E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

ESTUDIO SOBRE LA DETECCION DE DUPLICADOS EN ORIGENES DE DATOS HETEROGENEOS

Grado en Ingeniería Informática

Trabajo Fin de Grado

Ion Gorostizu Albeniz

Jose R. Gonzalez de Mendivil

Pamplona, 26 de Junio de 2014

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

Las aplicaciones de manipulación de datos realizan tareas sobre datos extraídos de muy diferentes fuentes, cada una de ellas con sus propias particularidades como, por ejemplo: errores tipográficos, diferentes cantidades de información o atributos distintos. La necesidad de trabajar con estos datos tan heterogéneos suele provocar el problema de tener que identificar distintas filas como el mismo objeto en la vida real, ya que cada origen de datos representa ese objeto de forma distinta.

"Record Linkage" (o *"Duplicate Detection"*) es el término referido a la tarea de encontrar filas que representan una misma entidad entre distintas fuentes, normalmente usando las denominadas "String Similarity Metrics". Una "String Metric" es una medida que establece la similitud o disimilitud entre dos cadenas de texto o "strings'. Algunas de las "String Metrics" más conocidas son: la distancia de Hamming, la distancia de Levenshtein, la distancia de Jaro-Winkler y la similitud de Jaccard.

En este proyecto, se estudian los algoritmos de detección de duplicados más comunes, probando concretamente la solución propuesta por Microsoft.

Contenido

Capítulo 1. Objetivos y Desarrollo

1.1. Introducción

Cada día, la gran cantidad de datos a los que se puede tener acceso crece de manera exponencial y con ella, aumenta también la necesidad de integrar dichos datos. Para poder aportar una visión consistente y precisa de esa información, con esos datos originados en fuentes tan dispares, es necesario determinar a qué entidad real se refieren.

La obtención de un sistema que permita identificar de forma automática registros que se refieren a la misma entidad real, entre distintas bases de datos, se convierte en un requisito clave para muchas empresas e instituciones.

1.2. Motivación

Las bases de datos y la explotación y transformación en información de las mismas es una parte muy relevante hoy en día en las empresas. Así pues, la calidad de dicha información es de suma importancia, ya que las decisiones que se tomen en la empresa estarán fuertemente influenciadas por esa información.

Al extraer los datos de distintas fuentes surge un importante problema: una entidad en la realidad está representada de forma distinta en cada origen de los datos, y para obtener información fiable de esa entidad es imprescindible averiguar cuáles son todas sus representaciones.

El problema en ocasiones va más allá de simples errores tipográficos. Cuando obtienes datos de fuentes tan distintas, como pueden ser dos empresas diferentes, los datos se habrán introducido de forma distinta, siguiendo la lógica particular de cada empresa, por lo que, aunque no haya errores tipográficos, ambas representaciones podrán diferir mucho entre sí.

La identificación de esos registros, que representan la misma entidad supone una tarea con un coste humano muy grande si no se automatiza lo máximo posible. Es por esto, que los algoritmos para determinar la similitud de dos registros son cada vez más importantes.

1.3. Objetivo del proyecto

El objetivo de este proyecto es realizar un estudio de algunas técnicas y algoritmos existentes para la detección de duplicados (*"Record Linkage"*), así como las técnicas más comunes de comparación difusa de cadenas de texto (*"String metrics"*) que se usan en dichos algoritmos. Se realizará una categorización estos algoritmos y se explicarán mostrando su funcionamiento.

Además, se realizarán una serie de pruebas con datos reales obtenidos de distintas fuentes para determinar la validez y viabilidad de realizar este proceso de detección de idénticos de forma automática.Para esto se usará la solución propietaria de Microsoft para este problema, un proceso llamado *"fuzzy lookup"* del *Integration Services* de SQL Server.

Por último, se propondrán mejoras y se probarán para poder comparar los resultados y determinar la eficacia y validez de estas propuestas de mejoras sobre el algoritmo "básico".

Capítulo 2. Estado del arte

En este capítulo se mostrarán las técnicas de *"String Metrics"* y de *"Duplicate Detection"* más comunes y las medidas que se suelen usar para comprobar su efectividad. Por último, hablar de los problemas de rendimiento que se dan al estar trabajando con volúmenes de datos tan grandes.

El capítulo pues se estructurara en tres partes, en la primera se tratan las técnicas para comparar campos y darles un valor de similitud, en la segunda se describen varios algoritmos que usan las técnicas descritas en la primera parte, y por último, se repasan los problemas de rendimiento que se originan y las soluciones que se suelen aplicar.

2.1. Técnicas de similitud entre campos

Hay muchos artículos que tratan este tema [1] [2] en donde se suele seguir la estructura que se va a emplear en este trabajo. Las técnicas de similitud entre campos se dividirán según el concepto en el que estén basadas. De este modo se organiza el capítulo agrupando las técnicas en técnicas basadas en carácter, técnicas basadas en *"token"* y técnicas basadas en la fonética.

2.1.1. Basadas en carácter

Distancia de edición

La distancia de edición es la técnica de similitud entre campos más común y más básica. Esta técnica se define como el número mínimo de operaciones para transformar una cadena de caracteres "A" en otra cadena de caracteres "B".

Existen tres tipos distintos de operaciones que se pueden aplicar a una cadena de texto:

- Inserción (insertar un carácter en el *"string"*)
- Borrado (eliminar un carácter del *"string"*)
- Sustitución (cambiar un carácter por otro si son distintos)

En la variación más simple de la distancia de edición, cada operación tiene un coste de 1 (si se necesita realizar una de las tres operaciones para transformar un *"string"* en otro se debe incrementar en uno la distancia). Esta versión de la distancia de edición se conoce como la distancia de Levenshtein [3].

Por ejemplo, calculemos la distancia de edición entre "COCHE" y "COSE"

A	C	O	C	H	E	**Total**
B	C	O	S	E		
Operación	-	-	Sustitución	Sustitución	Borrado	
Coste	+0	+0	+1	+1	+1	**3**

Tabla 1. Distancia de edición

El coste para transformar el *"string"* A en el B es 3 en el ejemplo dado. Esto significa que con 3 operaciones sobre "COCHE" se puede obtener "COSE".

Sin embargo la distancia de edición (el coste mínimo) es 2:

A	C	O	C	H	E	**Total**
B	C	O	S		E	
Operación	-	-	Sustitución	Borrado	-	
Coste	+0	+0	+1	+1	+0	**2**

Para calcular esta distancia existe un algoritmo de programación dinámica (4) que usa la siguiente función para completar la matriz que forman ambas cadenas.

$$D(i,j) = \min \begin{cases} D(i-1,j-1) + Coste(A_i, B_j) \\ D(i,j-1) + 1 \\ D(i-1,j) + 1 \end{cases}$$

Ecuación 1. Distancia de edición

Una vez rellenada la matriz el coste quedaría en la última celda de la misma (esquina inferior derecha). Ese número determina lo parecidos o distintos que son ambos *"strings"*, de este modo, si el numero supera cierto umbral se pueden establecer ambos *"string"* como distintos y en caso contrario como iguales.

El punto fuerte de esta técnica es su capacidad para detectar errores tipográficos típicos (Por ejemplo, errores que se producen al introducir un dato porque dos caracteres están muy próximos en un teclado).

El coste de este algoritmo es $O(|A| * |B|)$[1] ya que es necesario recorrer toda la matriz para determinar la distancia. Sin embargo existen algoritmos más complejos que mejoran este tiempo.

Distancia de Smith-Waterman

Esta técnica de comparación de cadenas de texto es una variación sobre la distancia de edición propuesta por Smith y Waterman [5]. En esta variación las cadenas de texto se comparan intentando identificar secciones parecidas. Esto se hace suponiendo que las diferencias al principio y final de los *"string"* tienen menor coste que las que están en el medio.

Por ejemplo, las cadenas "Google Company" y "Google C." serían identificadas como similares ya que al encontrar "oogle C" en ambos *"strings"* se asume que los caracteres al inicio y al final tienen menor peso.

[1] Donde |A| y |B| representan la longitud de las cadenas A y B respectivamente.

De nuevo la distancia se calcula usando una matriz rellenándola con la siguiente función:

$$D(i,j) = \max \begin{cases} 0 \\ D(i-1,j-1) + Coste(A_i, B_j) \\ D(i-1,j) + Gap \\ D(i,j-1) + Gap \end{cases}$$

Ecuación 2. Smith-Waterman

Donde:

$D(i,j)$ = puntuación del elemento i, j.

$Coste(A_i, B_j)$ = coste de transformar A_i en B_j. Si $A_i = B_j$ entonces $Coste = 1$ si no $Coste = -1$

$Gap = -2$

El proceso para rellenar la matriz es algo distinto al seguido en la distancia de edición. Primero se deben inicializar la primera fila y la primera columna a 0. El resto es igual, teniendo en cuenta las diferencias en los costes de las operaciones.

Esta matriz en el algoritmo de Smith-Waterman se usa para identificar la sección similar entre dos *"strigns"*. Una vez obtenida se debe retroceder desde la casilla con el número mayor hasta que se encuentre un 0.

Por ejemplo, tal y como se puede apreciar en la matriz resultado al comparar los *"strings"* "GUMBO" y "GAMBOL" [1]

		0	1	2	3	4	5
			G	**U**	**M**	**B**	**O**
0		0	0	0	0	0	0
1	**G**	0	1	0	0	0	0
2	**A**	0	0	0	0	0	0
3	**M**	0	0	0	**1**	0	0
4	**B**	0	0	0	0	**2**	0
5	**0**	0	0	0	0	0	**3**
6	**L**	0	0	0	0	0	1

Tabla 2. Smith-Waterman

Retrocediendo desde la posición 5,5 hasta la 3,3 tendríamos la parte similar entre los *"string"*

GU**MBO**
GA**MBO**L

Una vez hecho esto se establecen costes más bajos a las diferencias en las secciones del principio y el final y costes más altos en la sección similar. De nuevo se usa un umbral al que se somete el coste para determinar la clasificación de ambas cadenas de caracteres.

Al igual que en la distancia de edición es necesario recorrer toda la matriz para completarla por lo que el coste computacional es $O(|A| * |B|)$.

Distancia “affine gap”

La distancia “affine gap” [6] tiene la misma base que la distancia de Smith-Waterman pero con algunas diferencias. La distancia de Smith-Waterman falla en algunos casos cuando en una de las cadenas de texto hay una palabra en medio irrelevante (o poco relevante) y en la otra no. En ese caso el núcleo del *“string”* (donde más peso tienen las operaciones de transformación) tendría varias añadidos en el *“string”* al que le faltaba la palabra en medio por lo que la distancia seria alta.

Sin embargo, la distancia “affine gap” funciona bien en los casos en los que hay inserciones o borrados consecutivos de caracteres (Por ejemplo, inserciones y borrados de palabras). Así, una sola inserción (o borrado) larga es más “barata” (tiene menos peso) que una serie de inserciones cortas.

Por ejemplo, al comparar “STTTANDFOR” con “STANDFOR”

S	T	T	T	A	N	F	O	R	
S	-	T	-	A	N	F	O	R	
+0	+1	+0	+1	+0	+0	+0	+0	+0	**2**

Tabla 3. Distancia sin ordenar borrados

S	T	T	T	A	N	F	O	R	
S	T	-	-	A	N	F	O	R	
+0	+0	+1	+1	+0	+0	+0	+0	+0	**2**

Tabla 4. Distancia con borrados ordenados

Se ve que es irrelevante la organización de los borrados usando el método habitual para obtener la distancia entre las palabras, sin embargo, usando *“affine gap”* el coste de una inserción o un borrado sólo se contaría con el carácter que inicia esa inserción o borrado.

S	T	T	T	A	N	F	O	R	
S	-	T	-	A	N	F	O	R	
+0	+1	+0	+1	+0	+0	+0	+0	+0	**2**

Tabla 5. Affine gap, borrados no ordenados

S	T	T	T	A	N	F	O	R	
S	T	-	-	A	N	F	O	R	
+0	+0	+1	+0	+0	+0	+0	+0	+0	**1**

Tabla 6. Affine gap, borrados ordenados

La distancia es menor cuando hay borrados consecutivos. Normalmente esta técnica se programa usando programación dinámica al igual que en los casos anteriores.

Distancia de Jaro-Winkler

La distancia de Jaro-Winkler [7] fue desarrollada para el caso concreto de comparar nombres y apellidos entre sí (nombres con nombres y apellidos con apellidos).

Dadas dos cadenas de caracteres A y B, se siguen los siguientes pasos para calcular la distancia de Jaro-Winkler:

1. Se calculan las longitudes de los *"strings"* $(|A|, |B|)$.
2. Se buscan los caracteres comunes. Siendo los caracteres comunes todos los caracteres de $A[i]$ y $B[j]$ tal que $A[i] = B[j]$ y $|i - j| \leq {}^{1}/_{2} \min(|A|, |B|)$.
3. Se buscan el número de trasposiciones. Siendo esto el número de caracteres que no están en la misma posición que su homólogo.

La fórmula para calcular la distancia de Jaro-Winkler es:

$$d_j = \frac{1}{3}\left(\frac{m}{|A|} + \frac{m}{|B|} + \frac{m-t}{m}\right)$$

Ecuación 3. Jaro-Winkler

Donde m es el número de caracteres comunes y t es la mitad del número de trasposiciones.

Para entenderlo mejor se usara el siguiente ejemplo (se comparara "SYBIL" y "SYBBIEL")

	S	Y	B	I	L
S	1	0	0	0	0
Y	0	1	0	0	0
B	0	0	1	0	0
B	0	0	1	0	0
I	0	0	0	1	0
E	0	0	0	0	0
L	0	0	0	0	1

Tabla 7. Ejemplo Jaro-Winkler

Se rellena la matriz con 1 si y solo si $A[i] = B[j]$ y $|i - j| \leq {}^{1}/_{2} \min(|A|, |B|)$. Si no se pone 0. La suma de los unos será el número de caracteres comunes, y el número de trasposiciones serán los unos que no están en la diagonal.

$$d_j = \frac{1}{3}\left(\frac{5}{5} + \frac{5}{7} + \frac{5-1.5}{5}\right) = 0.8$$

2.1.2. Basadas en *"token"*

Las técnicas basadas en carácter son eficaces tratando errores tipográficos pero fallan cuando se están comparando cadenas de caracteres largas con palabras en distintos órdenes, (Por ejemplo, "Juan Pérez" o "Pérez, Juan). Las técnicas basadas en carácter clasifican esos casos como *"strings"* distintos aunque es obvio que son la misma entidad real.

Para paliar esa deficiencia se usan las técnicas basadas en *"token"*, en las cuales previamente se dividen las cadenas en *"tokens"* (se puede considerar una palabra como un *"token"*). Lo más común es partir la cadena por el espacio en blanco para obtener un conjunto de palabras.

Coeficiente de Jaccard

El coeficiente de Jaccard se usa para medir la similitud de dos conjuntos de *"tokens"*. Esta medida viene determinada por el tamaño de la intersección de ambos conjuntos de *"tokens"*.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Ecuación 4. Coeficiente de Jaccard

Por ejemplo, supongamos los dos *"strings"* A = "Dr. Jose Martin" y B = "Martin Jose". Se generarían los vectores de *"tokens"* $t_1 = [\text{"Dr."}, \text{"}Jose\text{"}, \text{"}Martin\text{"}]$ y $t_2 = [\text{"}Martin\text{"}, \text{"}Jose\text{"}]$.

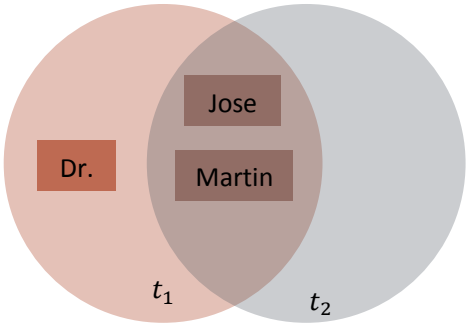


Figura 1. Representación intersección

Usando la formula anteriormente descrita se obtiene: JaccardSimilarity(A,B) = 2/3.

Existen varias desventajas en esta medida de similitud, por ejemplo, la similitud de Jaccard es muy sensible a los errores tipográficos, ya que un error de este tipo haría que dos *"tokens"* fuesen distintos cuando en realidad deberían ser el mismo (y de este modo ese *"token"* ya no se encontraría en la intersección).

La principal ventaja es que la similitud entre dos cadenas de texto no depende del orden en el que estén las palabras en la cadena ya que solo se tiene en cuenta que *"tokens"* están en ambos *"string"*.

Similitud del coseno

Al comparar cadenas de caracteres se pueden representar estas como vectores de n dimensiones. La similitud del coseno es el coseno del ángulo que forman dichos vectores. Este valor va de 0 a 1, donde 0 significa que ambos *"strings"* tienen similitud total y 1 lo contrario.

Si se va a usar esta medida, el primer paso suele ser dividir la cadena de caracteres en *"tokens"* (partir por espacio en blanco para formar un vector de palabras, por ejemplo). Esta medida de similitud es la más usada al tratar documentos de texto [8].

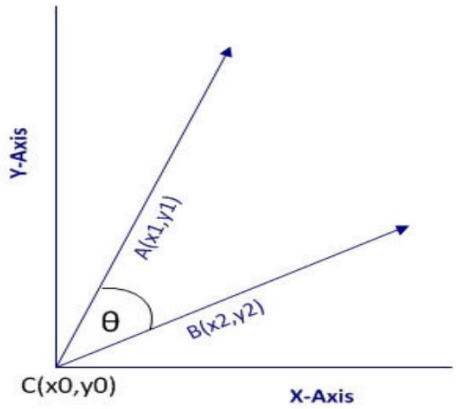


Figura 2. A y B tokenizado, similitud del coseno

Considérense dos *"strings"* A y B con *"tokens"* [A1,A2] y [B1,B2]. La similitud del coseno sería:

$$\text{CosenoSim(A, B)} = \cos\theta = \frac{A * B}{|A| * |B|} = \frac{x1 * x2 + y1 * y2}{\sqrt[2]{x1^2 + y1^2} * \sqrt[2]{x2^2 + y2^2}}$$

Ecuación 5. Similitud del coseno

Donde:

$$x1 = \log(1 + tf_{A1}) * \log(idf_{A1})$$
$$y1 = \log(1 + tf_{A2}) * \log(idf_{A2})$$
$$x2 = \log(1 + tf_{B1}) * \log(idf_{B1})$$
$$y2 = \log(1 + tf_{B2}) * \log(idf_{B2})$$

$$tf_{A1} = frecuencia\ del\ término\ del\ token\ A1\ en\ el\ string\ A$$
$$idf_{A1} = frecuencia\ del\ documento\ inversa\ del\ token\ A1\ en\ el\ string\ A$$

Frecuencia del término y Frecuencia del documento inversa (*tf-idf*) son medidas estadísticas para valorar la importancia de una palabra en un documento.

