REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 11, número 2
Págs. 7 - 23

Detección de Duplicados: Una Guía Metodológica

Iván Amón * Claudia Jiménez †
Fecha de Recibido: 21/05/2010 Fecha de Aprobación: 26/10/2010

Resumen

Cuando una misma entidad del mundo real se almacena más de una vez, a través de una o varias bases de datos, en tuplas con igual estructura pero sin un identificador único y éstas presentan diferencias en sus valores, se presenta el fenómeno conocido como *detección de duplicados*. Para esta tarea, se han desarrollado múltiples *funciones de similitud* las cuales detectan las cadenas de texto que son similares mas no idénticas. En este artículo se propone una guía metodológica para seleccionar entre nueve de estas *funciones de similitud* (Levenshtein, Brecha Afín, Smith-Waterman, Jaro, Jaro-Winkler, *Bi-grams*, *Tri-grams*, Monge-Elkan y SoftTF-IDF) la más adecuada para un caso específico o situación particular, de acuerdo con la naturaleza de los datos que se estén analizando.

Palabras clave: *Limpieza de datos, Preprocesamiento de datos, Calidad de datos, Detección de duplicados, Funciones de similitud.*

Abstract

When the same real world entity is stored more than once, through one or more databases in tuples with the same structure but without a unique identifier and these differ in their values, introduces the phenomenon known as detection duplicates. Researchers have developed multiple similarity functions to detect the strings similars but not identical. This paper proposes a methodological guide to select from nine of these similarity functions (Levenshtein, Af fine Gap, Smith-Waterman, Jaro, Jaro-Winkler, Bi-grams, Tri-grams, Monge-Elkan and SofTF-IDF) the more suited to a specific case or situation, according to the nature of the data being analyzed.

Keywords*: Data cleaning, data preprocessing, Data Quality, Record Linkage, Similarity functions.*

* Universidad Pontifica Bolivariana, Circular 1 N° 70 – 01Medellín-Colombia, ivan.amon@upb.edu.co
† Universidad Nacional de Colombia, Circular 59ª N° 63-20 Medellín-Colombia, csjimene@unal.edu.co

1 Introducción

Las organizaciones almacenan grandes cantidades de datos que utilizan para tomar las decisiones relacionadas con su funcionamiento. Sin embargo, con frecuencia los datos son almacenados con valores errados y con inconsistencias que pueden conducir a falsas inferencias o a decisiones equivocadas y éstas a su vez pueden ocasionar pérdidas de tiempo, dinero y credibilidad.

Uno de los posibles conflictos que pueden presentar los datos, se da cuando una misma entidad del mundo real se almacena dos o más veces (duplicada) a través de una o varias bases de datos, en tuplas con igual estructura, que no comparten un identificador único (por ejemplo el número de cédula de ciudadanía) y presentan diferencias textuales en sus valores.

Para ilustrar esta situación, se toma como ejemplo la base de datos ScienTI, donde se registra la actividad investigativa de Colombia. En ella, cada investigador actualiza el programa CvLAC (*Curriculum vitae Latinoamericano y del Caribe*) mediante el registro de los proyectos de investigación en los que participa. En este contexto, es fácil que se presente el conflicto de la detección de duplicados ya que en un proyecto (i) puede participar más de un investigador; (ii) cada investigador ingresa al sistema sus datos independientemente; (iii) los proyectos no cuentan con una identificación única, por tanto puede suceder que el nombre de un mismo proyecto no sea escrito en forma idéntica por parte de todos sus integrantes (por ejemplo: *Detección de Duplicados: Una Guía Metodológica* vs *Detección de Duplicados - Una Guía Metodológica)*. Si no se tomaran las medidas pertinentes, la contabilización de la cantidad de proyectos realizados se sobredimensionaría distorsionando la realidad.

Entre las causas que pueden ocasionar diferentes representaciones de una misma entidad se encuentran: restricciones de formato, de longitud y/o en el conjunto de caracteres permitidos, errores humanos al capturar los datos, errores que surgen integrando bases de datos diferentes o haciendo migración entre sistemas, modelos de datos mal diseñados, entre otras.

El proceso que detecta este conflicto se conoce con diversos nombres: *record linkage* o *record matching* entre la comunidad estadística; *database hardening* en el mundo de la Inteligencia Artificial; *merge-purge*, *data deduplication* o *instance identification* en el mundo de las bases de datos; otros nombres como *coreference resolution* y *duplicate record detection* también son usados. En este artículo se utilizará el término genérico *detección de duplicados*.

La historia del *record linkage,* comenzó en 1946 cuando Dunn esbozó la problemática [11]. El *record linkage* computarizado fue planteado primero por

el genetista Canadiense Howard Newcombe y sus colaboradores en 1959 [29], siendo formalizado por Fellegi y Sunter en 1969 como una regla de decisión probabilista [14]. Winkler, autor prolífico en el tema, ha propuesto algunas mejoras de este modelo [38, 39, 40, 41]. En forma simple, dicho proceso puede plantearse como sigue: dado un conjunto. 1) Se define un umbral real θ] [0.1] 2) Se compara cada registro de *R* con el resto; 3) Si la *similitud* entre una pareja de tuplas es mayor o igual que θ , se asumen duplicados; es decir, se consideran representaciones de una misma entidad real.

