Técnicas de agrupamiento de valores similares para la reducción de la inconsistencia en bases de datos

Sergio Luján Mora

Director: Manuel Palomar Sanz

Trabajo de investigación tutelado obligatorio (6 créditos)

Programa de doctorado 3798
Ingeniería lingüística, aprendizaje automático
y reconocimiento de formas

Departamento de Lenguajes y Sistemas Informáticos

Universitat d'Alacant
Universidad de Alicante

13 de noviembre de 2000

MANUEL PALOMAR SANZ, profesor titular de universidad del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante,

CERTIFICA que Sergio Luján Mora ha realizado bajo mí dirección la memoria de investigación "Técnicas de agrupamiento de valores similares para la reducción de la inconsistencia en bases de datos" como parte del programa de doctorado 3798 del Departamento de Lenguajes y Sistemas Informáticos y AUTORIZO la lectura y defensa de dicha memoria de investigación.

San Vicente del Raspeig, 13 de noviembre de 2000

Manuel Palomar Sanz

V.º B.º del Director del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante

Rafael C. Carrasco Jiménez

Técnicas de agrupamiento de valores similares para la reducción de la inconsistencia en bases de datos

Sergio Luján Mora

13 de noviembre de 2000

Sergio Luján Mora (slujan@dlsi.ua.es)
http://www.dlsi.ua.es/∼slujan/

Departamento de Lenguajes y Sistemas Informáticos
http://www.dlsi.ua.es

Universidad de Alicante
http://www.ua.es

Campus de San Vicente del Raspeig
Ap. Correos 99
E-03080 Alicante (España)

Prefacio

Durante el último año de mis estudios universitarios de segundo ciclo estuve trabajando en el Departamento de Informática del Ayuntamiento de San Vicente del Raspeig. Mi principal responsabilidad fue el desarrollo de nuevas bases de datos y el mantenimiento de las existentes.

Durante ese periodo de tiempo pude comprobar como toda la teoría sobre bases de datos (normalización, clave primaria, clave ajena, integridad referencial, etc.) que estudié en las asignaturas de bases de datos prácticamente no podía aplicarla, porque tenía que trabajar sobre bases de datos existentes que habían sido diseñadas por "informáticos de tiempo libre". Al carecer de unos conocimientos teóricos mínimos, los diseños realizados por los "informáticos de tiempo libre" originaban numerosos problemas de explotación, mantenimiento y actualización.

Uno de los principales problemas con los que me enfrenté fue la inconsistencia en la información, lo que producía a su vez la duplicación de la misma: un mismo dato repetido varias veces porque estaba escrito de distintas formas, ya fuese porque existieran errores ortográficos, abreviaturas o distinto orden en las palabras, por ejemplo.

Un buen diseño de una base de datos puede evitar este problema. Por ejemplo, si uno de los campos de un registro tiene que almacenar la provincia, en vez de dejar que el usuario introduzca el nombre de la provincia, es mejor obligarle a que la seleccione de una lista desplegable que obtiene los nombres de las provincias de una tabla de la base de datos que almacena el nombre de todas las provincias. En esta solución, es necesario crear relaciones, establecer claves primarias y ajenas, conocimientos que no suelen poseer los "informáticos de tiempo libre".

Sin embargo, incluso con un buen diseño también puede surgir el problema de la inconsistencia (no siempre tienen la culpa los "informáticos de tiempo libre"). Cuando se desea aprovechar la información almacenada de forma dispersa en distintas "islas de información" (por ejemplo, para realizar minería de datos o construir un almacén de datos), también puede surgir el problema de la inconsistencia: aunque cada una de las bases de datos que se desea

integrar sea consistente ella misma, entre ellas pueden existir discrepancias a la hora de nombrar los datos que almacenan. Pensemos, por ejemplo, en dos bases de datos que almacenan datos sobre clientes y sus direcciones, y los nombres de las provincias en una de ellas están en español y en la otra en valenciano.

Para lograr una adecuada relación con los clientes, primero hay que empezar por conocerlos. Desgraciadamente, la mayoría de las empresas tienen problemas para identificar y contactar con sus clientes por deficiencias en sus bases de datos. No es raro que una misma empresa maneje varias bases de datos y que existan clientes repetidos entre ellas. No es raro tampoco que el mismo cliente esté registrado en la misma base de datos de diversas formas. Además, es muy común encontrar que la dirección de dichos clientes sea inconsistente o esté incompleta. Y por supuesto, muchísimas bases de datos sufren una notable falta de estandarización y están llenas de errores de captura y basura, lo que hace su manejo aún más difícil, además de entorpecer muchos de los esfuerzos de comunicación directa.

La experiencia "sufrida" en el Ayuntamiento de San Vicente del Raspeig me motivó a investigar sobre el problema de la inconsistencia de la información almacenada en una base de datos. En el trabajo que presento a continuación, desarrollo una técnica automática que permite detectar en una base de datos aquellos valores que hacen referencia a un mismo término. Básicamente, la técnica consiste en agrupar las cadenas que cumplen un criterio de similitud, basado en una serie de distancias entre cadenas. Una vez detectados y agrupados los valores similares, se pueden sustituir por un único valor, con lo que se consigue eliminar la inconsistencia y la duplicación de información.

Una parte importante de la investigación científica es la comunicación: para que un resultado científico sea importante tiene que ser comunicado, ya que si su autor se lo lleva a la tumba, no habrá servido de nada su esfuerzo. En el proceso de comunicación, es necesario prestar atención tanto al contenido como al continente. Este trabajo me ha permitido (obligado a) aprender LaTeX 2$_\varepsilon$ y el uso de distintos paquetes (`algorithmic`, `crop`, `graphicx`, `makeidx`, etc.). Aunque he tenido que invertir mucho tiempo en aprender a trabajar con LaTeX 2$_\varepsilon$, sin el empleo de esta herramienta de composición de textos, el tiempo necesario para redactar este trabajo (y los futuros trabajos que realice) habría sido superior al invertido y nunca habría quedado igual de bien. Espero que haya quedado un trabajo *decente*, ya que este es el primer documento que realizo en LaTeX 2$_\varepsilon$.

Alicante, 13 de noviembre de 2000 *Sergio Luján Mora*

Índice General

Índice de Tablas

Índice de Figuras

Índice de Algoritmos

Índice de Acrónimos

CJ *Coeficiente de Jaccard*
Similitud entre dos conjuntos. Se define como la división del número de elementos que comparte los dos conjuntos (elementos en común: la intersección) entre el número de elementos distintos de los dos conjuntos (la unión).

DE *Distancia de edición*
Otro nombre con el que se conoce la DL.

DIPP *Distancia invariante a la posición de las palabras*
Distancia entre dos cadenas de caracteres. Se calcula dividiendo las dos cadenas por palabras y emparejando las palabras de las dos cadenas de forma que la suma de las DL de las parejas sea mínima.

DIPPM *Distancia invariante a la posición de las palabras modificada*
Distancia entre dos cadenas de caracteres. Como la DIPP, pero se relaja la condición de emparejamiento exacto entre dos cadenas.

DIT *Distancia invariante trasposicional*
Distancia entre dos cadenas de caracteres. Se define como:

$$\mathrm{DIT}(x,y) = \frac{1}{2}\left(\sum_{i=1}^{37} |x_{\alpha_i} - y_{\alpha_i}| + \mid \mathrm{LC}(x) - \mathrm{LC}(y) \mid\right) \qquad (1)$$

DL *Distancia de Levenshtein*
Distancia entre dos cadenas de caracteres. Se calcula como el mínimo coste de transformar una cadena en otra mediante la inserción, el borrado o la substitución de un carácter.

DLON *Distancia longitudinal*
Distancia entre dos cadenas de caracteres. Diferencia en valor absoluto de las LC de las dos cadenas.

FD *Factor de duplicación*
FD de un fichero. Indica cuántas cadenas similares existen de media en un fichero. También se puede interpretar como el tamaño medio de los agrupamientos de un fichero.

IC *Índice de consistencia*
IC de un agrupamiento. Para un agrupamiento compuesto de n cadenas se define como la suma de las DL entre todas las cadenas que forman el agrupamiento, dividido entre la suma de las longitudes de las n cadenas. Un agrupamiento con una sola cadena posee un IC nulo.

ICF *Índice de consistencia de un fichero*
Indica la media de los IC de todos los agrupamientos existentes en un fichero.

LC *Longitud característica*
Número de letras del alfabeto español y dígitos (37 caracteres distintos en total) que contiene una cadena.

MSC *Medida de similitud combinada*
Distancia entre dos cadenas de caracteres. Se calcula como el mínimo de DL, DIPP, DIPPM y CJ.

NA *Número de agrupamientos*
Es el número total de agrupamientos (correctos e incorrectos) que se han generado.

NAE *Número de agrupamientos exactos*
Número de agrupamientos que coinciden exactamente con los óptimos: contienen las mismas cadenas.

NAO *Número de agrupamientos óptimo*
Es el número de agrupamientos obtenidos de forma manual en un fichero (el número de agrupamientos que se obtiene al agruparlos de la mejor forma posible).

NAR *Número de agrupamientos erróneos*
De los agrupamientos generados, es el número de agrupamientos que contienen alguna cadena errónea (falsos positivos).

NCE *Número de cadenas erróneas*
Número de cadenas que se han incluido en agrupamientos que no son el suyo.

Capítulo 1

Introducción

En este capítulo presentamos el problema de la inconsistencia en una base de datos y sus orígenes, pero sin entrar en detalles técnicos. Además, situamos el problema estudiado en este trabajo dentro de su contexto (*la limpieza, normalización y enriquecimiento de los datos*) y por último, detallamos la estructura del resto del trabajo.

1.1 El problema: motivación

Los sistemas informáticos actuales permiten un acceso rápido y exacto a la información que almacenan en bases de datos. La posibilidad de acceder a la información almacenada electrónicamente de una forma precisa interesa cada vez más, debido a que el volumen de información almacenado cada vez es mayor.

Una de las principales aplicaciones de las bases de datos es la búsqueda de información. Los sistemas de búsqueda tradicionales se basan en el emparejamiento entre el término buscado y los valores almacenados en la base de datos correspondiente.

Si la información contenida en las bases de datos no es consistente[1] (un mismo término puede aparecer de distintas formas porque existan varias denominaciones o porque contenga errores de escritura, por ejemplo), al realizar una búsqueda mediante un valor no se obtendrá toda la información disponible. Por ejemplo, si tenemos una base de datos que almacena información sobre investigadores de universidades (nombre investigador, universidad a la que pertenece) y queremos obtener todos los investigadores de la *"Universidad de Alicante"*, puede ser que la base de datos tenga distintos valores

[1] En su segunda acepción, (Real Academia Española, 1992) define consistencia como "Trabazón, coherencia entre las partículas de una masa o elementos de un conjunto".

que representen a esa universidad: *"Universidad de Alicante"*, *"Universidad Alicante"*, *"U. de Alicante"* o *"Univ. de Alicante"* (con abreviaturas), *"Unibersidad de Alicante"* o *"Univesridad de Alicante"* (con faltas ortográficas o de mecanografiado) o incluso *"Universitat d'Alacant"* (en catalán / valenciano).

El mismo problema puede surgir también cuando se tiene la información deseada diseminada en diversas bases de datos, procedentes o no del mismo propietario, y se desea unificarla en una única base de datos. En ese caso, se debe analizar la información contenida en cada una de ellas, diseñar una estructura de datos única y cargar la información de forma que se eviten duplicados y se depure su contenido. Evidentemente, pueden existir duplicados porque en cada una de las bases de datos se haya empleado criterios distintos a la hora de introducir los datos.

Este problema no sólo afecta a los procesos de búsqueda: cuando se muestre la información (por pantalla o de forma impresa en papel), no es *apropiado* que un mismo término aparezca referenciado de distintas formas.

El problema de la inconsistencia de los valores de un campo en una base de datos suele estar resuelto en los catálogos bibliográficos de las bibliotecas mediante el uso de *diccionarios* o *ficheros de autoridades*[2].

En las bibliotecas, normalmente se tienen controlados mediante ficheros de autoridades los campos autor, materia y serie. Cuando se introduce un valor en uno de los campos anteriores, se verifica si existe ese valor o uno similar en el correspondiente fichero de autoridades. Si existe, se indica el valor correcto que se tiene que emplear; si no existe, se ofrece la posibilidad de introducirlo e incorporarlo en el fichero de autoridades. De este modo se impide que existan distintas variantes de un mismo término, ya que las distintas variantes se agrupan en una sola (la preferida, la más usual o la aceptada oficialmente).

[2]"[...] El principio básico y fundamental del control de autoridades es establecer una única forma autorizada para referirse a los datos bibliográficos (autor, título uniforme, serie, materia) que son potencialmente comunes para más de un documento. Las palabras usadas para expresar esa única forma autorizada o el orden en el que esas palabras aparecen es de consideración secundaria.

Una vez decidida esta forma única, se elabora un registro de autoridad estableciendo referencias desde otras formas de expresión bajo las cuales podrían aparecer esos mismos datos, para remitir a la forma autorizada y se citan las fuentes consultadas que justifican esa decisión. El registro así elaborado pasa a formar parte del fichero de autoridades.

Cuando el nombre, materia o título aparece en un documento con una forma diferente de la admitida, la forma autorizada prevalece sobre la forma que se encuentra en el documento, de modo que éste se incluye en el catálogo bajo el punto de acceso de la forma controlada.

Este sistema puede parecer caro y complicado, pero es la única manera de proporcionar con certeza la recuperación de todos los documentos de una colección que tienen un elemento en común: el mismo autor, materia..."(Soria González, n.d.).

El uso de ficheros de autoridades permite que los usuarios encuentren lo que buscan de una forma rápida, sencilla y precisa. Pero este método no puede garantizar a los usuarios un acceso unificado a los recursos de información existentes en catálogos colectivos y en otras bibliotecas que no se coordinen entre sí, ya que no se puede asegurar que distintas bibliotecas usen los mismos ficheros de autoridades (pueden emplear diferentes estándares de catalogación).

1.2 Contexto del problema

Una posible solución al problema de la inconsistencia de valores es la identificación de la información repetida, lo que permite la eliminación de duplicados. Este proceso se conoce en el mundo informático como *Merge/Purge*[3] (Hernández & Stolfo, 1995), *De-Dupe* o *Dedupe*, *Semantic integration*, *Instance identification* y *Record Linking*. En español se emplea el término *deduplicación*.

En este trabajo estudiamos un caso particular de la deduplicación: sólo trabajamos con un campo de cada registro (*field matching problem*). Un problema más complejo es determinar si dos registros de dos bases de datos distintas contienen información sobre el mismo término. En este caso, hay que tener en cuenta todos los campos del registro. Pero entonces puede aparecer un problema nuevo: las dos bases de datos puede ser que no compartan el mismo esquema[4] (*schema integration/matching problem*) (Batini *et al.*, 1986).

La deduplicación es uno de los pasos de una actividad que se suele denominar *DataCleansing* o *DataCleaning* (Moss, 1998; Adelman & Moss, 1999; Hall, 1998; Quass, 1999; Trillium Software System, n.d.). Esta actividad se puede definir como *la limpieza, normalización y enriquecimiento de los datos.* El fin de esta actividad es garantizar la calidad de los datos almacenados. Cada vez más, las empresas reconocen que la información con calidad es uno de los activos más valiosos que posee una empresa[5] (Huang *et al.*, 1998).

Las técnicas de deduplicación se emplean especialmente en las campañas de marketing directo (envío postal de publicidad) (Database & Consulting, n.d.; Giralpost Publicidad y Marketing Directo, n.d.; Omikron, n.d.). En el

[3]Diferentes bases de datos con registros similares se deben fusionar (*merge*) y los registros duplicados se deben purgar (*purge*).

[4]Por ejemplo, podemos tener dos bases de datos que almacenan información sobre clientes. Una de ellas puede almacenar la información en seis campos (nombre, apellidos, calle, número, población y código postal) y la otra en tres (cliente, dirección y localidad).

[5]La correcta gestión de la información, su valoración y su explotación para que la empresa obtenga el máximo beneficio se conoce como *Total Data Quality Management.*

envío de un mensaje personalizado, el coste del franqueo postal o de entregar en mano dicho mensaje es, quizás, el más elevado en la campaña. Si la base de datos que se emplea para realizar la campaña tiene registrada a una misma persona dos o más veces, se pagarán costes de envío inútiles . Además, el mensaje perderá fuerza, causará una mala imagen en el receptor y producirá un deterioro de la imagen de la empresa, al recibir una persona el mismo mensaje varias veces repetido.

Las empresas suelen emplear las herramientas de deduplicación para evitar el problema GIGO[6]. Cuando una empresa confía en información errónea para enviar sus facturas, sus productos o analizar la evolución de su negocio, seguramente acabará con facturas impagadas, envíos devueltos y oportunidades perdidas o decisiones erróneas.

1.3 Término y valor

A lo largo de este trabajo vamos a emplear los conceptos de término y valor. Por *término* entendemos entidad semántica, una entidad del mundo real, como puede ser una persona, una institución, una organización, etc. Por *valor* entendemos la representación que asignamos a un término al emplear el lenguaje natural. A un mismo término se le pueden asignar distintos valores.

Por ejemplo, la agencia europea que se encarga de los vuelos espaciales, de lanzar satélites, de investigar el espacio, de la observación de la Tierra desde el espacio, etc., es un término, que se representa normalmente con el valor *"Agencia Espacial Europea"* en español, pero que también se puede representar con otros valores: *"European Space Agency"*, *"ESA"*, *"Agencia Europea Espacial"*, *"Agencia Europea del Espacio"*, *"Agencia Espacial de Europa"*, etc.

1.4 Orígenes de la inconsistencia

Básicamente, el problema de la inconsistencia de los valores de una base de datos puede tener tres causas[7]:

1. Si no se tienen controlados los posibles valores que un campo de una base de datos puede tomar, una misma persona (o distintas, si son varias

[6] *garbage in equals garbage out* (basura dentro, basura fuera): axioma famoso en el mundo informático que significa que si entran datos incorrectos en un sistema, la salida también será incorrecta, ya que a partir de algo erróneo no se puede obtener nada correcto.

[7] Evidentemente, en una misma base de datos se pueden combinar las tres causas.

las que pueden insertar información en la base de datos) podrá insertar el mismo término con distintos valores. Por ejemplo, una base de datos que almacena los nombres de los departamentos de una universidad, puede contener distintos valores para un mismo término: *"Departamento de Lenguajes y Sistemas Informáticos"*, *"Depto. de Lenguajes y Sistemas Informáticos"*, *"Dpt. de lenguajes y sistemas informáticos"*, etc.

2. Cuando distintas bases de datos se intentan integrar en una sola (por ejemplo, cuando se intenta construir un almacén de datos(Bort, 1997; Apertus, n.d.; Trillium Software System, n.d.), o cuando se sustituye el sistema informático existente por uno nuevo y se reorganiza la información almacenada), en cada una de las bases de datos puede existir el problema anterior y, además, uno nuevo: aunque se haya garantizado la consistencia de cada una de las bases de datos por separado, se pueden haber establecido diferentes criterios en cada una de ellas. Al integrarlas todas ellas en una, se producirán problemas de consistencia. Por ejemplo, si tenemos tres catálogos bibliográficos y los queremos unir, puede ser que en cada uno de ellos los autores figuren con distintos valores: nombre y apellidos, *"Miguel de Cervantes Saavedra"*, apellidos y nombre, *"Cervantes Saavedra, Miguel de"* o nombre y primer apellido, *"Miguel de Cervantes"*.

3. Otra causa de inconsistencia es la multilingualidad. En una sociedad multilingüe, como la Comunidad Europea, al existir varios idiomas oficiales, es muy común encontrar los nombres oficiales escritos en diferentes idiomas. Por ejemplo, supongamos que consultamos una base de datos que almacena información sobre personal de la Comunidad Europea (por ejemplo, nombre de la persona, institución en la que trabaja), y queremos localizar aquellos que trabajen en el Tribunal de Cuentas Europeo. Fácilmente, si no se ha llevado cuidado al introducir los datos en la base de datos, podemos encontrar diferentes valores para ese término: *"La Cour des comptes"* (francés), *"Court of Auditors"* (inglés), *"O Tribunal de Contas Europeu"* (portugués) o *"La Corte dei conti"* (italiano).

Mientras que resolver los problemas producidos por la primera y tercera causa se puede considerar factible (simplemente hay que revisar los valores de la base de datos uno a uno y corregirlos), cuando nos enfrentamos a la segunda causa puede ser que no podamos corregir las bases de datos porque no tengamos permiso de sus propietarios para hacerlo. Por ejemplo, en Internet se pueden encontrar cada vez más catálogos bibliográficos en línea;

existen metabuscadores que permiten buscar simultáneamente en varios de ellos a partir de una única consulta[8]. Para que la búsqueda sea más precisa, puede ser necesario adaptar el término de la búsqueda a cada catálogo, tal como vemos en la figura 1.1.

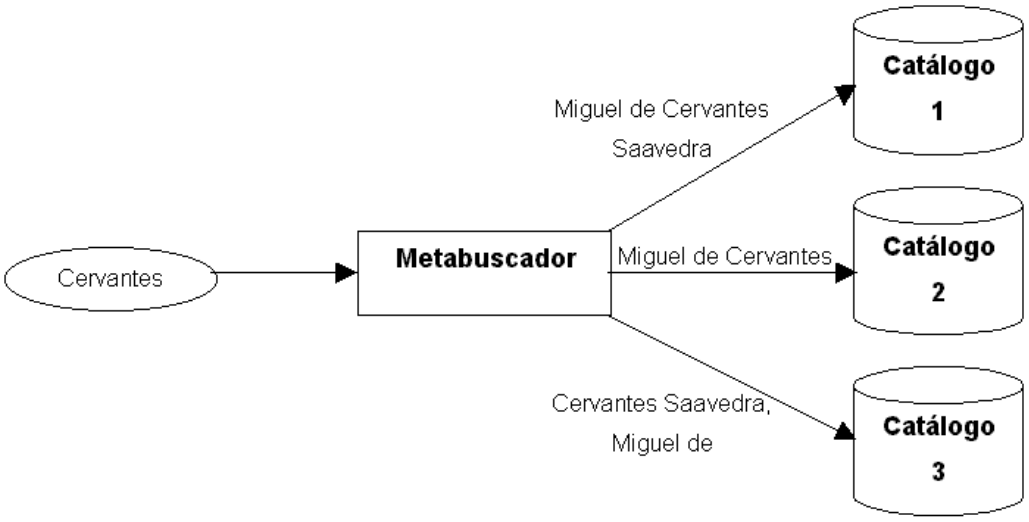

Figura 1.1: Metabuscador que adapta el valor de la búsqueda (Cervantes) a los distintos catálogos bibliográficos

1.5 Estructura de la memoria

El resto de la memoria de investigación se estructura de la siguiente forma:

- En el capítulo 2 presentamos algunos trabajos relacionados con nuestro estudio. En concreto, comentamos una serie de trabajos que se han realizado en la deduplicación de datos. A continuación, destacamos una serie de productos comerciales que tratan el problema de la deduplicación.

- En el capítulo 3 explicamos las principales causas que hemos detectado que producen varios valores a partir de un mismo término (omisión del acento ortográfico, abreviaturas, faltas de ortografía, etc.). En la segunda parte de este capítulo, mostramos una propuesta intuitiva que permite solucionar el problema de la inconsistencia en bases de datos.

[8]De cara al usuario, los catálogos parecen integrados, aunque físicamente no lo están.

- En el capítulo 4 describimos los ficheros empleados en las pruebas de evaluación realizadas. Se describen desde un punto de vista físico (tamaño en *bytes*, número de cadenas que contienen, etc.) y empleando dos métricas: el factor de duplicación y el índice de consistencia.

- En el capítulo 5 presentamos la primera de las técnicas de agrupamiento de valores similares que hemos desarrollado. Esta técnica se evalúa con los ficheros descritos en el capítulo 4. La evaluación se realiza calculando la precisión y el error de los agrupamientos producidos.

- En el capítulo 6 presentamos la segunda de las técnicas. El objetivo de esta técnica es corregir los errores detectados durante la evaluación de la primera técnica.

- En el capítulo 7 concluimos la exposición del trabajo desarrollado con las conclusiones y los trabajos que estamos realizando actualmente.

- Finalmente, la memoria de investigación se acompaña del apéndice A que contiene las direcciones en Internet de las organizaciones y compañías mencionadas (por si se quiere consultar más información), del apéndice B que contiene fragmentos de los cuatro ficheros de prueba empleados y, por último, se incluyen las referencias bibliográficas y un índice.

