HEC MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Comparaison de fonctions de similarité entre chaînes de caractères dans une tâche de détection de doublons approximatifs

par

Jean-Philippe Raymond

Sciences de la gestion

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

Mai 2005

DÉCLARATION DE L'ÉTUDIANTE, DE L'ÉTUDIANT

ÉTHIQUE EN RECHERCHE AUPRÈS DES ÊTRES HUMAINS

Recherche sans collecte directe d'informations

Cette recherche n'impliquait pas une collecte directe d'informations auprès de personnes (exemples : entrevues, questionnaires, appels téléphoniques, groupes de discussion, tests, observations participantes, communications écrites ou électroniques, etc.).

Cette recherche n'impliquait pas une consultation de documents, de dossiers ou de banques de données existants qui ne font pas partie du domaine public et qui contiennent des informations sur des personnes.

Titre de la recherche : Comparaison de fonctions de similarité entre chaînes de caractères dans une tâche de détection de doublons approximatifs.

Nom de l'étudiante, de l'étudiant : Jean-Philippe Raymond

Signature : J Raymond

Date : 16/03/2005

Sommaire

Plusieurs fonctions de similarité entre chaînes de caractères ont été évaluées et comparées dans une tâche de détection de doublons approximatifs. Les résultats montrent que des modifications apportées à des fonctions de similarité existantes améliorent considérablement leur performance. Notamment, une variante de la méthode de niveau 2 conçue par Monge et Elkan [25] et plus tard généralisée par Cohen, Ravikumar et Fienberg [8] est proposée. Cette méthode a été appliquée sur plusieurs fonctions de similarité et les expérimentations indiquent que la nouvelle variante est généralement plus précise que la méthode originale et qu'elle permet à des fonctions de se comparer avantageusement aux autres fonctions existantes considérées dans cette étude.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Au terme de ce mémoire, je tiens à remercier tous ceux qui ont participé de près où de loin à l'accomplissement de ce travail.

Tout d'abord, je voudrais remercier mes directeurs de recherche, M. François Bellavance et M. Paul Mireault pour leur encadrement et leur patience.

J'aimerais aussi remercier du fond du coeur mes parents pour leur support inconditionnel sans lequel il m'aurait été difficile d'entreprendre mes études à la maîtrise ainsi que la réalisation de ce mémoire.

Ma reconnaissance va également à Carl Létourneau pour ses précieux conseils quant à la réalisation technique des expériences faites dans le cadre de ce projet de recherche.

Enfin, je ne saurais terminer sans remercier ma douce pour son support moral dont j'ai pu bénéficier tout au long de la réalisation de ce travail et surtout, pour sa présence inestimable.

1 Introduction

La détection de doublons approximatifs, des enregistrements qui représentent la même entité sans toutefois être textuellement identiques, est une tâche importante du nettoyage de données et de l'intégration de données provenant de différentes sources. Cette tâche permet (1) d'identifier des enregistrements redondants à l'intérieur d'une même source de données et (2) de définir des jointures entre des sources de données ne partageant pas d'identifiant commun. Dans ses deux fonctions principales, la détection de doublons approximatifs améliore la qualité de l'information contenue dans une base de données et permet d'esquisser un portrait mieux intégré des entités décrites par une base de données. Aussi, la détection de doublons approximatifs permet l'interrogation d'une base de données sans qu'il soit nécessaire que les éléments de la requête de l'utilisateur soient textuellement identiques aux informations recherchées.

Les doublons approximatifs sont causés entre autres par la présence d'erreurs typographiques, d'erreurs d'épellation ou de valeurs manquantes, ou encore par l'utilisation de normes (formats de données, abréviations, codes, etc.) qui diffèrent d'une source de données à l'autre. Par exemple, les coordonnées fictives « Sylvain Leclerc, 1234 Saint-Denis, Montréal, Québec, (514) 555-7890 » et « S. Leclair, 1324 St-Denis, Montréal, QC, 555.7890 » pourraient très bien faire référence au même individu.

La figure 1 illustre le processus généralement utilisé pour détecter des doublons approximatifs [2]. Tout d'abord, les données peuvent être transformées afin de faciliter la détection des doublons. Un exemple de transformation est de corriger automatiquement les erreurs d'orthographe qu'il pourrait y avoir dans les données (voir par exemple [30]). À cette étape, on peut aussi utiliser des routines de standardisation. Par exemple, dans le cas où des adresses civiques sont traitées, il pourrait être utile de remplacer les occurrences de « blvd » et de « boul » par

« boulevard ». Plusieurs autres façons de transformer les données contenues dans des listes de noms et d'adresses civiques ont été décrites par Winkler [45].

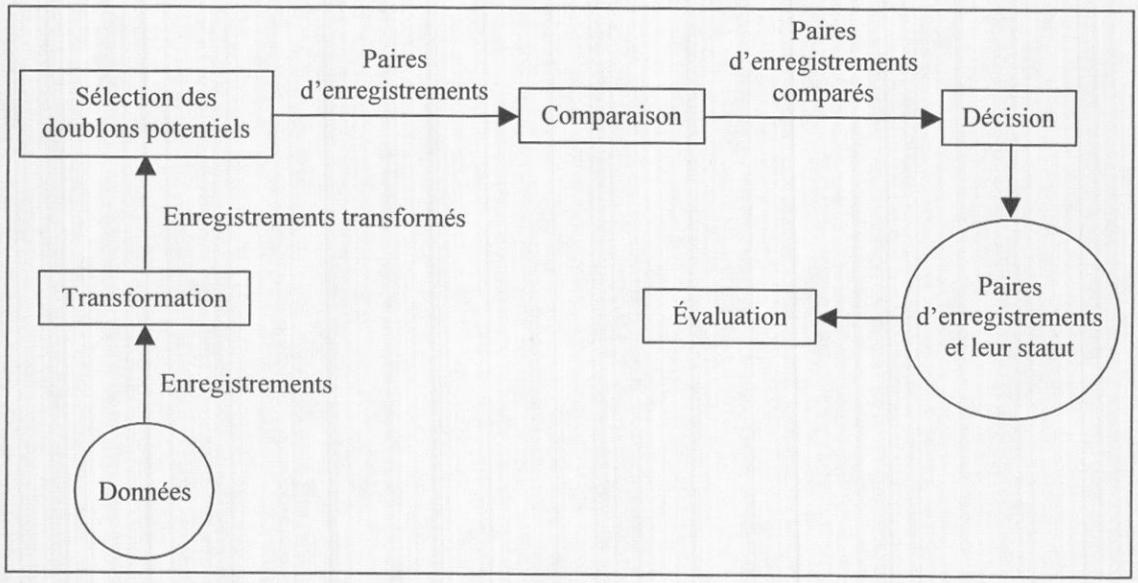


Figure 1 : Diagramme du processus de détection de doublons approximatifs

Comme chaque enregistrement est potentiellement un doublon de tous les autres enregistrements traités, ils doivent tous être comparés l'un à l'autre. Une approche largement utilisée afin de comparer deux enregistrements est d'évaluer leur similarité textuelle pour n'importe quelle combinaison de leurs composantes. Pour ce faire, plusieurs fonctions de similarité proposées dans la littérature de diverses communautés de recherche (ex. : bases de données, recherche d'information, statistique, intelligence artificielle) peuvent être utilisées. Ces fonctions comparent deux chaînes de caractères et indiquent si ces dernières doivent être considérées ou non comme similaires ou, pour la plupart, mesurent le niveau de similarité entre les deux chaînes de caractères comparées. Toutefois, il n'est pas pratique, voire même réalisable, d'évaluer de telles fonctions pour toutes les paires d'enregistrements lorsque le nombre d'enregistrements est moindrement élevé. Afin de palier à ce problème, des stratégies particulières doivent être utilisées pour réduire le nombre de doublons potentiels. Ces stratégies ont pour objectif d'éliminer à l'aide d'algorithmes rapides d'exécution les doublons potentiels qui

sont constitués d'enregistrements trop différents pour que cela vaille la peine de les examiner plus en détails, à l'aide d'une ou plusieurs fonctions de similarité par exemple.

Une fois les comparaisons effectuées pour toutes les paires d'enregistrements retenues, un modèle décisionnel doit être appliqué afin de déterminer le statut de chaque paire d'enregistrements. Une paire d'enregistrements peut être considérée comme étant constituée ou non de doublons ou, dans certains cas, peut être mise dans une troisième catégorie où l'on trouve des paires pour lesquelles il serait préférable de procéder à un examen supplémentaire (généralement l'inspection par un humain) afin de poser un verdict. Enfin, la précision de la classification des paires d'enregistrements peut être évaluée.

Le processus qui vient d'être décrit a été mis en application dans le cadre de ce mémoire afin d'évaluer et de comparer plusieurs fonctions de similarité sur des jeux de données réelles provenant de différents secteurs d'activité.

Ce mémoire est organisé de la façon suivante. Le prochain chapitre consiste en une revue de la littérature sur les différentes techniques utiles à la détection de doublons approximatifs. Le chapitre 3 présente les différentes fonctions de similarité qui sont évaluées dans ce mémoire et décrit la méthodologie utilisée pour comparer ces fonctions. Finalement, on retrouve dans le chapitre 4 une présentation et une discussion des résultats expérimentaux.

2 Revue de la littérature

La revue de la littérature est divisée en trois parties. Tout d'abord, plusieurs fonctions de similarité entre chaînes de caractères sont présentées. Ensuite, on y voit différentes approches utiles pour choisir un ensemble de règles de décision à appliquer pour déterminer si deux enregistrements représentent la même entité. Enfin, une discussion de l'apport des techniques de réduction du nombre de doublons potentiels est faite et quelques unes des ces techniques sont citées.

2.1 Fonctions de similarité

La contribution des fonctions de similarité à la détection de doublons approximatifs est considérable [31 et 46]. Winkler [46] a noté, lors de l'intégration de données dans le cadre d'un recensement d'une population de citoyens, que plus de 25% des doublons n'auraient pas été identifiés à l'aide d'une comparaison exacte (caractère par caractère) des enregistrements alors que l'utilisation d'une fonction de similarité a réduit cette proportion à 7%.

2.1.1 Méthodes basées sur l'analyse des caractères

L'une des premières méthodes proposées pour évaluer la similarité entre deux chaînes de caractères est le système Soundex de Odell et Russel [29] (voir aussi [21]). Ce système transforme toutes les chaînes de caractères en un code composé d'une lettre et de trois chiffres, pour ensuite déclarer similaires celles ayant des codes identiques. L'objectif de cette transformation est d'attribuer le même code aux noms à consonance semblable. Il est important de noter que ce système est basé sur la consonance anglo-saxonne du langage. Le système Soundex n'est pas sans failles. Il peut déclarer similaires des noms qui sont largement différents. Par exemple, les six mêmes codes sont obtenus à partir des paires de noms suivantes : Euler et Ellery, Gauss et Gosh, Hilbert et Heilbronn, Knuth et Kant, Lloyd et Ladd, ainsi que Lukasiewicz et Lissajous. Inversement, certains noms à

consonance semblable tels que Rogers et Rodgers, ou Sinclair et St-Clair, ou Tchebysheff et Chebyshev, ne sont pas considérés comme étant similaires [21].

Une approche semblable à celle du système Soundex a été utilisée par Davidson [13] afin d'identifier des erreurs d'épellations dans le système de réservation d'une compagnie aérienne. Davidson a défini des règles pour transformer le nom d'un individu en un code de quatre lettres en concaténant la première lettre du prénom avec une abréviation du nom de famille. Comme pour le système Soundex, on considère ici qu'il y a similarité entre deux noms si leurs codes sont identiques.

Glantz [14] propose de quantifier la similarité entre deux chaînes de caractères en les comparant position par position. La mesure de similarité ainsi trouvée est égale au nombre de caractères d'une chaîne qui sont identiques à ceux de l'autre chaîne divisé par la longueur de la plus longue chaîne. Une faiblesse de cette mesure est qu'elle est grandement influencée par l'omission ou l'insertion d'un caractère, plus particulièrement si cela survient au début de la chaîne de caractères.

Une catégorie importante de fonctions de similarité est constituée de fonctions basées sur la distance de Damerau-Levenshtein, en l'honneur des deux auteurs pionniers dans ce domaine. Damerau [12] a proposé un algorithme qui conclut que deux chaînes de caractères sont similaires s'il est possible d'obtenir l'une à partir de l'autre en effectuant au plus une des opérations, ou erreurs typographiques, suivantes : (1) la substitution d'un caractère par un autre, (2) la suppression d'un caractère, (3) l'insertion d'un caractère et (4) l'inversion de caractères adjacents. Dans le même ordre d'idées, Levenshtein [22] a défini la notion de distance d'édition entre deux chaînes de caractères comme étant le nombre minimal d'opérations nécessaire pour transformer une chaîne de caractères de sorte qu'elle devienne identique à l'autre.

Wagner et Fischer [42] ont résolu le problème d'optimisation posé par Levenshtein à l'aide d'un algorithme de programmation dynamique (le problème

peut aussi être résolu de façon récursive) qui trouve la séquence de coût minimal d'opérations nécessaires pour obtenir une chaîne de caractères à partir d'une autre (différents coûts peuvent être associés aux différentes opérations). Cette séquence a comme coût minimal $dist(c_m, d_n)$ qu'on calcule à l'aide de la relation récurrente

$$dist(c_i, d_j) = \min \begin{cases} dist(c_{i-1}, d_j) + sup, \\ dist(c_i, d_{j-1}) + sup, \\ dist(c_{i-1}, d_{j-1}) + sub(C_i, D_j); \end{cases}$$

où

$dist(c_i, d_j)$ est la distance d'édition entre les i premiers caractères de la chaîne c, de longueur m, et les j premiers caractères de la chaîne d, de longueur n,

sup est le coût pour supprimer un caractère (insérer un caractère est équivalent)

et $sub(C_i, D_j)$ est le coût pour substituer le i-ième caractère de c par le j-ième caractère de d (normalement 0 si les caractères sont identiques).

Enfin, afin de pouvoir remplir la matrice $dist(c, d)$, il convient de procéder aux assignations de valeurs suivantes : $dist(c_0, d_0) = 0$, $dist(c_i, d_0) = i \times sup$ et $dist(c_0, d_j) = j \times sup$.

Cet algorithme ne tient pas compte de toutes les opérations identifiées par Damerau. Il a fallu attendre un peu plus d'un an avant que Wagner, cette fois-ci en collaboration avec Lowrance [23], propose un algorithme qui puisse aussi

considérer l'inversion de caractères adjacents. Ce nouvel algorithme diffère du précédent seulement par l'ajout d'une possibilité dans la relation de récurrence

$$dist(c_i, d_j) = \min \begin{cases} dist(c_{i-1}, d_j) + sup, \\ dist(c_i, d_{j-1}) + sup, \\ dist(c_{i-1}, d_{j-1}) + sub(C_i, D_j), \\ dist(c_{i-2}, d_{j-2}) + inv(C_{i-1}, C_i, D_{j-1}, D_j); \end{cases}$$

où

$$inv(C_{i-1}, C_i, D_{j-1}, D_j) = \begin{cases} coût\ d'une\ inversion, si\ C_{i-1} = D_j\ et\ C_i = D_{j-1}, \\ \infty, sinon. \end{cases}$$

S'il n'est pas nécessaire de connaître la distance d'édition entre deux chaînes de caractères mais seulement de déterminer si cette distance est inférieure ou égale à un seuil prédéterminé, Ukkonen [40] a inclus à cet effet des conditions d'arrêt dans l'algorithme de Wagner et Fischer, réduisant ainsi son temps d'exécution. Ces modifications sont aussi applicables à l'algorithme de Lowrance et Wagner.

Par ailleurs, il est intéressant de constater que des fonctions de similarité de nature semblable à la distance d'édition ont été développées pour comparer des séquences biologiques (ex. : séquences d'ADN, séquences d'acides aminés). En effet, Sellers [37] a publié la même année que Wagner et Fischer un algorithme identique au leur pour mesurer la distance entre deux séquences biologiques. Sa découverte a été précédée des travaux importants de Needleman et Wunsch [27] qui ont introduit l'idée d'utiliser une matrice à deux dimensions (comme celle nécessaire dans les algorithmes permettant de calculer une distance d'édition) pour évaluer la similarité entre deux séquences d'acides aminés. Sans toutefois fournir un algorithme, mais plutôt une méthode générale, Needleman et Wunsch ont proposé de définir la similarité entre deux séquences comme étant le nombre maximum d'acides aminés d'une séquence qui peuvent être *pairés* (*matched* en

anglais) avec ceux de l'autre séquence tout en permettant des suppressions dans l'une ou l'autre des séquences. Deux acides aminés (ou caractères) ne peuvent être *pairés* que s'ils sont identiques. Dans le cas simple où les suppressions ne sont pas pénalisées et où on accorde un point pour chaque acide aminé d'une séquence qui est *pairé* à un acide de l'autre séquence, cela revient à calculer la longueur de la plus longue sous-séquence commune (PLSC). Selon la terminologie utilisée en biologie, une sous-séquence qui, comme cette dernière, maximise un système de pointage donné, constitue un alignement optimal pour une paire de séquences. Calculer une distance d'édition peut être vu comme un cas particulier de la méthode générale proposée par Needleman et Wunsch. On utilise pour ce cas particulier un système de pointage qui n'attribue aucun point pour les caractères *pairés* et où on pénalise chacune des opérations d'édition nécessaires pour passer d'une chaîne de caractères à une autre.

Waterman, Smith et Beyer [44] ont modifié l'algorithme de Sellers en généralisant la façon dont est pénalisée la suppression de caractères successifs. Les différentes formes de distance d'édition présentées jusqu'ici pénalisent la suppression de caractères successifs de façon linéaire (suppressions successives d'un seul caractère où chaque suppression est pénalisée de la même façon). Le nouvel algorithme proposé par les chercheurs offre la possibilité de pénaliser la suppression de k caractères successifs avec un coût $sup_k \leq k \times sup_1$. Pour évaluer le niveau de similarité $simBio(c_m, d_n)$ à l'aide de cet algorithme, il suffit d'utiliser la relation de récurrence

$$simBio(c_i, d_j) = \max \begin{cases} \max\limits_{1 \leq k \leq i} \{ simBio(c_{i-k}, d_j) - sup_k \}, \\ \max\limits_{1 \leq k \leq j} \{ simBio(c_i, d_{j-k}) - sup_k \}, \\ simBio(c_{i-1}, d_{j-1}) + sim(C_i, D_j); \end{cases}$$

où

$simBio(c_i, d_j)$ est le niveau de similarité entre les i premiers caractères de c et les j premiers caractères de d,

$sim(C_i, D_j)$ est le niveau de similarité entre le i-ème caractère de c et le j-ième caractère de d (par exemple, est égale à 1 si les caractères sont identiques, 0 sinon),

$simBio(c_i, d_0) = 0$, $simBio(c_0, d_j) = 0$ et $simBio(c_0, d_0) = 0$.

