UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA EVALUADOR DE CALIDAD DE DATOS EN MYSQL

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A :

ORTIZ MONREAL CARLOS GIBRAN

DIRECTORA DE TESIS:
DRA. MARÍA DEL PILAR ÁNGELES
Enero 2015

Agradecimientos y reconocimientos institucionales

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IN114413 Sistema Evaluador Universal de Calidad de Datos, SEUCAD. Agradezco a la DGAPA - UNAM la beca recibida.

Índice

Capítulo I

Introducción

1.1.- Planteamiento del problema

Los usuarios de los diversos Sistemas de Bases de Datos obtienen un conjunto de datos sin indicación de la calidad de los datos que reciben. De tal forma que los usuarios asumen que los datos son perfectos, provienen de una sola fuente de datos y son atómicos. Sin embargo, se sabe que tales presunciones son inválidas. Debido a esta problemática, es necesario para los usuarios de los diversos Sistemas Manejadores de Base de Datos el poder determinar qué tanto distan los datos reales que se poseen de tales presunciones; esto da origen al concepto de Calidad de Datos e implica la necesidad de poseer métricas que permitan evaluar dicha Calidad de Datos para las fuentes de datos en cuestión.

Por otro lado, en las Bases de Datos de hoy en día, se enfrenta otro problema con un impacto altamente significativo en las entidades que hacen uso de Bases de Datos. Este problema se refiere a la existencia de registros duplicados (ya sea en la misma fuente de datos o en diferentes fuentes) que hacen referencia a la misma entidad. Esta situación tiene fuertes implicaciones en el uso y alcance de los datos de tales fuentes de datos en las cuales existan duplicados. Debido a la enorme importancia que tiene el identificar tales registros duplicados, ha surgido el concepto de Vinculación de Registros; este tópico, así como sus implicaciones y consecuencias, será desarrollado extensivamente en el presente trabajo de tesis.

De esa manera, debido a la importancia de la Calidad de Datos y de la identificación de duplicados para las fuentes de datos con las que se vaya a trabajar, es necesario el poseer una herramienta de software que abarque tanto la parte de Calidad de Datos como la Vinculación de Registros. Por lo tanto, el problema que se aborda en el presente trabajo de tesis es el desarrollo de un sistema de software que permita la evaluación de la Calidad de Datos para Bases de Datos en el manejador MySQL y la ejecución de un proceso de Vinculación de Registros que resuelva el problema de los datos duplicados.

Finalmente, es necesario considerar que no todas las personas que trabajan con Bases de Datos se encuentran altamente versadas en los tópicos de Calidad de Datos y Vinculación de Registros, por lo que se busca que la herramienta de software que abarque estos tópicos sea de uso intuitivo para el operador final.

1.2.- Propuesta de la solución

Para la problemática de la Calidad de Datos se propone la implementación de software escrito en lenguaje SQL que pueda calcular las métricas de Calidad de Datos de completitud, densidad, cobertura y unicidad para cualquier Base de Datos existente en un Sistema Manejador de Base de Datos MySQL mediante el acceso a los metadatos de tal manejador.

Por otro lado, para la problemática de la Vinculación de Registros se utilizará y mejorará el software libre llamado Febrl. Febrl fue creado por el Dr. Peter Christen de la Universidad Nacional de Australia [1][2] y a dicho software se le agregará un método adicional de clasificación así como las métricas de Calidad de Datos mencionadas anteriormente, creando así un sistema integral que permita al usuario final tanto la evaluación de la Calidad de Datos como la Vinculación de Registros; el método adicional de clasificación presentará ciertas ventajas que lo harán idóneo para su uso por personas con poca experiencia en Vinculación de Registros pero con un buen conocimiento de los datos a vincular.

Las soluciones generadas para los tópicos tanto de Calidad de Datos como de Vinculación de Registros serán integradas en el producto final del presente trabajo de tesis. Tal producto será referido como “Sistema Evaluador de Calidad de Datos en MySQL”. Tal sistema será implementado de una manera en la que posteriormente se pueda integrar en un Sistema Evaluador Universal de Calidad de Datos, el cual proveerá las características enunciadas en el presente trabajo de tesis para cualquier Base de Datos para los principales Sistemas Manejadores de Base de Datos del mercado.

1.3.- Objetivo general

El objetivo es implementar un Sistema Evaluador de Calidad de Datos para el Sistema Manejador de Datos MySQL. Dicho sistema deberá ser capaz de conectarse a cualquier base de datos implementada en dicho manejador de Base de Datos.

Este sistema evaluador deberá estar integrado con la aplicación Febrl. Si bien la evaluación de datos se ejecutará sobre el sistema Febrl ya existente, se le aplicarán mejoras y extensiones a este último para beneficio del usuario final.

Además de las métricas de Calidad de Datos, el software a implementar también incluirá un método de clasificación de vectores adicional a los ya existentes en Febrl. Este método está diseñado para ser un método simple que permita a usuarios no expertos en Vinculación de Registros pero con un buen conocimiento

de los datos en cuestión el realizar el proceso de Vinculación de Registros de manera fácil y aprovechando sus conocimientos sobre dicha información.

Finalmente, todo lo anterior se podrá usar de manera gráfica desde la interfaz de Febrl, con la finalidad de que las personas que tengan poca experiencia en el campo de la Vinculación de Registros puedan utilizar el sistema con facilidad.

1.4.- Objetivos particulares

Los objetivos particulares de esta tesis son los siguientes:

I.- Programación de scripts en lenguaje SQL para el acceso al diccionario de datos y así poder operar sobre cualquier Base de Datos en el manejador MySQL.

II.- Programación de scripts en lenguaje SQL para el cálculo de las métricas de Calidad de Datos de completitud, unicidad, densidad y cobertura.

III.- Implementación de la interfaz de los scripts anteriores con el Vinculación de Registros Febrl.

IV.- Implementación de un nuevo método de clasificación de vectores para la identificación de registros duplicados. Dicho método será fácil de usar y permitirá que las personas con un buen conocimiento de las características de los datos puedan usar fácilmente el sistema.

V.- Implementación de la interfaz del método anterior de clasificación con el sistema Febrl para su uso desde la interfaz gráfica.

VI.- Evaluación del nuevo método implementado frente a los métodos de clasificación no supervisada ya implementados en Febrl. El análisis se hará a partir de datos de prueba generados con una herramienta ya existente incluida en Febrl.

1.5.- Descripción breve de los capítulos

El Capítulo I del presente trabajo brinda una introducción a la problemática que se aborda en el presente trabajo de tesis, así como el objetivo general y los objetivos particulares a lograr mediante el presente trabajo.

El Capítulo II brinda una introducción al concepto de Calidad de Datos y sus métricas asociadas. Además se mencionan y describen las métricas a utilizar en el presente trabajo de tesis.

El Capítulo III es un desarrollo detallado del proceso de Vinculación de Registros. Se define el concepto de Vinculación de Registros, se explican las diferentes fases que componen a dicho proceso y se presenta un ejemplo que se desarrolla a través de cada fase para ilustrar la Vinculación de Registros desde su fase inicial hasta que ha concluido.

El Capítulo IV se refiere al software libre Febrl. En ese capítulo se brinda una introducción a tal software, además de abordarse detalles de su arquitectura y funcionamiento.

El Capítulo V desarrollará la implementación realizada de las métricas de Calidad de Datos abordadas en el Capítulo II. Se abordarán detalles técnicos de la implementación y los resultados obtenidos.

El Capítulo VI corresponde a la mejora del proceso de Vinculación de Registros. En tal capítulo se describirá el nuevo método de clasificación para Febrl, las ventajas y desventajas asociadas a éste y su implementación. Así mismo, se abordarán las pruebas realizadas con este algoritmo y los resultados observados frente a otros métodos de clasificación ya existentes en el software Febrl original.

Finalmente, el Capítulo VII corresponde a la conclusión del presente trabajo de tesis.

Capítulo II

Calidad de Datos

2.1.- Introducción

En la actualidad los datos almacenados de manera electrónica tienen cada vez una mayor importancia en las actividades humanas. Debido a la difusión de las Tecnologías de la Información y las Comunicaciones, la magnitud de los datos electrónicos es cada vez mayor. Debido a esto, el poseer datos de calidad tiene una gran relevancia en la actualidad [3].

El concepto de Calidad de Datos ha sido abordado desde las perspectivas de disciplinas como la Administración, las Ciencias de la Computación y la Estadística. La Calidad de Datos se relaciona directamente con la Vinculación de Registros (tópico a tratar en el Capítulo III) pues los problemas iniciales de Calidad de Datos fueron entre otros, la existencia de datos duplicados en las fuentes de datos [4].

Al abordar la Calidad de Datos desde un punto de vista administrativo, es importante notar que los datos de baja calidad tienen un impacto muy significativo económicamente hablando. Por ejemplo, en Estados Unidos los datos de calidad pobre le cuestan a las empresas unos 600 billones de dólares anuales [5]. De hecho, el costo de datos de baja calidad puede corresponder a entre el 10 y el 25 por ciento de los ingresos de una compañía [6].

Es necesario comprender que los datos de calidad pobre pueden afectar tanto la operación diaria de una empresa como la implementación de diferentes proyectos de la misma [7]. Debido a esto, tanto gobiernos como empresas han identificado la importancia de poseer datos de calidad y han tomado determinaciones al respecto [8]. Así, es crítico el poder evaluar la calidad de los datos que una entidad posee. Dicha evaluación se realiza mediante el cálculo de métricas y éstas indicarán el grado de calidad presente en los datos. Esto permitirá realizar las acciones posteriores necesarias con el fin de que las fuentes de datos sobre las que se trabaje resulten idóneas para los procesos en los cuales se utilizarán.

2.2.- Métricas

De acuerdo a lo mencionado en la introducción de este capítulo, es crítico el poder evaluar la calidad de los datos presentes en una entidad. Para lograrlo existen diversas métricas (también llamadas *dimensiones* [9]) que proporcionan una base para la evaluación de la calidad de los datos.

En las siguientes secciones se describen las métricas de cobertura, densidad, completitud y unicidad debido a éstas fueron, debido a sus características, las que se seleccionaron para su implementación en el presente trabajo de tesis.

2.2.1.- Cobertura

La cobertura es una métrica de Calidad de Datos que indica la proporción en la que la fuente de datos actual cubre las diferentes entidades existentes en todas las fuentes de datos disponibles. De esa manera, la cobertura para una tabla puede ser calculada con respecto a todas las tablas que integren una base de datos en cuestión de acuerdo a la siguiente fórmula [10]:

$$\mathbf{c(S)} = \frac{\mathbf{|S|}}{\mathbf{UR}}$$

Dónde:

|S| es el número de entidades del mundo real diferentes representadas en la tabla

|UR| es el número de entidades del mundo real diferentes representadas en todas las tablas que integran la Base de Datos

De acuerdo a la formula anterior, la cobertura tendrá valores en el intervalo cerrado [0,1]. Estos valores se pueden considerar como una medida de la probabilidad de que una entidad se encuentre representada en la tabla en cuestión [10].

2.2.2.- Densidad

La densidad es una métrica de Calidad de Datos que indica que tan poblada se encuentra una fuente de datos con campos que tengan un valor diferente del nulo [11]. De esa manera, la densidad puede calcularse a nivel de atributo o columnar de acuerdo a la siguiente formula:

$$\mathbf{d_s(a) = \frac{|\{t \in S \mid t[a] \neq \perp\}|}{|S|}}$$

Dónde

t es un registro de la fuente de datos

a es el atributo para el cual se calcula la densidad columnar

t[a] es el valor del atributo a para el registro t

$\perp$ es el valor nulo

|S| es el número total de registros en la fuente de datos

De acuerdo a lo anterior la densidad tendrá valores en el intervalo cerrado [0,1], además es posible calcular la densidad correspondiente para una tabla de una Base de Datos dada al obtener las densidades mediante las densidades columnares de acuerdo a la siguiente formula [11]:

$$\mathbf{d(S) = \frac{1}{A} \sum_{a \in A} d_s(a)}$$

Dónde:

S es la fuente de datos

A es el conjunto de atributos existentes en la fuente de datos (Tabla de la Base de Datos)

|A| es la cardinalidad del conjunto de atributos existentes en la fuente de datos

d_s es la densidad de atributo o columnar

Mediante la fórmula anterior, es posible a su vez el cálculo de la métrica de densidad para toda una Base de Datos al calcular y promediar los valores obtenidos de densidad para sus respectivas tablas.

2.2.3.- Completitud

La completitud es una métrica de Calidad de Datos que relaciona la cantidad de datos de una fuente con la cantidad potencial de datos en la relación universal. La completitud se puede calcular de acuerdo a la siguiente fórmula [12]:

$$C(S) = c(S) \cdot d(s)$$

Dónde:

S es la fuente de datos

c(S) es la cobertura de la fuente de datos

d(S) es la densidad de la fuente de datos

La fórmula anterior permite el cálculo de la completitud a nivel tabla para una Base de Datos dada. Es también posible el cálculo de la completitud para toda una Base de Datos al obtener los valores de completitud correspondientes a todas sus tablas y promediar dichos valores. La completitud tiene valores en el intervalo cerrado [0,1].

2.2.4.- Unicidad

La unicidad es una métrica de Calidad de Datos que representa una medida del grado de duplicación de los datos presentes en una fuente de datos. La unicidad se puede calcular de acuerdo a la siguiente fórmula:

$$U(S) = 1 - \frac{d}{|S|}$$

Dónde:

S es la fuente de datos

d es el número de registros duplicados en S

|S| es el número de registros totales en S

De acuerdo a lo anterior, la unicidad puede calcularse a nivel Base de Datos al promediar las densidades obtenidas para todas las tablas de dicha Base de Datos [13]. El valor de densidad se encuentra en el intervalo cerrado [0,1].

Capítulo III

Vinculación de Registros

3.1.- Introducción

El término *Vinculación de Registros* (conocido en inglés como Record Linkage) se refiere a la tarea de identificar, corresponder y fusionar registros que se encuentran en diferentes bases de datos. Sin embargo, también existe el caso en el que se quiera aplicar todo lo anterior pero a los datos de una misma base de datos, en cuya situación se llamará a tal proceso como *Detección de Duplicados* [14].

Antes de proceder a explicar a fondo la Vinculación de Registros es necesario ubicarlo con respecto a otros procesos. Así, integrar datos de diversas fuentes es un objetivo difícil y ambicioso que se divide en tres partes diferentes, cada una con tareas y actividades asociadas y con sus dificultades intrínsecas.

El primero de dichos procesos es la *Vinculación de Esquemas* (conocida en inglés como Schema Matching), la cual se encarga de identificar en las diferentes bases de datos las tablas, atributos y estructuras conceptuales que corresponden al mismo tipo de información [15]. El segundo proceso es la *Vinculación de Registros*, también conocida como *Cruce de Datos* (en inglés Data Matching); dicho proceso se explicará a lo largo de este capítulo. Finalmente, la última fase a realizar es la parte de *Fusión de Datos* (conocida en inglés como Data Fusion), la cual consiste en integrar o fusionar los datos que han sido identificados como coincidentes o asociados a una misma entidad por el proceso de Cruce de Datos [16][17].

De esa manera, el proceso de Vinculación de Registros es de gran importancia debido a sus repercusiones. Por ejemplo, en el sector de la salud, muchas organizaciones tanto públicas como privadas están recolectando, almacenando, procesando y analizando cantidades crecientes de datos con millones de registros. La mayoría de estos datos son de personas (aunque también pueden referirse a otra información igualmente importante como registros de medicamentos o detalles de negocio) y la detección y adición de registros que se refieren a la misma persona se está volviendo cada vez más importante, debido a que los datos que han pasado por un proceso de Vinculación de Registros pueden contener información que no estaría disponible de otra manera [18][19].

Así, una aplicación de los datos a lo que se les ha aplicado un proceso de Vinculación de Registros es el de su subsiguiente uso en un proceso de Minería de Datos. La Minería de Datos es el proceso de descubrir información nueva y valiosa a partir de una colección de datos. [20].

La Vinculación de Registros es de vital interés pues los datos que han sido sometidos a un exitoso proceso de Cruce de Datos pueden ayudar a mejorar políticas de salud, descubrir fraudes, reducir costos, detectar reacciones adversas de medicamentos y usarse en lugar del caro proceso de encuestas en estudios epidemiológicos. Por ejemplo, en Australia Occidental se realizó un proyecto de investigación en el que tras un proceso de Vinculación de Registros entre una base de datos de paros cardiacos en ambulancias y la base de datos del hospital se obtuvieron resultados que indicaron que era necesaria la instalación de desfibriladores en las ambulancias y en las salas de hospitalización, lo que ha permitido salvar muchas vidas [18][21][22].

El Cruce de Datos es un proceso complejo que consta de varias etapas, todas con una importancia específica para el resultado final. Dichas etapas se abordan de una manera general en esta introducción, para luego ser abordadas a detalle en sus respectivos capítulos.

La primera fase que debemos realizar en un proceso de Vinculación de Registros es el asegurarnos que los datos a vincular sean en realidad aptos para dicha tarea. Como los datos del mundo real usualmente tienen poseen información incompleta, con errores o con un formato incorrecto, la limpieza y estandarización de dichos datos son pasos vitales para que el proyecto de Vinculación de Registros en cuestión pueda entregarnos resultados satisfactorios. Dichas actividades son también necesarias si deseamos hacer una Minería de Datos o enviar dicha información a un Almacén de Datos (conocido en inglés como *Data Warehouse*) [23].

La segunda fase en el proceso de Cruce de Datos es el indexado. Esta tarea tiene como finalidad el reducir la complejidad computacional inherente al proceso de Vinculación de Registros. Dicha complejidad es cuadrática y puede hacer prohibitivo o impráctico el análisis de grandes fuentes de datos, por lo que el indexado es crucial para poder abordar de manera práctica el proceso de Vinculación de Datos. Por ejemplo, si deseamos realizar una detección de duplicados sobre una base de datos **A**, el número total de comparaciones entre parejas de registros será de |**A**| x (|**A**| -1) / 2 [24]; de esa manera el número de comparaciones crece de manera cuadrática conforme al tamaño de la base de datos. Así, la realización del indexado evita que lo anterior ocurra al discriminar los datos de acuerdo al valor de uno o varios campos (dicho campo o campos se

denominan como *llave de bloque* (o blocking key en inglés) e insertar los registros con el mismo valor de índice en un bloque dado, para luego solamente comparar entre sí a aquellos registros que se encuentren en el mismo bloque [25].

La tercera fase del proceso de Vinculación de Registros es la realización de las comparaciones de parejas de registros. En esta etapa analizarán qué tan similares son los registros que poseen el mismo valor de índice de acuerdo a la fase anterior. Así, se comparan los campos deseados con funciones propias para este fin que nos regresen un valor de similitud para poder determinar qué tan parecidos son los mismos campos de dos vectores diferentes. Dicha operación se realiza para todos los campos que seleccionemos para dicho fin hasta formar un vector de comparación entre dos registros. Este proceso se repite para todos los elementos del bloque de indexado en cuestión y luego se repetirá para los bloques restantes hasta haber concluido las comparaciones por realizar [26].

La cuarta fase es la clasificación. Aquí se toman todos los vectores de comparación generados en la fase anterior y se procesan para clasificarlos a todos ellos. Ahora, al clasificar un vector de comparación el resultado será uno de dos o uno de tres (si así lo permite el método de clasificación) estados posibles: Si la clasificación solo permite dos estados entonces los vectores se clasificarán en coincidentes (los registros corresponden a la misma entidad) y no coincidentes (los registros se refieren a entidades diferentes). Por otro lado si la clasificación permite tres estados, entonces los vectores de comparación se clasificarán en coincidentes, no coincidentes y posiblemente coincidentes. Estos últimos requerirán posterior análisis para determinar su clasificación final.

Finalmente, una vez que la clasificación ha concluido, se procede a evaluar la calidad del proceso de Cruce de Datos. Esto incluye actividades como el determinar cuántos vectores de comparación fueron clasificados correcta e incorrectamente y el cálculo de métricas en cuanto a la complejidad del cómputo realizado.

Así el proceso de Vinculación de Registros puede resumirse esquemáticamente de la siguiente manera:

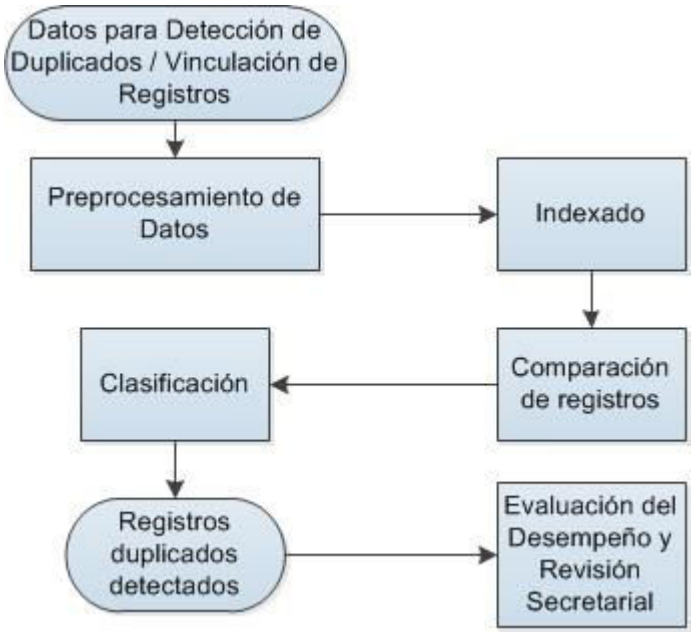


Figura 3.1 Esquema ilustrativo del proceso de Vinculación de Registros / Detección de Duplicados

De acuerdo a lo anteriormente expuesto, las fases del proceso de Vinculación de Registros que se acaban de presentar serán desarrolladas a lo largo de este capítulo.

3.2.- Preprocesamiento de Datos

El proceso de Vinculación de Registros se ve afectado dado que las Bases de Datos pueden variar tanto en estructura como en contenido. Esto puede ser un gran obstáculo para que el Cruce de Datos se realice de manera exitosa. Por ejemplo, consideremos las siguientes Bases de Datos **A** y **B** para un proceso de Vinculación de Registros:

ID	Nombre	Dir	DiaCump	MesCump	AñoCump
a1	Miller, Robert	Hopeview st. 3, New York City, NY, Zip code 49876	10	February	1989
a2	Katherine Clinton	Evergreen Street 17, White Plains, NY, Zip code 49237	23	1	1980
a3	Baltazar, John	LaBelle St 18, Dixon, California Zip Code: 1241732	18	Dec	1970
a4	Norman Harper	Byrne street 88, Sacramento CA, Zip code 1241345	14	March	1990
a5	Lynette Stockman	McGinness street 16, Atlanta, GA, Zip code 42354	14	11	1974
a6	Franklin Reagan	Robertson St. 65, Sacramento CA, 1241345	21	Jan	1981
a7	Martin MacDonald	White st. 71, Atlanta, Georgia, 42350	9	Jan	1991
a8	Robert Delacroix	Third street 41, Los Angeles, CA, Zip Code 1241278	5	12	1979
a9	Courteney Williams	Rainbow st. 11, White Plains NY, Zip code 49240	1	April	1987
a10	Sandy, Marvin	Winkler St. 91, Atlanta, GA, Zip Code 42355	10	Jan	1987

Tabla 3.1 Base de Datos A

ID	Nombre	Apellido	Dirección	Estado	FechaNacimiento
b1	Carlos	Ortiz Monreal	Fake st. 110, Seattle, Zip Code 114412	WA	March/10/1991
b2	Jonny	Baltasar	LaBelle street 18, Dickson, Zip Code 1241733	CA	Dec/18/1970
b3	Mildred	Smith	Byrne street 1117, Sacramento, Zip code 1241345	CA	Feb/11/1988
b4	Lynnette	Stockmann	McGuinnes street 18, Alanta, Zip code 42357	GA	Nov/19/1974
b5	Roger	Carlson	Ninth street 88, New York City, Zip Code 49870	NY	August/17/1957
b6	Roberto	Ortega	Ortega St. 457, Orlando, Zip Code 442123	FL	July/11/1978
b7	Charles	Harper	Figueroa St. 19, Bismark, Zip code 40986	NC	August/21/1971
b8	Julia	Stokman	McKinley street 16, Bismarck, Zip code 40987	NC	Feb/14/1980
b9	Karl-Heinz	Mueller	Dunhill Street 65, New York City, Zip Code 49876	NY	Oct-11-1978
b10	Martin	Smith	Steckelenburg St. 1145, Seattle, 114414	WA	24-Dec-1988

Tabla 3.2 Base de Datos B

En las tablas anteriores existen diferencias estructurales que imposibilitarán una comparación exitosa entre los diferentes campos de cada registro. Por ejemplo, en la base de datos **A** una dirección física se encuentra representada por el campo "Dir", mientras que en la base de datos **B** la misma dirección física se encuentra representada por los campos "Dirección" y "Estado". De esa manera, el preprocesamiento de datos tiene como finalidad asegurar que los atributos usados para el cruce de datos tengan la misma estructura y sus respectivos contenidos tengan el mismo formato y sigan las mismas restricciones.