Capítulo 2

Antecedentes

En este capítulo destacamos algunos de los trabajos que se han realizado en el campo de la deduplicación de datos. A continuación, comentamos una serie de productos comerciales; la mayoría de ellos son soluciones parciales, orientadas a la deduplicación de bases de datos que almacenen nombres de personas y direcciones.

2.1 El estado de la cuestión

Desde hace tiempo, existe un gran interés por estudiar la calidad de la información almacenada en las bases de datos (O'Neill & Vizine-Goetz, 1988; O'Neill & Vizine-Goetz, 1989; Motro & Rakov, 1996; Motro & Rakov, 1998; Huang *et al.*, 1998), a raíz de lo cual se han desarrollado diversos métodos para la reducción de la inconsistencia existente en las bases de datos.

En esta sección destacamos una serie de trabajos que se han realizado en torno al problema de la duplicación de datos en bases de datos. La deduplicación de datos es un proceso necesario para poder realizar tareas más complejas como la minería de datos (*data mining*), la construcción de almacenes de datos (*data marts* y *data warehouses*) o el descubrimiento de conocimiento (*knowledge discovery*), tareas donde es necesario combinar información procedente de fuentes heterogéneas.

- **Aproximación de Dina Bitton y David J. DeWitt**
  Fueron unos de los primeros que estudiaron el problema de la detección y eliminación de registros duplicados en grandes bases de datos. En su trabajo (Bitton & DeWitt, 1983) presentan el método tradicional (y más sencillo) de detección y eliminación de duplicados exactos en una base de datos. Este método se basa en dos fases: ordenar las tuplas de una tabla según un orden (lexicográfico generalmente), seguido de

una lectura secuencial para verificar si dos o más tuplas contiguas son idénticas (la ordenación unirá las tuplas idénticas).

Este método se puede ampliar para permitir la detección de duplicados aproximados si se sustituye la comparación de identidad por una medida de similitud (detectaría como duplicados *"Universidad de Alicante"* y *"Unibersidad de Alicante"*). Sin embargo, su método no permite detectar aquellos duplicados que son diferentes porque exista una transposición de las palabras (no detectaría *"Universidad de Alicante"* y *"Alicante, Universidad de"*, ya que en el proceso de ordenación quedarían muy distantes).

El objetivo principal de su estudio es reducir la complejidad temporal del proceso de deduplicación. Para ello, presentan una modificación del método tradicional que consiste en eliminar de forma gradual los duplicados durante la fase de ordenación y no esperar a que haya finalizado dicha fase (la detección de duplicados se realiza de forma simultánea con el proceso de ordenación). En su trabajo también desarrollan un modelo para evaluar el coste de eliminación de duplicados. El modelo desarrollado no es general, se basa en una serie de suposiciones; la más estricta de las suposiciones impone que todos los registros están duplicados el mismo número de veces.

- **Aproximación de Mauricio Antonio Hernández y Salvatore J. Stolfo**
  El trabajo de (Hernández & Stolfo, 1995; Hernández & Stolfo, 1998) estudia la primera fase (la fusión) del problema *Merge/Purge*. Su aproximación se basa en comparar los registros de la base de datos con aquellos que son más similares (de este modo se reduce el coste temporal, ya que no se comparan todos entre todos). Para ello, primero se ordenan los registros empleando alguna clave extraída a partir de la información que almacenan (la forma de extraer la clave depende del dominio de la información almacenada). Después se comparan los registros próximos [1] (*sorted neighborhood method*), desplazando una ventana de tamaño fijo sobre los registros ordenados de la base de datos: si la ventana tiene tamaño W, el registro i se compara con los registros que van desde $i-W+1$ hasta $i-1$ si $i \geq W$ y con los registros desde 1 hasta $i-1$ en otro caso. El número total de comparaciones que realiza el algoritmo es $O(TW)$, donde T es el número total de registro en la base de datos.

[1] Los registros próximos son aquellos que se encuentran dentro de un intervalo después de la ordenación. El resultado depende de la clave que se haya extraído y del método de ordenación empleado.

Si en el proceso de comparación dos registros resultan equivalentes, se agrupan en un mismo agrupamiento.

La equivalencia de dos registros se basa en el uso de unas reglas dependientes del dominio. Por ejemplo, una regla podría ser "si los nombres de dos personas se parecen[2] y sus direcciones también, entonces se trata de la misma persona". Su solución se basa en el conocimiento que proporcione un experto humano: las reglas se tienen que crear a mano a partir del conocimiento específico de un dominio que posea quien escribe las reglas. Evidentemente, para cada dominio se tienen que establecer un conjunto de reglas distintas. Estas reglas primero se escriben en OPS5[3] y luego se traducen a C a mano, para obtener una mayor velocidad de procesamiento.

En el proceso de evaluación del método que desarrollan Hernández y Stolfo, estudian la influencia del tamaño de la ventana en los resultados. Los resultados que obtienen demuestran que múltiples pasadas de su algoritmo con una ventana pequeña (empleando diferentes claves en cada pasada) logra una mejor precisión y un menor error que una sola pasada con una ventana grande.

En su segundo trabajo (Hernández & Stolfo, 1998), a partir de su primer trabajo desarrollan un algoritmo incremental: una vez realizado un primer agrupamiento sobre una base de datos, cuando se incorporan nuevos registros no es necesario realizar un agrupamiento completo con todos los registros. Cada vez que se incorpora información nueva, se realiza el agrupamiento con lo que ellos llaman *representantes primarios*: un conjunto de registros extraídos de cada agrupamiento que se emplean para representar los agrupamientos.

- **Aproximación de Alvaro E. Monge y Charles P. Elkan**
  En su primer trabajo (Monge & Elkan, 1996), describen y evalúan tres algoritmos que permiten resolver el problema del emparejamiento de campo (*field matching problem*). Este problema consiste en determinar si dos valores (valores sintácticos) hacen referencia a un mismo término (entidad semántica).

  Los tres algoritmos que emplean comparan los dos valores, para saber si

[2] El parecido se establece en función de una distancia más un umbral que establece la distancia máxima que puede separar a dos valores para considerarlos equivalentes.

[3] OPS5 es un lenguaje de reglas de producción desarrollado por un grupo de investigadores de la *Carnegie-Mellon University*. Se usa principalmente en investigaciones sobre inteligencia artificial, sistemas expertos e inteligencia cognitiva. (University of Tennessee Computing Center, 1993)

son semánticamente equivalentes. En la comparación de los dos valores no tienen en cuenta las palabras sin significado semántico (*stop words*). Los tres algoritmos de emparejamiento son:

1. Algoritmo básico de emparejamiento
   El grado de emparejamiento entre dos valores se define como el número de *cadenas atómicas* (secuencia de caracteres demilitada por signos de puntuación o espacios en blanco) que se emparejan dividido entre el número medio de cadenas atómicas que poseen los dos valores. Dos cadenas atómicas se emparejan si son la misma o una es prefijo de la otra (de este modo se pueden detectar algunas abreviaturas).
2. Algoritmo recursivo de emparejamiento
   El grado de emparejamiento de dos valores A y B se calcula como:

$$match(A, B) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} match(A_i, B_j) \tag{2.1}$$

   donde $match(A_i, B_j)$ vale 1 si A_i y B_j son la misma cadena atómica o una es abreviatura de la otra y 0 en caso contrario. El algoritmo incorpora cuatro patrones distintos de detección y expansión de abreviaturas y acrónimos.
3. Algoritmo Smith-Waterman
   Este algoritmo de programación dinámica (Smith & Waterman, 1981) se emplea para comparar ADN y secuencias de proteínas: obtiene el alineamiento óptimo de dichas secuencias. Se basa en emplear una matriz $|\Sigma| \times |\Sigma|$, donde Σ es el alfabeto empleado en las secuencias, en la que se asigna un peso a cada par de símbolos del alfabeto. Además, el comienzo de un fallo en el alineamiento y la continuación de un fallo se penalizan con distintos valores. El algoritmo calcula la serie de cambios con menor coste que convierte una secuencia en otra. Este algoritmo tiene el inconveniente de que es sensible a las transposiciones de palabras.

El rendimiento de los tres algoritmos lo evalúan como un problema de recuperación de información (*information retrieval*): dado un valor, recuperar del conjunto de valores aquellos que sean equivalentes. Para ello, emplean las típicas medidas de evaluación en la recuperación de información: precisión[4] y cobertura[5] (Rijsbergen, 1979; Salton & McGill, 1983). En función de un umbral que se emplea para decidir si dos

[4]Proporción de información recuperada que es relevante.
[5]Proporción de información relevante que se ha recuperado.

valores se emparejan (se consideran equivalentes), se puede variar la precisión y la cobertura obtenida[6].

En su segundo trabajo (Monge & Elkan, 1997), emplean otra vez el algoritmo de Smith-Waterman (Smith & Waterman, 1981). Según exponen en su trabajo, aplican otra vez este algoritmo porque es independiente del domino[7] (se puede aplicar sin modificaciones en una amplia gama de dominios). Para comprobarlo, lo aplican sobre dos conjuntos de datos distintos: sobre un lista de direcciones de correo creada aleatoriamente y sobre un catálogo bibliográfico.

Para detectar los registros duplicados en una base de datos, describen el problema en términos de determinar las componentes conexas de un grafo no dirigido. Cada registro se representa mediante un nodo del grafo. Si un nodo A es alcanzable desde un nodo B, significa que ambos nodos (registros) son duplicados. Cuando se toma un nuevo registro, no se compara con todos los agrupamiento existentes hasta el momento para obtener el más próximo. Sólo se compara con los agrupamientos almacenados en una cola. Esta cola almacena los n últimos agrupamientos creados. En las pruebas realizadas emplean una cola de longitud cuatro; de este modo reducen considerablemente el coste temporal.

Para medir la precisión del método que proponen, emplean el número de agrupamientos detectados como "puros". Un agrupamiento es puro si y sólo sí contiene registros que son realmente duplicados (no es necesario que esté completo, es decir, que contenga todos los registros que son duplicados de un mismo valor).

Las pruebas de evaluación las realizan modificando el número de duplicados por registro y el número de registros, para comprobar el impacto de estos dos valores en los resultados. Los resultados de su método los comparan con el propuesto en (Hernández & Stolfo, 1995; Hernández & Stolfo, 1998). La precisión que se obtiene es similar en ambos casos, pero el método que proponen Monge y Elkan realiza muchas menos comparaciones (del orden de cinco veces menos).

- **Aproximación de James C. French *et al.***
  El objetivo del trabajo de (French *et al.*, 1997a; French *et al.*, 1997b) es automatizar el control de autoridades. Para ello, plantean un método que permite la creación automática de ficheros de autoridades para

[6] Cuando la precisión aumenta, la cobertura disminuye y viceversa.

[7] Para considerar cierta la afirmación, imponen una serie de condiciones. La principal, es que suponen que no existe transposición de datos en las secuencias que se compararan.

catálogos bibliográficos ya existentes que no estén controlados. El método que proponen se orienta a mejorar la calidad de la información almacenada en bibliotecas digitales que se construyen a partir de la integración de información proveniente de diferentes fuentes.

Mediante un algoritmo de agrupamiento, detectan y agrupan aquellos valores que hacen referencia a un mismo término. Una vez agrupados, se puede sustituir todas ellos por un único valor.

Para detectar aquellos valores que hacen referencia a un mismo término, emplean el emparejamiento aproximado de palabras y el coeficiente de Jaccard (Rijsbergen, 1979). El emparejamiento de las palabras lo realizan en base a su distancia de edición.

La evaluación del método que proponen únicamente la realizan sobre un fichero[8] compuesto de 119 cadenas únicas y de 57 agrupamientos. Como veremos más adelante en la sección 4, el 73% de los agrupamientos de este fichero está formado por cadenas únicas. Además de sólo realizar la evaluación sobre este fichero, consideramos que es un fichero pequeño para realizar una evaluación correcta.

En su estudio, aplican el método en la creación de un fichero de autoridades para el campo "afiliación de un autor" en el *Astrophysics Data System*, una base de datos bibliográfica sobre revistas y conferencias de astronomía y astrofísica. El método lo aplican sobre 146.000 registros, obteniendo 20.168 cadenas únicas que se reducen a 8.928 agrupamientos (el resultado obtenido no lo pueden evaluar, ya que no realizan el agrupamiento manual de los registros).

En las referencias de los artículos de French *et al.*, no figura ninguna referencia a trabajos sobre algoritmos de agrupamiento o sobre deduplicación. En especial, destaca la ausencia de referencias a los trabajos de (Hernández & Stolfo, 1995; Monge & Elkan, 1996), ya que son anteriores a la fecha de publicación del artículo.

2.2 Productos comerciales

Existen diversos productos comerciales destinados a mejorar la calidad de la información almacenada en una base de datos. La mayoría de ellos se orientan a mejorar las direcciones postales almacenadas, mediante la eliminación de los duplicados. El principal beneficio que se obtiene, según las empresas

[8] En nuestras pruebas hemos empleado este fichero, al que hemos denominado fichero C.

que venden estos productos, es la reducción de los gastos postales. Por otro lado, también se mejora la imagen de la empresa, ya que una mala gestión de la información que almacena una empresa puede producir desconfianza y rechazo, tanto entre sus clientes como entre su personal.

Estas herramientas se suelen emplear como apoyo a las tareas de gestión de clientes (*customer relationship management*, CRM) y de gestión de recursos humanos (*enterprise resource planning*, ERP), aplicaciones informáticas estratégicas en la empresa moderna en las que se trabaja con muchos datos personales (nombres y direcciones) que son propensos de contener errores.

La mayoría de estos productos presentan un problema: están destinados a un país concreto y analizan las direcciones en base al formato que se emplea en las direcciones de ese país. Además, se basan en el empleo de callejeros para buscar los valores correctos de las direcciones, por lo que no son soluciones generales que se puedan aplicar en cualquier contexto.

A continuación presentamos una relación de algunos de los productos comerciales disponibles. En el apéndice A incluimos las direcciones de las páginas *web* de las empresas fabricantes de estos productos. Se puede encontrar una relación más completa en (Galhardas, 1999), una página web[9] con información sobre herramientas de limpieza e integración de datos.

- **DataCleanser DataBlade Module**
  Se trata de un módulo desarrollado por *Electronic Digital Documents, Inc.* para el sistema gestor de bases de datos *Informix*. Permite detectar registros duplicados que difieren debido a errores o a distintos formatos. Las principales características de este producto son:

  - Ofrece tres distancias para calcular la similitud entre dos valores (distancia de edición, distancia fonética y distancia de mecanografiado), que se pueden ampliar con distancias propias desarrolladas por el usuario.
  - Emplea un módulo basado en reglas, configurable por el usuario.
  - Procesamiento incremental. Permite añadir registros nuevos a los ya analizados y corregidos, sin tener que procesar todo el conjunto de registros de nuevo.

[9] La página muestra las herramientas clasificadas en tres grupos:

- Corrección, estandarización y mejora de nombres y direcciones.
- Deduplicación.
- Análisis de datos (sin limpieza).

  - Se basa en el método del vecino más próximo *sorted neighborhood method*. Este método posee las siguientes fases:
    * Creación de claves
      Calcula una clave para cada registro a partir de los campos más relevantes o porciones de dichos campos.
    * Ordenación de los datos
      Se ordenan los datos en base a la clave previamente calculada.
    * Combinación
      Se desplaza una ventana de tamaño fijo sobre los registros ordenados en el paso anterior, con lo que se limitan las comparaciones para buscar registros duplicados a los registros contenidos en la ventana.

  Este producto se basa en los trabajos de (Hernández & Stolfo, 1995; Hernández & Stolfo, 1998).

- **ETI EXTRACT Toolkit**
  Evolutionary Technologies International, Inc. desarrolla herramientas que facilitan la programación de aplicaciones que recuperan los datos de un sistema de información, los transforman y los cargan en otro sistema. Las operaciones de conversión se especifican en lenguaje natural. Se compone de tres módulos principales:

  - Editores: *Environment*, *Schema*, *Template*, *Grammar* y *Conversion*
    Se emplean para especificar las conversiones y mapeos de los datos. Permiten especificar filtros, integrar distintos orígenes de datos en uno solo, calcular campos nuevos como agregados de los existentes o dividir un campo en distintos campos.
  - *MetaStore*
    Repositorio central de información donde se almacena toda la metainformación necesaria para realizar las conversiones. También almacena un historial de las operaciones realizadas.
  - *Generation Engine*
    Traduce a código las especificaciones en lenguaje natural que indican las operaciones y conversiones que se desean realizar. Una característica interesante de este sistema es que se puede programar de forma centralizada y distribuir de forma automática los programas producidos a aquellos ordenadores que almacenan la información, para que se ejecute de forma remota.

- **Integrity Data Re-engineering Environment**
  Vality Technology ofrece un conjunto de herramientas bajo el nombre *Integrity Data Re-engineering Environment*, destinadas a mejorar la calidad de la información. Sus características principales son: descubre patrones en los datos, revela información oculta en campos de formato libre, saca a la luz prácticas comerciales no documentadas e identifica relaciones entre valores de distintos campos.

  El proceso de limpieza de los datos consta de cuatro fases:

  - Análisis
    Analiza el estado actual de los datos almacenados: tipo de los datos, extracción de entidades/atributos y análisis de frecuencias.
  - Estandarización
    Transformación de los datos a un esquema común.
  - Emparejamiento
    Identifica y agrupa los registros similares usando una técnica de emparejamiento estadístico, que funciona incluso si no existen campos comunes.
  - Supervivencia
    Entre las actividades incluidas en esta fase destacan: completa campos de entrada, resolución de conflictos, enriquecimiento de la información mediante fuentes externas y adaptación de la información a distintos formatos de salida.

- **MatchMaker**
  MatchMaker es una aplicación desarrollada por la empresa *InfoTech Ltd*, destinada a mejorar la calidad de las bases de datos sobre clientes. Limpia y estandariza direcciones, lo que permite a una empresa identificar y dirigirse a sus clientes de una forma más eficaz. *MatchMaker* ofrece las siguientes características:

  - Estandarización de direcciones
    Expande abreviaturas y corrige errores ortográficos comunes.
  - Eliminación de duplicados
    Cuando se estandarizan las direcciones, se pueden detectar los duplicados más fácilmente y eliminarlos.
  - Codificación geográfica
    Localiza direcciones en un mapa. Permite crear mapas que muestran la localizan de los clientes según su dirección.

  - Selecciones complejas
    Permite realizar selecciones basadas en restricciones geográfica.

- **Merge/Purge Plus**
  Este programa de la empresa *Group 1 Software* está destinado exclusivamente a identificar y eliminar registros duplicados que almacenan nombres y direcciones. El algoritmo que emplea (específico para nombres y direcciones) puede detectar duplicados incluso cuando existen variaciones o errores en los nombres o direcciones. Este programa ofrece una serie de opciones que permiten adaptarlo según las necesidades del usuario:

  - Se puede especificar el nivel de emparejamiento para el nombre: débil, medio, fuerte o idéntico.
  - Se puede incluir o excluir el nombre de pila, para así poder detectar los registros que pertenecen a miembros de una misma familia.
  - Se puede incluir o excluir el número de los pisos, para así detectar los registros que pertenecen a un mismo edificio.

- **QuickAddress Software**
  QAS Systems Ltd desarrolla una serie de herramientas destinadas a la recogida, limpieza y mantenimiento de direcciones y nombres. Entre los productos pertenecientes al grupo de herramientas *QuickAddress Software* destacan dos:

  - *QuickAddress Batch*
    Limpia y mejora los nombres y direcciones de los registros almacenados en una base de datos. También ayuda a mantener vigentes los registros a lo largo del tiempo. Verifica y corrige los códigos postales (los añade en las direcciones que no los tengan). Completa las direcciones incorrectas usando el código postal. Estandariza el formato de las direcciones. En una fase interactiva posterior a la automática, se corrigen aquellas direcciones que no se pudieron verificar de forma automática.
  - *QuickAddress Depu*
    Busca registros con direcciones idénticas en una base de datos. Las direcciones duplicadas se pueden eliminar del fichero automáticamente o verificar cada una manualmente.

- **Trillium Software System**
  La compañía *Trillium Software* proporciona software de limpieza de

datos complejos, estandarización y emparejamiento de datos. Se basa en una versión completamente rehecha a partir de los desarrollos de la empresa *Harte-Hanks Data Technologies*, pionera desde 1968 en el desarrollo e implementación de herramientas de limpieza de datos.

Su herramienta se compone de cuatro módulos:

- *Converter*
  Proporciona una serie de herramientas que permiten analizar, limpiar y filtrar los datos. Permite extraer información almacenada en campos de formato libre. Ofrece análisis de frecuencias, conteo de palabras, verificación de dominios, conversiones basadas en patrones, etc.
- *Parser*
  Reformatea, estandariza y verifica los datos. Identifica y transforma múltiples formatos de nombres (personales o comerciales). Analiza y transforma direcciones, tanto de los Estados Unidos como de otros 24 países. Transforma los datos de un formato no estructurado a estructurado.
- *Geocoder*
  Valida, corrige y amplía cualquier tipo de dirección o de información geográfica. Ofrece estandarización de las direcciones postales de Estados Unidos, Canada, Reino Unido, Brasil, Australia y Alemania.
- *Matcher*
  Identifica y empareja nombres y direcciones en diferentes registros.

Capítulo 3

Causas y propuesta intuitiva

En este capítulo presentamos las principales causas que hemos detectado que producen varios valores a partir de un mismo término. En la segunda parte del capítulo, presentamos una propuesta intuitiva que permite solucionar el problema.

3.1 Causas

Tras analizar varias bases de datos con información en español[1], hemos detectado que los distintos valores de un mismo término se pueden deber, principalmente, a una combinación de las siguientes causas:

1. Acento ortográfico
   La omisión o inclusión del acento ortográfico es una de las causas más comunes. Mucha gente, al escribir en mayúsculas, omite los acentos. Además, hay palabras que pueden llevar o no llevar acento.
   - *"Asociación Astronómica"* y *"Asociacion Astronomica"*.
   - *"Telefónica"* y *"Telefonica"*.
   - *"Período electoral"* y *"Periodo electoral"*.
2. Minúsculas y mayúsculas
   El uso de minúsculas y mayúsculas es muy arbitrario.
   - *"Departamento de Lenguajes y Sistemas Informáticos"* y *"Departamento de lenguajes y sistemas informáticos"*.
   - *"Ministerio de Educación y Ciencia"* y *"Ministerio de educación y ciencia"*.

[1] La mayoría de estas causas también se pueden aplicar a otros idiomas como el inglés o el francés.

- *"FORTRAN"* y *"Fortran"*.

3. Abreviaturas, acrónimos y siglas[2]
   Produce una gran diversidad de valores, debido a la ausencia de un estándar o a su mal uso. En las bases de datos con un tamaño de campo fijo, se usan bastante cuando no cabe completamente el valor que se desea introducir.

   - *"Departamento de Derecho Civil"* y *"Dpto. de Derecho Civil"*.
   - *"Instituto de Ingeniería Civil"* e *"Inst. Ing. Civil"*.
   - *"Caja de Ahorros del Mediterráneo"* y *"CAM"*.
   - *"Telefónica I+D, S.A."* y *"TIDSA"*.

4. Orden de las palabras
   Las transposiciones de las palabras de una cadena originan muchos valores diferentes.

   - *"Miguel de Cervantes Saavedra"* y *"Cervantes Saavedra, Miguel de"*.
   - *"Boletín Oficial del Estado - BOE"* y *"BOE - Boletín Oficial del Estado"*.
   - *"Asociación Campo y Playa"* y *"Asociación Playa y Campo"*.

5. Diferentes grafías
   Existen palabras que admiten varias grafías, lo que produce distintos valores para un mismo término.

   - *"Grupo Septiembre"* y *"Grupo Setiembre"*.
   - *"Club Hexágono"* y *"Club Exágono"*.

6. Puntos de puntuación
   Distinto uso o colocación de los signos de puntuación (guiones, comas, puntos, paréntesis, etc.).

[2] Aunque parecen lo mismo, se diferencian entre ellas. Según (Real Academia Española, 1992), una abreviatura es una "representación de las palabras en la escritura con solo varias o una de sus letras, empleando a veces únicamente mayúsculas, y poniendo punto después de la parte escrita de cada vocablo" (*Ud., Sr., D.*), un acrónimo es una "palabra formada por las iniciales, y a veces, por más letras, de otras palabras"(*RENFE, BANESTO*), y una sigla es la "letra inicial que se emplea como abreviatura de una palabra o el rótulo de denominación que se forma con varias siglas" (*S. D. M., INRI*).

- *"Laboratorio Multimedia (mmlab)"* y *"Laboratorio Multimedia - mmlab"*.
- *"Alicante / España"* y *"Alicante (España)"*.