De nuevo, como ocurría con el coeficiente de Jaccard, la similitud del coseno no depende del orden de las palabras.

q-GRAMS

Se define un "q-gram" como un *"substring"* de una cadena de caracteres cuya longitud es "q". El objetivo es partir el *"string"* en *"tokens"* usando "q-grams" de tamaño "q" y comparar dichos *"tokens"* para encontrar la similitud (usando caracteres de margen al principio y al final para asegurarse de que los caracteres de los extremos son tomados en consideración) [9].

Véase el siguiente ejemplo. Se va a comparar "Will" y "Bill" usando "q-grams" de tamaño 2. Así pues de "Will" se obtiene: "#W", "Wi", "il", "ll", "l#".

Y de "Bill": "#B", "Bi", "il", "ll", "l#".

Will	Bill
#W	#B
Wi	Bi
il	**il**
ll	**ll**
l#	**l#**

Tabla 8. Ejemplo q-GRAMS

Se tienen 3 coincidencias, a más coincidencias más similitud.

2.1.3. Basadas en la fonética

Las cadenas de texto pueden ser fonéticamente similares incluso si son sintácticamente distintas. Existen métodos para obtener esta medida de similitud entre palabras, generalmente se basan en convertir las palabras en cadenas de cuatro caracteres. Estas cadenas se construyen con la primera letra de la palabra y tres números calculados con una función de *"hash"*. Ese código debería describir la fonética de la palabra.

Sin embargo, estas técnicas de similitud son dependientes del idioma, lo cual puede suponer una desventaja (Por ejemplo, si el sistema trata palabras en distintos idiomas).

2.2. Técnicas de detección de duplicados

En apartados previos del trabajo se ha visto distintas formas de determinar la similitud de dos campos de un registro, sin embargo, es habitual que los datos se repartan entre distintos campos en los registros y sea necesario realizar algún tipo de agrupación o ponderación de las medidas que se han calculado y determinar así si una fila en su conjunto es similar a otra. Esto complica el problema de detectar registros duplicados.

A continuación se describirán los modelos de detección de duplicados (aplicando las técnicas descritas con anterioridad) más comunes. Hay muchos artículos y trabajos que tratan este tema [1] [2], normalmente clasificando estos modelos en:

- Modelos probabilísticos
- Aprendizaje supervisado
- Aprendizaje activo
- Técnicas basadas en la distancia
- Técnicas basadas en reglas
- Aprendizaje no supervisado

2.2.1. Modelos probabilísticos

Tal y como el nombre indica, los modelos probabilísticos usan la teoría de la probabilidad para clasificar los datos como duplicados o no [10].

Generalmente un algoritmo probabilístico para identificar idénticos sigue los siguientes pasos:

1. Extraer un conjunto de datos de ejemplo de entre todos los datos.
2. Clasificar manualmente el conjunto de datos de ejemplo.
3. Se determina qué campos deberían tener estadísticamente más peso y cuáles menos.
4. Se calcula un *"score"* combinado usando los pesos establecidos anteriormente que representa la probabilidad de que dos registros hagan referencia al mismo objeto del mundo real.
5. Habitualmente se establecen dos umbrales, uno "superior" para clasificar aquellos registros que lo superen como la misma entidad real y otro "inferior" que clasifica a los que se quedan por debajo como entidades distintas. El rango entre esos dos umbrales suele considerarse como "potencialmente iguales". Los umbrales se establecen por un experto en el dominio.

Uno de los modelos probabilísticos más comunes es el Modelo de decisión Bayesiano [11], que está basado en el teorema de Bayes para calcular la probabilidad de que dos registros representen a la misma entidad.

2.2.2. Aprendizaje supervisado

El aprendizaje en los humanos se puede producir gracias a experiencias pasadas; sin embargo, un ordenador no tiene esa posibilidad. El ordenador "aprende" de unos datos que representen esas experiencias. Esto significa que se puede usar un conjunto de aprendizaje que represente unas experiencias previas en un dominio [2].

Por ejemplo, podemos usar los siguientes datos que representan experiencias pasadas a la hora de conceder un crédito:

Edad	Trabajo	Casa	Decisión
25	Si	No	No
37	Si	Si	Si
54	No	Si	No
33	Si	No	Si

Tabla 9. Conjunto entrenamiento, aprendizaje supervisado

Con esos datos el sistema se "entrena" para responder a unos determinados criterios. Una vez entrenado, usa ese conjunto de entrenamiento para clasificar los registros que le lleguen. Por ejemplo, podría clasificar si dar o no un crédito en función de esos criterios (si la edad es baja no se da el crédito… si no tiene trabajo y la edad es mayor que x no se da el crédito… etc.).

El conjunto de entrenamiento normalmente se selecciona de forma manual, el experto identifica los datos que puedan representar todos o la mayor parte de los escenarios posibles. El hecho de predefinir este conjunto de aprendizaje da nombre a esta técnica, aprendizaje supervisado.

Una de las técnicas de aprendizaje supervisado más útiles son las SVMs [12] (*"Support Vector Machines").* Las SVMs aprenden usando ejemplos para luego clasificar los datos y asignarles etiquetas.

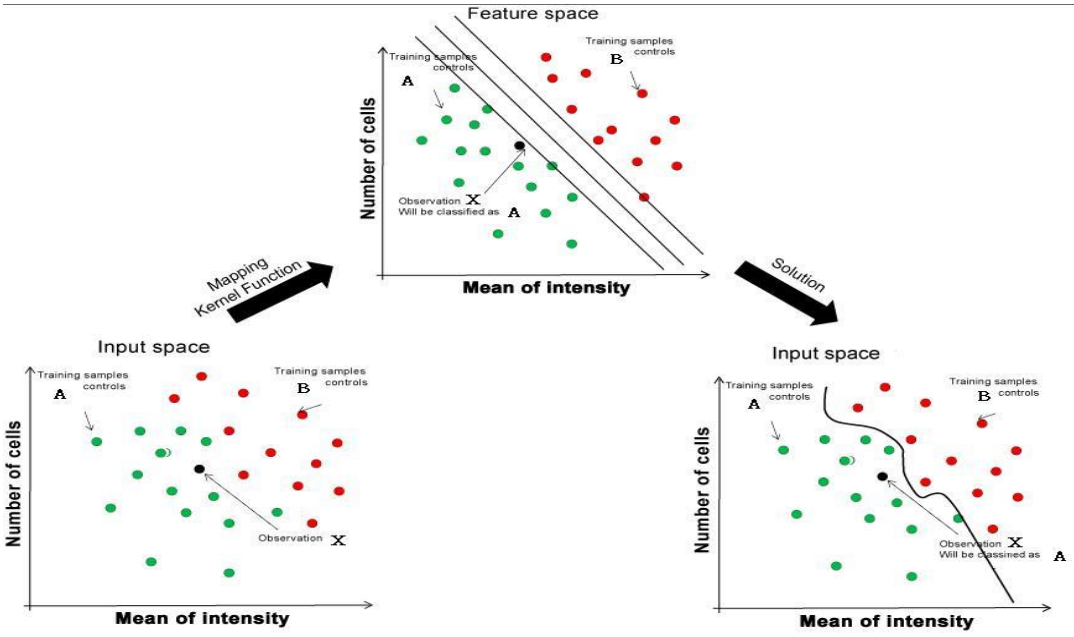


Figura 3. SVM

Es importante tener en cuenta que el resultado de la clasificación depende del conjunto de entrenamiento, si éste es realmente representativo, la SVM clasificará correctamente los datos. Normalmente definir ese conjunto de entrenamiento es un problema ya que se requiere un gran número de ejemplos de entrenamiento, además de que sean lo suficientemente variados y representen todos los posibles casos del dominio.

2.2.3. Técnicas de aprendizaje activo

Como se ha visto anteriormente, las SVM y otras técnicas de aprendizaje supervisado dependen completamente del conjunto de entrenamiento determinado de antemano. Sin embargo, en ocasiones no es posible (o no es factible) definir un conjunto de entrenamiento que represente por completo la realidad del dominio. Las técnicas de aprendizaje activo [13] procuran identificar esos ejemplos que proporcionan más precisión al conjunto de entrenamiento.

En cada iteración del aprendizaje activo se añaden más ejemplos al conjunto de entrenamiento. Los ejemplos que mejoren el sistema se identifican y etiquetan incluyéndose en el conjunto de entrenamiento. El sistema entonces se vuelve a entrenar para incluir el nuevo "conocimiento" que han añadido los nuevos ejemplos.

Existen tres clases de técnicas de aprendizaje activo:

1. Técnicas de muestreo para la incertidumbre: Este tipo de aprendizaje activo identifica los casos para los que el sistema tiene menos certidumbre (menos certeza) en su predicción y así mejorar estos casos.
2. "Consulta a un comité": Se utiliza un "comité" de aprendices (sistemas de aprendizaje) y se usa el desacuerdo entre ellos como medida para evaluar la cantidad de información que proporcionaran los ejemplos al conjunto de entrenamiento.
3. Técnicas de reducción del error: Se seleccionan los ejemplos que, una vez etiquetados, proporcionaran mayor reducción al error minimizando la varianza de la predicción.

2.2.4. Técnicas basadas en la distancia

Las técnicas de aprendizaje activo, así como las de aprendizaje supervisado, necesitan de un conjunto de entrenamiento previo o la participación de un humano que lo cree. Si no se dispone de la posibilidad de dicho conjunto previo para aprender de él no se pueden aplicar técnicas de aprendizaje supervisado o activo.

Una solución para este problema es definir una medida de "distancia" entre los registros que no requiera de un entrenamiento previo [14]. Se mide la distancia entre los campos individualmente (usando "string metrics") y después se calcula una distancia global ponderando las distancias individuales de los campos.

El principal problema de estas técnicas es la elección de un umbral apropiado pasa clasificar los registros. Con un conjunto de entrenamiento se puede aproximar un umbral que clasifique correctamente los registros. Sin embargo, esto eliminaría la mayor ventaja de estos métodos, que es la posibilidad de usarlos sin necesidad de un conjunto de entrenamiento previo.

2.2.5. Técnicas basadas en reglas

Las técnicas basadas en reglas [15] (tal y como indica su nombre) usan reglas que definen cuándo dos registros son similares.

Por ejemplo, dados dos registros, r1 y r2:

```
SI
      r1.nombre es igual que r2.nombre  Y
r1.nacimiento es igual que r2.nacimiento Y
r1.calle es muy parecida a r2.calle
ENTONCES
      r1 es equivalente a r2
FIN SI
```

La regla anterior se puede implementar usando técnicas de "string metrics" vistas anteriormente en aquellos casos que se requiera una medida difusa ("*r1.calle es muy parecida a r2.calle*") estableciendo un umbral para determinar qué es "muy parecida". Es muy importante elegir bien el umbral y las técnicas de "string metrics". Generalmente los umbrales se establecen gracias a la experiencia obtenida de experimentos.

La principal ventaja de estas técnicas es su capacidad para adaptarse y modificarse a distintos dominios ya que pueden personalizarse con facilidad (aunque esto hace que las reglas demasiado dependientes de un dominio quizás no puedan usarse en otros distintos).

2.2.6. Aprendizaje no supervisado

Comparándolo con el aprendizaje supervisado, el aprendizaje no supervisado [16] no requiere un conjunto de entrenamiento específico previamente definido. Aprendizaje no supervisado es el título general que se les dan a los sistemas que usan el aprendizaje en los cuales la entrada contiene los datos tal cual, sin información adicional (datos de entrenamiento). Una de las tareas del aprendizaje no supervisado es proporcionar la clasificación de forma automática. Se obtiene la similitud entre los registros para determinar si se pueden agrupar.

El aprendizaje no supervisado se puede dividir en dos tipos de problemas: agrupación de datos ("*data clustering*") y extracción de características (*"feature extraction"*).

La agrupación de datos tiene como objetivo encontrar la estructura de los datos proporcionados. Extracción de características, por otro lado, intenta reducir las dimensiones de los daos proporcionando una representación más "compacta" de los mismos.

2.3. Medidas

Hay tres medidas que se usan comúnmente en este campo para determinar la efectividad de las técnicas de detección de duplicados. Estas son *"precision"*, *"recall"* y *"f-measure"*.

- *"Precision"*: relación entre el número de duplicados identificados correctamente y el número total de duplicados que ha identificado el sistema.

$$Precision = \frac{duplicados\ identificados\ correctamente}{número\ duplicados\ identificados}$$

Ecuación 6. Precision

- *"Recall"*: relación entre el número de duplicados que el sistema ha identificado correctamente y el número de duplicados que realmente hay en los datos.

$$Recall = \frac{duplicados\ identificados\ correctamente}{número\ duplicados\ reales}$$

Ecuación 7. Recall

- *"f-measure"*: media armónica de la *"precision"* y el *"recall"*.

$$f - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Ecuación 8. f-measure

2.4. Rendimiento

Uno de los problemas más importantes derivados de la detección de duplicados es el descenso en el rendimiento y la falta de eficiencia al trabajar con volúmenes de datos muy grandes. Los algoritmos más simples de detección de duplicados comprueban por cada registro si este es similar a todos los demás.

Sobre este tema hay muchos artículos [17] que procuran mejorar la eficiencia agrupando los registros en conjuntos potencialmente similares para reducir el número de comparaciones que deben calcular. Dichos artículos proponen el uso de índices (indexación e indexación inversa), filtros por prefijo, posición o sufijo… para reducir el número de candidatos a duplicado de cada registro al mínimo manteniendo la precisión de los resultados.

Sin embargo, este proyecto se centrará en la eficacia de los métodos de detección de duplicados y no en su eficiencia.

Capítulo 3. "Fuzzy Lookup"

3.1. Introducción

Este trabajo se centrara en la solución propuesta por Microsoft [18] al problema de la detección de duplicados. Se trata de una funcionalidad incluida en paquete de *"SQL Server Integration Services"* llamada *"fuzzy lookup"*. El motivo de centrar las pruebas en esta herramienta es que los datos reales que se disponían estaban ya en una base de datos SQL Server y la integración con esta solución era casi inmediata. Además se trata de una solución que goza de buena fama en el campo y proporciona, aparentemente, unos resultados "aceptables".

Ante el problema descrito en este trabajo Microsoft desarrollo una solución robusta y eficiente de "mapeo difuso" independiente del dominio (pero que pudiese suponer una fuerte base para realizar mejoras dependientes de dominios concretos).

El aspecto general de este algoritmo sigue los siguientes pasos: 1. Se comprueba si la tupla de entrada coincide exactamente con alguna de las de la tabla de referencia, 2. Si no existe la coincidencia exacta se ejecuta el algoritmo de mapeo difuso buscando la tupla más parecida de las de la tabla de referencia, 3. Si se supera un umbral se considera con son la misma entidad.

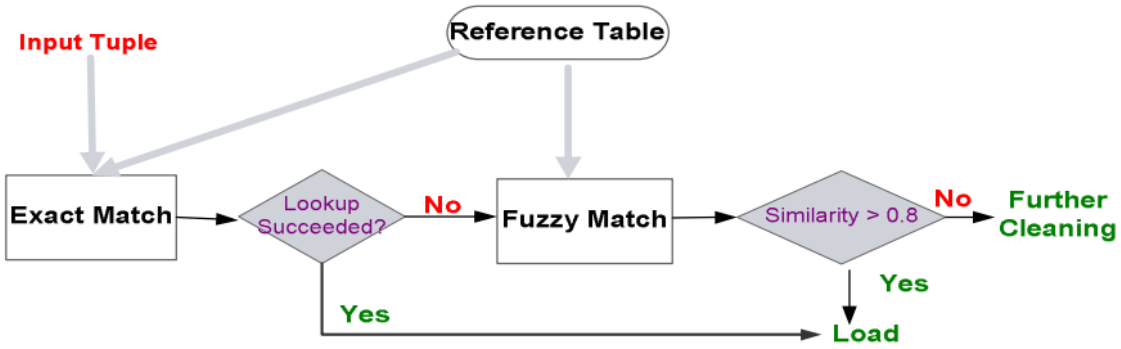


Figura 4. Uso del "mapeo difuso". Obtenida de [18]

Como se puede suponer, el componente crítico de un algoritmo de "mapeo difuso" es la función de similitud que se use para comparar registros (que en el fondo es la que proporciona la parte difusa).

El objetivo pues, de este algoritmo de mapeo difuso, es devolver el registro más similar a la tupla de entrada de entre una tabla de referencia. Una mejora de esta funcionalidad es proporcionar los "K" registros más similares. La última mejora del algoritmo consiste en proporcionar la opción de devolver como mucho "K" registros que superen un umbral definido por el usuario.

El desarrollo de este algoritmo gira en torno a una pieza clave, la definición de una función de similitud, denominada *"fuzzy match similarity"* (en adelante fms). Esta función ve un *"string"* como una secuencia de *"tokens"* y reconoce la importancia de cada *"token"* asociándolo a un peso (cuantificando así su importancia). Usando la "frecuencia inversa del documento" (*"inverse document frequency"*) hace que la importancia de un *"token"* disminuya con su frecuencia (que es el número de veces que un *"token"* se repite en la tabla de referencia).

3.2. Función de similitud

Para comenzar a hablar sobre la función de similitud se deben dar algunas definiciones:

- Distancia de edición: Ya descrita anteriormente. $ed(s_1, s_2)$
- Relacion de referencia: Sea $R[tid, A_1, \dots, A_n]$ una relación de referencia, donde A_i representa a la i-esima columna. Se asume que cada A_i es de tipo *"string"* y que tid es la clave de *R*.
- "Tokenizador: Sea *tok* una función "tokenizadora" que separa un *"string" s* en un conjunto de *"tokens"*, *tok(s)*, a partir de una serie de delimitadores. Para obtener los *"tokens"* generados por una columna determinada se usa *tok(v[col])*. De este modo, *tok(v)* sera la unión de los *"tokens"* de todas las columnas de la tupla *v*.
- Funcion de peso: Se adapta la función de peso IDF (*"inverse document frecuency"*) para acomodarla al dominio relacional. Se considera cada tupla como un documento de *"tokens"*. Sea la frecuencia de un *"token"* t en la columna i, denotada por *freq(t,i)*, el número de tuplas *v* en *R* tal que *tok(v[i])* contiene *t.* El valor IDF, *IDF(t, i)*, de un *"token" t* respecto a la i-esima columna de *R* se calcula (cuando $freq(t, i) > 0$):

$$w(t, i) = IDF(t, i) = \log \frac{|R|}{freq(t, i)}$$

Ecuación 9. Función de peso, fuzzy lookup

3.2.1. Fuzzy Similarity Function (fms)

La similitud entre una tupla de entrada y una tupla de referencia es el coste de transformar la primera en la segunda (a menor coste mayor similitud). Las operaciones de transformación son: sustitucion de *"token"*, inserción de *"token"* y borrado de "*token"*. Cada una de estas operaciones está asociada a un coste que depende del peso del *"token"* que se va a transformar. Sean *"u"* y *"v"* dos tuplas, se considera a *u* la de entrada y a *v* la de referencia:

1. Sustitución de *"token"*: El coste de sustituir un *"token"* $t_1$en *tok(u[i])* por el *"token"* t_2 de *tok(v[i])* es $ed(t_1, t_2) * w(t_1, i)$. Si t_1 y t_2 son de columnas distintas el coste es infinito.
2. Inserción de *"token"*: El coste de insertar un *"token" t* en *u[i]* es $c_{ins} * w(t, i)$, donde el *factor de inserción del "token"* c_{ins} es una constante entre 0 y 1.
3. Borrado de *"token"*: El coste de borrar un *"token" t* de *u[i]* es $w(t, i)$.