Un inconvénient de cet algorithme est qu'il a un ordre de complexité $O(m^3)$ alors que les algorithmes qui utilisent une fonction de coût linéaire pour pénaliser la suppression de caractères successifs ont un ordre de complexité $O(m^2)$. Cependant, Gotoh [15] a conçu un algorithme qui utilise une fonction de coût affine $sup_k = v + (k-1)u$ (un coût v pour débuter une séquence de suppressions successives et un coût additionnel u pour chaque caractère supplémentaire inclus dans la séquence) et qui s'exécute en mn étapes dans lesquelles est appliquée la relation de récurrence

$$simBio(c_i, d_j) = \max \begin{cases} p(c_i, d_j), \\ q(c_i, d_j), \\ simBio(c_{i-1}, d_{j-1}) + sim(C_i, D_j); \end{cases}$$

où

$$p(c_i, d_j) = \max \begin{cases} simBio(c_{i-1}, d_j) - v, \\ p(c_{i-1}, d_j) - u; \end{cases}$$

et

$$q(c_i, d_j) = \max \begin{cases} simBio(c_i, d_{j-1}) - v, \\ p(c_i, d_{j-1}) - u. \end{cases}$$

Au début de l'exécution de l'algorithme, il convient de procéder aux assignations de valeurs suivantes :

$$simBio(c_i, d_0) = p(c_i, d_0) = -sup_i$$

et

$$simBio(c_0, d_j) = p(c_0, d_j) = -sup_j .$$

Les alignements optimaux dont il a été question précédemment sont tous des alignements globaux; ils sont des alignements entre deux séquences complètes. L'algorithme de Smith-Waterman [39] fut le premier qui permet de trouver un alignement optimal local. La recherche d'un tel alignement consiste à trouver la paire de sous-chaîne (une sous-chaîne de chaque séquence comparée) pour laquelle le niveau de similarité est maximal. Une sous-chaîne est un cas particulier de la sous-séquence dans lequel on ne permet pas la suppression de caractères (ex. : « JP » est une sous-séquence, mais pas une sous-chaîne, de « Jean-Philippe »). Rechercher un alignement local pénalise dans une moindre mesure le niveau de similarité entre les chaînes de caractères qui sont différentes dans l'ensemble, mais qui partagent quand même des éléments similaires (ex. : « École des Hautes Études Commerciales, 3000 Chemin de la Côte-Sainte-Catherine » et «HEC Montréal, 3000 Chemin de la Côte-Sainte-Catherine »).

Trois changements doivent être apportés aux algorithmes vus jusqu'ici pour qu'ils recherchent le meilleur alignement local plutôt que global. Premièrement, on doit ajouter une possibilité dans la relation de récurrence, qui permet à chaque cellule de la matrice $simBio(c, d)$ de prendre la valeur 0 si toutes les autres options ont

des valeurs négatives. Deuxièmement, un alignement local peut se terminer à n'importe quelle cellule de la matrice $simBio(c, d)$, alors que pour un alignement global, la mesure de similarité entre deux chaînes de caractères se trouve dans la cellule $simBio(c_m, d_n)$. Ainsi, on doit, pour trouver la mesure de similarité d'un alignement local optimal, chercher la valeur la plus élevée de toute la matrice. Par exemple, la mesure de similarité calculée par l'algorithme de Smith-Waterman, est $\max_{1 \le i \le m \text{ et } 1 \le j \le n} simBio(c_i, d_j)$ où

$$simBio(c_i, d_j) = \max \begin{cases} \max_{1 \le k \le i} (simBio(c_{i-k}, d_j) - \sup_k), \\ \max_{1 \le k \le j} (simBio(c_i, d_{j-k}) - \sup_k), \\ simBio(c_{i-1}, d_{j-1}) + sim(C_i, D_j), \\ 0. \end{cases}$$

Monge et Elkan [25] ont proposé l'utilisation d'une variante de la mesure de similarité de Smith-Waterman. Cette variante utilise une fonction de coût affine pour pénaliser la suppression de caractères successifs et considère la substitution de caractères approximativement identiques. Deux caractères sont dits approximativement identiques s'ils font tous deux partie d'un des groupes suivants : $\{'d';'t'\}$, $\{'g';'j'\}$, $\{'l';'r'\}$, $\{'m';'n'\}$, $\{'b';'p';'v'\}$, $\{'a';'e';'i';'o';'u'\}$ et $\{',';'.'\}$.

Enfin, d'autres fonctions de similarité basées sur l'analyse des caractères ont été développées au bureau du recensement des États-Unis. Jaro [20] (voir aussi [45]) a introduit une mesure de similarité qui porte désormais son nom et qui se définit comme suit :

$$Jaro(c, d) = pc \frac{com}{m} + pd \frac{com}{n} + ptr \frac{com - tr}{com}$$

où

pc est le poids associé aux caractères présents dans *c* mais pas dans *d*,

pd est le poids associé aux caractères présents dans *d* mais pas dans *c*,

ptr est le poids associé aux caractères transposés,

tr est le nombre de transpositions de caractères

et *com* est le nombre de caractères communs à *c* et à *d*.

Deux caractères sont considérés comme étant en commun s'ils sont identiques et si leur position dans leurs chaînes respectives différent au maximum de $\max(m,n)/2-1$ positions. Le nombre de transpositions est égal à la moitié (arrondie à l'entier inférieur) du nombre de caractères communs qui ne sont pas dans le même ordre dans les deux chaînes. Par exemple, si on compare les chaînes « barnes » et « anderson », on a « arnes » comme caractères communs et ont doit considérer une transposition puisque 'r', 'n' et 'e' ne sont pas placés dans le même ordre dans les deux chaînes.

Dans la littérature, on utilise généralement une version normalisée (qui retourne une valeur comprise entre 0 et 1, où 1 représente un niveau de similarité maximal) de la fonction de similarité de Jaro ($pc + pd + ptr = 1$) avec des poids égaux :

$$Jaro(c,d) = \frac{1}{3}\left(\frac{com}{m} + \frac{com}{n} + \frac{com - tr}{com}\right).$$

À partir de l'observation faite par Pollock et Zamora [30] à l'effet que les erreurs d'orthographe surviennent plus fréquemment vers la fin d'un mot, Winkler [46] a proposé une méthode qui peut être appliquée à n'importe quelle fonction de similarité normalisée $sim(c,d)$ afin que les chaînes de caractères qui ont des préfixes identiques aient un niveau de similarité plus élevé. Spécifiquement, pour $i = 1,2,3,4$,

$$Winkler(c,d) = sim(c,d) + i \times 0{,}1 \times (1 - sim(c,d))$$

si les i premiers caractères de c et de d sont identiques. Il est important de noter que la nouvelle fonction de similarité ainsi formée est aussi une fonction normalisée. Winkler, tout comme plusieurs chercheurs et praticiens dans le domaine de la détection de doublons approximatifs, semble n'avoir appliqué cette méthode que sur la mesure de similarité de Jaro et cette combinaison est généralement reconnue comme la mesure de Jaro-Winkler.

Récemment, Yancey [47] a utilisé une variante de la distance d'édition à trois opérations (il n'a pas permis l'inversion de caractères adjacents), chacune de coût unitaire. Plus précisément, la fonction de similarité qu'il a utilisée consiste en une moyenne de la distance d'édition et de la longueur de la PLSC normalisées. Puisque la plus grande distance d'édition qu'il est possible d'obtenir à partir de n'importe quelle paires de chaînes de caractères est égale à la longueur de la plus longue des deux chaînes (si on utilise des coûts d'opération d'édition unitaires), on obtient une mesure de similarité normalisée x à partir de la distance d'édition à l'aide de la formule

$$x = 1 - \frac{dist(c_m, d_n)}{\max(m,n)}.$$

En ce qui concerne la longueur de la PLSC, Yancey a décidé de la normaliser en la divisant par la longueur de la plus courte des deux chaînes de caractères comparées.

2.1.2 Méthodes basées sur l'analyse des lexèmes ou des *n*-grams

Une chaîne de caractères peut aussi être vue comme un ensemble de lexèmes (*tokens* en anglais). Un lexème est une suite de caractères compris entre deux délimiteurs successifs. Par exemple, si on considère les parenthèses et le tiret

comme délimiteurs, le numéro de téléphone « (999)555-1212 » est formé de trois lexèmes : « 999 », « 555 » et « 1212 ».

Formellement, une chaîne de caractères peut être représentée sous la forme d'un vecteur de lexèmes $c = \{l_1, p_{c1}; l_2, p_{c2}; ...; l_l, p_{cl}\}$ où p_{ci} est le poids du lexème l_i dans c. Si on utilise des poids binaires (1 si l_i est présent au moins une fois dans c, 0 sinon), on peut évaluer la similarité entre deux chaînes de caractères à l'aide du coefficient de Jaccard [19], le ratio de la taille de l'intersection de deux ensembles sur la taille de leur union, grâce à la formule

$$Jaccard(c,d) = \frac{\sum_{i=1}^{l}(p_{ci} \times p_{di})}{\sum_{i=1}^{l} p_{ci} + \sum_{i=1}^{l} p_{di} - \sum_{i=1}^{l}(p_{ci} \times p_{di})}.$$

Une autre façon d'évaluer la similarité entre deux chaînes de caractères consiste à déterminer le cosinus de l'angle entre leurs vecteurs de lexèmes [33]. Cette mesure largement utilisée dans le domaine de la recherche d'information pour évaluer la similarité entre deux documents, ou encore la similarité entre une requête et un document, se calcule de la façon suivante :

$$simCosinus(c,d) = \frac{\sum_{i=1}^{l}(p_{ci} \times p_{di})}{\sqrt{\sum_{i=1}^{l} p_{ci}^{\,2}} \times \sqrt{\sum_{i=1}^{l} p_{di}^{\,2}}}.$$

Cette mesure est normalisée entre 0 et 1, où la valeur 1 signifie un maximum de similarité entre les deux chaînes de caractères comparées (angle nul entre les deux vecteurs de lexèmes).

Diverses méthodes ont été proposées pour déterminer les poids à accorder à chacun des lexèmes présents dans un document (ou chaîne de caractères). Salton

et Buckley [34] en ont d'ailleurs comparées plusieurs pour des applications en recherche d'information. Une méthode bien connue sous l'acronyme TFIDF (*term frequency* × *inverse document frequency*) [35] consiste à utiliser le produit de TF_{ci}, la fréquence du lexème l_i dans c, et de IDF_i, l'inverse de la fraction des chaînes de caractères du corpus (par exemple, l'ensemble des chaînes de caractères, ou enregistrements, pour lesquels on désire détecter les doublons) qui contiennent au moins une fois le lexème l_i. L'idée générale derrière la méthode TFIDF est d'accorder, dans l'évaluation de la similarité entre deux chaînes de caractères, une plus grande importance aux lexèmes peu communs. Cohen, Ravikumar et Fienberg [8], dans une tâche de détection de doublons approximatifs, ont obtenu des résultats intéressants à partir de la fonction *simCosinus* appliquée sur des vecteurs de lexèmes avec des poids $p_{ci} = \log(TF_{ci} + 1) \times \log(IDF_i)$.

Les fonctions de similarité vues dans cette section peuvent aussi être évaluées à partir de vecteurs de *n*-grams. La décomposition en *n*-grams d'une chaîne de caractères considère toutes les combinaisons de *n* caractères successifs qu'on y trouve. Par exemple, la décomposition en 5-grams de la chaîne de caractères « (999)555-1212 » donne l'ensemble suivant : « (999) », « 999)5 », « 99)55 », …, « -1212 ».

Hylton [18] s'est servi de la décomposition en 3-grams pour évaluer le niveau de similarité entre deux chaînes de caractères. Spécifiquement, il a utilisé comme mesure de dissimilarité la distance euclidienne entre les vecteurs de 3-grams des deux chaînes de caractères comparées;

$$Hylton(c, d) = \sqrt{\sum_{i=1}^{I} (p_{ci} - p_{di})^2}$$

où p_{ci} est la fréquence du 3-grams l_i dans c.

Les fonctions de similarité basées sur l'analyse des lexèmes (ou *n*-grams) sont généralement plus rapides à évaluer que les fonctions basées sur l'analyse de caractères puisqu'elles nécessitent moins de comparaisons; pour une paire de chaînes de caractères donnée, on retrouve normalement moins de lexèmes que de caractères. De plus, une fois les vecteurs de lexèmes construits, les mesures de similarité présentées dans cette section ont un ordre de complexité linéaire alors que la plupart des mesures présentées dans la section précédente ont un ordre de complexité quadratique.

2.1.3 Méthodes hybrides

Enfin, il existe des méthodes hybrides qui, dans un premier temps, décomposent une paire de chaînes de caractères en lexèmes et, dans un deuxième temps, comparent les lexèmes à l'aide d'une fonction de similarité basée sur l'analyse des caractères. Monge et Elkan [25] ont proposé l'algorithme récursif suivant pour comparer deux chaînes de caractères. On décompose d'abord c et d en lexèmes $c = \{a_1, ..., a_o\}$ et $d = \{b_1, ..., b_p\}$, et on définit $niveau2(c, d)$, le niveau de similarité entre les deux chaînes de caractères, de sorte que

$$niveau2(c, d) = \frac{1}{o} \sum_{i=1}^{o} \max_{j=1}^{p} sim'(a_i, b_j)$$

où sim' est égale à 1 si a_i est identique à b_j ou si l'un est une abréviation de l'autre. Autrement, sim' est égale à 0. On considère ici quatre formes d'abréviations :

(1) l'abréviation est un préfixe de son expansion (ex. : « Univ » est une abréviation de « Université »), ou

(2) l'abréviation combine un préfixe et un suffixe de son expansion (ex. : « Dépt » est une abréviation de « Département »), ou

(3) l'abréviation est un acronyme de son expansion (ex. : « HEC » est une abréviation de « Hautes Études Commerciales »), ou

(4) l'abréviation est une concaténation de préfixes de son expansion (ex. : « Caltech » est une abréviation de « California Institute of Technology »).

On note que (3) est un cas particulier de (4). De plus, il est important de souligner que, contrairement aux autres fonctions de similarité présentées jusqu'ici, cette fonction n'est pas symétrique.

Monge et Elkan ont comparé cette méthode asymétrique avec leur variante de la fonction de similarité de Smith-Waterman et ont obtenu de meilleurs résultats avec la seconde approche.

Cohen, Ravikumar et Fienberg [8] ont plus tard généralisé l'algorithme récursif de Monge et Elkan en y substituant la fonction *sim'* par différentes fonctions de similarité (variante de Smith-Waterman proposée par Monge et Elkan, Jaro et Jaro-Winkler). Ils ont nommé ce type de fonctions hybrides des fonctions de similarité de niveau 2. Une telle fonction pour laquelle *sim'* est normalisée est elle aussi normalisée.

Ces mêmes chercheurs ont aussi conçu une version « souple » de la méthode TFIDF qui tient compte, en plus des lexèmes communs aux deux chaînes de caractères comparées, des lexèmes d'une chaîne qui sont suffisamment similaires à ceux de l'autre. Spécifiquement,

$$softTFIDF(c, d) = \frac{\sum_{i=1}^{l} \left(p_{ci} \times p_{ds} \times s(l_i, d, \theta) \right)}{\sqrt{\sum_{i=1}^{l} p_{ci}^{\ 2}} \times \sqrt{\sum_{i=1}^{l} p_{di}^{\ 2}}}$$

où

$$s(l_i, d, \theta) = \max_{j=1}^{p} \begin{cases} sim(l_i, b_j), si\ sim(l_i, b_j) > \theta, \\ 0, sinon; \end{cases}$$

et p_{ds} est le poids du lexème l_s dans d pour lequel $s(l_i, d, \theta)$ est maximale ($p_{ds} = p_{di}$ si l_i est inclus dans d).

Dans leurs expérimentations, Cohen, Ravikumar et Fienberg ont utilisé Jaro-Winkler comme fonction de similarité interne (*sim*) avec un seuil $\theta = 0{,}9$. Cette combinaison a, en moyenne sur plusieurs jeux de données provenant de différents secteurs d'activité, obtenu les meilleurs résultats parmi toutes les fonctions de similarité qu'ils ont testées. Enfin, il importe de constater que, comme la méthode de niveau 2, la fonction softTFIDF est asymétrique.

2.2 Utilisation d'algorithmes d'apprentissage pour combiner des fonctions de similarité

Lors de la comparaison de deux enregistrements, un modèle décisionnel doit être utilisé afin de déterminer si les deux enregistrements représentent la même entité. Un modèle simple consiste à évaluer la similarité entre les deux enregistrements à l'aide d'une fonction de similarité et de conclure que ces enregistrements sont des doublons si leur niveau de similarité est supérieur à un seuil prédéterminé. Un modèle décisionnel peut aussi être élaboré manuellement à partir d'une connaissance logique, statistique et empirique du domaine et, plus spécifiquement, des données dans lesquelles il est appliqué. De telles approches ont été utilisées avec succès dans différents domaines [16, 17, 18 et 43].

Par contre, il est possible de combiner des fonctions de similarité à l'aide de méthodes génériques et autonomes qui ne nécessitent pas une bonne connaissance du domaine d'application ni un examen minutieux des données à traiter et des situations particulières expliquant la présence de doublons approximatifs.

Plusieurs auteurs ont utilisé des algorithmes d'apprentissage afin de combiner efficacement des fonctions de similarité. Afin de résoudre ce problème de classification, des mesures de similarité évaluées pour chaque paire d'enregistrements, pour lesquelles a été déterminé au préalable s'il s'agissait de doublons, sont utilisées comme variables prédictives. L'objectif des algorithmes ainsi utilisés est donc d'apprendre comment agencer les fonctions de similarité afin d'identifier le plus efficacement possible les doublons approximatifs dans un ensemble d'enregistrements donné.

Adoptant une telle approche, Cohen et Richman [6] (voir aussi [7]) ont utilisé un algorithme de classification automatique à entropie maximale [28]. Ils ont entre autres utilisé les variables binaires suivantes qu'ils ont évaluées pour chaque paire de champs :

- *Préfixe* : vraie si une chaîne de caractères est un préfixe de l'autre.
- $DistanceÉdition(l)$: vraie si la distance d'édition entre les deux chaînes de caractères est inférieure à l (pour $l \in \{0,5;1;2;4;8;16;32;64\}$).
- $LexèmePrésent1(i)$: vraie si le i-ième lexème de la première chaîne de caractères est présent dans l'autre chaîne de caractères.

Avec cette approche, les auteurs ont obtenu des résultats comparables à sinon meilleurs que ceux obtenus à partir d'une fonction de similarité basée sur la distance d'édition ou à partir d'une fonction basée sur la méthode TFIDF.

Cohen, Ravikumar et Fienberg [8] ont comparé une combinaison de mesures de similarité formée à l'aide d'un algorithme d'apprentissage de type SVM (*support vector machines*) [3] avec la méthode hybride softTFIDF. Selon leurs expérimentations, la performance de la combinaison issue d'un apprentissage serait légèrement supérieure.

Il est aussi possible d'utiliser des algorithmes d'apprentissage actifs [11] pour déterminer automatiquement le statut d'une paire d'enregistrements. Les

algorithmes d'apprentissage actifs se distinguent des algorithmes d'apprentissage cités précédemment, qu'on dit de type passif, par leur capacité à choisir les exemples (des paires d'enregistrement dans ce cas-ci) sur lesquels ils s'entraînent. En tentant de choisir les paires d'enregistrement qui, une fois identifiées comme étant ou non des doublons, lui procureront un maximum d'information, un algorithme d'apprentissage actif converge en général beaucoup plus rapidement (à l'aide de moins d'exemples que si on avait utilisé un algorithme d'apprentissage passif) vers un modèle décisionnel satisfaisant. Du coup, cette technique d'apprentissage réduit substantiellement la tâche d'un humain qui aurait à examiner des paires d'enregistrements pour constituer une banque d'exemples d'entraînement.