7. Errores de escritura
   Debidos a varias causas: faltas ortográficas (exceptuando acentos), equivalencias fonéticas de algunos sonidos (*b* y *d*, *p* y *t*) y errores de mecanografiado (ausencia de caracteres, intercambio de caracteres adyacentes, etc.).

   - *"Xerox"* y *"Serox"*.
   - *"Gabinete de imagen"* y *"Gavinete de imagen"*.
   - *"Actitud académica"* y *"Aptitud académica"*.
   - *"Bill Meyer"* y *"Bill Meier"*.

8. Empleo de números
   En un término, los números pueden producir distintos valores ya que se pueden escribir con letras o con dígitos.

   - *"Uni2"* y *"UniDos"*.
   - *"Registro de la propiedad número 2"* y *"Registro de la propiedad número dos"*.
   - *"Antena 3"* y *"Antena Tres"*.

9. Palabras extra
   El añadido o ausencia de palabras.

   - *"Real Club de Regatas"* y *"Club de Regatas"*.
   - *"Sociedad Española de Oftalmología"* y *"Sociedad de Oftalmología"*.
   - *"Organización de Consumidores"* y *"Organización de Consumidores (OCU)"*.

10. Distintas denominaciones
    Un mismo término puede tener distintas denominaciones, que pueden ser correctas (estar aceptadas) o incorrectas.

    - *"Ortega y Gasset"* y *"Ortega Gasset"*.
    - *"Unidad de Registro Sismológico"* y *"Unidad de Registro Sísmico"*.
    - *"Canal +"* y *"Canal Plus"*.
    - *"José Giménez"* y *"Pepe Jiménez"*.

11. Distintos idiomas
    Uso de distintos idiomas por existir más de una lengua oficial o por emplear la denominación de un término en su lengua original.

    - *"Universidad de Alicante"* y *"Universitat d'Alacant"*.
    - *"Orense"* y *"Ourense"*.
    - *"Nueva York"* y *"New York"*.

Damerau (Damerau, 1964) señala que el 80% de los errores de escritura se debe a uno de estos cuatro tipos:

- inserción accidental de un carácter extraño,
- omisión accidental de un carácter,
- substitución accidental de un carácter por otro, y
- transposición accidental de dos caracteres adyacentes.

Este resultado ha sido confirmado por los estudios de (Morgan, 1970) y (Shaffer & Hardwick, 1968). Un estudio posterior de Pollock y Zamora (Pollock & Zamora, 1984a; Pollock & Zamora, 1984b) sugiere que cerca del 90% de los errores de escritura se debe a un único error, aunque no necesariamente a los identificados por Damerau.

3.2 Propuesta intuitiva del método

El método que planteamos permite solucionar todas las causas detalladas en la sección 3.1 excepto las tres últimas, ya que las diferencias entre dos valores pueden ser muy grandes (depende de cómo de diferentes sean las dos cadenas que se intenten agrupar). Por ejemplo, nuestra solución permite agrupar variantes de un mismo término en distintos idiomas cuando los idiomas son muy similares (tienen un origen común y comparten una grafía similar), como el castellano y el valenciano, pero no cuando son muy dispares, como el castellano y el alemán. El método que proponemos se puede dividir en siete fases:

1. Concatenación
   Dadas una o más bases de datos, los contenidos de los campos inconsistentes entre sí se concatenan todos ellos en un solo fichero.

2. Acondicionamiento
   Según el dominio en el que se trabaje, puede ser necesario acondicionar (limpiar) las cadenas antes de aplicarles el algoritmo: eliminar espacios en blanco al principio o al final, eliminar signos de puntuación (punto final), etc.

3. Lectura
   Se repite para cada una de las cadenas contenidas en el fichero de entrada el siguiente proceso:

   - Se lee una cadena.
   - Se expanden las abreviaturas, acrónimos y siglas de la cadena: se sustituyen por la cadena que representan. Ejemplo: *"Dpto. de Lenguajes y Sist. Informáticos"* se convierte en *"Departamento de Lenguajes y Sistemas Informáticos"*.
   - Se eliminan los acentos de la cadena. Ejemplo: *Á* y *À* pasan a ser *A*, *á* y *à* pasan a ser *a*.
   - Se pasa a minúsculas la cadena. Ejemplo: *"Universidad de Alicante"* se transforma en *"universidad de alicante"*.
   - Se almacena la cadena. Si la cadena ya se había encontrado (por tanto, ya está almacenada), se incrementa su frecuencia de aparición en una unidad.

4. Ordenación
   Se ordenan las cadenas en función de su frecuencia de aparición en el fichero, de mayor a menor frecuencia (posteriormente, el algoritmo de agrupamiento va eligiendo las cadenas de mayor a menor frecuencia de aparición).

5. Agrupamiento (*clustering*[3])
   Se elige la cadena más frecuente y empleando una distancia se compara con el resto de cadenas para encontrar las similares[4] a la cadena elegida (puede ser que no se encuentre ninguna), con las que se forma un agrupamiento. Para cada agrupamiento se elige un representante (*representante canónico*). Se repite el proceso sucesivamente con las cadenas que van quedando hasta que se hayan agrupado todas.

[3]El término *clustering* se toma prestado de la terminología empleada en el reconocimiento estadístico de patrones (*statistical pattern recognition*).

[4]El método se basa en la suposición de que los valores de un mismo término serán más similares entre sí que con los valores de otros términos.

6. Revisión
   Se verifican los agrupamientos producidos, localizando los errores que se hayan producido (cadenas agrupadas incorrectamente o cadenas que no se hayan agrupado).

7. Actualización
   Se actualizan las bases de datos de donde provienen las cadenas: se sustituyen[5] las cadenas de un grupo por su representante canónico o se crea un nivel intermedio que realice la sustitución del término cuando se realice una búsqueda, adaptándolo a las distintas bases de datos, tal como se muestra en la figura 1.1.

3.3 Un ejemplo

En la tabla 3.1 mostramos dos tablas provenientes de dos bases de datos distintas. Ambas tablas almacenan la misma información: nombre de alumno y universidad a la que pertenece. La tabla A podemos observar que almacena datos inconsistentes: con faltas de ortografía (*"Unibersidad de Alicante"*) y escritos en varios idiomas (español, valenciano e inglés). La tabla B no almacena información inconsistente.

Si intentamos integrar las tablas A y B en un repositorio común, los valores del campo **Universidad** serán inconsistentes entre sí. Incluso si cada una de las tablas fuera consistente por separado, pueden aparecer inconsistencias al fusionarlas. Por ejemplo, la tabla A podría emplear *"Universitat d'Alacant"* en todos los casos y la tabla B *"Universidad de Alicante"*.

En la figura 3.1 mostramos los agrupamientos que se obtienen tras aplicar el método que proponemos. En **negrita** se ha señalado el representante de cada agrupamiento. Se ha elegido como representante el valor más frecuente de cada agrupamiento, porque se considera que el valor más repetido es el más probable de ser el correcto.

En la tabla 3.2 mostramos las tablas A y B integradas una vez que se ha corregido el campo **Universidad**. Los valores *"Unibersidad de Alicante"*, *"Universitat d'Alacant"* y *"Alicante University"* se han sustituido por su representante *"Universidad de Alicante"*. Los valores *"Universidad Jaume I"* y

[5]Evidentemente, cuando se vayan a sustituir las cadenas por sus representantes canónicos, las cadenas empleadas en los agrupamientos no coincidirán con las existentes en las bases de datos (porque se han pasado a mínúsculas, se han eliminado los acentos y signos de puntuación, etc.). Por tanto, en el proceso de agrupamiento habrá que conservar con cada cadena un valor que indique a que registros representa (por ejemplo, el número de registro o la clave primaria de cada registro).

Tabla A	
Nombre	**Universidad**
Sergio Luján	Universidad de Alicante
Marisa Lapita	Universitat d'Alacant
Jose Benatar	Universitat Jaume I
Chati Mona	Unibersidad de Alicante
M.Z. Fornieles	Universidad Jaume I
John Doe	Alicante University

Tabla B	
Nombre	**Universidad**
Marisa Zayas	Universidad de Alicante
Juan Pizarro	Universidad Jaime I
Blas García	Universidad de Alicante
Gloria Mora	Universidad Jaime I
Mario Luján	Universidad Jaime I
Ale Segoviano	Universidad de Alicante

Tabla 3.1: Dos tablas con valores inconsistentes en el campo Universidad

"Universitat Jaume I" se han sustituido por su representante *"Universidad Jaime I"*.

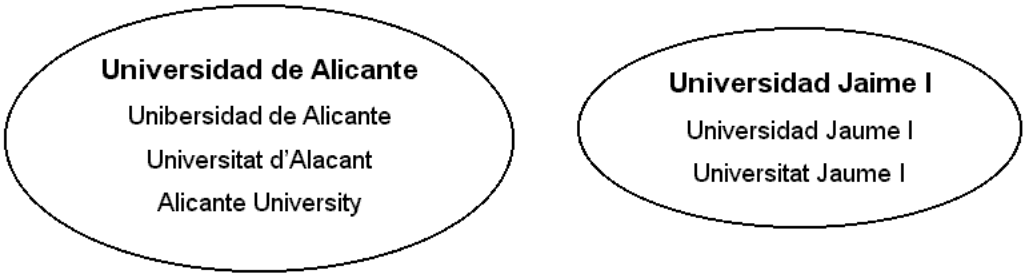

Figura 3.1: Agrupamientos obtenidos tras aplicar el método

Tabla A + B	
Nombre	**Universidad**
Sergio Luján	Universidad de Alicante
Marisa Lapita	Universidad de Alicante
Jose Benatar	Universidad Jaime I
Chati Mona	Universidad de Alicante
M.Z. Fornieles	Universidad Jaime I
John Doe	Universidad de Alicante
Marisa Zayas	Universidad de Alicante
Juan Pizarro	Universidad Jaime I
Blas García	Universidad de Alicante
Gloria Mora	Universidad Jaime I
Mario Luján	Universidad Jaime I
Ale Segoviano	Universidad de Alicante

Tabla 3.2: Tablas A y B integradas con el campo Universidad corregido

Capítulo 4

Ficheros de prueba

En las pruebas realizadas para evaluar los dos métodos que proponemos en este trabajo, se han empleado cuatro ficheros de prueba. En este capítulo describimos los ficheros empleados. Primero realizamos una descripción física de los ficheros: tamaño en *bytes*, número de cadenas, número de agrupamientos óptimo, etc. Después, describimos los ficheros utilizando dos métricas. Para ello, definimos los conceptos de factor de duplicación y de índice de consistencia de un fichero, que nos permiten obtener un valor que representa la complejidad de los agrupamientos de un fichero.

4.1 Descripción de los ficheros

A lo largo de las pruebas se han empleado cuatro ficheros de prueba (A, B, C y D). Los ficheros A, B y D almacenan información en español, mientras que el C la almacena en inglés. En el apéndice B se han incluido algunos fragmentos de los cuatro ficheros, para proporcionar una imagen de su contenido.

El fichero A se obtuvo a partir de una lista de los servidores web de la *Universidad de Alicante* (Universidad de Alicante, 2000). Esta lista incluye departamentos, institutos, grupos de investigación, secretariados, etc. Para cada entrada de la lista, se buscaron otras formas de denominación a través de la web de la Universidad de Alicante. Además, se introdujeron abreviaturas y se tradujeron del español al valenciano y viceversa.

El fichero B se obtuvo a partir del fichero A mediante una serie de transformaciones realizadas por un programa (conjunto de datos sintéticos). Cada cadena del fichero A, se duplicó un número aleatorio de veces. Las cadenas duplicadas tenían una cierta probabilidad de ser transformas para que tuviesen errores. Las transformaciones eran aleatorias y consistían en transponer dos caracteres adyacentes, sustituir un carácter por otro elegido de forma

aleatorio, sustituir un carácter por el anterior o por el posterior e introducir un carácter adicional en la cadena. El objetivo de las transformaciones era introducir errores ortográficos o tipográficos.

El fichero C se obtuvo del directorio público de ftp de James C. French en el servidor de la *Universidad de Virginia*[1]. Este fichero se empleó en los trabajos de (French *et al.*, 1997a; French *et al.*, 1997b), que hemos comentado en la sección 2.1. Este fichero almacena el nombre de instituciones y otras entidades localizadas en el estado de *Virginia* (EE.UU.), junto con algunas más de *West Virginia* (incluidas por el autor del fichero porque son difíciles de diferenciar de las de Virginia y suponen todo un reto).

El fichero D se obtuvo a partir de una relación de entidades conectadas a *RedIRIS* (RedIRIS, 2000). La relación de entidades incluye universidades, hospitales, centros de investigación y otras instituciones. El fichero se entregó a tres personas junto con una serie de instrucciones: para cada una de las entidades, debían de crear duplicados, introduciendo para ello faltas de ortografía, abreviaturas, transposiciones de palabras, denominaciones similares, etc. Luego, los tres ficheros se fusionaron en uno solo, que es el que se ha empleado en las pruebas.

4.1.1 Características de los ficheros

En la tabla 4.1 se resumen las características principales de los ficheros. Los ficheros contienen un valor (una cadena) por línea. El tamaño de los ficheros en *bytes* se refiere al tamaño que ocupan en *Linux*, donde el salto de línea se representa por un único carácter.

La columna *Número de agrupamientos óptimo* (NAO) indica el número de agrupamientos que se obtiene al realizar el agrupamiento *a mano*[2] (el mejor agrupamiento posible). Este valor se emplea en la evaluación de los dos métodos de agrupamiento que proponemos: se comparan los agrupamientos generados de forma automática con los obtenidos de forma manual.

Las cuatro últimas columnas de la tabla indican las cadenas que posee el fichero, las cadenas no duplicadas[3] que posee sin expandir las abreviaturas (SIN), las cadenas no duplicadas que posee cuando se expanden las abrevia-

[1]Este fichero se encuentra disponible a través de ftp anónimo en la dirección `ftp://ftp.cs.virginia.edu/pub/french/ads/va.benchmark.tar`.

[2]Como los agrupamientos se tienen que realizar a mano para poder evaluar los métodos propuestos, no se pueden emplear ficheros de prueba muy grandes (con muchos agrupamientos y muchas cadenas), ya que llevaría mucho tiempo obtener los agrupamientos.

[3]Para obtener las cadenas no duplicadas, se pasan a minúsculas, se eliminan los acentos y los espacios en blanco al principio y al final de la cadena. Es lo que se describe en las secciones 5 y 6 como *procesamiento previo de las cadenas*.

turas (CON[4]) y la tasa de reducción que se produce en el número de cadenas al expandir las abreviaturas (al expandir las abreviaturas se reduce el número de cadenas, ya que se producen duplicados que son eliminados). Las pruebas se han realizado sin y con expansión de abreviaturas, para poder evaluar el impacto de las abreviaturas en los resultados.

Fichero	**Tamaño (Bytes)**	**NAO**	**Cadenas fichero**	**Cadenas SIN**	**Cadenas CON**	**Reducción (%)**
A	10.399	92	234	234	145	38,0
B	1.717.706	92	37.599	1.212	1.117	7,8
C	108.608	57	2.206	119	118	0,8
D	24.364	226	596	584	505	13,5

Tabla 4.1: Descripción de ficheros

Como vemos en la tabla 4.1, los ficheros A y B tienen el mismo NAO, ya que el fichero B se obtuvo aplicando una serie de transformaciones a las cadenas del fichero A.

4.1.2 Tamaño de los agrupamientos

En este apartado describimos cada fichero desde el punto de vista del tamaño de los agrupamientos que contiene cada uno. En las tablas 4.2, 4.3, 4.4 y 4.5 mostramos el número de agrupamientos que posee cada fichero en función del número de cadenas (tamaño) que poseen los agrupamientos. El tamaño de los agrupamientos va desde *1* (agrupamientos formados por una única cadena) hasta *10+* (agrupamientos formados por 10 o más cadenas).

En cada tabla, se muestra como influye la expansión de las abreviaturas (SIN y CON) en el tamaño de los agrupamientos.

En la tabla 4.2 podemos ver que la expansión de abreviaturas aumenta considerablemente el número de agrupamientos de tamaño 1, pasando de un 20% a un 60% aproximadamente. Los agrupamientos de tamaño 3 disminuyen considerablemente, pasando de un 46% a un 6%. Estos valores confirman la reducción de un 38% del número de cadenas que mostramos en la tabla 4.1.

En la tabla 4.3 podemos observar que la mayoría de los agrupamientos del fichero B poseen 10 o más cadenas (69,6%). La expansión de abreviaturas afecta muy poco al tamaño de los agrupamientos, ya que la reducción del número de cadenas es pequeña (un 7,8%, tal como vemos en la tabla 4.1).

[4]A partir de ahora, utilizaremos SIN para indicar que no se ha empleado la expansión de abreviaturas y CON para cuando sí que se emplea.

Número cadenas	**SIN**		**CON**	
	Total	%	**Total**	%
1	18	19,6	54	58,7
2	20	21,7	28	30,4
3	43	46,7	6	6,5
4	8	8,7	3	3,3
5	3	3,3	1	1,1
6	0	0,0	0	0,0
7	0	0,0	0	0,0
8	0	0,0	0	0,0
9	0	0,0	0	0,0
10+	0	0,0	0	0,0
Total	92	100	92	100

Tabla 4.2: Tamaño de los agrupamientos fichero A

Número cadenas	**SIN**		**CON**	
	Total	%	**Total**	%
1	0	0,0	0	0,0
2	0	0,0	0	0,0
3	2	2,2	3	3,3
4	0	0,0	0	0,0
5	4	4,3	3	3,3
6	2	2,2	7	7,6
7	5	5,4	9	9,8
8	6	6,5	6	6,5
9	9	9,8	6	6,5
10+	64	69,6	58	63,0
Total	92	100	92	100

Tabla 4.3: Tamaño de los agrupamientos fichero B

En la tabla 4.4 vemos los tamaños de los agrupamientos para el fichero C. Como podemos apreciar, la expansión de abreviaturas parece que no afecta para nada al tamaño de los grupos. Si consultamos la tabla 4.1, observaremos que la expansión de abreviaturas sólo reduce una cadena. Esta reducción se produce en un agrupamiento que contiene 10 o más cadenas, por lo que no se refleja en la tabla 4.4.

Número	**SIN**		**CON**	
cadenas	**Total**	**%**	**Total**	**%**
1	42	73,7	42	73,7
2	6	10,6	6	10,6
3	3	5,3	3	5,3
4	1	1,7	1	1,7
5	1	1,7	1	1,7
6	0	0,0	0	0,0
7	0	0,0	0	0,0
8	0	0,0	0	0,0
9	2	3,5	2	3,5
10+	2	3,5	2	3,5
Total	57	100	57	100

Tabla 4.4: Tamaño de los agrupamientos fichero C

En la tabla 4.5 vemos los tamaños de los agrupamientos del fichero D. La expansión de abreviaturas produce una reducción del número de agrupamientos de tamaños 3 y 4, mientras que los agrupamientos de tamaño 1 y 2 aumenta.

Al expandir las abreviaturas, se eliminan cadenas de los agrupamientos porque aparecen duplicados. Esto conlleva que algunos agrupamientos pasen a tener un tamaño inferior, tal como podemos comprobar en las tablas 4.2, 4.3, 4.4 y 4.5 si comparamos las columnas SIN y CON.

Si observamos las tablas de los tamaños de los agrupamientos podemos deducir que el fichero C será el que mayor precisión obtendrá en el proceso de agrupamiento, ya que es el que más agrupamientos de tamaño 1 posee (73,7%). Por otro lado, el fichero B será el que menor precisión obtendrá, ya que posee más de un 60% de agrupamientos con un tamaño igual o superior a 10. Cuantas menos cadenas posea un agrupamiento, más fácil será agrupar sus cadenas en general.

Para estar seguros de estas suposiciones, necesitamos conocer la complejidad de los agrupamientos, ya que no es lo mismo un agrupamiento formado por cadenas muy similares que por cadenas muy dispares. El factor de du-

Número cadenas	SIN		CON	
	Total	%	**Total**	%
1	56	24,8	69	30,5
2	49	21,7	76	33,6
3	57	25,2	42	18,6
4	61	27,0	37	16,4
5	3	1,3	2	0,9
6	0	0,0	0	0,0
7	0	0,0	0	0,0
8	0	0,0	0	0,0
9	0	0,0	0	0,0
10+	0	0,0	0	0,0
Total	226	100	226	100

Tabla 4.5: Tamaño de los agrupamientos fichero D

plicación y el índice de consistencia de un fichero nos permiten obtener una representación de la complejidad de los agrupamientos.

4.2 Factor de duplicación

La cantidad de duplicación que contiene un fichero la hemos medido empleado el *Factor de duplicación* (FD) (Bitton & DeWitt, 1983), que indica cuántas cadenas similares existen de media en el fichero.

Un FD de valor 1 significa que los agrupamientos están formados por una única cadena: en el fichero no existe ninguna cadena similar a ella (no hay duplicados de esa cadena). Un FD de valor 2 significa que, de media, los agrupamientos están formados por dos cadenas.

En la tabla 4.6 mostramos el FD de los cuatro ficheros sin y con expansión de abreviaturas. Como el FD se calcula como la media del número de cadenas de cada agrupamiento, también mostramos la desviación estándar[5] correspondiente.

Como vemos, el fichero B tiene un FD muy superior al resto de los ficheros. Esto concuerda con los valores de la tabla 4.3, donde observamos que más del 60% de los agrupamientos del fichero B posee 10 o más cadenas.

La tasa de reducción del FD es similar a la tasa de reducción del número de cadenas que mostramos en la tabla 4.1, ya que ambas están directamente

[5]La desviación estándar se ha calculado empleando la fórmula $\sigma = \sqrt{\sum_{i=1}^{n} \frac{x_i^2}{n} - \bar{x}^2}$.

relacionadas.

Fichero	FD SIN	Desviación estándar	FD CON	Desviación estándar	Reducción (%)
A	2,54	1,00	1,58	0,84	37,8
B	13,17	5,39	12,14	5,23	7,8
C	2,09	2,88	2,07	2,80	0,9
D	2,58	1,16	2,23	1,08	13,6

Tabla 4.6: Factor de duplicación de los ficheros de prueba

4.3 Índice de consistencia

El agrupamiento de las cadenas será más difícil cuanto más dispares sean las cadenas que forman un agrupamiento. Es de esperar que los ficheros más difíciles obtengan peores resultados que los más sencillos.

Para poder comparar la complejidad de los ficheros, hemos desarrollado una métrica que nos permite evaluar la complejidad de un agrupamiento: cuanto mayor es su valor, más dispares son las cadenas que forman el agrupamiento (un valor nulo indica que el agrupamiento contiene una única cadena). El *Índice de consistencia* (IC) de un agrupamiento compuesto de n cadenas se define como la suma de las distancias de Levenshtein (la definición de esta distancia se encuentra en la sección 5.2.1, página 42) entre todas las cadenas que forman el agrupamiento, dividido entre la suma de las longitudes de todas las cadenas del agrupamiento (ecuación 4.1).

$$\mathrm{IC} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{DL}(x_i, x_j)}{\sum_{i=1}^{n} |x_i|} \tag{4.1}$$

El *Índice de consistencia de un fichero* (ICF) que contiene m agrupamientos, se define entonces como la media de los índices de consistencia de todos los agrupamientos existentes en el fichero (ecuación 4.2).

$$\mathrm{ICF} = \frac{\sum_{i=1}^{m} \mathrm{IC}_i}{m} \tag{4.2}$$

En la tabla 4.7 mostramos los índices de consistencia de los ficheros[6] de

[6]Como el ICF es una media, incluimos también la desviación estándar correspondiente.

prueba sin y con expansión de abreviaturas. Como se puede observar, los agrupamientos del fichero B son los más complejos, ya que su ICF es el mayor (1,72 sin expansión de abreviaturas y 1,11 con expansión).

En todos los casos, el ICF se reduce al expandir las abreviaturas, ya que las discrepancias entre las cadenas de un mismo agrupamiento tienden a disminuir. La reducción del ICF al expandir las abreviaturas está relacionado con la reducción del número de cadenas, tal como podemos observar en las tablas 4.1 y 4.7: si ordenamos los ficheros de menor a mayor porcentaje de reducción, se obtiene el mismo orden en ambos casos (C, B, D y A).