Se ve que los costes asociados a la inserción y borrado de un mismo *"token"* pueden ser distintos. Esto es así porque se cree que dejarse un *"token"* durante la introducción de datos es más común, por eso la ausencia de *"tokens"* no está muy penalizada.

Para transformar *u* en *v* es necesario que cada columna *u[i]* sea transformada en *v[i]* a través de un conjunto de operaciones de transformación, cuyo coste final será la suma de los costes de las operaciones de dicho conjunto. El coste de transformación $tc(u[i], v[i])$ es el mínimo coste de entre todas las posibles formas de transformar *u[i]* en *v[i]*. El coste $tc(u, v)$ será pues la suma de los costes de transformación de cada columna:

$$tc(u, v) = \sum_i tc(u[i], v[i])$$

Ecuación 10. Coste de transformación, fuzzy lookup

El mínimo coste de transformación $tc(u[i], v[i])$ se puede calcular usando el algoritmo de programación dinámica que se usa para la distancia de edición [5].

Definición de fms: Sea *w(u)* la suma de todos los pesos de todos los *"tokens"* en *tok(u)* de la tupla de entrada *u*:

$$fms(u,v) = 1 - \min\left(\frac{tc(u,v)}{w(u)}, 1.0\right)$$

Ecuación 11. fms, fuzzy lookup

3.3. Mapeo Difuso

El problema de los K-mapeos difusos: Dada una relación de referencia *R*, un umbral de similitud *c* (0 < *c* < 1), la función de similitud *f* y un registro de entrada *u*, encontrar el conjunto *FM(u)* de mapeos difusos con como mucho K registros de *R* tal que:

1. $fms(u,v) \geq c\ para\ todo\ v\ en\ FM(u)$
2. $fms(u,v) \geq fms(u,v')\ para\ todo\ v\ en\ FM(u)\ y\ v' en\ R - FM(u)$

Se puede ver que usando un umbral de 0 se puede simular el escenario en el que el usuario está interesado en los K registros más parecidos.

Para solucionar este problema, un algoritmo simple buscaría en la relación de referencia R comparando cada registro con la tupla de entrada *u*. Una aproximación más eficiente es construir un índice en la relación de referencia para obtener más rápidamente un conjunto de mapeos difusos. Las estructuras de indexación habituales no se pueden aplicar en este contexto porque solo se pueden usar para coincidencias exactas o coincidencias de prefijos. De todas formas, durante una fase de pre-procesamiento, se recopila información adicional de indexación para implementar eficientemente el mapeo difuso. Esta información se almacena en una relación normal de una base de datos y se indexa. A esta relación indexada se la denomina índice tolerante a errores o *"error tolerant index"* (ETI).

La solución para mejorar la eficiencia se basa en derivar de la *fms* una función de similitud fácilmente indexable, fms^{apx}, con las siguientes características:

1. fms^{apx} acota superiormente a *fms* con una alta probabilidad.
2. Se puede construir un índice tolerante a errores (ETI) para proporcionar eficientemente un pequeño conjunto de candidatos cuya similitud es mayor (probabilísticamente hablando) que el umbral de similitud *c*.

De ese conjunto de candidatos se devuelven las K tuplas mas similares a *u* como mapeos difusos.

En los siguientes apartados se define la aproximación de *fms*, el índice tolerante a errores y el algoritmo básico del mapeo difuso.

3.3.1. Aproximación de *"fms"*

Como se ha visto anteriormente, se necesita derivar una fms^{apx} a partir de la *fms* original para poder construir una relación indexada y mejorar la eficiencia. La función fms^{apx} es una versión de la fms que se obtiene:

1. Ignorando las diferencias en el orden entre los *"tokens"* de los registros de entrada y de la tabla de referencia.
2. Permitiendo a cada *"token"* de entrada mapearse con el *"token* más cercano"

Teniendo en cuenta estas dos características la comparación entre dos tuplas solo puede generar mayor similitud que la función original. Es por esto que fms^{apx} acota superiormente a fms.

En fms^{apx} se mide la "cercanía" entre dos *"tokens"* a través de la similitud entre "conjuntos de substrings" (o también denominados conjuntos de *q-grams*) de los *"tokens"*.

Conjunto de *Q-grams*: Dado un *"string" s* y un entero positivo *q*, el conjunto $QG(s)$ de *q-grams* de *s* es el conjunto de todos los *"substrings"* de *s* de tamaño *q.*

Coeficiente de Jaccard: Ya definido anteriormente. $sim(s_1, s_2)$

Similitud del mínimo "*has"* o *"Min-has similarity"*: Sea *H* la función de "*has"* que mapea uniformemente y aleatoriamente *"strings"* con un conjunto de números naturales *N*. Sea *S* un conjunto de *"strings"*. Se define la "firma del mínimo has" (o *"min-has signature"*) *mh(S)* de *S* como el vector $[mh_1(S), \dots, mh_H(S)]$ donde la i-esima coordenada $mh_i(S)$ viene definida por $mh_i(S) = arg_{a \in S} \min h_i(a)$. Sea I[X] una variable indicadora sobre un booleano X. La similitud del mínimo *"has"* viene dada por:

$$sim_{mh}(t_1, t_2) = \frac{1}{H} \sum_{i=1}^{H} I\left[mh_i\big(QG(t_1)\big) = mh_i\big(QG(t_2)\big)\right]$$

Ecuación 12. Similitud del mínimo "has"

Definición de fms^{apx}: Sean *u, v* dos tupas y $d_q = (1 - 1/q)$ un término de ajuste:

$$fms^{apx}(u, v) = \frac{1}{w(u)} \sum_{i} \sum_{t \in tok(u[i])} w(t) * \max_{r \in tok(v[i])} \left(\frac{2}{q} sim_{mh}(QG(t), QG(r)) + d_q\right)$$

Ecuación 13. fms aproximada, fuzzy lookup

3.3.2. Índice tolerante a errores o ¨*Error Tolerant Index*¨ (ETI)

El principal objetivo del *ETI* es proporcionar de forma eficiente, para cada registro de entrada *u*, un conjunto de candidatos *S* de entre los registros de la tabla de referencia cuya similitud con *u* es mayor que el umbral de similitud mínima *c*. Para determinar ese conjunto de candidatos, hace falta identificar eficientemente por cada *"token" t* en *tok(u)*, un conjunto de tuplas de la tabla de referencia que comparten el *"has"* de los *q-grams* con *t*.

Para ello, se almacena en el *ETI* cada *q-gram s* junto con la lista de todas las claves de las filas de la tabla de referencia con *"tokens"* cuya firma *"has"* contienen *s*.

3.3.3. Algoritmo Básico

El algoritmo básico para procesar consultas de mapeos difusos dada una tupla de entrada *u* es el siguiente: Para cada *"token" t* en *tok(u)*, se calcula su peso *IDF*, *w(t)*, el cual requiere la frecuencia de *t*. Esas frecuencias se pueden almacenar en el *ETI*, de todas formas, se asume que las frecuencias de los *"tokens"* pueden ser obtenidas con rapidez. Entonces, determina la mínima firma *"has" mh(t)* de cada *"token" t*. Después se asigna el peso *w(t)/|mh(t)|* a cada *q-gram* en *mh(t)*. Usando el *ETI*, se determina un conjunto de candidatos *S* de filas de la tabla de referencia cuya similitud con el registro de entrada *u* es mayor que *c*. A continuación se verifica que las tuplas de *S* superan el umbral *c* y cuando se ha pasado dicha comprobación se devuelven los *K* registros con mayor similitud.

FuzzyMatch(input tuple u, H, ETI, R, c)

1. Initialize hash table TidScores; AdjustmentTerm = 0
2. Tokenize u and compute min-hash signatures Q of all tokens
3. Assign token weights; RemWt = sum of all token weights
4. threshold = c·RemWt
5. For each q-gram s in Q s.t. $s = mh_i(t)$ of token t in column col
6. if ($mh_i(t)$ is the first q-gram of mh(t) to be looked up)
7. AdjustmentTerm += w(t)·(1-1/q)
8. Fetch tid-list(s) by looking up (s, i, col) against ETI
9. Update TidScores
   a. Increment scores of existing tids by w(t)/|mh(t)|
   b. If RemWt ≥ threshold, insert new tids with score w(t)/|mh(t)|.
10. RemWt −= w(s)
11. Fetch tuples from R for TIDs with score ≥ c–AdjustmentTerm
12. Compare, using f, each of these tuples with u
13. Return K (or less) most similar tuples with similarity above w(u)·c

Figura 5. Algoritmo básico, fuzzy lookup [18]

Capítulo 4. Pruebas y resultados

En este capítulo se realizarán las pruebas sobre datos reales extraídos de distintos supermercados. Se comprobará la eficacia que son capaces de proporcionar los algoritmos de detección de duplicados para comprobar hasta qué punto pueden reducir la intervención de un humano en este problema.

Los datos usados en las pruebas constan de artículos de alimentación repartidos entre tres categorías: aceites, chocolates y cereales. La decisión de tomar estos datos en concreto y separarlos entre sí se ha debido a que las diferencias entre los tres dominios proporciona una visión más general del problema. Además, separando los dominios, las carencias y virtudes de la detección automática de idénticos se puede analizar más fácilmente.

El capítulo se distribuirá de la siguiente manera:

- Metodología: en este apartado se explica los pasos que se dan para realizar las pruebas y extraer la información necesaria de los resultados.
- Medidas: se definen las medidas que se usaran en el capítulo para determinar la eficacia de las pruebas que se han llevado a cabo.
- Fuzzy lookup “básico”: primera versión de prueba con la solución de Microsoft
- Fuzzy lookup “acotado”: mejora de la versión anterior ayudando al algoritmo de Microsoft a acotar el conjunto de registros donde buscar idénticos.
- Fuzzy lookup “plus”: siguiente versión añadiendo información extra a las filas para mejorar los resultados.
- Fuzzy lookup independiente del dominio: análisis de los resultados obtenidos con el fuzzy lookup teniendo las tres categorías juntas.

- Calidad de los datos: se analiza la calidad de los datos de entrada.
- Pruebas con otros algoritmos básicos: se prueban otros algoritmos básicos descritos en el capítulo 2 de este trabajo para compararlos con una solución más elaborada como la que propone Microsoft.

4.1. Metodología

Para realizar las pruebas se siguen en todos los casos prácticamente el mismo proceso. Los pasos generales de la metodología que se ha llevado a cabo para realizar las pruebas son:

1. Se separa cada categoría en una tabla (para facilitar el acceso a los datos más adelante). Este paso tan sólo se realiza una vez, estas tablas se usarán para todas las pruebas. Cabe destacar que estas son las tablas sobre las que se realizaran los pre-procesos necesarios para las pruebas del fuzzy lookup "acotado" y del fuzzy lookup "plus".
2. Se realizan los pre-procesos necesarios (en el caso del fuzzy lookup "básico" este paso se omite).
3. Se lanza la prueba en sí misma. El diseño de la prueba es muy simple, en cualquier caso consta de un componente de lectura de datos, un componente donde se ejecuta la función de detección de duplicados y un último componente de escritura de los datos. En el caso del fuzzy lookup la integración de SLQ Server con el SSIS (SQL Server Integration Services) es inmediata y existen ya los componentes necesarios para conectarse a la base de datos, leer, lanzar el fuzzy lookup (pudiendo configurarlo) y escribir los resultados en la base de datos. El caso de los algoritmos básicos es algo distinto, aunque en esencia consta de las mismas partes, y se explicará algo más en detalle en su correspondiente apartado.

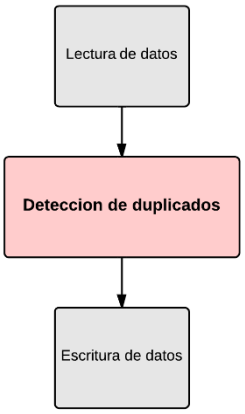


Figura 6. Pruebas, detección duplicados

4. Se extraen los datos de la base de datos conectando Excel a SQL Server. Se presentan los resultados en una tabla dinámica para facilitar el análisis.
5. Se crean las tablas en Excel para obtener las medidas y las gráficas de las mismas.
6. Por último se presentan y analizan los resultados.

Estos pasos se siguen en todas las pruebas. Hay que tener en cuenta que previamente se han identificado los idénticos a mano para poder comprobar si la identificación automática es correcta y poder obtener las medidas.

4.2. Medidas

Las medidas utilizadas en las pruebas para analizar los resultados son las ya definidas en capítulo 2 de este documento. Se tratan de medidas que identifican principalmente cuantos mapeos de los resultantes son correctos y cuantos de los que se deberían haber obtenido se han conseguido. Se tratan de la precisión y el recall respectivamente. Además, se usa también la media armónica, *f-measure*, de las otras dos medidas.

Además de estas tres medidas estándar se usaran los complementarios de la precisión y el recall. Así obtenemos el "número de falsos positivos" y el número de "no identificados".

Falsos positivos (FP): Numero de duplicados que superan el umbral pero son incorrectos.

$$FP = \frac{duplicados\ identificados\ incorrectamente}{número\ duplicados\ identificados} = 1 - precision$$

Ecuación 14. Falsos positivos

No identificados (NI): Numero de duplicados que no ha identificado el sistema de los duplicados reales.

$$NI = \frac{duplicados\ reales\ no\ identificados}{número\ duplicados\ reales} = 1 - recall$$

Ecuación 15. No identificados

Estos datos se presentarán en tablas y en graficas en función del umbral a partir del cual se esté considerando que el duplicado es correcto. Así se puede ver la evolución de estas medidas cuando el umbral varía. La plantilla que se completara en el resto de apartados es la siguiente:

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8								
0.6								
0.5								
0.4								
0.2								
Min Score[2]								

De la tabla se obtiene la gráfica que permite observar más fácilmente la progresión de las medidas más importantes (precisión y recall) cuando varía el umbral.

4.3. Fuzzy Lookup "básico"

En este apartado se pueden ver las pruebas realizadas con un "fuzzy lookup básico". Los ítems se comparan entre si obteniendo la similitud a través de los atributos "nombre", "marca" y "formato". Se obtiene una similitud en cada campo, sin establecer un mínimo y luego el "fuzzy lookup" las pondera y proporciona una similitud total.

Esta forma de comparar ítems da pie a pensar que pueden originarse casos de falsos positivos obvios, por ejemplo ítems con marcas parecidas pero distintas que sean identificados como el mismo, sin embargo dos artículos de distinta marca nunca pueden ser el mismo. De todos modos es la aproximación más sencilla y básica y las características de los datos pueden hacer que con este algoritmo simple se consigan buenos resultados.

[2] Indica el score del identificado correctamente con similarity más baja.

Los procesos que se van a seguir, tanto en este caso como en todos los demás ("fuzzy lookup acotado" y "fuzzy lookup plus"), se pueden observar en la *"Figura 7"*.

El primer proceso que se realiza es el acceso a los datos de los que se pretende encontrar su idéntico. En el caso de la figura se cargan los productos del distribuidor 1, Carrefour. En este primer paso se realiza la conexión con la base de datos, el SSIS proporciona conectores con las bases de datos más comunes, en concreto se está usando una base de datos SQL Server por lo que como se puede deducir, la conexión es muy simple y sencilla.

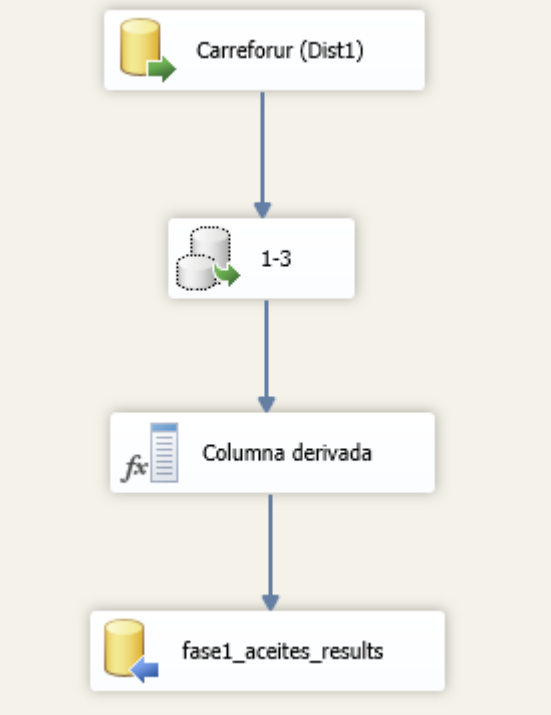


Figura 7. Procesos fuzzy lookup

El siguiente proceso es la búsqueda difusa propiamente dicha, el "fuzzy lookup". En este momento se determina la tabla o vista de la base de datos que va a actuar como tabla de referencia, así, por cada tupla de la tabla de entrada (obtenida en el proceso anterior) se buscan los "*k*" registros más parecidos que cumplan las condiciones que se han establecido en las propiedades del "fuzzy lookup". Es en este paso donde se deciden que atributos van a formar parte de esta búsqueda difusa, si el mapeo entre el atributo de la tabla de entrada y el atributo de la tabla de referencia debe ser una coincidencia exacta o una aproximación difusa, el umbral mínimo, etc… En la siguiente figura (*"Figura 8"*) se pueden apreciar los mapeos entre los distintos atributos. Se trata de una operación muy sencilla, tan solo hay que unir los atributos que se pretenden mapear. Como se puede ver, los tres mapeos están representados por líneas discontinuas, esto significa que el tipo de asignación es difusa. Haciendo clic derecho en una de las líneas y seleccionando "editor de relaciones" se pueden configurar el umbral mínimo de cada mapeo para que sea considerado lo suficientemente parecido como para pasar la selección y poder ser mostrado como idéntico. Todas estas propiedades hacen que los distintos dominios se puedan configurar estableciendo umbrales específicos a sus características para proporcionar la máxima eficacia, una gran ventaja cuando se trabaja con datos que se pueden agrupar en secciones particulares y heterogéneas, sin embargo se requiere realizar un estudio para determinar la configuración optima del proceso de búsqueda difusa.