Adoptant la stratégie décrite ci haut, Tejada, Knoblock et Minton [41] ont proposé l'utilisation de la technique *query by bagging* [1], une variante de la technique *query by committee* [38], qui repose sur le *bagging* [4]. Cette technique procède par itérations où un comité d'algorithmes d'apprentissage (les chercheurs ont choisi d'utiliser des arbres de décision [32]), entraînés chacun à partir de groupes d'exemples différents, se prononcent sur le statut des paires d'enregistrements pour lesquelles on ne sait pas s'il s'agit de doublons. La paire d'enregistrements pour laquelle le niveau de désaccord à l'intérieur du comité est le plus élevé est inspectée par un humain. Une fois le statut de cette paire d'enregistrements déterminé, elle est ajouté à la banque d'exemples d'entraînement qui serviront lors de la prochaine itération. Les itérations se succèdent ainsi jusqu'à ce qu'on atteigne un taux de bonne classification satisfaisant. Sarawagi et Bhamidipaty [36] ont aussi utilisé des approches semblables. Comme on pouvait s'y attendre, les expérimentations des deux groupes de chercheurs montrent que l'utilisation de techniques d'apprentissage actives réduit considérablement le nombre de paires d'enregistrements qui doivent être examinées afin d'obtenir un niveau de précision satisfaisant. Dans leurs expérimentations, le recours a une approche active a permis de réduire jusqu'à 50 fois le nombre de paires d'enregistrements qui devaient être examinées.

2.3 Techniques pour réduire le nombre de doublons potentiels

Tel que noté précédemment, appliquer des règles de décision complexes, qui reposent sur des mesures de similarité entre chaînes de caractères, sur toutes les paires d'enregistrements dans lesquelles on désire détecter la présence de doublons n'est pas une tâche réalisable lorsque le nombre d'enregistrements est moindrement élevé. Il en est de même lors de l'entraînement d'un algorithme d'apprentissage destiné à combiner des mesures de similarité. À titre d'exemple, considérer comme doublons potentiels toutes les paires d'enregistrements dans un fichier de seulement 1000 enregistrements revient à effectuer près de 500 000 comparaisons $\left(\frac{1000(1000-1)}{2}\right)$. Pour contrer ce problème, plusieurs chercheurs ont proposé des méthodes pour réduire substantiellement le nombre de doublons potentiels à considérer [8, 17, 24, 26 et 41]. Baxter, Christen et Churches [2] en ont d'ailleurs comparées quelques-unes.

Les méthodes de réduction du nombre de doublons potentiels influencent la précision du processus de détection de doublons approximatifs. En effet, il est possible qu'elles éliminent des paires d'enregistrements qui sont réellement constituées de doublons. Cependant, en réduisant le nombre de paires d'enregistrements à comparer, elles permettent l'utilisation de fonctions de similarité qui demandent davantage de ressources *computationnelles*, et qui sont par conséquent potentiellement plus précises.

3 Méthodologie

Dans le cadre de ce mémoire, plusieurs fonctions de similarité ont été comparées sur des jeux de données réelles à l'aide de la librairie Java SecondString [10] développée par Cohen, Ravikumar et Fienberg [8]. Spécifiquement, huit fonctions basées sur l'analyse des caractères, deux fonctions basées sur les lexèmes ainsi qu'une fonction basée sur les *n*-grams ont été évaluées. De plus, certaines de ces fonctions ont été utilisées dans des méthodes hybrides pour former d'autres fonctions de similarité qui ont elles aussi été comparées.

3.1 Spécifications des fonctions de similarité évaluées

Les distances d'édition à trois et quatre opérations de coût unitaire, la longueur de la PLSC, la fonction Jaro normalisée avec des poids égaux et la distance d'édition de Yancey ont été évaluées. De plus, les mesures de similarité de Smith et Waterman, avec suppression de caractères successifs pénalisée de façon linéaire et affine, ainsi que la variante de cette mesure proposée par Monge et Elkan complètent la liste des fonctions de similarité basées sur l'analyse des caractères qui ont été comparées dans cette étude. Pour la fonction de similarité de Smith et Waterman et ses dérivés, le système de pointage utilisé par Monge et Elkan [25] a été retenu. C'est-à-dire que

$$sim(C_i, D_j) = \begin{cases} 5, si\ C_i = D_j, \\ 2, si\ C_i\ et\ D_j\ sont\ approximativement\ identiques, \\ -5, sinon; \end{cases}$$

v, le coût pour débuter une séquence de suppressions successives (ou pour supprimer un caractère dans le cas de la version linéaire), est égal à 5 et *u*, le coût pour poursuivre la séquence (ne s'applique qu'à la version affine), est égal à 1. Évidemment, les caractères approximativement identiques sont considérés que dans la variante de Monge et Elkan.

Comme les mesures de similarité non normalisées dépendent généralement de la longueur des chaînes de caractères comparées (ex. : il est plus probable que la distance d'édition entre deux longues chaînes soit plus élevée que celle entre des chaînes de petites tailles) et que cela peut compromettre la précision des mesures de similarité dans des jeux de données où la longueur des chaînes de caractères varie considérablement, il a été jugé opportun de normaliser les fonctions de similarité qui ne l'étaient pas. À l'exception des fonctions de similarité de Jaro et Yancey, qui sont déjà normalisées, les fonctions basées sur l'analyse des caractères considérées dans ce projet de recherche ont toutes été évaluées en version originale et normalisée.

Comme il a été vu précédemment, la distance d'édition avec opérations de coût unitaire peut être normalisée en fonction de la longueur de la plus longue des deux chaînes de caractères comparées.

La PLSC qu'il est possible d'obtenir à partir de n'importe quelle paire de chaînes de caractères est évidemment la plus courte des deux chaînes. Normaliser la longueur de la PLSC en la divisant par la longueur de la plus courte des deux chaînes revient à accorder un niveau de similarité maximal lorsqu'une chaîne est une abréviation de l'autre et non seulement lorsque les deux chaînes sont identiques. Pour palier à ce possible inconvénient, il a aussi été décidé de normaliser la longueur de la PLSC d'une deuxième façon : en la divisant par la longueur de la plus longue des deux chaînes de caractères comparées.

À partir de la fonction de similarité de Smith et Waterman ou de ses dérivés, on obtient le niveau de similarité le plus élevé possible entre deux chaînes de caractères lorsque l'une est une sous-chaîne de l'autre. Dans un tel cas, le niveau de similarité maximal *simMax* est égal à la longueur de la plus courte des deux chaînes de caractères multipliée par le nombre de points accordés pour chaque caractère *pairé* (5 points dans la présente étude). On peut donc obtenir une version normalisée de la mesure de Smith et Waterman en la divisant simplement par

simMax. Selon un raisonnement analogue à celui évoqué pour la PLSC, il a aussi été convenu de normaliser la mesure de similarité de Smith et Waterman ou de ses dérivés en fonction de la plus longue des deux chaînes de caractères comparées (*simMax* est alors égale à la longueur de la plus longue des deux chaînes multipliée par le nombre de points accordés pour chaque caractère *pairé*).

La mesure de similarité de Jaccard ainsi que la méthode TFIDF telle qu'utilisée par Cohen, Ravikumar et Fienberg sont les deux fonctions de similarité basées sur l'analyse des lexèmes qui ont été évaluées dans le cadre de ce mémoire.

En ce qui concerne les fonctions de similarité basées sur les *n*-grams, la méthode utilisée par Hylton ainsi qu'une variante normalisée ont été étudiées. Pour n'importe quelle paire de chaînes de caractères, la distance maximale qu'il est possible d'obtenir à partir de la méthode de Hylton survient lorsque les deux vecteurs de 3-grams comparés sont orthogonaux (aucun 3-gram commun aux deux chaînes de caractères). À partir de ce constat, on trouve que la distance euclidienne de Hylton peut être transformée en mesure de similarité normalisée avec la formule

$$HyltonNormalisée(c,d) = 1 - \frac{Hylton(c,d)}{\sqrt{\sum_{i=1}^{I} p_{ci}^{\ 2} + \sum_{i=1}^{I} p_{di}^{\ 2}}}.$$

La fonction softTFIDF telle qu'utilisée par Cohen, Ravikumar et Fienberg a été comparée aux autres fonctions de similarité listées dans cette section. Tant pour cette fonction hybride que pour la méthode TFIDF, le poids *IDF* de chacun des lexèmes a été déterminé à partir de l'ensemble des chaînes de caractères comprises dans le jeu de données où la fonction de similarité a été appliquée.

De plus, les mesures de similarité normalisées basées sur l'analyse des caractères ont été évaluées avec et sans l'application de la méthode de Winkler. D'autres

fonctions de similarité ont aussi été formées en utilisant toutes les mesures basées sur les caractères ou les 3-grams, avec et sans l'application de la méthode de Winkler (pour les mesures normalisées basées sur les caractères), dans la méthode hybride de niveau 2.

Par ailleurs, de par leur nature, la précision des fonctions de similarité basées sur l'analyse des lexèmes (et des fonctions hybrides) est grandement moins affectée par la façon dont sont utilisés les caractères de ponctuation dans les chaînes de caractères comparées. Par exemple, les numéros de téléphone « 999-555-1212 », « (999) 555 1212 » et « 999.555.1212 » sont tous considérés identiques par des mesures de similarité basées sur les lexèmes alors que les autres types de mesures sont pénalisées par les caractères de ponctuation qui diffèrent d'une chaîne à l'autre. Pour contrevenir à ce « faux » avantage qu'ont les fonctions de similarité basées sur les lexèmes et mesurer son impact, la performance des autres fonctions a été évaluée avec et sans l'élimination des caractères de ponctuation (tous les caractères à l'exception des lettres et des chiffres).

Ainsi, 99 combinaisons (voir tableau 1) ont été évaluées sur huit jeux de données réelles provenant de différents secteurs d'activité. Lors de ces expériences, aucune distinction n'a été faite en ce qui concerne la casse des lettres (ex. : « John Doe » et « john doe » ont été considérées identiques). Pour les deux méthodes hybrides considérées dans cette étude (softTFIDF et niveau 2), il a été décidé de les évaluer en double, en interchangeant l'ordre des chaînes de caractères comparées, afin d'évaluer les effets de leur asymétrie. Puisque seule la fonction de similarité interne Jaro-Winkler a été considérée pour la méthode softTFIDF, alors que plusieurs fonctions de similarité internes ont été utilisées pour la méthode de niveau 2, les effets de l'asymétrie des deux méthodes hybrides ont été évaluées que pour la fonction Jaro-Winkler.

Fonction de similarité de base	Combinaison de méthodes appliquée sur les fonctions de similarité de base			
	Élimination des caractères de ponctuation	Normalisation	Winkler	Méthode de niveau 2
distance d'édition à trois opérations	-	-	n/a	-
distance d'édition à quatre opérations	-	-	n/a	2
	-	nl	-	-
	-	nl	-	2
	-	nl	W	-
	-	nl	W	2
	ep	-	n/a	n/a
	ep	nl	-	n/a
	ep	nl	W	n/a
PLSC	-	-	n/a	-
Smith-Waterman linéaire	-	-	n/a	2
Smith-Waterman affine	-	nc	-	-
Monge-Elkan	-	nc	-	2
	-	nc	W	-
	-	nc	W	2
	-	nl	-	-
	-	nl	-	2
	-	nl	W	-
	-	nl	W	2
	ep	-	n/a	n/a
	ep	nc	-	n/a
	ep	nc	W	n/a
	ep	nl	-	n/a
	ep	nl	W	n/a
Jaro	-	n/a*	-	-
Yancey	-	n/a*	-	2
	-	n/a*	W	-
	-	n/a*	W	2
	ep	n/a*	-	n/a
	ep	n/a*	W	n/a
Hylton	-	-	-	-
	-	-	-	2
	-	n	-	-
	-	n	-	2
	ep	-	-	n/a
	ep	n	-	n/a
Jaccard TFIDF softTFIDF	-	n/a*	-	n/a

ep : élimination des caractères de ponctuation
n : normalisée
nl : normalisée en fonction de la plus longue chaîne de caractères
nc : normalisée en fonction de la plus courte chaîne de caractères
W : ajustée par la méthode de Winkler
2 : utilisée comme mesure de similarité interne dans *niveau2*
* : les fonctions Jaro, Yancey Jaccard, TFIDF et softTFIDF sont déjà normalisées

Tableau 1 : Liste des fonctions de similarité évaluées

3.2 Description des jeux de données

Les tableaux 2 et 3 décrivent les huit jeux de données réelles sur lesquels ont été réalisées les expériences. Ces jeux de données font partie de ceux qui ont été utilisés par Cohen, Ravikumar et Fienberg [8] (ces jeux de données sont d'ailleurs

fournis dans la librairie SecondString). Plus particulièrement, six d'entre eux (j, o{1-4} et p) ont été décrits et utilisés dans des tâches semblables pour la première fois par Cohen [5] alors que le jeu de données traitant des restaurants (r) a été utilisé initialement par Tejada, Knoblock et Minton [41].

Jeu de données	**Description** **Exemple de doublon**
Jeux (j)	Noms de jeux vidéos éducatifs « Colorforms Computer Fun Set - Power Rangers » « Power Rangers Colorforms Computer Fun Set »
oiseaux1 (o1)	Noms communs et/ou scientifiques (latins) d'oiseaux d'Amérique du Nord « Song Sparrow » « Song Sparrow (Melospiza melodia) »
oiseaux2 (o2)	Noms communs d'oiseaux d'Amérique du Nord quelquefois accompagnés d'éléments d'information relatifs à une photo de l'oiseau « Yellow-rumped Warbler « Yellow-rumped ("Myrtle") Warbler 41kB Dendroica coronata Blue Mountain Lake, May 1996. »
oiseaux3 (o3)	Noms communs d'oiseaux d'Amérique du Nord quelquefois accompagnés du nom francophone et/ou du nom scientifique « American Avocet » « Avocet, American (Avocette d'Amérique) : Recurvirostra americana »
oiseaux4 (o4)	Noms communs d'oiseaux d'Amérique du Nord généralement accompagnés du nom scientifique et quelquefois du nom francophone « Clark's Nutcracker Nucifraga columbiana » « Nutcracker, Clark's : Nucifraga columbiana »
parcs (p)	Noms de parcs américains « Nez Perce NHP » « Nez Perce National Historical Park »
restaurants (r)	Noms de restaurants accompagnés d'une adresse, d'un numéro de téléphone et du type de cuisine qu'on y sert « Mesa Grill 102 Fifth Ave. New York City 212-807-7400 Southwestern » « Mesa Grill 102 5th Ave. between 15th and 16th Sts. New York 212/807-7400 American »
universités (u)	Noms d'université et de centres de recherche généralement accompagnés de la ville et quelquefois de l'état où se situe l'organisation « Virginia, University, Charlottesville » « University of VA., Charlottesville »

Tableau 2 : Description des jeux de données utilisés et exemples de doublons

Jeu de données	Nombre de chaînes de caractères	Nombre d'entités	Longueur moyenne des chaînes de caractères	Écart-type de la longueur des chaînes de caractères
jeux	968	901	29	32
oiseaux1	336	317	39	8
oiseaux2	1050	918	23	32
oiseaux3	38	23	22	16
oiseaux4	719	564	39	11
parcs	654	399	19	7
restaurants	863	749	69	12
universités	116	57	46	19

Tableau 3 : Description quantitative des jeux de données utilisés

Dans ces jeux de données anglophones, les entités sont décrites à l'aide d'un ou plusieurs éléments d'information (ex. : nom, ville, numéro de téléphone, etc.) contenus à l'intérieur d'une même chaîne de caractères. Les éléments d'information ne sont pas toujours tous présents ni délimités selon une structure bien définie. De plus, peu de normes sont présentes quant à la façon dont sont inscrits les différents éléments d'information (ex. : « 168 first avenue » et « 168 1st ave. » servent à désigner la même adresse). Enfin, les relations entre les chaînes de caractères et les entités qu'elles décrivent sont connues et ce, pour tous les jeux de données.

3.3 Réduction du nombre de doublons potentiels

Afin de limiter le nombre de paires de chaînes de caractères sur lesquelles ont été appliquées les fonctions de similarité, il a été convenu d'utiliser la même approche que Cohen, Ravikumar et Fienberg [8]. Spécifiquement, la similarité d'une paire de chaînes de caractères n'a été évaluée que si les deux chaînes partageaient au moins un lexème. Le tableau 4 présente la performance de cette méthode pour chacun des jeux de données. Le rappel, tel que présenté dans ce tableau, consiste en la proportion de doublons qui n'ont pas été éliminés par la technique de réduction du nombre de doublons potentiels.

Jeu de données	Réduction du nombre de doublons potentiels	Rappel
jeux	90,9%	100,0%
oiseaux1	93,9%	100,0%
oiseaux2	95,7%	100,0%
oiseaux3	96,0%	93,3%
oiseaux4	95,6%	100,0%
parcs	86,5%	96,9%
restaurants	42,4%	100,0%
universités	74,1%	95,7%
Moyenne	84,4%	98,2%

Tableau 4 : Performance de la réduction du nombre de doublons potentiels

3.4 Critères d'évaluation des fonctions de similarité

Pour évaluer la performance d'une fonction de similarité sur un jeu de données, les paires de chaînes de caractères ont été triées en ordre décroissant de niveau de similarité afin de calculer la précision et le rappel pour différents rangs. Ces deux mesures sont largement utilisées en recherche d'information. La précision au rang i est égale à $d(i)$, le nombre de doublons présents jusqu'à la position i, divisé par i. Le rappel au rang i est égal à $d(i)$ divisé par D, le nombre total de doublons présents parmi les paires de chaînes de caractères comparées (celles qui n'ont pas été éliminées par la technique de réduction du nombre de doublons potentiels).

Typiquement, lorsque i augmente, le rappel augmente et la précision diminue. En pratique, une valeur sur l'espace engendré par la fonction de similarité est choisie comme seuil pour établir une règle de décision. Ce seuil est déterminé de façon à obtenir un compromis qui satisfait les besoins spécifiques de l'application. Ici, afin d'évaluer la performance des fonctions de similarité à la fois en terme de précision et de rappel, la précision moyenne non interpolée et la valeur maximum de la mesure F1 (maxF1) ont été calculées ainsi que la précision aux 11 niveaux de rappel suivants : 0,0; 0,1; …, 0,9; 1,0. La précision moyenne est $\frac{1}{D}\sum_{i=1}^{N}\frac{d(i)\times\delta(i)}{i}$

où N est le nombre de paires de chaînes de caractères comparées et $\delta(i) = 1$ si la paire de chaîne de caractères au rang i est un doublon, 0 sinon. La valeur maximum de la mesure F1 est $\max_{i=1}^{N} \mathrm{F1}(i)$ où $\mathrm{F1}(i)$ est la moyenne harmonique du rappel et de la précision au rang i;

$$\mathrm{F1}(i) = \frac{2 \times d(i)/D \times d(i)/i}{d(i)/D + d(i)/i}.$$

La précision pour un niveau de rappel r est la précision la plus élevée parmi tous les rangs où le rappel est supérieur ou égal à r.

4 Résultats expérimentaux et discussion

4.1 Effets de l'asymétrie de la méthode de niveau 2 et de softTFIDF

Les figures 2 et 3 illustrent les effets de l'asymétrie de la méthode de niveau 2 (avec Jaro-Winkler comme mesure de similarité interne) et de la fonction softTFIDF. Chaque expérience y est représentée sous la forme d'une cordonnée (x, y) où x et y sont les précisions moyennes (ou les valeurs de la mesure maxF1) obtenues en interchangeant l'ordre des chaînes de caractères comparées. Sur un tel graphique, les coordonnées obtenues à partir d'une fonction de similarité symétrique seraient toutes situées sur la droite $y = x$.