Fichero	**ICF SIN**	**Desviación estándar**	**ICF CON**	**Desviación estándar**	**Reducción (%)**
A	0,31	0,29	0,12	0,26	61,2
B	1,72	1,26	1,11	1,14	35,4
C	0,33	1,18	0,31	1,13	6,0
D	0,53	0,44	0,27	0,27	49,0

Tabla 4.7: Índice de consistencia de los ficheros de prueba

En las figuras 4.1 y 4.2 mostramos la distribución del IC en los cuatro ficheros de prueba: porcentaje de agrupamientos que poseen un IC dado. El IC se representa en el eje de abcisas de forma agrupada (en intervalos): *0,1* representa un IC menor o igual que 0,1; *0,2* representa un IC mayor que 0,1 y menor o igual que 0,2; así hasta llegar a *3+* que representa un IC mayor que 3. En la figura 4.1 se muestra sin expansión de abreviaturas y en la figura 4.2 con expansión de abreviaturas.

En ambas figuras hemos cortado el eje de ordenadas por el 40%, aunque hay valores mayores, con el fin de obtener una mejor representación. En la figura 4.1, el máximo valor se alcanza en el fichero C, para un IC de 0,1 con un 77,2%. En la figura 4.2, los máximos valores se alcanzan en los ficheros A y C para un IC de 0,1 con un 67,4% y un 79% respectivamente.

Si comparamos las dos figuras 4.1 y 4.2, observamos que la distribución del IC en la segunda figura se desplaza hacia la izquierda, lo que significa que disminuye el porcentaje de agrupamientos con un IC alto y aumenta el porcentaje de los agrupamientos con un IC bajo: al expandir las abreviaturas, la complejidad de los agrupamientos disminuye (las cadenas de un mismo agrupamiento se parecen más entre sí). Por tanto, la expansión de abreviaturas es un paso importante en todo el proceso, ya que disminuye el tamaño de los agrupamientos y su complejidad.

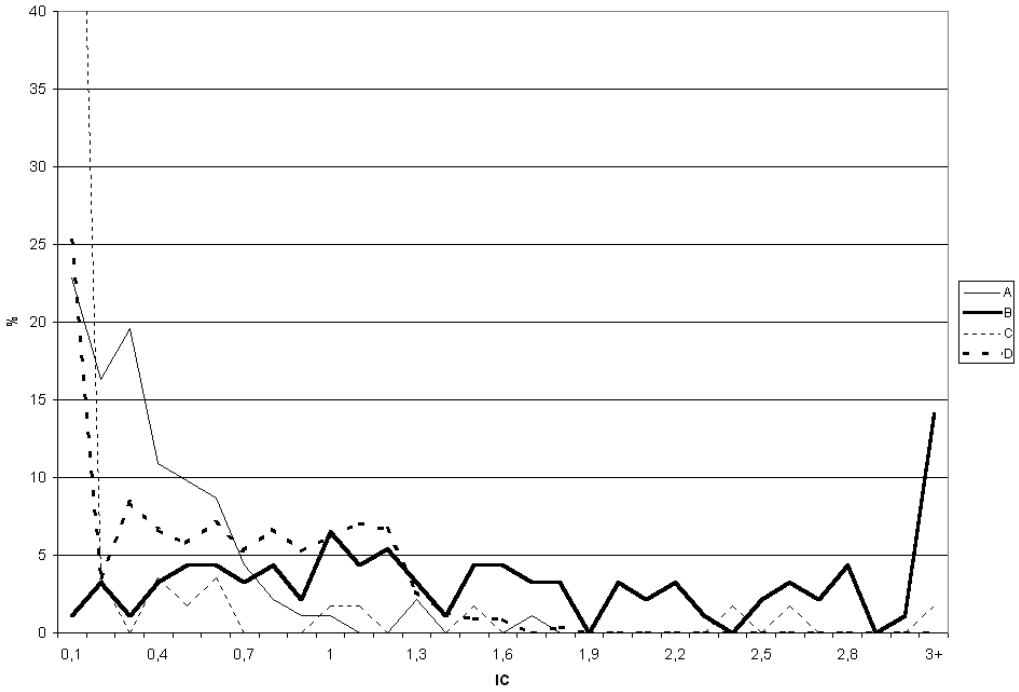

Figura 4.1: Distribución del IC en los ficheros de prueba sin expansión de abreviaturas (SIN)

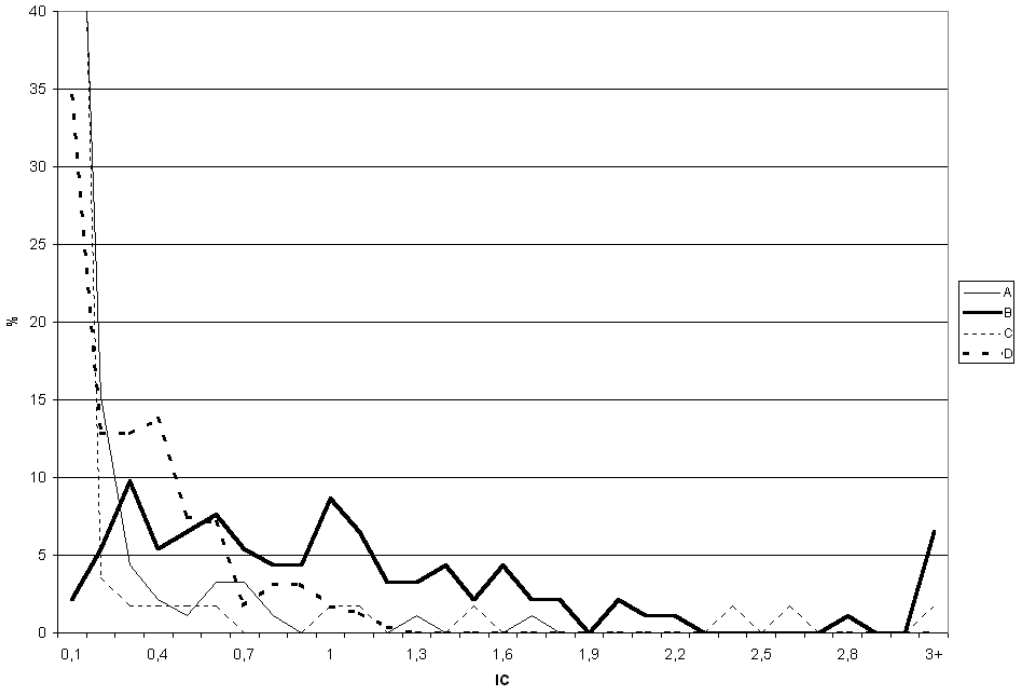

Figura 4.2: Distribución del IC en los ficheros de prueba con expansión de abreviaturas (CON)

Capítulo 5

Agrupamiento de valores similares: aproximación 1

En este capítulo presentamos la primera solución que hemos desarrollado (Luján-Mora & Palomar, 2000). El objetivo de esta solución es optimizar el tiempo de procesamiento. Para lograrlo, veremos como al aplicar una serie de medidas de similitud aproximadas pero sencillas de calcular, se evitará el tener que calcular las más precisas y costosas desde un punto de vista computacional. Además, cuando se realiza la búsqueda del agrupamiento al que pertenece una cadena, no se busca el más óptimo sino el primero que cumple una condición. Al final del capítulo incluimos las medidas de evaluación que empleamos para evaluar el comportamiento de nuestro método, junto con los resultados obtenidos.

5.1 Procesamiento previo de las cadenas

Antes de aplicar el algoritmo de agrupamiento, las cadenas sufren un procesamiento previo con el fin de obtener mejores resultados en el proceso de agrupamiento. El objetivo de este procesamiento es anular las tres primeras causas que originan distintos valores a partir de un mismo término: acentos, minúsculas/mayúsculas y abreviaturas. Para lograrlo, se eliminan los acentos de las cadenas, se pasan a minúsculas y se expanden las abreviaturas (las abreviaturas y siglas dependen del dominio en que se esté trabajando: una misma abreviatura puede expandirse a distintas cadenas en distintos dominios[1]).

A lo largo de la exposición se va a usar como ejemplo de aplicación del método las siete cadenas de la tabla 5.1, donde aparece cada cadena con su

[1]Por ejemplo, *"Ind."* puede expandirse a *"Industrial"* o a *"Independiente"* y *"Pol."* a *"Polígono"* o *"Polideportivo"*.

longitud y su longitud característica[2]. Estas cadenas se pueden agrupar en tres grupos: $\{c_1, c_2, c_3, c_4\}$, $\{c_5, c_6\}$ y $\{c_7\}$, donde cada grupo representa una institución distinta.

	Cadena	**Longitud**	**Longitud característica**
c_1	Universidad de Alicante	23	21
c_2	Universitat d'Alacant	21	19
c_3	University of Alicante	22	20
c_4	Alicante University	19	18
c_5	Ciencias, Universidad de Valencia	33	29
c_6	Universitat de València, Ciències	33	29
c_7	Universidad Politécnica de Valencia	35	32

Tabla 5.1: Cadenas de ejemplo

En la tabla 5.2 vemos las cadenas una vez que han pasado por el procesamiento previo. En este ejemplo, no ha habido expansión de abreviaturas.

Cadena	**Cadena procesada**
c_1	universidad de alicante
c_2	universitat d'alacant
c_3	university of alicante
c_4	alicante university
c_5	ciencias, universidad de valencia
c_6	universitat de valencia, ciencies
c_7	universidad politecnica de valencia

Tabla 5.2: Cadenas de ejemplo una vez procesadas

5.2 Similitud entre dos cadenas

El núcleo del algoritmo de agrupamiento se basa en el cálculo de la similitud entre dos cadenas. Necesitamos una medida que permita agrupar las cadenas de la tabla 5.1 en los grupos $\{c_1, c_2, c_3, c_4\}$, $\{c_5, c_6\}$ y $\{c_7\}$.

Existen diversos métodos para calcular la similitud entre dos cadenas de caracteres. Estos métodos se suelen emplear en la detección y corrección de

[2]La longitud característica de una cadena es el número de caracteres del alfabeto español y dígitos que posee (sección 5.2.2).

errores de escritura (Pollock & Zamora, 1984a; Pollock & Zamora, 1984b; Kukich, 1992). Algunos de estos métodos se basan en la inversión de los errores: suponen las posibles causas que pueden conducir a diferencias entre dos cadenas de caracteres. Las cuatro causas más importantes son las ya comentadas en la sección 3.1 y que expone Damerau (Damerau, 1964): borrado de un carácter, inserción de un carácter, sustitución de un carácter o intercambio de dos caracteres adyacentes.

Con un modelo tan sencillo como el anterior, no se pueden contemplar casos más complejos, como los mostrados en la sección 3.1 (distinto orden de palabras, distintas denominaciones, etc.). Un modelo más general se puede lograr con el uso de los *n-gramas* (Freund & Willett, 1982; Angell *et al.*, 1983; Rapp, 1997), una de las técnicas más comunes en el emparejamiento de cadenas.

Un n-grama es un conjunto de n caracteres consecutivos extraídos de una cadena (palabra). La idea principal que subyace en este enfoque es que las cadenas similares tienen una mayor proporción de n-gramas en común que con las cadenas que no son similares. Los valores típicos de n que se emplean son 2 (bigramas) y 3 (trigramas). Por ejemplo, la cadena *"Universidad de Alicante"* tiene los siguientes bigramas y trigramas (el asterísco * representa un espacio en blanco):

bigramas = {*U, Un, ni, iv, ve, er, rs, si, id, da, ad, d*, *d, de, e*, *A, Al, li, ic, ca, an, nt, te, e*}

trigramas = {**U, *Un, Uni, niv, ive, ver, ers, rsi, sid, ida, dad, ad*, d*d, *de, de*, e*A, *Al, Ali, lic, ica, can, ant, nte, te*, e**}

En una cadena de n caracteres, existen $n+1$ bigramas y $n+2$ trigramas de este tipo[3].

Si usamos el modelo de n-gramas, se puede calcular la similitud (s) entre dos cadenas relacionando el número de n-gramas que tienen en común (c) con el número total de n-gramas que poseen:

$$s = \frac{2c}{2(n-1) + l + m} \tag{5.1}$$

donde l y m son las longitudes de las dos cadenas. El valor de n influye en la tolerancia del cálculo de la similitud entre dos cadenas: un valor pequeño de n permite una mayor tolerancia, mientras que un valor grande de n produce

[3]Si no se consideran los espacios en blanco al principio y al final de la cadena, existen $n-1$ bigramas y $n-2$ trigramas.

una menor tolerancia[4].

Otros métodos de comparación entre cadenas se basan en la fonética, cómo se pronuncian las palabras, en vez de cómo se escriben. Uno de los más utilizados es *Soundex* (National Archives and Records Administration, 1997; National Archives and Records Administration, 2000). El sistema de codificación Soundex fue desarrollado por la Administración para los Archivos y Registros Nacionales de los Estados Unidos para mejorar las búsquedas de apellidos en el censo federal. El objetivo de esta codificación era localizar un apellido aunque se hubiese escrito de distintas formas[5]. El método se basa en asignar los mismos códigos a aquellas grupos de consonantes que son fonéticamente similares.

En el método que proponemos, la distancia de edición o de Levenshtein es la base para el cálculo de la similitud entre dos cadenas. Esta distancia se ha comprobado que es más efectiva que Soundex (Zobel & Dart, 1996) en el emparejamiento fonético de cadenas. En otros estudios (Hernández & Stolfo, 1998) se han comparado la distancia de edición, la distancia fonética y la distancia de mecanografiado, eligiéndose la primera porque no se observaban diferencias con las otras dos.

Para calcular la similitud entre dos cadenas, empleamos la distancia de Levenshtein y la distancia invariante a la posición de las palabras (que se basa en la distancia de Levenshtein). Como estas distancias son computacionalmente costosas de calcular, empleamos dos distancias más, la distancia longitudinal y la distancia invariante trasposicional, para decidir si es necesario calcular las dos primeras (actúan como filtros).

5.2.1 Distancia de Levenshtein (DL)

La *Distancia de edición* (DE) o *Distancia de Levenshtein* (DL) (Levenshtein, 1966) se suele emplear en la búsqueda aproximada de cadenas (Zobel & Dart, 1995) y en la detección y corrección de errores ortográficos de forma automática (Kukich, 1992). En su forma más simple, la DL entre las cadenas x e y se define como *el mínimo número de operaciones de edición necesarias para transformar* x *en* y[6]. Cuanto menor sea el número de operaciones necesarias,

[4]Dos ejemplos extremos demuestran esta idea. Si se emplean 1-gramas, la tolerancia es máxima: dos cadenas se consideran iguales (100% de similitud) si poseen los mismos caracteres, independientemente del orden. Por otro lado, si los n-gramas tienen la longitud de las cadenas (suponiendo que todas las cadenas tienen la misma longitud), dos cadenas se consideran iguales si son idénticas (los mismos caracteres en el mismo orden), ya que cada cadena sólo tendrá un n-grama.

[5]Así, por ejemplo, "Smith" y "Smyth" o "Johnsen", "Johnson", "Jonsen", "Jonson", "Jonnsen" y "Jonssen" comparten el mismo código Soundex.

[6]Se obtiene la misma DL al calcular la transformación de x en y o de y en x.

más se parecerán las cadenas entre sí (su similitud será mayor).

Las operaciones de edición consideradas normalmente son la *inserción* de un carácter, el *borrado* de un carácter y la *substitución* de un carácter por otro[7], con costes[8] $\delta(\lambda,\sigma)$, $\delta(\sigma,\lambda)$ y $\delta(\sigma_1,\sigma_2)$ respectivamente. En nuestro método, hemos tomado una función de coste unitario para todas las operaciones y para todos los caracteres.

La DL entre dos cadenas de longitudes m y n se puede calcular mediante un algoritmo de programación dinámica con un coste temporal y espacial de $O(mn)$. Existen refinamientos al algoritmo básico de cálculo, que permiten obtener soluciones subcuadráticas (Hirschberg, 1997). Uno de los algoritmos que posee menor coste temporal y espacial es el de (Myers, 1986), que se emplea en la implementación que realiza *GNU* del programa `diff` (MacKenzie *et al.*, 1993).

Si tenemos las cadenas $x_i(i = 1 \ldots m)$ e $y_j(j = 1 \ldots n)$, la distancia $DL(m,n)$ entre las dos cadenas se puede calcular con la siguiente relación de recurrencia (Hirschberg, 1997):

$$\mathrm{DL}(i,j) = \begin{cases} 0 & \text{si } i=0 \text{ y } j=0 \\ \mathrm{DL}(0,j-1)+\delta(\lambda,y_j) & \text{si } i=0 \text{ y } j>0 \\ \mathrm{DL}(i-1,0)+\delta(x_i,\lambda) & \text{si } i>0 \text{ y } j=0 \\ \mathrm{DL}(i-1,j-1) & \text{si } x_i = y_j \\ \min\left\{\begin{array}{l} \mathrm{DL}(i,j-1)+\delta(\lambda,y_j) \\ \mathrm{DL}(i-1,j)+\delta(x_i,\lambda) \\ \mathrm{DL}(i-1,j-1)+\delta(x_i,y_j) \end{array}\right\} & \text{si } x_i \neq y_j \end{cases} \tag{5.2}$$

Dos cadenas se van a considerar similares si su distancia de edición es menor o igual que un umbral dado (α_{DL}). No se puede tomar un umbral constante para todas las cadenas: un umbral de 5 parece razonable cuando se comparan dos cadenas de 45 y 50 caracteres respectivamente, pero es excesivo cuando las cadenas poseen 5 y 10 caracteres (las cadenas más largas tendrán más errores o discrepancias que las más cortas). Por tanto, parece razonable utilizar un umbral relativo que dependa de la longitud de las cadenas que se comparan. Tras realizar varias pruebas, hemos elegido un umbral que es una fracción (β_{DL}) de la longitud de la cadena más corta:

[7] La inserción, el borrado y la substitución son las operaciones empleadas en la forma más simple de la DL. Existen otras medidas de similitud entre cadenas que son casos especiales o generalizaciones de la DL. Por ejemplo, una variante es considerar como operación de edición la transposición de dos caracteres adyacentes.

[8] δ representa la función de coste (depende de los caracteres implicados), λ representa el carácter vacío y σ un carácter cualquiera.

$$\alpha_{\mathrm{DL}}(x, y) = \beta_{\mathrm{DL}} \min(|x|, |y|) \tag{5.3}$$

En la tabla 5.3 mostramos la matriz triangular superior resultado del cálculo de la DL para todos los posibles emparejamientos de las cadenas de ejemplo de la tabla 5.1. En la tabla 5.4 mostramos si dos cadenas se consideran similares para distintos valores de β_{DL}: *0,3*, *0,5* y *0,7*. Por ejemplo, c_2 y c_3 no se consideran similares para 0,3, pero sí para 0,5 y 0,7.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	0	6	5	19	17	14	19
c_2		0	7	18	20	15	21
c_3			0	18	22	16	21
c_4				0	23	25	28
c_5					0	22	16
c_6						0	19
c_7							0

Tabla 5.3: Matriz cálculo DL

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	-	S-S-S	S-S-S	N-N-N	N-N-N	N-N-S	N-N-N
c_2		-	N-S-S	N-N-N	N-N-N	N-N-N	N-N-N
c_3			-	N-N-N	N-N-N	N-N-N	N-N-N
c_4				-	N-N-N	N-N-N	N-N-N
c_5					-	N-N-S	N-S-S
c_6						-	N-N-S
c_7							-

Tabla 5.4: Similitud entre pares de cadenas para la DL y β_{DL} igual a *0,3*, *0,5* y *0,7* (S: Sí, N: No).

5.2.2 Distancia longitudinal (DLON)

La DL tiene una complejidad espacial y temporal elevada. Para reducir el tiempo de cálculo, se puede evitar calcular la DL entre aquellas cadenas que difieran bastante en su longitud. Esta condición viene expresada en la ecuación 5.4, cuya demostración se puede encontrar en (French *et al.*, 1997a).

$$\mathrm{DL}(x,y) \leq \alpha_{\mathrm{DL}} \text{ si } \mid |x| - |y| \mid \leq \alpha_{\mathrm{DL}} \tag{5.4}$$

Al calcular la similitud entre dos cadenas no queremos tener en cuenta los espacios en blanco, los signos de puntuación, etc., ya que no suelen incorporar información importante a la cadena y producen distintos valores a partir de un mismo término. Por ello, empleamos la *Longitud característica* (LC) de una cadena, que la definimos como el número de letras del alfabeto español y dígitos (37 caracteres distintos en total) que contiene una cadena. En la tabla 5.1 aparecen las longitudes y las LC de las cadenas empleadas como ejemplo.

A partir de la LC, definimos una nueva distancia, la *Distancia longitudinal* (DLON) entre dos cadenas, que la definimos como la diferencia en valor absoluto de las LC de las dos cadenas. Al usar esta distancia, sólo se calculará la DL cuando la DLON sea menor que un umbral establecido. Al igual que con la DL, usamos un umbral que depende de las cadenas que se comparan: el umbral es una fracción del mínimo de las LC de las dos cadenas, tal como mostramos en la ecuación 5.5. El coste temporal de calcular DLON es constante $O(1)$, ya que las LC se pueden tener almacenadas con cada cadena, por lo que sólo es necesario realizar una comparación entre dos números enteros.

$$\alpha_{\mathrm{DLON}}(x,y) = \beta_{\mathrm{DLON}} \min(\mathrm{LC}(x), \mathrm{LC}(y)) \tag{5.5}$$

En la tabla 5.6 mostramos cuándo la DLON evita tener que calcular la DL. Para β_{DLON} igual a 0,3, se evita tener que calcular el 57% (12 de 21) de las DL, para 0,5 el 38% (8) y para 0,7 casi un 5% (1). Cuanto mayor sea el valor de β_{DLON}, más DL se calculan, ya que la diferencia entre las LC de dos cadenas puede ser mayor para que se pase a calcular la DL.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	0	2	1	3	8	8	11
c_2		0	1	1	10	10	13
c_3			0	2	9	9	12
c_4				0	11	11	14
c_5					0	0	3
c_6						0	3
c_7							0

Tabla 5.5: Matriz cálculo DLON

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	-	N-N-N	N-N-N	N-N-N	S-N-N	S-N-N	S-S-N
c_2		-	N-N-N	N-N-N	S-S-N	S-S-N	S-S-N
c_3			-	N-N-N	S-N-N	S-N-N	S-S-N
c_4				-	S-S-N	S-S-N	S-S-S
c_5					-	N-N-N	N-N-N
c_6						-	N-N-N
c_7							-

Tabla 5.6: Cuándo la DLON evita tener que calcular la DL para β_{DLON} igual a *0,3*, *0,5* y *0,7* (S: Sí, N: No).

5.2.3 Distancia invariante trasposicional (DIT)

Incluso si se usa la DLON como filtro, se realizan demasiados cálculos de la DL. Además, se trata de una métrica poco precisa: sólo se basa en la longitud de las cadenas. Para reducir aún más el número de veces que se tiene que calcular la DL, empleamos otra distancia que es menos costosa de calcular que la DL (pero más que la DLON), la *Distancia invariante trasposicional* (DIT) entre dos cadenas, que se define como (Díaz *et al.*, 1993):

$$\text{DIT}(x,y) = \frac{1}{2}\left(\sum_{i=1}^{m} |x_{\alpha_i} - y_{\alpha_i}| + |\,|x| - |y|\,|\right) \tag{5.6}$$

donde x_{α_i} e x_{α_i} son las frecuencias de aparición del carácter α_i en x e y respectivamente y m es el número de caracteres distintos que forman las cadenas (el juego de caracteres). El coste temporal de calcular la DIT es constante[9] $O(1)$. Para emplearla en nuestra investigación, hemos adaptado la DIT: sólo consideramos como caracteres las letras del alfabeto español y los dígitos, al igual que en la DLON. Por tanto, con esta modificación, la distancia queda tal como vemos en la ecuación 5.7.

$$\text{DIT}(x,y) = \frac{1}{2}\left(\sum_{i=1}^{37} |x_{\alpha_i} - y_{\alpha_i}| + |\,\text{LC}(x) - \text{LC}(y)\,|\right) \tag{5.7}$$

Al igual que con la DLON, empleamos un umbral para decidir cuando se calculará la DL. Si $\text{DIT}(x,y) \leq \alpha_{\text{DIT}}$, entonces las cadenas son similares y

[9]Las frecuencias de aparición de cada carácter en cada cadena las tenemos almacenadas junto con cada cadena en un vector de 37 componentes (se calcula cuando se leen las cadenas desde el fichero de entrada). Cuando se calcula esta distancia, sólo hay que recorrer el vector (no depende de la longitud de las cadenas).

es necesario calcular la DL. El umbral se calcula a partir de las longitudes características de las dos cadenas, tal como vemos en la ecuación 5.8.

$$\alpha_{\mathrm{DIT}}(x, y) = \beta_{\mathrm{DIT}} \min(\mathrm{LC}(x), \mathrm{LC}(y)) \tag{5.8}$$

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	0	5	5	5	9	10	11
c_2		0	5	4	13	12	15
c_3			0	2	14	12	15
c_4				0	14	12	16
c_5					0	3	4
c_6						0	6
c_7							0

Tabla 5.7: Matriz cálculo DIT

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	-	N-N-N	N-N-N	N-N-N	S-N-N	S-N-N	S-S-N
c_2		-	N-N-N	N-N-N	S-S-N	S-S-N	S-S-S
c_3			-	N-N-N	S-S-N	S-S-N	S-S-S
c_4				-	S-S-S	S-S-N	S-S-S
c_5					-	N-N-N	N-N-N
c_6						-	N-N-N
c_7							-

Tabla 5.8: Cuándo la DIT evita tener que calcular la DL para β_{DIT} igual a *0,3*, *0,5* y *0,7* (S: Sí, N: No).