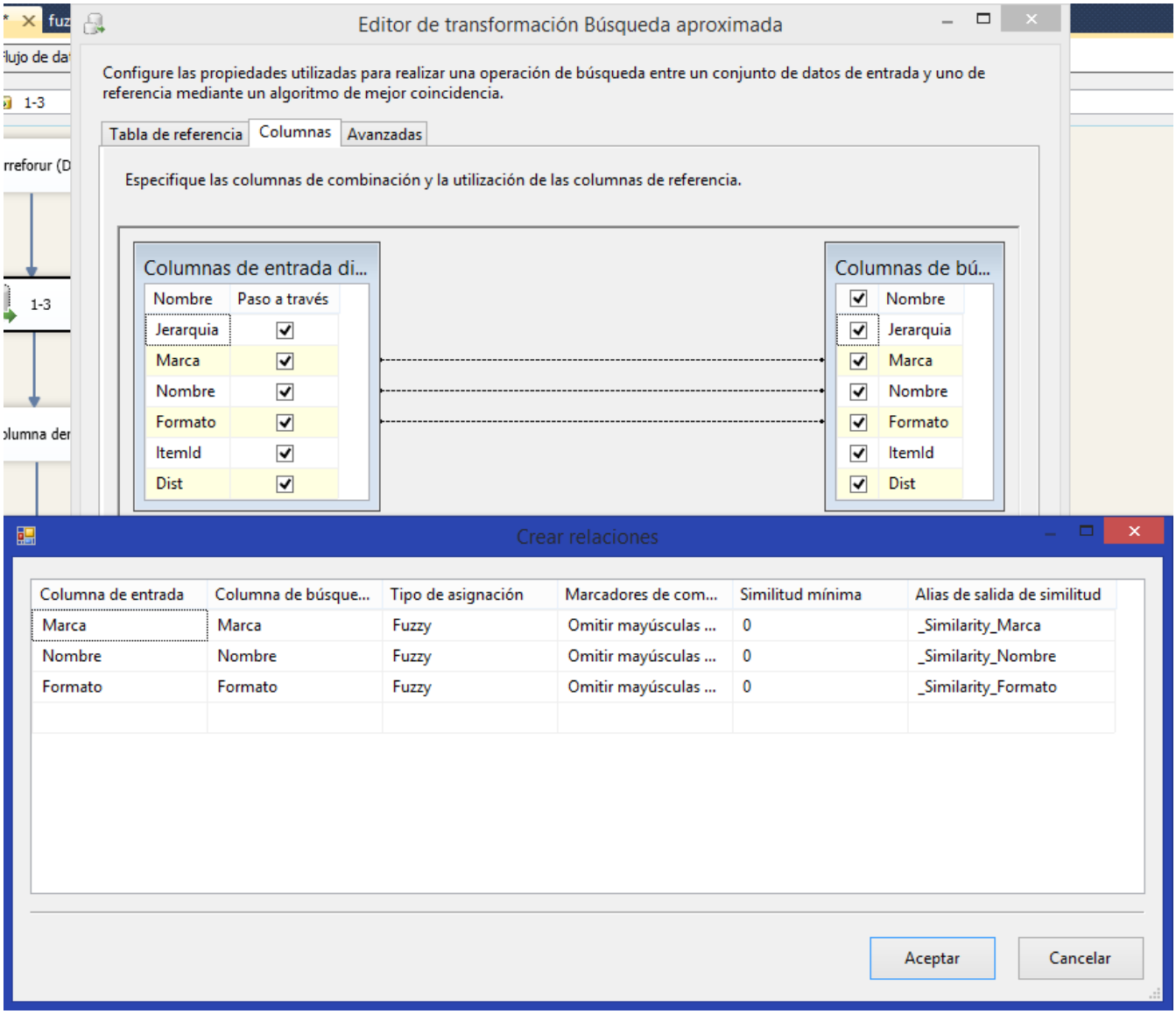


Figura 8. Propiedades fuzzy lookup básico

El tercer proceso simplemente añade información útil a la salida del "fuzzy lookup" en forma de atributos extra. Por ejemplo se puede incluir la fecha, un número de ejecución, los distribuidores que han intervenido en la búsqueda difusa, algún tipo de transformación o de operación que sea interesante realizar a los datos de salida del "fuzzy lookup"…

Por último se almacenan los resultados en la base de datos.

Estos procesos se repiten en todas las pruebas, tan solo varía la configuración de "fuzzy lookup" y el tratamiento previo que se realiza a los datos, en este caso no hay ningún pre-proceso que trate los datos. A continuación se muestran los resultados obtenidos separándolos según su categoría (aceites, chocolates o cereales). En cada apartado se mostraran las medidas definidas anteriormente en función de cinco umbrales, a saber: 0.2, 0.4, 0.5, 0.6 y 0.8. Los datos se muestran, además de en unas tablas, en gráficas para facilitar el análisis y la comparación entre "precisión" y "recall".

4.3.1. Aceites

En el caso de los aceites se ha usado el distribuidor 3 (El Corte Ingles o ECI) como tabla de referencia, de modo que se están buscando artículos de ECI iguales a los que tiene Carrefour. Los resultados obtenidos son los siguientes:

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	5	7	20	**71%**	29%	**25%**	75%	0,37
0.6	10	18	20	**56%**	44%	**50%**	50%	0,53
0.5	14	27	20	**52%**	48%	**70%**	30%	0,60
0.4	16	37	20	**43%**	57%	**80%**	20%	0,56
0.2	16	43	20	**37%**	63%	**80%**	20%	0,51
Min Score	0,4276							

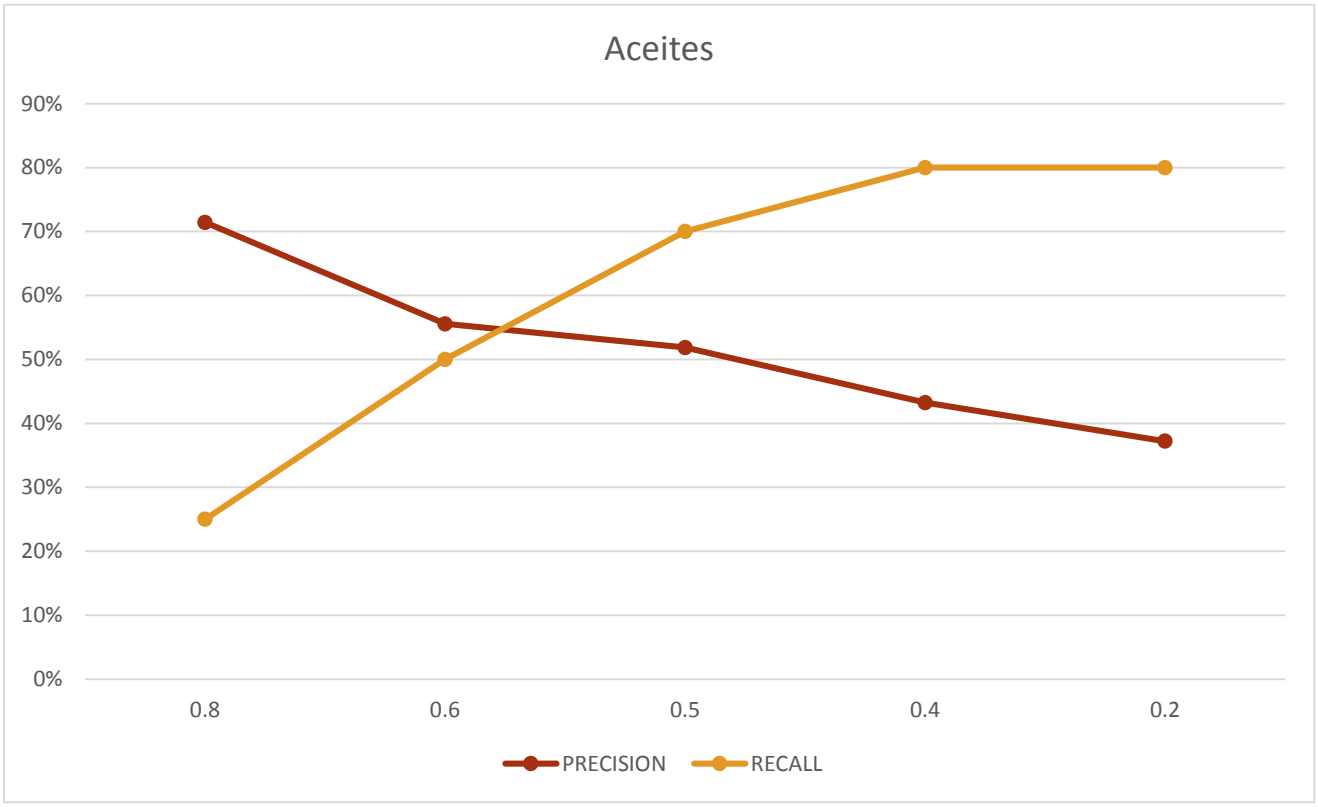


Figura 9. Aceites, fuzzy lookup básico

Análisis de resultados

Analizando la gráfica los resultados parecen bastante aceptables, aunque esto depende en gran medida de la necesidad de exactitud. Generalmente determinar dos registros como idénticos de manera errónea puede suponer un perjuicio considerable (el error se arrastra en el resto de

aplicaciones que usan la base de datos y se termina presentando a usuario dos cosas distintas como si fuesen lo mismo…). Esa necesidad de evitar los falsos positivos en la medida de lo posible hace que los resultados sean menos alentadores, en este caso la solución básica no realiza detecciones sin fallos (o al menos con un margen de fallos menos al 10%) con ninguno de los umbrales mostrados. Habría que subir el umbral mucho, probablemente acercarlo mucho al 1) para tener cierta seguridad de que no se van a generar falsos positivos, lo cual es un problema.

En esta grafica (*"Figura 9"*) se puede ver el dibujo típico que sigue el problema de la detección de duplicados, cuanto más estricto se es más aumenta la precisión pero disminuye el recall (número de idénticos que deberían identificarse) y viceversa, al definir un umbral más laxo el recall sube (pues da igual como de parecidos sean dos registros, simplemente con el que más se parezca vale) pero la precisión cae en picado (porque se está buscando un idéntico a todos los registros, sin importar si se parecen muy poco, por lo que hay muchos idénticos identificados y sin embargo la mayoría de los artículos no tienen idéntico en la tabla de referencia).

Tal y como se suponía anteriormente, realizar un mapeo difuso entre los atributos "marca" y "formato" lleva a resultados de falsos positivos de artículos con distinta marca o de distinto formato.

(Dist 1)(hojiblanca)(3 l.) aceite de oliva virgen extra //
(Dist 3)(HOJIBLANCA)(botella 1 l) aceite de oliva virgen extra (score 0,59304)

(Dist 1)(ybarra)(750 ml.) aceite de oliva virgen extra //
(Dist 3)(LA ESPAÑOLA)(botella 750 ml) aceite de oliva virgen extra (score 0,58125)

Generalmente la marca coincide exactamente en el origen y en la tabla de referencia, es un atributo que no suele tener diferencias entre los distintos orígenes de datos, tan solo algún error tipográfico de vez en cuando. El formato si suele presentar alguna diferencia más, aunque suele ser bastante similar. Debido a esto esos dos campos, la marca y el formato, no generan muchos falsos positivos en los que la marca o el formato sean el problema, sin embargo, el hecho de tener tres campos que se deben ponderar para determinar la similitud total, hace que cuando la marca y el formato son muy similares la similitud (o score) aumente y tenga un valor mayor al que tendría si se comparase tan solo el nombre.

(Dist 1)(carbonell)(1 l.) aceite de oliva virgen //
(Dist 3)(CARBONELL)(botella 1 l) aceite de oliva virgen extra (score 0,8904)

En este caso la marca coincide totalmente, lo que ha originado que el score total suba hasta superar el umbral del 0.8 cuando sin embargo la similitud del nombre es de 0,7555 por lo que de no ser por el "empujón" que le ha dado la marca el mapeo no habría superado el umbral del 0.8.

Así pues parece obvio que el siguiente paso sea procurar restringir la marca y el formato para evitar casos de idénticos con distinta marca o formato y para dar más peso al nombre.

En el siguiente apartado se analizara la sección de los chocolates.

4.3.2. Chocolates

En este apartado se presentan los resultados de la solución básica del "fuzzy lookup" en la categoría de chocolates (Carrefour contra Eroski) y se realiza un análisis de los mismos. A continuación se pueden observar los resultados:

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	17	19	41	**89%**	11%	**41%**	59%	0,57
0.6	28	46	41	**61%**	39%	**68%**	32%	0,64
0.5	31	55	41	**56%**	44%	**76%**	24%	0,65
0.4	31	67	41	**46%**	54%	**76%**	24%	0,57
0.2	31	82	41	**38%**	62%	**76%**	24%	0,50
Min Score	0,5402							

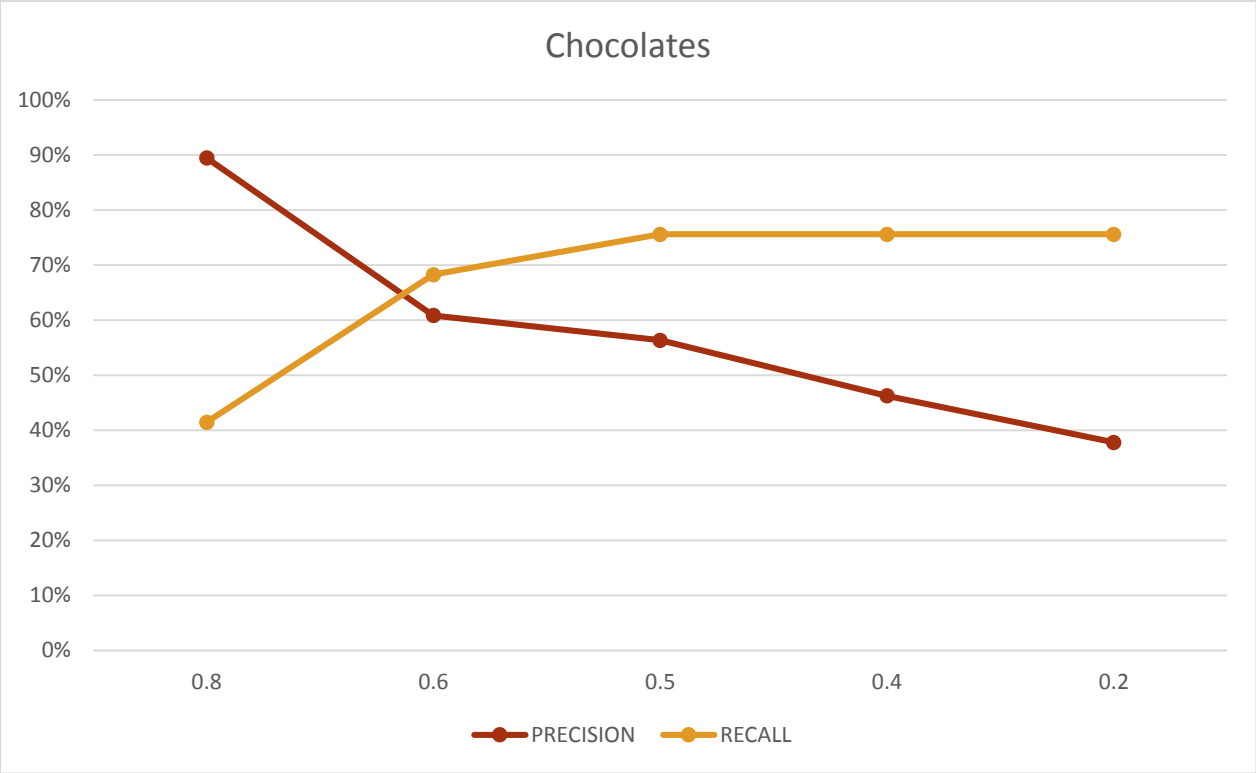


Figura 10. Chocolates, fuzzy lookup básico

Análisis de resultados

De nuevo, en esta categoría, se puede ver que en ningún momento se alcanza un 100% de precisión, aunque con un alto umbral se consigue un 90%. El problema de idénticos con distinta marca o formato vuelve a repetirse en esta categoría, hay mapeos que independientemente del

parecido de los nombres de los artículos es obvio que no son el mismo, pues su marca o formato es distinto. Sin embargo el sistema los identifica como idénticos:

(Dist 1)(valor)(250 g.) chocolate puro con avellanas //
(Dist 2)(VALOR)(tableta 200 g) Chocolate puro con avellanas (score 0,8911)

(Dist 1)(milka)(125 g.) chocolate extrafino con leche //
(Dist 2)(LINDT)(tableta 125 g) Chocolate extrafino con leche (score 0,75976)

Además de los problemas derivados de permitir mapear artículos con distinta marca y formato se puede ver como en ocasiones los datos en el origen imposibilitan detectar un falso positivo de forma automática:

(Dist 1)(valor)(300 g.) chocolate puro con leche //
(Dist 2)(VALOR)(tableta 300 g) Chocolate puro (score 0,78379)

Como se puede ver es difícil determinar si ambos artículos son el mismo o no. En el primero pone "chocolate puro con leche" lo cual parece ser una contradicción, es raro que sea chocolate puro y a la vez chocolate con leche. El segundo es chocolate puro así que es imposible determinar si son el mismo o no sin mirar las imágenes de los distribuidores.

En otros casos se catalogan dos ítems como idénticos aun cuando tienen una cualidad diferenciable que los distingue, sin embargo, al tratarse de un solo "*token*" distinto y tener la misma marca y el mismo formato el score es alto:

(Dist 1)(valor)(150 g.) chocolate puro con avellanas sin azúcar //
(Dist 2)(VALOR)(tableta 150 g) Chocolate puro con almendras sin azúcar (score 0,84288)

Como se puede ver uno tiene avellanas y el otro almendras, está claro que no son el mismo, pero tan solo difieren en una palabra. El sistema al no encontrar otro ítem más parecido (para que fuese más parecido tendría que haber en el otro distribuidor un ítem con el mismo nombre) se mapean los dos artículos como el mismo con un score bastante alto.

Se puede empezar a ver que determinadas palabras o cualidades de los artículos deberían ser menos tolerantes a discrepancias que otras. Es lógico pensar que un chocolate con almendras nunca podrá ser el mismo que uno con avellanas. Esta información puede ser útil en posteriores mejoras del sistema.

Por último, como se ve en la gráfica (*"Figura 10"*) de nuevo se observa un aumento en el recall conforme la precisión disminuye. Con un umbral muy bajo se consiguen identificar casi un 80% de los idénticos que debería haber, pero a costa de generar muchos falsos positivos.

En el siguiente apartado se analizaran los resultados obtenidos en la categoría de cereales.

4.3.3. Cereales

Como se puede ver en la tabla y en la gráfica esta categoría es distinta a las dos anteriores. A continuación pasaremos a analizar los datos (Carrefour contra ECI) y determinar el motivo de unos resultados tan anómalos.