La figure 2 montre clairement que l'ordre avec lequel sont considérées les chaînes de caractères peut avoir un impact important sur la performance de la méthode de niveau 2. Pour le jeu de données *o1* par exemple, la précision moyenne passe de 0,93 à 0,05 lorsqu'on change l'ordre des chaînes de caractères comparées. En ce qui concerne la fonction softTFIDF, les effets de son asymétrie sont nettement moins marqués que pour la méthode niveau 2. Pour tous les jeux de données, la précision moyenne et la valeur de la mesure maxF1 de la fonction softTFIDF sont, à quelques centièmes près, identiques lorsqu'on change l'ordre des chaînes de caractères comparées.

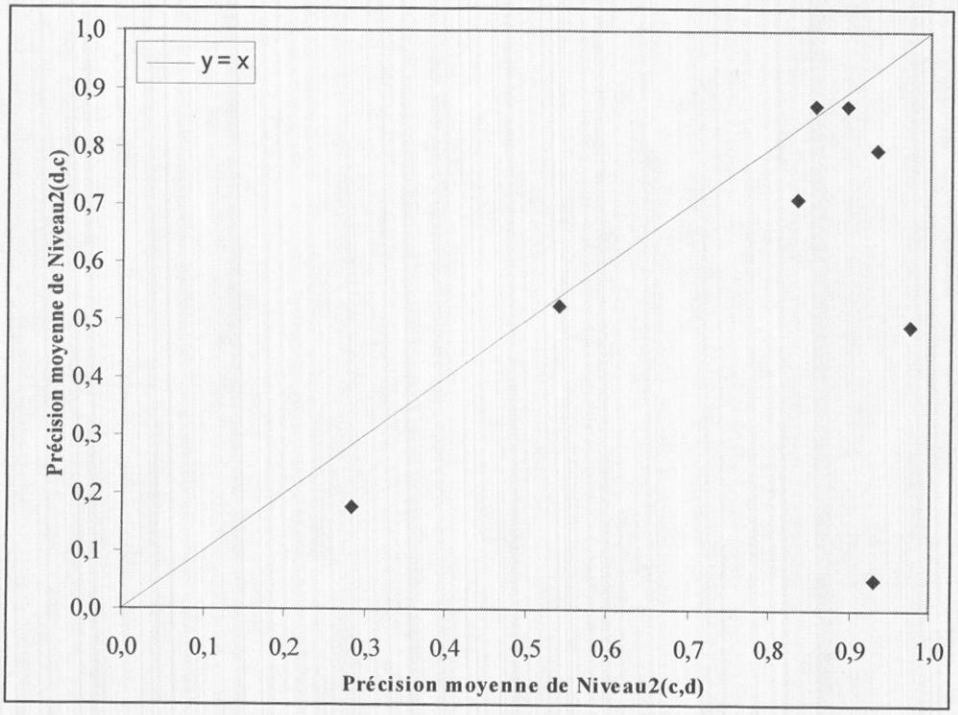
Cette distinction importante entre ces deux méthodes hybrides, quant aux effets de leur asymétrie, est en grande partie causée par le seuil θ, utilisé dans softTFIDF, qui fait en sorte que les paires de lexèmes qui ne sont pas suffisamment similaires ne sont pas considérées dans le calcul de la similarité entre deux chaînes de caractères. Par exemple, si on compare deux chaînes *c* et *d*, qui sont constituées respectivement de trois et quatre lexèmes distincts, il est fort probable qu'un des quatre lexèmes de *d* ne soit similaire à aucun des lexèmes de *c*. En comparant *d* et

c (et non c et d) à l'aide de la méthode de niveau 2, on doit quand même considérer la similarité entre ce lexème et n'importe quel lexème de d, et ce même si les lexèmes ne se ressemblent pas. Par conséquent, la similarité entre c et d et celle entre d et c peut varier considérablement. L'utilisation du seuil θ, dans softTFIDF, vient régler en bonne partie ce problème.

Suite à l'observation que l'asymétrie de la méthode de niveau 2 pose un problème évident, il a été décidé de considérer une variante symétrique de cette méthode;

$$niveau2Symétrique(c,d) = \frac{niveau2(c,d) + niveau2(d,c)}{2}.$$

Pour évaluer la performance de cette nouvelle variante, elle a été appliquée sur toutes les fonctions de similarité où la variante originale de la méthode de niveau 2 a aussi été appliquée.



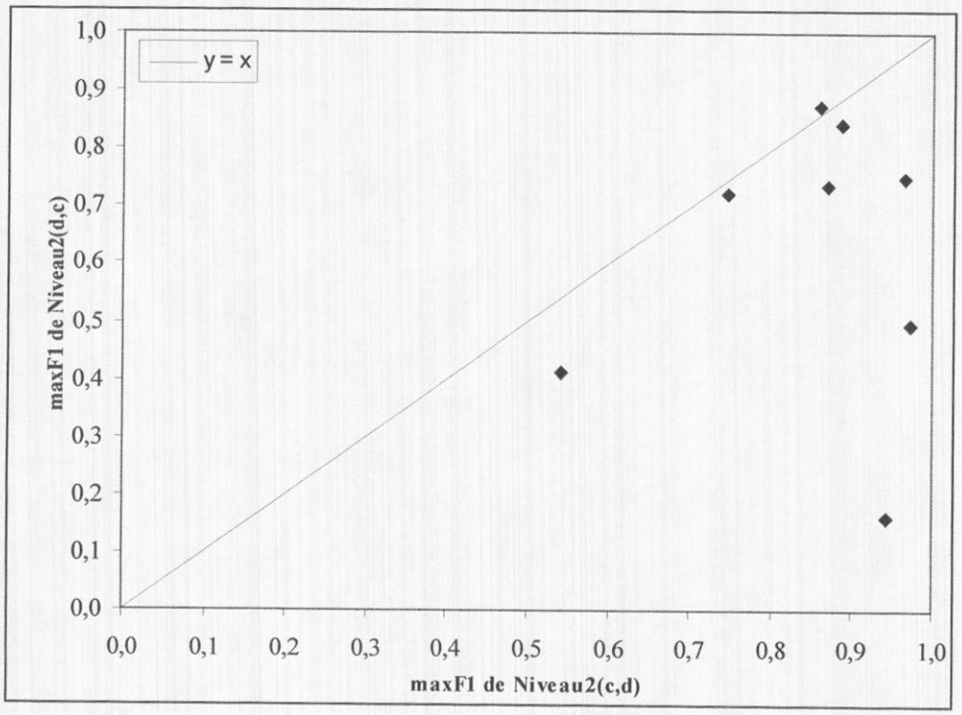


Figure 2 : Effets de l'asymétrie de la méthode de niveau 2 (avec Jaro-Winkler comme mesure de similarité interne)

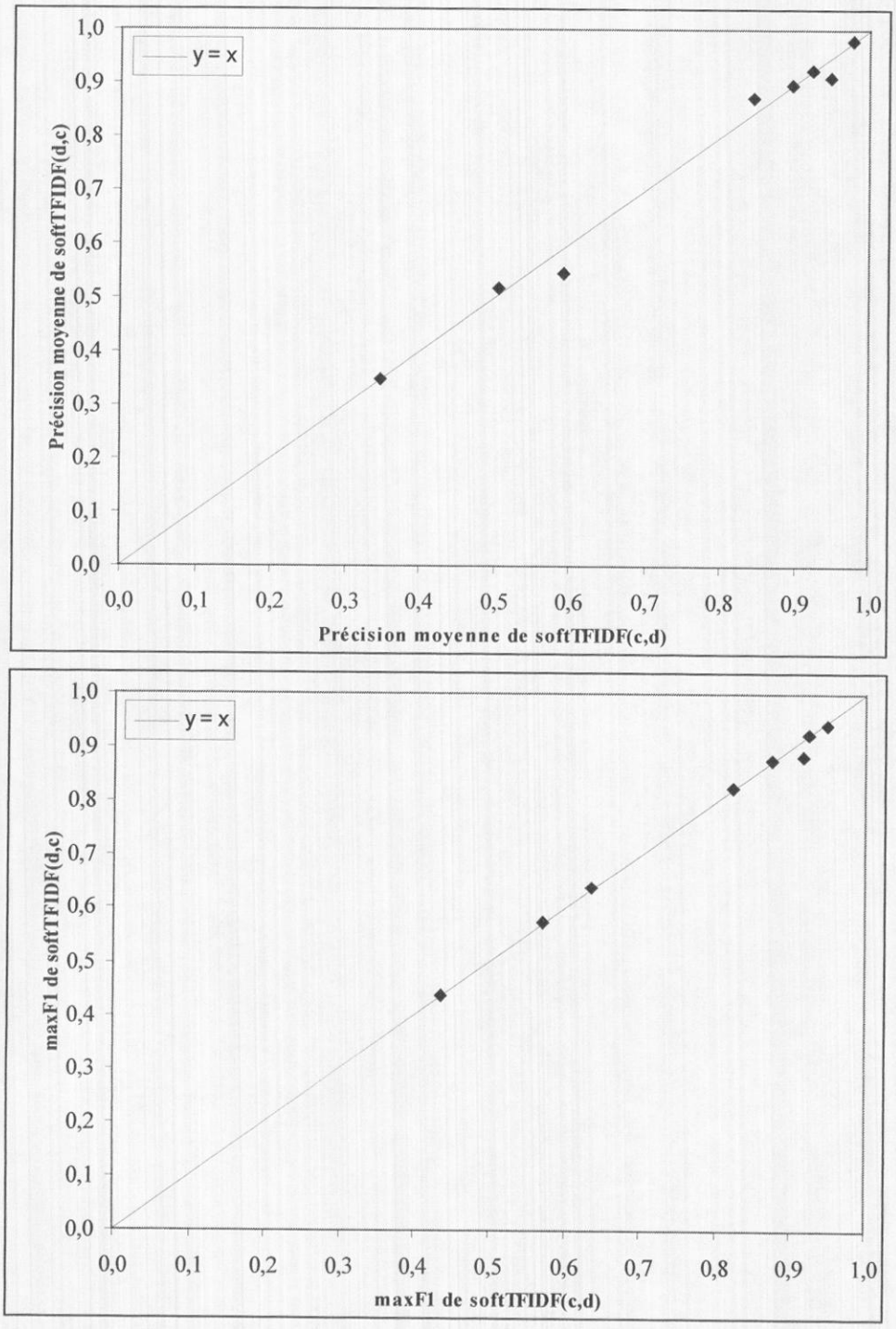


Figure 3 : Effets de l'asymétrie de la fonction softTFIDF

4.2 Évaluation générale des fonctions de similarité

Le tableau 5 présente les résultats obtenus pour 11 fonctions de similarité qui retiennent l'attention (voir l'annexe pour les résultats de toutes les fonctions). C'est-à-dire les cinq fonctions qui, en moyenne sur l'ensemble des jeux de données, sont les plus précises (en terme de précision moyenne) et les fonctions les plus précises pour chacun des jeux de données. On y retrouve aussi les résultats de la méthode TFIDF qui constitue un très bon compromis entre précision et temps d'exécution. Compte tenu de sa rapidité d'exécution, la méthode TFIDF peut d'ailleurs être utilisée dans des techniques de réduction du nombre de doublons potentiels (voir par exemple [24]). Le choix d'avoir ordonné, pour chacun des jeux de données, les fonctions selon leur précision moyenne plutôt que selon le critère maxF1 est purement arbitraire et sans impact important puisque ces deux critères d'évaluation sont fortement corrélés ($\hat{\rho} = 0{,}967$). D'ailleurs, pour six des huit jeux de données, la fonction de similarité qui a la précision moyenne la plus élevée a aussi la valeur maxF1 la plus élevée. Pour les autres jeux de données, la différence est négligeable. De plus, les deux fonctions qui ont, en moyenne sur l'ensemble des jeux de données, les précisions moyennes les plus grandes ont aussi les valeurs moyennes de la mesure maxF1 les plus grandes.

Les fonctions de similarité s'avèrent performantes dans tous les jeux de données à l'exception de celui traitant des jeux vidéo éducatifs (j) où la méthode la plus précise donne des résultats nettement inférieurs aux résultats les plus élevés obtenus dans les autres jeux de données. Cette faible performance s'explique en grande partie par la présence de jeux vidéos distincts qui sont des suites (par exemple, « Mario Teaches Typing » et « Mario Teaches Typing 2 »). À moins de fixer un seuil très élevé pour trancher entre les paires de chaînes de caractères qu'on doit considérer comme étant des doublons et les autres, ce qui aurait pour conséquence de laisser de côté plusieurs doublons, cette situation particulière crée évidemment un nombre important de faux positifs (des paires identifiées comme étant des doublons, mais qui ne le sont pas) peu importe la fonction de similarité

utilisée. Ce jeu de données est un bon exemple de contexte où il peut être nécessaire d'utiliser des règles de décision propres au domaine d'application pour obtenir des résultats satisfaisants.

Fonction de similarité	Rang par jeu de données* (précision moyenne)								Moyennes†	Temps d'éxécution moyen
	j	o1	o2	o3	o4	p	r	u		
Smith-Waterman affine (ep, nc)	63 (0,26)	5 (0,93)	**1 (0,87)**	59 (0,82)	26 (0,85)	5 (0,93)	52 (0,89)	45 (0,80)	32,0 (0,79)	48,9 s
Monge-Elkan (ep, nc)	64 (0,26)	6 (0,93)	2 (0,87)	64 (0,78)	27 (0,84)	3 (0,93)	47 (0,89)	47 (0,80)	32,5 (0,79)	95,6 s
Smith-Waterman linéaire (nl, 2sym)	8 (0,47)	17 (0,76)	33 (0,36)	**1 (0,93)**	2 (0,97)	55 (0,89)	11 (0,95)	19 (0,89)	18,3 (0,78)	62,8 s
Smith-Waterman affine (nl, 2sym)	9 (0,47)	18 (0,76)	34 (0,36)	**1 (0,93)**	**1 (0,97)**	56 (0,89)	10 (0,95)	20 (0,89)	18,6 (0,78)	109,8 s
Smith-Waterman linéaire (nl, W, 2sym)	3 (0,47)	20 (0,72)	36 (0,35)	**1 (0,93)**	3 (0,97)	44 (0,90)	14 (0,95)	13 (0,89)	16,8 (0,77)	63,4 s
TFIDF	2 (0,49)	19 (0,76)	40 (0,34)	45 (0,84)	6 (0,96)	31 (0,91)	2 (0,98)	39 (0,82)	23,0 (0,76)	0,1 s
Monge-Elkan (ep, nc, W)	66 (0,26)	4 (0,94)	4 (0,87)	86 (0,66)	37 (0,70)	**1 (0,93)**	3 (0,96)	60 (0,78)	32,6 (0,76)	96,5 s
distance d'édition à trois opérations (nl, 2sym)	13 (0,46)	27 (0,65)	41 (0,33)	**1 (0,93)**	12 (0,96)	69 (0,88)	8 (0,95)	17 (0,89)	23,5 (0,76)	38,6 s
softTFIDF	**1 (0,52)**	35 (0,54)	35 (0,35)	30 (0,87)	23 (0,91)	15 (0,92)	**1 (0,98)**	7 (0,90)	18,4 (0,75)	27,3 s
Smith-Waterman linéaire (ep, nc, W)	68 (0,26)	**1 (0,95)**	9 (0,86)	83 (0,66)	94 (0,30)	4 (0,93)	34 (0,91)	84 (0,75)	47,1 (0,70)	27,9 s
Monge-Elkan (nc, W, 2sym)	82 (0,18)	92 (0,06)	112 (0,12)	**1 (0,93)**	55 (0,56)	65 (0,89)	61 (0,87)	**1 (0,90)**	58,6 (0,56)	155,7 s

* : voir tableau 2 pour la liste des jeux de données
† : rang moyen et moyenne de la précision moyenne sur l'ensemble des jeux de données
2sym : utilisée comme mesure de similarité interne dans *niveau2Symétrique*

Tableau 5 : Performance de 11 fonctions de similarité

Par ailleurs, on remarque que toutes les fonctions de similarité présentées dans le tableau 5 sont normalisées entre 0 et 1. Cela vient appuyer le raisonnement décrit

précédemment à l'effet que la précision des fonctions de similarité non normalisées peut être compromise dans des jeux de données où la longueur des chaînes de caractères varie considérablement. De plus, les résultats obtenus semblent indiquer que normaliser les chaînes de caractères selon la plus courte chaîne est plus efficace lorsque les fonctions de similarité sont utilisées telles quelles, sans être incluse dans une méthode hybride. Toutefois, lorsqu'elles sont utilisées comme mesure de similarité interne de la variante symétrique de la méthode hybride de niveau 2, il apparaît être généralement plus judicieux de les normaliser en fonction de la plus longue chaîne. Cela est probablement causé par le fait qu'elles sont appliquées dans ce cas sur de plus petites chaînes de caractères (des lexèmes) où, intuitivement, il importe de pénaliser davantage le niveau de similarité mesuré entre deux chaînes de longueurs différentes.

Les résultats montrent aussi qu'aucune fonction de similarité ne fait partie des plus précises pour tous les jeux de données. À titre d'exemple, la fonction Monge-Elkan normalisée en fonction de la plus courte chaîne de caractères, ajustée par la méthode de Winkler et ensuite utilisée comme fonction interne dans une variante symétrique de la méthode de niveau 2 est la méthode la plus précise pour deux des huit jeux de données (*o3* et *u*), mais procure des résultats très peu satisfaisants dans quatre jeux de données (*j* et *o*{*1*, *2*, *4*}). D'un autre côté, la fonction de similarité de Smith et Waterman avec suppression de caractères successifs pénalisée de façon affine, normalisée en fonction de la plus courte chaîne de caractères et pour laquelle les caractères de ponctuation ont été éliminés, qui donne de bons résultats dans tous les jeux de données à l'exception de celui traitant des jeux vidéo éducatifs (*j*), n'est toutefois pas comparables aux fonctions les plus précises dans tous ces jeux de données. Par exemple, elle obtient une précision moyenne de 0,85 dans le jeu de données *o4* alors que la fonction la plus précise pour ce jeu de données a une précision moyenne de 0,97.

De façon plus générale, on constate que les fonctions de similarité basées sur l'analyse des caractères se démarquent dans les jeux de données *o*{*1*, *2*} alors que

les fonctions basées sur les lexèmes (ou les fonctions hybrides) ont un net avantage dans les jeux de données *j*, *o*{*3*, *4*} et *u*. En ce qui concerne les jeux de données *p* et *r*, la démarcation entre les fonctions basées sur les caractères et celles basées sur les lexèmes est moins claire. Dans les deux cas, on obtient de bons résultats à partir de fonctions de similarité basées sur les caractères ou sur les lexèmes. Cependant, les fonctions les plus précises dans le jeu de données *p* sont celles basées sur les caractères alors qu'on observe l'inverse dans le jeu de données *r*.

La grande variabilité de la précision des fonctions de similarité lorsqu'elles sont appliquées sur différents jeux de données démontrent l'importance d'expérimenter et de comparer plusieurs fonctions lorsqu'on est soumis à une tâche de détection de doublons approximatifs. Comme il a été vu précédemment, il est aussi possible de faire appel à des algorithmes d'apprentissage afin de bâtir efficacement une combinaison de fonctions de similarité pour un domaine d'application donné.