El preprocesamiento de datos consiste en varias actividades. Cabe mencionar que entre mayor sea el éxito en dichas actividades, mejor será el resultado del Cruce de Datos. Así, de acuerdo a Christen [27] el preprocesamiento se compone de las siguientes actividades:

a) Remover palabras y caracteres no deseados

b) Expandir abreviaturas y corregir errores ortográficos

c) Segmentar los atributos iniciales en atributos bien definidos y consistentes

d) Verificar qué tan correctos son los valores de los atributos

El primer punto corresponde a eliminar caracteres que no son significativos o estorban para las demás fases de la Vinculación de Registros; estos caracteres usualmente son comas, puntos, comillas, entre otros. Sin embargo, estos caracteres se utilizan en ocasiones para hacer una segmentación entre valores de campos no segmentados. Por ejemplo, en la Tabla 3.1 se observa en los datos correspondientes a la Base de Datos A que en atributo “Nombre” del registro a1, el valor es “Miller, Robert”. Es evidente que se usó dicho campo para almacenar un nombre completo (Robert Miller) y que la coma funge como separadora del nombre propio y el apellido, por lo que no se puede simplemente eliminar la coma sino que se deben tener en consideración aspectos de este tipo.

El segundo punto se refiere a corregir errores ortográficos y expandir las abreviaturas presentes en los campos de las tablas. Este tópico se puede abordar mediante el uso de diccionarios o tablas de búsqueda.

El tercer punto se refiere a que se deben descomponer en campos más pequeños y consistentes los campos presentes en las bases de datos cuando esto sea posible. Así, en la Tabla 3.1, podemos observar que el campo “Nombre” (Usado para el nombre completo) puede segmentarse en “Nombre” y “Apellido”; más campos pueden ser segmentados de dicha manera tanto en la Base de Datos **A** como en la **B**. La salida de dichas segmentaciones produce atributos bien definidos y consistentes. Dicha segmentación ayudará a tener tablas compatibles en la fase de comparación de registros en el proceso de Vinculación de Registros.

El cuarto y último punto consiste en verificar que tan correctos o ciertos son los valores que poseen los campos de las tablas. Un enfoque para solventar este problema sería la consulta de bases de datos externas para verificar la validez de una dirección al revisar, por ejemplo, si la dirección que se muestra es válida para el código postal dado. Dicha verificación podría incluso permitir que se corrigiesen los valores. Sin embargo, pueden existir dificultades para realizar lo anterior (como el obtener acceso a la Base de Datos que nos permita realizar las verificaciones). Además, debido a la naturaleza dinámica de los datos, se podrían obtener correcciones erróneas.

Una vez mencionadas las cuatro actividades de preprocesamiento de datos, es importante hacer notar que es posible que no todas se apliquen debido a las dificultades inherentes a dichas actividades (por ejemplo el verificar si una dirección es válida para un código postal dado).

Tras la fase de preprocesamiento se debe obtener como salida fuentes de datos idóneas para la Vinculación de Registros. Las Tablas 3.3 y 3.4 que se muestran a continuación son el resultado de efectuar la fase de preprocesamiento de datos sobre las Bases de Datos **A** y **B** anteriormente mostradas en las Tablas 3.1 y 3.2; ambas Tablas serán referidas posteriormente.

ID	Nombre	Apellido	Calle	Num	Ciudad	Estado	C.P.	DiaNac	MesNac	ANac
a1	Robert	Miller	Hopeview	3	New York City	NY	49876	10	2	1989
a2	Katherine	Clinton	Evergreen	17	White Plains	NY	49237	23	1	1980
a3	John	Baltazar	LaBelle	18	Dixon	CA	1241732	18	12	1970
a4	Norman	Harper	Byrne	88	Sacramento	CA	1241345	14	3	1990
a5	Lynette	Stockman	McGinness	16	Atlanta	GA	42354	14	11	1974
a6	Franklin	Reagan	Robertson	65	Sacramento	CA	1241345	21	1	1981
a7	Martin	MacDonald	White	71	Atlanta	GA	42350	9	1	1991
a8	Robert	Delacroix	Third	41	Los Angeles	CA	1241278	5	12	1979
a9	Courteney	Williams	Rainbow	11	White Plains	NY	49240	1	4	1987
a10	Marvin	Sandy	Winkler	91	Atlanta	GA	42355	10	1	1987

Tabla 3.3 Base de Datos A después del preprocesamiento

ID	Nombre	Apellido	Calle	Num	Ciudad	Estado	C.P.	DiaNac	MesNac	ANac
b1	Carlos	Ortiz Monreal	Fake	110	Seattle	WA	114412	10	3	1991
b2	Jonny	Baltasar	LaBelle	18	Dickson	CA	1241733	18	12	1970
b3	Mildred	Smith	Byrne	1117	Sacramento	CA	1241345	11	2	1988
b4	Lynnette	Stockmann	McGuinnes	18	Alanta	GA	42357	19	11	1974
b5	Roger	Carlson	Ninth	88	New York City	NY	49870	17	8	1957
b6	Roberto	Ortega	Ortega	457	Orlando	FL	442123	11	7	1978
b7	Charles	Harper	Figueroa	19	Bismark	NC	40986	21	8	1971
b8	Julia	Stokman	McKinley	16	Bismark	NC	40987	14	2	1980
b9	Karl-Heinz	Mueller	Dunhill	65	New York City	NY	49876	11	10	1978
b10	Martin	Smith	Steckelenburg	1145	114414	WA	40017	24	12	1988

Tabla 3.4 Base de Datos B después del preprocesamiento

Se puede observar en las Tablas 3.3 y 3.4 que ahora las Bases de Datos **A** y **B** poseen un formato similar en cuanto a sus atributos y valores. Los beneficios que lo anterior implica para el proceso de Vinculación de Registros resultarán evidentes en la fase de comparación de registros.

3.3.- Indexado

Una vez que se ha realizado el preprocesamiento de los datos a cruzar, los registros están listos para ser comparados entre sí para determinar su similitud. Sin embargo, si se procediera inmediatamente a realizar lo anterior, se tendría el problema de que cada registro debe compararse con todos y cada uno de los demás registros pertenecientes a la fuente de datos sobre la cual se desea realizar el proceso de Vinculación de Registros.

Por ejemplo, si se quiere realizar una Vinculación de Registros de una base de datos **A** con una base de datos **B**, el número de comparaciones a realizar está dado por |**A**| x |**B**|, debido a que todos los elementos de la base de datos **A** pueden ser duplicados de cualquier elemento de otra base de datos **B**, lo que degenera en que se comparen todos los elementos de la base de datos **A** con todos aquellos pertenecientes a la base de datos **B**.

Por otro lado, si se realiza un proceso de Vinculación de Registros de una única base de datos **A** (lo que recibe el nombre de Detección de Duplicados, como ya se mencionó con anterioridad), el número de comparaciones a realizar estaría dado por **A**| x (|**A**| -1) / 2 [24]; dado que cada registro puede ser el duplicado de otro, es necesario comparar todas las parejas de registros en **A**, lo que degenera en complejidad cuadrática [28].

Otro punto a considerar es que, en una base de datos dada, el número de datos duplicados es usualmente mucho menor al número de datos originales [29]. Por ejemplo, de acuerdo a la American Health Information Maintenance Association, en las bases de datos del Registro Maestro de Pacientes (Master Patient Index) con más un millón de registros, el promedio de registros duplicados es el 9.4% del total de registros [30].

De esa manera, dado que la mayoría de los registros son no duplicados, la mayoría de las comparaciones que se realizarían al cruzar datos serían comparaciones entre registros no duplicados [31]; debido a esto, la reducción del número de las mismas en vital.

Suponga que se desea vincular las bases de datos **A** y **B**, cada una con 1,000,000 registros y que una vez que se ha realizado el preposesamiento de ambas

fuentes de datos se desea proceder directamente a realizar las comparaciones entre los registros de ambas bases de datos. Si se procediese de esa manera, el número de comparaciones a realizar sería de 1,000,000,000,000. Así, suponiendo que una comparación se realiza en 10 microsegundos, el total de las comparaciones se realizaría en 116 días [32]. Ahora, si el tamaño de las bases de datos fuera significativamente mayor a 1,000,000 registros, lo más probable es que no exista poder de cómputo suficiente para realizar tales cálculos [33].

Para evitar lo anterior es que se realiza el proceso de indexado, el cual tiene la finalidad de agrupar en bloques a los registros que posiblemente son duplicados entre sí. Dicho de otra manera, el indexado funciona como un filtro que agrupa en un bloque a valores o registros similares. Es necesario entonces, el seleccionar uno o varios atributos de los registros para a partir de ahí determinar en qué bloque o bloques se insertarán los registros. Dichos atributos reciben el nombre de *llave de bloque* (*blocking key* en inglés); a partir de los valores en los campos que integren la llave de bloque se generará un *valor de llave de bloque* (conocido como *blocking key value* en inglés), el cual determinará en que bloque o bloques se inserta el registro para su posterior comparación.

De acuerdo a lo anterior, el proceso de comparación de registros se realiza bloque a bloque; es decir, solamente se comparan entre sí los registros que se encuentren agrupados en el mismo bloque. Entonces, la selección de la o las columnas que integrarán el índice es crucial debido a que dicha selección debe permitir que se agrupen en cada bloque a registros similares entre sí y diferentes a los que integran los demás bloques. Ahora, la similitud entre los campos de los registros puede ser fonética (qué tan similares suenan), visual (qué tan similares son las cadenas de caracteres que los componen) o numérica (qué tan cercanos son dos valores numéricamente). De ese modo, es posible aplicar funciones de codificación fonética a alguno o a todos los campos de la llave de bloque con el fin de asegurar que los registros con sonido similar se inserten en el mismo bloque aunque tengan alguna variación tipográfica entre ellos [32]. El concepto de codificación fonética se explicará a detalle más adelante.

Considere las Tablas 3.3 y 3.4 correspondientes a las Bases de Datos **A** y **B** respectivamente; dichas Bases de Datos ya fueron preprocesadas pero si se procediera directamente a la comparación de registros, el número de comparaciones sería de cien debido a la complejidad cuadrática, a pesar de que hay registros que claramente no se refieren a la misma entidad. Por ende, es posible realizar un proceso de indexado escogiendo en este caso al campo “Apellido” como miembro único de la llave de bloque. Además, es posible aplicar una codificación fonética para identificar a aquellos valores que posean un sonido

similar e introducirlos en el mismo bloque de registros a pesar de tener diferencias tipográficas. En la siguiente sección se describirá la codificación fonética.

3.3.1.- Codificación fonética

La codificación fonética es el proceso mediante el cual toma una cadena de entrada (la cual usualmente corresponde a un nombre o apellido) y a partir de ella producir un código o valor asociado al sonido que produce dicha cadena al pronunciarse.

Para mejorar el proceso de indexado, a un campo dado que forme parte de la llave de bloque, se le puede aplicar una codificación fonética. Esto se hace debido a que en la codificación fonética, valores que posean un sonido similar generan la misma clave tras codificarlos, lo que permite que campos escritos de manera diferente pero con un sonido similar puedan contribuir con la misma clave al valor de la llave de bloque como ya se mencionó anteriormente.

Existen una serie de algoritmos de codificación fonética; una gran parte de ellos se diseñaron para trabajar principalmente sobre cadenas en lengua inglesa, por lo que pueden no arrojar resultados convenientes al utilizarlos el español o algún otro lenguaje que no sea el inglés. Lo anterior debe mantenerse en mente a la hora de escoger algún algoritmo de codificación fonética para el proceso de indexado.

Afortunadamente existen métodos de codificación fonética que están orientados a trabajar con otras lenguas además de la inglesa. Entre dichos algoritmos se encuentra el “Double Metaphone”, el cual se encuentra implementado en el sistema Febrl. A continuación se describen los algoritmos de codificación fonética Soundex y Double Metaphone.

3.3.1.1.- Soundex

El algoritmo de codificación fonética Soundex fue patentado en 1918 por Robert C. Russell [34]. Este algoritmo es muy utilizado actualmente y muchos de los algoritmos de codificación fonética utilizados hoy en día están basados o son mejoras de Soundex [35].

Soundex toma la cadena de entrada y provee como salida un código fonético consistente de una letra y tres dígitos. La letra en cuestión corresponde al primer carácter de la cadena que estemos codificando, mientras que los tres dígitos se

calculan a partir del resto de la cadena. De esa manera, la codificación fonética de una cadena se calcula con Soundex siguiendo el siguiente algoritmo:

1) Se conserva sin codificar la primera letra de la cadena.
2) Después de la primera letra, el resto de los caracteres de la cadena se codifican de acuerdo a la Tabla 3.5.
3) Si hay varios dígitos iguales adyacentes, se mantiene sólo uno de ellos y resto de dichas repeticiones adyacentes se borran.
4) Se borran todas los ceros
5) Si al código que tenemos hasta este momento le faltan dígitos para tener los tres dígitos correspondientes a un código Soundex, le agregamos los ceros necesarios a la derecha del código existente. Por otro lado, si el código es mayor a tres dígitos, simplemente se trunca a tres dígitos.

Caracter	Codificación
A,E,I,O,U,H,Y,W	0
B,F,P,V	1
C,G,J,K,Q,S,X,Z	2
D,T	3
L	4
M,N	5
R	6

Tabla 3.5 Reglas de codificación de Soundex

De esa manera, al aplicar el algoritmo anterior a la cadena “Ortiz” se obtiene lo siguiente:

ORTIZ -> O6302 ->O632

3.3.1.2.- Double Metaphone

El algoritmo Double Metaphone fue publicado por Lawrence Phillips en el año 2000 [36] como una mejora del algoritmo Metaphone que él mismo había desarrollado en 1990. Una de las ventajas más significativas de este algoritmo frente a Soundex es que está diseñado para trabajar con otras lenguas además de la inglesa.

Double Metaphone es un algoritmo notablemente complejo. Aplica una larga serie de reglas sobre la cadena de entrada y toma en cuenta las diferentes pronunciaciones que puede tener un carácter dado de acuerdo a su posición con respecto a los demás caracteres. Por ejemplo, el algoritmo considera que la letra C puede tener un el sonido correspondiente a la letra K o a la letra S de acuerdo a la posición del carácter C con respecto a los demás caracteres [36].

Una característica muy importante de la codificación por Double Metaphone es que el algoritmo entrega dos codificaciones fonéticas a partir de la cadena de entrada; esto permite mejorar el índice de detección de palabras similares con variaciones ortográficas. Cabe mencionar que las codificaciones entregadas por Double Metaphone se componen únicamente de letras y que el algoritmo puede entregar para la cadena de entrada dos codificaciones iguales si así lo determina conveniente.

La Tabla 3.6 muestra las codificaciones correspondientes al algoritmo Double Metaphone para una serie de nombres de diferentes orígenes étnicos. Se observa que con excepción de las codificaciones correspondientes a “Qualgliarella” y “Kuczewski”, las dos codificaciones de salida fueron iguales.

Entrada	Codificación 1	Codificación 2
Ortiz	ARTS	ARTS
Miller	MLR	MLR
Mueller	MLR	MLR
Quagliarella	KKLR	KLRL
LeClerk	LKLR	LKLR
Hernández	HRNN	HRNN
Stockmann	STKM	STKM
Stokman	STKM	STKM
Ricci	RX	RX
Kuczewski	KSSK	KXFS

Tabla 3.6 Codificación Double Metaphone para apellidos de diferente origen étnico

3.3.2.- Ejemplo de la fase de indexado

Considere las Bases de Datos **A** y **B** mostradas en las Tablas 3.3 y 3.4 respectivamente que se obtuvieron como salida del preprocesamiento de datos. El siguiente paso a ejecutar sobre ellas es el indexado para reducir los problemas de complejidad computacional mencionados anteriormente. Aplicando la codificación fonética Soundex sobre la Base de Datos A se obtiene el siguiente resultado:

ID	Apellido	ValorLlaveDeBloque
a1	Miller	M-460
a2	Clinton	C-453
a3	Baltazar	B-432
a4	Harper	H-616
a5	Stockman	S-325
a6	Reagan	R-250
a7	MacDonald	M-235
a8	Delacroix	D-426
a9	Williams	W-452
a10	Sandy	S-530

Tabla 3.7 Resultado de aplicar la codificación fonética Soundex al atributo “Apellido”, integrante único de la llave de bloque, en la Base de Datos A

Aplicando la misma codificación en la Base de Datos B se obtiene el siguiente resultado:

ID	Apellido	ValorLlaveDeBloque
b1	Ortiz Monreal	O-632
b2	Baltasar	B-432
b3	Smith	S-530
b4	Stockmann	S-325
b5	Carlson	C-642
b6	Ortega	O-632
b7	Harper	H-616
b8	Stokman	S-325
b9	Mueller	M-460
b10	Smith	S-530

Tabla 3.8
Resultado de aplicar la codificación fonética Soundex al atributo “Apellido”, integrante único de la llave de bloque, en la Base de Datos B

El último paso del proceso de indexado sería el agrupar a los registros en bloques de acuerdo al valor de llave de bloque obtenido anteriormente. Los bloques resultantes para la Vinculación de Registros de las Bases de Datos **A** y **B** mostradas en las Tablas 3.3 y 3.4 son los siguientes:

ValorLlaveDeBloque	Registros
M-460	(a1,b9)
B-432	(a3,b2)
H-616	(a4,b7)
S-325	(a5,b4) , (a5,b8)
S-530	(a10,b3) , (a10,b10)

Tabla 3.9 Bloques de registros candidatos a duplicados obtenidos a partir de aplicar la codificación fonética Soundex al atributo "Apellido" en las Bases de Datos A y B. Esta tabla será referida al ejemplificar la fase de comparación de pares de registros

Como ya se mencionó con anterioridad, los bloques de registros que sean la salida de la fase de indexado agrupan a las parejas de registros que sean candidatos a ser duplicados de acuerdo a su valor de llave de bloque. Dichos bloques serán la entrada de la fase de comparación y reducirán la complejidad computacional cuadrática que se tendría sin el indexado. En la siguiente sección se describe la fase de comparación.

3.4.- Comparación de registros

Una vez que el proceso de indexado ha terminado, se tienen bloques formados por registros candidatos a ser duplicados, lo cual reduce la complejidad cuadrática inherente a una comparación sin indexado como se mencionó anteriormente.

Considerando que los datos de las fuentes con las que se trabaja son proclives a poseer datos de mala calidad, la comparación entre diferentes fuentes de datos es un reto por sí mismo, pues incluso con las mejores prácticas de limpieza y estandarización de los datos se pueden tener datos diferentes pero que se refieren a la misma entidad. Lo anterior se debe a motivos tan variados como los errores tipográficos, la existencia de variaciones en los nombres, la actualización y obsolescencia de datos personales (por ejemplo un cambio de domicilio). Estos problemas de calidad en los datos implican que se pueden tener valores diferentes pero que se refieren a la misma entidad.

Lo anterior presenta el problema de determinar cuándo dos cadenas diferentes son lo suficientemente similares como para determinar que son coincidentes y

cuando son lo suficientemente diferentes para poder asegurar que dichas cadenas no corresponden a la misma entidad [37] [38]. Así, para resolver lo anterior, se deben usar funciones de comparación aproximada; dichas funciones deberán de tener como salida un número normalizado entre cero y uno, para a partir de tales cantidades poder establecer un criterio de si dos entidades son iguales o no a partir de la puntuación obtenida, la cual se conoce como valor de similitud [37].

De acuerdo a Christen [37], una función de comparación aproximada dada debe cumplir con las siguientes propiedades:

a) El resultado de comparar una entidad consigo misma deberá arrojar un valor de similitud de 1.

b) El resultado de comparar dos entidades “completamente diferentes” deberá arrojar un valor de similitud de 0.

c) El resultado de comparar dos entidades “aproximadamente similares” entre sí deberá arrojar un valor de similitud entre 0 y 1.

Considerando todo lo anterior, el proceso de comparación recibirá como entrada los bloques de registros que salen del proceso de indexado. Dichos bloques poseen, como ya se mencionó, a los registros que son candidatos a duplicados. Entonces se compararán tales registros para poder determinar qué tan similares son entre sí. Para ese propósito, se seleccionará una o varias columnas de los registros para comparar sus respectivos valores en las parejas de registros candidatos a ser duplicados. Se compararán los valores para esa misma columna en los dos registros con una función de comparación y esto se repetirá hasta que hayan sido procesadas todas las columnas que hayan sido seleccionadas para la comparación. Luego, todo el procedimiento anterior se repetirá para todas las parejas de vectores de los bloques creados por el proceso de indexado.

Es necesario el contar con funciones que permitan realizar una comparación aproximada de acuerdo al tipo de datos que estamos analizando y así poder otorgar como salida un conjunto de vectores de comparación, los cuales se procesarán en la fase de clasificación. A continuación se describen tanto la comparación exacta como diversas distancias de comparación aproximada; estas últimas tienen como salida un valor de similitud representativo de qué tan parecidas entre sí son las variables de entrada.

3.4.1.- Comparación exacta

Esta es la forma más básica de comparación. En ella la función de comparación regresa un valor numérico (usualmente cero o uno) o lógico (verdadero o falso) que indica si los elementos comparados son exactamente iguales o no.

También es posible que se realice un truncamiento de la cadena a comparar y de esa manera aplicar una comparación exacta sobre n caracteres en una posición dada en las cadenas a comparar.

Debido a la naturaleza de los datos, la comparación exacta es inútil en la mayoría de las circunstancias debido a las inconsistencias y errores tipográficos presentes en los datos envueltos en el proceso de Vinculación de Registros. Si tras la fase de preprocesamiento de datos se tienen campos de los que se tenga certeza total, la comparación exacta podría tener utilidad. Sin embargo, como esta situación difícilmente se presenta, es necesario tener métricas de comparación aproximada.

3.4.2.- Distancia de Jaro

La Distancia de Jaro es una métrica para la comparación aproximada de cadenas inventada por Matthew Jaro [39]. La distancia de Jaro se calcula al contar el número de caracteres coincidentes que son comunes para ambas cadenas en una longitud igual a la mitad de la longitud de la cadena más larga. La métrica de Jaro también considera el número de transposiciones de caracteres presentes en las cadenas a comparar [40].

Una transposición se define de la siguiente manera. Considere a s_1' como una cadena formada por los caracteres comunes a ambas cadenas en el orden que aparecen en s_1. De la misma manera, considere a s_2' como una cadena formada por los caracteres comunes a ambas cadenas en el orden que aparecen en s_2. De esa manera, existe una transposición cuando para las cadenas s_1' y s_2', el caracter observado en el índice i es diferente.

Si no hay caracteres coincidentes dentro de la longitud permitida, la Distancia de Jaro es igual a cero. En caso contrario, la Distancia de Jaro está dada por la siguiente fórmula [41]:

$$\mathbf{sim}_{\text{jaro}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{1}{3}\left(\frac{\mathbf{c}}{|\mathbf{s}_1|} + \frac{\mathbf{c}}{|\mathbf{s}_2|} + \frac{\mathbf{c}-\mathbf{t}}{|\mathbf{c}|}\right)$$

Dónde:

$|s_1|$ es la longitud de la cadena 1

$|s_2|$ es la longitud de la cadena 2

c es el número de caracteres coincidentes en dentro de la longitud permitida (aquella menor a la longitud de la cadena más larga dividida entre dos)

t es el número de transposiciones en las cadenas dividido entre dos.

La siguiente tabla muestra valores de Distancia de Jaro para diferentes cadenas, junto con los valores de sus parámetros [42]:

Cadena 1	Cadena 2	c	t	Distancia de Jaro
Shackleford	Shackelford	11	1	0.9697
Nichleson	Nichulson	8	0	0.9259
Jones	Johnson	4	0	0.7905
Massey	Massie	5	0	0.8889
Jeraldine	Geraldine	8	0	0.9259
Michelle	Michael	6	0	0.8690

Tabla 3.10 Valores de Distancia de Jaro para diferentes cadenas correspondientes a nombres y apellidos

3.4.3.- Distancia de Jaro-Winkler

La Distancia de Jaro-Winkler fue publicada por William E. Winkler, en 1990 [43]. Esta métrica se basa en la Distancia de Jaro y le suma otros parámetros para mejorar la comparación de las cadenas.

La Distancia de Winkler se calcula de acuerdo a la siguiente fórmula [44]:

$$\mathbf{sim_{winkler}(s_1, s_2) = sim_{jaro}(s_1, s_2) + (1.0 - sim_{jaro}(s_1, s_2))\ \frac{p}{10}}$$

Dónde:

sim_{jaro} es la Distancia de Jaro

p es el número de caracteres coincidentes al inicio de la cadena. El valor permitido de p es de uno a cuatro.