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	1	1	34	**100%**	0%	**3%**	97%	0,06
0.6	6	10	34	**60%**	40%	**18%**	82%	0,27
0.5	10	16	34	**63%**	38%	**29%**	71%	0,40
0.4	14	25	34	**56%**	44%	**41%**	59%	0,47
0.2	27	53	34	**51%**	49%	**79%**	21%	0,62
Min Score	0,2846							

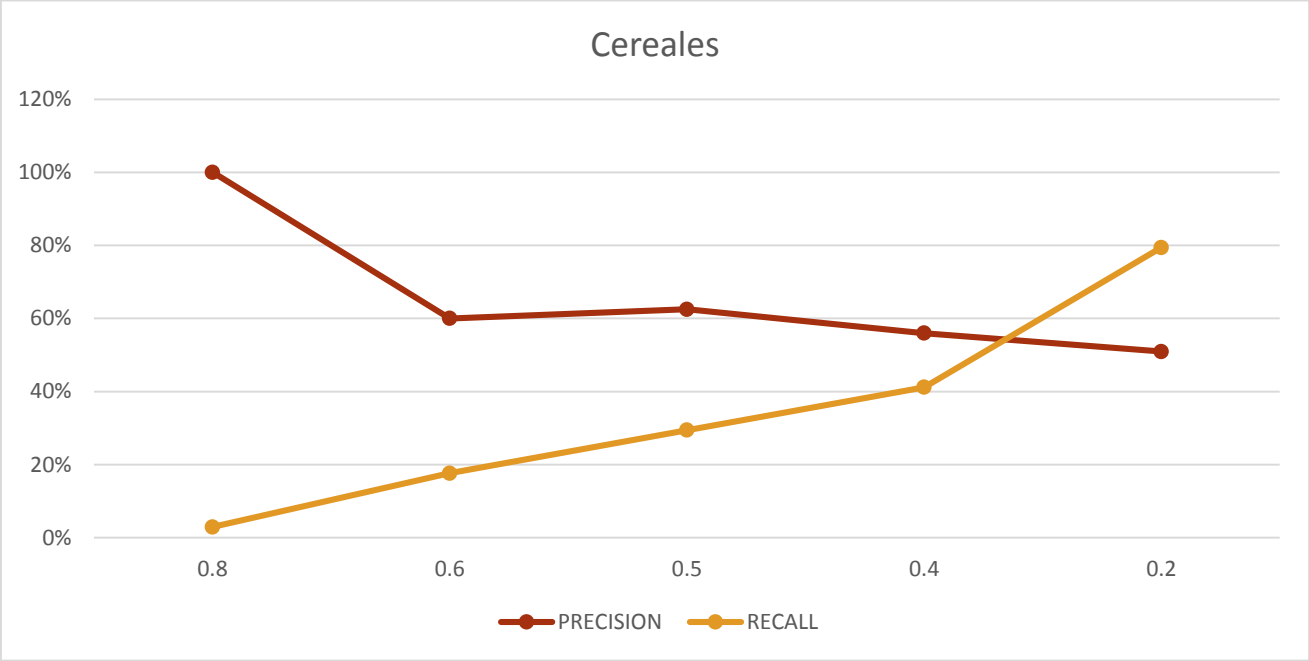


Figura 11. Cereales, fuzzy lookup básico

Análisis de resultados

En este caso sí que se consigue un 100% de precisión usando un umbral de 0.8. A simple vista esto parece positivo, sin embargo el recall es de un 3%, con este umbral el sistema tan solo ha identificado un idéntico, lo cual sirve para bastante poco.

Analizando los artículos en los distribuidores se ve que los datos son tan heterogéneos que la detección de idénticos por el sistema se dificulta en gran medida.

(Dist 1)(smacks - kellogg s)(500 g.) trigo inflado con azúcar, jarabe de glucosa y miel
(Dist 3)(KELLOGG'S SMACKS)(paquete 500 g) cereales de desayuno (score 0,28462)

En el ejemplo anterior se ve como dos artículos con nombres totalmente distintos son en realidad el mismo. En la muchos casos ocurre esto, los nombres en ambos distribuidores son tan distintos que su score es muy bajo, por eso un gran número de idénticos se acumulan en las zonas con umbral bajo.

Además del problema de la pésima calidad de los datos en el origen se siguen encontrando idénticos proporcionados por el sistema incorrectamente con marca o formato distinto.

(Dist 1)(nestlé - fitness)(600 g.) cereal con chocolate //
(Dist 3)(NESTLE FITNESS)(paquete 375 g) cereales de desayuno con chocolate (score 0,6652)

(Dist 1)(nestlé - nesquik)(625 g.) cereales de cacao //
(Dist 3)(NESTLE NESQUIK)(paquete 375 g) cereales de desayuno (score 0,5775)

(Dist 1)(special k - kellogg s)(300 g.) frutas rojas: copos tostados de arroz y trigo con frutas rojas liofilizadas //
(Dist 3)(ALIADA)(paquete 300 g) cereales de desayuno en copos tostados de arroz y trigo integral con frutos rojos (score 0,4051)

Sin embargo, en estos casos los score son relativamente bajos, por lo que no supone un gran problema.

Por último, una de las peculiaridades de esta sección más importantes es la de las submarcas. Algunos artículos incluyen la submarca en el atributo "marca", lo que dificulta la detección de idénticos pues en ocasiones un distribuidor emplea la marca en ese artículo pero otro emplea la submarca.

(Dist 1)(nestlé)(375 g.) copos de trigo integral con chocolate negro //
(Dist 3)(NESTLE FITNESS)(paquete 375 g) cereales de desayuno integrales con chocolate negro (score 0,65446)

En este ejemplo se puede ver como dos artículos a priori idénticos no lo son pues pertenece uno a la marca (que se puede considerar además como una submarca) "nestle" y el otro a la submarca de nestle "nestle fitness". En el fondo se deberían considerar marca y submarca como si fuesen dos marcas distintas, aunque esto plantea un problema de difícil solución, cuando un distribuidor incluye la submarca pero otro no.

4.4. Fuzzy Lookup "acotado"

En este caso se realiza una mejora al algoritmo simple del "fuzzy lookup básico". Como se ha visto no tiene sentido que el sistema identifique idénticos con distinta marca y formato. Aprovechando que se tratan de atributos que no suelen presentar grandes diferencias entre los distribuidores se va a suponer que se ha realizado una limpieza de marcas y formatos previa a la fase de detección de duplicados que se encarga de normalizar dichos campos. Para realizar las pruebas se llevará a cabo una limpieza de los campos "marca" y "formato" manual.

Así pues se puede exigir coincidencia absoluta en el fuzzy lookup. De este modo acotamos los artículos de entre los que buscara idénticos el fuzzy lookup y podremos obtener el artículo más

parecido en la tabla de referencia que tenga la misma marca y el mismo formato que el artículo de origen.

El proceso de detección de duplicados es exactamente el mismo que en el caso anterior, la única diferencia se encuentra en el proceso del fuzzy lookup, donde se configura el mapeo entre los campos “marca” y “formato” para que sea una coincidencia exacta y no una aproximación difusa.

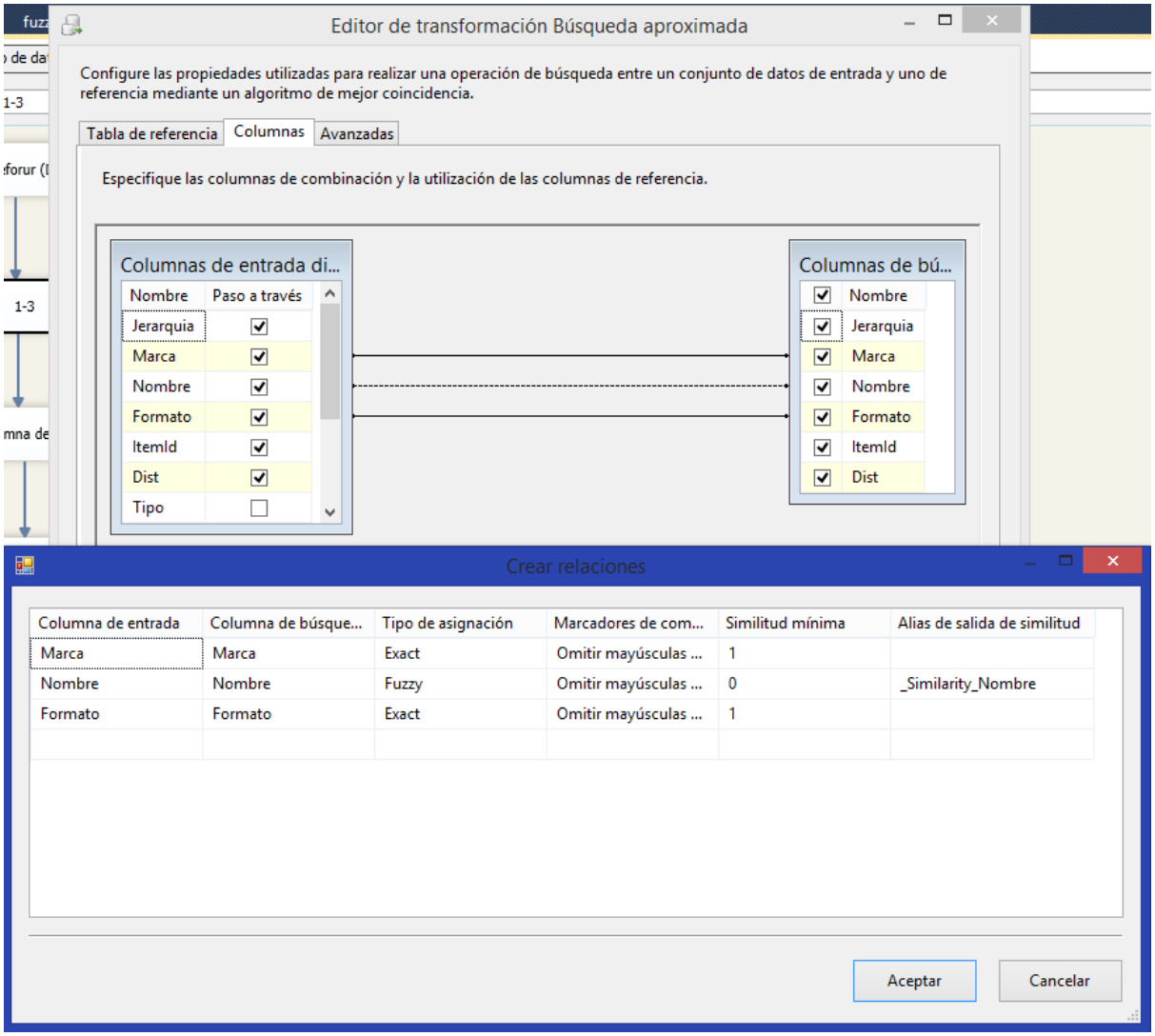


Figura 12. Fuzzy lookup acotado

Como se ha visto en el “fuzzy lookup básico” se dividirán los resultados entre las categorías de aceites, chocolates y cereales. En las siguientes secciones se pueden ver los resultados y un análisis de los mismos.

4.3.1. Aceites

En este caso se realiza una búsqueda difusa entre Carrefour y ECI. A simple vista ya se puede observar cierta mejora con respecto a la categoría de aceites del "fuzzy lookup básico".

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	6	6	20	**100,00%**	0,00%	**30,00%**	70,00%	0,46
0.6	8	10	20	**80,00%**	20,00%	**40,00%**	60,00%	0,53
0.5	9	12	20	**75,00%**	25,00%	**45,00%**	55,00%	0,56
0.4	11	14	20	**78,57%**	21,43%	**55,00%**	45,00%	0,65
0.2	17	24	20	**70,83%**	29,17%	**85,00%**	15,00%	0,77
Min Score	0,2049							

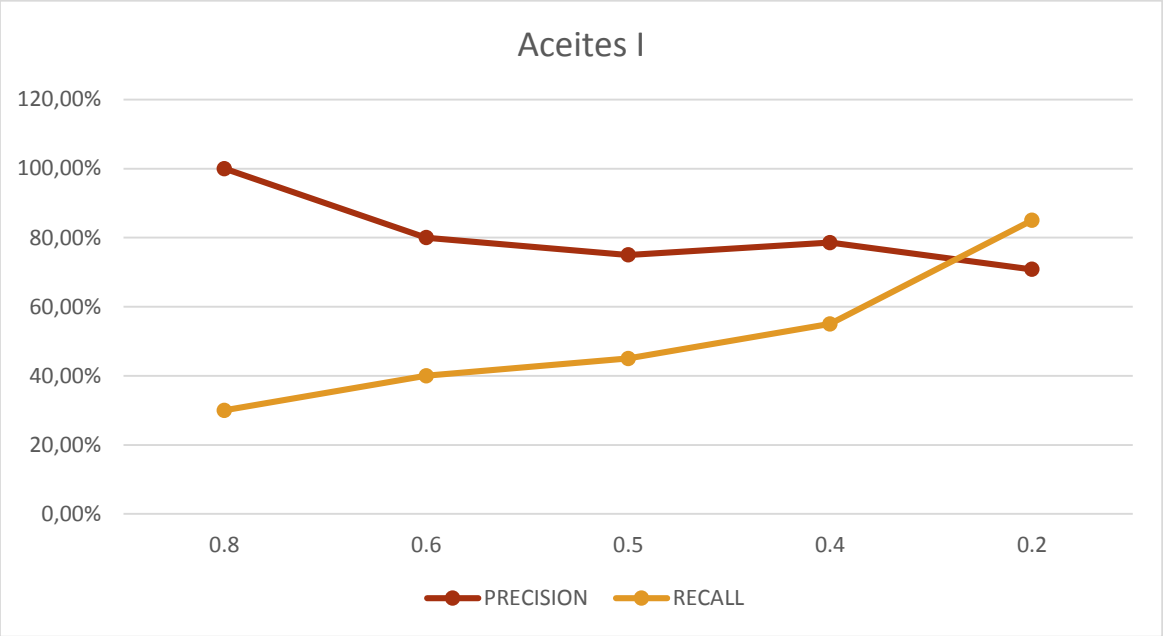


Figura 13. Aceites I, fuzzy lookup acotado

Análisis de resultados

Una vez evitado el problema de idénticos con distinta marca o formato se puede ver que con un umbral del 0.8 se obtiene un 100% de precisión. Además de conseguir no generar ni un solo falso positivo el sistema ha identificado el 30% de los mapeos que debería haber identificado. Resueltos estos primeros problemas se puede pasar a tratar otros.

El problema que se ve tras esta prueba es la disparidad en los datos entre distintos distribuidores originan situaciones en las que dos ítems con nombres muy parecidos eran distintos:

(Dist 1)(carbonell)(1 l) aceite de oliva virgen //
(Dist 3)(CARBONELL)(1 l) aceite de oliva virgen extra (score ***0,759296775****)*

O ítems que son el mismo pero en un distribuidor hay más información que en el otro, por lo que el score es bajo:

(Dist 1)(coosur)(1 l) aceite de oliva virgen extra sabor intenso variedad picual //
(Dist 3)(COOSUR)(1 l) aceite de oliva virgen extra picual intenso (socore 0,296644926)

En el primer caso el hecho de que un aceite sea extra o no es suficiente para que dos ítems sean distintos, pero la omisión de una palabra nada mas (aunque sea de las poco frecuentes) no disminuye el score lo suficiente. Y en el segundo caso se puede ver como un distribuidor a incluido información extra con palabras que en principio no hacen distinguible a un ítem de otro (sabor, variedad…). Sin embargo en este caso se han omitido dos palabras poco comunes (posiblemente menos comunes aún que extra) y el score cae en picado.

Otro problema que se encontró tiene que ver con las distintas formas de representar ciertos datos que tienen los distribuidores, en concreto había ítems que tenían el símbolo "º" separado del número y otros no. Esto hacia que en unos casos se considerasen el número y el símbolo como *"tokens"* distintos y en otros no, lo que reducía la calidad de los resultados:

```
(Dist 1)(coosur)(1 l) aceite de oliva 1 º sabor intenso //
(Dist 3)(COOSUR)(1 l) aceite de oliva intenso 1º (score 0,204980955)
```

Debido a esto se pensó en proporcionar un pre-procesamiento a los datos para procurar mejorar los resultados. En concreto se normalizo el uso del símbolo "º" dejándolo siempre pegado al número y se eliminaron las palabras *"de", "para", "y", "con", "variedad", "sabor" y "refinado"*.

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	12	12	20	**100,00%**	0,00%	**60,00%**	40,00%	0,75
0.6	15	17	20	**88,24%**	11,76%	**75,00%**	25,00%	0,81
0.5	16	19	20	**84,21%**	15,79%	**80,00%**	20,00%	0,82
0.4	16	19	20	**84,21%**	15,79%	**80,00%**	20,00%	0,82
0.2	18	25	20	**72,00%**	28,00%	**90,00%**	10,00%	0,80
Min Score	0,2769							

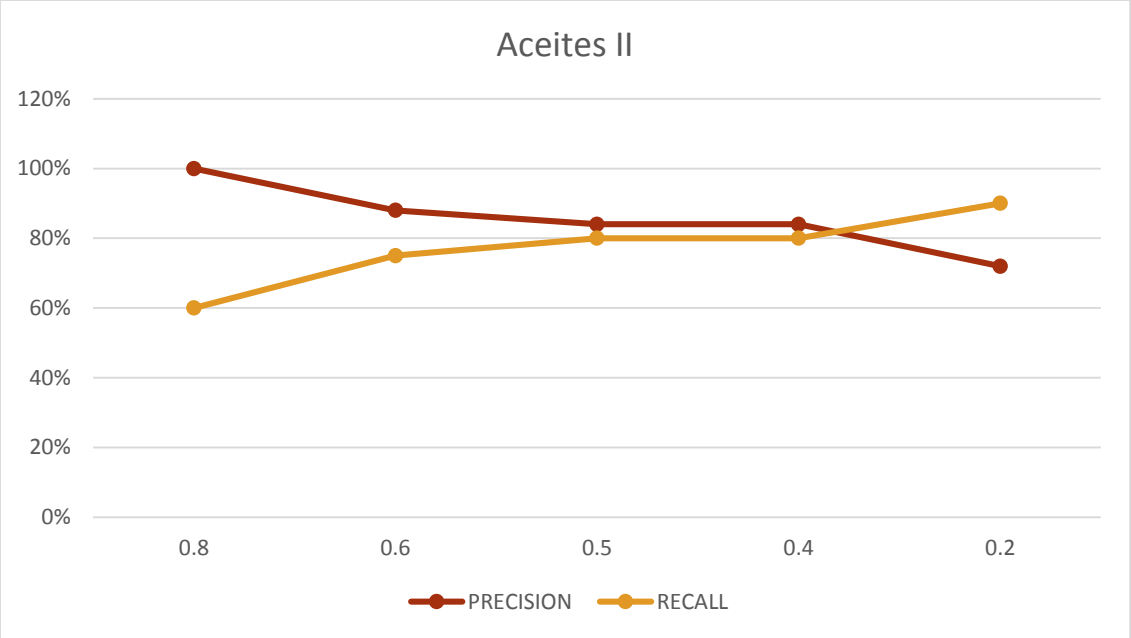


Figura 14. Aceites II, fuzzy lookup acotado

Se puede ver que ahora la relación entre los artículos que ha catalogado correctamente y los idénticos reales (CC/IR) aumenta más rápido al ir bajando el umbral. Esto es porque ítems catalogados correctamente con score muy bajo ahora han subido su score por lo que "salen a la luz" antes (sin necesidad de bajar el umbral tanto como antes).