Toutefois, le choix d'une fonction de similarité ou d'une combinaison de fonctions nécessite un effort supplémentaire important. Notamment, pour être en mesure d'utiliser des algorithmes d'apprentissage ou tout simplement pour pouvoir évaluer la précision de différentes fonctions de similarité, cela requiert que plusieurs centaines (voire même milliers) de paires de chaînes de caractères ou d'enregistrements soient examinés afin de déterminer s'il s'agit de doublons. À défaut de fournir un tel effort, on doit utiliser des fonctions de similarité qui sont généralement bonnes dans divers domaines d'application ou mieux, qui ont fait leur preuve dans le domaine d'application dans lequel on désire détecter des doublons approximatifs.

À cet effet, le tableau 5 présente les cinq fonctions de similarité qui sont, en moyenne sur les huit jeux de données, les plus précises. La performance de ces cinq fonctions est aussi illustrée dans la figure 4 à l'aide de courbes moyennes de précision-rappel. Pour former chacune de ces courbes, on calcule la moyenne de la

précision, sur tous les jeux de données, pour chacun des 11 niveaux de rappel prédéterminés. Pour fins de comparaison, on retrouve aussi sur ce graphique la fonction de similarité softTFIDF qui a été identifiée récemment par Cohen, Ravikumar et Fienberg comme étant, en moyenne sur plusieurs jeux de données, la fonction la plus précise parmi les nombreuses fonctions qu'ils ont évaluées.

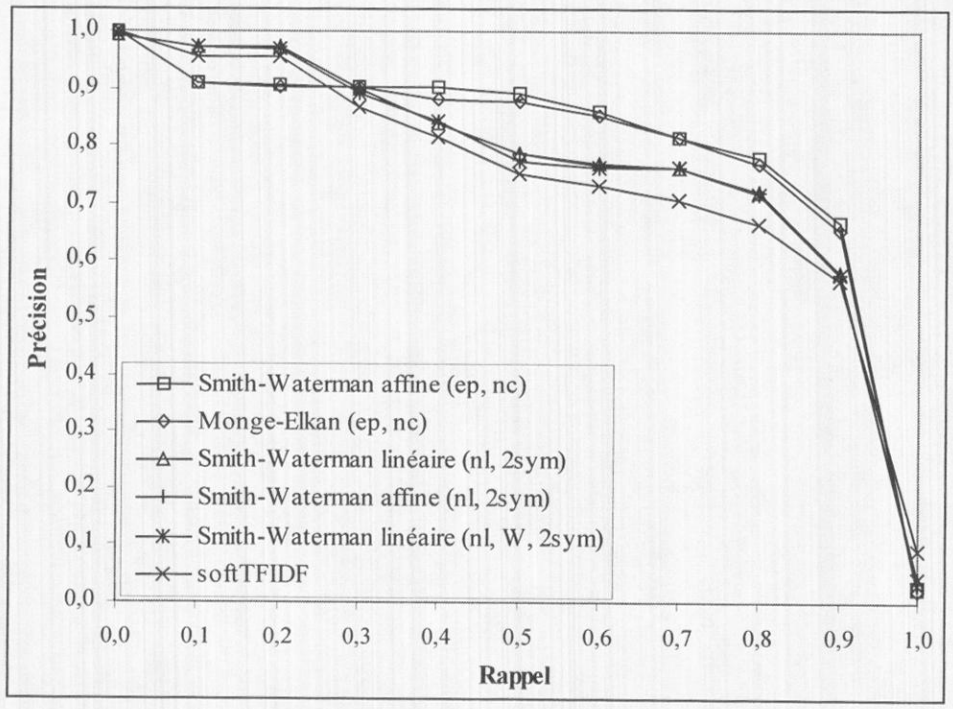


Figure 4 : Courbes moyennes (sur tous les jeux de données) de précision-rappel pour six fonctions de similarité

Le graphique de la figure 4 et le tableau 5 montrent que des modifications apportées à certaines fonctions de similarité dans le cadre de ce mémoire, et qui n'avaient pas été considérées par Cohen, Ravikumar et Fienberg, donnent de bons résultats.

Premièrement, l'élimination des caractères de ponctuation a permis à deux fonctions de similarité basées sur l'analyse des caractères d'être les plus précises sur l'ensemble des jeux de données. D'ailleurs, les fonctions basées sur les caractères présentées dans le tableau 5 sont toutes des fonctions pour lesquelles

ont été éliminés les caractères de ponctuation. L'impact de l'élimination des caractères de ponctuation est examiné plus en détails dans la section 4.2.

Deuxièmement, l'application d'une variante symétrique de la méthode hybride de niveau 2 sur différentes fonctions de similarité basées sur l'analyse des caractères crée plusieurs fonctions qui sont plus précises que la méthode softTFIDF. Inversement, on note qu'aucune des fonctions présentées dans le tableau 5 n'est la résultante de l'application de la version originale, asymétrique, de la méthode de niveau 2. Une étude de la performance de la variante symétrique des fonctions de similarité de niveau 2 est présentée dans la section 4.3.

4.3 Impact de l'élimination des caractères de ponctuation

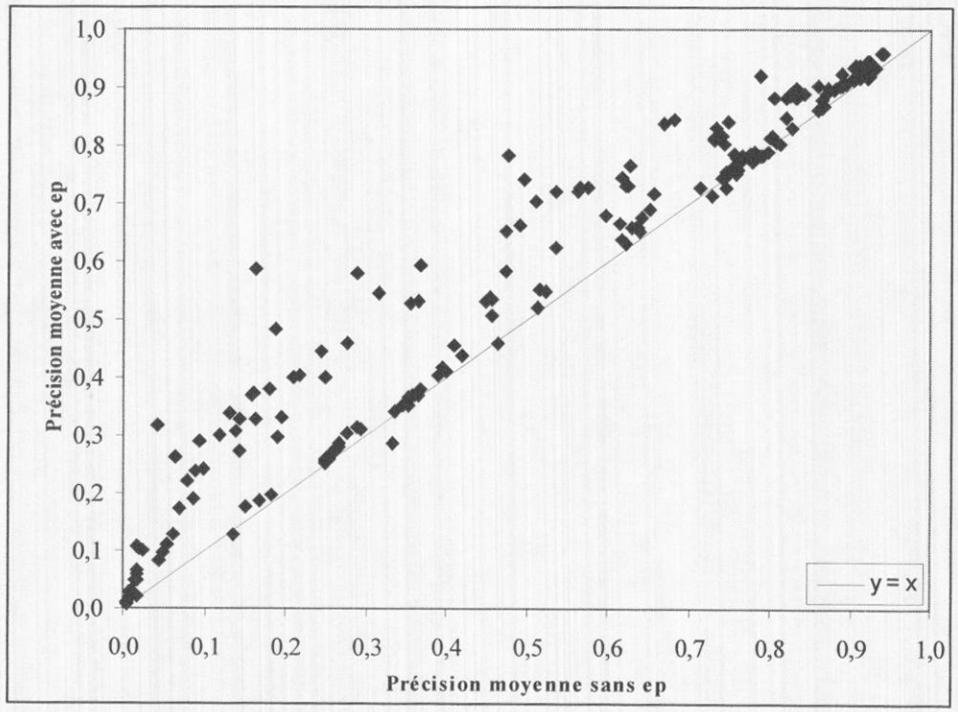
Parmi les 11 fonctions de similarité qui retiennent l'attention (présentées dans le tableau 5), on retrouve quatre fonctions basées sur l'analyse des caractères. Pour ces quatre fonctions, les caractères de ponctuation ont été éliminés. Afin de mieux comprendre l'impact de ce traitement, le tableau 6 présente la performance de ces quatre fonctions avec et sans l'élimination des caractères de ponctuation.

Mis à part le jeu de données *u*, où l'élimination des caractères de ponctuation a un léger impact négatif sur la précision, on constate que ce traitement a un impact positif sur la précision moyenne des quatre fonctions de similarité pour tous les jeux de données. D'un autre côté, on observe qu'en réduisant la longueur des chaînes de caractères à comparer, l'élimination des caractères de ponctuation réduit considérablement le temps d'exécution des fonctions de similarité basées sur l'analyse des caractères. Pour les quatre fonctions présentées dans le tableau 6, la réduction du temps d'exécution moyen se situe entre 33 et 43%.

Fonction de similarité	Rang par jeu de données (précision moyenne)								Moyennes	Temps d'éxécution moyen
	j	o1	o2	o3	o4	p	r	u		
Smith-Waterman affine (nc)	69 (0,25)	13 (0,93)	6 (0,87)	75 (0,74)	38 (0,68)	11 (0,93)	71 (0,84)	41 (0,81)	40,5 (0,75)	85,3 s
Smith-Waterman affine (ep, nc)	63 (0,26)	5 (0,93)	1 (0,87)	59 (0,82)	26 (0,85)	5 (0,93)	52 (0,89)	45 (0,80)	32,0 (0,79)	48,9 s
Monge-Elkan (nc)	70 (0,25)	14 (0,91)	8 (0,86)	71 (0,76)	39 (0,67)	12 (0,93)	73 (0,84)	42 (0,81)	41,1 (0,75)	153,7 s
Monge-Elkan (ep, nc)	64 (0,26)	6 (0,93)	2 (0,87)	64 (0,78)	27 (0,84)	3 (0,93)	47 (0,89)	47 (0,80)	32,5 (0,79)	95,6 s
Monge-Elkan (nc, W)	72 (0,25)	8 (0,92)	7 (0,86)	89 (0,64)	70 (0,51)	9 (0,93)	25 (0,94)	53 (0,79)	41,6 (0,73)	153,2 s
Monge-Elkan (ep, nc, W)	66 (0,26)	4 (0,94)	4 (0,87)	86 (0,66)	37 (0,70)	1 (0,93)	3 (0,96)	60 (0,78)	32,6 (0,76)	96,5 s
Smith-Waterman linéaire (nc, W)	74 (0,25)	9 (0,92)	11 (0,86)	95 (0,62)	104 (0,19)	10 (0,93)	68 (0,86)	78 (0,76)	56,1 (0,67)	42,0 s
Smith-Waterman linéaire (ep, nc, W)	68 (0,26)	1 (0,95)	9 (0,86)	83 (0,66)	94 (0,30)	4 (0,93)	34 (0,91)	84 (0,75)	47,1 (0,70)	27,9 s

Tableau 6 : Performance de quatre fonctions de similarité avec et sans l'élimination des caractères de ponctuation

Les deux graphiques de la figure 5 montrent la précision moyenne et la valeur de la mesure maxF1 des 256 expériences (32 fonctions appliquées sur huit jeux de données) où une fonction de similarité basée sur l'analyse des caractères ou des 3-grams a été évaluée. Sur ces graphiques, un point situé au-dessus de la droite $y = x$ signifie que l'élimination des caractères de ponctuation améliore la performance de la fonction de similarité sur laquelle elle a été appliquée.



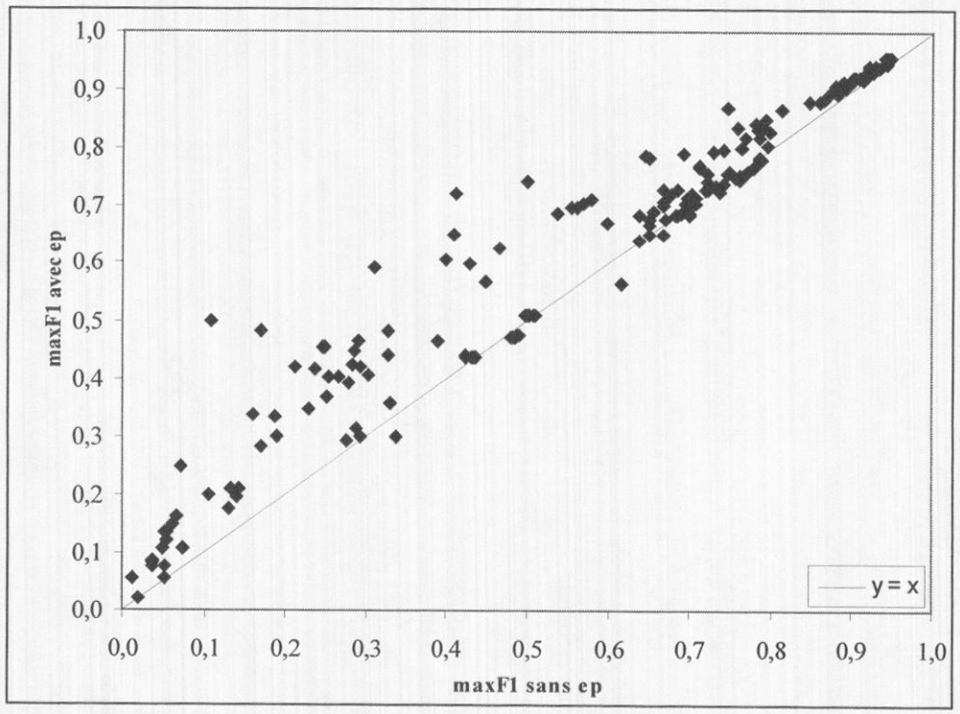


Figure 5 : Précision moyenne et maxF1 des fonctions de similarité avec et sans l'élimination des caractères de ponctuation

Les deux graphiques illustrent qu'on peut tirer avantage de l'élimination des caractères de ponctuation dans presque toutes les situations, c'est-à-dire pour

pratiquement toutes les fonctions de similarité appliquées sur tous les jeux de données considérés dans cette étude. Il s'agit donc d'un traitement qu'on peut appliquer de façon générale sans risque important d'affecter négativement la précision des fonctions de similarité et ce, tout en réduisant considérablement le temps d'exécution de ces dernières.

4.4 Performance de la variante symétrique des fonctions de similarité de niveau 2

Parmi les 11 fonctions de similarité présentées au tableau 5, on retrouve cinq fonctions qui résultent de l'application d'une variante symétrique de la méthode de niveau 2. Afin de mieux évaluer l'apport de cette méthode hybride, le tableau 7 présente les résultats obtenus à partir de ces cinq fonctions avec et sans l'application de méthodes de niveau 2 (asymétrique et symétrique). Pour les cas où aucune méthode de niveau 2 n'a été utilisée, afin que les comparaisons soient plus justes, le tableau 7 présente les résultats obtenus lorsque les caractères de ponctuation ont été éliminés.

Pour quatre des cinq fonctions de similarité présentées dans le tableau 7, l'application de la variante symétrique de la méthode de niveau 2 améliore la précision moyenne dans six jeux de données. De plus, pour ces quatre fonctions, l'utilisation de la variante symétrique procure, en moyenne sur l'ensemble des jeux de données, une précision moyenne nettement plus élevée que si la fonction avait été utilisée telle quelle. Cependant, dans le jeu de données *p*, la variante symétrique réduit quelque peu la précision moyenne des fonctions de similarité du tableau 7 sur lesquelles elle a été appliquée.

Fonction de similarité	Rang par jeu de données (précision moyenne)								Moyennes	Temps d'éxécution moyen
	j	o1	o2	o3	o4	p	r	u		
Smith-Waterman linéaire (ep, nl)	30 (0,38)	64 (0,11)	77 (0,26)	80 (0,73)	96 (0,27)	40 (0,90)	104 (0,68)	51 (0,79)	67,8 (0,51)	29,5 s
Smith-Waterman linéaire (nl, 2)	90 (0,18)	58 (0,13)	23 (0,52)	56 (0,82)	43 (0,59)	83 (0,88)	38 (0,90)	102 (0,73)	61,6 (0,59)	59,1 s
Smith-Waterman linéaire (nl, 2sym)	8 (0,47)	17 (0,76)	33 (0,36)	1 (0,93)	2 (0,97)	55 (0,89)	11 (0,95)	19 (0,89)	18,3 (0,78)	62,8 s
Smith-Waterman affine (ep, nl)	27 (0,38)	66 (0,10)	73 (0,26)	79 (0,73)	75 (0,46)	32 (0,91)	81 (0,81)	38 (0,82)	58,9 (0,56)	47,5 s
Smith-Waterman affine (nl, 2)	91 (0,18)	59 (0,13)	24 (0,52)	56 (0,82)	44 (0,59)	84 (0,88)	37 (0,90)	101 (0,73)	62,0 (0,59)	104,7 s
Smith-Waterman affine (nl, 2sym)	9 (0,47)	18 (0,76)	34 (0,36)	1 (0,93)	1 (0,97)	56 (0,89)	10 (0,95)	20 (0,89)	18,6 (0,78)	109,8s
Smith-Waterman linéaire (ep, nl, W)	33 (0,37)	44 (0,34)	66 (0,27)	104 (0,54)	105 (0,19)	24 (0,92)	50 (0,89)	81 (0,75)	63,4 (0,53)	28,5 s
Smith-Waterman linéaire (nl, W, 2)	86 (0,18)	62 (0,12)	26 (0,52)	39 (0,85)	46 (0,59)	86 (0,88)	44 (0,89)	98 (0,73)	60,9 (0,59)	59,8 s
Smith-Waterman linéaire (nl, W, 2sym)	3 (0,47)	20 (0,72)	36 (0,35)	1 (0,93)	3 (0,97)	44 (0,90)	14 (0,95)	13 (0,89)	16,8 (0,77)	63,4 s
distance d'édition à trois opérations (ep, nl)	38 (0,36)	103 (0,03)	81 (0,26)	124 (0,37)	83 (0,37)	42 (0,90)	99 (0,73)	59 (0,78)	78,6 (0,48)	20,2 s
distance d'édition à trois opérations (nl, 2)	95 (0,17)	73 (0,08)	19 (0,52)	48 (0,84)	52 (0,57)	93 (0,87)	39 (0,90)	112 (0,73)	66,4 (0,58)	37,0 s
distance d'édition à trois opérations (nl, 2sym)	13 (0,46)	27 (0,65)	41 (0,33)	1 (0,93)	12 (0,96)	69 (0,88)	8 (0,95)	17 (0,89)	23,5 (0,76)	38,6 s
Monge-Elkan (ep, nc, W)	66 (0,26)	4 (0,94)	4 (0,87)	86 (0,66)	37 (0,70)	1 (0,93)	3 (0,96)	60 (0,78)	32,6 (0,76)	96,5 s
Monge-Elkan (nc, W, 2)	121 (0,03)	122 (0,01)	123 (0,05)	46 (0,84)	115 (0,13)	122 (0,45)	88 (0,75)	96 (0,73)	104,1 (0,38)	152,2 s
Monge-Elkan (nc, W, 2sym)	82 (0,18)	92 (0,06)	112 (0,12)	1 (0,93)	55 (0,56)	65 (0,89)	61 (0,87)	1 (0,90)	58,6 (0,56)	155,7 s

Tableau 7 : Performance de cinq fonctions de similarité avec et sans l'application de méthodes de niveau 2 (asymétrique et symétrique)

Par ailleurs, l'application de la variante symétrique de la méthode de niveau 2 permet à la fonction de similarité Monge-Elkan normalisée en fonction de la plus courte chaîne de caractères et ajustée par la méthode de Winkler d'être la plus précise dans deux jeux de données. Toutefois, cette même fonction donne de meilleurs résultats dans les autres jeux de données lorsque utilisée telle quelle, sans l'application d'une méthode de niveau 2. Il en est de même pour les deux fonctions les plus précises sur l'ensemble des jeux de données (voir tableau 8).