Esta métrica es más costosa de calcular que DLON, pero mucho más precisa, ya que tiene en cuenta tanto la longitud de las cadenas como los caracteres que las forman. Por otro lado, se ha probado que $\mathrm{DIT}(x, y) \leq \mathrm{DL}(x, y)$[10] y que computacionalmente es menos costosa de calcular (Santana *et al.*, 1987; Santana *et al.*, 1989).

En la tabla 5.8 mostramos cuándo la DIT evita tener que calcular la DL. Para β_{DIT} igual a 0,3, se evita tener que calcular el 57% (12 de 21) de las DL, para 0,5 el 47% (10) y para 0,7 19% (4). Si observamos las tablas 5.6 y 5.8, comprobamos que la DIT evita tener que calcular la DL más veces que la DLON.

[10]Por tanto, si $\mathrm{DIT}(x, y) \geq \alpha_{\mathrm{DIT}}$, también se cumple que $\mathrm{DL}(x, y) \geq \alpha_{\mathrm{DIT}}$.

5.2.4 Distancia invariante a la posición de las palabras (DIPP)

Si tenemos varias cadenas que hacen referencia al mismo término, formadas por las mismas palabras pero en distinto orden, la DL no nos va a permitir agruparlas. Por ejemplo, en la tabla 5.3 podemos notar que la cadena c_1 es más similar a la cadena c_6 ($\mathrm{DL}(c_1, c_6) = 14$) que a c_4 ($\mathrm{DL}(c_1, c_4) = 19$). Y sin embargo, c_1 y c_4 son valores que representan el mismo término (hacen referencia a la misma institución) y nos interesaría agruparlas. Lo mismo ocurre con otras cadenas: c_2 está más próxima a c_6 ($\mathrm{DL}(c_2, c_6) = 15$) que a c_4 ($\mathrm{DL}(c_2, c_4) = 18$) y c_5 está más próxima a c_7 ($\mathrm{DL}(c_5, c_7) = 16$) que a c_6 ($\mathrm{DL}(c_5, c_6) = 22$), aunque c_5 y c_6 forman un mismo grupo.

Para solucionar este problema, empleamos otra distancia que hemos denominado *Distancia invariante a la posición de las palabras* (DIPP). Para calcular la DIPP entre dos cadenas, primero se dividen por palabras (se considera una palabra a la sucesión de letras del alfabeto español y dígitos, el resto de los caracteres no se tienen en cuenta). En la tabla 5.9 mostramos el resultado de este primer paso aplicado a las cadenas de ejemplo. Después, las palabras de ambas cadenas se van emparejando de forma que la suma de las DL de las parejas sea mínima. Si las cadenas contienen distinto número de palabras, quedarán palabras sin emparejar. En ese caso, se toma por DL la longitud de la palabra no emparejada (se considera que se ha emparejado con una palabra de longitud nula).

Cadena	**Número de palabras**	**Palabras**
c_1	3	universidad, de, alicante
c_2	3	universitat, d, alacant
c_3	3	university, of, alicante
c_4	2	alicante, university
c_5	4	ciencias, universidad, de, valencia
c_6	4	universitat, de, valencia, ciencies
c_7	4	universidad, politecnica, de, valencia

Tabla 5.9: Segmentación en palabras

En la tabla 5.10 mostramos la matriz triangular superior resultado del cálculo de la DIPP para todos los posibles emparejamientos de las cadenas de ejemplo de la tabla 5.1. En la tabla 5.11 mostramos si dos cadenas se consideran similares para distintos valores de β_{DIPP}.

Si comparamos las tablas 5.4 y 5.11, podemos comprobar que la DIPP refleja mejor la similitud entre cadenas que la DL. Las cadenas c_1, c_2, c_3 y c_4 se consideran similares en todos los casos excepto uno (c_2 y c_3 con β_{DIPP} igual a 0,3) y no son similares a ninguna otra. Las cadenas c_5 y c_6 son similares entre sí para los tres valores de β_{DIPP}. El único caso en el que la DIPP produce una falsa similitud es con la cadena c_7, que se considera similar a c_5 y c_6, cuando no debería ser similar a ninguna cadena.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	0	5	5	5	15	17	16
c_2		0	6	5	17	15	20
c_3			0	2	20	19	21
c_4				0	20	19	21
c_5					0	3	8
c_6						0	10
c_7							0

Tabla 5.10: Matriz cálculo DIPP

	c_1	c_2	c_3	c_4	c_5	c_6	c_7
c_1	-	S-S-S	S-S-S	S-S-S	N-N-N	N-N-N	N-N-N
c_2		-	N-S-S	S-S-S	N-N-N	N-N-N	N-N-N
c_3			-	S-S-S	N-N-N	N-N-N	N-N-N
c_4				-	N-N-N	N-N-N	N-N-N
c_5					-	S-S-S	S-S-S
c_6						-	N-S-S
c_7							-

Tabla 5.11: Similitud entre pares de cadenas para la DIPP y β_{DIPP} igual a *0,3*, *0,5* y *0,7* (S: Sí, N: No).

Notar que normalmente $\text{DIPP}(x, y) < \text{DL}(x, y)$, aunque no siempre se cumple[11]. Por ello, para decidir si dos cadenas son similares, se emplean ambos criterios. Primero se evalúa la DL, que es menos costosa de calcular que la DIPP: si se obtiene un valor menor que el umbral establecido α_{DL}, se considera que las dos cadenas son similares; en otro caso, se calcula la DIPP. Para la DIPP también empleamos otro umbral, que se calcula a partir de las

[11]Por ejemplo, para las cadenas *"abc def"* y *"a bcd ef"*, la DL vale 3 y la DIPP 4.

LC de las dos cadenas[12], tal como vemos en la ecuación 5.9.

$$\alpha_{\mathrm{DIPP}}(x, y) = \beta_{\mathrm{DIPP}} \min(\mathrm{LC}(x), \mathrm{LC}(y)) \tag{5.9}$$

En el algoritmo 5.1 mostramos el modo de calcular la DIPP. Básicamente, la clave del algoritmo reside en el método *emparejamiento*, que se muestra definido en el algoritmo 5.3. Este método obtiene el mejor emparejamiento posible de las palabras de las dos cadenas por medio de un esquema de ramificación y poda (*branch and bound*).

5.3 Algoritmo de agrupamiento

El algoritmo de agrupamiento que proponemos funciona de la siguiente forma:

1. Se selecciona una cadena nueva entre las cadenas que quedan si seleccionar (las cadenas se encuentran ordenadas de mayor a menor frecuencia de aparición porque se supone que las más frecuentes tienen más probabilidades de ser correctas).

2. Cada vez que se selecciona una cadena nueva, se crea un agrupamiento nuevo con la cadena como representante canónico.

3. Se comparan todas las cadenas que no han sido seleccionadas con el representante canónico del nuevo agrupamiento. Si alguna cadena se considera similar, se incorpora al agrupamiento y se elimina del conjunto de cadenas no seleccionadas.

4. El proceso se repite mientras queden cadenas sin seleccionar.

En el algoritmo 5.5 mostramos el algoritmo que hemos empleado para realizar el agrupamiento. Se basa en el *algoritmo líder* (*leader algorithm*) (Hartigan, 1975; Ryzin, 1977). Hemos elegido este algoritmo en vez de otros más complejos, como el algoritmo de k-medios[13] (k-*means*) (Hartigan, 1975; Farnstrom *et al.*, 2000) o el algoritmo de Fisher[14] (Hartigan, 1975), porque son más lentos (uno de los objetivos de esta primera aproximación es optimizar el tiempo de ejecución) y el número de agrupamientos que se quiere obtener es desconocido a priori.

[12]Se emplea la LC y no la longitud de las cadenas, porque las cadenas se dividen en palabras y sólo se tienen en cuenta las letras del alfabeto y los dígitos.

[13]Este algoritmo necesita conocer de antemano el valor de k, el número de agrupamientos que existen.

[14]Este algoritmo necesita que los datos de entrada se puedan ordenar (se suele emplear con datos numéricos).

Algoritmo 5.1 Cálculo de la DIPP (parte 1)

Entrada:
S: Array de palabras $(s_1, s_2, \ldots, s_m)$
m: Entero, longitud de S
T: Array de palabras $(t_1, t_2, \ldots, t_n)$
n: Entero, longitud de T

Salida:
$DIPP$: Entero

Variables:
i, j, aux: Entero
D: Matriz $(m+1)$ x $(n+1)$ de Entero

{Llena D con ceros}
para $i = 0$ hasta m **hacer**
  para $j = 0$ hasta n **hacer**
    $D[i][j] = 0$
  fin para
fin para
{Llena la primera columna y la primera fila de D con la longitud de s_i y t_j respectivamente}
para $i = 1$ hasta m **hacer**
  $D[i][0] = |s_i|$
fin para
para $j = 1$ hasta n **hacer**
  $D[0][j] = |t_j|$
fin para
{Sigue}

Algoritmo 5.2 Cálculo de la DIPP (parte 2)

```
{Continúa}
{Marca las palabras que se emparejan perfectamente entre ellas (tienen
una DL nula)}
para i = 1 hasta m hacer
  para j = 1 hasta n hacer
    si D[i][j] = 0 entonces
      D[i][j] = DL(s_i, t_j)
      si D[i][j] = 0 entonces
        {Marca toda la columna}
        para aux = 0 hasta m hacer
          D[aux][j] = −1
        fin para
        {Marca toda la fila}
        para aux = 0 hasta n hacer
          D[i][aux] = −1
        fin para
      fin si
    fin si
  fin para
fin para
{Llama al método de emparejamiento}
DIPP = −1
emparejamiento(D, m, n, 1, 0, DIPP)
```

Algoritmo 5.3 Emparejamiento (parte 1)

Entrada:
D: Matriz $(m+1)$ x $(n+1)$ de Entero
$m, n, i, coste, DIPP$: Entero

Salida:
$DIPP$: Entero

Variables:
j, aux: Entero

si $i > m$ **entonces**
{Todas las palabras de S se han emparejado; si alguna palabra de T no se ha emparejado, su longitud se suma a la distancia final}
$aux = coste$
para $j = 1$ hasta n **hacer**
si $D[0][j] > 0$ **entonces**
$aux = aux + D[0][j]$
fin si
fin para
si $aux < DIPP$ o $DIPP = -1$ **entonces**
$DIPP = aux$
fin si
fin si
{Sigue}

Algoritmo 5.4 Emparejamiento (parte 2)

```
{Continúa}
si i ≤ m entonces
  {Si no se han probado todas las cadenas de S}
  si D[i][0] ≠ −1 entonces
    {La palabra s_i aún no ha sido emparejada}
    para j = 1 hasta n hacer
      {Si la palabra t_j aún no ha sido emparejada}
      si D[i][j] ≠ −1 y D[0][j] > 0 entonces
        aux = coste + D[i][j]
        si aux < DIPP o DIPP = −1 entonces
          {Marca la palabra t_j como emparejada}
          D[0][j] = −D[0][j]
          emparejamiento(D, m, n, i + 1, aux, DIPP)
          {Elimina la marca de la palabra t_j}
          D[0][j] = −D[0][j]
        fin si
      fin si
    fin para
    {El algoritmo considera que la palabra s_i no se ha emparejado, por lo
    que su longitud se suma a la distancia final}
    aux = coste + D[i][0]
    si aux < DIPP o DIPP = −1 entonces
      emparejamiento(D, m, n, i + 1, aux, DIPP)
    fin si
  sino
    {La palabra s_i ya se ha emparejado, el algoritmo continúa con la
    siguiente palabra}
    emparejamiento(D, m, n, i + 1, coste, DIPP)
  fin si
fin si
```

El algoritmo líder es muy rápido, sólo requiere una pasada a través de los datos, pero tiene una serie de inconvenientes:

- la partición que se obtiene de los datos de entrada no es invariante si se reordenan los datos,
- el primer agrupamiento tiende a ser más grande que los demás porque las comparaciones empiezan por el primero y paran cuando se encuentra el primer agrupamiento válido (aunque no sea el mejor) y,
- por último, el número final de los agrupamientos obtenidos depende del valor de los umbrales empleado.

El algoritmo que proponemos depende de cuatro parámetros: β_{DLON}, β_{DIT}, β_{DL} y β_{DIPP} (empleados en el cálculo de los correspondientes umbrales α_{DLON}, α_{DIT}, α_{DL} y α_{DIPP}).

Los dos primeros parámetros reducen el número de distancias DL y DIPP que se tienen que calcular. Si su valor es muy pequeño, menos distancias DL y DIPP se tendrán que calcular, disminuyendo el tiempo de ejecución, pero los agrupamientos obtenidos tendrán una menor calidad (no se agruparán cadenas que representan el mismo término). Si su valor es muy grande, las distancias DL y DIPP se calcularán con casi todas las cadenas, aumentando el tiempo de ejecución, pero los agrupamientos obtenidos tendrán una mayor calidad.

El algoritmo de agrupamiento posee una complejidad cuadrática $O(n^2)$:

$$(n-1)+(n-2)+\ldots+1+0=\frac{n^2-n}{2} \tag{5.10}$$

5.4 Evaluación de los resultados

5.4.1 Medidas de evaluación

Para evaluar los agrupamientos obtenidos al aplicar nuestro método a los ficheros de prueba, hemos calculado cuatro medidas a partir de los agrupamientos generados:

1. *Número de agrupamientos* (NA)
   Es el número total de agrupamientos (correctos e incorrectos) que se han generado.

Algoritmo 5.5 Algoritmo de agrupamiento (aproximación 1)

Entrada:
C: Cadenas ordenadas de mayor a menor frecuencia de aparición $(c_1, c_2, \ldots, c_m)$
$\beta_{\mathrm{DLON}}, \beta_{\mathrm{DIT}}, \beta_{\mathrm{DL}}, \beta_{\mathrm{DIPP}}$: Real, Parámetros

Salida:
G: Conjunto de agrupamientos (grupos) obtenidos $(g_1, g_2, \ldots, g_n)$

Variables:
i, k: Entero

$k = 1$
para todo $c_i \in C$ **hacer**
  $C = C - \{c_i\}$
  {Crea un agrupamiento nuevo con la cadena elegida}
  $g_k = \{c_i\}$
  {Compara el representante canónico con las cadenas que quedan sin elegir}
  para todo $c_j \in C$ **hacer**
    si $\mathrm{DLON}(c_i, c_j) < \alpha_{\mathrm{DLON}}(c_i, c_j)$ **entonces**
      si $\mathrm{DIT}(c_i, c_j) < \alpha_{\mathrm{DIT}}(c_i, c_j)$ **entonces**
        si $\mathrm{DL}(c_i, c_j) < \alpha_{\mathrm{DL}}(c_i, c_j)$ **entonces**
          {Incluye la cadena seleccionada en el agrupamiento}
          $g_k = g_k + \{c_j\}$
          $C = C - \{c_j\}$
        sino si $\mathrm{DIPP}(c_i, c_j) < \alpha_{\mathrm{DIPP}}(c_i, c_j)$ **entonces**
          {Incluye la cadena seleccionada en el agrupamiento}
          $g_k = g_k + \{c_j\}$
          $C = C - \{c_j\}$
        fin si
      fin si
    fin si
  fin para
  $k = k + 1$
fin para

2. *Número de agrupamientos exactos* (NAE)
   En el número de agrupamientos que coinciden exactamente con los óptimos: contienen las mismas cadenas. A partir de esta medida obtenemos la *Precisión*: NAE dividido entre NAO.

3. *Número de agrupamientos erróneos* (NAR)
   Es el número de agrupamientos que contienen alguna cadena errónea (falsos positivos). Si un agrupamiento no contiene ninguna cadena errónea, pero no está completo (no contiene todas las cadenas que forman el agrupamiento), no se considera erróneo. A partir de esta medida obtenemos el *Error*: NAR divido entre NAO.

4. *Número de cadenas erróneas* (NCE)
   Es el número de cadenas que se han incluido en agrupamientos que no son el suyo.

Otros autores (French *et al.*, 1997a; French *et al.*, 1997b; Monge & Elkan, 1997) emplean como medida de precisión la *Pureza*, que definen como el porcentaje de agrupamientos que no contienen cadenas erróneas. Empleando nuestras medidas, la Pureza se calcularía como

$$Pureza = \frac{NA - NAR}{NA} \tag{5.11}$$

Pensamos que las medidas que nosotros empleamos, la precisión y el error, reflejan mejor la calidad de los agrupamientos que la pureza. Por ejemplo, si las cadenas no se agrupan (se genera un agrupamiento individual por cada cadena), se obtiene una pureza del 100%, ya que ninguno de los agrupamientos contiene cadenas erróneas; con nuestras medidas, se obtiene una precisión que es el porcentaje de agrupamientos individuales que existen en la solución óptima y un error del 0%.

5.4.2 Valores obtenidos

Las pruebas se han realizado con los ficheros A, B, C y D. Como se ha comentado varias veces, el algoritmo depende de cuatro parámetros. Las pruebas se han realizado asignando distintos valores a dichos parámetros.

Los dos primeros parámetros β_{DLON} y β_{DIT} sirven para reducir el número de distancias DL y DIPP que se tienen que calcular. Hemos realizado pruebas fijando el valor de ambos en *0,3*, *0,4*, *0,5* y *0,6*. Valores inferiores a *0,3* producen una disminución en el tiempo de ejecución, pero la precisión que se obtiene es peor (el error no aumenta). Valores superiores a *0.6* no producen un aumento de la precisión, pero si que aumenta el tiempo de ejecución.

Para los parámetros β_{DL} y β_{DIPP} hemos tomado los siguientes valores: *0,0*, *0,05*, *0,1*, *0,15*, *0,2*, *0,25*, *0,3*, *0,35*, *0,4*, *0,45* y *0,5*. Valores mayores que *0,5* no producen un aumento de la precision y sí del error. Las pruebas se han repetido dos veces: una sin expansión de abreviaturas y otra con expansión. En total se han realizado 3872 (4 x 4 x 11 x 11 x 2) pruebas con cada fichero.

En la tabla 5.12 mostramos las mejores precisiones, con su correspondiente error, que se obtienen para los ficheros A, B, C y D sin expandir las abreviaturas. Como se puede observar, los valores para los que se obtienen las mejores precisiones no son fijos, ya que cambian de un fichero a otro. En la tabla 5.14 mostramos los valores de NA, NAE, NAR y NCE correspondientes a las mejores precisiones obtenidas.

La tabla 5.13 muestra los mismos resultados pero con expansión de las abreviaturas. En la tabla 5.15 mostramos los correspondientes valores de las medidas de evaluación.

Fichero	**Parámetros**				**Precisión**	**Error**
	β_{DLON}	β_{DIT}	β_{DL}	β_{DIPP}	(%)	(%)
A	0,4	0,4	0,1	0,35	70,7	7,6
B	0,4	0,4	0,3	0,35	67,4	8,7
C	0,3	0,3	0,25	0,25	85,9	1,7
D	0,5	0,6	0,45	0,45	43,8	16,8

Tabla 5.12: Precisión y error sin expansión de abreviaturas

Fichero	**Parámetros**				**Precisión**	**Error**
	β_{DLON}	β_{DIT}	β_{DL}	β_{DIPP}	(%)	(%)
A	0,4	0,4	0,1	0,2	84,8	0
B	0,4	0,4	0,3	0,35	72,8	6,5
C	0,3	0,3	0,25	0,25	84,2	1,7
D	0,3	0,3	0,2	0,3	67,7	6,2

Tabla 5.13: Precisión y error con expansión de abreviaturas

Los ficheros A y C obtiene una mejor precisión que el fichero B, debido a que sus agrupamientos son menos complejos: si consultamos la tabla 4.7[15], comprobaremos que el ICF de los ficheros A y C vale alrededor de 0,3, mientras que el ICF del fichero B vale 1,7, casi seis veces más. Los resultados

[15] Página 36.

obtenidos con el fichero D nos han extrañado, ya que aunque su ICF es menor que el del fichero B, obtiene la menor precisión sin y con expansión de abreviaturas.

En los ficheros A, B y D, la expansión de abreviaturas produce mejoras en los resultados: la precisión aumenta y el error disminuye, tal como se comprueba al comparar las tablas 5.12 y 5.13. Para los ficheros A, B y D, la precisión máxima que se obtiene es 70,7%, 67,4% y 43,8% sin expansión de abreviaturas y 84,8%, 72,8% y 67,7% con expansión. En el fichero C, la expansión de abreviaturas no produce una mejora en la precisión y el error. Esto se debe a que la reducción del número de cadenas al expandirse las abreviaturas es irrelevante (una reducción del 0,8%, como podemos ver en la tabla 4.1[16]), lo que se traduce en que el ICF no disminuya prácticamente (pasa de 0,33 a 0,31).

Fichero	**Parámetros**				**NA**	**NAE**	**NAR**	**NCE**
	β_{DLON}	β_{DIT}	β_{DL}	β_{DIPP}				
A	0,4	0,4	0,1	0,35	114	65	7	15
B	0,4	0,4	0,3	0,35	117	62	8	59
C	0,3	0,3	0,25	0,25	72	49	1	1
D	0,5	0,6	0,45	0,45	266	99	38	106

Tabla 5.14: Medidas de evaluación sin expansión de abreviaturas

Fichero	**Parámetros**				**NA**	**NAE**	**NAR**	**NCE**
	β_{DLON}	β_{DIT}	β_{DL}	β_{DIPP}				
A	0,4	0,4	0,1	0,2	107	78	0	0
B	0,4	0,4	0,3	0,35	104	67	6	75
C	0,3	0,3	0,25	0,25	70	48	1	1
D	0,3	0,3	0,2	0,3	274	153	14	25

Tabla 5.15: Medidas de evaluación con expansión de abreviaturas

En las tablas 5.16 y 5.17 mostramos como influye el valor del parámetro β_{DIPP} en las medidas de evaluación en los ficheros A y B. En las tablas 5.18 y 5.19 podemos ver como influye en la precisión y en el error. En las cuatro tablas los parámetros β_{DLON}, β_{DIT} y β_{DL} toman los valores mostrados en las tablas 5.12 y 5.13, que son los que producen los mejores resultados en cada fichero.

[16]Página 31.

β_{DIPP}	NA		NAE		NAR		NCE	
	SIN	CON	SIN	CON	SIN	CON	SIN	CON
0,1	179	130	21	62	0	0	0	0
0,2	148	107	40	78	0	0	0	0
0,3	119	97	63	73	4	3	9	8
0,35	114	94	65	71	7	5	15	11
0,4	106	86	63	64	11	8	22	18
0,45	101	83	65	63	11	11	28	22
0,5	95	74	57	48	16	17	42	32

Tabla 5.16: Resultados fichero A para $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,1$

β_{DIPP}	NA		NAE		NAR		NCE	
	SIN	CON	SIN	CON	SIN	CON	SIN	CON
0,1	123	108	57	66	8	5	51	52
0,2	123	108	57	66	8	5	51	52
0,3	122	107	58	66	8	6	51	52
0,35	117	104	62	67	8	6	59	75
0,4	109	99	62	65	11	8	87	94
0,45	107	99	56	60	15	10	110	115
0,5	97	92	40	41	26	21	186	203

Tabla 5.17: Resultados fichero B para $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,3$

No es posible comparar de forma directa los resultados obtenidos con otros métodos, al no disponer de los ficheros empleados en otros trabajos. Sólo disponemos del fichero C, que pertenece al trabajo desarrollado en (French *et al.*, 1997a; French *et al.*, 1997b). En este trabajo, se emplea como medida de evaluación la pureza; como ya hemos comentado, consideramos que esta medida no refleja correctamente la calidad de los agrupamientos obtenidos.