La siguiente tabla muestra valores de Distancia de Winkler para diferentes cadenas, junto con los valores de sus parámetros [42]:

Cadena 1	Cadena 2	c	t	p	Distancia de Jaro-Winkler
Shackleford	Shackelford	11	1	4	0.9818
Nichleson	Nichulson	8	0	4	0.9556
Jones	Johnson	4	0	2	0.8324
Massey	Massie	5	0	4	0.9333
Jeraldine	Geraldine	8	0	0	0.9259
Michelle	Michael	6	0	4	0.9214

Tabla 3.11 Valores de Distancia de Jaro-Winkler para diferentes cadenas

3.4.4.- Comparación por Distancia de Levenshtein

La Distancia de Levenstein se define como qué tan distantes son dos cadenas entre sí de acuerdo al número de operaciones de edición que se tendrían que realizar sobre una cadena para que fuese exactamente igual a la otra [45].

En la Distancia de Levenshtein las operaciones permitidas de edición son la inserción (se añade un caracter extra), la modificación (se cambia un caracter por otro) y el borrado (se elimina un caracter de la cadena). A estas operaciones se les asigna un costo para el cálculo de la Distancia de Levenshtein; lo más usual es que a todas se les asigne un costo de uno.

Cadena 1	Cadena 2	Distancia de Levenshtein
Shackleford	Shackelford	2
Nichleson	Nichulson	2
Jones	Johnson	4
Massey	Massie	2
Jeraldine	Geraldine	1
Michelle	Michael	3

Tabla 3.12 Distancias de Levenshtein para diferentes cadenas correspondientes a nombres y apellidos

Al analizar la Tabla 3.12 se observa que si bien la Distancia de Levenshtein brinda información de que tan distantes entre sí son las cadenas comparadas. Sin embargo, para poder usar esa información como una métrica eficaz de similitud, es necesario realizar un ajuste para obtener valores de similitud normalizados entre cero y uno. Así, la métrica de similitud por Distancia de Levenshtein se obtiene de acuerdo a la siguiente fórmula [46]:

$$\mathbf{sim}_{\mathbf{levenshtein}}(s_1, s_2) = \mathbf{1.0} - \frac{\mathbf{dist}_{\mathbf{levenshtein}}(s_1, s_2)}{\mathbf{max}(s_1, s_2)}$$

Dónde:

s_1 es la cadena 1 a comparar

s_2 es la cadena 2 a comparar

$|s_1|$ es la longitud de la cadena 1

$|s_2|$ es la longitud de la cadena 2

Aplicando la fórmula a las cadenas de la Tabla 3.12 se obtienen las siguientes métricas de similitud por Distancia de Levenshtein:

Cadena 1	Cadena 2	Similitud por Levenshtein
Shackleford	Shackelford	0.8181
Nichleson	Nichulson	0.7777
Jones	Johnson	0.4285
Massey	Massie	0.6667
Jeraldine	Geraldine	0.8889
Michelle	Michael	0.625

Tabla 3.13 Métricas de similitud por Distancia de Levenshtein para cadenas correspondientes a nombres y apellidos

3.4.5.- Ejemplo de la fase de comparación

Considere los bloques de registros candidatos a similitud mostrados en la Tabla 3.9 obtenidos tras aplicar el proceso de indexado a las Bases de Datos A y B mostradas en las Tablas 3.3 y 3.4 respectivamente. El paso siguiente a realizar en el proceso de Vinculación de Registros la fase de comparación. Para este ejemplo se seleccionan las columnas “Nombre”, “Calle”, “Num”, “Ciudad”, “Estado”, “C.P.”, “DiaNac”, “MesNac” y “ANac”; los valores de similitud para cada uno de estos atributos integrarán los vectores que la fase de comparación otorga como salida.

Para este ejemplo se selecciona la comparación aproximada por Distancia de Levenshtein para todas las columnas seleccionadas para la comparación.

rec_id1	rec_id2	Nombre	Calle	Num	Ciudad	Estado	C.P.	DiaNac	MesNac	ANac
a1	b9	0.2	0.125	0	1	1	1	0.5	0	0.5
a3	b2	0.6	1	1	0.57142857	1	0.85714	1	1	1
a4	b7	0.14285	0.125	0	0.1	0	0.14285	0	0	0.5
a5	b4	0.875	0.777777	0.5	0.85714286	1	0.8	0.5	1	1
a5	b8	0	0.555555	1	0	0	0.2	1	0	0.5
a10	b3	0.14285	0.142857	0.25	0.4	0.5	0.42857	0.5	0	0.75
a10	b10	0.83333	0.307692	0.25	0	0.5	0.2	0	0.5	0.75

Tabla 3.14 Vectores de comparación generados al aplicar la función de similitud por Distancia de Levenshtein a los registros seleccionados agrupados por bloques durante el proceso de indexado. Esta tabla será referida posteriormente para ilustrar la fase de clasificación

La Tabla 3.14 muestra la salida de la fase de comparación. Se han generado vectores con valores de comparación aproximada por la Distancia de Levenshtein. Estos valores indican cuán similares son los valores de cada atributo comparado para una pareja de registros candidatos a ser duplicados. Este conjunto de vectores será la entrada para la fase de clasificación, donde se determinará qué parejas de vectores son duplicados, cuáles no lo son y cuáles podrían serlo. La fase de clasificación de proceso de Vinculación de Registros se describe a continuación.

3.5.- Clasificación

Tras la finalización de la fase de comparación, los vectores de comparación generados por esta última pasan a la fase de clasificación, donde, de acuerdo al grado de similitud obtenido mediante las comparaciones, se determinará si la pareja de registros asociados a un vector de comparación dado, son duplicados, no lo son o si se requiere de una posterior revisión para determinar su estado final. Los métodos de clasificación se dividen principalmente en clasificación supervisada y clasificación no supervisada.

En los métodos supervisados es necesario poseer un conjunto de entrenamiento para poder utilizar el método. Dicho conjunto de entrenamiento son vectores de comparación que ya han sido clasificados de manera correcta. Al tener ya un estado correcto y conocido, se usan para entrenar al clasificador para que este pueda entonces operar sobre vectores de entrada que tengan un estado de clasificación desconocido y poder así clasificarlos de acuerdo a lo que el método clasificador observó durante su etapa de entrenamiento. Sin embargo, un problema con lo anterior es que los vectores de entrenamiento deben haberse obtenido usando las mismas funciones de comparación que se usarán para comparar los vectores que poseen un estado desconocido. Además, este conjunto de entrenamiento debe ser de alta calidad y debe cubrir un gran número de variantes y posibilidades para que el entrenamiento del clasificador sea satisfactorio [47]. Sin embargo, obtener un conjunto de entrenamiento con las características anteriormente mencionadas es difícil [48].

Los clasificadores no supervisados, por su parte, son aquellos que no requieren de un conjunto de entrenamiento para su operación. Estos métodos operan simplemente con los valores de similitud existentes en los vectores de comparación obtenidos en la fase de comparación y con los parámetros que el clasificador requiera, los cuales son proporcionados por el usuario del algoritmo.

Al terminar la fase de clasificación se tendrá como salida la lista de estados de clasificación para cada vector de comparación. Como ya se mencionó, algunos clasificadores pueden otorgar estados indeterminados de clasificación a los vectores; estos requerirán de un proceso de Revisión Secretarial para determinar su estado final de clasificación. Si no existen vectores con clasificación indeterminada, su puede proceder directamente al cálculo de métricas para la evaluación de la calidad del proceso de Vinculación de Registros.

En las siguientes secciones se describen algoritmos de clasificación generalmente utilizados para la Vinculación de Registros.

3.5.1.- Clasificador por umbral de similitud sumada

La manera más sencilla de clasificar los vectores de comparación es de acuerdo a la suma de los valores correspondientes a cada dimensión de dichos vectores. La suma de estos valores se conoce como *valor de similitud sumada*. Se establece entonces un umbral t para clasificar a los vectores de comparación de registros en coincidentes y no coincidentes de la siguiente manera [49]:

a) $SimSum(r_i, r_j) \geq T \rightarrow (r_i, r_j)$ es coincidente

b) $SimSum(r_i, r_j) < T \rightarrow (r_i, r_j)$ es no coincidente

Dónde:

$SimSum(r_i, r_j)$ es el valor de similitud sumada para el vector de comparación correspondiente a los registros r_i y r_j

T es el umbral

Alternativamente, se pueden establecer dos umbrales (superior e inferior) para permitir la que los vectores de comparación puedan ser etiquetados como coincidentes, no coincidentes y posiblemente coincidentes de la siguiente manera

a) $SimSum(r_i, r_j) \geq T_u \rightarrow (r_i, r_j)$ es coincidente

b) $T_l < SimSum(r_i, r_j) < T_u \rightarrow (r_i, r_j)$ es posiblemente coincidente

c) $SimSum(r_i, r_j) \leq T_l \rightarrow (r_i, r_j)$ es no coincidente

Dónde:

SimSum(r_i, r_j) es el valor de similitud sumada para el vector de comparación correspondiente a los registros r_i y r_j

T_l es el umbral inferior

T_u es el umbral superior

La selección de valores para el o los umbrales tiene un impacto directo en la calidad de la clasificación. Si se establece un solo umbral y su valor es muy alto, es altamente probable que parejas de registros coincidentes se determinen como no coincidentes. Por otro lado, si el valor del único umbral es muy bajo, es probable que vectores correspondientes a pareja no coincidentes se clasifiquen erróneamente como coincidentes.

Si se decide utilizar dos umbrales y para el umbral inferior se establece un valor muy bajo mientras que para el superior se establece un umbral muy alto, entonces es altamente probable que la mayoría de los registros se clasifiquen como posiblemente coincidentes, lo que implica que se requiere una posterior revisión secretarial, con la consiguiente inversión en tiempo y recursos que ésta requiere.

De acuerdo a Christen, la clasificación por umbral de similitud sumada tiene las siguientes desventajas [50]:

a) Si los vectores de comparación usan valores normalizados, todas las similitudes de los atributos contribuyen con el mismo peso al valor de similitud sumada.

b) La información detallada de valores de similitud se pierde al momento de realizar la suma

La primera desventaja se refiere a que si en la fase de comparación se generaron valores normalizados para las diferentes dimensiones de los vectores de comparación, entonces todos los atributos comparados tendrán la misma importancia para determinar si los registros son duplicados o no. Esto es un problema porque, dependiendo de las características de los datos, pueden existir atributos más significativos que otros al momento de determinar el estado de clasificación de los registros (por ejemplo los apellidos de una persona); lo anterior se pierde si se utiliza un clasificador de similitud sumada con valores normalizados en los vectores de comparación.

La segunda desventaja se refiere a que al sumar los valores de similitud contenidos en los vectores de comparación se pierde, para el propósito de la

clasificación, los valores detallados que estos contenían. Esto puede permitir que registros con altamente disimilares tengan el mismo valor de similitud sumada que registros altamente similares. Este problema será referido en el Capítulo VI, donde se abordará la mejora propuesta al proceso de Vinculación de Registros.

Las ventajas principales de este método de clasificación son su simplicidad y su bajo costo computacional, pues se computan operaciones de suma para las dimensiones de todos los vectores.

3.5.2.- K medias

El algoritmo K medias fue publicado inicialmente por Stuart Lloyd en 1957 [51]. Es un algoritmo de búsqueda local que particiona un conjunto X de n vectores de entrada en k grupos o *clusters*. El número de grupos es un parámetro del algoritmo, por lo que se debe saber cuántos grupos se desea obtener del algoritmo antes de ejecutarlo.

El algoritmo K medias es el siguiente [52]:

0. Escoja arbitrariamente k centros $c_1, c_2, \ldots, c_k$
1. Para cada $1 \leq i \leq k$, establezca que el grupo C_i sea el conjunto de puntos en X que estén más cerca de c_i que de cualquier c_j para $i \neq j$
2. Para cada $1 \leq i \leq k$, establezca

$$\mathbf{c_i} = \frac{\mathbf{1}}{|\mathbf{C_i}|} \sum_{\mathbf{x} \in \mathbf{C_i}} \mathbf{x}$$

3. Repita los pasos 1 y 2 hasta que los clusters C_i y los centros c_i no cambien. El resultado será el conjunto X particionado en los grupos $C_1, C_2, \ldots, C_k$.

El algoritmo se basa en el cálculo de cada vector al grupo cuyo centro se encuentre más cercano. Entonces, tras la adición de un nuevo vector, se calcula de nuevo la media o centroide de cada grupo. Esto se repite hasta que los grupos y los centroides no cambian.

Las principales ventajas de este algoritmo es que en la práctica se ha observado que el número de iteraciones que debe ejecutar el algoritmo es usualmente menor al número de muestras que el algoritmo clasificará; sin embargo el algoritmo tiene

una cota inferior teórica exponencial de $\Omega(2^n)$ para datos con dos o más dimensiones [52].

Por otro lado, este algoritmo está expuesto que conforme los datos con los que se trabaja crecen en su número de dimensiones, es cada vez más complicado el obtener una distancia significativa durante el cálculo de las distancias entre los centroides y los vectores debido a la complejidad de un espacio de d dimensiones [53].

A continuación se ilustra gráficamente el algoritmo K medias para un conjunto X de doce vectores en un espacio euclidiano bidimensional con un valor de tres para k. La Figura 3.2 muestra el conjunto inicial de vectores, representados por cuadrados y sus centroides. La Figura 3.3 muestra la partición inicial de los vectores. La Figura 3.4 muestra el cálculo de la nueva media. Finalmente la Figura 3.5 muestra la partición final cuando el algoritmo ha concluido.

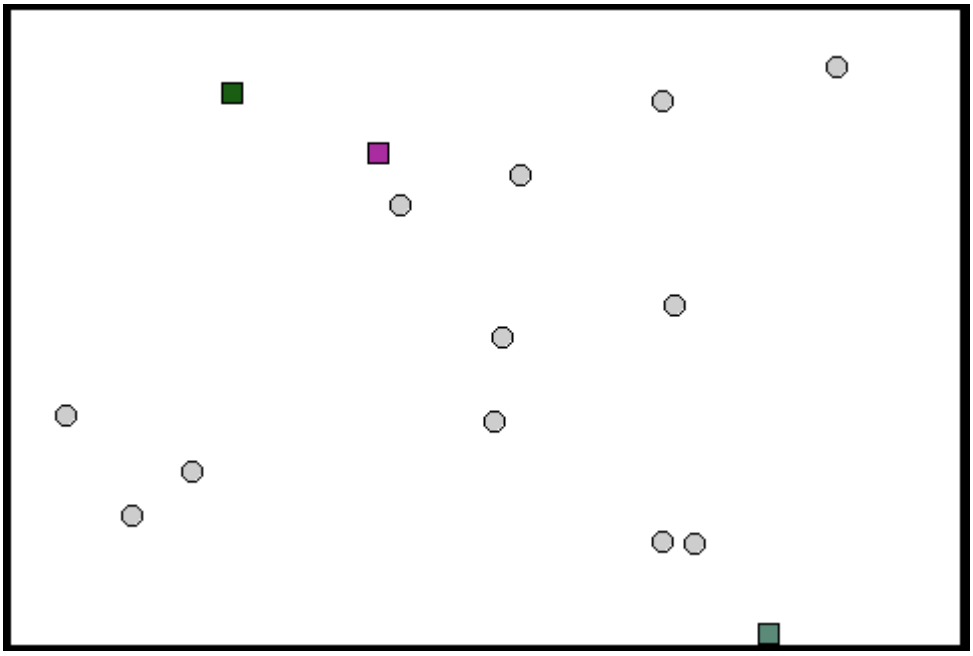


Figura 3.2 Paso 0 del Algoritmo K medias. Los vectores están indicados por círculos y los centroides por cuadrados

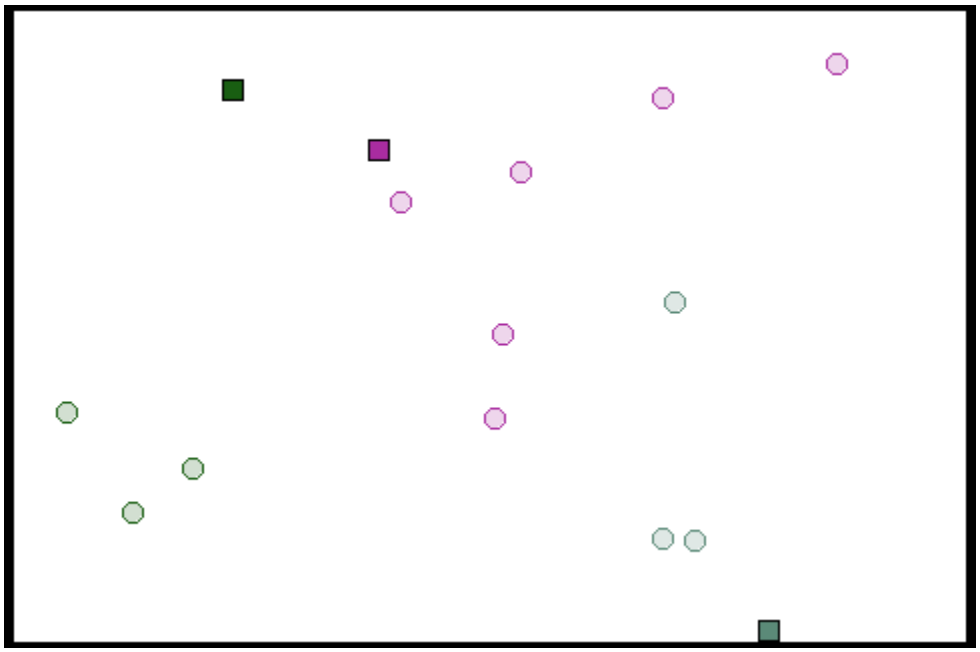


Figura 3.3 Paso 1 del algoritmo K medias. Partición con los centroides iniciales. Los grupos de vectores se indican con el color de su centroide

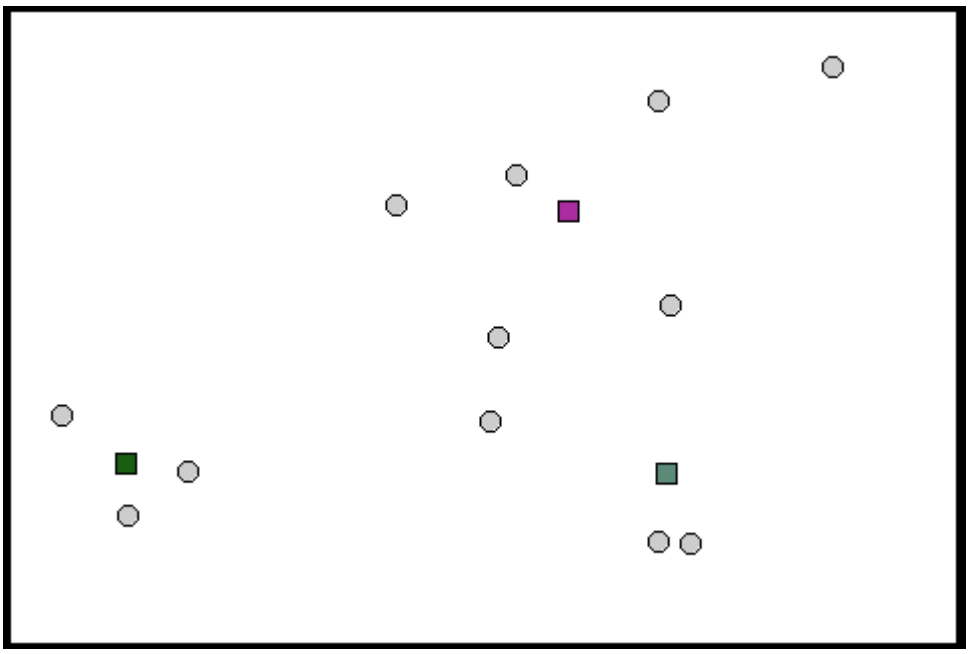


Figura 3.4 Paso 2 del algoritmo K medias. Cálculo de los nuevos centroides

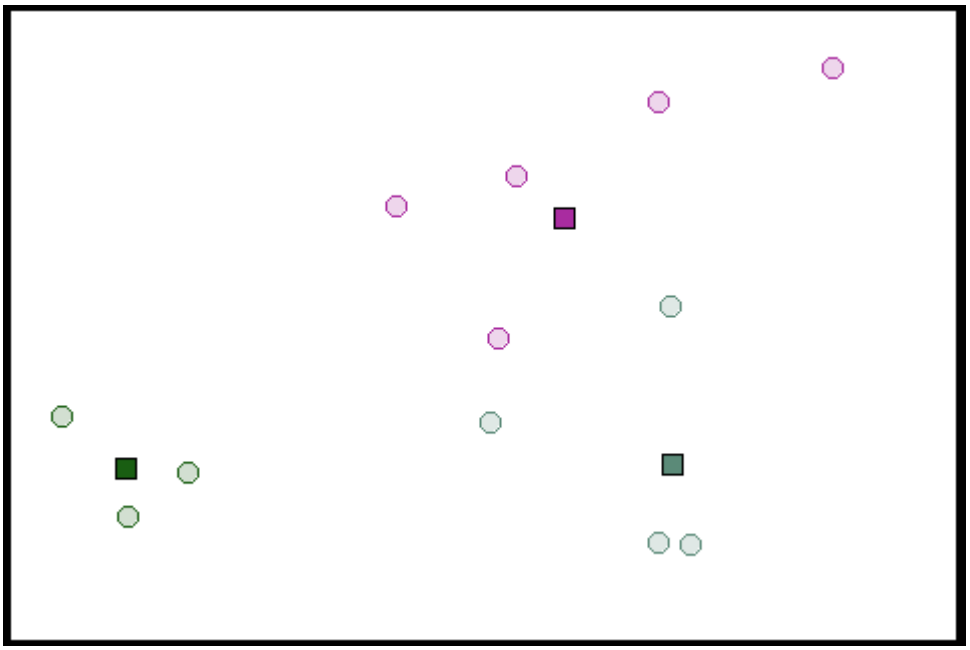


Figura 3.5 Paso 3 del algoritmo K medias. Partición con los nuevos centroides.

Las principales ventajas y desventajas del algoritmo K medias son las siguientes [54]:

Ventajas

a) El algoritmo K medias es simple y flexible

b) El algoritmo es fácil de comprender e implementar

Desventajas

a) El número k de grupos a obtener debe conocerse desde antes de ejecutar el algoritmo

b) Las agrupaciones obtenidas dependen de los centroides escogidos inicialmente, por lo que no hay garantía de una solución óptima.

3.5.3.- ISODATA

El algoritmo de agrupamiento ISODATA fue publicado en 1965 por Geoffrey Ball y David J. Hall [55]. Este algoritmo es una modificación a K medias pero tiene como ventaja que no es necesario el conocer de antemano el número de grupos a obtener. Esto se logra mediante parámetros provistos por el usuario del algoritmo que permiten la fusión de grupos en un grupo más grande, la división de un grupo en agrupamientos más pequeños y la eliminación de grupos con muy pocos elementos con la finalidad de obtener resultados más significativos. Esto permite que el número de agrupamientos que el algoritmo arroja sea variable.

3.5.4.- Farthest First

El algoritmo Farthest First fue publicado inicialmente por Teófilo González en 1985 [56]. Este método es simplemente una variable del algoritmo K medias. La diferencia ocurre al momento del cálculo de las medias o centroides y consiste en asignar el centroide para el primer grupo de manera aleatoria. Para el segundo centroide se escoge el punto que se encuentre más alejado de la primera media. Para los centroides faltantes se escoge el punto que esté más alejado de los centroides existentes hasta ese momento.

Entre las principales ventajas de Farthest First se encuentra que brinda una mejora en cuanto a la velocidad del agrupamiento con respecto a K medias debido a que se realiza un menor reasignamiento de centroides [57].

3.5.5.- Ejemplo de la fase de clasificación

Considere los vectores de comparación de registros generados en la fase de comparación que se muestran en la Tabla 3.14. Para este ejemplo esos vectores se clasificarán mediante clasificación por umbral de similitud sumada con un valor de 7 para el umbral inferior y de 7.5 para el umbral superior.

rec_id1	rec_id2	SimSum	Estado de clasificación
a1	b9	4.325	No coincidente
a3	b2	8.02857143	Coincidente
a4	b7	1.01071429	No coincidente
a5	b4	7.30992064	Posiblemente coincidente
a5	b8	3.25555556	No coincidente
a10	b3	3.11428572	No coincidente
a10	b10	3.34102564	No coincidente

Tabla 3.15 Valores de similitud sumada correspondientes a los vectores de comparación

De acuerdo a la Tabla 3.15, de todos los vectores de comparación de registros, únicamente el vector correspondiente a la pareja (a3,b2) tiene un valor de similitud sumada que es mayor o igual al valor del umbral superior; por ende dicho vector se clasifica como coincidente. Por otro lado el vector correspondiente a la pareja de registros (a5,b4) tuvo un valor superior al establecido para el umbral inferior e inferior al valor de umbral superior, por lo que dicho registro requiere posterior análisis mediante el proceso de Revisión Secretarial, el cual se abordará en la siguiente sección.