Fonction de similarité	Rang par jeu de données (précision moyenne)								Moyennes	Temps d'éxécution moyen
	j	o1	o2	o3	o4	p	r	u		
Smith-Waterman affine (ep, nc)	63 (0,26)	5 (0,93)	1 (0,87)	59 (0,82)	26 (0,85)	5 (0,93)	52 (0,89)	45 (0,80)	32,0 (0,79)	48,9 s
Smith-Waterman affine (nc, 2)	125 (0,03)	118 (0,01)	125 (0,05)	54 (0,83)	120 (0,13)	124 (0,45)	95 (0,74)	110 (0,73)	108,9 (0,37)	103,5 s
Smith-Waterman affine (nc, 2sym)	81 (0,18)	82 (0,06)	108 (0,13)	1 (0,93)	62 (0,54)	72 (0,88)	65 (0,87)	6 (0,90)	59,6 (0,56)	110,2 s
Monge-Elkan (ep, nc)	64 (0,26)	6 (0,93)	2 (0,87)	64 (0,78)	27 (0,84)	3 (0,93)	47 (0,89)	47 (0,80)	32,5 (0,79)	95,6 s
Monge-Elkan (nc, 2)	120 (0,03)	121 (0,01)	126 (0,05)	42 (0,84)	116 (0,13)	125 (0,44)	87 (0,76)	99 (0,73)	104,5 (0,38)	153,5 s
Monge-Elkan (nc, 2sym)	83 (0,18)	91 (0,06)	111 (0,13)	1 (0,93)	56 (0,56)	75 (0,88)	59 (0,88)	2 (0,90)	59,8 (0,56)	155,2 s

Tableau 8 : Performance de deux fonctions de similarité avec et sans l'application de méthodes de niveau 2

Bref, les résultats présentés aux tableaux 7 et 8 montrent que la variante symétrique de la méthode de niveau 2 améliore considérablement la précision de diverses fonctions de similarité dans certains cas tout en n'étant d'aucune utilité dans d'autres situations. La figure 6, où sont présentées la précision moyenne et la valeur de la mesure maxF1 de 256 expériences (32 fonctions appliquées sur huit jeux de données) avec et sans l'application de la variante symétrique, donne un aperçu plus vaste de cette réalité.

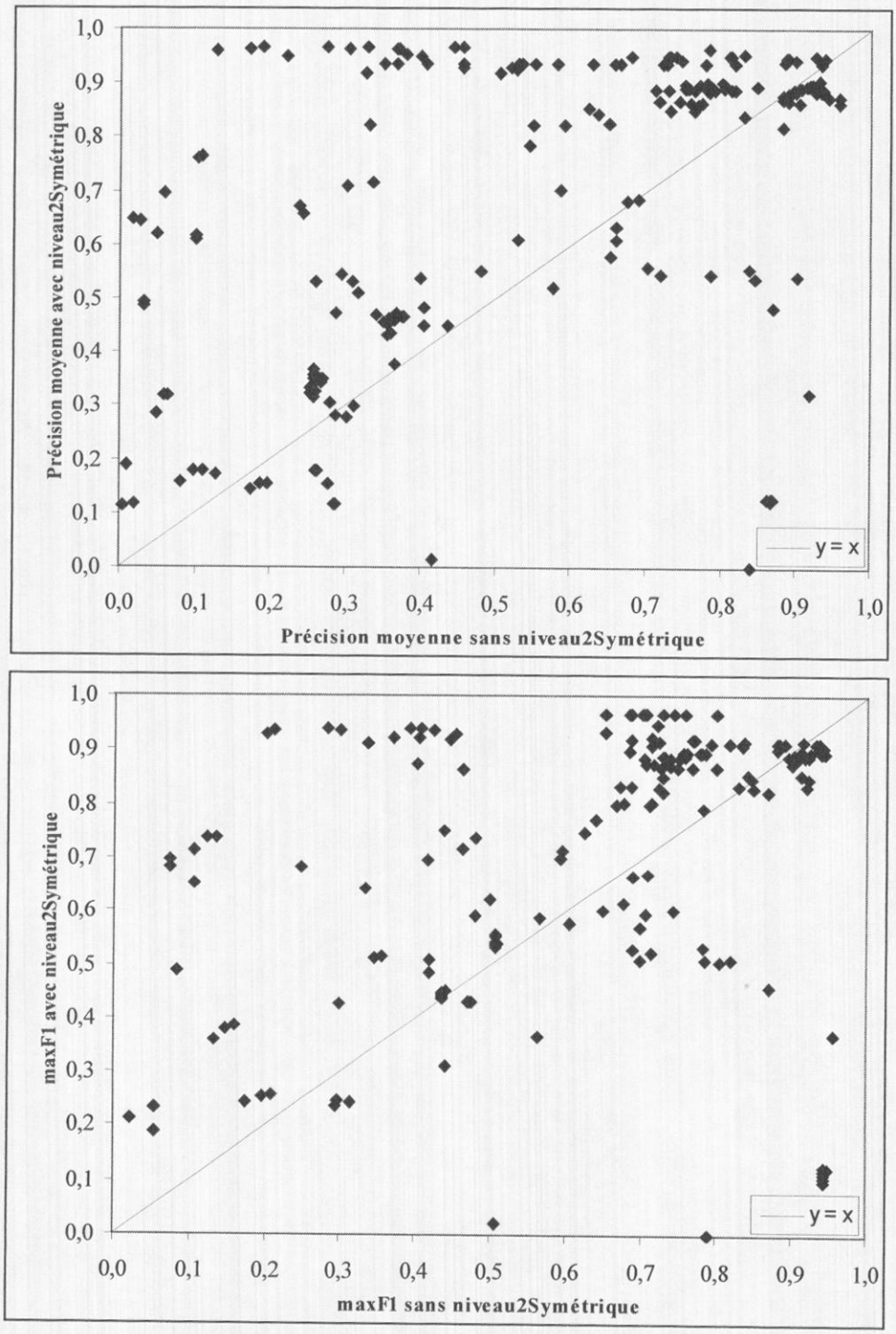
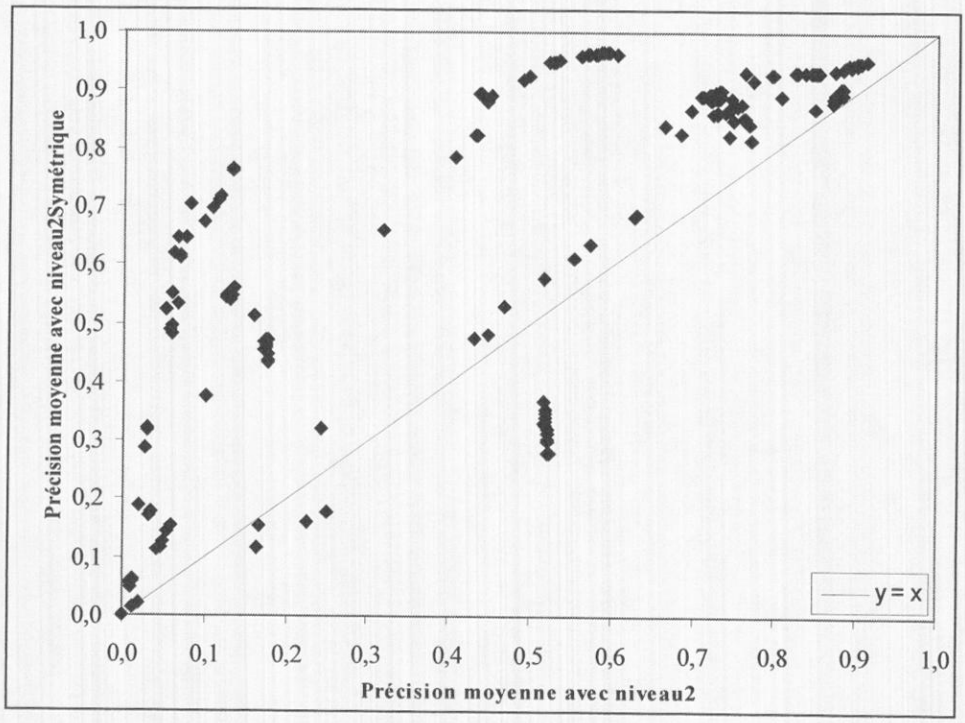


Figure 6 : Précision moyenne et maxF1 des fonctions de similarité avec et sans l'application d'une variante symétrique de la méthode de niveau 2



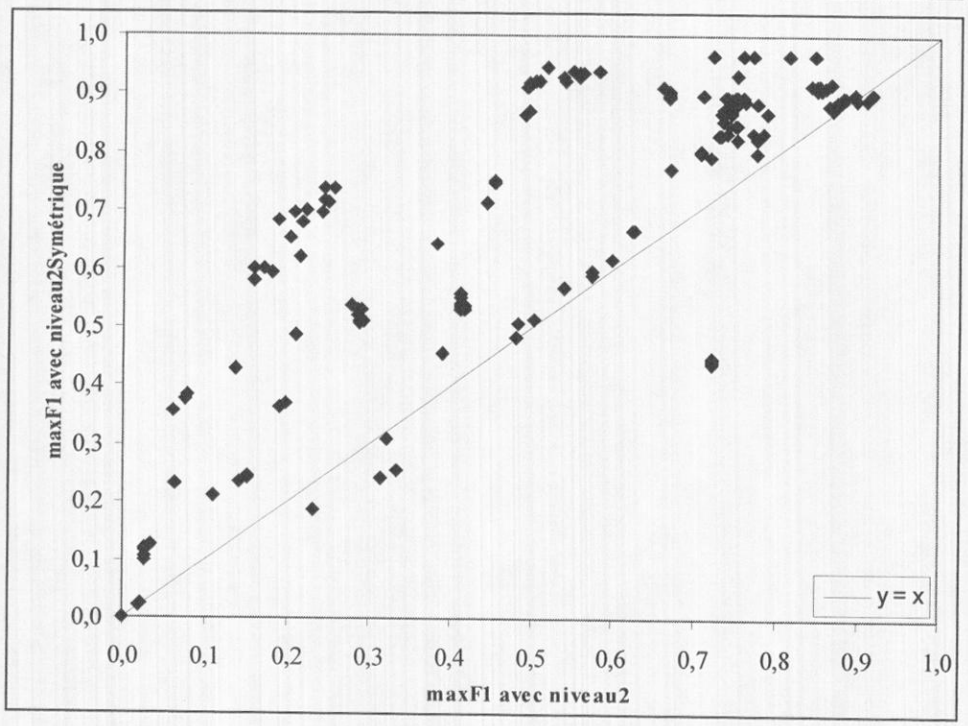


Figure 7 : Précision moyenne et maxF1 des fonctions de similarité avec l'application de méthodes de niveau 2

En ce qui concerne la version originale, asymétrique, de la méthode de niveau 2, les tableaux 7 et 8 ne soulignent aucune situation où elle permet à la fonction de

similarité sur laquelle elle est appliquée d'obtenir des résultats comparables à ceux obtenus à partir des fonctions les plus précises. La figure 7 montre d'ailleurs que la variante symétrique de la méthode de niveau 2 est généralement plus précise que la méthode originale.

Le fait que la version symétrique donne de meilleurs résultats que la version originale ne signifie pas pour autant qu'il ne faut pas considérer la version originale. En fait, la version symétrique proposée dans ce mémoire est elle-même dérivée de la version originale. De plus, elle n'est qu'une façon parmi d'autres de combiner $niveau2(c,d)$ et $niveau2(d,c)$. Si la méthode de niveau 2 devait être utilisée pour générer des variables prédictives dans le but d'entraîner un algorithme d'apprentissage à détecter des doublons approximatifs, il serait probablement mieux dans ce cas d'utiliser la version asymétrique évaluée en double, en interchangeant l'ordre des chaînes de caractères comparées, et de laisser à l'algorithme d'apprentissage la possibilité de trouver la combinaison la plus précise.

5 Conclusions

Plusieurs fonctions de similarité ont été évaluées et comparées dans une tâche de détection de doublons approximatifs, une tâche importante lors de l'intégration de données provenant de différentes sources. Les expérimentations ont été faites sur des jeux de données provenant de différents secteurs d'activité et, bien que l'apport des fonctions de similarité soit satisfaisant dans sept des huit jeux de données, les résultats montrent qu'aucune fonction ne fait partie des plus précises pour tous les jeux de données. Plus généralement, on observe que certains jeux de données sont mieux traités par des fonctions de similarité basées sur l'analyse des caractères alors que dans d'autres jeux de données, ce sont les fonctions basées sur les lexèmes qui sont les plus précises.

Les résultats montrent aussi que des modifications proposées dans ce mémoire améliorent significativement les fonctions de similarité sur lesquelles elles sont appliquées. En moyenne sur l'ensemble des jeux de données traités dans cette étude, ces modifications permettent à plusieurs fonctions de se comparer avantageusement à la fonction softTFIDF, identifiée récemment par Cohen, Ravikumar et Fienberg comme étant la plus précise parmi de nombreuses fonctions.

Premièrement, le fait d'éliminer les caractères de ponctuation dans les chaînes lorsqu'elles sont comparées à l'aide de fonctions de similarité basées sur l'analyse des caractères améliore la précision de ces fonctions dans la plupart des situations tout en réduisant considérablement leur temps d'exécution. D'ailleurs, les deux fonctions les plus précises, en moyenne sur l'ensemble des jeux de données, sont des fonctions où les caractères de ponctuation ont été éliminés.

Deuxièmement, l'application sur des fonctions de similarité d'une variante symétrique de la méthode hybride de niveau 2 procure dans presque tous les cas de meilleurs résultats que l'application de la méthode originale, asymétrique.

Toutefois, contrairement à l'élimination des caractères de ponctuation, l'application de cette variante symétrique a un impact négatif important dans plusieurs situations.

Les fonctions de similarité évaluées dans le cadre de ce projet de recherche sont relativement simples à implanter pour quiconque désirerait détecter automatiquement des doublons approximatifs dans une base de données. Toutefois, lorsqu'on compare deux enregistrements afin de déterminer s'ils représentent la même entité, on doit aussi choisir les éléments d'information (champs) à comparer. De plus, on doit déterminer comment comparer les éléments d'information. On peut le faire comme dans la présente étude, en comparant simplement les chaînes de caractères résultant de la concaténation de tous les éléments d'information, ou encore, en comparant les éléments d'information un à un. Cohen, Ravikumar et Fienberg [9] ont d'ailleurs comparé ces deux approches et ont obtenu de meilleurs résultats avec la seconde. Si on compare les éléments d'information un à un, on ajoute une complexité dans l'élaboration du modèle décisionnel qui servira à déterminer le statut des paires d'enregistrements comparées ; on doit alors prendre en considération l'importance de chaque élément d'information quant à leur utilité à indiquer si deux enregistrements font référence à la même entité. Par ailleurs, pour qu'un tel modèle décisionnel soit appliqué sur des bases de données de tailles comparables à celles qu'on retrouve habituellement dans les organisations, il faut compter sur des techniques de réduction du nombre de doublons potentiels plus sophistiquées que celle utilisée ici. Ces dernières réflexions rappellent que la détection de doublons approximatifs ne se limite pas à l'utilisation de fonctions de similarité et qu'elle est composée d'autres tâches qui nécessitent elles aussi une attention et une expertise particulières.

Enfin, dans le cadre d'un futur projet de recherche, il serait intéressant d'évaluer d'autres façons de rendre symétrique la méthode de niveau 2. Ici, cela a été fait à l'aide de la moyenne de $niveau2(c,d)$ et de $niveau2(d,c)$. Peut-être qu'utiliser le

minimum ou le maximum serait plus approprié. Aussi, pour effectuer une comparaison plus juste de ces méthodes hybrides avec la méthode softTFIDF, il serait judicieux de considérer pour cette dernière méthode plusieurs mesures de similarité internes et différents seuils θ contrairement à la présente étude, où seulement une combinaison, celle proposée par Cohen, Ravikumar et Fienberg, a été utilisée.

6 Bibliographie

[1] ABE, N. et MAMITSUKA, H.. *Query Learning Strategies using Boosting and Bagging*. In Proceedings of the International Conference on Machine Learning, juillet 1998, pp 1-9.

[2] BAXTER, R., CHRISTEN, P. et CHURCHES. T.. *A Comparison of Fast Blocking Methods for Record Linkage*. In Proceedings of the ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Identification, août 2003, pp 25-27.

[3] BOSER, B. E., GUYON, I. M. et VAPNIK, V. N.. *A Training Algorithm for Optimal Margin Classifiers*. In Proceedings of the ACM Workshop on Computational Learning Theory, juillet 1992, pp 144-152.

[4] BREIMAN, L.. *Bagging Predictors*. Machine Learning, 1996, vol. 24, n° 2, pp 123-140.

[5] COHEN, W. W.. *Data Integration Using Similarity Joins and a Word-Based Information Representation Language*. ACM Transactions on Information Systems, 2000, vol. 18, n° 3, pp 288-321.

[6] COHEN, W. W. et RICHMAN, J.. *Learning to Match and Cluster Entity Names*. In Proceedings of the ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval, septembre 2001.

[7] COHEN, W. W. et RICHMAN, J.. *Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration*. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, juillet 2002, pp 475-480.

[8] COHEN, W. W., RAVIKUMAR, P. et FIENBERG, S. E.. *A Comparison of String Distance Metrics for Name-Matching Tasks*. In Proceedings of the IJCAI Workshop on Integration Information on the Web, août 2003, pp 73-78.

[9] COHEN, W. W., RAVIKUMAR, P. et FIENBERG, S. E.. *A Comparison of String Metrics for Matching Names and Records*. In Proceedings of the ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Identification, août 2003, pp 13-18.

[10] COHEN, W. W. et RAVIKUMAR, P.. *SecondString Project Page* **[en ligne]**. Disponible sur : <http://secondstring.sourceforge.net> (consulté le 16.03.2005).

[11] COHN, D., ATLAS, L. et LADNER, R.. *Improving Generalization with Active Learning*. Machine Learning, 1994, vol. 15, n° 2, pp 201-221.

[12] DAMERAU, F. J.. *A Technique for Computer Detection and Correction of Spelling Errors*. Communications of the Association for Computing Machinery, 1964, vol. 7, n° 3, pp 171-176.

[13] DAVIDSON, L.. *Retrieval of Misspelled Names in an Airlines Passenger Record System*. Communications of the Association for Computing Machinery, 1962, vol. 5, n° 3, pp 169-171.

[14] GLANTZ, H. T.. *On the Recognition of Information With a Digital Computer*. Journal of the Association for Computing Machinery, 1957, vol. 4, n° 2, pp 178-188.

[15] GOTOH, O.. *An Improved Algorithm for Matching Sequences*. Journal of Molecular Biology, 1982, vol. 162, pp 705-708.

[16] HERNANDEZ, M. A.. *A Generalization of Band Joins and the Merge/Purge Problem*. Thèse de doctorat. New-York : Columbia University, 1996, 171 p.

[17] HERNANDEZ, M. A. et STOLFO, S. J.. *The Merge/Purge Problem for Large Databases*. In Proceedings of the ACM International Conference on Management of Data, mai 1995, pp 127-138.

[18] HYLTON, J. A.. *Identifying and Merging Related Bibliographic Records*. Mémoire de maîtrise. Cambridge : Massachusetts Institute of Technology, 1996, 99 p.

[19] JACCARD, P.. *The Distribution of the Flora in the Alpine Zone*. New Phytologist, 1912, vol. 11, n° 2, pp 37-50.

[20] JARO, M. A.. UNIMATCH: *A Record Linkage System: User's Manual*. États-Unis : U.S. Bureau of the census, 1976.

[21] KNUTH, D. E.. *The Art of Computer Programming. vol 3. (Sorting and Searching)*. États-Unis: Addison-Wesley Publishing Company, 1973, 723 p.

[22] LEVENSHTEIN, V. I.. *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals*. Soviet Physics – Doklady, 1966, vol. 10, n° 8, pp 707-710.

