUNIG M., KRIEGEL H., NG R., SANDER J., « LOF : Identifying density-based local outliers », *International Conference ACM SIGMOD*, p. 93-104, 2000.

[CHA 03] CHAUDHURI S., GANJAM K., GANTI V., MOTWANI R., « Robust and efficient fuzzy match for online data cleaning », *ACM International Conference on the Management of Data*, 2003.

[CHA 05] CHAWLA S., SUN P., « SLOM : a new measure for local spatial outliers », *Knowledge and Information Systems,* 2005.

[CHR 08] CHRISTEN P., « Febrl – An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User Interface », *ACM SIGKDD 2008 Conference*, Las Vegas, août 2008.

[DAS 02] DASU T., JOHNSON T., MUTHUKRISHNAN S., SHKAPENYUK V., « Mining database structure or, How to build a data quality browser », *ACM SIGMOD Conference,* 2002.

[DEV 06] DEVILLERS R., JEANSOULIN R., *Fundamentals of Spatial Data Quality*, Wiley, New York, 2006.

[ELM 07] ELMAGARMID A.K., IPEIROTIS P.G. VERYKIOS V.S., « Duplicate record detection : A survey », *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, n° 1, p. 1-16, 2007.

[GAL 01] GALHARDAS H., FLORESCU D., SHASHA D., SIMON E., SAITA C., « Declarative data cleaning : Language, model, and algorithms », *International Conference on Very Large Databases (VLDB)*, p. 371-380, 2001.

[GRU 69] GRUBBS F.E., « Procedures for detecting outlying observations in samples », *Technometrics* 11, p. 1-21, 1969.

[HER 95] HERNÀNDEZ M.A., STOLFO S.J., « The merge/purge problem for large databases », *International Conference on the Management of Data*, 1995.

[JAR 89] JARO M.A., « Advances in record linking methodology as applied to matching the 1985 census of Tampa Florida », *Journal of the Americal Statistics Association*, vol. 84, n° 406, p. 414-420, 1989.

[KAN 08] KANG H., GETOOR L., SHNEIDERMAN B., BILGIC M., LICAMELE L., « Interactive Entity Resolution in Relational Data : A Visual Analytic Tool and Its Evaluation », *IEEE Transactions on Visualization and Computer Graphics,* vol. 14, n° 5, p. 999-1014, 2008.

[KNO 98] KNORR E., NG R., « Algorithms for mining distance-based outliers in large datasets », *International Conference on Very Large Databases (VLDB)*, p. 392-403, 1998.

[KRI 10] KRIEGEL H.-P., KRÖGER P., ZIMEK A., « Outlier detection techniques », *16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[LOW 01] LOW W.L., LEE M.L., LING T.W., « A knowledge-based approach for duplicate elimination in data cleaning », *Information System*, vol. 26, n° 8, 2001.

[LUC 03] LU C.-T., CHEN D., KOU Y., « Algorithms for Spatial Outlier Detection », *IEEE International Conference on Data Mining*, 2003.

[MON 97] MONGE A.E., ELKAN C.P., « An efficient domain-independent algorithm for detecting approximately duplicate database records », *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Tuscon, Arizona, Etats-Unis, mai 1997.

[NAU 10] NAUMANN F., HERSCHEL M., *An introduction to duplicate detection*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2010.

[NAV 03] NAVARRO G., « A guided tour to approximate string matching », *ACM Computing Surveys,* vol. 33, n° 1, p. 31-88, 2001.

[PAP 03] PAPADIMITRIOU S., KITAGAWA H., GIBBONS P.B., FALOUTSOS C., « LOCI : Fast outlier detection using the local correlation integral », *Proc. 19th IEEE Int. Conf. on Data Engineering (ICDE '03)*, 2003.

[RAM 00] RAMASWAMY S., RASTOGI R., SHIM K., « Efficient Algorithms for Mining Outliers from Large Data Sets », *International Conference ACM SIGMOD*, Dallas, Texas, 2000.

[RAM 01] RAMAN V., HELLERSTEIN J.M., « Potter's Wheel : an Interactive data cleaning system », *International Conference on Very Large Databases (VLDB)*, 2001.

[WEI 07] WEIS M., MANOLESCU I., « Declarative XML Data Cleaning with XClean », *Proceedings of CAiSE 2007*, p. 96-110, 2007.

[WEI 08] WEIS M., NAUMANN F., JEHLE U., LUFTER J., SCHUSTER H., « Industry-scale duplicate detection », *Proceedings of the VLDB*, vol. 1, n° 2, p. 1253-1264, 2008.

[WIN 91] WINKLER W.E., THIBOUDEAU Y., An application of the Felligi Sunter Model of record linkage to the 1990 US Decennial Census, US Bureau of the Census, 1991.

[ZHA 09] ZHANG K., HUTTER M., JIN H., « A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data », *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2009)*, 2009.

Berti-Equille Laure (ed.). (2012)

Application de mesures de distance pour la détection de problèmes de qualité de données

In : Herschel M., Berti-Equille Laure. *La qualité et la gouvernance des données au service de la performance des entreprises*

Cachan : Lavoisier-Hermes Science, p. 145-177. (Informatique et SI)

ISBN 978-2-7462-2510-7

ISSN 2111-0360