3.6.- Revisión Secretarial y Evaluación del Desempeño

Una vez que la fase de clasificación concluye, los vectores de comparación de registros han sido etiquetados como coincidentes, no coincidentes y posiblemente coincidentes. Éstos últimos requieren de un posterior análisis para poder determinar su estado final de clasificación. Dicho análisis se realiza mediante el proceso de *Revisión Secretarial* (conocido en inglés como Clerical Review). Una vez que el proceso de Revisión Secretarial ha concluido se tendrán estados de clasificación para todos los vectores de comparación de registros.

Por otra parte, ya que todos los vectores hayan sido clasificados, es posible el cálculo de métricas para determinar la calidad de los resultados arrojados por el

proceso de Vinculación de Registros o de Detección de Duplicados según sea el caso. Es importante notar que el cálculo de dichas métricas requiere que se conozca el estado verdadero de clasificación de los registros, es decir cuales son duplicados y cuales no lo son, para poder compararlos con los resultados obtenidos prácticamente.

3.6.1.- Revisión Secretarial

Como se ha mencionado con anterioridad, tras la fase de clasificación de registros, es posible que haya vectores de comparación clasificados como posiblemente coincidentes. Estos vectores deben enviarse al proceso de Revisión Secretarial para poder determinar su estado final.

El proceso de Revisión Secretarial consiste en el análisis de los registros que son posiblemente coincidentes para poder llegar a un veredicto final sobre su estado de clasificación. Esto requiere que un individuo, preferentemente alguien con conocimiento de los datos en cuestión, inspeccione cuidadosamente los campos correspondientes a los registros candidatos a duplicación. Tras dicha inspección se determinará un estado de clasificación de acuerdo a lo que el revisor haya observado en los datos.

Si bien dicha inspección es necesaria, lo anterior tiene, de acuerdo a Christen, las siguientes desventajas [58]:

a) Es difícil determinar si una pareja de registros (r_i, r_j) corresponde a duplicados si solo se observan los registros r_i y r_j, por lo que es necesario observar registros similares. Esto implica que la revisión manual es difícil incluso para un experto en Vinculación de Registros y en los datos en cuestión.

b) La clasificación manual depende de la apreciación humana, la cual es sensible a factores como el tedio, el estrés, el nivel de concentración y el número de registros a clasificar manualmente.

Para resolver estos problemas se puede hacer que más de una persona revise los registros, sin embargo esto prolonga el proceso de Revisión Secretarial e incrementa los costos del proyecto. Otra opción es, de ser posible, la consulta de bases de datos externas para obtener información que contribuya a emitir una clasificación final. Otro enfoque para mejorar el proceso de Revisión Secretarial es

el *Muestreo de Aceptación* (Acceptance Sampling en inglés), el cual muestra resultados promisorios [59].

Una ventaja del proceso de Revisión Secretarial es que los veredictos generados en este proceso se pueden usar como datos de entrenamiento para una clasificación supervisada, sin embargo se debe tener en consideración las inconsistencias asociadas a una clasificación humana [60].

3.6.1.1.- Ejemplo de la Revisión Secretarial

Considere la Tabla 3.15 anteriormente mostrada. En ella se indica que tras la salida de la fase de clasificación en la que se realizó una clasificación por similitud sumada, se determinó que el vector de comparación correspondiente a la pareja de registros (a5,b4) se clasificó como posiblemente coincidente, por lo que requiere de una revisión manual para determinar su estado de clasificación final.

ID	Nombre	Apellido	Calle	Num	Ciudad	Estado	C.P.	DiaNac	MesNac	ANac
a5	Lynette	Stockman	McGinness	16	Atlanta	GA	42354	14	11	1974
b4	Lynnette	Stockmann	McGuinnes	18	Alanta	GA	42357	19	11	1974

Tabla 3.16 Revisión Secretarial de la pareja de registros (a5,b4)

La Tabla 3.16 muestra a los registros (a5,b4) para su inspección manual. Debido a que los registros son altamente similares, se puede afirmar, con un alto grado de probabilidad, que corresponden a la misma persona. Esto concluye el proceso de Vinculación de Registros para los ejemplos desarrollados a partir de las Bases de Datos **A** y **B** mostradas en las Tablas 3.1 y 3.2 respectivamente. La Tabla 3.17 muestra la salida final del proceso de Vinculación de Registros para este ejemplo.

rec_id1	rec_id2	Estado de clasificación
a1	b9	No coincidente
a3	b2	Coincidente
a4	b7	No coincidente
a5	b4	Coincidente
a5	b8	No coincidente
a10	b3	No coincidente
a10	b10	No coincidente

Tabla 3.17 Salida final del proceso de Vinculación de Registros tras la fase Revisión Secretarial

3.6.2.- Métricas de evaluación del desempeño

Una vez que todos los vectores de comparación han sido clasificados es posible el cálculo de métricas para evaluar el desempeño del proceso de Vinculación de Registros. Para realizar lo anterior es necesario tener los valores verdaderos de clasificación de los registros, es decir, saber de antemano cuales son duplicados y cuales no lo son, para poder comparar los resultados experimentalmente obtenidos con lo obtenido experimentalmente. Cabe mencionar que en proyectos reales, difícilmente se tiene tal información.

Para el cálculo de las métricas es necesario considerar que cada vector clasificado pues caer en una de las siguientes categorías:

a) Positivos verdaderos (TP): Estos vectores fueron clasificados correctamente como coincidentes.

b) Positivos falsos (FP): Estos vectores fueron erróneamente clasificados como coincidentes.

c) Negativos verdaderos (TN): Estos vectores fueron clasificados correctamente como no coincidentes.

d) Negativos falsos (FN): Estos vectores fueron erróneamente clasificados como no coincidentes.

Mediante un conteo del número de vectores en cada categoría es posible el cálculo de las métricas de evaluación del desempeño.

La primera métrica a considerar es la exactitud (Accuracy en inglés). Esta medida es significativa principalmente cuando el número de elementos en las clases está balanceado, lo cual difícilmente ocurre en la Vinculación de Registros, pues el valor de TN (vectores correctamente clasificados como no coincidentes) tiende a ser mucho mayor que los demás [61]. La exactitud se calcula de la siguiente manera:

$$\mathbf{acc} = \frac{\mathbf{TP}+\mathbf{TN}}{\mathbf{TP}+\mathbf{FP}+\mathbf{TN}+\mathbf{FN}}$$

Otra métrica a considerar es la precisión (Precision en inglés). Esta métrica evalúa cuán preciso fue el proceso de Vinculación de Registros en cuanto a la identificación de los vectores correspondientes a la categoría de positivos verdaderos. La precisión se calcula de la siguiente manera:

$$prec = \frac{TP}{TP + FP}$$

La tercera métrica a considerar es la retirada (Recall en inglés). Esta métrica evalúa cuantas de las parejas de vectores que corresponden efectivamente a registros coincidentes han sido clasificadas como tales. La retirada se calcula de la siguiente manera:

$$rec = \frac{TP}{TP + FN}$$

Capítulo IV

Sistema Febrl

4.1.- Introducción

El software Freely Extensible Biomedical Record Linkage (Febrl) fue desarrollado durante el proyecto de investigación “Research in data matching, record linkage and entity resolution” a cargo del Dr. Peter Christen del Departamento de Ciencias de la Computación de la Universidad Nacional de Australia. Febrl es una herramienta que permite realizar la Vinculación de Registros sobre fuentes de datos consistentes de hasta varios cientos de miles de registros [62]. Además, al ser un software de código abierto, posibilita el aprendizaje y la experimentación sobre el proceso de Vinculación de Registros.

Para el presente trabajo de tesis se utilizó la versión más reciente de Febrl (versión 0.4.2). Cabe mencionar que dicha versión originalmente no opera de manera directa sobre una Base de Datos existente en un Sistema Manejador de Base de Datos, sino que opera sobre archivos de texto planos generados a partir de una Base de Datos, lo cual permite el uso de la herramienta sin temor a perder los datos iniciales, pero también implica que es necesario el volcado de las fuentes de datos en archivos de texto planos.

4.2.- Arquitectura

La arquitectura del software Febrl es modular. Dichos módulos se encuentran escritos en el lenguaje de programación Python y agrupan clases correspondientes a las diferentes fases del proceso de Vinculación de Registros descritas en el Capítulo III. Adicionalmente, existen otros módulos con funcionalidades auxiliares y un módulo dedicado únicamente a la parte lógica de la interfaz gráfica.

Adicionalmente, existen varios archivos XML que se encargan de la creación de la interfaz gráfica. Estos archivos determinan la apariencia de las diferentes ventanas que el usuario visualizará durante su operación de Febrl, mientras que el módulo de la interfaz gráfica escrito en Python se encargará de validar y manejar la entrada dada por el usuario final, crear un script de Python correspondiente al proceso de Vinculación de Registros a ejecutar y darle al usuario la salida

correspondiente. De acuerdo a lo anterior, la tabla 4.1 muestra los módulos correspondientes a Febrl y su descripción.

Módulo	Descripción
auxiliary.py	Diversas funciones auxiliares para los demás módulos
classification.py	Implementa la fase de clasificación del proceso de Vinculación de Registros
comparison.py	Implementa la fase de comparación del proceso de Vinculación de Registros
dataset.py	Módulo encargado del manejo de los conjuntos de datos de entrada para el proceso de Vinculación de Registros
encode.py	Módulo correspondiente a las funciones de codificación fonética para la fase de indexado.
evalClassification.py	Implementa diferentes evaluaciones de los clasificadores existentes en el módulo classification.py
evalIndexing.py	Implementa diferentes evaluaciones de los clasificadores existentes en el módulo classification.py
guiFebrl.py	Encargado de la parte lógica de la interfaz gráfica
indexing.py	Implementa la fase de indexado
lookup.py	Implementa diversas tablas de búsqueda auxiliares para los demás módulos
measurements.py	Implementa el cálculo de métricas de evaluación del desempeño
mymath.py	Rutinas numéricas diversas
output.py	Funciones diversas para la salida del proceso de Vinculación de Registros
phonenum.py	Rutinas para la descomposición y formato de números telefónicos
simplehmm.py	Implementación de un Modelo Oculto de Markov Simple
standarization.py	Implementa las funciones principales para la limpieza y estandarización de los datos de entrada
stringcmp.py	Implementa diferentes funciones de comparación de cadenas
svm.py	Implementa la funcionalidad de Máquina de Vectores de Soporte
svmutil.py	Módulo asistente para creación de un modelo de Máquina de Vectores de Soporte
trainhmm.py	Módulo que implementa el entrenamiento de los Modelos Ocultos de Markov

Tabla 4.1 Módulos correspondientes a la arquitectura de Febrl

4.3.- Funcionamiento

Esta sección describe de manera general el funcionamiento de Febrl para un proyecto de Vinculación de Registros desde la interfaz gráfica de usuario. Cabe mencionar que también es posible la ejecución de proyectos desde la consola de comandos del sistema operativo en cuestión donde se esté ejecutando Febrl. Sin embargo, la operación en modo gráfico es mucho más amigable para el usuario final, por lo que solamente la ejecución de Febrl en tal modalidad es abordada en esta sección.

Para iniciar Febrl es necesario ejecutar el módulo “guiFebrl.py”; esto iniciará la interfaz gráfica de Febrl. Una vez ahí se debe seleccionar si el proceso a realizar es una Vinculación de Registros, una Limpieza y Estandarización de Datos o una Detección de Duplicados y se ingresan los datos de entrada.

La Figura 4.1 muestra un ejemplo del ingreso de datos para un proceso de Detección de Duplicados en Febrl sobre un archivo de nombre “dataset_A_1000.csv”. El archivo anterior es de texto plano en formato CSV; el nombre de tal formato es debido a que el acrónimo en inglés de “Valores Separados Por Comas”, siendo esa la manera en que se organiza la información.

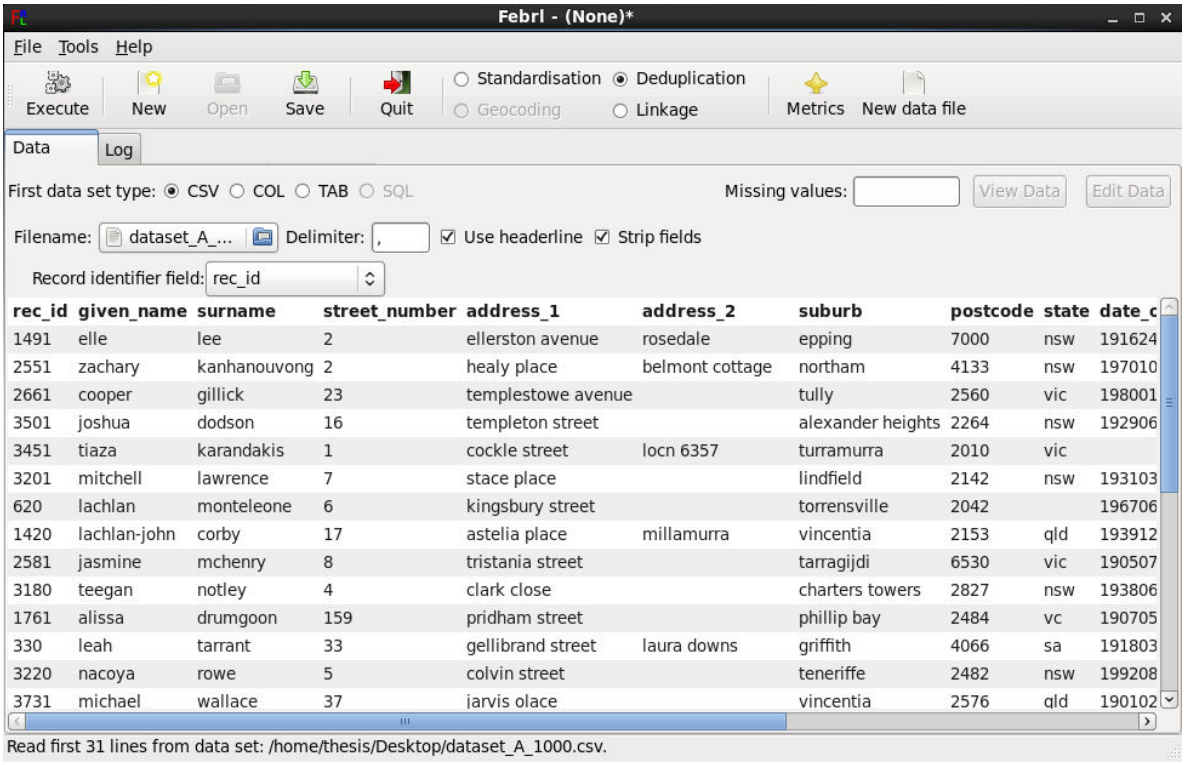


Figura 4.1 Selección del tipo de proyecto a realizar y de los datos de entrada en Febrl

La siguiente actividad a realizar es la selección de atributos y métodos para la fase de indexado del proceso de Vinculación de Registros. La Figura 4.2 muestra la selección del método de indexado “Double Metaphone” a ejecutarse sobre el atributo “surname” para la realización del indexado por bloques.

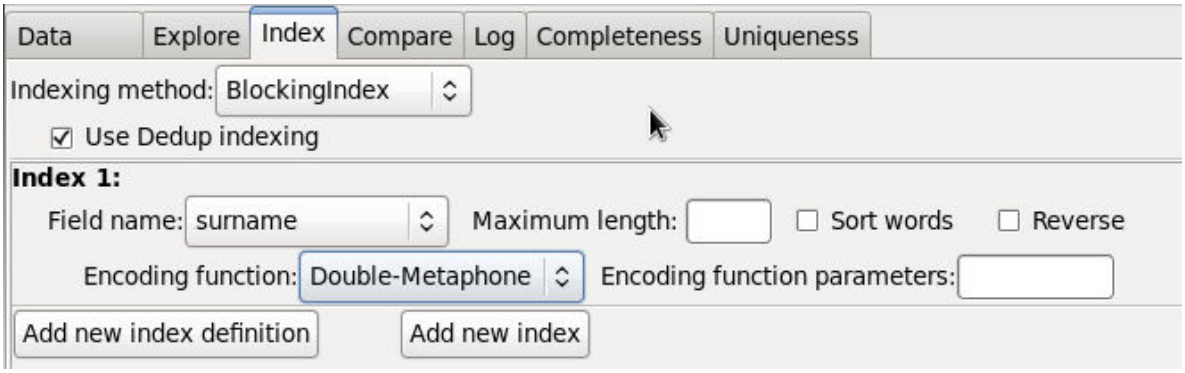


Figura 4.2 Indexado mediante “Double Metaphone” sobre el atributo “surname”

Posteriormente se accede al menú correspondiente a la fase de comparación para seleccionar los atributos que serán comparados y las funciones que realizarán tales comparaciones. La Figura 4.3 muestra la selección de los atributos “given_name” y “suburb” para ser comparados mediante la función “Edit-Dist”, la cual corresponde la métrica de Similitud por Distancia de Levenshtein.

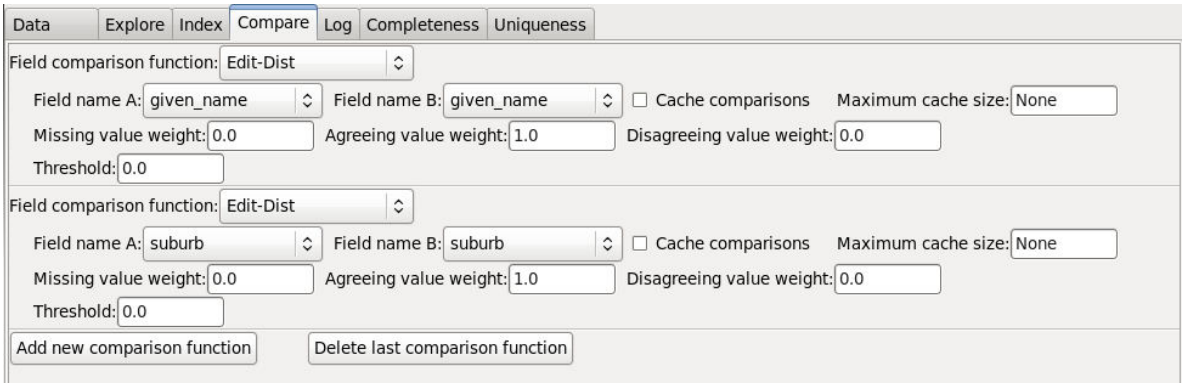


Figura 4.3 Comparación mediante la Similitud por Distancia de Levenshtein

La siguiente fase a configurar es la de comparación; Febrl solicita al usuario que seleccione un método de clasificación y los parámetros que dicho método requiera. La Figura 4.4 muestra la selección del método de clasificación “FellegiSunter” con valores de 1.6 para el umbral inferior y de 1.75 para el umbral superior. El método de clasificación mostrado en Febrl como “FellegiSunter” corresponde al clasificador por umbral de similitud sumada descrito en el Capítulo III.

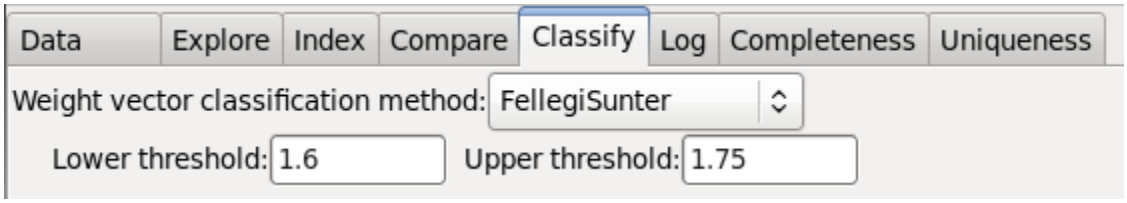


Figura 4.4 Clasificación mediante el clasificador por umbral de similitud sumada

Posteriormente, Febrl permite la selección de la salida del proyecto de Vinculación de Registros en los archivos que el usuario seleccione para este fin. La salida puede ser un archivo que muestre los identificadores de registro junto con su estado de clasificación o el conjunto de datos inicial más una columna que indique su estado final de clasificación. La Figura 4.5 muestra la selección del archivo "dataset_A_1000-Salida-Res.csv" para almacenar la salida del proyecto como el conjunto de datos inicial más la adición de un atributo "match_id" que mostrará el estado de clasificación de los registros.

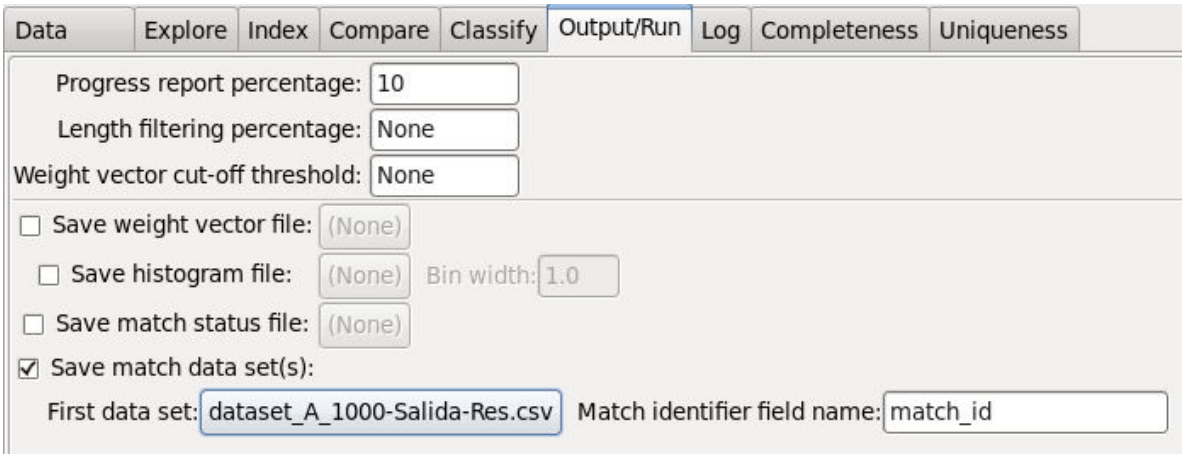
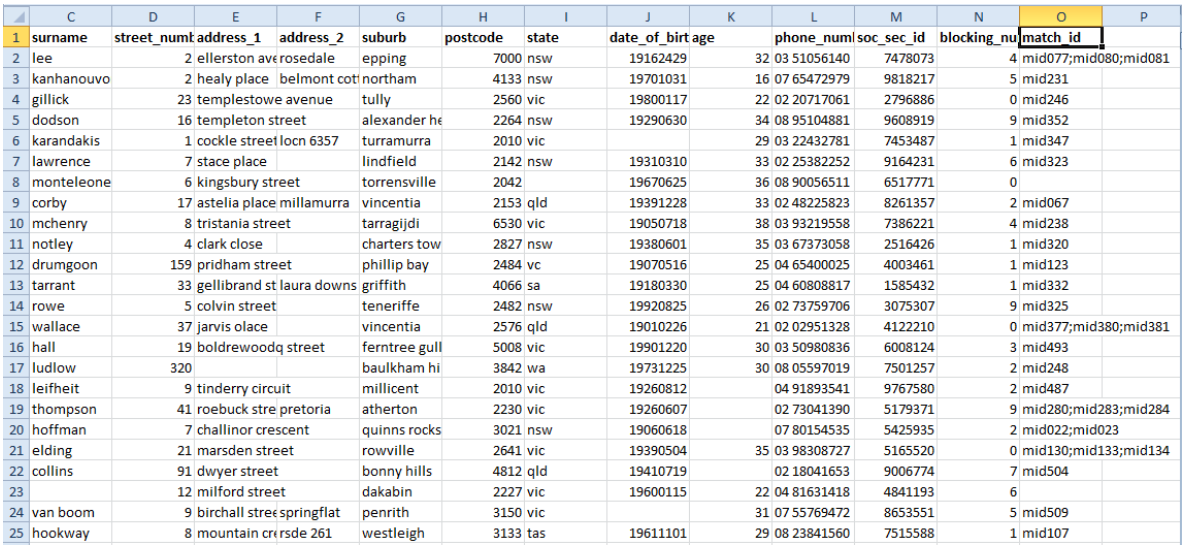


Figura 4.5 Salida del proyecto en forma de conjunto de datos

Una vez que el proyecto se haya ejecutado, la salida se almacena de acuerdo a la configuración provista por el usuario del sistema. La Figura 4.6 muestra el archivo de salida "dataset_A_1000-Salida-Res.csv". Como se mencionó con anterioridad, tal archivo contiene al conjunto de datos original más la adición del atributo "match_id" que indica el estado de clasificación de los registros correspondientes; si no hay texto en dicho campo el registro no posee duplicados, en caso contrario el registro es duplicado de aquellos que compartan sus valores de "match_id".