5.4.3 Distancias calculadas

En las tablas 5.20 y 5.21 mostramos el número de distancias de cada tipo (DLON, DIT, DL y DIPP) calculadas para los parámetros que logran los mejores resultados en los ficheros A y B respectivamente. Se muestran los resultados sin y con expansión de abreviaturas. En ambos casos, el número de DL y DIPP que se calcula se reduce en más de un 80% respecto al número de DLON calculadas. Si no se empleasen las DLON y DIT como filtros, se

β_{DIPP}	**Precisión (%)**		**Error (%)**	
	SIN	**CON**	**SIN**	**CON**
0,1	22,8	67,4	0,0	0,0
0,2	43,5	84,8	0,0	0,0
0,3	68,5	79,3	4,3	3,3
0,35	70,7	77,2	7,6	5,4
0,4	68,5	69,6	12,0	8,7
0,45	70,7	68,5	12,0	12,0
0,5	62,0	52,2	17,4	18,5

Tabla 5.18: Precisión y Error fichero A para $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,1$

β_{DIPP}	**Precisión (%)**		**Error (%)**	
	SIN	**CON**	**SIN**	**CON**
0,1	62,0	71,7	8,7	5,4
0,2	62,0	71,7	8,7	5,4
0,3	63,0	71,7	8,7	6,5
0,35	67,4	72,8	8,7	6,5
0,4	67,4	70,7	12,0	8,7
0,45	60,9	65,2	16,3	10,9
0,5	43,5	44,6	28,3	22,8

Tabla 5.19: Precisión y Error fichero B para $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,3$

tendrían que calcular tantas DL y DIPP como indica la columna de la DLON.

También podemos comprobar como afecta la expansión de abreviaturas en el número de distancias calculadas. En todos los casos, la expansión produce una reducción considerable en el número de distancias calculadas. En el fichero A, la expansión de abreviaturas produce una reducción mucho mayor que en el fichero B. Esto se debe a que la reducción en el número de cadenas es mayor en el A (38%) que en el B (7,8%).

β_{DIPP}	**DLON**		**DIT**		**DL**		**DIPP**	
	SIN	**CON**	**SIN**	**CON**	**SIN**	**CON**	**SIN**	**CON**
0,1	19.903	9.253	9.398	4.733	3.472	1.936	3.422	1.927
0,2	15.372	7.721	7.134	3.883	2.547	1.587	2.505	1.580
0,3	11.956	6.695	5.429	3.495	1.860	1.462	1.824	1.457
0,35	11.071	6.458	5.106	3.348	1.789	1.404	1.756	1.399
0,4	9.978	5.802	4.592	2.989	1.616	1.264	1.587	1.260
0,45	9.196	5.616	4.276	2.919	1.504	1.241	1.475	1.238
0,5	8.257	4.601	3.852	2.348	1.367	989	1.342	986

Tabla 5.20: Número de distancias calculadas en el fichero A con $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,1$

β_{DIPP}	**DLON**		**DIT**		**DL**		**DIPP**	
	SIN	**CON**	**SIN**	**CON**	**SIN**	**CON**	**SIN**	**CON**
0,1	53.308	43.285	24.700	21.563	9.620	9.122	8.532	8.114
0,2	53.262	43.274	24.676	21.555	9.618	9.123	8.531	8.116
0,3	53.102	43.122	24.641	21.514	9.603	9.094	8.519	8.107
0,35	51.416	42.751	23.877	21.288	9.397	9.024	8.340	8.054
0,4	48.267	39.993	22.723	20.372	9.056	8.659	8.034	7.722
0,45	46.830	38.524	22.071	19.731	8.839	8.455	7.845	7.541
0,5	42.915	32.622	19.941	16.429	7.987	7.069	7.061	6.224

Tabla 5.21: Número de distancias calculadas en el fichero B con $\beta_{\text{DLON}} = 0,4$, $\beta_{\text{DIT}} = 0,4$ y $\beta_{\text{DL}} = 0,3$

Capítulo 6

Agrupamiento de valores similares: aproximación 2

En este capítulo presentamos la segunda solución que hemos planteado. El objetivo de esta solución es obtener la mejor precisión con el menor error posible. Para lograrlo, se introducen tres medidas de similitud entre cadenas adicionales a las estudiadas en el capítulo anterior y se emplea un algoritmo de agrupamiento distinto al de la aproximación 1. Este segundo método se evalúa probando las cinco medidas de similitud entre cadenas con los cuatro ficheros usados en la aproximación 1.

6.1 Introducción

El método propuesto en la aproximación 1 obtiene unos resultados satisfactorios, tal como vemos resumido en la tabla 6.1. Sin embargo, presenta una serie de inconvenientes que sería deseable corregir:

- El *algoritmo líder* es simple de calcular y entender, pero posee una serie de desventajas. La principal es que favorece el primer agrupamiento: la comparación entre una cadena nueva y los agrupamientos existentes se realiza hasta que se localiza un agrupamiento que satisface la condición, sin considerar la posibilidad de que pueda existir un agrupamiento mejor.

- El algoritmo depende de cuatro parámetros, ¿qué valores se tienen que elegir para cada uno de ellos?

- Aunque la DIPP es mejor que la DL porque no le afecta el orden de las palabras dentro de una cadena, también plantea problemas porque

en algunos casos considera similares cadenas que sólo se diferencian en una palabra.

Fichero	SIN		CON	
	Precisión	Error	Precisión	Error
A	70,7	7,6	84,8	0
B	67,4	8,7	72,8	6,5
C	85,9	1,7	84,2	1,7
D	43,8	16,8	67,7	6,2

Tabla 6.1: Precisión y Error sin y con expansión de abreviaturas (aproximación 1)

6.2 Procesamiento previo de las cadenas

Las cadenas antes de ser agrupadas sufren el mismo procesamiento que hemos detallado en la sección 5.1[1]. El objetivo de este procesamiento previo es eliminar las tres causas principales que originan distintos valores a partir de un mismo término: los acentos ortográficos, el uso de minúsculas/mayúsculas y las abreviaturas, siglas y acrónimos.

6.3 Similitud entre dos cadenas

Como distancia de similitud entre dos cadenas, además de la DL y la DIPP, introducimos tres distancias más: *Distancia invariante a la posición de las palabras modificada* (DIPPM), el *Coeficiente de Jaccard* (CJ) y la *Medida de similitud combinada* (MSC), que es el mínimo de las cuatro distancias anteriores para cada cadena.

6.3.1 Distancia invariante a la posición de las palabras modificada (DIPPM)

Damerau (Damerau, 1964) señala que el 80% de los errores de escritura se debe a una de las cuatro operaciones de edición simples (inserción, borrado, substitución o transposición). Por tanto, cuando la distancia de edición entre dos palabras sea 1, podemos considerar que se trata de la misma palabra.

[1] Página 39.

Bajo este supuesto, definimos la *Distancia invariante a la posición de las palabras modificada* (DIPPM) a partir de la DIPP: cuando se emparejan las palabras en la parte inicial del algoritmo de cálculo de la DIPP, permitimos que dos palabras se emparejen perfectamente si su DL es menor que un cierto valor (ecuación 6.1), que depende de la longitud de las palabras (de este modo, a las palabras con mayor longitud se les permiten más errores).

$$\mathrm{DL}(x, y) \leq 1 + \frac{|x| + |y|}{20} \tag{6.1}$$

En el algoritmo 6.1 mostramos el algoritmo de cálculo de la DIPPM dividido en tres partes. La modificación incorporada al algoritmo DIPP se muestra en la tercera parte (algoritmo 6.3): son las líneas previas a la llamada de la función *emparejamiento* (esta función se sigue calculando como se muestra en el algoritmo 5.3[2]).

6.3.2 Coeficiente de Jaccard (CJ)

El *Coeficiente de Jaccard* (CJ) (Rijsbergen, 1979) se emplea normalmente para calcular la similitud entre dos conjuntos de elementos. Se define como la división del número de elementos que comparten los dos conjuntos (elementos en común) entre el número de elementos distintos de los dos conjuntos (ecuación 6.2).

$$\mathrm{CJ(X, Y)} = \frac{|X \cap Y|}{|X \cup Y|} \tag{6.2}$$

Cuando el CJ se aplica para calcular la similitud entre dos cadenas, X e Y representan los conjuntos de palabras de las dos cadenas. Si el CJ vale 0 significa que las dos cadenas no tienen ninguna palabra en común, si vale 1 significa que poseen las mismas palabras (el orden de las palabras en la cadena no influye).

6.3.3 Medida de similitud combinada (MSC)

Finalmente, hemos combinado las cuatro medidas de similitud presentadas (DL, DIPP, DIPPM y CJ) en una sola que hemos denominado *Medida de similitud combinada* (MSC): para cada par de cadenas que se comparan, la MSC toma el valor mínimo de las otras cuatro medidas de similitud. Equivale a considerar que dos cadenas serán similares si al menos una de las cuatro distancias que empleamos las considera similares.

[2]Página 53.

Algoritmo 6.1 Cálculo de la DIPPM (parte 1)

Entrada:
S: Array de palabras $(s_1, s_2, \ldots, s_m)$
m: Entero, longitud de S
T: Array de palabras $(t_1, t_2, \ldots, t_n)$
n: Entero, longitud de T

Salida:
$DIPP$: Entero

Variables:
i, j, aux: Entero
D: Matriz $(m+1)$ x $(n+1)$ de Entero

{Llena D con ceros}
para $i = 0$ hasta m **hacer**
  para $j = 0$ hasta n **hacer**
    $D[i][j] = 0$
  fin para
fin para
{Llena la primera columna y la primera fila de D con la longitud de s_i y t_j respectivamente}
para $i = 1$ hasta m **hacer**
  $D[i][0] = |s_i|$
fin para
para $j = 1$ hasta n **hacer**
  $D[0][j] = |t_j|$
fin para
{Sigue}

Algoritmo 6.2 Cálculo de la DIPPM (parte 2)

```
{Continúa}
{Marca las palabras que se emparejan perfectamente entre ellas (tienen
una DL nula)}
para i = 1 hasta m hacer
  para j = 1 hasta n hacer
    si D[i][j] = 0 entonces
      D[i][j] = DL(s_i, t_j)
      si D[i][j] = 0 entonces
        {Marca toda la columna}
        para aux = 0 hasta m hacer
          D[aux][j] = −1
        fin para
        {Marca toda la fila}
        para aux = 0 hasta n hacer
          D[i][aux] = −1
        fin para
      fin si
    fin si
  fin para
fin para
{Sigue}
```

Algoritmo 6.3 Cálculo de la DIPPM (parte 3)

```
{Continúa}
{INICIO MODIFICACIÓN ->}
{Marca las palabras que se emparejan con un error menor que la cota}
para i = 1 hasta m hacer
  para j = 1 hasta n hacer
    si D[i][j] > 0 entonces
      si D[i][j] ≤ 1 + (D[i][0]+D[0][j])/20 entonces
        {Marca toda la columna}
        para aux = 0 hasta m hacer
          D[aux][j] = -1
        fin para
        {Marca toda la fila}
        para aux = 0 hasta n hacer
          D[i][aux] = -1
        fin para
      fin si
    fin si
  fin para
fin para
{<- FIN MODIFICACIÓN}
{Llama al método de emparejamiento}
DIPP = -1
emparejamiento(D, m, n, 1, 0, DIPP)
```

Para poder calcular el mínimo de las cuatro distancias es necesario normalizarlas, ya que no es lo mismo una distancia de 2 entre dos cadenas de longitud 5 que entre dos cadenas de longitud 20. Las distancias las normalizamos de la siguiente forma:

- DL, DIPP y DIPPM se dividen entre la longitud de la cadena más larga de las dos. De este modo, se obtiene un valor entre 0 (las dos cadenas son la misma, máxima similitud) y 1 (las dos cadenas son totalmente diferentes, mínima similitud).
- El CJ produce un valor entre 0 (mínima similitud) y 1 (máxima similitud). Para poder comparar esta medida con las anteriores, empleamos $1 - \mathrm{CJ}$.

6.4 Algoritmo de agrupamiento

El algoritmo que empleamos en la segunda aproximación evita el principal problema del *algoritmo líder*: cuando se elige una cadena nueva, se compara con todos los agrupamientos existentes hasta ese momento, para localizar el agrupamiento más similar a la cadena (no sólo el primero, como hacía la aproximación 1). Básicamente, el algoritmo funciona de la siguiente forma:

1. Por cada agrupamiento, se mantiene un *centroide*[3]: una cadena que se toma como representante de las demás cadenas del agrupamiento, similar al *representante canónico* de la aproximación 1.
2. Al contrario que en la primera aproximación, donde una vez elegido un representante canónico, éste no cambiaba durante todo el algoritmo, el centroide sí que puede variar: se recalcula cada vez que se incorpora una cadena nueva al agrupamiento. El centroide se elige de forma que minimice la suma de los cuadrados de las distancias de él al resto de las cadenas del agrupamiento:

$$\sum_{i=1}^{n} (\mathrm{D}(c_i, C))^2 \tag{6.3}$$

   donde n es el número de cadenas asignadas al agrupamiento, C es el centroide del agrupamiento y D es una de las distancias empleadas (DL, DIPP, DIPPM, CJ y MSC).

[3]El centroide de un objeto es su centro de gravedad. En estadística se emplea para indicar el elemento que representa a un conjunto.

3. Cada vez que se elige una cadena nueva, se compara con los centroides de todos los agrupamientos existentes hasta el momento, para localizar el más próximo (el que más se parezca). Para ello, se calcula la distancia de la cadena a cada uno de los centroides.

4. Si una cadena no se puede asignar al agrupamiento más próximo, porque no es suficientemente similar a él (se establece un umbral α), se crea un agrupamiento nuevo para ella.

5. El proceso se repite mientras queden cadenas sin seleccionar.

En el algoritmo 6.4 mostramos el algoritmo empleado en la segunda aproximación. La complejidad de este algoritmo es cuadrática ($O(n^2)$) en el peor de los casos[4]:

$$1 + 2 + \ldots + (n-2) + (n-1) = \frac{n^2 - n}{2} \tag{6.4}$$

6.5 Evaluación de los resultados

El algoritmo de agrupamiento depende de un parámetro (α). En las pruebas hemos asignado a este parámetro valores entre 0,0 y 0,599[5], en incrementos de 0,001 (600 valores distintos). Hemos estudiado el impacto de las abreviaturas en los resultados: hemos realizado todas las pruebas sin expansión de abreviaturas y con expansión. Las pruebas se han realizado cinco veces, cada vez con una distancia distinta. Los ficheros empleados han sido los cuatro ficheros usados en la aproximación 1. Por tanto, en total se han realizado 240.000[6] pruebas. No podemos mostrar, evidentemente, todos los valores obtenidos.

El proceso de evaluación ha seguido el mismo procedimiento que en la aproximación 1: primero hemos obtenido cuatro medidas a partir de los agrupamientos obtenidos (NA, NAE, NAR y NCE) y a partir de ellas hemos obtenido la *Precisión* y el *Error*.

En la figura 6.1 mostramos el NA y el NAE que se obtienen en el fichero A sin (SIN) y con expansión de abreviaturas (CON) al emplear la MSC. También se ha representado el NAO, que es la línea horizontal que corta el

[4] Cuando todos los agrupamientos se compongan de una única cadena.

[5] En las pruebas previas a los resultados que mostramos, hemos comprobado que valores superiores a 0,6 no son convenientes, ya que la precisión disminuye bastante y el error aumenta.

[6] 600 valores de parámetros, 2 modos (sin y con expansión de abreviaturas), 5 distancias y 4 ficheros.

Algoritmo 6.4 Algoritmo de agrupamiento (aproximación 2)

Entrada:
C: Cadenas ordenadas de mayor a menor frecuencia de aparición $(c_1, c_2, \ldots, c_m)$
α: Real, Umbral

Salida:
G: Conjunto de agrupamientos (grupos) obtenidos $(g_1, g_2, \ldots, g_n)$

Variables:
b, d, i, j, k, l: Entero

$i = 1$
$k = 1$
{Se crea el primer agrupamiento con la primera cadena}
$g_k = \{c_i\}$
para $i = 2$ hasta m **hacer**
  {El mejor agrupamiento, por ahora, es el primero}
  $b = 1$
  $d = \mathrm{D}(c_i, \mathrm{centroide}(g_b))$
  para $j = 1$ hasta k **hacer**
    si $\mathrm{D}(c_i, \mathrm{centroide}(g_j) < d$ **entonces**
      {Se ha encontrado un agrupamiento mejor}
      $b = j$
      $d = \mathrm{D}(c_i, \mathrm{centroide}(g_b))$
    fin si
  fin para
  si $d \leq \alpha$ **entonces**
    {Se añade al mejor agrupamiento que se ha encontrado}
    $g_b = g_b + \{c_i\}$
    recalcularCentroide(g_b)
  sino
    {Ningún agrupamiento de los existentes es adecuado, se crea un agrupamiento nuevo}
    $k = k + 1$
    $g_k = \{c_i\}$
  fin si
fin para

eje de ordenadas entre 80 y 100 (exactamente 92). Cuanto más se aproximen las curvas de NA y NAE a NAO, mejores son los resultados obtenidos, porque más se aproximan al óptimo. Como podemos observar, la expansión de abreviaturas reduce NA y aumenta NAE. Sin embargo, conforme se aumenta el umbral, la expansión de abreviaturas deja de notarse.

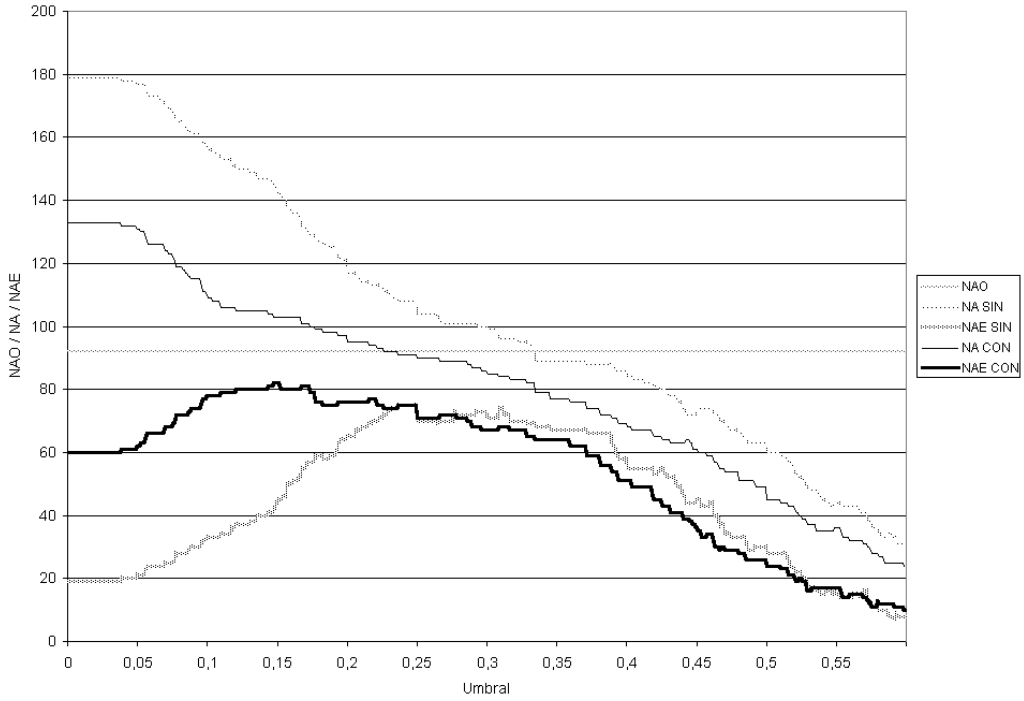

Figura 6.1: NA y NAE del fichero A sin y con expansión de abreviaturas (MSC)

En la figura 6.2 mostramos el NA y el NAE que se obtienen en el fichero D sin (SIN) y con expansión de abreviaturas (CON) al emplear la MSC. El comportamiento es similar al observado en la figura 6.1 con el fichero A. En esta figura, el NAO vale 226.

En las tablas 6.2, 6.3, 6.4, 6.5 y 6.6 comparamos la precisión y el error que se obtienen al aplicar las distintas distancias (DL, DIPP, DIPPM, CJ y MSC). Para cada distancia, mostramos la mejor precisión y el error correspondiente que se obtiene en cada uno de los ficheros. También aparecen los valores del umbral (α) que producen la mejor precisión.

En todos los casos excepto uno (fichero C con CJ, tabla 6.5), la expansión de abreviaturas mejora la precisión y disminuye el error. Además, también

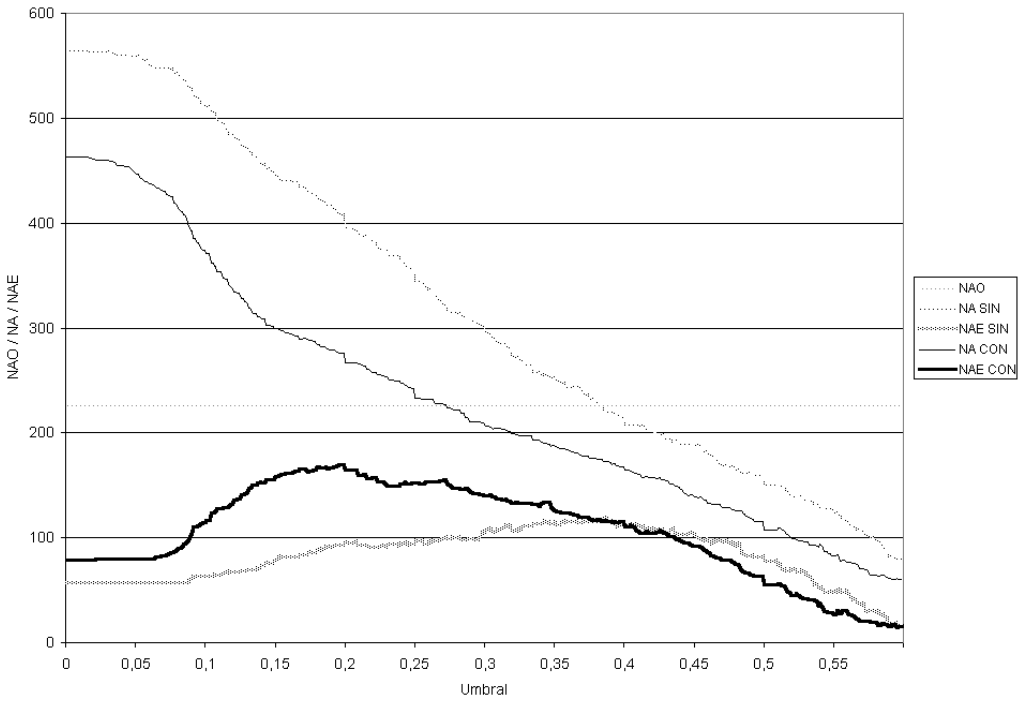

Figura 6.2: NA y NAE del fichero D sin y con expansión de abreviaturas (MSC)

disminuye el valor del umbral al que se obtiene la mejor precisión.

La mejor precisión se obtiene en cada fichero cuando se emplean distintas distancias (no hay una distancia que siempre produzca la mejor precisión en todos los ficheros):

- Fichero A
  - SIN: 81,4 de precisión y 8,6 de error con MSC.
  - CON: 89,1 de precisión y 0 de error con DIPPM y MSC.
- Fichero B
  - SIN: 71,7 de precisión y 9,7 de error con MSC.
  - CON: 77,1 de precisión y 2,1 de error con MSC.
- Fichero C
  - SIN: 89,4 de precisión y 1,7 de error con CJ.
  - CON: 87,7 de precisión y 1,7 de error con CJ.
- Fichero D
  - SIN: 53,0 de precisión y 15,9 de error con DIPP.
  - CON: 76,1 de precisión y 2,6 de error con DIPPM.

Fichero		α	**Precisión** (%)	**Error** (%)
A	SIN	0,311	76,0	8,6
	CON	[0,146, 0,151]	83,6	0
B	SIN	[0,272, 0,281]	68,4	9,7
	CON	[0,276, 0,281]	73,9	5,4
C	SIN	[0,159, 0,199]	84,2	1,7
	CON	[0,100, 0,127]	84,2	0
D	SIN	[0,429, 0,433]	49,5	19,9
	CON	[0,320, 0,321]	63,2	11,9

Tabla 6.2: DL

Si comparamos las precisiones obtenidas con las distintas distancias, ninguna destaca sobre las demás. La MSC obtiene la mejor precisión o una precisión muy próxima a la mejor en la mayoría de los casos.