Una *función de similitud* devuelve un número real en el intervalo [0,1] igual a uno si ambas tuplas son idénticas y menor entre más diferentes sean. Ya que una tupla está compuesta por atributos, es necesaria una función de similitud al nivel de atributo, siendo frecuente tratar los atributos como cadenas de texto y concatenarlos formando una única cadena. Dichas funciones vienen siendo investigadas por décadas. Elmagarmid *et. al.* clasifican las funciones propuestas en dos categorías: basadas en caracteres y basadas en *tokens* o palabras [13]. Las primeras consideran cada cadena como una secuencia ininterrumpida de caracteres. Las segundas como un conjunto de subcadenas delimitadas por caracteres especiales como espacios en blanco, comas y puntos, es decir, consideran una cadena como un conjunto de *tokens* y calculan la similitud entre cada pareja de *tokens* mediante alguna de las funciones que se basan en caracteres.

Por todo lo anterior, Elmagarmid *et al.* [13] concluyen que la gran cantidad de formas en que el valor de un mismo atributo puede aparecer representado convierten la elección de la función de similitud en todo un problema que requiere aún mayor investigación. Se han realizado algunos estudios comparativos [6, 7, 10, 17, 26, 42] pero ninguno de ellos compara la eficacia de las técnicas basándose en distintas situaciones problemáticas como las planteadas en el presente trabajo (introducción de errores ortográficos, uso de abreviaturas, palabras faltantes, introducción de prefijos/sufijos sin valor semántico, reordenamiento de palabras y eliminación/adición de espacios en blanco) ni propone una guía metodológica para ayudar en la selección de las técnicas más adecuadas para una situación particular de acuerdo con la naturaleza de los datos.

La guía metodológica en cuestión, principal aporte de este artículo, había sido planteada en [1]. Su construcción se basó en los resultados arrojados por un estudio comparativo propio, para el cual se diseñó un experimento en el que se construyeron diez conjuntos de datos, cada uno con doscientas parejas de datos, diseñados especialmente para cada una de las seis situaciones problemáticas estudiadas. Las nueve funciones de similitud motivo de comparación en este trabajo (Levenshtein, Brecha Afín, Smith-Waterman, Jaro, Jaro-Winkler, *Bi-grams*, *Tri-grams*, Monge-Elkan y SoftTF-IDF) fueron aplicadas sobre los conjuntos de datos tratando de determinar la eficacia de las funciones ante cada situación problemática.

El resto del presente artículo está organizado como sigue: la sección 2 menciona sin detalle las nueve funciones de similitud sobre cadenas de texto que conforman la guía propuesta en este artículo. La sección 3

describe la métrica utilizada para la evaluación de las funciones de similitud. La sección 4 describe el diseño del experimento realizado para evaluar la eficacia de las funciones. La sección 5 muestra los resultados obtenidos y con base en ellos se elabora la guía metodológica propuesta en la sección 6. Por último se presentan las conclusiones y posibles trabajos futuros en la sección 7.

2 Funciones de similitud sobre cadenas de texto

Como se mencionó anteriormente, pueden clasificarse como basadas en caracteres y basadas en *tokens*. Las funciones de similitud basadas en caracteres, consideran cada cadena de caracteres como una secuencia ininterrumpida de caracteres. En este trabajo se consideraron las siguientes: Distancia de edición, Distancia de brecha afín, Similitud Smith-Waterman, Similitud de Jaro y Similitud de *q-grams*. Las funciones de similitud basadas en *tokens* consideran que cada cadena está compuesta por un conjunto de subcadenas separadas por caracteres especiales (espacios en blanco, puntos y comas), es decir, como un conjunto de palabras o *tokens*, y calculan la similitud entre cada pareja de *tokens* mediante alguna función de similitud basada en caracteres. En este trabajo se consideraron dos de las funciones basadas en *tokens* más comunes: Monge-Elkan y coseno TF-IDF.

Para conocer acerca del funcionamiento, algoritmos existentes y orden de complejidad computacional, remítase a las referencias de las tablas 1 a 7.

Tema	Referencias
Modelo original con tres operaciones de edición de costo unitario conocido como distancia de Levenshtein.	[21]
Modelo modificado para permitir operaciones de edición con distinto costo, permitiendo modelar errores ortográficos y tipográficos comunes.	[27]
Modelo que permite la trasposición de dos caracteres adyacentes como una cuarta operación de edición, usualmente referido como distancia de Damerau-Levenshtein.	[22]
Normalización de distancias dividiendo por la cadena más larga, por la suma de las longitudes de ambas cadenas o por el número de operaciones de edición.	[6] [37] [23]
Modelo que aprende automáticamente a minimizar los costos de las operaciones de edición.	[31]
Desarrollo de la primera técnica de normalización que satisface la desigualdad triangular.	[43]
Algoritmo de programación dinámica para el cálculo de la distancia de edición no normalizada en un orden de complejidad $O(nm)$	[36]
Algoritmo que toma $O(n \cdot max(1, m / log\ n))$	[24]
Algoritmo para verificar si la distancia de Damerau-Levenshtein entre dos cadenas es menor que cierta constante d	[18]
El orden de complejidad de la técnica de normalización propuesta por Marzal y Vidal que toma $O(n2m)$ es reducido a $O(nm \log m)$, y a $O(nm)$.	[2] [35]