	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	surname	street_numb	address_1	address_2	suburb	postcode	state	date_of_birt	age	phone_num	soc_sec_id	blocking_nu	match_id	
2	lee	2	ellerston ave	rosedale	epping	7000	nsw	19162429	32	03 51056140	7478073	4	mid077;mid080;mid081	
3	kanhanouvo	2	healy place	belmont cot	northam	4133	nsw	19701031	16	07 65472979	9818217	5	mid231	
4	gillick	23	templestowe	avenue	tully	2560	vic	19800117	22	02 20717061	2796886	0	mid246	
5	dodson	16	templeton	street	alexander he	2264	nsw	19290630	34	08 95104881	9608919	9	mid352	
6	karandakis	1	cockle street	locn 6357	turramurra	2010	vic		29	03 22432781	7453487	1	mid347	
7	lawrence	7	stace place		lindfield	2142	nsw	19310310	33	02 25382252	9164231	6	mid323	
8	monteleone	6	kingsbury	street	torrensville	2042		19670625	36	08 90056511	6517771	0		
9	corby	17	astelia place	millamurra	vincentia	2153	qld	19391228	33	02 48225823	8261357	2	mid067	
10	mchenry	8	tristania	street	tarragijdi	6530	vic	19050718	38	03 93219558	7386221	4	mid238	
11	notley	4	clark close		charters tow	2827	nsw	19380601	35	03 67373058	2516426	1	mid320	
12	drumgoon	159	pridham	street	phillip bay	2484	vc	19070516	25	04 65400025	4003461	1	mid123	
13	tarrant	33	gellibrand st	laura downs	griffith	4066	sa	19180330	25	04 60808817	1585432	1	mid332	
14	rowe	5	colvin	street	teneriffe	2482	nsw	19920825	26	02 73759706	3075307	9	mid325	
15	wallace	37	jarvis olace		vincentia	2576	qld	19010226	21	02 02951328	4122210	0	mid377;mid380;mid381	
16	hall	19	boldrewoodq	street	ferntree gull	5008	vic	19901220	30	03 50980836	6008124	3	mid493	
17	ludlow	320			baulkham hi	3842	wa	19731225	30	08 05597019	7501257	2	mid248	
18	leifheit	9	tinderry	circuit	millicent	2010	vic	19260812		04 91893541	9767580	2	mid487	
19	thompson	41	roebuck stre	pretoria	atherton	2230	vic	19260607		02 73041390	5179371	9	mid280;mid283;mid284	
20	hoffman	7	challinor	crescent	quinns rocks	3021	nsw	19060618		07 80154535	5425935	2	mid022;mid023	
21	elding	21	marsden	street	rowville	2641	vic	19390504	35	03 98308727	5165520	0	mid130;mid133;mid134	
22	collins	91	dwyer	street	bonny hills	4812	qld	19410719		02 18041653	9006774	7	mid504	
23		12	milford	street	dakabin	2227	vic	19600115	22	04 81631418	4841193	6		
24	van boom	9	birchall stree	springflat	penrith	3150	vic		31	07 55769472	8653551	5	mid509	
25	hookway	8	mountain cr	rsde 261	westleigh	3133	tas	19611101	29	08 23841560	7515588	1	mid107	

Figura 4.6 Archivo de salida generado por Febrl para Detección de Duplicados.

Capítulo V

Implementación de métricas de Calidad de Datos en Febrl para MySQL

5.1.- Implementación

En el Capítulo II se abordó el tópico de la Calidad de Datos y se describieron las métricas seleccionadas para el presente trabajo de tesis. Debido a que el objetivo del presente trabajo de tesis es la generación de un Sistema Evaluador de Calidad de Datos en MySQL, es necesaria la implementación de dichas métricas y su integración sucesiva con el Sistema Febrl original para proveer al usuario final la capacidad de evaluación de la Calidad de Datos de las fuentes con las que éste trabaje.

Para la implementación de las métricas de Calidad de Datos se determinaron los siguientes requerimientos:

a) Las métricas de Calidad de Datos a implementar deben operar directamente sobre el Sistema Manejador de Base de Datos MySQL, sin ser necesario el volcado de los datos en un archivo de texto plano. Dado que el cálculo de métricas de Calidad de Datos es una operación que requiere no alterar los datos previamente existentes, el volcado a archivos de texto es innecesario e impráctico para el usuario final. Debido a esto, es necesario que tales métodos se implementen en lenguaje SQL y que se instalen en el Sistema Manejador de Base de Datos dónde se encuentre la fuente de datos cuyas métricas de calidad se deseen calcular.

b) Las métricas anteriormente mencionadas y su integración con Febrl deben de ser implementadas de manera que sea posible operar sobre cualquier Base de Datos de MySQL. Esto implica que el Sistema Evaluador de Calidad de Datos en MySQL debe ser capaz de calcular las métricas de Calidad de Datos a partir únicamente del nombre de la Base de Datos y las credenciales de autenticación para MySQL, sin que se requiera que el operador del sistema conozca o ingrese nombres de las tablas y columnas presentes en la Base de Datos en cuestión.

A fin de lograr el primer requerimiento, los procedimientos correspondientes a las métricas de Calidad de Datos deben instalarse en el Sistema Manejador de Base de Datos cuando el Sistema Evaluador de Calidad de Datos en MySQL inicie sesión para ser llamados conforme sea necesario. Una vez que las métricas de Calidad de Datos hayan sido calculadas para la fuente de datos solicitada por el usuario, los procedimientos almacenados serán desinstalados y la Base de Datos quedará intacta.

Con respecto al segundo requerimiento, éste hace referencia a que en el lenguaje SQL tradicional, es necesario el conocer de antemano las tablas o columnas sobre las que operará una consulta para que ésta se ejecute. Dado que el objetivo es que el Sistema Evaluador de Calidad de Datos en MySQL opera sobre cualquier Base de Datos, resulta impráctico que el operador conozca tales metadatos y los ingrese a las consultas, por lo que el software debe determinar de manera automática tal información y a partir de ella crear las consultas en lenguaje SQL de manera dinámica. La determinación de dichos metadatos será abordada en la siguiente sección.

Los procedimientos que el Sistema Evaluador de Calidad de Datos instala sobre el Sistema Manejador de Base de Datos interactuarán con la interfaz gráfica del Sistema Evaluador Universal de Calidad de Datos; el usuario ingresará en una ventana de la interfaz las credenciales de sesión para MySQL y el nombre de la Base de Datos para la cual se evaluará la Calidad de Datos. El usuario podrá seleccionar si desea calcular la unicidad o completitud de la Base de Datos en cuestión; si se selecciona esta última, será calculada a partir de las métricas de densidad y cobertura, justificando la necesidad de su implementación.

Las funcionalidades mencionadas anteriormente requieren, además de los scripts en SQL, la codificación en lenguaje Python y XML. Se determinó que esta codificación debía realizarse con las variables y los comentarios en lengua inglesa para facilitar la comprensión y modificación del presente software por parte de sus usuarios finales.

Además de la implementación correspondiente a las métricas de Calidad de Datos, también se determinó añadir la funcionalidad de volcado de una Base de Datos a un archivo de texto plano en formato de valores separados por comas (CSV). Si bien esto no es necesario para la parte de evaluación de Calidad de Datos, sí es requerido para el uso del Sistema Evaluador de Calidad de Datos en MySQL para realizar el proceso de Vinculación de Registros.

La Figura 5.1 muestra la ventana del Sistema Evaluador de Calidad de Datos en MySQL que permite al usuario solicitar la evaluación de la Calidad de Datos para una Base de Datos existente en MySQL.

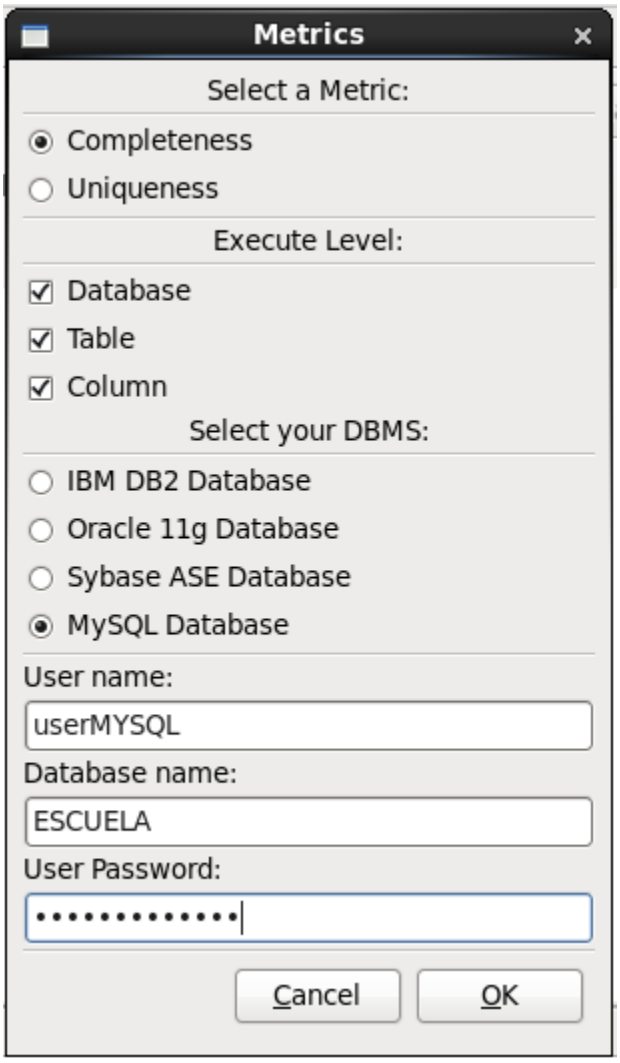


Figura 5.1 Ventana implementada para la entrada de datos del usuario para el cálculo de las métricas de Calidad de Datos de completitud (completeness) y unicidad (uniqueness)

Por otra parte, la Figura 5.2 muestra la ventana de la utilidad que permite al usuario el volcado de una Base de Datos dada en un archivo de texto plano en formato de valores separados por comas para su posterior uso.

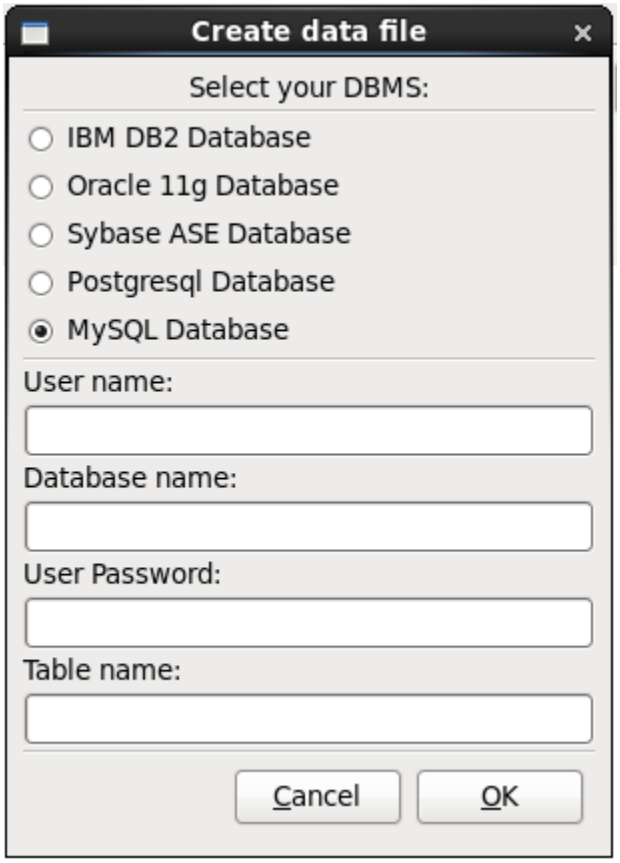


Figura 5.2 Ventana para el ingreso de datos para la utilidad de volcado de una Base de Datos a un archivo de texto

Es importante notar que en ambas figuras aparece la opción de seleccionar el Sistema Manejador de Base de Datos; esto se debe a la posterior integración a realizar del Sistema Evaluador de Calidad de Datos en MySQL dentro de un Sistema Evaluador Universal de Calidad de Datos.

Las siguientes secciones del presente capítulo describen los detalles más importantes de la implementación de las métricas de Calidad de Datos para el Sistema Evaluador de Calidad de Datos en MySQL.

5.1.1.- Acceso a los metadatos del Diccionario de Datos

En la sección anterior se mencionó que el Sistema Evaluador de Calidad de Datos en MySQL debe ser capaz de calcular las métricas de Calidad de Datos de cualquier Base de Datos en MySQL. Para lograr dicho propósito es necesario conocer, en tiempo de ejecución, las tablas y columnas que pertenecen a una

Base de Datos para así poder crear de manera dinámica los procedimientos correspondientes a las métricas de Calidad de Datos.

MySQL proporciona su Diccionario de Datos con el nombre de "INFORMATION_SCHEMA". Para la versión 5.6 del MySQL, el Diccionario de Datos se compone de 29 tablas principales de solo lectura [63]; sin embargo, para el presente trabajo de tesis solo es necesaria la consulta de la tabla "INFORMATION_SCHEMA.COLUMNS". Esta tabla posee información suficiente para que el Sistema Evaluador de Calidad de Datos en MySQL pueda crear sus consultas de manera dinámica y así calcular las métricas de Calidad de Datos. La Figuras 5.3 muestra la descripción que MySQL brinda para dicha tabla.

File Edit View Search Terminal Help

```
mysql> describe INFORMATION_SCHEMA.COLUMNS;
+--------------------------+---------------------+------+-----+---------+-------+
| Field                    | Type                | Null | Key | Default | Extra |
+--------------------------+---------------------+------+-----+---------+-------+
| TABLE_CATALOG            | varchar(512)        | NO   |     |         |       |
| TABLE_SCHEMA             | varchar(64)         | NO   |     |         |       |
| TABLE_NAME               | varchar(64)         | NO   |     |         |       |
| COLUMN_NAME              | varchar(64)         | NO   |     |         |       |
| ORDINAL_POSITION         | bigint(21) unsigned | NO   |     | 0       |       |
| COLUMN_DEFAULT           | longtext            | YES  |     | NULL    |       |
| IS_NULLABLE              | varchar(3)          | NO   |     |         |       |
| DATA_TYPE                | varchar(64)         | NO   |     |         |       |
| CHARACTER_MAXIMUM_LENGTH | bigint(21) unsigned | YES  |     | NULL    |       |
| CHARACTER_OCTET_LENGTH   | bigint(21) unsigned | YES  |     | NULL    |       |
| NUMERIC_PRECISION        | bigint(21) unsigned | YES  |     | NULL    |       |
| NUMERIC_SCALE            | bigint(21) unsigned | YES  |     | NULL    |       |
| DATETIME_PRECISION       | bigint(21) unsigned | YES  |     | NULL    |       |
| CHARACTER_SET_NAME       | varchar(32)         | YES  |     | NULL    |       |
| COLLATION_NAME           | varchar(32)         | YES  |     | NULL    |       |
| COLUMN_TYPE              | longtext            | NO   |     | NULL    |       |
| COLUMN_KEY               | varchar(3)          | NO   |     |         |       |
| EXTRA                    | varchar(30)         | NO   |     |         |       |
| PRIVILEGES               | varchar(80)         | NO   |     |         |       |
| COLUMN_COMMENT           | varchar(1024)       | NO   |     |         |       |
+--------------------------+---------------------+------+-----+---------+-------+
20 rows in set (0.00 sec)
```

Figura 5.3 Descripción de la tabla INFORMATION_SCHEMA.COLUMNS de acuerdo a MySQL

De la tabla anterior se seleccionarán los atributos "TABLE_NAME" y "COLUMN_NAME". Esta información se guardará en la tabla temporal "myDataDict" que permitirá la creación de los scripts de las métricas de Calidad de Datos y evitará más consultas al Diccionario de Datos, concluyendo así el acceso a los metadatos de MySQL. La Figura 5.4 muestra el código fuente en lenguaje

SQL que corresponde a la extracción de la información necesaria del Diccionario de Datos.

```
CREATE PROCEDURE initDQM(dbName varchar(64))
begin
   DROP TEMPORARY TABLE IF EXISTS myDataDict;
   create temporary table myDataDict
   as select TABLE_NAME,COLUMN_NAME from INFORMATION_SCHEMA.COLUMNS
   where TABLE_SCHEMA = dbName;

   DROP TEMPORARY TABLE IF EXISTS resultsColLevelDQM;
   create temporary table resultsColLevelDQM(tableName
varchar(64),colName varchar(64),colDensity double);

   DROP TEMPORARY TABLE IF EXISTS resultsTableLevelDQM;
   create temporary table resultsTableLevelDQM(tableName
varchar(64),tCoverage double,tDensity double,tCompleteness
double,tUniqueness double);

   DROP TEMPORARY TABLE IF EXISTS resultsDBLevelDQM;
   create temporary table resultsDBLevelDQM(dbName varchar(64),dbCoverage
double,dbDensity double,dbCompleteness double,dbUniqueness double);

end
```

Figura 5.4 Implementación en lenguaje SQL de la extracción de metadatos del Diccionario de Datos

El procedimiento mostrado en la Figura 5.4, además de encargarse de la extracción de metadatos, también crea tres tablas temporales para almacenar los resultados del cálculo de las métricas de Calidad de Datos para imprimir tales valores una vez que el procedimiento haya concluido. Dichas tablas son “resultsColLevelDQM”, “resultsTableLevelDQM”, y “resultsDBLevelDQM” y éstas almacenarán los valores de evaluación de Calidad de Datos para granularidad de nivel columna, nivel tabla y nivel de Base de Datos respectivamente.

5.1.2.- Densidad

El cálculo de la métrica de densidad de acuerdo a su descripción en el Capítulo II puede realizarse a nivel de columna, de tabla y de Base de Datos. Los valores de la métrica se obtienen a nivel de columna y a partir de ellos se calculan a nivel de tabla y de Base de Datos si el usuario así lo desea. Los resultados son guardados en las tablas temporales mencionadas en la sección anterior para su posterior impresión en pantalla en la interfaz gráfica.

La Figura 5.5 muestra el procedimiento `columnDensity` que calcula la densidad a nivel columna.

```
CREATE PROCEDURE columnDensity(dbName varchar(64),tableName
varchar(64),colName varchar(64))
begin
DECLARE nonNullsCol int default 0;
DECLARE fieldsPerCol int default 0;
   call numbNonNullInCol(dbName,tableName,colName);
   select @res into nonNullsCol;

   SET @dqpst1 = concat('select count(*) into @res from
',dbName,'.',tableName);
   PREPARE dqps1 from @dqpst1;
   EXECUTE dqps1;
   DEALLOCATE PREPARE dqps1;
   select @res into fieldsPerCol;
   set @res := nonNullsCol / fieldsPerCol;
end
```

Figura 5.5 Procedimiento que calcula de la densidad a nivel columna

El procedimiento `columnDensity` llama a la función auxiliar `numbNonNullInCol` que calcula el número de valores no nulos en una columna dada. El código SQL correspondiente a dicha función se muestra en la figura 5.6.

```
CREATE PROCEDURE numbNonNullInCol(dbName varchar(64),tableName
varchar(64),colName varchar(64))
begin
SET @dqpst1 = concat('select count(',colName,') into @res from
',dbName,'.',tableName);
PREPARE dqps1 from @dqpst1;
EXECUTE dqps1;
DEALLOCATE PREPARE dqps1;
end
```

Figura 5.6 Procedimiento que calcula el número de campos no nulos en una columna dada

Por otro lado, en la Figura 5.7 se muestra el código SQL del procedimiento `tableDensity` que realiza el cálculo de la densidad a nivel de tabla. Cabe mencionar que la función que opera a nivel tabla es llamada incluso si el usuario solo desea conocer la densidad a nivel columnar. Esto se debe a que el procedimiento a nivel de columna requiere como parámetros la base de datos, tabla y columna sobre la cual va a operar; tales parámetros le son provistos por la función que opera a nivel de tabla.

```
CREATE PROCEDURE tableDensity(dbName varchar(64),tabName varchar(64))
begin

DECLARE DONE INT DEFAULT FALSE;
DECLARE columnName varchar(64);
DECLARE numColsInTable int default 0;
DECLARE sumIndDensity double default 0;
DECLARE curTempColNam CURSOR for select COLUMN_NAME from myDataDictCols;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS myDataDictCols;

create temporary table myDataDictCols as select COLUMN_NAME from
myDataDict where TABLE_NAME = tabName;

OPEN curTempColNam;
read_loop:LOOP
   FETCH curTempColNam into columnName;
   if done then
      LEAVE read_loop;
   end if;

   call columnDensity(dbName,tabName,columnName);
   set sumIndDensity := sumIndDensity + @res;
   insert into resultsColLevelDQM values(tabName,columnName,@res);
   set numColsInTable := numColsInTable + 1;
end LOOP;

set @res := sumIndDensity / numColsInTable;
update resultsTableLevelDQM
set tDensity = @res
where tableName = tabName ;

end
```

Figura 5.7 Procedimiento de cálculo de densidad a nivel tabla.

Finalmente, la Figura 5.8 muestra el código en lenguaje SQL correspondiente al cálculo de la densidad a nivel de Base de Datos. De acuerdo a la definición de dicha métrica abordada en el Capítulo II, la densidad a nivel de Base de Datos se obtiene al promediar los valores de densidad de todas las tablas de la Base de Datos.

```
CREATE PROCEDURE schemaDensity(schemaName varchar(64))
begin
DECLARE DONE INT DEFAULT FALSE;
DECLARE numTablesInSchema int default 0;
DECLARE sumIndDensity double default 0;
DECLARE tableName varchar(64);
DECLARE curTempTabNam CURSOR for select TABLE_NAME from myDataDictTabs;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS myDataDictTabs;
create temporary table myDataDictTabs as select distinct(TABLE_NAME) from
myDataDict;

OPEN curTempTabNam;
read_loop:LOOP
   FETCH curTempTabNam into tableName;
   if done then
      LEAVE read_loop;
   end if;

   call tableDensity(schemaName,tableName);
   set sumIndDensity := sumIndDensity + @res;
   set numTablesInSchema := numTablesInSchema + 1;
end LOOP;
set @res := sumIndDensity / numTablesInSchema;
update resultsDBLevelDQM
set dbDensity = @res;
end
```

Figura 5.8 Procedimiento del cálculo de densidad a nivel Base de Datos

5.1.3.- Cobertura

En el Capítulo II se definió la métrica de cobertura. Debido a que la definición ahí mostrada puede usarse como una medida del grado en el que una tabla representa al total de las entidades de una Base de Datos, la cobertura no puede ser calculada a nivel de columna, siendo posible únicamente su medición a nivel de tabla y a nivel de Base de Datos.

De esa manera, el valor de la métrica de cobertura para una tabla dada se obtiene al dividir el número de entidades presentes en la tabla sobre el número total de entidades diferentes en la Base de Datos; si hay dos atributos con el mismo nombre pero en tablas diferentes, se trabajará bajo el supuesto de que tales atributos se refieren a la misma entidad. Por otra parte, el valor de la cobertura de una Base de Datos se obtiene al promediar la cobertura de sus tablas.

La Figura 5.9 muestra el código SQL corresponde al cálculo de la cobertura a nivel de tabla.

```
CREATE PROCEDURE tableCoverage(dbName varchar(64),tabName varchar(64))
begin
DECLARE numEntDb,numEntTable int default 0;
select count(distinct COLUMN_NAME) into @res from
INFORMATION_SCHEMA.COLUMNS where TABLE_SCHEMA = dbName;
select @res into numEntDb;
select count(distinct COLUMN_NAME) into @res from
INFORMATION_SCHEMA.COLUMNS where TABLE_SCHEMA = dbName and TABLE_NAME =
tabName;
select @res into numEntTable;
select (numEntTable / numEntDb) into @res;
insert into resultsTableLevelDQM(tableName,tCoverage)
values(tabName,@res);
end
```

Figura 5.9 Procedimiento correspondiente al cálculo de la cobertura a nivel de tabla

Finalmente, en la Figura 5.10 se muestra el código SQL del procedimiento `schemaCoverage`, el cual calcula la cobertura a nivel de Base de Datos.

```
CREATE PROCEDURE schemaCoverage(schemaName varchar(64))
begin
DECLARE DONE INT DEFAULT FALSE;
DECLARE numTablesInSchema int default 0;
DECLARE sumIndCoverage double default 0;
DECLARE tableName varchar(64);
DECLARE curTempTabNam CURSOR for select TABLE_NAME from myDataDictTabs;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS myDataDict;
CREATE TEMPORARY TABLE myDataDict
as select TABLE_NAME,COLUMN_NAME from INFORMATION_SCHEMA.COLUMNS where
TABLE_SCHEMA = schemaName;

DROP TEMPORARY TABLE IF EXISTS myDataDictTabs;
create temporary table myDataDictTabs as select distinct(TABLE_NAME) from
myDataDict;
OPEN curTempTabNam;
read_loop:LOOP
   FETCH curTempTabNam into tableName;
   if done then
      LEAVE read_loop;
   end if;
   call tableCoverage(schemaName,tableName);
   set sumIndCoverage := sumIndCoverage + @res;
   set numTablesInSchema := numTablesInSchema + 1;
end LOOP;
set @res := sumIndCoverage / numTablesInSchema;
insert into resultsDBLevelDQM(dbCoverage) values(@res);
end
```

Figura 5.10 Procedimiento correspondiente al cálculo de la cobertura a nivel de Base de Datos

5.1.4.- Completitud

La métrica de completitud, de acuerdo a su definición mostrada en el Capítulo II, requiere del cálculo previo de las métricas de densidad y cobertura para poder ser determinada. Debido a ello, la completitud no puede ser calculada a nivel de columna, sino solamente a nivel de tabla y de Base de Datos.