[23] LOWRANCE, R. et WAGNER, R. A.. *An Extension of the String-to-String Correction Problem*. Journal of the Association for Computing Machinery, 1975, vol. 22, n° 2, pp 177-183.

[24] McCALLUM, A., NIGAM, K. et UNGAR, L. H.. *Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching*. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, août 2000, pp 169-178.

[25] MONGE, A. E. et ELKAN, C. P.. *The Field Matching Problem: Algorithms and Applications*. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, août 1996, pp 267-270.

[26] MONGE, A. E. et ELKAN, C. P.. *An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records*. In Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, mai 1997, pp 23-29.

[27] NEEDLEMAN, S. B. et WUNSCH, C. D.. *A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins*. Journal of Molecular Biology, 1970, vol. 48, pp 443-453.

[28] NIGAM, K., LAFFERTY, J. et McCALLUM, A.. *Using Maximum Entropy for Text Classification*. In Proceedings of the IJCAI Workshop on Machine Learning for Information Filtering, juillet et août 1999, pp 61-67.

[29] ODELL, M. K. et RUSSELL, R. C.. U.S. Patent nos 1 261 167 (1918) et 1 435 663 (1922).

[30] POLLOCK, J. J. et ZAMORA, A.. *Automatic Spelling Correction in Scientific and Scholarly Text*. Communication of the ACM, 1984, vol. 27, n° 4, pp 358-368.

[31] PORTER, E. H. et WINKLER, W. E.. *Approximate String Comparison and its Effect on an Advanced Record Linkage System*. Record Linkage

Techniques : In Proceedings of an International Workshop and Exposition, mars 1997, pp 190-199.

[32] QUINLAN, J. R.. *Improved Use of Continuous Attributes in C4.5*. Journal of Artificial Intelligence Research, 1996, vol. 4, pp. 77-90.

[33] SALTON, G.. *The SMART Retrieval System: Experiments in Automatic Document Processing*. États-Unis: Prentice-Hall, 1971, 556 p.

[34] SALTON, G. et BUCKLEY, C.. *Term-Weighting Approaches in Automatic Text Retrieval*. Information Processing and Management, 1988, vol. 24, n° 5, pp 513-523.

[35] SALTON, G. et YANG, C. S.. *On the Specification of Term Values in Automatic Indexing*. Journal of Documentation, 1973, vol. 29, n° 4, pp 351-372.

[36] SARAWAGI, S. et BHAMIDIPATY, A.. *Interactive Deduplication using Active Learning*. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, juillet 2002, pp 269-278.

[37] SELLERS, P. H.. *On the Theory and Computation of Evolutionary Distances*. SIAM Journal on Applied Mathematics, 1974, vol. 26, n° 4, pp 787-793.

[38] SEUNG, H. S., OPPER, M. et SOMPOLINSKY, H.. *Query by Committee*. In Proceedings of the ACM Workshop on Computational Learning Theory, juillet 1992, pp 287-294.

[39] SMITH, T. F. et WATERMAN, M. S.. *Identification of Common Molecular Subsequences*. Journal of Molecular Biology, 1981, vol. 147, pp 195-197.

[40] UKKONEN, E.. *Finding Approximate Patterns in Strings*. Journal of Algorithms, 1985, vol. 6, pp 132-137.

[41] TEJADA, S., KNOBLOCK, C. A. et MINTON, S.. *Learning Object Identification Rules for Information Integration*. Information Systems, 2001, vol. 26, n° 8, pp 607-633.

[42] WAGNER, R. A. et FISCHER, M. J.. *The String-to-String Correction Problem*. Journal of the Association for Computing Machinery, 1974, vol. 21, n° 1, pp 168-173.

[43] WANG, G., CHEN, H. et ATABAKHSH, H.. *Automatically Detecting Deceptive Criminal Identities*. Communications of the ACM, 2004, vol. 47, n° 3, pp 71-76.

[44] WATERMAN, M. S., SMITH, T. F. et BEYER, W. A.. *Some Biological Sequence Metrics*. Advances in Mathematics, 1976, vol. 20, pp 367-387.

[45] WINKLER, W. E.. *Preprocessing of Lists and String Comparison*. In Proceedings of the Workshop on Exact Matching Methodologies, mai 1985, pp 181-187.

[46] WINKLER, W. E.. *String Comparator and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*. In Proceedings of the Section on Survey Research Methods, American Statistical Association, 1990, pp 354-359.

[47] YANCEY, W. E.. *An Adaptive String Comparator for Record Linkage*. In Proceedings of the Section on Survey Research Methods, American Statistical Association (à paraître).

7 Annexe

Fonction de similarité	Rang Précision moyenne Temps d'éxécution								Moyennes
	j	o1	o2	o3	o4	p	r	u	
Smith-Waterman affine (ep, nc)	63	5	1	59	26	5	52	45	32
	0,26	0,93	0,87	0,82	0,85	0,93	0,89	0,80	0,79
	37,6	1,8	16,8	0,0	8,1	4,3	320,8	1,6	48,9
Monge-Elkan (ep, nc)	64	6	2	64	27	3	47	47	33
	0,26	0,93	0,87	0,78	0,84	0,93	0,89	0,80	0,79
	74,6	3,6	33,9	0,0	15,8	7,7	625,4	3,4	95,6
Smith-Waterman linéaire (nl, 2sym)	8	17	33	1	2	55	11	19	18
	0,47	0,76	0,36	0,93	0,97	0,89	0,95	0,89	0,78
	38,9	1,7	14,6	0,0	7,5	4,2	433,9	1,4	62,8
Smith-Waterman affine (nl, 2sym)	9	18	34	1	1	56	10	20	19
	0,47	0,76	0,36	0,93	0,97	0,89	0,95	0,89	0,78
	67,8	2,8	25,3	0,0	13,1	7,5	759,1	2,7	109,8
Smith-Waterman linéaire (nl, W, 2sym)	3	20	36	1	3	44	14	13	17
	0,47	0,72	0,35	0,93	0,97	0,90	0,95	0,89	0,77
	39,0	1,5	14,6	0,0	7,6	4,4	438,4	1,7	63,4
Smith-Waterman affine (nl, W, 2sym)	4	21	37	1	4	45	13	14	17
	0,47	0,71	0,35	0,93	0,97	0,90	0,95	0,89	0,77
	69,5	2,8	26,1	0,0	13,3	7,4	747,3	2,7	108,6
Monge-Elkan (nl, 2sym)	10	23	38	1	5	59	12	21	21
	0,47	0,70	0,35	0,93	0,97	0,89	0,95	0,89	0,77
	100,0	4,6	37,3	0,0	20,3	10,8	1060,1	4,4	154,7
Smith-Waterman affine (ep, nc, W)	65	2	3	83	36	2	4	57	32
	0,26	0,95	0,87	0,66	0,72	0,93	0,96	0,78	0,77
	36,0	1,7	16,6	0,0	7,9	4,2	303,3	1,7	46,4
Monge-Elkan (nl, W, 2sym)	5	25	39	1	8	51	17	16	20
	0,47	0,67	0,34	0,93	0,96	0,89	0,95	0,89	0,76
	99,5	4,4	37,9	0,0	20,7	11,2	1085,1	4,5	157,9
TFIDF	2	19	40	45	6	31	2	39	23
	0,49	0,76	0,34	0,84	0,96	0,91	0,98	0,82	0,76
	0,1	0,0	0,0	0,0	0,0	0,0	0,7	0,0	0,1
Monge-Elkan (ep, nc, W)	66	4	4	86	37	1	3	60	33
	0,26	0,94	0,87	0,66	0,70	0,93	0,96	0,78	0,76
	75,4	3,6	34,0	0,0	16,0	8,0	632,0	3,5	96,5
Distance d'édition à trois opérations (nl, 2sym)	13	27	41	1	12	69	8	17	24
	0,46	0,65	0,33	0,93	0,96	0,88	0,95	0,89	0,76
	24,6	1,1	8,8	0,0	4,7	2,8	265,7	0,9	38,6
Distance d'édition à quatre opérations (nl, 2sym)	14	28	42	1	11	70	9	18	24
	0,46	0,65	0,33	0,93	0,96	0,88	0,95	0,89	0,76
	27,5	1,2	10,0	0,0	5,4	3,0	296,6	1,3	43,1
Monge-Elkan (nc)	70	14	8	71	39	12	73	42	41
	0,25	0,91	0,86	0,76	0,67	0,93	0,84	0,81	0,75
	114,1	5,6	59,5	0,0	21,4	9,9	1014,2	4,5	153,7
Smith-Waterman affine (nc)	69	13	6	75	38	11	71	41	41
	0,25	0,91	0,87	0,74	0,68	0,93	0,84	0,81	0,75
	56,0	2,8	30,3	0,0	10,9	5,3	574,9	2,2	85,3

Distance d'édition à quatre opérations (nl, W, 2sym)	12	30	48	1	13	61	16	12	24
	0,46	0,62	0,33	0,93	0,96	0,89	0,95	0,89	0,75
	27,9	1,3	10,3	0,0	5,8	3,3	302,6	1,1	44,0
Distance d'édition à trois opérations (nl, W, 2sym)	11	31	47	1	14	60	15	11	24
	0,46	0,61	0,33	0,93	0,96	0,89	0,95	0,89	0,75
	25,1	1,0	9,2	0,0	4,9	2,8	272,3	1,2	39,6
PLSC (nl, 2sym)	16	26	49	1	15	80	19	25	29
	0,45	0,65	0,32	0,93	0,95	0,88	0,94	0,89	0,75
	25,4	1,0	9,0	0,0	4,8	2,9	275,7	1,0	40,0
softTFIDF	1	35	35	30	23	15	1	7	18
	0,52	0,54	0,35	0,87	0,91	0,92	0,98	0,90	0,75
	17,6	0,7	6,3	0,0	3,3	1,7	188,6	0,6	27,3
PLSC (nl, W, 2sym)	15	29	50	1	17	66	20	23	28
	0,46	0,62	0,31	0,93	0,95	0,89	0,94	0,89	0,75
	25,8	1,0	9,4	0,0	5,0	3,0	282,0	1,0	40,9
Hylton (2sym)	29	22	32	1	7	97	46	33	33
	0,38	0,70	0,37	0,93	0,96	0,87	0,89	0,84	0,74
	39,4	1,7	14,4	0,0	7,8	4,5	444,7	1,6	64,3
Yancey (2sym)	17	36	52	1	16	41	6	9	22
	0,45	0,53	0,30	0,93	0,95	0,90	0,95	0,89	0,74
	49,4	2,1	18,1	0,0	9,6	5,5	531,5	2,0	77,3
Yancey (W, 2sym)	18	42	54	1	19	39	7	8	24
	0,45	0,48	0,30	0,93	0,95	0,90	0,95	0,89	0,73
	50,6	2,1	18,1	0,0	9,5	5,5	538,1	2,1	78,2
Smith-Waterman affine (nc, W)	71	7	5	95	64	8	22	50	40
	0,25	0,92	0,87	0,62	0,54	0,93	0,94	0,79	0,73
	55,4	2,7	30,0	0,0	10,7	5,3	483,5	2,2	73,7
Monge-Elkan (nc, W)	72	8	7	89	70	9	25	53	42
	0,25	0,92	0,86	0,64	0,51	0,93	0,94	0,79	0,73
	114,0	5,5	59,3	0,0	21,4	11,1	1010,1	4,5	153,2
Jaro (2sym)	20	34	59	24	20	52	21	15	31
	0,44	0,55	0,28	0,93	0,92	0,89	0,94	0,89	0,73
	21,8	0,9	8,3	0,0	4,3	2,6	237,4	0,9	34,5
Jaro (W, 2sym)	21	37	61	24	22	48	26	10	31
	0,43	0,52	0,28	0,93	0,92	0,89	0,94	0,89	0,72
	22,6	1,0	8,2	0,0	4,2	2,5	242,4	0,9	35,2
Jaccard	28	24	64	28	18	103	5	34	38
	0,38	0,68	0,27	0,90	0,95	0,82	0,95	0,84	0,72
	0,1	0,0	0,0	0,0	0,0	0,0	0,7	0,0	0,1
Distance d'édition à quatre opérations (2sym)	7	39	46	26	9	50	70	28	34
	0,47	0,49	0,33	0,92	0,96	0,89	0,85	0,87	0,72
	28,9	1,3	10,1	0,0	5,5	3,0	292,3	1,0	42,8
Distance d'édition à trois opérations (2sym)	6	40	45	26	10	49	69	27	34
	0,47	0,49	0,33	0,92	0,96	0,89	0,85	0,87	0,72
	24,6	0,9	9,0	0,0	4,8	2,8	263,8	1,1	38,4
Smith-Waterman linéaire (ep, nc)	67	3	10	59	80	13	103	70	51
	0,26	0,94	0,86	0,82	0,40	0,93	0,72	0,77	0,71
	21,2	1,1	9,3	0,0	4,9	2,6	181,2	1,1	27,7
Smith-Waterman linéaire (ep, nc, W)	68	1	9	83	94	4	34	84	47
	0,26	0,95	0,86	0,66	0,30	0,93	0,91	0,75	0,70
	21,2	1,1	9,4	0,0	4,9	2,6	182,7	1,0	27,9
Hylton (ep, n)	23	47	63	29	21	18	72	26	37
	0,42	0,32	0,28	0,88	0,92	0,92	0,84	0,87	0,68
	9,6	0,7	3,9	0,0	2,4	2,4	72,7	0,4	11,5

PLSC (ep, nc)	112	11	57	85	24	22	58	103	59
	0,13	0,92	0,29	0,66	0,90	0,92	0,88	0,73	0,68
	15,0	0,7	6,7	0,0	3,3	1,8	133,3	0,7	20,2
Smith-Waterman linéaire (nc, W)	74	9	11	95	104	10	68	78	56
	0,25	0,92	0,86	0,62	0,19	0,93	0,86	0,76	0,67
	30,7	1,6	15,5	0,0	6,2	3,2	277,6	1,4	42,0
PLSC (nc)	111	15	43	87	25	21	79	89	59
	0,14	0,90	0,33	0,64	0,87	0,92	0,82	0,74	0,67
	21,1	1,2	10,7	0,0	4,2	1,9	205,0	0,8	30,6
Smith-Waterman linéaire (nc)	73	16	12	75	97	23	106	62	58
	0,25	0,90	0,86	0,74	0,25	0,92	0,66	0,78	0,67
	30,6	1,6	15,4	0,0	6,3	3,0	276,8	1,2	41,9
PLSC (nc, W)	110	12	44	93	34	16	29	109	56
	0,14	0,91	0,33	0,62	0,76	0,92	0,92	0,73	0,67
	26,4	1,4	13,8	0,0	5,2	2,5	254,7	1,2	38,2
PLSC (ep, nc, W)	113	10	58	90	31	19	18	121	58
	0,13	0,92	0,29	0,63	0,79	0,92	0,94	0,71	0,67
	19,7	1,0	8,2	0,0	4,1	2,1	166,4	1,0	25,3
Hylton (ep)	36	32	79	78	32	79	84	35	57
	0,37	0,59	0,26	0,73	0,78	0,88	0,80	0,83	0,65
	8,3	0,6	3,5	0,0	2,1	2,2	64,8	0,4	10,2
Hylton (n)	25	98	72	52	30	28	90	29	53
	0,39	0,04	0,26	0,83	0,79	0,91	0,75	0,86	0,61
	10,5	0,8	4,3	0,0	2,5	2,5	83,3	0,5	13,0
Yancey (ep)	19	51	60	101	35	7	76	44	49
	0,44	0,26	0,28	0,55	0,74	0,93	0,83	0,81	0,60
	29,1	1,5	13,0	0,0	6,3	3,3	260,9	1,5	39,4
Jaro (ep)	41	41	56	107	67	25	55	52	56
	0,36	0,48	0,29	0,52	0,53	0,92	0,88	0,79	0,60
	9,1	0,5	4,3	0,0	2,4	1,5	96,1	0,5	14,3
Smith-Waterman linéaire (nl, 2)	90	58	23	56	43	83	38	102	62
	0,18	0,13	0,52	0,82	0,59	0,88	0,90	0,73	0,59
	36,7	1,6	13,9	0,0	7,1	4,1	408,1	1,5	59,1
Smith-Waterman affine (nl, 2)	91	59	24	56	44	84	37	101	62
	0,18	0,13	0,52	0,82	0,59	0,88	0,90	0,73	0,59
	65,7	2,8	24,4	0,0	12,8	7,2	722,4	2,7	104,7
Smith-Waterman linéaire (nl, W, 2)	86	62	26	39	46	86	44	98	61
	0,18	0,12	0,52	0,85	0,59	0,88	0,89	0,73	0,59
	37,3	1,6	13,9	0,0	7,0	4,1	413,2	1,5	59,8
Monge-Elkan (nl, 2)	98	65	25	31	48	85	35	104	61
	0,17	0,11	0,52	0,85	0,59	0,88	0,90	0,73	0,59
	97,1	4,4	36,7	0,0	21,1	10,8	1065,3	4,3	155,0
Smith-Waterman affine (nl, W, 2)	87	63	27	39	47	87	43	97	61
	0,18	0,12	0,52	0,85	0,59	0,88	0,89	0,73	0,59
	66,6	2,7	24,3	0,0	12,6	7,2	744,3	2,5	107,5
Monge-Elkan (nl, W, 2)	94	69	28	31	49	88	42	100	63
	0,17	0,10	0,52	0,85	0,58	0,88	0,90	0,73	0,59
	94,6	4,3	36,0	0,0	19,9	10,7	1049,3	4,2	152,4
distance d'édition à quatre opérations (nl, 2)	96	72	20	48	51	94	40	113	67
	0,17	0,08	0,52	0,84	0,57	0,87	0,90	0,73	0,58
	27,1	1,3	9,5	0,0	5,5	2,9	282,0	1,0	41,2
distance d'édition à trois opérations (nl, 2)	95	73	19	48	52	93	39	112	66
	0,17	0,08	0,52	0,84	0,57	0,87	0,90	0,73	0,58
	23,1	1,1	8,4	0,0	4,6	2,7	255,4	1,1	37,0