Tabla 1. Referencias Distancia de Edición

Tema	Referencias
Descripción de la distancia de brecha afín y algoritmo de programación dinámica para calcular la distancia de brecha afín no normalizada en un orden de $O(nm)$	[16]
Modelo para entrenar automáticamente esta función de similitud a partir de un conjunto de datos.	[4]

Tabla 2. Referencias Distancia de Brecha Afín

Tema	Referencias
Descripción similitud Smith-Waterman y algoritmo para calcularla en un orden de complejidad de *O(nm)*	[32]
Normalización de la similitud de Smith-Waterman con base en la longitud de la cadena de mayor longitud, la de menor longitud o la longitud media de ambas cadenas (coeficientes de Jaccar, Overlap y Dice respectivamente).	[6]
Normalización por la suma de la longitud de las subcadenas de texto *A′ y B′ más similares*	[2]
Modelo que puede ser entrenado automáticamente a partir de un conjunto de datos.	[5]
Algoritmo que toma *O(n)* para verificar si la similitud Smith-Waterman entre dos cadenas es menor que cierta constante *k*	[3]

Tabla 3. Referencias Similitud Smith-Waterman

Tema	Referencias
Descripción de la función de similitud de Jaro que define la trasposición de dos caracteres como la única operación de edición permitida.	[19]
Variante de la función de similitud de Jaro que asigna puntajes de similitud mayores a cadenas que comparten algún prefijo basándose en un estudio realizado por Pollock y Zamora [30].	[39]
Modelo basado en distribuciones Gaussianas.	[7]
Comparación de la eficacia de la similitud de Jaro y algunas de sus variantes.	[42]

Tabla 4. Referencias Similitud de Jaro

Tema	Referencias
Descripción de similitud de q-grams.	[42]
Extensión natural a los *q-grams* conocida como *q-grams* posicionales.	[33]
Modelo alternativo conocido bajo el nombre de *k-skip-q-grams* : *q-grams* que omiten *k* caracteres adyacentes.	[20]
Cálculo de la similitud de *q-grams* en *O(n)*	[34] [8]

Tabla 5. Referencias Similitud de q-grams

Tema	Referencias
Descripción similitud Monge-Elkan.	[25]
Modelo basado en la media aritmética generalizada, en lugar del promedio, el cual supera al modelo original sobre varios conjuntos de datos.	[15]

Tabla 6. Referencias Similitud de Monge-Elkan

Tema	Referencias
Descripción Similitud Coseno.	[9]
Descripción Similitud Coseno SoftTF-IDF.	[7]

Tabla 7. Referencias Similitud Coseno TF-IDF

3 Evaluación de funciones de similitud sobre cadenas de texto

Las *curvas de precisión y de memoria* son frecuentemente usadas para evaluar funciones de similitud sobre cadenas de texto. Dichas curvas se obtienen a partir de un tipo especial de consultas conocidas como *top-k queries* [17], en las cuales se retornan las *k* instancias más similares a la cadena buscada. Para una función de similitud particular, su curva de precisión y de memoria mide la capacidad que tiene para mover las cadenas relevantes a la buscada dentro de los primeros *k* resultados. Sin embargo, en el proceso de detección de duplicados se utilizan *consultas*

de rango, en las cuales se retornan todas las cadenas cuya similitud con la buscada es mayor que cierto umbral *t* previamente definido. En este contexto, una técnica de evaluación acertada debe tener en cuenta [17]:

1. Si la función de similitud consigue separar correctamente cadenas relevantes de irrelevantes, asignando puntajes superiores al umbral a las primeras, e inferiores al umbral a las últimas.
2. El grado de separación entre cadenas relevantes e irrelevantes. Una buena función de similitud no sólo debe distinguir entre unos y otros, sino ponerlos dentro de una distancia razonable de forma que creen dos conjuntos distintos claramente definidos.
3. La variación del umbral *t* seleccionado, pues la distribución de valores de similitud puede variar de un conjunto de datos a otro.

Las curvas de precisión y de memoria sólo tienen en cuenta el primer aspecto. Por esta razón, en el presente trabajo se utiliza una métrica de evaluación propuesta en el año 2007 por Da Silva *et al.* [10], llamada *función de discernibilidad*.

3.1 Función de discernibilidad

La discernibilidad es una métrica que ha sido utilizada en trabajos comparativos como es el caso de la herramienta SimEval [17] para determinar cuán eficaces son las funciones de similitud identificando las entradas que realmente corresponden a un mismo objeto. Esta métrica intrínsecamente incorpora los cuatro elementos tradicionales de una tabla de contingencia o matriz de confusión de 2*2: aciertos, desaciertos, falsos positivos y falsos negativos.