La Figura 5.10 muestra el código SQL correspondiente al cálculo de la completitud a nivel de tabla. Note la dependencia de los valores de densidad y cobertura de la misma granularidad.

```
CREATE PROCEDURE tableCompleteness(dbName varchar(64),tabName
varchar(64))
begin

DECLARE tableDensity,tableCoverage 60double default 0;
select tDensity into tableDensity
from resultsTableLevelDQM
where tableName = tabName ;

select tCoverage into tableCoverage
from resultsTableLevelDQM
where tableName = tabName ;

set @res := tableDensity * tableCoverage;

update resultsTableLevelDQM

set tCompleteness = @res
where tableName = tabName;

end
```

Figura 5.10 Procedimiento correspondiente al cálculo de la completitud a nivel de tabla

La Figura 5.11 muestra el código SQL correspondiente al cálculo de la métrica de completitud a nivel de Base de Datos. Observe que el cálculo es dependiente de los valores de las métricas de cobertura y de densidad a nivel de Base de Datos.

```
CREATE PROCEDURE schemaCompleteness(schemaName varchar(64))
begin
DECLARE DONE INT DEFAULT FALSE;
DECLARE schemaDensity,schemaCoverage double default 0;
DECLARE tableName varchar(64);
DECLARE curTempTabNam CURSOR for select TABLE_NAME from myDataDict;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

call schemaCoverage(schemaName);
call schemaDensity(schemaName);

OPEN curTempTabNam;
read_loop:LOOP
   FETCH curTempTabNam into tableName;
   if done then
      LEAVE read_loop;
   end if;

   call tableCompleteness(schemaName,tableName);
end LOOP;

select dbDensity into schemaDensity
from resultsDBLevelDQM;

select dbCoverage into schemaCoverage
from resultsDBLevelDQM;

set @res := schemaDensity * schemaCoverage;

update resultsDBLevelDQM
set dbCompleteness := @res;
end
```

Figura 5.11 Procedimiento correspondiente al cálculo de la métrica de completitud a nivel de Base de Datos

5.1.5.- Unicidad

La unicidad, de acuerdo a la definición presente en el Capítulo II, es una medida de qué tan únicos son los datos presentes en una fuente de datos. Debido a ello, esta métrica solamente puede calcularse a nivel de tabla y a nivel de Base de Datos.

La Figura 5.12 muestra la implementación correspondiente para el nivel de tabla. Cabe mencionar que este código tiene un alto costo computacional debido a la complejidad computacional asociada a comparar entre sí a los registros de una tabla para determinar si son únicos o no. Esto implica que el tiempo de ejecución de este procedimiento es elevado para tablas con muchos registros.

```
CREATE PROCEDURE tableUniqueness(schemaName varchar(64),tableName varchar(64))
begin
DECLARE DONE INT DEFAULT FALSE;
DECLARE comma INT;
DECLARE numTuples INT;
DECLARE uniqueTuples INT;
DECLARE columnName varchar(64);
DECLARE curTempColNam CURSOR for select COLUMN_NAME from myDataDictCols;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS myDataDict;
CREATE TEMPORARY TABLE myDataDict
as   select   TABLE_NAME,COLUMN_NAME   from   INFORMATION_SCHEMA.COLUMNS   where
TABLE_SCHEMA = schemaName;

DROP TEMPORARY TABLE IF EXISTS myDataDictTabs;
create  temporary  table  myDataDictTabs  as  select  distinct(TABLE_NAME)  from
myDataDict;

DROP TEMPORARY TABLE IF EXISTS myDataDictCols;
CREATE TEMPORARY TABLE myDataDictCols
as select COLUMN_NAME from myDataDict where TABLE_NAME = tableName;

SET @dqpst1 = concat('select count(*) into @res from ',schemaName,'.',tableName);
PREPARE dqps1 from @dqpst1;
EXECUTE dqps1;
DEALLOCATE PREPARE dqps1;
select @res into numTuples;
OPEN curTempColNam;
SET comma := 0;
SET @colNamesOMC = '';
read_loop:LOOP
   FETCH curTempColNam into columnName;
   if done then
      LEAVE read_loop;
   end if;

   if comma then
      SET @colNamesOMC = concat(@colNamesOMC,',');
   end if;

   SET @colNamesOMC = concat(@colNamesOMC,columnName);
   SET comma := 1;
end LOOP;

SET @dqpst1 = 'select count(*) into @res from (select ';
SET @dqpst1 = concat(@dqpst1,@colNamesOMC);
SET @dqpst1 = concat(@dqpst1,' from ',schemaName,'.',tableName,' group by ');
SET @dqpst1 = concat(@dqpst1,@colNamesOMC);
SET @dqpst1 = concat(@dqpst1,' having count(*) = 1)tableAlias1OMC;');
PREPARE dqps1 from @dqpst1;
EXECUTE dqps1;
DEALLOCATE PREPARE dqps1;

select @res into uniqueTuples;

set @res = (1-((numTuples-uniqueTuples) / numTuples));
end
```

Figura 5.12 Procedimiento correspondiente al cálculo de unicidad a nivel de tabla

Finalmente, la Figura 5.13 corresponde al cálculo de la unicidad a nivel de Base de Datos. Dicho cálculo se realiza al promediar los valores de unicidad para todas las tablas existentes en la Base de Datos en cuestión.

```
CREATE PROCEDURE schemaUniqueness(schemaName varchar(64))
begin
DECLARE DONE INT DEFAULT FALSE;
DECLARE numTablesInSchema int default 0;
DECLARE sumIndUniqueness double default 0;
DECLARE tableName varchar(64);
DECLARE curTempTabNam CURSOR for select TABLE_NAME from myDataDictTabs;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

DROP TEMPORARY TABLE IF EXISTS myDataDict;
CREATE TEMPORARY TABLE myDataDict
as select TABLE_NAME,COLUMN_NAME from INFORMATION_SCHEMA.COLUMNS where
TABLE_SCHEMA = schemaName;

DROP TEMPORARY TABLE IF EXISTS myDataDictTabs;
create temporary table myDataDictTabs as select distinct(TABLE_NAME) from
myDataDict;

OPEN curTempTabNam;
read_loop:LOOP
   FETCH curTempTabNam into tableName;
   if done then
      LEAVE read_loop;
   end if;

   call tableUniqueness(schemaName,tableName);
   insert into resultsTableLevelDQM(tableName,tUniqueness)
values(tableName,@res);
   set sumIndUniqueness := sumIndUniqueness + @res;
   set numTablesInSchema := numTablesInSchema + 1;
end LOOP;
set @res := sumIndUniqueness / numTablesInSchema;
insert into resultsDBLevelDQM(dbUniqueness) values(@res);
end
```

Figura 5.13 Procedimiento correspondiente al cálculo de la unicidad a nivel de Base de Datos

5.1.6.- Pruebas

Para realizar las pruebas de las implementaciones anteriormente descritas se diseñó e instaló en MySQL una Base de Datos de prueba de nombre “ESCUELA”. Dicha Base de Datos está integrada por las tablas “Alumnos” y “Profesores”. El contenido de dichas tablas se muestra en las Figuras 5.14 y 5.15 respectivamente.

```
File  Edit  View  Search  Terminal  Help
mysql> select * from ESCUELA.Alumnos;
+--------+---------------+----------+
| numAlu | nombre        | promedio |
+--------+---------------+----------+
|   NULL | NULL          |     NULL |
|      1 | Carlos Ortiz  |     8.75 |
|   NULL | Alu Generico  |        8 |
|      2 | NULL          |     9.99 |
|      3 | Nuevo Recluta |     NULL |
|   NULL | NULL          |     NULL |
|     10 | NULL          |       10 |
+--------+---------------+----------+
7 rows in set (0.00 sec)
```

Figura 5.14 Contenido de la tabla “Alumnos” de la Base de Datos “ESCUELA” mostrado por MySQL

```
File  Edit  View  Search  Terminal  Help
mysql> select * from ESCUELA.Profesores;
+---------+---------------------+------+
| numProf | nombre              | edad |
+---------+---------------------+------+
|       1 | Profesora Aleatoria | NULL |
|       2 | NULL                |   50 |
|    NULL | Profesor Aleatorio  |   50 |
|    NULL | NULL                | NULL |
|       3 | Dr. Steven Skiena   |   50 |
+---------+---------------------+------+
5 rows in set (0.00 sec)
```

Figura 5.15 Contenido de la tabla “Profesores” de la Base de Datos “ESCUELA” mostrado por MySQL

Con la existencia de la Base de Datos anteriormente descrita se pueden probar las implementaciones de las métricas de Calidad de Datos abordadas en el presente

trabajo de tesis. La Figura 5.16 muestra el ingreso de los datos necesarios para que el Sistema Evaluador de Calidad de Datos en MySQL calcule la métrica de completitud.

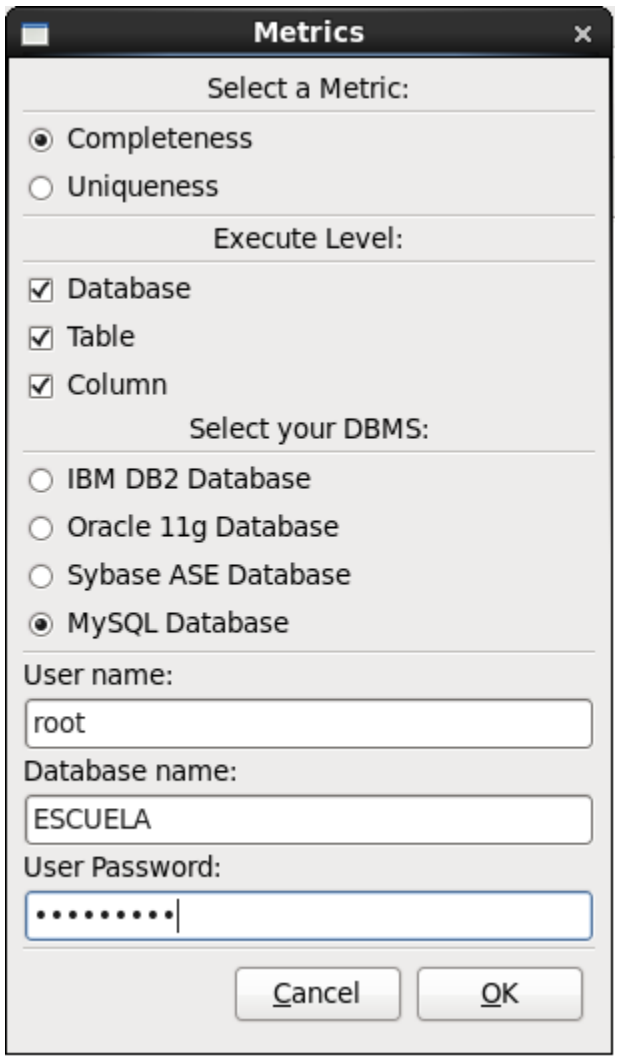


Figura 5.16 Interacción con la interfaz gráfica para el cálculo de la completitud

La Figura 5.17 muestra la salida de dicha petición. Debido que la completitud depende de la densidad y la cobertura, los valores de estas últimas también se muestran.

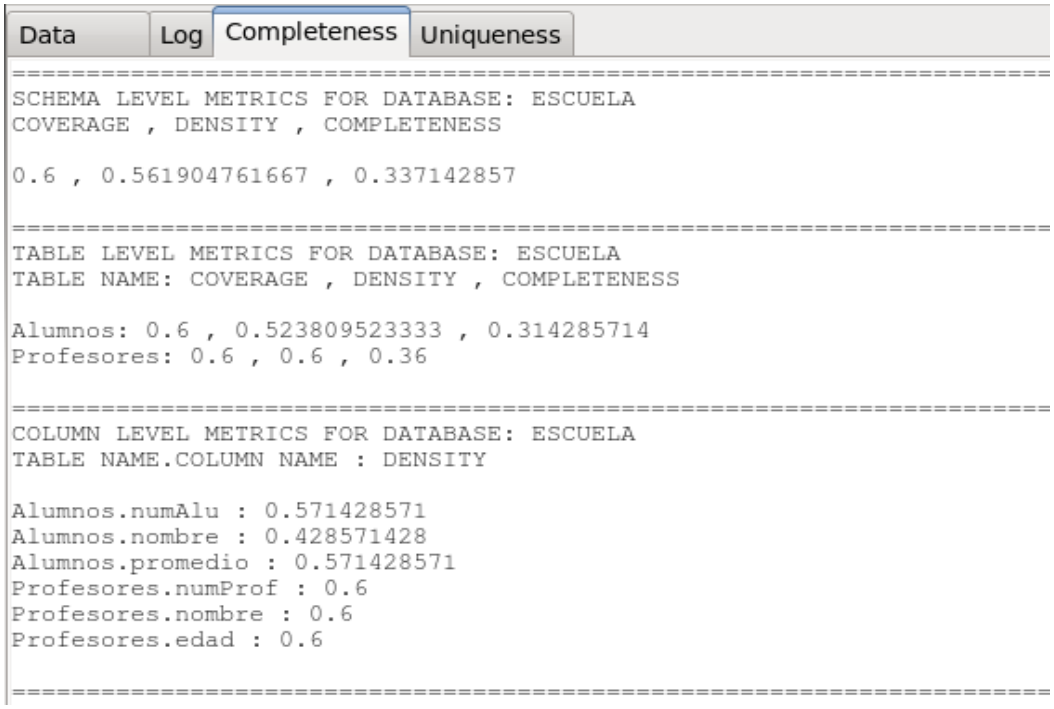


Figura 5.17 Valores de las métricas de completitud, densidad y cobertura para diferentes granularidades

Con la Base de Datos de prueba también es posible probar la implementación de la métrica de unicidad. La Figura 5.18 corresponde a la salida de la ejecución de dicha métrica para nivel de tabla y de Base de Datos.

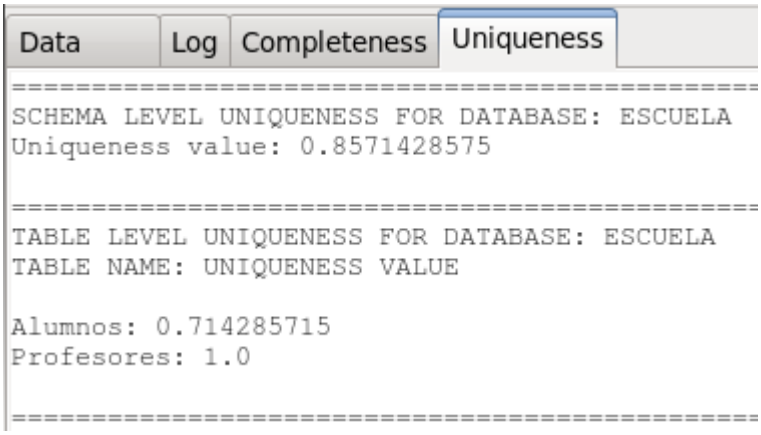


Figura 5.18 Valor de la métrica de unicidad a nivel de tabla y de Base de Datos

5.1.7.- Análisis de resultados

Para analizar los resultados obtenidos por el Sistema Evaluador de Calidad de Datos en MySQL es necesario conocer los valores reales de las métricas calculadas. Tales valores se pueden hallar fácilmente aplicando la fórmula correspondiente a la métrica.

Para la métrica de unicidad se observa que en la tabla “Alumnos” dos de los siete registros son duplicados, por lo que la métrica de unicidad para esa tabla tendrá un valor de 1 – (2/7) lo cual corresponde decimalmente a aproximadamente 0.714285.

Por otro lado, en la tabla “Profesores” se observa que no existen registros duplicados, por lo que la métrica de unicidad para dicha tabla tendrá un valor de 1 – (0/5), lo cual es igual a 1. Estos valores corresponden a aquellos calculados por el Sistema Evaluador de Calidad de Datos en MySQL, con lo que se demuestra que la implementación de la métrica de unicidad es correcta.

Siguiendo el ejemplo anterior es posible calcular los valores de las demás métricas. La tabla 5.1 muestra el resultado de evaluar la métrica, su nivel de ejecución, el resultado de evaluar su fórmula y la aproximación decimal de dicha evaluación. Para todos los casos, el sistema calculó correctamente la métrica en cuestión, si bien existen diferencia mínimas entre los valores por causa de redondeo y truncamiento de dígitos.

Métrica	Granularidad	Entidad evaluada	Formula evaluada	Valor decimal aproximado
Densidad	Columna	Alumnos.numAlu	4/7	0.57142857142857142
Densidad	Columna	Alumnos.nombre	3/7	0.42857142857142857
Densidad	Columna	Alumnos.promedio	4/7	0.57142857142857142
Densidad	Columna	Profesores.numProf	3/5	0.6
Densidad	Columna	Profesores.nombre	3/5	0.6
Densidad	Columna	Profesores.edad	3/5	0.6
Cobertura	Tabla	Alumnos	3/5	0.6
Cobertura	Tabla	Profesores	3/5	0.6
Densidad	Tabla	Alumnos	((4/7) + (3/7) + (4/7)) / 3	0.52380952380952380
Densidad	Tabla	Profesores	((3/5) + (3/7) + (3/5)) / 3	0.6
Completitud	Tabla	Alumnos	(0.6) (0.52380)	0.31428571428571428
Completitud	Tabla	Profesores	(0.6) (0.6)	0.36
Cobertura	Base de Datos	ESCUELA	((0.6) + (0.6)) / 2	0.6
Densidad	Base de Datos	ESCUELA	((0.6) + (0.52380)) / 2	0.5619047619047619
Completitud	Base de Datos	ESCUELA	(0.6) (0.56190)	0.337142857142857

Tabla 5.1 Valores de las métricas para corroborar el correcto funcionamiento de los procedimientos correspondientes a las métricas de Calidad de Datos

Es necesario mencionar que si bien estas pruebas fueron realizadas sobre una Base de Datos muy pequeña, esto fue para ilustrar fácilmente que la implementación de las métricas de Calidad de Datos abordadas en el presente trabajo de tesis es correcta. Sin embargo, la funcionalidad del cálculo de métricas de Calidad de Datos ha sido probada exitosamente sobre Bases de Datos como la TPC-H [64], cuyo tamaño está en el orden de los Gigabytes.

Capítulo VI

Mejora de métodos de Vinculación de Registros

6.1.- Mejora a métodos de clasificación ya existentes en Febrl

Como se mencionó a lo largo del Capítulo III, en la fase de clasificación se determina, para cada registro de los datos de entrada, si es un registro no duplicado, duplicado o posiblemente duplicado. Debido a ello, para el presente trabajo de tesis se escogió la fase de clasificación para implementar mejoras a los métodos existentes en Febrl.

Febrl tiene tres métodos de clasificación no supervisada: K medias, Farthest First y un clasificador por umbral de similitud sumada (el cual aparece en Febrl bajo el nombre de "FellegiSunter"); tales algoritmos se abordan en el Capítulo III. Tras analizar las ventajas y desventajas de cada método, se determinó mejorar el método de clasificación por clasificador por umbral de similitud sumada. Esta decisión se tomó debido a la simplicidad de dicho método, la cual facilita el proceso de Vinculación de Registros para el usuario final.

Cabe mencionar que se decidió trabajar únicamente sobre la clasificación no supervisada debido a que ésta no requiere un conjunto de datos de entrenamiento; dado que tales datos de entrenamiento son usualmente inexistentes en proyectos reales de Vinculación de Registros, se optó por no abordar la clasificación supervisada.

6.1.1.- Ventajas y desventajas del método a mejorar

La principal ventaja del clasificador por umbral de similitud sumada es su simplicidad [50]. Esto se debe a que el usuario únicamente debe de ingresar un umbral (si desea clasificar los registros en duplicados y no duplicados) o dos umbrales (para clasificar en duplicados, no duplicados y posiblemente duplicados), sin embargo, tal simplicidad implica, de acuerdo a Christen [50], las siguientes desventajas:

a) Si la fase de comparación tuvo como salida vectores con valores normalizados entre cero y uno, entonces todos los atributos comparados

aportan de igual manera al valor final de similitud sumada, ignorando así el poder discriminativo que cada atributo tiene. Como ejemplo de esto considere que el nombre propio y el apellido de una persona son más útiles para identificarla que su dirección, debido a la dificultad de que dicha persona cambie de nombre o apellido.

b) Aún si se usan valores ponderados en la fase de comparación, al sumar todos los valores de similitud se pierde la información detallada correspondiente a cada uno de los valores de similitud de los atributos comparados.

Para ejemplificar la segunda desventaja considere la Tabla 6.1, la cual fue propuesta por Christen para ilustrar las desventajas de la clasificación por umbral de similitud sumada [65]. En ella se muestran dos parejas de registros y sus respectivos vectores de comparación, los cuales son el resultado de comparar los atributos mediante la Similitud por Distancia de Levenshtein y aparecen sombreados debajo de su pareja correspondiente. Note que existe un atributo llamado “SimSum”, el cual no es propio de los registros sino que indica la suma de los valores de similitud de los vectores de comparación de registros.

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	SimSum
a1	john	smith	18	miller st	dickson	12	11	1970	--
b1	jonny	smyth	73	miller st	dixon	11	12	1970	--
--	0.6	0.8	0.0	1.0	0.6	0.5	0.5	1.0	5.0
a2	mary	harris	42	swamp rd	sydney	21	4	1918	--
b2	mandy	garrett	42	smither pl	sydneyham	27	4	1979	--
--	0.6	0.4	1.0	0.4	0.6	0.5	1.0	0.5	5.0

Tabla 6.1 Parejas de vectores comparados (a1,b1) y (a2,b2)

De acuerdo a lo mostrado en la Tabla 6.1, se observa que los registros de la pareja (a1,b1) son altamente similares entre sí, por lo que existe una alta probabilidad de que se refieran a la misma persona. Por otro lado, los registros de la pareja (a2,b2) son altamente disimilares entre sí. Sin embargo, ambas parejas de registros tienen un valor de similitud sumada de 5.0. Por lo tanto, si se tuviese un umbral de 5.0, ambas parejas de registros serían clasificadas como coincidentes, a pesar de que con casi total certeza los registros (a2,b2) corresponden a personas diferentes. De la misma manera, si se utiliza un umbral de 4.9, ambas parejas serían clasificadas como no coincidentes, a pesar de que los registros (a1,b1) son, con una alta probabilidad, coincidentes.

6.1.2.- Propuesta de mejora sobre los métodos ya existentes en FebrI

Se propone implementar un método de clasificación que elimine las desventajas de un clasificador por umbral de similitud sumada. Así, tal implementación permitirá tomar en cuenta (aun cuando la salida de la fase de comparación sean vectores normalizados con valores entre cero y uno) la importancia de cada atributo para la discriminación de si los registros son duplicados o no. Esta implementación también impedirá la perdida de la información detallada correspondiente a los valores de similitud de los vectores de comparación.