Por ejemplo, el último caso de ítem con score bajo que hemos visto:

"(Dist 1)(coosur)(1 l) aceite de oliva 1 º sabor intenso //
(Dist 3)(COOSUR)(1 l) aceite de oliva intenso 1º (score 0,204980955)"

Pasa a una posición más "alta" (con más escore) ya que ahora tan solo difieren un ítem del otro en el orden de dos *"tokens"*:

(Dist 1)(coosur)(1 l) aceite oliva 1º intenso //
(Dist 3)(COOSUR)(1 l) aceite oliva intenso 1º (score 0,980000019)

En el otro caso similar ocurre lo mismo:

(Dist 1)(coosur)(1 l) aceite oliva virgen extra intenso picual //
(Dist 3)(COOSUR)(1 l) aceite oliva virgen extra picual intenso (score 0,980000019)

Sin embargo esta mejora no evita que los ítems con nombres muy parecidos que en realidad son distintos salgan como idénticos con score muy alto.

4.3.2. Chocolates

La siguiente categoría en la que se ha lanzado la prueba es en las de las tabletas de chocolate (Carrefour contra Eroski). De nuevo se puede apreciar la mejora a simple vista.

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	16	16	41	**100,00%**	0,00%	**39,02%**	60,98%	0,56
0.6	19	24	41	**79,17%**	20,83%	**46,34%**	53,66%	0,58
0.5	27	35	41	**77,14%**	22,86%	**65,85%**	34,15%	0,71
0.4	29	44	41	**65,91%**	34,09%	**70,73%**	29,27%	0,68
0.2	33	56	41	**58,93%**	41,07%	**80,49%**	19,51%	0,68
Min Score	0,0440							

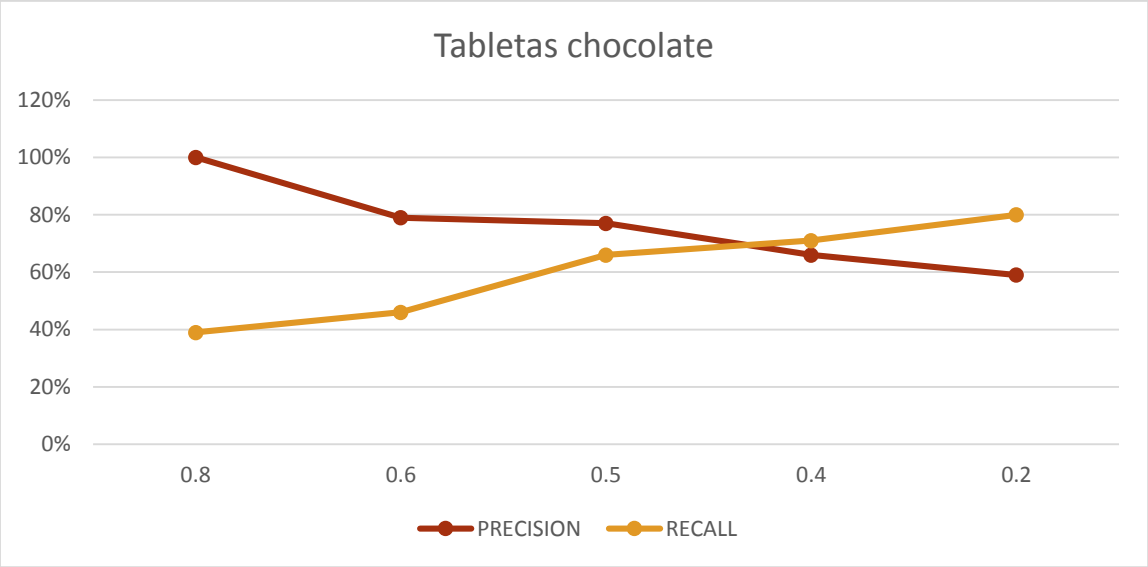


Figura 15. Chocolates, fuzzy lookup acotado

Análisis de resultados

En chocolates se pueden observar varios problemas. Primero, en cuanto a los "falsos positivos", tal y como se ha visto en la primera solución propuesta del fuzzy lookup en ocasiones los ítems tienen nombres en esencia iguales, tan solo varían una o dos palabras que son las que hacen referencia al sabor, por lo que como el algoritmo muestra lo más parecido si no hay un ítem con ese mismo sabor propondrá otro (con misma marca y formato) con otro sabor, y al solo variar un *"token"* el score no será muy bajo.

(Dist 1)(valor)(150 g) chocolate puro con avellanas sin azúcar //
(Dist 2)(VALOR)(150 g) Chocolate puro con almendras sin azúcar (score 0,785583615)

(Dist 1)(nestlé)(200 g) chocolate con leche y frutos secos //
(Dist 2)(NESTLÉ)(200 g) Chocolate negro con frutos secos (score 0,662446141)

(Dist 1)(milka)(300 g) cholate relleno de fresa //
(Dist 2)(MILKA)(300 g) Chocolate relleno de trufa-almendra (score 0,636274636)

(Dist 1)(valor)(300 g) chocolate puro con leche //
(Dist 2)(VALOR)(300 g) Chocolate puro (score 0,630470395)

(Dist 1)(valor)(250 g) chocolate puro con avellanas //
(Dist 2)(VALOR)(250 g) Chocolate puro con almendras (score 0,614168823)

Una forma de poder solucionar esto sería usar la idea del post-procesado usando atributos y dominios. Así podríamos bajar el umbral con el que aceptar automáticamente un idéntico pero después de pasar ese umbral se debería cumplir la condición de que si un ítem tiene un atributo con un valor del dominio de ese atributo el otro debería tener o ningún valor o el mismo. Con esto, en el caso de 0.6 como umbral por ejemplo, **de los 5 ítems que ha catalogado incorrectamente** (24-19, los mostrados anteriormente) **4 no pasarían el post-procesado** suponiendo que disponemos de los atributos "sabor" y "suavidad" con los dominios {avellanas, almendras, frutos secos, trufa-almendra, fresa} y {leche, negro} respectivamente.

De este modo la nueva proporción entre catalogados correctamente e idénticos catalogados (CC/IC) será 19/20 = **95% en vez del 79%.**

El problema de esto sería el estudio previo de cada jerarquía para establecer los atributos y sus dominios y el hecho de que si no los defines bien podrían catalogarse automáticamente muchos ítems que no deberían al haber establecido un umbral relativamente bajo.

En cuanto al problema de artículos que son idénticos pero tienen un score muy bajo podemos ver que uno de los problemas concretos de esta jerarquía es el uso "indiscriminado" de la palabra "extrafino" en un distribuidor (se usa en todas las marcas cuando realmente solo usa el termino Nestlé).

(Dist 1)(milka)(125 g) chocolate extrafino con leche y avellanas troceadas //
(Dist 2)(MILKA)(125 g) Chocolate con leche-avellanas (score 0,277355313)

En este caso además del problema de "extrafino" existe la dificultad añadida de que en un distribuidor usen "leche y avellanas" y en otro "leche-avellanas" (en un caso serían 3 *"tokens"* y en el otro solo 1…), esto junto con la palabra "troceadas" que tiene uno y no el otro hace que el score baje mucho.

(Dist 1)(milka)(125 g) chocolate extrafino con leche //
(Dist 2)(MILKA)(125 g) Chocolate con leche (score 0,361326307)

En este caso el problema de "extrafino" por si solo hace que el score baje. Quizás se podría reducir el impacto de este problema eliminando la palabra "extrafino" del nombre del ítem si la submarca "nestle extrafino" estuviese en el campo "marca" del ítem, pero es un problema muy característico de los chocolates que no surge hasta que no tratas esa jerarquía en concreto.

Además, en ocasiones los datos que vienen del distribuidor hacen casi imposible identificar su idéntico:

(Dist 1)(nestlé)(3x150 g) chocolate con leche extrafino //
(Dist 2)(NESTLÉ)(3x150 g) Chocolate (score 0,04408823)

4.3.3. Cereales

En este caso se realiza la prueba del "fuzzy lookup acotado" en la jerarquía de cereales (de nuevo Carrefour contra ECI). Como se puede ver, ahora los resultados son incluso más anómalos que antes.

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	0	0	34	**NA**	NA	**0,00%**	100,00%	NA
0.6	5	5	34	**100,00%**	0,00%	**14,71%**	85,29%	0,26
0.5	6	6	34	**100,00%**	0,00%	**17,65%**	82,35%	0,30
0.4	8	8	34	**100,00%**	0,00%	**23,53%**	76,47%	0,38
0.2	26	26	34	**100,00%**	0,00%	**76,47%**	23,53%	0,87
Min Score	0,0125							

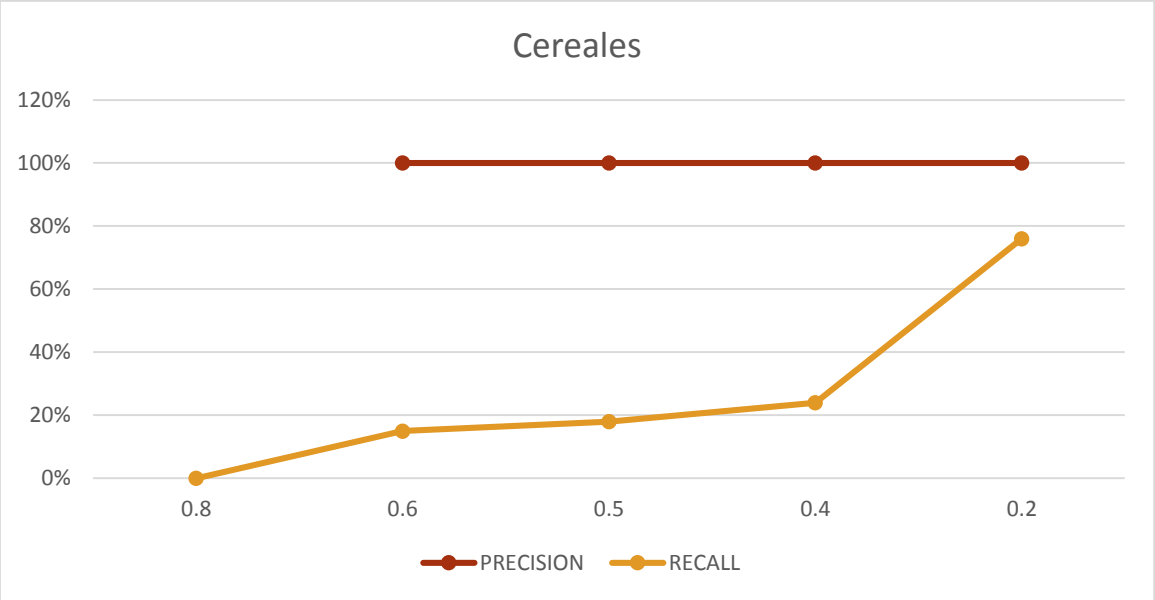


Figura 16. Cereales, fuzzy lookup acotado

Análisis de resultados

Con cereales se da un caso muy concreto derivado de la gran disparidad de datos entre los distintos distribuidores. Esto hace que los nombres de los ítems sean muy muy distintos entre sí por lo que ni siquiera hay propuestas de catalogación con 0.8 de score. Esto unido al hecho de que ambos distribuidores tenían las submarcas en el campo "marca" (aunque hubo que normalizarlas a mano) ha hecho que un ítem de una submarca y con un formato concreto tenga muy pocos posibles idénticos en el otro distribuidor con misma submarca y formato (en ocasiones tan solo había una opción posible) por lo que las propuestas de idénticos son correctas aunque el score sea muy bajo (generalmente una submarca de cereales con un formato concreto tiene muy pocos artículos distintos, uno o dos, por lo que si un distribuidor tiene un ítem con una submarca y un formato y el otro distribuidor también es muy probable que sean el mismo, por lo menos en cereales).

4.5. Fuzzy Lookup "plus"

La mejora sobre las pruebas anteriores consiste en definir una serie de atributos con dominios determinados en cada categoría (jerarquía). Dichos atributos se incluyen en la búsqueda fuzzy para procurar mejorar el score de los mapeos.

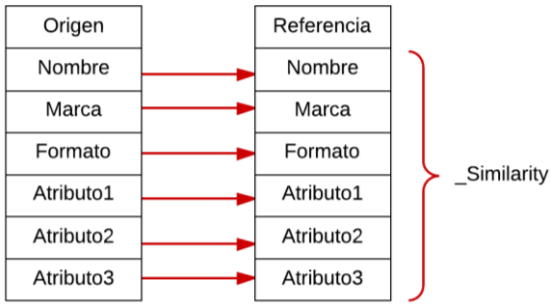


Figura 17. Mejora fuzzy lookup plus

Para representar esto se ha realizado un análisis de los datos manualmente para determinar que atributos podrían propiciar una mejora. Una vez identificados se ha añadido un campo en la tabla por cada atributo y se ha ido artículo por artículo rellenando dichos atributos si correspondía.

En cuanto a los procesos del sistema son los mismos, sin embargo ahora se tienen más campos que comparar en el proceso del fuzzy lookup. Se ha optado por la opción más sencilla, se mapean los nuevos campos entre sí sin determinar ninguna similitud mínima y realizando un mapeo difuso.

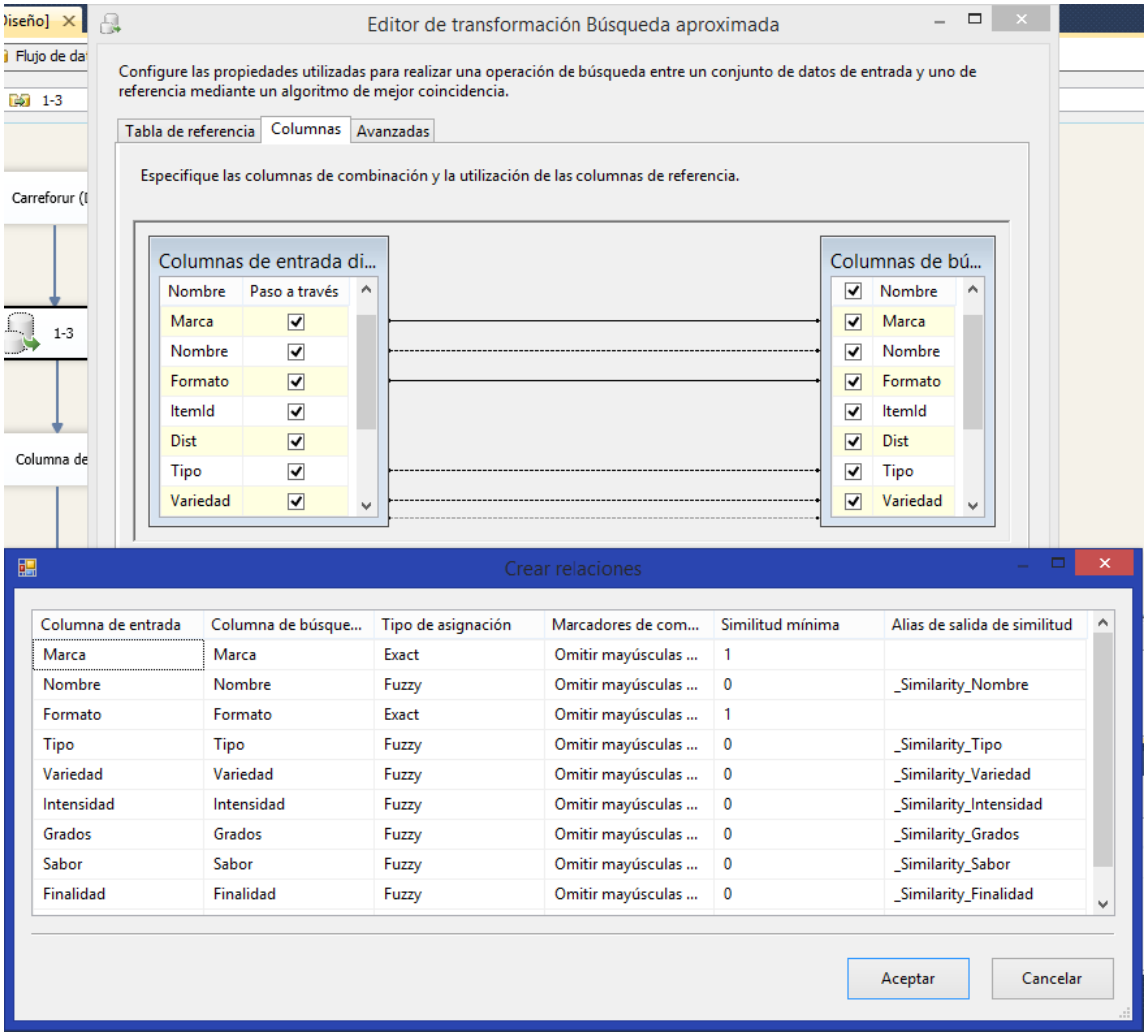


Figura 18. Fuzzy lookup plus

En los siguientes apartados se pueden ver los resultados separados según las tres jerarquías: aceites, chocolates y cereales. Además cada apartado incluirá una lista con los atributos definidos para la mejora del "fuzzy lookup plus" así como el dominio de cada uno de dichos atributos.

La preparación previa de los datos ha consistido en procurar determinar para cada atributo un valor de entre el dominio en todos los artículos tanto de la tabla de referencia como de la tabla de origen.

4.3.1. Aceites

Los atributos definidos para esta jerarquía, junto con sus dominios son:

- Tipo: oliva virgen, oliva virgen extra, girasol, maíz, oliva.
- Variedad: arbequina, albahaca, picual, hojiblanca, olivares.
- Intensidad: suave, intenso.

- Grados: 0.4º, 1º, 0.2º
- Sabor: ajo, guindilla.
- Finalidad: tostadas, repostería, cocina mediterránea, ensaladas pastas, plancha, freir.

Fuzzy lookup plus entre Carrefour (dist1) y ECI (dist3).