La discernibilidad de una función de similitud se calcula como [10]:

$$\frac{c_1}{c_1 + c_2}\left(t_{max}^{Óptimo} - t_{min}^{Óptimo}\right) + \frac{c_2}{c_1 + c_2}\left(\frac{F_{max}}{2\,|Q|}\right) \qquad (1)$$

El rango $[t_{min}^{óptimo}, t_{max}^{óptimo}]$ determina el intervalo de umbrales que mejor separa cadenas relevantes de irrelevantes, pues actúa como indicador de la distancia entre éstas (así, entre mayor sea, mejor), y puede ser calculado por cualquiera de los dos algoritmos propuestos en [10]. El término $F_{max} / 2\,|Q|$ indica el porcentaje de separaciones correctas logrado. Los coeficientes c_1 y c_2 permiten controlar la importancia de cada uno de los dos aspectos anteriores. Independientemente de los valores dados a c_1 y c_2, los valores de discernibilidad siempre caerán en el intervalo [-1,1]: -1 en el peor caso y 1 en el caso de la función de similitud ideal, que logra separar correctamente todas las cadenas relevantes de las irrelevantes a la distancia máxima posible.

4 Diseño del experimento para la comparación de las técnicas

Usando la métrica de la discerniblidad, como medida de bondad de cada técnica, se compararon nueve funciones de similitud sobre cadenas de texto, bajo las seis situaciones problemáticas detalladas en la tabla 8. Cada situación corresponde a un caso normalmente encontrado en la representación textual de una misma entidad. Las técnicas se comparan con el fin de establecer si algunas de ellas son más eficaces para detectar duplicados en presencia de ciertas situaciones problemáticas.

Para las pruebas realizadas se definió $c_1 = 3$ y $c_2 = 7$ en (6), de forma que el intervalo óptimo contribuya en 30% al valor de discernibilidad y $F_{max} / 2 |Q|$ en 70%. Esto por dos razones: primero, para fines prácticos, es mucho más importante que la función de similitud separe correctamente las cadenas relevantes de las irrelevantes, independientemente de la distancia a la que ubiquen unas de otras. Segundo, al dar igual importancia a ambos factores ($c_1 = 1$ y $c_2 = 1$) una función de similitud podría obtener un valor de discernibilidad considerablemente alto con un intervalo óptimo largo de umbrales que logren un porcentaje de separaciones correctas bajo, lo cual no tiene sentido.

Situación Problemática	Ejemplo
Errores ortográficos (ERR)	Juan Alberto Pérez Zuluaga vs.Joan Alverto Peres Suluaga
Abreviaturas: truncamiento de uno o más *tokens* (ABR)	Juan Alberto Pérez Zuluaga vs. Juan A. Pérez Z
Tokens faltantes: eliminación de uno o más *tokens* (TFL)	Juan Alberto Pérez Zuluaga vs. Juan Pérez
Prefijos/sufijos sin valor semántico: presencia de subcadenas al principio y/o al final (PSF)	Juan Alberto Pérez Zuluaga vs. PhD Juan Alberto Pérez Zuluaga, U. Nacional de Colombia
Tokens en desorden (TDR)	Juan Alberto Pérez Zuluaga vs. Pérez Zuluaga Juan Alberto
Espacios en blanco: eliminación o adición de espacios en blanco que causan la unión y/o separación de uno o más *tokens* (ESP)	Juan Alberto Pérez Zuluaga vs. JuanAlberto Pérez Zuluaga

Tabla 8. Situaciones Problemática para Comparación de Funciones de Similitud

Mediante la herramienta Web FakeNameGenerator[1] se generaron aleatoriamente conjuntos de datos compuestos por registros con campos como nombre, apellidos, dirección, ocupación y correo electrónico. A partir de estos registros, otros fueron derivados de acuerdo con la variación textual o situación problemática a probar. Así, para la situación problemática ERR (Errores ortográficos), se generaron nuevas cadenas a las cuales se les introdujeron errores de ortografía cambiando unas letras por otras (por ejemplo g por j, v por b, c por s, entre otras). Para la situación problemática ABR (Abreviaturas), se generaron nuevas cadenas a las cuales se les recortaron los segundos nombres, segundos apellidos y las ocupaciones, dejando sólo la inicial o primera letra de estos. Para la

1 http://www.fakenamegenerator.com/

situación problemática TFL (*Tokens* Faltantes), se generaron nuevas cadenas en las cuales se les omitieron una o varias palabras. Para la situación problemática PSF (prefijos/sufijos), se generaron nuevas cadenas a las cuales se antepuso o pospuso al nombre completo de la persona, textos de diferentes longitudes como Doctor, Magíster, Estudiante, Universidad Nacional de Colombia, entre otros. Para la situación problemática TDR (*Tokens* en desorden), se cambió al nombre completo el orden de las palabras colocando primero los dos apellidos y luego los nombres y al oficio también se le cambió el orden de las palabras. Para la situación problemática ESP (Espacios en Blanco), se agregaron/eliminaron espacios en blanco entre las palabras.

Se generaron diez conjuntos de datos de prueba para cada situación problemática con las respectivas variaciones textuales. Cada conjunto de datos tiene 400 cadenas representando 200 entidades (dos cadenas por entidad). Entonces se comparó para cada situación problemática, la eficacia de las siguientes funciones de similitud: distancia de Levenshtein, distancia de brecha afín, similitud Smith-Waterman, similitud de Jaro y Jaro-Winkler, similitud de *bi-grams* y *tri-grams*, similitud de Monge-Elkan y similitud SoftTF-IDF.