Para implementar todas las mejoras anteriormente mencionadas, se propone la implementación de un nuevo método basado en la clasificación por umbral pero que no considere la similitud sumada sino que trabaje sobre los valores de similitud independientes de cada atributo. El nuevo método propuesto se describe a continuación:

a) El usuario ingresará umbrales superiores e inferiores para cada uno de los atributos que hayan sido seleccionados para la fase de comparación de registros. Tales valores podrán ser coincidentes si el usuario desea clasificar los registros en coincidentes y no coincidentes, o diferentes si se desea clasificarlos en coincidentes, no coincidentes y posiblemente coincidentes. Tales umbrales se ingresarán en forma de dos vectores; uno correspondiente a los umbrales superiores y otro correspondiente a los umbrales inferiores.

b) El usuario ingresará valores de ponderación (pesos) para cada uno de los atributos que hayan sido seleccionados para la fase de comparación de registros. Estos valores corresponden al poder que cada atributo tiene de diferenciar dos registros diferentes. Tales valores serán determinados por el usuario final de acuerdo a su conocimiento de las características de los datos en cuestión. Tales valores se ingresarán en forma de un vector.

c) Los vectores de umbrales y de ponderación se validarán para asegurar que el usuario no haya ingresado caracteres inválidos o que el umbral inferior sea igual al superior.

d) Se inicializarán a cero tres variables que mantendrán una puntuación para determinar si los registros del vector de comparación se clasifican como duplicados, no duplicados o posiblemente duplicados. La puntuación anteriormente referida se calculará a partir de los vectores de umbrales y el vector de ponderaciones.

e) Se comparará el valor de similitud existente del atributo en cuestión del vector de comparación con los umbrales superiores e inferiores ingresados por el usuario. Si el valor de similitud del atributo es menor al umbral inferior, la ponderación asociada a dicho atributo se sumará a la variable de puntuación que implica un estado de clasificación de no duplicado. De la misma forma, si el valor de similitud está entre los umbrales superior e inferior, la puntuación de tal atributo se suma a la variable que implica un estado de clasificación de posiblemente duplicado. Finalmente, si el valor del atributo en cuestión es mayor o igual al del umbral superior, la suma de la ponderación se hace a la variable que implica una clasificación de duplicado.

f) Se repite el paso anterior para todos los atributos del vector de comparación.

g) Se determina la variable con mayor puntuación. Si fue la variable asociada a una clasificación de duplicado, la pareja de registros se clasifica de esa manera. Se procede de igual manera para los otros dos posibles estados de clasificación; si no existe un valor máximo único de las variables, entonces los registros se clasifican como posiblemente duplicados.

h) Se repite desde el paso d) hasta el paso g) para todos los vectores resultantes de la fase de comparación.

La Figura 6.1 muestra el algoritmo anteriormente descrito en forma de Diagrama de Flujo. La notación utilizada para la elaboración de este último es la descrita en [66]. Además, la siguiente sección muestra la implementación del algoritmo propuesto. Dicho método elimina las desventajas del clasificador por umbral de similitud sumada que se expusieron en la sección anterior. De igual manera, en la sección 6.2.1 se ejemplifica el algoritmo y se demuestra la eliminación de las desventajas anteriormente mencionadas al escoger valores apropiados de umbrales y ponderaciones.

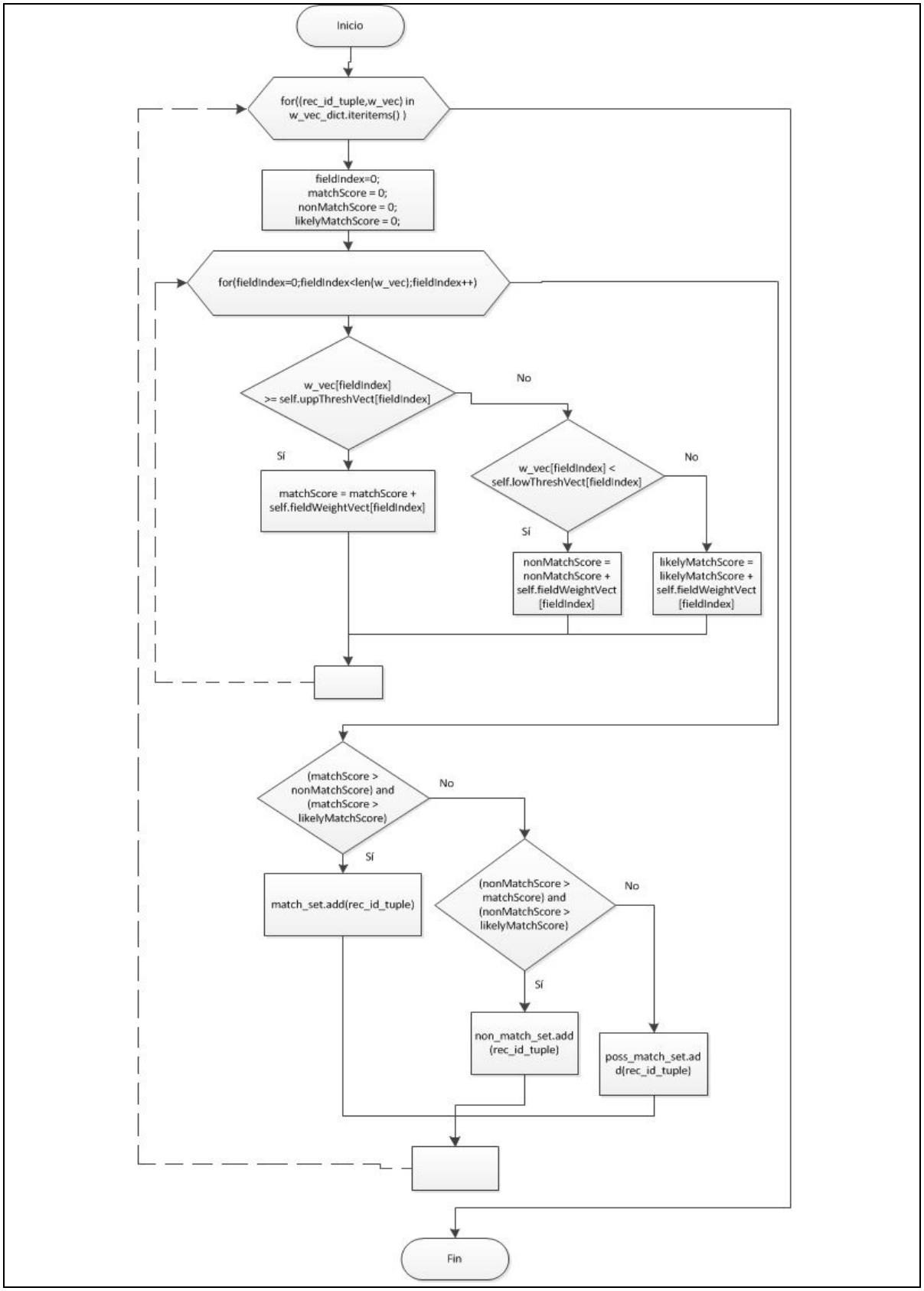


Figura 6.1 Diagrama de Flujo del nuevo método de clasificación

6.1.3.- Implementación del método de mejora

La implementación del método de mejora anteriormente descrito se realizó sobre el módulo “classification.py” del sistema Febrl. La Figura 6.2 muestra el código principal de la implementación.

```
  def classify(self, w_vec_dict):
    auxiliary.check_is_dictionary('w_vec_dict', w_vec_dict)

    match_set =      set()
    non_match_set =  set()
    poss_match_set = set()

    for (rec_id_tuple, w_vec) in w_vec_dict.iteritems():
      fieldIndex = 0
      matchScore = 0
      nonMatchScore = 0
      likelyMatchScore = 0

      for fieldIndex in range(len(w_vec)):
        if (w_vec[fieldIndex] >= self.uppThreshVect[fieldIndex]):
          matchScore = matchScore + self.fieldWeightVect[fieldIndex]

        elif (w_vec[fieldIndex] < self.lowThreshVect[fieldIndex]):
          nonMatchScore = nonMatchScore + \
              self.fieldWeightVect[fieldIndex]

        else:
          likelyMatchScore = likelyMatchScore + \
              self.fieldWeightVect[fieldIndex]

      if ((matchScore > nonMatchScore) and \
        (matchScore > likelyMatchScore)):
        match_set.add(rec_id_tuple)

      elif ((nonMatchScore > matchScore) and \
        (nonMatchScore > likelyMatchScore)):
        non_match_set.add(rec_id_tuple)

      else:
        poss_match_set.add(rec_id_tuple)

    assert (len(match_set) + len(non_match_set) + \
      len(poss_match_set)) == len(w_vec_dict)

    logging.info('Classified %d weight vectors: %d as matches, %d as ' %\
                 (len(w_vec_dict), len(match_set), len(non_match_set)) +\
                 'non-matches, and %d as possible matches' % \
                 (len(poss_match_set)))

    return match_set, non_match_set, poss_match_set
```

Figura 6.2 Código principal de la implementación del método de mejora de la fase de clasificación

El código anteriormente mostrado corresponde únicamente a la implementación del algoritmo como tal sobre el Febrl. Además de esto, fue necesario escribir el código correspondiente a los siguientes puntos:

a) Modificación a la interfaz gráfica de Ferbl para detectar la existencia del nuevo método y poder llamarlo

b) Verificaciones de la validez de los umbrales y ponderaciones proporcionadas por el usuario

c) Interacción del nuevo método con la salida estándar de Febrl para poder entregar los resultados del algoritmo

Tales funcionalidades son requeridas para que el nuevo método de clasificación pueda utilizarse desde el Sistema Evaluador de Calidad de Datos en MySQL; debido a la gran extensión del código fuente que las implementa, éste no se muestra en el presente trabajo escrito.

6.2.- Pruebas de experimentación del nuevo método

Una vez que el nuevo método se implementó e integró a Febrl, se diseñó y realizó un plan de pruebas para evaluar la funcionalidad y las ventajas y desventajas que el nuevo algoritmo presenta. A continuación se explican brevemente las pruebas que integran dicho plan:

a) Prueba ilustrativa: Esta prueba se realiza sobre los registros presentes en la Tabla 6.1 anteriormente mostrada. Esta prueba tiene la finalidad de ilustrar el funcionamiento básico del nuevo método y demostrar que elimina las desventajas del clasificador por umbral de similitud sumada descritas a lo largo de este capítulo.

b) Prueba de volumen: Esta prueba tiene como objeto realizar un análisis comparativo entre el nuevo método implementado y los métodos de clasificación no supervisada ya existentes en Febrl bajo un volumen significativo de datos. Esto permitirá el evaluar el desempeño del nuevo método en circunstancias similares a las de un proyecto real de Vinculación de Registros y sus resultados frente a los que arrojen los demás métodos de clasificación no supervisada.

Para la prueba de volumen es necesario obtener datos de entrada que permitan probar el nuevo método de manera efectiva. Para ello se generó un conjunto de datos de prueba con la herramienta “dsgen” (Data Set Generator). Dicha

herramienta es parte de Febrl y permite la generación de datos de prueba con características específicas. La Tabla 6.2 muestra las características del conjunto de datos de entrada para la prueba de volumen.

Número de registros totales	2000
Número de registros originales	1800
Número de registros duplicados	200
Número máximo de registros duplicados para un registro original que posea duplicados	3
Número máximo de diferencias entre el registro original y el registro duplicado por atributo	2
Número máximo de diferencias entre el registro original y el registro duplicado	8
Atributos de los datos	"rec_id", "given_name", "surname", "street_number", "address_1", "address_2", "suburb", "postcode", "state", "date_of_birth", "age", "phone_number", "soc_sec_id"

Tabla 6.2 Características de los datos de entrada para la prueba de volumen

6.2.1.- Prueba ilustrativa

Considere la Tabla 6.1 mostrada con anterioridad. Como ya se mencionó, los datos contenidos en dicha tabla no pueden ser clasificados correctamente por el clasificador por umbral de similitud sumada debido a las desventajas que este posee. Dichos datos pueden ser clasificados correctamente mediante el nuevo método implementado. La Tabla 6.3 muestra valores de umbrales y ponderaciones que permiten la clasificación correcta.

	GivenName	**Surname**	**StrNum**	**StrName**	**Suburb**	**BDay**	**BMonth**	**BYear**
Umbral superior	0.8	0.8	0.7	0.8	0.8	0.6	0.6	0.75
Umbral inferior	0.6	0.6	0.5	0.6	0.6	0.5	0.5	0.75
Ponderación del atributo	3	5	1	2	2	1	1	4

Tabla 6.3 Umbrales y pesos para el nuevo método implementado que permiten la clasificación correcta de los registros de la Tabla 6.1

Con los valores de umbrales y ponderaciones de la Tabla 6.3, el algoritmo calcula las puntuaciones que cada atributo tiene para un estado de clasificación dado. La Tabla 6.4 ilustra el uso de los umbrales y ponderaciones de la Tabla 6.3 para la clasificación correcta de los registros de la Tabla 6.1 mediante sus vectores de comparación. Es necesario notar, para la Tabla 6.4, que en el renglón de "Puntuación tras umbrales", la abreviatura "c." corresponde a coincidente, "n.c." corresponde a no coincidente y "p.c." corresponde a posiblemente coincidente, mientras que los vectores de comparación de las parejas de registros aparecen sombreados debajo de su pareja de registros correspondiente.

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear
a1	john	smith	18	miller st	dickson	12	11	1970
b1	jonny	smyth	73	miller st	dixon	11	12	1970
(a1,b1)	0.6	0.8	0.0	1.0	0.6	0.5	0.5	1.0
Umbral superior	0.8	0.8	0.7	0.8	0.8	0.6	0.6	0.75
Umbral inferior	0.6	0.6	0.5	0.6	0.6	0.5	0.5	0.75
Ponderación del atributo	3	5	1	2	2	1	1	4
Puntuación tras umbrales	+3 para p.c.	+5 para c.	+1 para n.c.	+2 para c.	+2 para p.c.	+1 para p.c.	+1 para p.c.	+4 para c.
a2	mary	Harris	42	swamp rd	sydney	21	4	1918
b2	mandy	garrett	42	smither pl	sydneyham	27	4	1979
(a2,b2)	0.6	0.4	1.0	0.4	0.6	0.5	1.0	0.5
Umbral superior	0.8	0.8	0.7	0.8	0.8	0.6	0.6	0.75
Umbral inferior	0.6	0.6	0.5	0.6	0.6	0.5	0.5	0.75
Ponderación del atributo	3	5	1	2	2	1	1	4
Puntuación del atributo	+3 para p.c.	+5 para n.c.	+1 para c.	+2 para n.c.	+2 para p.c.	+1 para p.c.	+1 para c.	+4 para n.c.

Tabla 6.4 Uso de umbrales y ponderaciones para clasificar correctamente los registros de la Tabla 6.1

La clasificación final depende de qué variable de clasificación haya tenido una mayor puntuación. La Tabla 6.5 muestra las puntuaciones obtenidas para las parejas de registros junto con su estado resultante de clasificación. En dicha tabla se observa que la pareja de registros (a1,b1) ha sido clasificada como coincidente, mientras que la pareja (a2,b2) ha sido clasificada como no coincidente, demostrando que el nuevo método carece de las desventajas del clasificador por umbral de similitud sumada descrito anteriormente.

Pareja de registros	Puntuación para clasificación coincidente	Puntuación para clasificación no coincidente	Puntuación para clasificación posiblemente coincidente	Estado de clasificación resultante
(a1,b1)	11	1	7	Coincidente
(a2,b2)	2	11	6	No coincidente

Tabla 6.5 Puntuaciones finales del algoritmo y resultados de clasificación

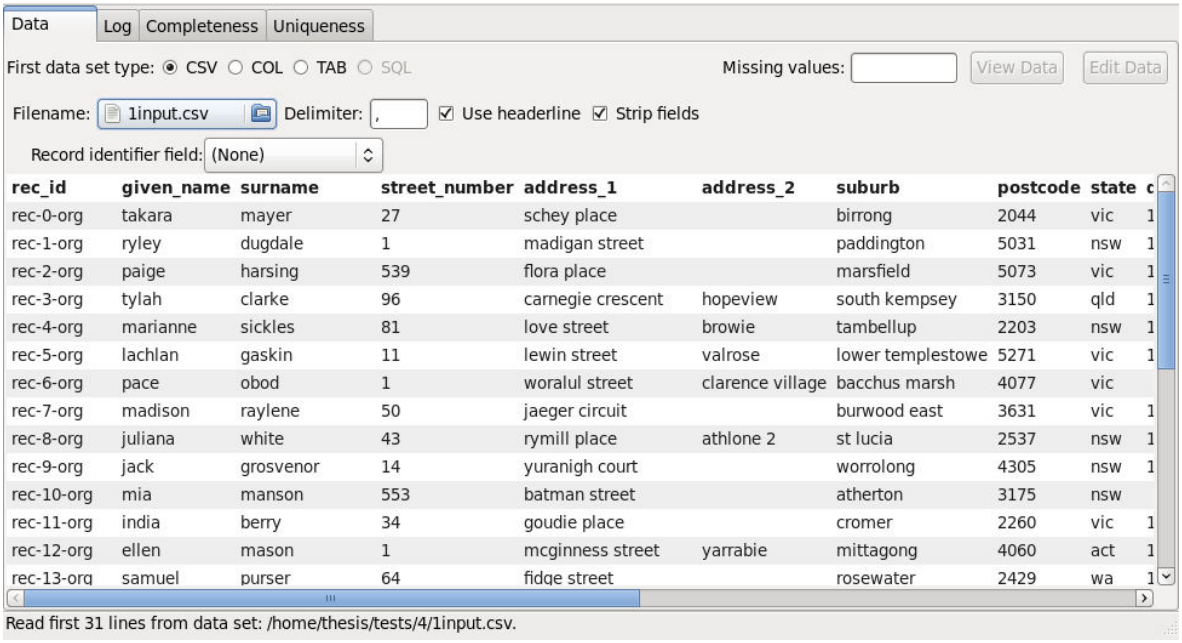
6.2.2.- Prueba de volumen

Considere los datos descritos en la Tabla 6.2. Para clasificarlos se determinó usar los valores de umbrales y ponderaciones descritos en la Tabla 6.6.

Atributo	Umbral inferior	Umbral superior	Ponderación
given_name	0.6	0.75	3
street_number	0.5	0.6	1
address_1	0.6	0.8	3
suburb	0.6	0.7	3
postcode	0.6	0.7	3
state	0.5	0.75	1
date_of_birth	0.6	0.8	4
phone_number	0.6	0.75	2
soc_sec_id	0.65	0.85	4

Tabla 6.6 Umbrales y ponderaciones a utilizar para la prueba de volumen sobre los datos de la Tabla 6.2

Una vez determinados dichos coeficientes se sigue el procedimiento descrito en el Capítulo IV que corresponde al proceso de Vinculación de Registros en Febrl. La Figura 6.3 muestra la apertura del archivo “1input.csv”, el cual corresponde a los datos de entrada generados mediante el programa auxiliar “dsgen” para esta prueba de volumen.



rec_id	given_name	surname	street_number	address_1	address_2	suburb	postcode	state	c
rec-0-org	takara	mayer	27	schey place		birrong	2044	vic	1
rec-1-org	ryley	dugdale	1	madigan street		paddington	5031	nsw	1
rec-2-org	paige	harsing	539	flora place		marsfield	5073	vic	1
rec-3-org	tylah	clarke	96	carnegie crescent	hopeview	south kempsey	3150	qld	1
rec-4-org	marianne	sickles	81	love street	browie	tambellup	2203	nsw	1
rec-5-org	lachlan	gaskin	11	lewin street	valrose	lower templestowe	5271	vic	1
rec-6-org	pace	obod	1	woralul street	clarence village	bacchus marsh	4077	vic	
rec-7-org	madison	raylene	50	jaeger circuit		burwood east	3631	vic	1
rec-8-org	juliana	white	43	rymill place	athlone 2	st lucia	2537	nsw	1
rec-9-org	jack	grosvenor	14	yuranigh court		worrolong	4305	nsw	1
rec-10-org	mia	manson	553	batman street		atherton	3175	nsw	
rec-11-org	india	berry	34	goudie place		cromer	2260	vic	1
rec-12-org	ellen	mason	1	mcginness street	yarrabie	mittagong	4060	act	1
rec-13-org	samuel	purser	64	fidge street		rosewater	2429	wa	1

Figura 6.3 Apertura del archivo “1input.csv” el cual contiene los datos de entrada para la prueba de volumen

Para la fase de indexado se determinó utilizar el atributo “surname”, el cual corresponde al apellido. La Figura 6.4 muestra la selección de tal atributo para el indexado con la función de codificación fonética “Double Metaphone”.

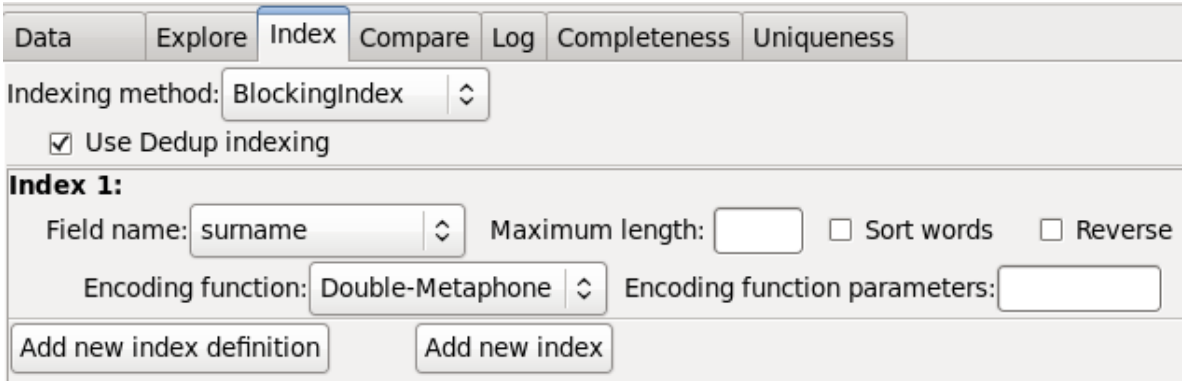


Figura 6.4 Indexado para la prueba de volumen

Para la fase de comparación se seleccionaron los atributos mostrados en la Tabla 6.6. El método de comparación escogido fue la Similitud por Distancia de Levenshtein para todos los casos. La Figura 6.5 muestra dicha configuración de la fase de comparación. Note que el atributo “surname” no ha sido considerado para la fase de comparación. Esto se debe a que fue la columna seleccionada para el indexado.

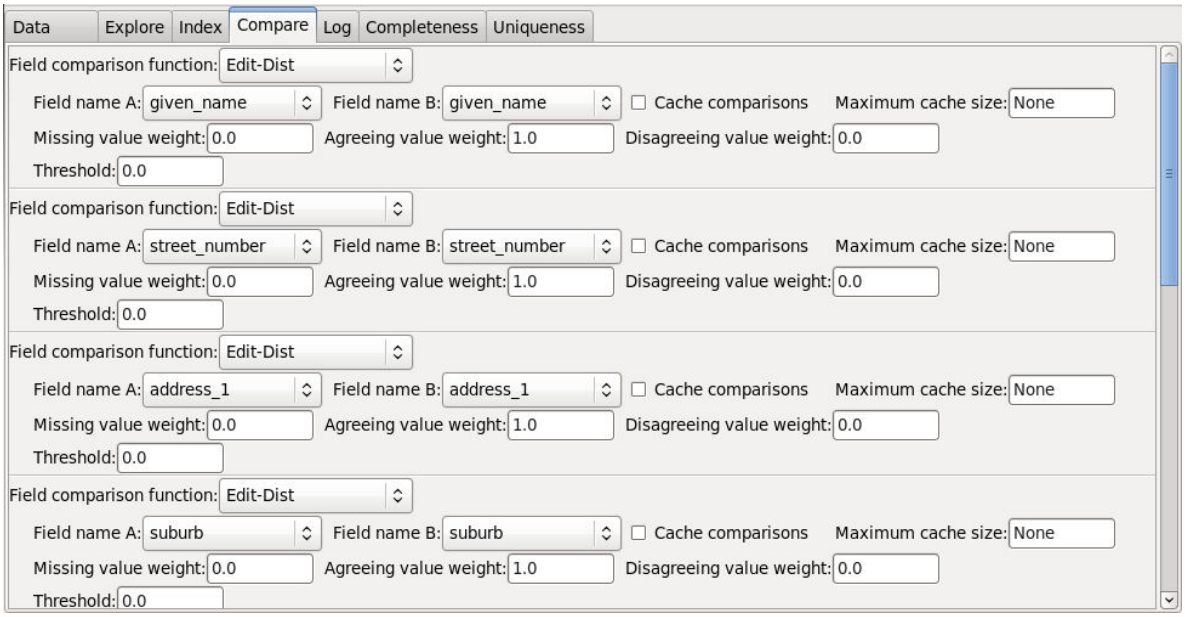


Figura 6.5 Uso de la Similitud por Distancia de Levenshtein sobre los atributos de la Tabla 6.6 para la fase de comparación

En la fase de comparación se selecciona el nuevo método implementado (mostrado como “ModifFellegiSunter”) y se ingresan los parámetros que éste requiere. La Figura 6.6 muestra el ingreso de los umbrales y las ponderaciones de la Tabla 6.6.