Fichero		α	Precisión (%)	Error (%)
A	SIN	[0,334, 0,344]	81,5	10,8
	CON	[0,160, 0,166]	84,7	0
B	SIN	[0,347, 0,369]	64,1	11,9
	CON	[0,341, 0,342]	67,3	8,6
C	SIN	[0,143, 0,227]	82,4	1,7
	CON	[0,072, 0,119]	82,4	0
D	SIN	[0,435, 0,437]	53,0	15,9
	CON	[0,218, 0,222]	72,1	3,0

Tabla 6.3: DIPP

Los ficheros A y C obtienen mejores precisiones que los ficheros B y D. Este resultado era de esperar, ya que si consultamos los valores del FD (tabla 4.6[7]) y del ICF (tabla 4.7[8]), los ficheros A y C poseen los menores valores.

Los resultados obtenidos con el fichero D nos han sorprendido: aunque posee un ICF muy inferior al fichero B (0,53 frente a 1,72 sin expansión de abreviaturas y 0,27 frente a 1,11 con expansión), la precisión obtenida es inferior. Destaca especialmente la mejor precisión obtenida sin expansión de abreviaturas: 53,0% con un 15,9% de error cuando se emplea la DIPP (tabla 6.3), mientras que el fichero B obtiene un 71,5% con la MSC. Cuando se expanden las abreviaturas, el fichero D obtiene una precisión menor que el resto de ficheros.

En las figuras 6.3, 6.4, 6.6 y 6.7 mostramos la precisión y el error que se obtienen en cada uno de los ficheros cuando se emplea la MSC. Se muestra sin y con expansión de abreviaturas, para poder comparar el efecto que produce la expansión en la precisión. En todos los casos, la expansión produce una mejora en la precisión cuando el umbral empleado se encuentra por debajo de 0,3. Para valores superiores, las diferencias obtenidas en la precisión sin y con expansión de abreviaturas disminuyen.

La figura 6.5 muestra la precisión que se obtiene en el fichero C sin expansión de abreviaturas cuando se emplean las distintas distancias de similitud. El CJ obtiene la máxima precisión (cerca de un 90%). Todas las medidas, excepto el CJ, muestran un comportamiento similar: empiezan en un valor similar de precisión (75%), aumentan hasta un 85% y disminuyen hasta un 20% en el umbral máximo. Sin embargo, el CJ mantiene un valor estable por

[7]Página 35.
[8]Página 36.

Fichero		α	**Precisión** (%)	**Error** (%)
A	SIN	[0,276, 0,277]	80,4	9,7
	CON	[0,153, 0,166]	89,1	0
B	SIN	[0,381, 0,382]	66,3	16,3
	CON	[0,369, 0,370]	68,4	8,6
C	SIN	[0,143, 0,227]	82,4	1,7
	CON	[0,072, 0,119]	84,2	0
D	SIN	0,388	52,2	16,3
	CON	0,187	76,1	2,6

Tabla 6.4: DIPPM

Fichero		α	**Precisión** (%)	**Error** (%)
A	SIN	[0,400, 0,416]	72,8	6,5
	CON	[0,286, 0,299]	85,8	0
B	SIN	[0,500, 0,538]	56,5	17,3
	CON	[0,455, 0,461]	58,6	7,6
C	SIN	[0,471, 0,499]	89,4	1,7
	CON	[0,471, 0,499]	87,7	1,7
D	SIN	[0,500, 0,533]	52,6	13,2
	CON	[0,385, 0,399]	73,4	3,5

Tabla 6.5: CJ

Fichero		α	**Precisión** (%)	**Error** (%)
A	SIN	[0,236, 0,249]	81,5	8,6
	CON	[0,147, 0,151]	89,1	0
B	SIN	[0,270, 0,288]	71,7	9,7
	CON	[0,174, 0,176]	77,1	2,1
C	SIN	[0,143, 0,199]	84,2	1,7
	CON	[0,097, 0,119]	84,2	0
D	SIN	[0,385, 0,387]	52,6	17,2
	CON	[0,195, 0,199]	75,2	3,9

Tabla 6.6: MSC

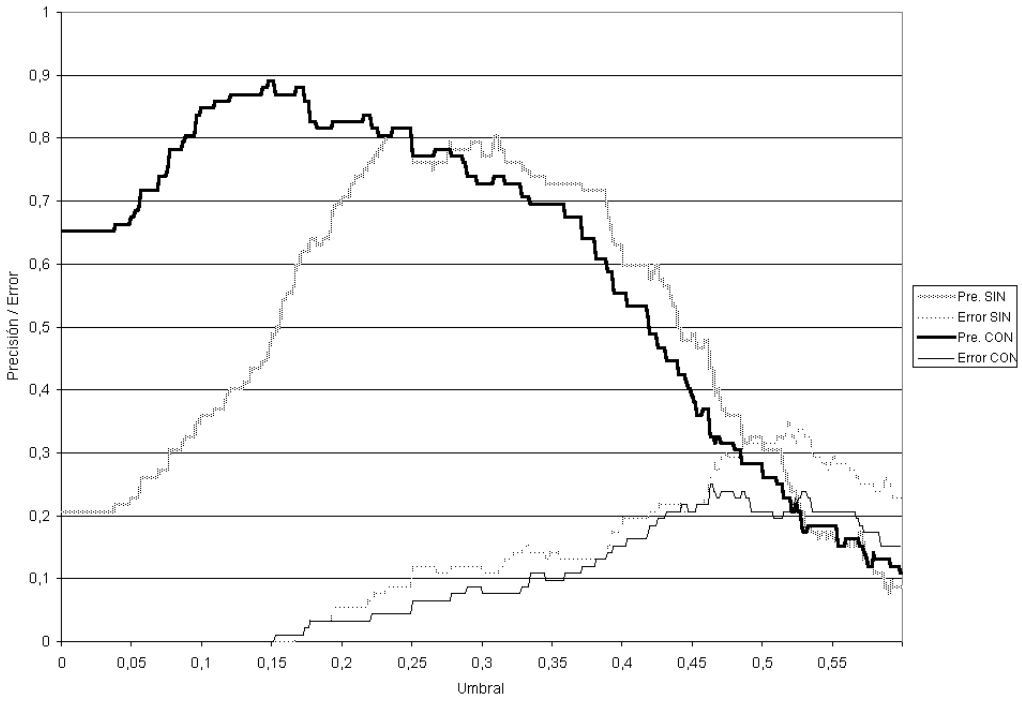

Figura 6.3: Precisión y Error del fichero A sin y con expansión de abreviaturas (MSC)

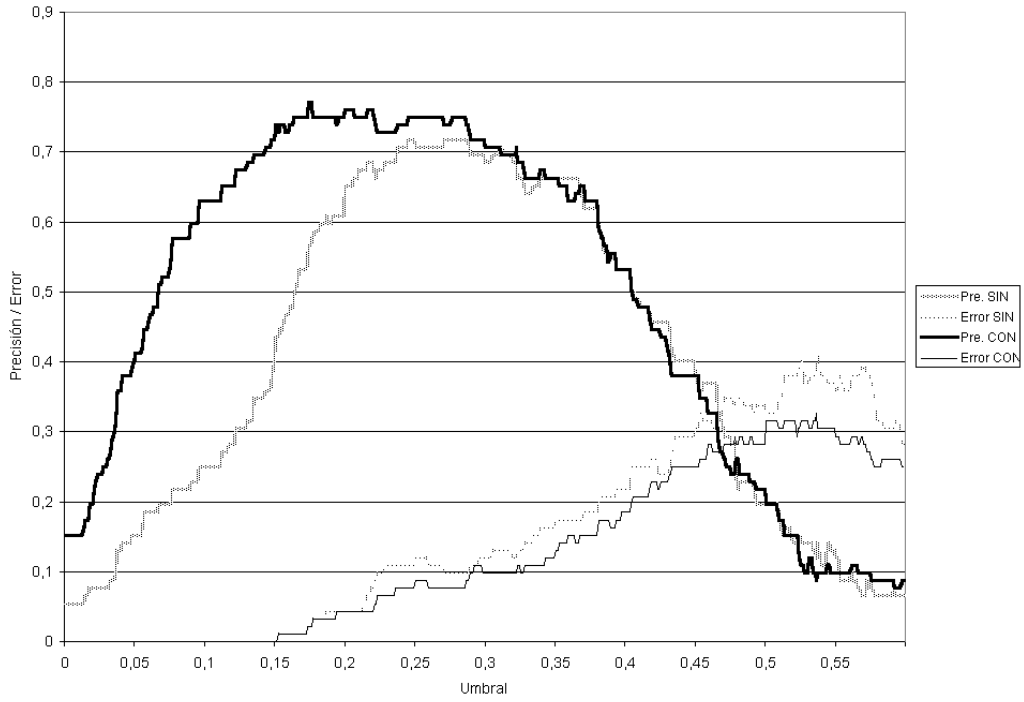

Figura 6.4: Precisión y Error del fichero B sin y con expansión de abreviaturas (MSC)

encima del 75% para todos los valores del umbral.

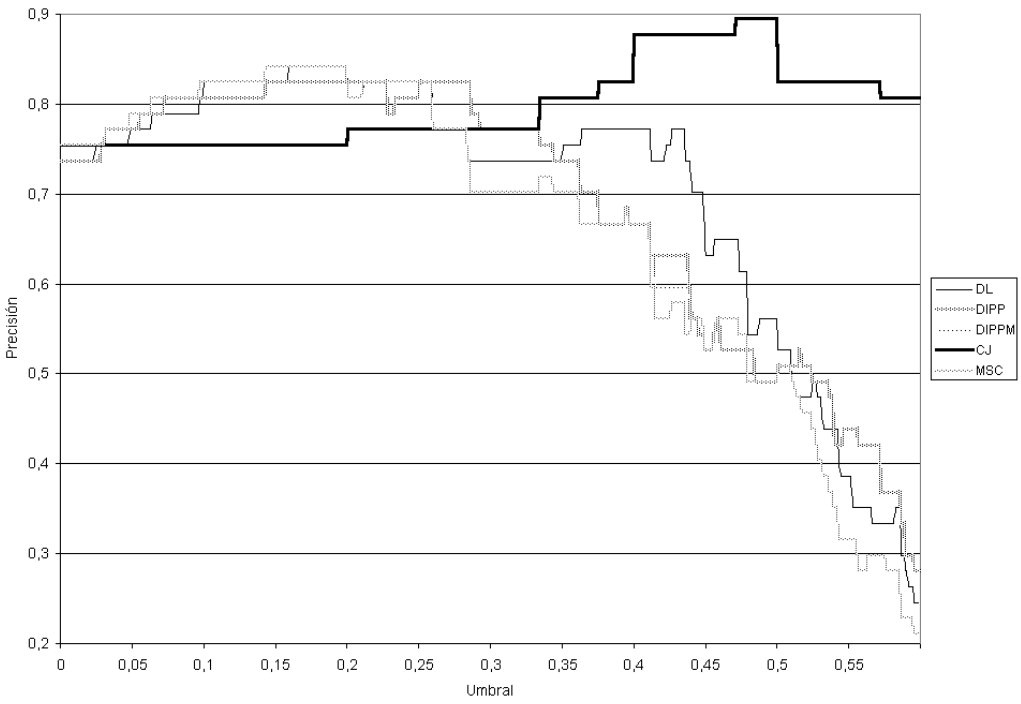

Figura 6.5: Precisión del fichero C sin expansión de abreviaturas (con las distintas medidas de similitud)

6.6 Comparación aproximación 1 y 2 y conclusiones

En la tabla 6.7 mostramos resumida la mejor precisión con su correspondiente error que se obtiene en las dos aproximaciones para los cuatro ficheros de prueba. La segunda aproximación obtiene una mejor precisión en todos los casos. El error disminuye o se mantiene en torno a un valor similar.

El número de agrupamientos que se obtiene depende del umbral que se emplee. Un valor pequeño produce un agrupamiento conservador, con un número grande de pequeños agrupamientos. Un valor grande (agrupamiento agresivo) produce un número pequeño de agrupamientos grandes. A la vista de las gráficas que mostramos y otros datos obtenidos en nuestro estudio, proponemos el empleo de un umbral entre 0,1 y 0,25 con expansión de

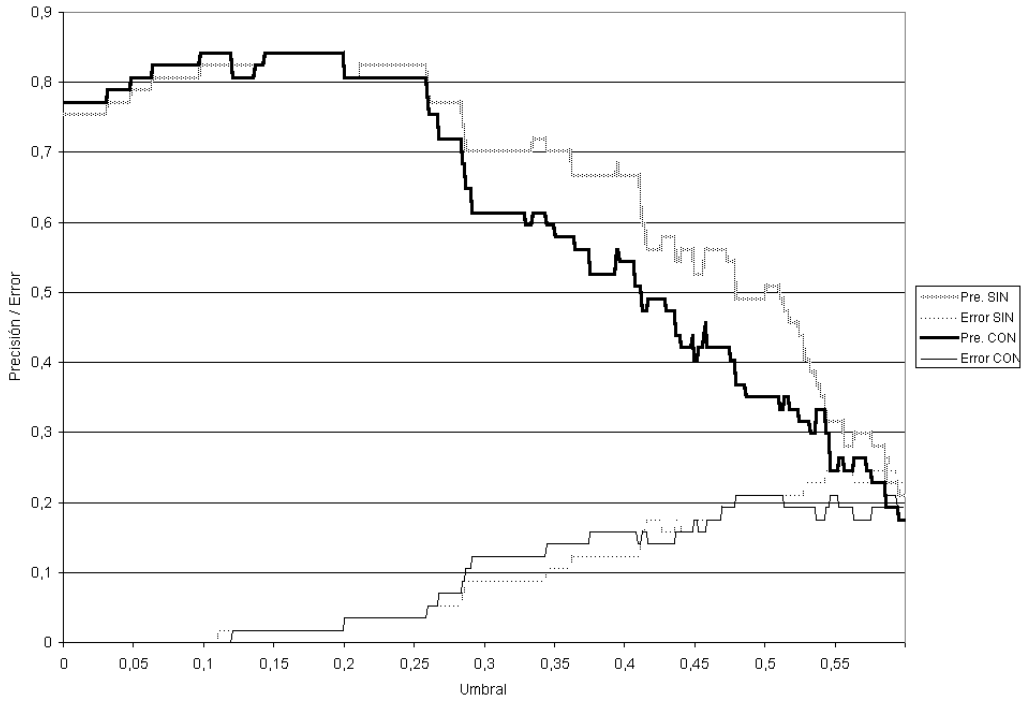

Figura 6.6: Precisión y Error del fichero C sin y con expansión de abreviaturas (MSC)

Fichero		**Aproximación 1**		**Aproximación 2**	
		Precisión (%)	**Error** (%)	**Precisión** (%)	**Error** (%)
A	SIN	70,7	7,6	81,5	8,6
	CON	84,8	0	89,1	0
B	SIN	67,4	8,7	71,7	9,7
	CON	72,8	6,5	77,1	2,1
C	SIN	85,9	1,7	89,4	1,7
	CON	84,2	1,7	87,7	1,7
D	SIN	43,8	16,8	53,0	15,9
	CON	67,7	6,2	76,1	2,6

Tabla 6.7: Comparación resultados aproximación 1/2

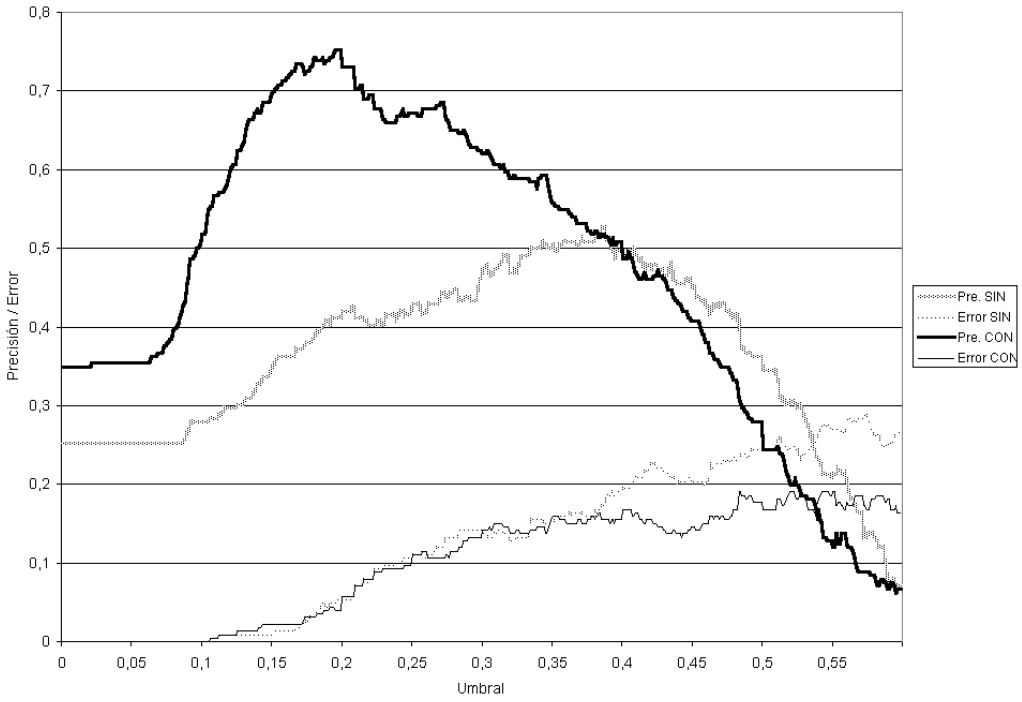

Figura 6.7: Precisión y Error del fichero D sin y con expansión de abreviaturas (MSC)

abreviaturas y un valor superior sin expansión.

Capítulo 7

Conclusiones, aportaciones y trabajos en desarrollo

En este último capítulo presentamos las principales conclusiones a las que hemos llegado, las aportaciones que hemos realizado con nuestro estudio y los trabajos que actualmente estamos desarrollando (o tenemos pensado desarrollar) para mejorar o ampliar los dos métodos propuestos.

7.1 Conclusiones

En este trabajo hemos estudiado el problema de la inconsistencia de valores en bases de datos: un mismo término (concepto) representado por distintos valores (cadenas). A partir de la revisión de varias bases de datos, hemos detallado las posibles causas que producen diferentes valores.

Los dos métodos que hemos propuesto los hemos evaluado con cuatro ficheros. Tres de ellos contienen información en español y el cuarto en inglés.

El segundo método obtiene mejores resultados (mejor precisión con un menor error) en todos los ficheros. Además, tiene la ventaja de que sólo emplea un parámetro, en vez de los cuatro parámetros que tiene el primer método. En todas la pruebas realizadas hemos comprobado como la expansión de las abreviaturas ayuda a mejorar la calidad de los agrupamientos obtenidos.

El problema de qué valor elegir para el parámetro depende de cada fichero. Si se desarrollara una herramienta real a partir de nuestro trabajo, tendría que existir una opción para especificar el nivel de agrupamiento deseado: desde muy débil (umbral próximo a cero) hasta muy fuerte (umbral próximo a uno).

Queda pendiente un problema importante: la escalabilidad y la precisión.

Las pruebas se han realizado con unos ficheros de prueba pequeños. Una base de datos real puede contener millones de registros. La aplicación de los métodos propuestos requeriría varios días o semanas de procesamiento. El cálculo de la DL tiene un coste cuadrático tal como la hemos implementado. Estamos trabajando en el uso de algoritmos que poseen menor coste temporal y espacial, como es el de (Myers, 1986).

Finalmente, a partir del método propuesto en la primera aproximación, hemos publicado una comunicación en el congreso *"The International Joint Conference IBERAMIA'2000 SBIA'2000"* (Luján-Mora & Palomar, 2000). Los resultados de la segunda aproximación pensamos presentarlos en otra comunicación que estamos preparando.

7.2 Aportaciones

Las principales aportaciones de nuestro trabajo son:

- Aplicación al español
  No conocemos ningún otro trabajo similar que se haya realizado y que estudie el problema de la inconsistencia de valores en bases de datos en español.

- Independencia del idioma
  Aunque nos hemos centrado principalmente en resolver el problema de la inconsistencia en bases de datos en español, también hemos probado nuestro método con una base de datos en inglés. Nuestro método es independiente del idioma (siempre que el idioma se base en el alfabeto occidental).

- Independencia del dominio
  Algunos métodos de deduplicación se centran en dominios muy concretos (principalmente, en bases de datos con nombres y direcciones de correo). Nuestro método es independiente del dominio.

- IC e ICF
  Hemos desarrollado dos métricas que permiten cuantificar la complejidad de los agrupamientos.

- DLON y DIT
  Hemos empleado una serie de distancias entre cadenas sencillas de calcular y con un coste temporal despreciable que permiten acelerar el proceso de agrupamiento.

7.3 Trabajos en desarrollo y futuros

Actualmente, estamos trabajando en una serie de mejoras:

- Multilingualidad
  Pensamos aplicar el método propuesto a bases de datos que almacenen valores en distintos idiomas. Para ello, utilizaremos diccionarios multilingües / redes semánticas como EuroWordNet (Vossen, 1997).

- Bigramas y trigramas
  Uso de bigramas y trigramas como medida de similitud (Freund & Willett, 1982; Angell *et al.*, 1983; Rapp, 1997) u otras medidas de similitud entre cadenas.

- Algoritmos de agrupamiento
  Uso de otros algoritmos de agrupamiento más eficientes. En (DSS LAB, 1999) existe una excelente relación de algoritmo de agrupamiento y clasificación (*clustering and classification*).

- Expansión automática de las abreviaturas
  Si se consideran las abreviaturas como prefijos de las cadenas a las que se expanden, se puede realizar una búsqueda sobre todas las cadenas para localizar a cuales se puede expandir. De este modo se evita tener que indicar las abreviaturas.

- Eliminación de palabras sin contenido semántico (*stop words*)
  Los artículos, preposiciones, conjunciones, etc. no añaden información semántica. En el proceso de agrupamiento, se pueden esperar mejoras si se eliminan todas estas palabras.

- Aplicación en Internet
  Desarrollo de una aplicación demostrativa del método propuesto que se pueda acceder a ella a través de Internet.

- Artículo
  Estamos preparando un artículo en el que presentamos el segundo método propuesto junto con los resultados obtenidos al aplicarlo.

Apéndice A

Direcciones en Internet

En este apéndice incluimos las direcciones de Internet (http://) de las compañías y organizaciones citadas en el trabajo.

- *Evolutionary Technologies International, Inc.*: `www.evtech.com`
- *Electronic Digital Documents, Inc.*: `www.npsa.com/edd`
- *European Space Agency*: `www.esa.int`
- *GNU*: `www.gnu.org`
- *Group 1 Software*: `www.g1.com`
- *Informix Corporation*: `www.informix.com`
- *InfoTech Ltd*: `www.infotech.ie`
- *National Archives and Records Administration*: `www.nara.gov`
- *QAS Systems Ltd*: `www.qas.com`
- *RedIRIS*: `www.rediris.es`
- *Trillium Software*: `www.trilliumsoft.com`
- *Universidad de Alicante*: `www.ua.es`
- *Universidad de Virginia*: `www.virginia.edu`
- *Vality Technology*: `www.vality.com`

Apéndice B

Ficheros de prueba

Incluimos en este apéndice fragmentos de los ficheros de prueba. Para cada uno de ellos (A, B, C y D), se incluyen los seis primeros agrupamientos (por orden alfabético) y los seis últimos. Los agrupamientos[1] que mostramos están compuestos por todas las cadenas que los forman (algunos agrupamientos están formados por una única cadenas, mientras que otros poseen diez o más cadenas).

B.1 Fichero A

```
Agrupación Astronómica Universitaria de Alicante - AAUA

Área de Alemán del Departamento de Filologías Integradas
Área de Alemán del Depto.  de Filologías Integradas
Área de Alemán del Dpto.  de Filologías Integradas

Área de Estudios Árabes e Islámicos del Departamento de
Filologías Integradas
Área de Estudios Árabes e Islámicos del Depto.  de Filologías
Integradas
Área de Estudios Árabes e Islámicos del Dpto.  de Filologías
Integradas

Área de Filología Francesa del Departamento de Filologías
Integradas
Área de Filología Francesa del Depto.  de Filologías Integradas
Área de Filología Francesa del Dpto.  de Filologías Integradas
```

[1] Los agrupamientos se han construido a mano. Estos agrupamientos son los que se han empleado para evaluar la precisión de los métodos propuestos.

Asociación Universitaria de Profesores de Didáctica de las Ciencias Sociales

Cátedra Rafael Altamira

⋮

Relaciones Internacionales de la Universidad de Alicante, S.A.