También se comparó la eficacia de las anteriores funciones de similitud sobre dos conjuntos de datos reales extraídos de la base datos ScienTI[2], en los cuáles se presentan varias situaciones problemáticas simultáneamente. En ésta se registra gran parte de la actividad investigativa en Colombia, incluyendo: grupos de investigación, investigadores, proyectos en curso y proyectos finalizados. En ocasiones el título de un mismo proyecto se registra – en texto libre – varias veces (por ejemplo, una vez por cada participante). De igual forma, una misma persona registra – de nuevo, en texto libre – varias veces su nombre (por ejemplo, una vez por cada proyecto en el que participe). Así, es muy probable que una misma entidad, sea título de proyecto o nombre de investigador, aparezca representada varias veces de distintas formas.

El primer conjunto de datos se formó con 642 nombres personales representados en 1284 cadenas diferentes, extraídas y etiquetadas manualmente; cada nombre es representado de dos formas distintas. De igual forma, el segundo conjunto de datos se formó con 303 títulos de proyectos representados en 606 cadenas diferentes; cada título es representado de dos formas distintas. Las variaciones encontradas con mayor frecuencia en ambos conjuntos de datos fueron:

- Errores ortográficos y tipográficos.
- Abreviaturas: en el primer conjunto de datos, usualmente de un segundo nombre y/o apellido.

2 http://thirina.colciencias.gov.co:8081/scienti/

- Tokens faltantes: en el primer conjunto de datos, usualmente la ausencia de un segundo nombre o apellido. En el segundo conjunto de datos, la omisión de una o más palabras en el título de un mismo proyecto, como por ejemplo Antimaláricos en artritis reumatoidea vs. Antimaláricos en el tratamiento de la artritis reumatoidea.

5 Resultados

Luego de generados los archivos con los datos de prueba, se calculó la discernibilidad de las nueve funciones de similitud bajo estudio, para cada uno de los diez conjuntos de datos con cada situación problemática. Para todas las situaciones problemáticas, las funciones de similitud arrojan resultados variables de la discernibilidad, pudiéndose pensar que algunas de ellas se comportan mejor que otras ante ciertas situaciones problemáticas. Para poder llegar a conclusiones válidas, se quiso realizar un análisis de varianza (ANOVA) con el fin de determinar si los resultados obtenidos son estadísticamente significativos, pero luego de realizar la prueba de Kolmogorov-Smirnov se determinó que el supuesto de normalidad no se cumple. Por ello, se aplicó la prueba no paramétrica de Kruskal-Wallis. Ésta permite probar la hipótesis nula de igualdad de las medianas de la discernibilidad para las nueve funciones de similitud. La prueba realizada arrojó como resultado para todas las situaciones problemáticas un valor p inferior a 0,05, pudiéndose afirmar que hay diferencias estadísticamente significativas entre las medianas, con un nivel de confianza del 95,0%. Para determinar cuáles son las medianas significativamente diferentes entre sí, se realizó el Gráfico de Caja y Bigotes. La figura 1 presenta dicho gráfico para la situación problemática ABR.

El análisis del gráfico permite identificar las funciones de similitud Brecha Afín, Soft TF- IDF y Smith-Waterman como las de mejores resultados, en su orden, ante la situación problemática Abreviaturas. Un análisis similar para las otras situaciones problemáticas, permitió identificar a las funciones de similitud de mejores resultados en cada caso. La tabla 9, en cada fila presenta las posiciones obtenidas para una situación problemática por las diferentes funciones.

La tabla 10, en cada fila presenta las posiciones obtenidas pero vistas por función de similitud, lo cual permite observar qué tan buena es una función en las diferentes situaciones problemáticas. Las dos últimas columnas corresponden a la media aritmética y la desviación estándar de las posiciones ocupadas por cada una de las funciones de similitud. Nótese como un valor promedio bajo acompañado de una desviación estándar baja, es indicativo de que la función obtuvo buenos resultados, no sólo para una función problemática, sino que tuvo un buen comportamiento general. Obsérvese como la función Tri-grams obtuvo la mejor posición promedio (2.8) y su desviación estándar es moderada (1.47), lo que significa que para todas las

funciones obtuvo posiciones no muy alejadas del promedio. Tri-grams, es seguida por Smith Waterman con un promedio de 3.2 una desviación estándar de 2.64, esto es, un promedio mayor y valores menos uniformes. Asimismo, la función Jaro-Winkler, obtuvo el promedio más alto (8.7) con una desviación estándar baja (0.82), lo cual significa que logró resultados uniformemente desfavorables para todas las situaciones problemáticas.

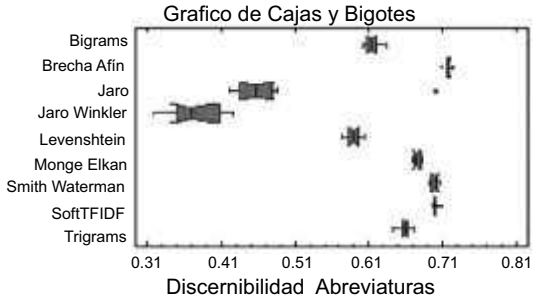

Fig. 1. Gráfico de Cajas y Bigotes para la situación problemática ABR.