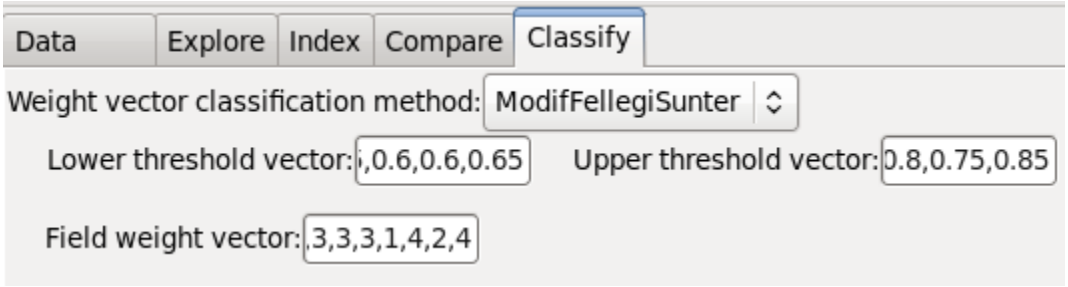


Figura 6.6 Clasificación con el nuevo método implementado (“ModifFellegiSunter”) mediante los umbrales y ponderaciones de la Tabla 6.6

La salida del proceso se realizará a un archivo igual a los datos de entrada más la adición de una columna como indicador del estado de clasificación de los registros. La Figura 6.7 muestra la selección del archivo “1input-match.csv” para almacenar la salida con el atributo “match_id” para el estado de clasificación; tal atributo se anexa después de las columnas originales de los datos iniciales.

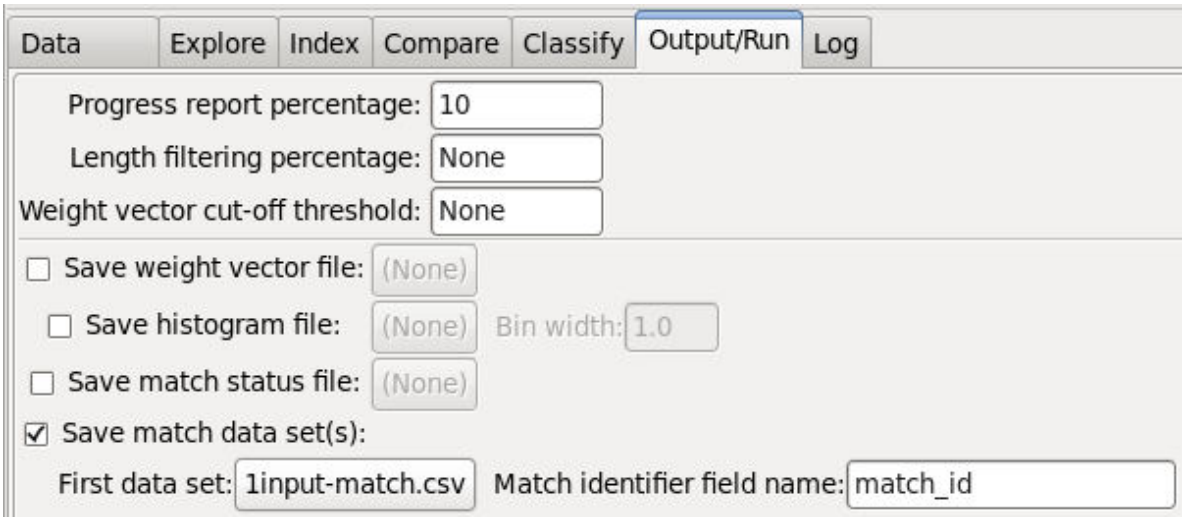


Figura 6.7 Configuración de la salida

Una vez realizada la clasificación se obtiene la salida en el formato especificado. La Figura 6.8 muestra una porción del archivo de salida "1input-match.csv". Note que la última columna corresponde al atributo "match_id", el cual no corresponde a los datos originales, sino que indica el estado de clasificación.

rec_id	given_name	surname	street_numb	address_1	address_2	suburb	postcode	state	date_of_birt	age	phone_numl	soc_sec_id	match_id
rec-0-org	takara	mayer	27	schey place		birrong	2044	vic	19790210		04 24214749	2520744	
rec-1-org	ryley	dugdale	1	madigan street		paddington	5031	nsw	19400209		07 15464380	9174816	
rec-2-org	paige	harsing	539	flora place		marsfield	5073	vic	19881119	33	03 24420355	3493428	
rec-3-org	tylah	clarke	96	carnegie cre	hopeview	south kemps	3150	qld	19530617	10	02 75302178	9377817	
rec-4-org	marianne	sickles	81	love street	browie	tambellup	2203	nsw	19040505	24	04 61255846	5425952	
rec-5-org	lachlan	gaskin	11	lewin street	valrose	lower templ	5271	vic	19131211	29	02 92260803	9540499	
rec-6-org	pace	obod	1	woralul stre	clarence villa	bacchus mar	4077	vic		27	02 50169537	8115293	
rec-7-org	madison	raylene	50	jaeger circuit		burwood eas	3631	vic	19381005	28	08 38246262	5253131	
rec-8-org	juliana	white	43	rymill place	athlone 2	st lucia	2537	nsw	19001020	28	03 11083679	6400873	
rec-9-org	jack	grosvenor	14	yuranigh court		worrolong	4305	nsw	19920525	34	02 28167229	4769432	
rec-10-org	mia	manson	553	batman street		atherton	3175	nsw		38	03 60873797	9710090	
rec-11-org	india	berry	34	goudie place		cromer	2260	vic	19710105	31	02 51141769	8914556	
rec-12-org	ellen	mason	1	mcginness st	yarrabie	mittagong	4060	act	19411028		08 58054781	3343827	
rec-13-org	samuel	purser	64	fidge street		rosewater	2429	wa	19950408	34	03 03923589	3546886	
rec-14-org	michael	hughston-wy	6	northmore c	purtell park	croydon nort	2429	qld	19320715	27	02 43755836	4580681	
rec-15-org	tristan	korff	15	byrne street	kurrajong	williamstow	3564	nsw	19220118	26	03 68255920	5776471	mid125
rec-15-dup-(	tristan	kordef	15	byrne street	kurrajong	williamstow	3564	nsw	19220118		03 95328671	5766371	mid125
rec-16-org	amy	huster	4	novar street		nunawading	3042	sa	19131027	31	03 45583927	3044583	
rec-17-org	jordan	crouch	12	cussen street		wanbi	4077	vic	19390224	27	07 31212958	7711619	
rec-18-org	trevor	brammy	36	hugh mckay	crescent	goonellabah	3152	nsw	19391111	27	08 15751716	4454267	
rec-19-org	liam	lahey	45	dane close		bacchus mar	7009		19560128	33	04 06446506	3783281	
rec-20-org	sarah	clarke	19	castieau street		malvern	2280	qld	19241231	41	02 43711093	8298174	

Figura 6.8 Archivo de salida "1input-match.csv"

Para conocer el número de registros duplicados detectados es necesario realizar un análisis de los valores de la columna "match_id". Debido a que los datos de prueba contienen en algunos casos hasta tres duplicados por registro original, realizar dicho análisis es una labor extensiva por lo que los detalles de su realización no se incluyen en el presente trabajo escrito. Una vez concluido dicho

procedimiento se obtienen detalles de desempeño del nuevo método implementado.

La Tabla 6.7 muestra información que posibilita la comparación del nuevo método sobre los ya existentes en Febrl para los datos de esta prueba de volumen. Para los valores ahí mostrados, el término “coincidencias verdaderas detectadas indirectamente” se refiere a que tales registros fueron detectados inicialmente como posiblemente duplicados y tras Revisión Secretarial se determinó que eran duplicados de algún otro registro. Por otro lado el término “coincidencias verdaderas detectadas directamente” significa que tales registros fueron identificados de manera inmediata por el nuevo método como duplicados, sin tener que pasar por un proceso de Revisión Secretarial. Finalmente, el término “no coincidencias falsas” se refiere a aquellos registros duplicados que no fueron detectados. Estos mismos criterios se utilizarán para realizar un análisis comparativo entre el nuevo algoritmo y aquellos métodos de clasificación no supervisada ya existentes en Febrl.

Resultados de la clasificación	**Número de registros**
Coincidencias verdaderas detectadas directamente	199
Coincidencias verdaderas detectadas indirectamente	1
Coincidencias verdaderas totales	200
Coincidencias falsas	0
No coincidencias falsas	0

Tabla 6.7 Análisis de resultados de la prueba de volumen para el nuevo método de clasificación

Al analizar la información mostrada en la Tabla 6.7 se aprecia que, de acuerdo a la descripción de los datos de entrada para la prueba de volumen mostrada en la Tabla 6.2 y los valores de umbrales y ponderaciones descritos en la Tabla 6.6, el nuevo método de clasificación detectó la totalidad de los duplicados. Además, se observa que solamente un registro tuvo que ser detectado por Revisión Secretarial y que no hubo registros erróneamente clasificados como duplicados.

6.3.- Análisis de resultados comparativo frente a los métodos de clasificación no supervisada ya existentes en Febrl

Para poder comparar el desempeño del nuevo algoritmo implementado frente a los métodos de clasificación no supervisada ya existentes en Febrl se determinó el realizar la prueba de volumen descrita en la sección anterior sobre tales métodos no supervisados. Los métodos de clasificación y sus respectivos parámetros utilizados para las pruebas comparativas de volumen son los siguientes:

a) Clasificador por umbral de similitud sumada: Se utilizó un umbral inferior de 15 y un umbral superior de 17.5.

b) K medias: Se utilizó la distancia euclidiana, inicialización de vectores en valores máximos y mínimos. Además, se asignó un máximo de 10000 iteraciones para este método.

c) Farthest First: Se utilizó distancia euclidiana y un máximo de 10000 iteraciones.

Para hacer más competitiva la prueba, para la fase de comparación en la prueba de volumen del clasificador por umbral de similitud sumada, se pidió a Febrl utilizar valores no normalizados para los vectores de comparación como se muestra en la Tabla 6.8. Para los otros dos métodos el procedimiento es igual que el descrito para la prueba de volumen del nuevo método en la sección anterior, difiriendo únicamente en el método de clasificación seleccionado.

Atributo	Coeficiente de Desnormalización
given_name	3
street_number	1
address_1	3
suburb	3
postcode	3
state	1
date_of_birth	4
phone_number	2
soc_sec_id	4

Tabla 6.8 Coeficientes para el uso de valores de similitud no normalizados en la fase de comparación para el clasificador por umbral de similitud sumada

Tras la realización de las pruebas de volumen para los métodos de clasificación no supervisada ya existentes en Febrl se obtuvieron los resultados mostrados en la Tabla 6.9. Además, se muestran los resultados obtenidos para el nuevo método implementado con el objeto de facilitar la comparación entre el rendimiento de los diferentes métodos de clasificación.

Método de clasificación	**Coincidencias verdaderas detectadas directamente (C.V.D.D.)**	**Coincidencias verdaderas detectadas indirectamente (C.V.D.I.)**	**Coincidencias verdaderas totales (C.V.T.)**	**Coincidencias falsas (C.F.)**	**No coincidencias falsas (N.C.F.)**
Clasificador por umbral de similitud sumada	182	11	193	0	7
K medias	199	1	200	2	0
Farthest First	199	1	200	1	0
Nuevo método implementado	199	1	200	0	0

Tabla 6.9 Resultados de la realización de pruebas de volumen sobre el nuevo método y los métodos de clasificación no supervisada ya existentes en Febrl

La Figura 6.9 muestra un histograma elaborado a partir de la información de la Tabla 6.9. Considere que las abreviaturas “C.V.T”, “N.C.F.”, “C.F.”, “C.V.D.I” y “C.V.D.D.” corresponden a las mostradas en las columnas de la Tabla 6.9.

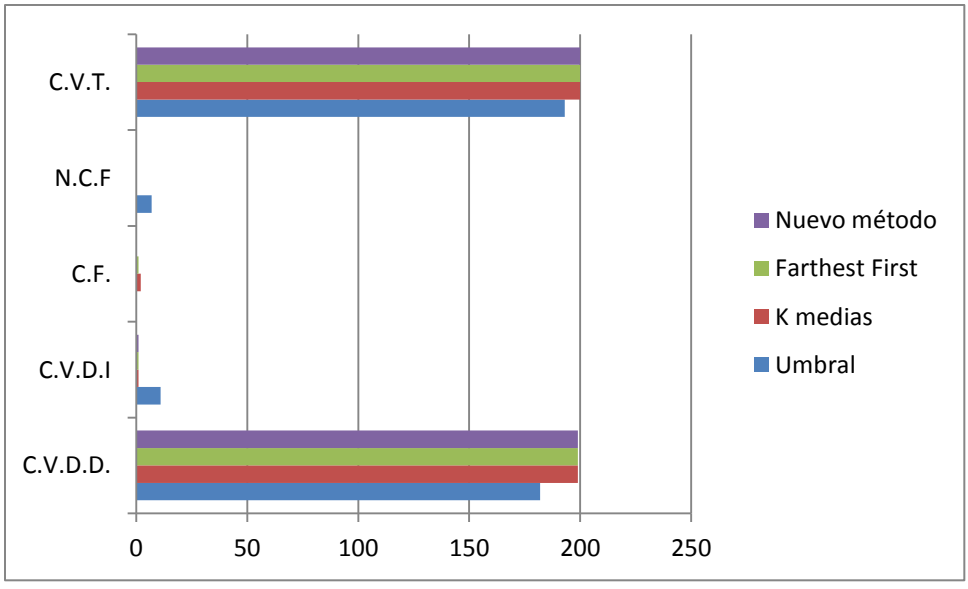


Figura 6.9 Histograma de los resultados de la prueba de volumen

Los datos de la Tabla 6.9 y de la Figura 6.9 permiten el siguiente análisis sobre los resultados que los diferentes métodos tuvieron sobre la prueba de volumen:

a) El clasificador por umbral de similitud sumada tuvo el peor rendimiento en la prueba de volumen, pues siete registros duplicados no fueron detectados. Además, once registros requirieron inspección manual mediante el proceso de Revisión Secretarial.

b) El método K medias detectó la totalidad de los registros duplicados; de ellos solamente uno fue detectado por Revisión Secretarial. Sin embargo, dos registros originales fueron erróneamente clasificados como duplicados.

c) El método Farthest First detectó la totalidad de los registros duplicados; de ellos solamente uno fue detectado por Revisión Secretarial. Además, un registro original fue erróneamente clasificado como duplicado.

d) El nuevo método implementado detectó la totalidad de los registros duplicados; de ellos solamente uno requirió ser detectado por Revisión Secretarial. No hubo registros erróneamente clasificados como no duplicados.

Los resultados anteriores muestran que el nuevo método implementado fue superior a los otros clasificadores no supervisados para la prueba de volumen descrita a lo largo de este capítulo. Sin embargo, se requiere de una exhaustiva experimentación posterior con diferentes parámetros y fuentes de datos para obtener conclusiones más significativas del comportamiento del nuevo método. Tal experimentación, debido a su extensión y complejidad, escapa al alcance del presente trabajo de tesis.

Capítulo VII

Conclusión

La realización del presente trabajo de tesis tuvo como resultado la creación de un software prototipo llamado “Sistema Evaluador de Calidad de Datos en MySQL”. Este software tiene la capacidad de realizar la evaluación de las métricas de Calidad de Datos descritas en el Capítulo II para cualquier Base de Datos existente en un Sistema Manejador de Base de Datos MySQL.

El Sistema Evaluador de Calidad de Datos en MySQL también posee la capacidad de realizar el proceso de Vinculación de Registros. Esta capacidad es debida al software Febrl original, el cual se extendió con la implementación de la evaluación de Calidad de Datos y con el desarrollo del nuevo método de clasificación para el proceso de Vinculación de Registros. Dicho método posibilita el uso del software por personas con poca instrucción formal en computación, pero con un buen conocimiento empírico de los datos sobre los cuales se realizará el proceso de Vinculación de Registros.

El Sistema Evaluador de Calidad de Datos en MySQL también posee la capacidad de realizar el volcado de una Base de Datos a un archivo de texto plano separado por comas. De esa manera, es posible acceder a una Base de Datos dada, evaluar su calidad, volcar los datos a un archivo y entonces realizar un proceso de Vinculación de Registros sobre tal fuente de datos. Esto permite que el Sistema Evaluador de Calidad de Datos en MySQL sea una herramienta versátil y útil para quienes trabajen con Bases de Datos.

Finalmente, la elaboración del presente trabajo de tesis permite al autor las siguientes conclusiones:

a) El nuevo método implementado elimina las desventajas de la clasificación por umbral de similitud sumada abordadas en el Capítulo VI.

b) Aun cuando se trate de asignar pesos discriminativos a cada atributo en un clasificador por umbral de similitud sumada al utilizar valores no normalizados en la fase de comparación, la suma de los valores de similitud hace que la información específica que cada atributo tiene para discriminar registros entre sí se pierda, cosa que no ocurre con el nuevo método implementado, por lo que éste tiene un desempeño superior.

c) Para los datos y parámetros descritos a lo largo del Capítulo VI para las pruebas de volumen, el nuevo método implementado tuvo el mejor

rendimiento de los cuatro métodos de clasificación no supervisada que se evaluaron.

d) El desempeño del nuevo método implementado es altamente dependiente de los umbrales y las ponderaciones que el usuario determine. El proceso de determinar tales valores es altamente empírico por lo que gente con buenos conocimientos de los datos y que se familiarice con aunque sea unas cuantas funciones de comparación, puede obtener altos porcentajes de clasificación exitosa con el nuevo método.

e) Para tanto la prueba ilustrativa como la prueba de volumen para el nuevo método, los valores de los umbrales y ponderaciones mostrados en el presente trabajo de tesis, fueron los primeros que se determinaron empíricamente tras analizar visualmente los datos por un par de minutos y una vez que se tenía cierta familiaridad con el método de comparación de Similitud por Distancia de Levenshtein, por lo que se concluye que el nuevo método puede dar buenos resultados incluso para usuarios relativamente inexpertos.

Las conclusiones anteriores fueron resultado de las implementaciones y pruebas realizadas bajo las condiciones descritas en el presente trabajo de tesis. Se requiere una extensiva experimentación adicional para determinar las circunstancias y parámetros en los que el nuevo método implementado es superior a otros métodos de clasificación no supervisada. Dicha experimentación, debido a su complejidad, se reserva para labores posteriores.

Referencias Bibliográficas

[1] Página del Dr. Peter Christen [en línea]. Disponible en <http://cs.anu.edu.au/~Peter.Christen/>. Fecha de consulta: 17/Noviembre/2013.

[2] Página del proyecto Febrl de la Universidad Nacional de Australia [en línea]. Disponible en <http://cs.anu.edu.au/~Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/>. Fecha de consulta: 17/Noviembre/2013.

[3] Batini, C., & Scannapieco, M. (2006). Data quality: concepts, methodologies and techniques. Springer. Prefacio.

[4] Batini, C., & Scannapieco, M. (2006). Data quality: concepts, methodologies and techniques. Springer. pp. 4.

[5] Eckerson, W. W. (2002). Data quality and the bottom line. *TDWI Report, The Data Warehouse Institute*. pp. 3.

[6] Eckerson, W. W. (2002). Data quality and the bottom line. TDWI Report, The Data Warehouse Institute. pp. 8.

[7] Eckerson, W. W. (2002). Data quality and the bottom line. TDWI Report, The Data Warehouse Institute. pp. 10.

[8] Batini, C., & Scannapieco, M. (2006). Data quality: concepts, methodologies and techniques. Springer. pp. 1-4.

[9] The Six Primary Dimensions of Data Quality Assessment [En línea]. Disponible en <http://www.damauk.org/RWFilePub.php?&cat=403&dx=2&ob=3&rpn=catviewleafpublic403&id=106193>. DAMA UK (2013). Fecha de Consulta: 02/Febrero/2014. pp. 1-2

[10] Naumann, F., Freytag, J. C., & Leser, U. (2004). Completeness of integrated information sources. *Information Systems*, *29*(7), pp. 603.

[11] Naumann, F., Freytag, J. C., & Leser, U. (2004). Completeness of integrated information sources. *Information Systems*, *29*(7), pp. 604.

[12] Naumann, F., Freytag, J. C., & Leser, U. (2004). Completeness of integrated information sources. *Information Systems*, *29*(7), pp. 605.

[13] Ángeles, M., & MacKinnon, L. M. (2010). Assessing data quality of integrated data by quality aggregation of its ancestors. *Computación y Sistemas*, *13*(3), pp. 336.

[14] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. (vii)

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

[16] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 4.

[17] Bleiholder, J., & Naumann, F. (2008). Data fusion. *ACM Computing Surveys (CSUR)*, *41*(1), 1.

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

[19] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 3-4.

[20] Han, J., & Kamber, M. (2006). *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan kaufmann.

[21] Brook, E. L., Rosman, D. L., Holman, C. D. J., & Trutwein, B. (2004). Summary report: research outputs project, WA Data Linkage Unit (1995-2003). *population*, *23*(5), 464-467.

[22] Clark, D. E. (2004). Practical introduction to record linkage for injury research. *Injury Prevention*, *10*(3), 186-191.

[23] Christen, P. (2008). Febrl: a freely available record linkage system with a graphical user interface. In *Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80* (pp. 18). Australian Computer Society, Inc..

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

[25] Baxter, R., Christen, P., & Churches, T. (2003, August). A comparison of fast blocking methods for record linkage. In *ACM SIGKDD* (Vol. 3, pp. 25-27).

[26] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 29-30.

[27] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 25-26.

[28] Skiena, S. (2008). The algorithm design manual. Springer. pp. 39.

[29] Goiser, K., & Christen, P. (2006, November). Towards automated record linkage. In *Proceedings of the fifth Australasian conference on Data mining and analystics-Volume 61* (pp. 25). Australian Computer Society, Inc..

[30] Ensuring Data Integrity in Health Information Exchange [En línea]. Disponible en <http://library.ahima.org/xpedio/groups/public/documents/ahima/bok1_049675.pdf>. AHIMA (2012). Fecha de consulta: 21/Enero/2014. pp. 2

[31] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 26.

[32] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 28.

[33] Skiena, S. (2008). The algorithm design manual. Springer. pp. 38.

[34] Patente de Soundex [En línea]. Disponible en <http://www.google.com/patents?vid=1261167>. U.S. Patent and Trademark Office (1918). Fecha de consulta: 01/Febrero/2014. pp. 1.

[35] Phonetic Matching: A Better Soundex [En línea]. Disponible en <http://stevemorse.org/phonetics/bmpm2.htm>. Fecha de consulta: 21/Marzo/2014.

[36] Philips, L. (2000). The double metaphone search algorithm. *C/C++ users journal*, *18*(6), 38-43.

[37] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 101.

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

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

[40] Cohen, W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation* (Vol. 3, pp. 73-78).

[41] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 109.

[42] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 108-109.

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

[44] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 110.

[45] Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, *33*(1), pp. 37.

[46] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 104.

[47] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 129.

[48] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 163-164.

[49] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 131.

[50] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 131-132.

[51] Vora, P., & Oza, B. (2013). A survey on k-mean clustering and particle swarm optimization. *Int. J. Sci. Modern Eng*, *1*, pp. 24.

[52] Vattani, A. (2011). K-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, *45*(4), pp. 596.

[53] Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In *New Directions in Statistical Physics* (pp. 273-282). Springer Berlin Heidelberg.

[54] Vora, P., & Oza, B. (2013). A survey on k-mean clustering and particle swarm optimization. *Int. J. Sci. Modern Eng*, *1*, pp. 25.

[55] Ball, G. H., & Hall, D. J. (1965). *ISODATA, a novel method of data analysis and pattern classification*. STANFORD RESEARCH INST MENLO PARK CA.

[56] Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, *38*, 293-306.

[57] Aher, S. B., & Lobo, L. M. R. J. (2012). A Comparative Study for Selecting the Best Unsupervised Learning Algorithm in E-Learning System. *International Journal of Computer Applications*, *41*(3), pp. 29.

[58] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 174-175.

[59] Guiver, T. (2011). Sampling-based clerical review methods in probabilistic linking. Methodology Research Paper, Cat. no. 1351.0.55.034. Australian Bureau of Statistics, Canberra.

[60] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 175.

[61] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 167.

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

[63] Referencia a las tablas de la Base de Datos INFORMATION_SCHEMA en el Manual de Referencia de MySQL [En línea]. Disponible en <http://dev.mysql.com/doc/refman/5.6/en/information-schema.html>. Fecha de consulta: 21/Enero/2014.

[64] Pagina web del benchmark TPC-H [En línea]. Disponible en <http://www.tpc.org/tpch/>. Fecha de consulta: 28/Jul/2013.

[65] Christen, P. (2012). Data matching. *Data-Centric Systems and Appl., Springer*. pp. 130.

[66] Solórzano, J.F., Millán, A., Solórzano, C. & Sánchez, A. (2012). Fundamentos de Computación: Panorama histórico y programación. UNAM, Facultad de Ingeniería, pp. 297-299.