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	6	6	20	**100,00%**	0,00%	**30,00%**	70,00%	0,46
0.6	8	10	20	**80,00%**	20,00%	**40,00%**	60,00%	0,53
0.5	9	12	20	**75,00%**	25,00%	**45,00%**	55,00%	0,56
0.4	11	14	20	**78,57%**	21,43%	**55,00%**	45,00%	0,65
0.2	17	24	20	**70,83%**	29,17%	**85,00%**	15,00%	0,77
Min Score	0,2769							

Umbral	Correctos	Identificados	Reales	PRECISION(+)	FP(+)	RECALL(+)	NI(+)	*f-measure(+)*
0.8	13	13	20	**100,00%**	0,00%	**65,00%**	35,00%	0,79
0.6	16	18	20	**88,89%**	11,11%	**80,00%**	20,00%	0,84
0.5	16	19	20	**84,21%**	15,79%	**80,00%**	20,00%	0,82
0.4	17	20	20	**85,00%**	15,00%	**85,00%**	15,00%	0,85
0.2	18	24	20	**75,00%**	25,00%	**90,00%**	10,00%	0,82
Min Score	0,3867							

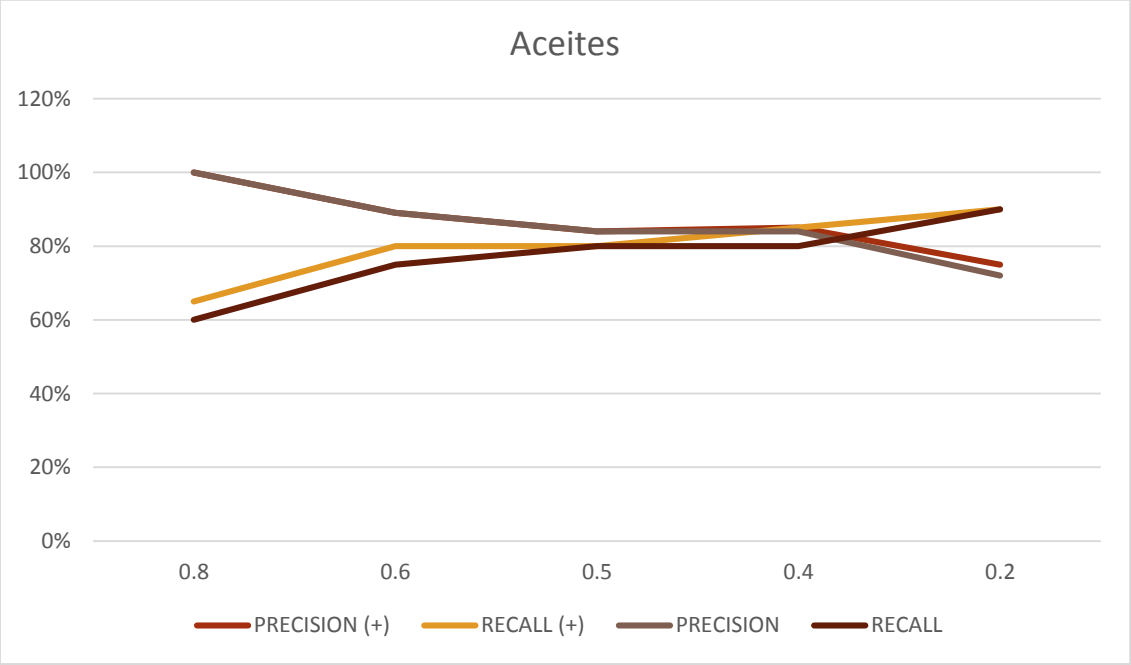


Figura 19. Aceites, fuzzy lookup plus

Análisis de resultados

Se ve que las variaciones son mínimas, apenas hay diferencia en los falsos positivos ya que aunque aumente en uno el número de identificados correctamente también aumenta en uno el numero di identificados incorrectamente (umbral 0.6 y 0.4 por ejemplo). Sí que hay ciertos casos donde el score (o _Similarity) mejora levemente y el ítem consigue pasar el umbral cuando antes no lo pasaba. Con umbral 0.8 por ejemplo, el ítem:

(Dist 1)(koipe)(1 l) aceite oliva 1º intenso //
(Dist 3)(KOIPE)(1 l) sabor aceite oliva intenso 1º

Con el "fuzzy lookup plus" tiene un score de 0.844823778 mientras que antes tenía un score de 0.7788202276. El incremento es muy leve pero lo suficiente para que si se estableciese un umbral de 0.8 se identificaría un idéntico más que antes y se seguiría manteniendo los falsos positivos a 0.

Con el resto de umbrales sucede lo mismo, hay ítems que consiguen superar un umbral que antes no podían por poco gracias a una leve mejora en su score, sin embargo los ítems incorrectos de ese umbral no disminuyen su score (de echo pude aumentar muy muy levemente ya que aunque ambos ítems no tengan el mismo valor en un atributo este puede ser lo suficientemente parecido para que el score total aumente). Por ejemplo:

(Dist 1)(carbonell)(1 l) aceite oliva virgen //
(Dist 3)(CARBONELL)(1 l) aceite oliva virgen extra

Estos ítems tienen un score de 0.755710721 con el "fuzzy lookup plus" cuando antes tenían un score de 0.755540894 (que coincide con la similitud del campo "nombre" de ambos ítems). Esto ocurre porque la similitud entre los atributos "tipo" de los ítems ("oliva virgen" y "oliva virgen extra") es de 0.755890369 que al ser mayor que la similitud que tiene el campo "nombre" hace que la similitud total suba un poco.

4.3.2. Chocolates

Los atributos y dominios en este caso son:

- Tipo: negro, leche, puro, blanco, mousse
- Complemento: naranja, almendra, kikos, fresa, trufa almendra, frutos secos, 3 chocolates, luflee, queso fresa, lacasitos, caramelo avellana, caramelo, tuc, galleta, dulce de leche, dulce, oreo.
- Cacao: 70%, 52%.
- Azúcar: sin azúcar.

Items de entrada de Carrefour, items de referencia Eroski:

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	16	16	41	**100,00%**	0,00%	**39,02%**	60,98%	0,56
0.6	19	24	41	**79,17%**	20,83%	**46,34%**	53,66%	0,58
0.5	27	35	41	**77,14%**	22,86%	**65,85%**	34,15%	0,71
0.4	29	44	41	**65,91%**	34,09%	**70,73%**	29,27%	0,68
0.2	33	56	41	**58,93%**	41,07%	**80,49%**	19,51%	0,68
Min Score	0,0440							

Umbral	Correctos	Identificados	Reales	PRECISION(+)	FP(+)	RECALL(+)	NI(+)	*f-measure(+)*
0.8	17	17	41	**100,00%**	0,00%	**41,46%**	58,54%	0,59
0.6	23	25	41	**92,00%**	8,00%	**56,10%**	43,90%	0,70
0.5	28	36	41	**77,78%**	22,22%	**68,29%**	31,71%	0,73
0.4	32	43	41	**74,42%**	25,58%	**78,05%**	21,95%	0,76
0.2	36	57	41	**63,16%**	36,84%	**87,80%**	12,20%	0,73
Min Score	0,0365							

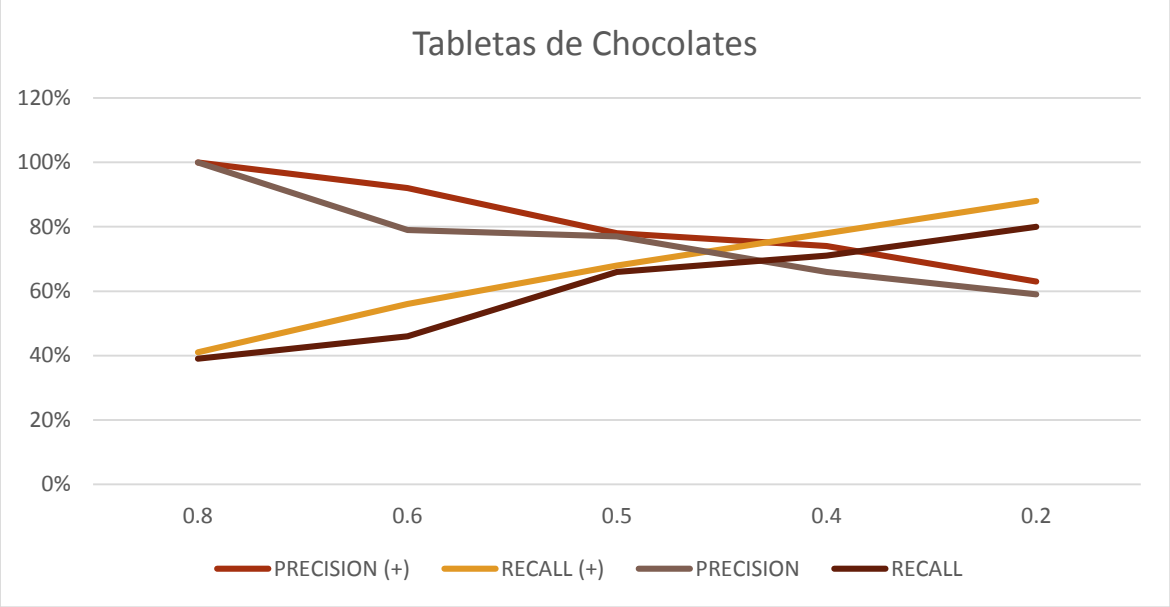


Figura 20. Chocolates, fuzzy lookup plus

Análisis de resultados

En este caso se ven más diferencias entre la mejora "fuzzy lookup plus" y el "fuzzy lookup simple". Se ve como las líneas de las nuevas pruebas se encuentran siempre por encima de las de las pruebas anteriores (mas es mejor).

A diferencia de la jerarquía de aceites se ve que en el caso del umbral 0.6, por ejemplo, no solo aumentan el número de ítems catalogados correctamente que entran en ese umbral, sino que hay ítems que estaban mal catalogados en ese umbral que ya no están (5 incorrectos antes, 2 ahora).

Esto ocurre porque, casualmente, los ítems tienen similarity muy baja en los nuevos atributos y se consigue que el score final baje visiblemente hasta quedar fuera del umbral en el que estaba antes. Por ejemplo:

```
(Dist 1)(valor)(250 g) chocolate puro con avellanas //
(Dist 2)(VALOR)(250 g) Chocolate puro con almendras
```

Con la nueva versión estos ítems tienen un score (o _Similarity) total de 0.583399653 mientras que antes tenía 0.614168823 (score del campo "nombre"). Se ha conseguido pues que estos ítems mal catalogados salgan del umbral 0.6 y pasen al 0.5 reduciendo asi los falsos positivos del umbral 0.6.

Esto ocurre porque el atributo “complemento” (“avellanas” y “almendras”) tiene un score muy bajo (0.0125) y hace que al ponderar la similaridad total baje lo suficiente para salir del umbral 0.6.

4.3.3. Cereales

Atributos y dominios:

- Tipo: arroz, trigo, avena, arroz y trigo, muesli, avena trigo y arroz, arroz trigo cebada, maíz
- Sabor: chocolate, chocolate y chocolate blanco, chocolate blanco, chocolate y caramelo, yogur, vainilla, fresa, chocolate avellana, frambuesa manzana, chocolate fresa, almendra.
- Complemento: fibra, fruta, miel, integral, fruta fibra, fruta nueces, frutos rojos, frutos secos, miel azúcar, fruta integral, azúcar moreno, fruta frutos secos, cacahuetes miel.

Fuzzy lookup realizado con articulos de entrada de Carrefour y articulos de referencia de ECI.

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	*f-measure*
0.8	0	0	34	**NA**	NA	**0,00%**	100,00%	NA
0.6	5	5	34	**100,00%**	0,00%	**14,71%**	85,29%	0,26
0.5	6	6	34	**100,00%**	0,00%	**17,65%**	82,35%	0,30
0.4	8	8	34	**100,00%**	0,00%	**23,53%**	76,47%	0,38
0.2	26	26	34	**100,00%**	0,00%	**76,47%**	23,53%	0,87
Min Score	0,0125							

Umbral	Correctos	Identificados	Reales	PRECISION(+)	FP(+)	RECALL(+)	NI(+)	*f-measure(+)*
0.8	1	1	34	**100,00%**	0,00%	**2,94%**	97,06%	0,06
0.6	5	5	34	**100,00%**	0,00%	**14,71%**	85,29%	0,26
0.5	6	6	34	**100,00%**	0,00%	**17,65%**	82,35%	0,30
0.4	8	8	34	**100,00%**	0,00%	**23,53%**	76,47%	0,38
0.2	25	25	34	**100,00%**	0,00%	**73,53%**	26,47%	0,85
Min Score	0,0125							

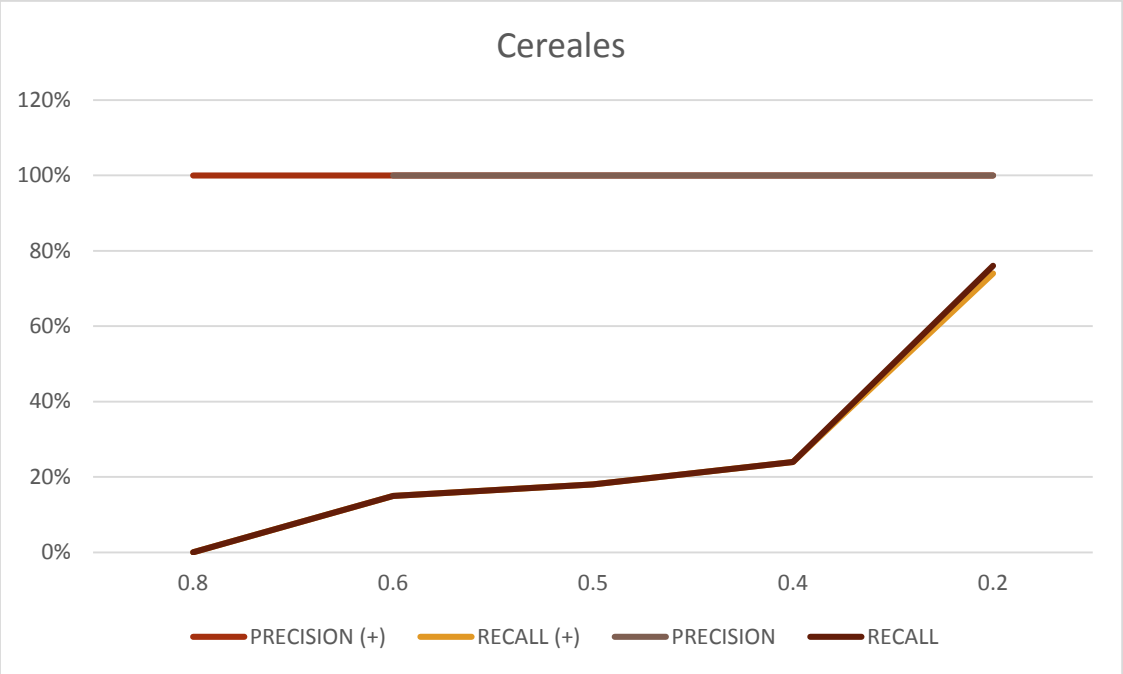


Figura 21. Cereales, fuzzy lookup plus

Análisis de resultados

De nuevo las diferencias son inapreciables (por no decir inexistentes). Los datos en este caso en concreto y el hecho de que las submarcas se encuentren definidas en el campo "marca" hace que este caso sea especial, ya que los posibles ítems donde buscar el idéntico son muy pocos y como se ve en la gráfica (siempre 100% de acierto) esto hace que se acierte siempre por lo que usar atributos para reducir falsos positivos no mejora nada.

4.5. Calidad de los datos

El principal factor determinante a la hora de obtener unos buenos resultados con un sistema automático de detección de duplicados es la calidad de los datos de entrada. Hay que ser conscientes de que pueden existir casos que resulten imposibles de identificar, en ocasiones ni siquiera por un humano experto.

Como ya se ha visto en apartados anteriores es más habitual de lo que cabría esperar encontrarse con artículos con nombres completamente distintos, o incluso con algún adjetivo que hace parecerlos distintos, que son en realidad el mismo. En el caso real que se está tratando en este trabajo (datos extraídos de supermercados) generalmente existe distinta "precisión" entre los distintos distribuidores. En uno los artículos están descritos en el nombre con más detalle que en otro, lo que hace que en ocasiones un "chocolate" sea igual que "chocolate con leche extrafino", por ejemplo. Generalmente hace falta acudir a las imágenes de los distribuidores para intentar determinar si ambos artículos son el mismo e incluso en ocasiones la imagen no es la misma (reediciones del empaquetamiento del producto) pero realmente son el mismo artículo.

No se pretende que el sistema sea capaz de identificar duplicados que un humano no es capaz de identificar. Por esto, se ha realizado un análisis de los duplicados reales para determinar cuáles de ellos se pueden considerar imposibles de identificar debido a que para su mapeo manual

ha sido necesaria información extra (comparación de imágenes). De este modo, se pueden reajustar los datos obtenidos de las pruebas para poder compararlos mejor con los duplicados que habría podido identificar un humano haciendo uso de los mismos datos.

A continuación se pueden observar algunos ejemplos de duplicados imposibles de determinar sin recurrir a las imágenes de las fuentes:

aceite de oliva virgen	aceite de oliva virgen Selección Almazara
aceite de oliva virgen extra	aceite de oliva virgen extra Gran Selección
choco swing cookie	Chococookie
chocolate puro con leche	Chocolate con leche
chocolate leche mousse avellana	Mousse sin azùcar con avellanas
aceite de oliva virgen extra spray plancha	aceite de oliva para plancha
chocolate extrafino con leche relleno de almendras	Chocolate relleno de trufa-almendra
plus: salvado de trigo	Plus cereales de desayuno
chocolate con leche: copos tostados de arroz y trigo con virutas de chocolate con leche	cereales de desayuno con chocolate con leche
classic: copos tostados de arroz, trigo y cebada	Classic cereales de desayuno
aceite refinado de girasol premium	Aceite de girasol especial para freir
flakes: copos tostados de trigo integral enriquecidos con salvado de trigo	Flakes cereales de desayuno
arroz tostado	cereales de desayuno
copos de maíz tostados y azucarados	cereales de desayuno
chocolate con leche extrafino	Chocolate
cereales línea fitness	Original cereales de desayuno integrales formato ahorro

Esos casos son tan solo un ejemplo. En total, de 95 duplicados reales, 30 se han considerado como no relevantes para el sistema (ya que un humano tendría demasiadas dudas sin poder comparar las imágenes del artículo como para identificarlos como duplicados o directamente creería que son distintos). De esos 30, 7 corresponden a aceites, 7 a chocolates y 16 a cereales.

Tomando como partida los resultados obtenidos en la prueba del "fuzzy lookup plus" se han reajustado los duplicados reales, los cuales pasan a tomar los valores: 13 (aceites), 34 (chocolates) y 18 (cereales).