Situación Problemática	Posición								
	1	2	3	4	5	6	7	8	9
ERR	L	W	2	3	B	M	J	S	Jw
ABR	B	S	W	M	3	2	L	J	Jw
TFL	W	3	B	L	2	S	M	J	Jw
PSF	S	B	3	W	2	M	L	J	Jw
TDR	3	2	S	M	J	B	Jw	W	L
ESP	W	3	L	2	B	S	M	J	Jw

B: Brecha Afín S: Soft TF-IDF W: Smith-Waterman
L: Levenshtein 2: Bi-grams 3: Tri-grams
M: Monge-Elkan J: Jaro Jw: Jaro Winkler

Tabla 9. Resultados Eficacia por Situación Problemática

Función Similitud	Situación Problemática							
	ERR	ABR	TFL	PSF	TDR	ESP	Prom	Desv
LevenshTein	1	7	4	7	9	3	5.2	2.99
Brecha Afín	5	1	3	2	6	5	3.7	1.97
Bi-grams	3	6	5	5	2	4	4.2	1.47
Tri-grams	4	5	2	3	1	2		1.47
Jaro	7	8	8	8	5	8	7.3	1.21
Jaro Winkler	9	9	9	9	7	9	8.7	0.82
Smith Waterman	2	3	1	4	8	1	3.2	2.64
Monge Elkan	6	4	7	6	4	7	5.7	1.37
Soft TF-IDF	8	2	6	1	3	6	4.3	2.73

Tabla 10. Resultados Eficacia por Función de Similitud

La tabla 11 presenta los resultados de la discernibilidad obtenida por las diferentes funciones de similitud sobre los conjuntos de datos reales

extraídos de la base de datos Scienti, en los cuales los principales problemas observados son errores ortográficos y tipográficos, abreviaturas y *tokens* faltantes.

	Nombres Personales	Títulos de Proyectos
Levenshtein	0,608	0,706
Brecha afín	0,666	0,698
Smith-Waterman	0,619	0,693
Jaro	0,570	0,677
Jaro-Winkler	0,567	0,659
Bi-grams	0,614	0,659
Tri-grams	0,606	0,695
Monge-Elkan	0,667	0,695
SoftTF-IDF	0,656	0,692

Tabla 11. Resultados de la Discernibilidad para los Conjuntos de Datos Extraídos de ScienTI.

En esta ocasión, puede verse que las diferencias en la discernibilidad de las funciones de similitud son menos marcadas, esto es, la eficacia de las funciones es relativamente similar, pudiéndose pensar que en presencia de varias situaciones problemáticas en forma simultánea la eficacia de las funciones tiende a confundirse.

6 Guía Metodológica

El principal aporte de este trabajo es proveer una guía que oriente a los analistas de datos en la selección de las técnicas más adecuadas a la situación particular de unos datos con problemas de duplicados. La guía en cuestión, se elaboró bajo la forma de un diagrama de flujo de datos, el cual se presenta en la figura 2.

Este diagrama, en esencia, recomienda una función de similitud para cada una de las situaciones problemáticas en caso de que éstas se presenten individualmente en los datos bajo evaluación. La tabla 12 resume las funciones recomendables en cada caso.

Situación Problemática	Mejor función de similitud
ERR	Levenshtein
ABR	Brecha Afín
TFL	Smith-Waterman
PSF	Soft TF-IDF
TDR	Tri-grams
EST	Smith-Waterman

Tabla 12. Funciones de similitud más eficaces para cada situación problemática

Aunque este trabajo se basó principalmente en la eficacia de los métodos para detectar duplicados más que en la eficiencia computacional, este es un aspecto siempre importante, máxime con los altos volúmenes de

datos de las bases de datos actuales. Es por esto, que al examinar el costo computacional de las funciones a recomendar (ver referencias tablas 1 a 7), sobresale el alto costo de la función Soft TF-IDF, el cual se basa en el número de veces que aparece cada *token* de una cadena en otra cadena y adicionalmente en el número de veces que aparece dentro de las cadenas a detectar duplicados. Debido a esto, la situación problemática PSF (Prefijos/Sufijos), merece atención especial ya que en caso de observarse esta situación no se recomienda inmediatamente la función Soft TF-IDF como correspondería al primer lugar ocupado por esta función para esa situación problemática. Esto se debe a que la discernibilidad lograda por la función Brecha Afín es muy cercana a la de Soft TF-IDF, pero la eficiencia computacional de Brecha Afín es mayor, siendo preferible usar esta función en casos donde se tenga un alto volumen de datos y utilizar a Soft TF-IDF para volúmenes bajos de datos.

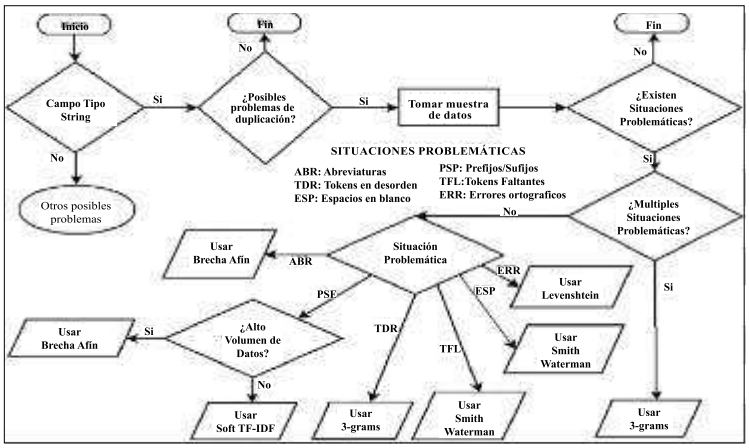

Fig. 2. Diagrama de flujo para guiar la selección de técnicas para detección de duplicados.