Secretariado de las Sedes Universitarias y Cursos Especiales
Secretariat de les Seus Universitàries i Cursos Especials

Secretariado de Promoción del Valenciano
Secretariat de Promoció del Valencià

Taller de Imagen, S.A.
Taller de Imagen, Sociedad Anónima
Taller de Imagen de la Universidad de Alicante

Unidad de Registro Sismológico - Universidad de Alicante
Unidad de Registro Sismológico (Universidad de Alicante)
URSUA Unidad de Registro Sismológico (Universidad de Alicante)
Unidad de Registro Sísmico-Universidad de Alicante

Universidad de Alicante
Universitat d'Alacant

B.2 Fichero B

Agrupación Astronómica Universitaria de Alicant - AAUA
Agrupación Astronómica Universitaria de Alicante - AAUA
Agrupación Astronómica Universitaria de Alicante - AUAA
Agrupación Astronómica Universitaria de Aljcante - AAUA
Agrupación Astronómica Univursitaria de Alicante - AAUA

Área de Alemán del Departamehto de Filologías Integradas
Área de Alemán del Departamento de Filologías Integradas
Área de Alemán del Departamento de Filologías Integradls
Área de Alemán del Departamento de Filologías Intgeradas
Área de Alemán del Departamento de FilologíassIntegradas
Área de Alemán del Departamento de Kilologías Integradas
Área de Alemán del Departmaento de Filologías Integradas
Área dq Alemán del Departamento de Filologías Integradas
Área de Alemán del Depto. de Filologías Integradas
Área de Alemán del Dpto. de Filologías Integradas

Área de Estudios rabes e Islámicos del Departamento de Filologías Integradas
Área de Estudios Áiabes e Islámicos del Departamento de Filologías Integradas
Área de Estudios Árabes Islámicos del Departamento de Filologías Integradas
Área de Estudios Árabes e Islámicos del Departamento dd Filologías Integradas
Área de Estudios Árabes e Islámicos del Departamento de Fhlologías Integradas
Área de Estudios Árabes e Islámicos del Departamento de Filllogías Integradas
Área de Estudios Árabes e Islámicos del Departamento de Filologíac Integradas
Área de Estudios Árabes e Islámicos del Departamento de Filologíae Integradas
Área de Estudios Árabes e Islámicos del Departamento de Filologías Integradas
Área de Estudios Árabes e Islámicos del Departamento de Filologíks Integradas
Área de Estudios Árabes e Islámicos del Departametto de Filologías Integradas
Área de Estudios Árabes e Islámicos del Depar{amento de Filologías Integradas
Área de Estudios Árabes e Islámicos del De{artamento de Filologías Integradas
Área de EstudiosÁÁrabes e Islámicos del Departamento de Filologías Integradas
Área de Estudios Árabes e Islámicos de Depto. de Filologías Integradas
Área de Estudios Árabes e Islámicos del Depto. de Filologías Integradas
Área de Estudios Árabes e Islámicossdel Depto. de Filologías Integradas
Área de Estudios Ártbes e Islámicos del Depto. de Filologías Integradas
Área de Estudios Árabes e Islámicos del Dpto. de Filologías Integradas

Área de Filelogía Francesa del Departamento de Filologías Integradas

```
Área de Filologaí Francesa del Departamento de Filologías
Integradas
Área de Filología Francesa del Departamento de Filologías
Integradas
Área de Filología Francesa del Departamento de Filologías
Sntegradas
Área de Filología Francesa del Departamento de Tilologías
Integradas
Área de Filología Francesa del Departamesto de Filologías
Integradas
Área de Filología Frzncesa del Departamento de Filologías
Integradas
Área de Yilología Francrsu del Departamento de Filologías
Integradas
Área de iilología Francesa del Departamento de Filologías
Integradas
Área de Filología Francesa del Depto.  de Filologías Integradas
Área de Filología Francesa del Dpto.  de Filologías Integradas

Asociación Universitaria de Profeeores de Didáctica de las
Ciencias Sociales
Asociación Universitaria de Profesores de Didáctica de las
Ciencias Soccales
Asociación Universitaria de Profesores de Didáctica de las
Ciencias Sociales
Asociación Universitaria de Profesores de Didáctica deelas
Ciencias Sociales
Asociación Universitaria de Profesores ee Didáctica de las
Ciencias Sociales
Asociación Universitaria ze Profesores de Didáctica de las
Ciencias Sociales

Cátdera Rafael Altamira
Cátedar Rafael Altamira
Cátedda Rafael Atlamira
Cátedr aRafael Altamira
Cátedra Rafael Altamira
Cátedra Rafael Altamnra
Cátedra Rafael Altamrra
Cátedra Rafael Aytamira
Cátedra Rafael A{tamira
```

Cátedra Raffee Altaimra
Cátedra Rahael Altamira
Cátedra Refael Altamira
Cátedra aafael Altamira
CátedraR afael Altamira
Cáted{a Rafael Altamira

⋮

Relaciones Internacinnales de la Universidad de Alicante, S.A.
Relaciones Internacionales de la niversidad de Alicante, S.A.
Relaciones Internacionales de la Unirersidad de Alicante, S.A.
Relaciones Internacionales de la Universiadd de Alicante, Y.A.
Relaciones Internacionales de la Universidad de Aliaante, S.A.
Relaciones Internacionales de la Universidad de Alicante, .A.
Relaciones Internacionales de la Universidad de Alicante, ..A.
Relaciones Internacionales de la Universidad de Alicante, S.A.
Relaciones Internacionales de la Universidad de Alicante, S.AA
Relaciones Internacionales de la Universidad de Aligante, S.A.
Relaciones Internacionales de la Universidad de Allcante, S.A.
Relaciones Internacionales de la nniversidad de Alicante, S.A.
Relaciones Internacionalesdde la Universidad de Alicante, S.A.
Relaciones Internacionalss de la Universidad de Alicante, S.A.
Relaciones Internafionales de la Universidad de Alicante, S.A.
Relacione{ Internacionales de la Universidad de Alicante, S.A.

Secertariado de las Sedes Universitarias y Cursos Especiales
Secretariada de las Sedes Universitarias y Cursos Especiales
Secretariado de las Sedes Universitarias y Cursos Especiales
Secretariado de las Sedes Universitariasyy Cursos Especiales
Secretariado de lqs Sedes Universitarias y Cursos Especiales
Secretariado deelas Sedes Universitarias y Cursos Especiales
Secretarivdo de las Sedes Universitarias y Cursos Especiales
Secretariat de les Seus Univeristàries i Cursos Especials
Secretariat de les Seus Universitàries i Cursos Especials
Secret{riat de les Seus Universitàries i Cursos Especials

Secreeariado de Promoción del Valenciano
Secretaraido de Promoción dec Valenciano
Secretariado de Poomoción del Valenciano
Secretariado de Promoción del Valenciano

Secretariado de Promoción dey Valenciano
Secretariado de Prrmoción del Valenciano
Secretariat de Promació del Valencià
Secretariat de Promoció del Valencià
Secretariat de Promoció del Vclencià
Secretariat de Promocióddel Valencià

Taller e Imagen, S.A.
Taller de Iaagen, S.A.
Taller de Imagen, S.A.
Taller de mmagen, S.A.
Taller dj Imagen, S.A.
Taller de Imagen, Sociedad Anónima
Taller de Imagen de la Universidad de Alicante

URSUA Unidad dc Registro Sismológico (Universidad de Alicante)
URSUA Unidad de Registro Sismológico (Universidad de Alicante)
Unidad d eRegistro Sismológico - Universidad de Alicante
Unidad de Regintro Sismológico - Universidad de Alicante
Unidad de Registko Sismológico - Universidad de Alicante
Unidad de Registro Sismológico -Universidad de Alicante
Unidad de Registro Sismológico - niversidad de Alicante
Unidad de Registro Sismológico - Universidad d eAlicante
Unidad de Registro Sismológico - Universidad de Alicante
Unidad de Registro Sismológico - Universidad de Amicante
Unidad de Resistro Sismológico - Universidad de Alicante
Unidad de eRgistro Sismológico - Universidad de Alicante
Unidad de Registro Sismológico (Universidad de Alicante)
Unidad de Registro Sismoólgico (Universidad de Alicante)
Unidad de Registro Sísmico-Universidad de Alicante
Unidad de Registro Sísmico-Universidad deA licante

Universidad dd Alicante
Universidad de Aliaante
Universidad de Alicante
Universidad ed Alicante
Universimad de Alicante
Universitat d'Alacant
Universitta d'Alacknt
Univyrsitat d'Alacant

B.3 Fichero C

```
Analytical Mechanics Associates, Inc., Hampton

Anser Analytic Services, Inc., Arlington

BDM Corporation, McLean

Beers Associates, Inc., Reston

Bionetics Corporation, Hampton

College of William and Mary, Williamsburg
College of William and Mary, Williamsburgh

   ⋮

Virginia, Medical College, Richmond

Virginia Military Institute, Lexington

Virginia Polytechnic Institute and State University, Blacksburg
Virginia, Polytechnic Institute and State University,
Blacksburg
Virginia Polytechnic Institute and State University
Polytechnic Institute & State University, Blacksburg
Virginia Polytechnic and State University, Blacksburg
Virginia Polytechnic Institute and State University,
Charlottesville
Virginia Polytechnic Institute and State Univesity, Blacksburg
Virginia Polytechic Institute and State University, Blacksburg
Virginia Polytechnic Institute

Virginia University, Charlottesville
Virginia, University, Charlottesville
University of Virginia
University of Virginia, Virginia
University of Virginia, Charlottsville
University of Virgina, Charlottesville
University of Virginia, Charlottesville
University of Virginia, Charlottesville, Virginia
University of Virginia, Charlottesvill
```

Virgina University, Charlottesville
Virgina, University, Charlottesville
Virginia University
Virginia, University
Leander McCormick Observatory, Charlottesville
Leander J. McCormick Observatory, Charlottesville
University of VA., Charlottesville

W. J. Schafer Associates, Inc., Hampton, Virginia

West Virginia University, Morgantown
West Virginia University, Morgantown, W.

B.4 Fichero D

ASOC. HISPANOAMERICANA DE CENTROS DE INV. Y EMPRESAS DE TELECOM (AHCIET)
Asociación Hispanoamericana de Centros de Investigación y Empresas de Telecomunicaciones
Asociacion Hispanoamericana de Centros de Investigacion y Empresas de Telecomunicaciones
Asociación Hisp. de Centros de Investig. y Empresas de Telec.

ASOC. INT. PARA LA INV. OCEANOGRÁFICA Y MEDIO AMBIENTE (AINCO)*
Asociación Internacional para la Investigación Oceanográfica y del Medio Ambiente
Asociacion Int. para la Investigacion Oceanografica y del Medio Ambiente
Asoc. Internacional para la Investigación Ocean. y del Medio Amb.

ASOC. DE I+D EN LA INDUSTRIA DEL MUEBLE Y AFINES (AIDIMA)
Asociación de Investigación y Desarrollo en la Industria del Mueble y Afines
Asociacion de Investigacion y Desarrollo en la Industria del Mueble y Afines (AIDIMA)
Asociación de Invest. y Des. en la Industria del Mueble y Afines

ASOC. DE INV. IND. DE LA MÁQUINA-HERRAMIENTA (INVEMA)
Asociación de Investigación Industrial de la Máquina-Herramienta

Fundación de Investigación de la Máquina-Herramienta
Asociacion de Invest. Ind. de la Máquina y la Herramienta

ASOC. DE INV. TEC. DE EQUIPOS MINEROS (AITEMIN)*
Asociación de investigación tecnológica de equipos mineros
Asociacion de invest. tec. de equipos mineros
Asoc. de investig. tecnológica de equipos mineros

ASOC. DE INV. TÉCNICA DE LA INDUSTRIA PAPELERA ESPAÑOLA (IPE)
Asociación de Investigación Técnica de la Industria Papelera Española
Fundación de Invest. Técnica de la Ind. Papelera Española (IPE)
Asoc. de Invest. Técnica de la Ind. Papelera Española

⋮

UNIVERSITAT JAUME I (UJI)
Universitat Jaume I
Universidad Jaime I
Uni. Jaime I

UNIVERSITAT OBERTA DE CATALUNYA (UOC)
Universitat Oberta de Catalunya
Universidad Abierta de Cataluña

UNIVERSITAT POLITÉCNICA DE CATALUNYA (UPC)
Universidad Politécnica de Cataluña
Uni. Pol. de Cataluña

UNIVERSITAT POLITECNICA DE VALENCIA (UPV)
Universidad Politécnica de Valencia
Uni. Politécnica de Valencia
Univ. Pol. de Valencia

UNIVERSITAT POMPEU FABRA (UPF)

UNIVERSITAT RAMON LLULL (URL)
Universidad Ramón Llull

Referencias

Adelman, Sid, & Moss, Larissa T. 1999. Practical Guidelines for Data Acquisition / Data Cleansing. *The Navigator*.

Angell, R.C., Freund, G.E., & Willett, P. 1983. Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, **19**(4), 255–261.

Apertus. *Meeting the Data Integration Challenge*. Internet: http://www.techguide.com/. Visitada el 30/09/2000.

Batini, C., Lenzerini, M., & Navathe, S. 1986. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, **18**(4), 323–364.

Bitton, Dina, & DeWitt, David J. 1983. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, **8**(2), 255–265.

Bort, Julie. 1997 (Mayo). *The wiser, gentler data warehouse*. Internet: http://www.sunworld.com/sunworldonline/swol-05-1997/swol-05-datawarehouse_p.html. Visitada el 21/06/2000.

Damerau, F.J. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM*, **7**(3), 171–176.

Database & Consulting. *Mantenimiento y normalización de Bases de Datos*. Internet: http://www.database-consulting.com/mnb.htm. Visitada el 24/10/2000.

Díaz, M., Pérez, J., & Santana, O. 1993 (Abril 12-16). Distancia Dependiente de la Subsecuencia Común más Larga entre Cadenas de Caracteres. *Páginas 117–123 de: Anales de las II Jornadas de Ingeniería de Sistemas Informáticos y de Computación*.

DSS LAB. 1999. *Data Mining Information Sources.* Internet: http://www.dsslab.com/resources/DMINFO.htm. Visitada el 10/11/2000.

Farnstrom, Fredrik, Lewis, James, & Elkan, Charles. 2000. Scalability for Clustering Algorithms Revisited. *SIGKDD Explorations. Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, **2**(1), 51–57.

French, James C., Powell, Allison L., & Schulman, Eric. 1997a. Applications of Approximate Word Matching in Information Retrieval. *Páginas 9–15 de:* Golshani, Forouzan, & Makki, Kia (eds), *Proceedings of the Sixth International Conference on Information and Knowledge Management (CIKM 1997).* Las Vegas (EE.UU.): ACM Press.

French, James C., Powell, Allison L., Schulman, Eric, & Pfaltz, John L. 1997b. Automating the Construction of Authority Files in Digital Libraries: A Case Study. *Páginas 55–71 de:* Peters, Carol, & Thanos, Costantino (eds), *Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries (ECDL 1997).* Lecture Notes in Computer Science, vol. 1324. Pisa (Italia): Springer-Verlag.

Freund, G.E., & Willett, P. 1982. Online identification of word variants and arbitrary truncation searching using a string similarity measure. *Information Technology: Research and Development*, **1**, 177–187.

Galhardas, Helena. 1999 (Noviembre). *Data Cleaning and Integration.* Internet: http://caravel.inria.fr/∼galharda/cleaning.html. Visitada el 03/09/2000.

Giralpost Publicidad y Marketing Directo. *Giralpost: Servicios Informáticos, Listados.* Internet: http://www.arrakis.es/∼daga/giralpost/-servinf.htm. Visitada el 24/10/2000.

Hall, Curt. 1998. Data Integrity and Cleansing: Tools and Techniques. *Data Management Strategies (ahora Business Intelligence Advisor)*, Noviembre.

Hartigan, John A. 1975. *Clustering Algorithms.* A Wiley Publication in Applied Statistics. Nueva York (EE.UU.): John Wiley & Sons.

Hernández, Mauricio Antonio, & Stolfo, Salvatore J. 1995. The Merge/Purge Problem for Large Databases. *Páginas 127–138 de:* Carey, Michael J., & Schneider, Donovan A. (eds), *Proceedings of the 1995 ACM SIGMOD*

International Conference on Management of Data. San Jose (EE.UU.): ACM Press. También publicado como SIGMOD Record 24(2), Junio 1995.

Hernández, Mauricio Antonio, & Stolfo, Salvatore J. 1998. Real-world data is dirty: Data cleansing and the merge/purge problem. *Journal of Data Mining and Knowledge Discovery*, **2**(1), 9–37.

Hirschberg, Daniel S. 1997. Serial Computations of Levenshtein Distances. *Páginas 123–141 de:* Apostolico, Alberto, & Galil, Zvi (eds), *Pattern Matching Algorithms*. Nueva York (EE.UU.): Oxford University Press.

Huang, Kuan-Tsae, Lee, Yang W., & Wang, Richard Y. 1998. *Quality Information and Knowledge Management*. 1 edn. Prentice Hall.

Kukich, Karen. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, **24**(4), 377–439.

Levenshtein, Vladimir I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, **10**, 707–710.

Luján-Mora, Sergio, & Palomar, Manuel. 2000 (Noviembre 19-22). Clustering of Similar Values, in Spanish, for the Improvement of Search Systems. *En: International Joint Conference IBERAMIA-SBIA 2000*. (Aceptado y pendiente de publicación).

MacKenzie, David, Eggert, Paul, & Stallman, Richard. 1993 (Septiembre). *Comparing and Merging Files*. Internet: http://www.gnu.org/manual/-diffutils/html_chapter/diff_1.html. Visitada el 13/09/2000.

Monge, Alvaro E., & Elkan, Charles P. 1996. The field matching problem: Algorithms and applications. *Páginas 267–270 de:* Simoudis, Evangelos, Han, Jiawei, & Fayyad, Usama M. (eds), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD 1996)*. Portland (EE.UU.): AAAI Press.

Monge, Alvaro E., & Elkan, Charles P. 1997 (Mayo 11). An efficient domain-independent algorithm for detecting approximately duplicate database records. *Páginas 23–29 de: SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97)*.

Morgan, H.L. 1970. Spelling Correction in Systems Programs. *Communications of the ACM*, **13**(2), 90–94.

Moss, Larissa T. 1998. Data Cleansing: A Dichotomy of Data Warehousing? *DM Review*, **8**(2).

Motro, Amihai, & Rakov, Igor. 1996 (Octubre). Estimating the Quality of Data in Relational Databases. *Páginas 94–106 de:* Wang, Richard Y. (ed), *Proceedings of the 1996 Conference on Information Quality.*

Motro, Amihai, & Rakov, Igor. 1998. Estimating the Quality of Databases. *Páginas 298–307 de:* Andreasen, T., Christiansen, H., & Larsen, H.L. (eds), *Proceedings of FQAS 98: Third International Conference on Flexible Query Answering Systems.* Lecture Notes in Artificial Intelligence, vol. 1495. Roskilde (Dinamarca): Springer-Verlag.

Myers, Eugene W. 1986. An O(ND) Difference Algorithm and Its Variations. *Algorithmica*, **1**(2), 251–266.

National Archives and Records Administration. 1997 (Abril). *The Soundex Machine.* Internet: http://www.nara.gov/genealogy/soundex/-soundex.html. Visitada el 24/09/2000.

National Archives and Records Administration. 2000 (Febrero). *The Soundex Indexing System.* Internet: http://www.nara.gov/genealogy/-coding.html. Visitada el 24/09/2000.

Omikron. *Del fichero de direcciones a la base de datos de marketing: el software de deduplicaciones.* Internet: http://www.omikron.net/Espanol/-index.html. Visitada el 07/09/2000.

O'Neill, Edward T., & Vizine-Goetz, Diane. 1988. Quality Control in Online Databases. *Páginas 125–156 de:* Williams, Martha E. (ed), *Annual Review of Information Science and Technology (ARIST)*, vol. 23. Nueva York (EE.UU.): Elsevier Science Publishers.

O'Neill, Edward T., & Vizine-Goetz, Diane. 1989. The Impact of Spelling Errors on Databases and Indexes. *Páginas 313–320 de:* Nixon, Carol, & Padgett, Lauree (eds), *10th National Online Meeting Proceedings.* Nueva York (EE.UU.): Learned Information Inc.

Pollock, Joseph J., & Zamora, Antonio. 1984a. Automatic Spelling Correction in Scientific and Scholarly Text. *Communications of the ACM*, **27**(4), 358–368.

Pollock, Joseph J., & Zamora, Antonio. 1984b. System Design for Detection and Correction of Spelling Errors in Scientific and Scholarly Text. *Journal of the American Society for Information Science*, **35**(2), 104–109.

Quass, Dallan. 1999. *A Framework for Research in Data Cleaning.* Internet: http://caravel.inria.fr/~galharda/quass.cleaningII.ps. Visitada el 03/09/2000.

Rapp, Reinhard. 1997. Text-Detector: Fault-tolerant Retrieval Made Simple. *c't*, **4**, 386.

Real Academia Española. 1992. *Diccionario de la lengua española / Real Academia Española.* 21 edn. Madrid (España): Espasa-Calpe.

RedIRIS. 2000 (Septiembre). *RedIRIS - Centros afiliados a RedIRIS.* Internet: http://www.rediris.es/recursos/centros/afiliacion.es.html. Visitada el 18/09/2000.

Rijsbergen, Cornelis Joost Van. 1979. *Information Retrieval.* 2 edn. Londres (Reino Unido): Butterworths.

Ryzin, J. Van (ed). 1977. *Classification and Clustering.* Mathematics Research Center, The University of Wisconsin at Madison, vol. 37. Nueva York (EE.UU.): Academic Press, Inc.

Salton, Gerard, & McGill, Michael J. 1983. *Introduction to Modern Information Retrieval.* McGraw Hill.

Santana, O., Díaz, M., Mayor, O., & Pérez, J. 1987 (Noviembre). Esquemas y Estructura para la Búsqueda de las Palabras más Similares a una Dada. *Páginas 1169–1189 de: XIII Conferencia Latinoamericana de Informática*, vol. II.

Santana, O., Díaz, M., Duque, J.D., & Rodríguez, G. 1989 (Junio 10-14). Estructuración de las Componentes de la Distancia Invariante Trasposicional, DIT, con Compartición de la Zona No-Discriminante en la Búsqueda de las Cadenas más Similares. *Páginas 335–341 de: XV Conferencia Latinoamericana de Informática*, vol. II.

Shaffer, L., & Hardwick, J. 1968. Typing Performance as a Function of Text. *Quarterly Journal of Experimental Psychology*, **20**, 360–369.

Smith, Temple F., & Waterman, Michael S. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology*, **147**, 195–197.

Soria González, Inocencia. *La organización de la información, los lenguajes documentales y la normalización.* Internet: http://www.csic.es/cbic/-intrared/publicaciones/soria.htm. Visitada el 24/10/2000.

Trillium Software System. *A Practical Guide to Achieving Enterprise Data Quality*. Internet: http://www.techguide.com/. Visitada el 30/09/2000.

Universidad de Alicante. 2000 (Septiembre). *Otros Webs de la UA*. Internet: http://www.ua.es/es/internet/otroswwwua.html. Visitada el 18/09/2000.

University of Tennessee Computing Center. 1993. *VMScluster User's Guide*. Internet: http://www.cas.utk.edu/utcc/user_services/users_guides/-OpenVMS_guide/vms-OPS5.html. Visitada el 06/10/2000.

Vossen, Piek. 1997. EuroWordNet: a multilingual database for information retrieval. *Páginas 85–94 de:* Sheridan, Páraic (ed), *Third DELOS Workshop on Cross-Language Information Retrieval*. ERCIM Workshop Proceedings, vol. 97-W003. Zurich (Suiza): European Research Consortium for Informatics and Mathematics.

Zobel, Justin, & Dart, Philip W. 1995. Finding approximate matches in large lexicons. *Software-Practice and Experience*, **25**(3), 331–345.

Zobel, Justin, & Dart, Philip W. 1996. Phonetic String Matching: Lessons from Information Retrieval. *Páginas 166–172 de:* Frei, Hans-Peter, Harman, Donna, Schäble, Peter, & Wilkinson, Ross (eds), *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*. Zurich (Suiza): ACM Press.

NOTA: Muchos de los documentos citados es este trabajo sólo están disponibles en formato electrónico a través de Internet. En estos casos, es difícil facilitar una referencia completa (autor, fecha, etc.) y duradera, debido a la "volatilidad" de la información disponible en Internet. Incluimos las URLs de los documentos citados, junto con la fecha de la última actualización (si estaba disponible cuando fue consultado). En todo caso, también incluimos la fecha en que obtuvimos el documento.

Si en una URL aparece un guión (-) justo después de una barra inclinada (/), no se debe de considerar como parte de ella (se incluye como divisor de palabra).

Índice de Materias