Distance d'édition à quatre opérations (nl, W, 2)	93	74	22	37	53	91	49	115	67
	0,17	0,07	0,52	0,85	0,56	0,87	0,89	0,73	0,58
	26,4	1,1	9,7	0,0	5,3	2,9	290,8	1,1	42,2
Distance d'édition à trois opérations (nl, W, 2)	92	75	21	37	54	90	48	114	66
	0,18	0,07	0,52	0,85	0,56	0,87	0,89	0,73	0,58
	24,9	1,0	8,7	0,0	4,7	2,6	262,5	1,2	38,2
Yancey (2)	84	77	15	43	68	78	30	118	64
	0,18	0,07	0,52	0,84	0,53	0,88	0,91	0,72	0,58
	47,0	2,0	17,3	0,0	9,4	5,2	521,4	2,2	75,6
Yancey (W, 2)	85	84	17	36	69	81	36	120	66
	0,18	0,06	0,52	0,85	0,52	0,88	0,90	0,72	0,58
	47,5	2,0	17,5	0,0	9,3	5,4	524,3	2,1	76,0
Jaro (ep, W)	48	33	53	114	89	27	27	79	59
	0,36	0,58	0,30	0,46	0,33	0,91	0,93	0,76	0,58
	9,1	0,5	4,2	0,0	2,4	1,5	96,5	0,5	14,3
PLSC (nl, W, 2)	99	80	16	31	66	95	56	117	70
	0,17	0,06	0,52	0,85	0,53	0,87	0,88	0,72	0,58
	24,3	1,1	8,8	0,0	4,8	2,7	269,9	1,0	39,1
PLSC (nl, 2)	102	76	18	53	65	96	45	119	72
	0,17	0,07	0,52	0,83	0,54	0,87	0,89	0,72	0,58
	23,7	1,1	8,7	0,0	4,7	2,6	260,9	0,9	37,8
Yancey (ep, W)	24	43	51	122	81	6	28	63	52
	0,40	0,40	0,31	0,41	0,40	0,93	0,93	0,78	0,57
	35,9	1,7	16,3	0,0	7,9	4,0	323,7	1,7	48,9
Jaro (2)	88	89	13	63	71	89	57	123	74
	0,18	0,06	0,52	0,79	0,50	0,87	0,88	0,71	0,56
	20,5	0,9	7,6	0,0	4,1	2,3	220,9	0,9	32,2
Monge-Elkan (nc, W, 2sym)	82	92	112	1	55	65	61	1	59
	0,18	0,06	0,12	0,93	0,56	0,89	0,87	0,90	0,56
	101,0	4,5	37,3	0,0	20,5	10,8	1067,3	4,2	155,7
Monge-Elkan (nc, 2sym)	83	91	111	1	56	75	59	2	60
	0,18	0,06	0,13	0,93	0,56	0,88	0,88	0,90	0,56
	100,0	4,6	37,2	0,0	20,4	10,8	1064,2	4,4	155,2
Jaro (W, 2)	89	93	14	62	73	92	62	122	76
	0,18	0,05	0,52	0,80	0,49	0,87	0,87	0,71	0,56
	20,8	0,9	7,8	0,0	4,2	2,4	231,7	0,8	33,6
Smith-Waterman linéaire (nc, W, 2sym)	78	83	109	1	59	57	66	3	57
	0,18	0,06	0,13	0,93	0,54	0,89	0,86	0,90	0,56
	39,3	1,7	14,5	0,0	7,5	4,5	438,2	1,7	63,4
Smith-Waterman affine (nc, W, 2sym)	79	85	110	1	60	58	67	4	58
	0,18	0,06	0,13	0,93	0,54	0,89	0,86	0,90	0,56
	68,1	2,8	25,8	0,0	13,1	7,6	759,5	2,7	109,9
Smith-Waterman linéaire (nc, 2sym)	80	81	107	1	63	71	64	5	59
	0,18	0,06	0,13	0,93	0,54	0,88	0,87	0,90	0,56
	38,7	1,5	14,6	0,0	7,6	4,3	436,9	1,7	63,2
Smith-Waterman linéaire (2sym)	106	45	101	34	28	107	114	31	71
	0,15	0,32	0,18	0,85	0,82	0,69	0,61	0,86	0,56
	42,8	1,7	14,6	0,0	7,5	4,4	432,3	1,5	63,1
Smith-Waterman affine (nc, 2sym)	81	82	108	1	62	72	65	6	60
	0,18	0,06	0,13	0,93	0,54	0,88	0,87	0,90	0,56
	67,8	2,8	25,3	0,0	13,0	7,6	762,5	2,7	110,2
Smith-Waterman affine (2sym)	107	46	102	34	29	108	113	30	71
	0,15	0,32	0,18	0,85	0,82	0,68	0,61	0,86	0,56
	70,1	3,0	26,2	0,0	13,6	7,8	788,7	2,7	114,0

PLSC (nc, 2sym)	104	96	113	1	61	53	60	22	64
	0,17	0,05	0,12	0,93	0,54	0,89	0,88	0,89	0,56
	25,3	1,2	9,1	0,0	4,9	2,8	272,2	0,9	39,5
PLSC (nc, W, 2sym)	103	97	114	1	58	47	63	24	63
	0,17	0,05	0,12	0,93	0,55	0,90	0,87	0,89	0,56
	25,7	1,1	9,4	0,0	5,1	2,9	281,5	1,0	40,8
distance d'édition à trois opérations (2)	100	90	30	65	42	73	85	125	76
	0,17	0,06	0,52	0,77	0,61	0,88	0,76	0,70	0,56
	23,0	0,9	8,4	0,0	4,7	2,6	244,0	1,0	35,6
distance d'édition à quatre opérations (2)	101	88	31	65	41	74	86	124	76
	0,17	0,06	0,52	0,77	0,61	0,88	0,76	0,70	0,56
	25,4	1,1	9,9	0,0	5,5	3,1	279,5	1,3	40,7
Smith-Waterman affine (ep, nl)	27	66	73	79	75	32	81	38	59
	0,38	0,10	0,26	0,73	0,46	0,91	0,81	0,82	0,56
	38,6	1,9	18,4	0,0	7,9	4,3	307,2	1,7	47,5
Smith-Waterman affine (ep, nl, W)	32	48	65	103	88	20	23	54	54
	0,37	0,30	0,27	0,54	0,33	0,92	0,94	0,79	0,56
	36,1	1,8	16,6	0,0	8,0	4,2	301,5	1,8	46,3
Monge-Elkan (ep, nl)	31	87	76	81	76	38	80	40	64
	0,38	0,06	0,26	0,72	0,44	0,90	0,81	0,81	0,55
	76,0	3,6	34,8	0,0	16,2	8,0	641,1	3,4	97,9
Hylton	52	56	95	82	74	99	93	36	73
	0,35	0,16	0,25	0,71	0,48	0,86	0,74	0,83	0,55
	9,3	0,6	3,7	0,0	2,2	2,2	71,7	0,4	11,3
Hylton (2)	115	71	29	70	50	100	82	127	81
	0,10	0,08	0,52	0,76	0,58	0,85	0,81	0,66	0,54
	36,1	1,5	13,7	0,0	7,5	4,3	416,3	1,5	60,1
Monge-Elkan (ep, nl, W)	34	52	67	105	93	26	24	56	57
	0,37	0,24	0,27	0,53	0,31	0,92	0,94	0,78	0,54
	75,0	3,6	33,6	0,0	15,6	7,5	617,6	3,3	94,5
Monge-Elkan (2sym)	109	50	105	47	33	113	116	32	76
	0,14	0,29	0,16	0,84	0,78	0,63	0,58	0,86	0,54
	101,5	4,6	39,8	0,0	21,0	11,1	1103,2	4,6	160,7
Smith-Waterman linéaire (ep, nl, W)	33	44	66	104	105	24	50	81	63
	0,37	0,34	0,27	0,54	0,19	0,92	0,89	0,75	0,53
	21,3	1,1	9,3	0,0	4,9	2,7	187,7	1,2	28,5
Yancey	22	78	68	108	72	17	98	43	63
	0,42	0,06	0,27	0,52	0,50	0,92	0,73	0,81	0,53
	42,1	2,2	21,7	0,0	8,3	4,1	408,7	1,9	61,1
Jaro	51	55	69	109	86	33	83	49	67
	0,35	0,19	0,27	0,51	0,35	0,91	0,81	0,80	0,52
	12,3	0,7	6,3	0,0	2,9	1,6	138,5	0,5	20,4
Smith-Waterman linéaire (ep, nl)	30	64	77	80	96	40	104	51	68
	0,38	0,11	0,26	0,73	0,27	0,90	0,68	0,79	0,51
	21,4	1,2	10,6	0,0	5,2	2,7	194,0	1,1	29,5
Jaro (W)	57	49	62	113	113	36	32	75	67
	0,35	0,29	0,28	0,46	0,14	0,90	0,91	0,76	0,51
	12,0	0,8	6,2	0,0	2,9	1,7	139,0	0,5	20,4
Yancey (W)	26	53	55	123	102	14	51	64	61
	0,39	0,22	0,29	0,40	0,21	0,92	0,89	0,78	0,51
	52,8	2,7	27,5	0,0	10,5	4,9	507,5	2,2	76,0
PLSC (ep, nl)	47	113	85	97	82	64	91	55	79
	0,36	0,02	0,25	0,58	0,38	0,89	0,75	0,78	0,50
	14,9	0,8	6,7	0,0	3,2	1,8	133,0	0,7	20,1

Smith-Waterman affine (nl, W)	42	61	75	119	109	29	31	65	66
	0,36	0,12	0,26	0,45	0,16	0,91	0,91	0,78	0,49
	55,8	2,8	30,0	0,0	10,5	5,2	485,4	2,2	74,0
Smith-Waterman affine (nl)	35	109	87	99	95	54	100	46	78
	0,37	0,02	0,25	0,57	0,28	0,89	0,73	0,80	0,49
	58,8	2,9	30,2	0,0	10,4	5,3	485,2	2,2	74,4
PLSC (ep, nl, W)	53	95	80	117	101	46	41	74	76
	0,35	0,05	0,26	0,46	0,22	0,90	0,90	0,76	0,49
	18,2	1,0	8,3	0,0	4,1	2,1	165,0	0,8	24,9
Monge-Elkan (nl, W)	45	70	78	120	114	37	33	67	71
	0,36	0,09	0,26	0,45	0,14	0,90	0,91	0,77	0,48
	113,6	5,5	58,5	0,0	21,1	9,7	991,3	4,5	150,5
Monge-Elkan (nl)	37	114	91	100	98	67	102	48	82
	0,37	0,02	0,25	0,56	0,24	0,89	0,73	0,80	0,48
	116,4	5,7	60,8	0,0	22,0	10,1	1042,8	4,6	157,8
Distance d'édition à trois opérations (ep, nl)	38	103	81	124	83	42	99	59	79
	0,36	0,03	0,26	0,37	0,37	0,90	0,73	0,78	0,48
	15,0	0,7	6,8	0,0	3,4	1,8	133,4	0,7	20,2
Distance d'édition à quatre opérations (ep, nl)	39	104	82	124	84	43	101	58	79
	0,36	0,03	0,26	0,37	0,37	0,90	0,73	0,78	0,47
	18,9	0,9	8,5	0,0	4,2	2,1	166,0	0,9	25,2
Smith-Waterman linéaire (nl, W)	46	60	74	118	126	30	74	88	77
	0,36	0,13	0,26	0,46	0,09	0,91	0,83	0,75	0,47
	30,7	1,7	16,0	0,0	6,3	3,0	285,5	1,2	43,1
Smith-Waterman affine (ep)	76	86	117	77	45	109	105	68	85
	0,19	0,06	0,10	0,73	0,59	0,68	0,66	0,77	0,47
	36,3	1,8	17,4	0,0	8,2	4,4	319,4	1,8	48,7
Distance d'édition à trois opérations (ep, nl, W)	43	67	70	130	121	34	53	83	75
	0,36	0,10	0,26	0,35	0,13	0,91	0,89	0,75	0,47
	18,1	1,0	8,1	0,0	3,9	2,1	162,0	0,9	24,5
Distance d'édition à quatre opérations (ep, nl, W)	44	68	71	130	122	35	54	82	76
	0,36	0,10	0,26	0,35	0,13	0,91	0,89	0,75	0,47
	19,0	0,9	8,6	0,0	4,2	2,1	168,5	0,8	25,5
PLSC (2sym)	114	54	115	58	40	118	125	37	83
	0,11	0,19	0,12	0,82	0,66	0,53	0,47	0,83	0,47
	25,1	1,0	9,2	0,0	4,8	2,8	269,4	1,1	39,2
Smith-Waterman linéaire (nl)	40	110	86	98	112	68	115	61	86
	0,36	0,02	0,25	0,58	0,14	0,89	0,60	0,78	0,45
	30,7	1,5	16,5	0,0	6,7	3,2	296,9	1,5	44,6
PLSC (nl, W)	58	115	88	121	127	82	75	80	93
	0,35	0,02	0,25	0,41	0,08	0,88	0,83	0,76	0,45
	26,4	1,4	13,8	0,0	5,3	2,4	254,8	1,2	38,2
Monge-Elkan (ep)	97	94	118	88	57	110	107	71	93
	0,17	0,05	0,08	0,64	0,55	0,66	0,65	0,77	0,45
	76,1	3,6	34,8	0,0	16,1	7,8	638,9	3,5	97,6
Distance d'édition à trois opérations (ep)	59	99	89	110	107	101	108	92	96
	0,34	0,03	0,25	0,51	0,17	0,85	0,63	0,74	0,44
	14,5	0,8	6,6	0,0	3,2	1,8	133,3	1,0	20,1
Distance d'édition à quatre opérations (ep)	60	100	90	110	108	102	109	91	96
	0,34	0,03	0,25	0,51	0,17	0,85	0,62	0,74	0,44
	19,7	1,1	8,8	0,0	4,0	2,1	174,9	0,9	26,4
PLSC (nl)	56	130	99	112	106	98	112	72	98
	0,35	0,01	0,25	0,47	0,18	0,87	0,62	0,77	0,44
	21,1	1,2	10,9	0,0	4,2	2,0	205,2	0,8	30,7

Distance d'édition à trois opérations (nl, W)	54	106	83	128	130	62	77	86	91
	0,35	0,03	0,26	0,35	0,06	0,89	0,83	0,75	0,44
	26,0	1,4	13,7	0,0	5,1	2,4	252,4	1,0	37,7
Distance d'édition à quatre opérations (nl, W)	55	107	84	128	131	63	78	87	92
	0,35	0,03	0,26	0,35	0,06	0,89	0,82	0,75	0,44
	26,9	1,5	14,3	0,0	5,3	2,5	260,1	1,2	39,0
Smith-Waterman linéaire (ep)	75	79	116	69	87	106	121	77	91
	0,20	0,06	0,11	0,77	0,33	0,69	0,53	0,76	0,43
	21,0	1,1	9,3	0,0	4,8	2,6	178,6	1,1	27,3
Smith-Waterman linéaire (2)	116	101	94	72	77	114	118	108	100
	0,06	0,03	0,25	0,75	0,43	0,63	0,55	0,73	0,43
	39,1	1,5	13,6	0,0	7,0	4,1	402,1	1,4	58,6
Smith-Waterman affine (2)	117	102	98	72	78	116	117	107	101
	0,06	0,03	0,25	0,75	0,43	0,63	0,55	0,73	0,43
	71,9	2,7	24,1	0,0	12,6	7,2	718,5	2,6	105,0
Distance d'édition à trois opérations (nl)	49	128	92	126	110	76	110	66	95
	0,36	0,01	0,25	0,37	0,16	0,88	0,62	0,77	0,43
	24,3	1,0	11,3	0,0	4,3	2,0	201,7	1,0	30,7
Distance d'édition à quatre opérations (nl)	50	129	93	126	111	77	111	69	96
	0,36	0,01	0,25	0,37	0,16	0,88	0,62	0,77	0,43
	27,3	1,3	14,5	0,0	5,4	2,5	263,1	1,2	39,4
Monge-Elkan (2)	118	105	100	68	79	117	122	116	103
	0,05	0,03	0,23	0,77	0,41	0,57	0,52	0,72	0,41
	94,1	4,2	35,9	0,0	20,2	10,6	1012,6	4,1	147,7
Distance d'édition à trois opérations	61	126	96	115	128	104	119	94	105
	0,33	0,01	0,25	0,46	0,07	0,82	0,54	0,74	0,40
	23,3	1,2	10,9	0,0	4,2	1,9	202,8	1,6	30,7
Distance d'édition à quatre opérations	62	127	97	115	129	105	120	95	106
	0,33	0,01	0,25	0,46	0,07	0,82	0,54	0,74	0,40
	28,0	1,5	14,2	0,0	5,3	2,6	271,0	1,2	40,5
Smith-Waterman affine	105	112	120	92	85	112	123	73	103
	0,17	0,02	0,05	0,63	0,37	0,64	0,49	0,76	0,39
	55,2	2,9	30,9	0,0	10,7	5,4	504,0	2,3	76,4
Monge-Elkan	108	116	129	94	92	115	124	76	107
	0,15	0,01	0,05	0,62	0,32	0,63	0,47	0,76	0,38
	112,4	5,4	57,6	0,0	20,8	9,7	982,9	4,4	149,2
Monge-Elkan (nc, 2)	120	121	126	42	116	125	87	99	105
	0,03	0,01	0,05	0,84	0,13	0,44	0,76	0,73	0,38
	95,1	4,4	36,2	0,0	20,4	10,7	1056,8	4,2	153,5
Monge-Elkan (nc, W, 2)	121	122	123	46	115	122	88	96	104
	0,03	0,01	0,05	0,84	0,13	0,45	0,75	0,73	0,38
	96,0	4,4	36,0	0,0	20,0	10,6	1046,6	4,2	152,2
PLSC (nc, 2)	127	124	128	41	124	127	89	90	106
	0,03	0,01	0,05	0,85	0,12	0,44	0,75	0,74	0,37
	23,7	1,0	8,7	0,0	4,7	2,8	260,0	1,0	37,7
PLSC (nc, W, 2)	126	125	127	44	123	126	92	93	107
	0,03	0,01	0,05	0,84	0,13	0,44	0,74	0,74	0,37
	24,3	1,1	8,8	0,0	4,6	2,7	267,5	1,2	38,8
Smith-Waterman affine (nc, W, 2)	123	120	122	50	118	121	97	105	107
	0,03	0,01	0,05	0,84	0,13	0,45	0,74	0,73	0,37
	66,3	2,8	24,4	0,0	12,8	7,3	747,1	2,7	107,9
Smith-Waterman linéaire (nc, W, 2)	122	119	121	50	117	120	96	106	106
	0,03	0,01	0,05	0,84	0,13	0,45	0,74	0,73	0,37
	37,2	1,6	14,0	0,0	7,0	4,1	414,5	1,5	60,0

Smith-Waterman linéaire (nc, 2)	124	117	124	54	119	123	94	111	108
	0,03	0,01	0,05	0,83	0,13	0,45	0,74	0,73	0,37
	37,9	1,6	13,7	0,0	7,0	4,0	409,7	1,5	59,4
Smith-Waterman affine (nc, 2)	125	118	125	54	120	124	95	110	109
	0,03	0,01	0,05	0,83	0,13	0,45	0,74	0,73	0,37
	64,7	2,7	24,5	0,0	12,8	7,5	713,2	2,6	103,5
PLSC (2)	119	108	104	74	91	119	126	126	108
	0,04	0,02	0,16	0,74	0,32	0,47	0,43	0,68	0,36
	23,8	0,9	8,7	0,0	4,6	2,7	254,3	0,9	37,0
Smith-Waterman linéaire	77	111	119	91	103	111	127	85	103
	0,18	0,02	0,05	0,63	0,20	0,65	0,36	0,75	0,36
	39,2	1,6	15,2	0,0	6,1	3,0	277,1	1,2	42,9
Hylton (n, 2sym)	129	38	106	61	90	131	131	130	102
	0,01	0,51	0,15	0,82	0,32	0,02	0,00	0,48	0,29
	46,5	2,0	17,1	0,0	9,3	5,3	529,3	1,7	76,4
PLSC (ep)	130	123	130	102	100	128	128	128	121
	0,01	0,01	0,02	0,55	0,24	0,31	0,29	0,65	0,26
	15,2	0,7	6,7	0,0	3,3	1,8	132,3	0,7	20,1
Hylton (n, 2)	128	57	103	67	99	130	130	131	106
	0,01	0,16	0,17	0,77	0,24	0,02	0,00	0,45	0,23
	42,8	1,9	16,1	0,0	8,9	5,0	492,6	1,8	71,1
PLSC	131	131	131	106	125	129	129	129	126
	0,01	0,00	0,02	0,52	0,10	0,29	0,09	0,64	0,21
	21,0	1,1	10,9	0,0	4,2	2,0	203,9	0,8	30,5