El diagrama comienza indagando si el tipo de datos de una columna dada a la cual se desee hacer limpieza es de tipo *String* ya que la duplicación es un problema exclusivo de las cadenas de texto. Para campos que tengan otros tipos de datos –y no se traten como *Strings*- , se debería buscar otros tipos de problemas de calidad de datos (por ejemplo: valores extremos para campos numéricos), pero esto se sale del alcance de este trabajo y por tanto el diagrama no indica qué hacer en esos casos.

Para atributos tipo texto, se pregunta si existen posibles problemas de duplicación. Aunque aparentemente es una pregunta difícil de responder por parte de un usuario, realmente no lo es tanto. Para un usuario conocedor de los datos y que entienda el concepto de detección de duplicados, no es difícil prever si sus datos son susceptibles de esta situación. Recuérdese que se habla de detección de duplicados cuando el contenido de un campo

o de un registro completo, aparece dos o más veces (duplicado) con diferencias textuales en sus valores y no se tiene un identificador único. Si por ejemplo, en una tabla se almacenan datos de proyectos de investigación, en un proyecto pueden participar varios investigadores y no existe un identificador único como un código que se le asigne previamente a cada proyecto, es fácil que cada investigador entre el título del proyecto con alguna variación en el texto, haciendo que pueda considerarse como un proyecto distinto. Otro ejemplo, podría ser aquella situación en la cual se almacenan pedidos de clientes en una hoja de cálculo, sin contar con un identificador único para cada cliente sino haciéndolo a través de sus nombres. Como un cliente puede colocar varios pedidos, es factible que los nombres no sean ingresados exactamente igual.

En caso de existir posibles problemas de duplicados, la guía sugiere extraer una muestra de datos para tomar decisiones con base en ella. Esto es necesario si se tiene un alto volumen de datos, pues de lo contrario se analiza todo el conjunto de datos. La muestra debe ser representativa de la totalidad de los datos.

Sobre la muestra de datos, debe examinarse si se detecta alguna de las seis situaciones problemáticas, es decir, si en los datos se visualizan errores ortográficos, abreviaturas, *tokens* faltantes o en desorden, presencia de prefijos y sufijos o espacios en blanco adicionales o suprimidos. En caso de que se detecte una o más de estas situaciones problemáticas, debe establecerse si se perciben varias de ellas simultáneamente o si hay un alto predominio de sólo una de ellas. En caso de predominar una sola situación problemática, se recomienda la técnica de mayor eficacia según la tabla 12, excepto para el caso de PSF en el que la función recomendada dependerá del volumen de datos. En caso de observarse en los datos varias situaciones problemáticas simultáneamente, se recomienda la técnica tri-grams ya que es de bajo costo computacional y obtuvo la mejor posición promedio y desviación estándar moderada de todas las funciones (ver tabla 10**).**

7 Conclusiones

Existen diversas funciones de similitud sobre cadenas de texto. Como se mostró, algunas son más eficaces detectando ciertas situaciones problemáticas o variaciones textuales que otras. En este trabajo, mediante conjuntos de datos especialmente diseñados para cada situación, se determinó la función más eficaz en cada caso y se construyó un diagrama para guiar la selección de la técnica más adecuada para un caso particular de la vida real.

Como trabajo futuro, se tiene planeado validar la guía siguiéndola con otros conjuntos de datos reales para obtener conclusiones sobre su uso y sobre la eficiencia de la misma. Además, en la actualidad se están

desarrollando guías similares para otros conflictos en los datos como valores extremos (*outliers*) y valores faltantes.

Referencias

[1] I. Amón y C. Jiménez. Hacia una Metodología para la selección de técnicas de depuración de datos. *Avances en Sistemas e Informática*, 6(1):185-190, 2009.

[2] A.N. Arslan y O. Eğecioğlu, A New Approach to Sequence Comparison: Normalized Sequence. *Alignment. Bioinformatics,* 17(4):327-337, 2001.

[3] R. Baeza-Yates y G.H. Gonnet. A new approach to Text Searching. *Communications of the ACM,* 35(10):74-82, 1992.

[4] M. Bilenko y R.J. Mooney. Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases. *En Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pags. 39-48, 2003.

[5] E. Breimer y M. Goldberg. Learning Significant Alignments: An Alternative to Normalized Local Alignment. *En Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, pags. 37-45, 2002.

[6] P. Christen. A Comparison of Personal Name Matching: Techniques and Practical Issues. *En Sixth IEEE International Conference on Data Mining*, pags. 290-294, 2006.

[7] W.W. Cohen, P. Ravikumarand y S.E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. *En International Joint Conference on Artificial Intelligence*, pags. 73-78, 2003.

[8] J.D.Cohen. Recursive Hashing Functions for n-Grams, *ACM Transactions on Information Systems,* 15(3):291-320, 1997.

[9] W.W. Cohen. Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity. *En Proceedings of the SIGMOD International Conference Management of Data SIGMOD'98*, pags. 201-212, 1998.

[10] R. da Silva *et al*. Measuring Quality of Similarity Functions in Approximate Data Matching. *Journal of Informetrics*, 1(1):35-46, 2007.

[11] H.L. Dunn. Record Linkage. *Americal Journal of Public Health*, 36(12): 1412-1416, 1946.

[12] O. Eğecioğlu y M. Ibel. Parallel Algorithms for Fast Computation of Normalized Edit Distances. *En Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing*, pags. 496-503, 1996.

[13] A.K. Elmagarmid; P.G. Ipeirotis y V.S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1-40, 2007.

[14] I.P. Fellegi y A.B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183-1210, 1969.

[15] A. Gelbukh *et al*. Generalized Mongue-Elkan Method for Approximate Text String Comparison. *En Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, pags. 559-570, 2009.

[16] O. Gotoh. An Improved Algorithm for Matching Biological Sequences. *Journal of Molecular Biology*, 162(3):705-708, 1982.

[17] C.A. Heuser; F.N. Krieser y V.M. Orengo. SimEval - A Tool for Evaluating the Quality of Similarity Functions. *En Tutorials, posters, panels and industrial contributions at the 26th International Conference on Conceptual Modeling*, pags. 71-76, 2007.

[18] H. Hyyrö. A Bit-vector Algorithm for Computing Levenshtein and Damerau Edit Distances. *En The Prague Stringology Conference '02*, pags. 29-39, 2002.

[19] M.A. Jaro. Unimatch: A Record Linkage System User's Manual. Technical report, Washington, D.C.: US Bureau of the Census, 1976.

[20] H. Keskustalo *et al*. Non-adjacent Digrams Improve Matching of Cross-Lingual Spelling Variants. *En International Symposium on String Processing and Information Retrieval*, pags. 252-256, 2003.

[21] V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707-710, 1966.

[22] R. Lowrence y R.A. Wagner. An Extension of the String-to-String Correction Problem. *Journal of the ACM*, 22(2):177-183, 1975.

[23] A. Marzal y E. Vidal. Computation of Normalized Edit Distance and Applications. *IEEE Transactions on Patern Analysis and Machine Intelligence*, 15(9): 926-932, 1993.

[24] W.J. Masek. A Faster Algorithm for Computing String Edit Distances. *Journal of Computer and System Sciences*, 20(1):18-31, 1980.

[25] A.E. Monge y C.P. Elkan. The Field Matching Problem: Algorithms and Applications. *En Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pags. 267-270, 1996.

[26] E. Moreau; F. Yvon y O. Cappé. Robust Similarity Measures for Named Entities Matching. *En Proceedings of the 22nd International Conference on Computational Linguistics*, pags. 593-600, 2008.

[27] S.B. Needleman y C.D. Wunsh. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins, *J Mol Biol,* **48(**3):443–453, 1970.

[28] H. Newcombe y J. Kennedy. Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information, *Communications of the ACM*, 5(11):563- 566, 1962.

[29] H. Newcombe *et al.* Automatic Linkage of Vital Records. *Science*, 130(3381): 954-959, 1959.

[30] J.J.Pollock y A. Zamora. Automatic Spelling Correction in Scientific and Scholarly Text. *Communications of the ACM*, 27(4):358-368, 1984.

[31] E. Ristad y P. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5): 522-532, 1998.

[32] T.F. Smith y M.S. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147(1):195-197, 1981.

[33] E. Sutinen y J. Tarhio. On Using Q-Gram Locations in Approximate String Matching. *En Proceedings of the Third*

Annual European Symposium on Algorithms, pags. 327-340, 1985.

[34] E. Ukkonen. Aproximate String-Matching With Q-grams and Maximal Matches. *Theoretical Computer Science*, 92(1):191-211, 1992.

[35] E. Vidal; A. Marzal y P. Aibar. Fast Computation of Normalized Edit Distances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):899-902, 1995.

[36] R.A. Wagner y M.J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168-173, 1974.

[37] A. Weigel y F. Fein. Normalizing the Weighted Edit Distance. *En Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pags. 399-402, 1994.

[38] W.E. Winkler. Frequency-Based Matching in the Fellegi-Sunter Model of Record Linkage. *En Proceedings of the Section on Survey Research Methods*, pags. 778-783, 1989.

[39] W.E. Winkler. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *En Proceedings of the Section on Survey Research Methods*, pags. 354-359, 1990.

[40] W.E. Winkler. Improved Decision Rules in the Felligi-Sunter Model of Record Linkage. *En Proceedings of the Section on Survey Research Methods*, pags. 274-279, 1993.

[41] W.E. Winkler. Using the EM Algorithm for Weight Computation in the Fellegi-Sunter Model of Record Linkage. *En Proceedings of the Section on Survey Research Methods*, pag. 667-671, 2000.

[42] W.E. Yancey. Evaluating String Comparator Performance for Record Linkage. *En Proceedings of the Fifth Australasian Conference on Data mining and Analytics*, pags. 21-23, 2006.

[43] L. Yujian y L. Bo. A Normalized Levenshtein Distance Metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091-1095, 2007.