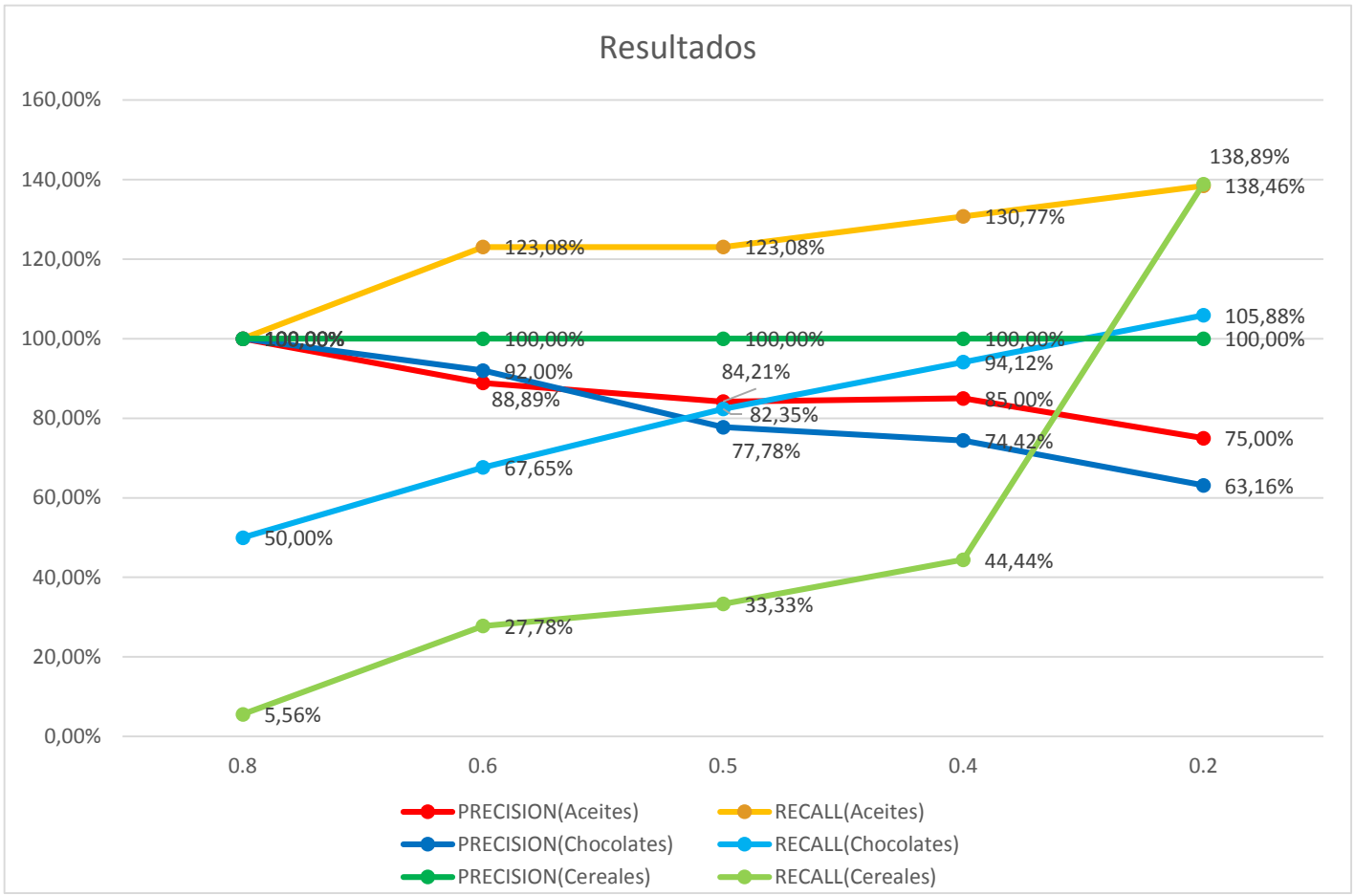


Figura 22. Resultados con ajuste

Como era de esperar la precisión es la misma pero el recall ha aumentado. En ocasiones el recall aumenta por encima del 100%, esto es porque el sistema siempre que exista algún artículo en la tabla de referencia con misma marca y formato que el de la tabla de origen ofrecerá un idéntico (el que más se parezca). Por esto, al ir disminuyendo el umbral el recall termina subiendo por encima del 100%, ya que los duplicados que se han considerado como imposibles de identificar terminan apareciendo porque son lo mejor que puede proporcionar el sistema.

Excepto en aceites esto ocurre al bajar mucho el umbral (con umbral 0.2), el sistema está ofreciendo esos duplicados como mapeos de muy poca similitud, lo cual tiene sentido ya que son duplicados que se han definido como "imposibles de identificar". En la categoría de aceites este fenómeno ocurre antes (ya desde el umbral 0.8 se obtiene un recall del 100%). Esto es porque los duplicados que se han descartado en esta categoría no se han considerado imposibles de identificar debido a su extrema diferencia, sino porque uno de los dos artículos del mapeo contenía alguna palabra clave o adjetivo que lo hacía a priori distinto del otro, pero sin embargo son el mismo. Esto hace que sintácticamente no sean muy distintos (aunque uno tenga virgen extra y el otro no, lo cual parece indicar que tienen que ser distintos) por lo que el sistema les asigna una similitud relativamente alta.

Por último, hay que tener en cuenta que estos últimos datos (*"Figura 22"*) son tan solo una aproximación para comparar el sistema con un humano (teniendo ambos los mismos datos) ya que los duplicados reales que se han usado en este caso no son los auténticos (ha sido un ajuste para realizar dicha aproximación).

4.6. Pruebas con otros algoritmos básicos.

En este apartado se ve como respondería la aproximación más básica y simple a la solución del problema de la detección de duplicados. Es lógico pensar que la primera solución en la que siempre se piensa al encontrarse con dicho problema es: distancia de Levenshtein a cada campo y media de las similitudes resultantes.

Para realizar la prueba de este algoritmo básico y poder llevar a cabo una comparación con la solución que se ha estado probando en este trabajo usaremos un framework existente que implementa la mayoría de los algoritmos comunes para la detección de duplicados y facilita su uso. Se trata de DuDe (**Du**plicate **De**tection).

4.6.1. DuDe

Tal y como se define en la página web [19], esta herramienta de detección de duplicados consiste en un conjunto de componentes con claras interfaces que facilitan la creación de algoritmos y módulos para ampliar el sistema. Además de esto, DuDe provee la mayoría de algoritmos y medidas de similitud básicas así como datos para poder compararlos entre sí. Se trata de un framework fácil de usar, fácil de extender, que soporta una gran variedad de orígenes de datos, proporciona la posibilidad de crear una gran cantidad de algoritmos de detección de duplicados y provee de un buen número de medidas de similitud.

La arquitectura de esta herramienta está ampliamente definida [20] y no se hará hincapié en este proyecto.

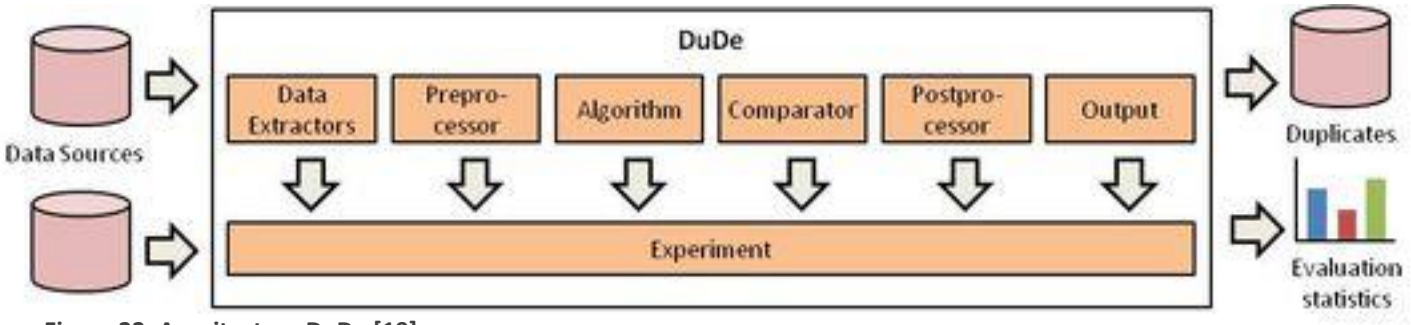


Figura 23. Arquitectura DuDe [19]

Como se puede ver en la imagen (*"Figura 23"*) el framework consta de 6 componentes:

1. Data Extractors: Componente encargado de extraer los datos de la fuente origen (ya tiene implementados conectores con fuentes de datos comunes como ficheros csv o json).
2. Preprocessor: En este componente se realizan pre-procesos a los datos (limpieza de los datos).
3. Algorithm: Este componente define el algoritmo de detección de duplicados que se usará (la forma en la que se van a buscar las parejas de similares)
4. Comparator: Es el encargado de comparar dos registros y determinar la similitud o disimilitud.
5. Postprocessor: Post-procesado de los datos, procesos a posteriori sobre los datos antes de enviarlos al destino.
6. Output: Conector con distintos orígenes de datos (ficheros, bases de datos…).

Todos estos componentes son fácilmente extensible, lo que proporciona al usuario la capacidad de crear sus propios componentes para realizar su experimento. Por ejemplo se puede incluir una conexión con tu base de datos usando la interfaz que proporciona DuDe para los "Data Extractors", o si quieres probar una medida de similitud en concreta tan solo hay que implementarla y usar el resto de componentes del framework.

4.6.2. Las pruebas

Tal y como se ha descrito anteriormente, la intención de esta prueba es poder comparar la solución propuesta en este trabajo con la primera propuesta que se suele emplear al enfrentarse al problema que se ha estado tratando en todo el documento.

Por este motivo, la prueba consiste en un algoritmo muy simple: Primero todos los ítems se incluyen en un mismo fichero (primero los de la tabla de origen y después los de la tabla de referencia). Después de esto se usa el framework DuDe para leer los datos del fichero, por cada registro obtener la media de las similitudes de los campos marca, formato y nombre (usando la distancia de Levenshtein), y escribir el resultado en un fichero cuando la pareja es de distribuidores distintos). Tras esto se inserta el contenido de este último fichero en la base de datos y se obtiene una vista para mostrar por cada ítem de origen el mapeo en el que interviene con máxima similitud.

Por ejemplo, si tenemos:

Item1, dist1
Item2, dist1
Item3, dist2
Item4, dist2
Item5, dist2

Se compara cada uno con los siguientes (NaiveDuplicateDetection del framework):

Item1, dist1 – Item2, dist1 [sim x1] //no se escribe, mismo dist
Item1, dist1 – Item3, dist1 [sim x2] //no se escribe, mismo dist
Item1, dist1 – Item4, dist2 [sim x3] //si se escribe, distinto dist
Item1, dist1 – Item5, dist2 [sim x3] //si se escribe, distinto dist

Y así hasta finalizar.

Cabe destacar que se puede dar el caso de que varios mapeos con mismo ítem origen tengan la misma similitud (ya sea por casualidad o por falta de precisión y redondeo), lo cual hace que dicho ítem origen pueda tener varios ítems de referencia relacionados con máxima similitud. Dada esta observación se pueden ver los resultados obtenidos:

Aceites

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	0	0	20	NA	NA	0%	100%	NA
0.6	4	30	20	13%	87%	20%	80%	0,16
0.5	8	77	20	10%	90%	40%	60%	0,16
0.4	8	88	20	9%	91%	40%	60%	0,15
0.2	8	88	20	9%	91%	40%	60%	0,15

Chocolates

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	0	0	41	NA	NA	0%	100%	NA
0.6	9	14	41	64%	36%	22%	78%	0,33
0.5	23	64	41	36%	64%	56%	44%	0,44
0.4	24	93	41	26%	74%	59%	41%	0,36
0.2	24	95	41	25%	75%	59%	41%	0,35

Cereales

Umbral	Correctos	Identificados	Reales	PRECISION	FP	RECALL	NI	f-measure
0.8	0	0	34	NA	NA	0%	100%	NA
0.6	17	31	34	55%	45%	50%	50%	0,52
0.5	18	43	34	42%	58%	53%	47%	0,47
0.4	20	58	34	34%	66%	59%	41%	0,43
0.2	20	58	34	34%	66%	59%	41%	0,43

Graficas:

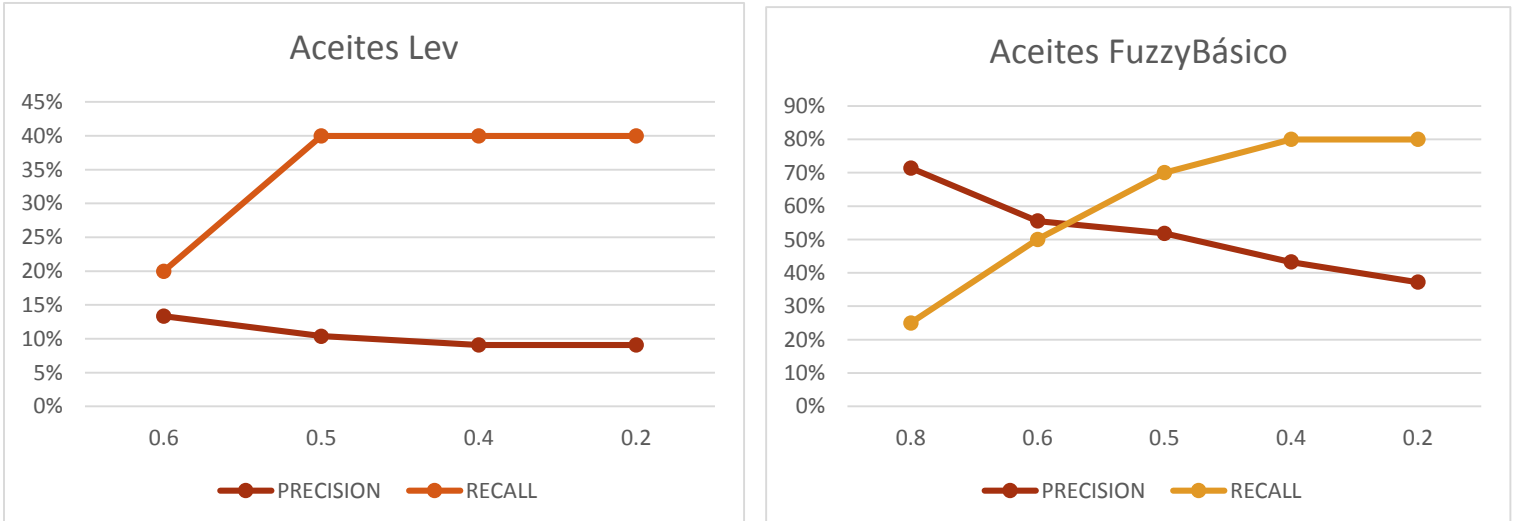
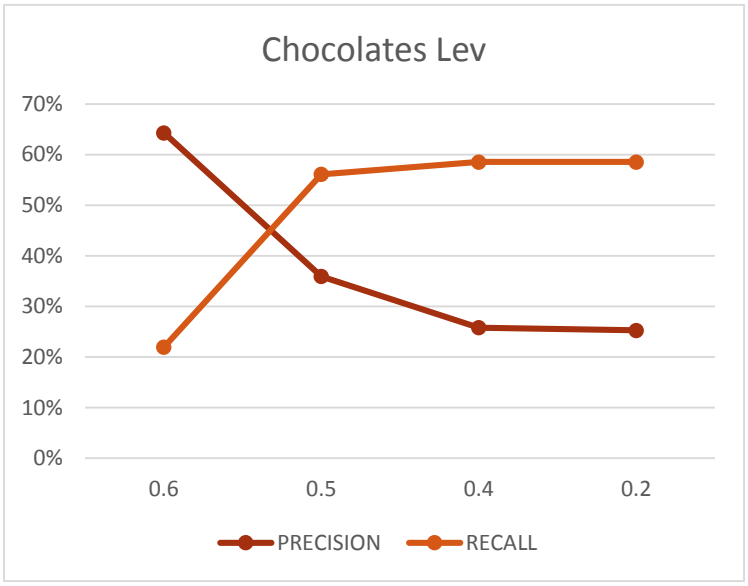


Figura 24. Aceites, comparación Levensthein/Fuzzy Básico



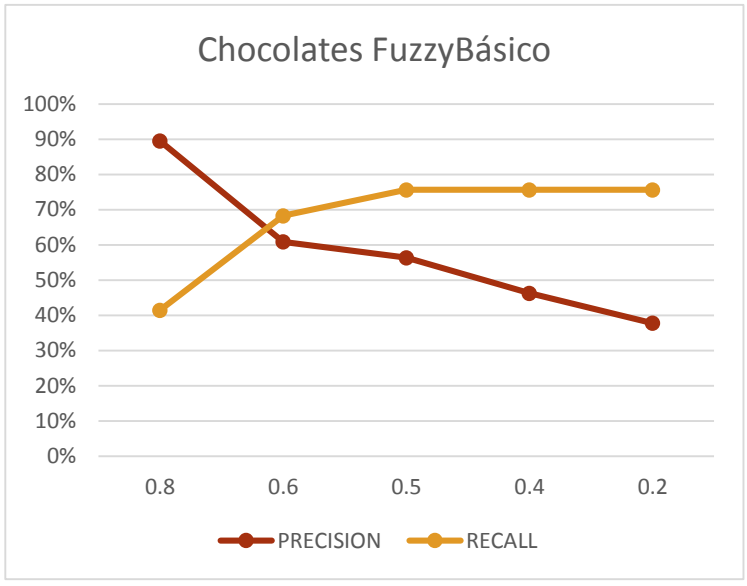
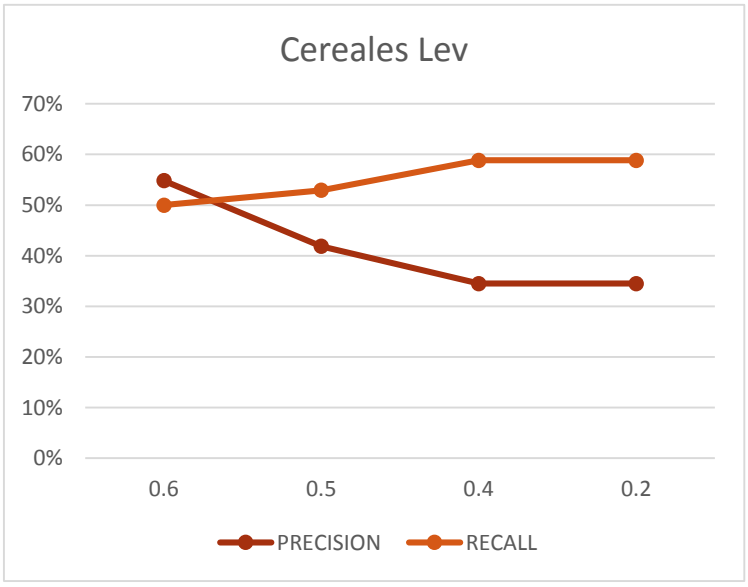


Figura 26. Chocolates, comparación Levensthein/Fuzzy Básico



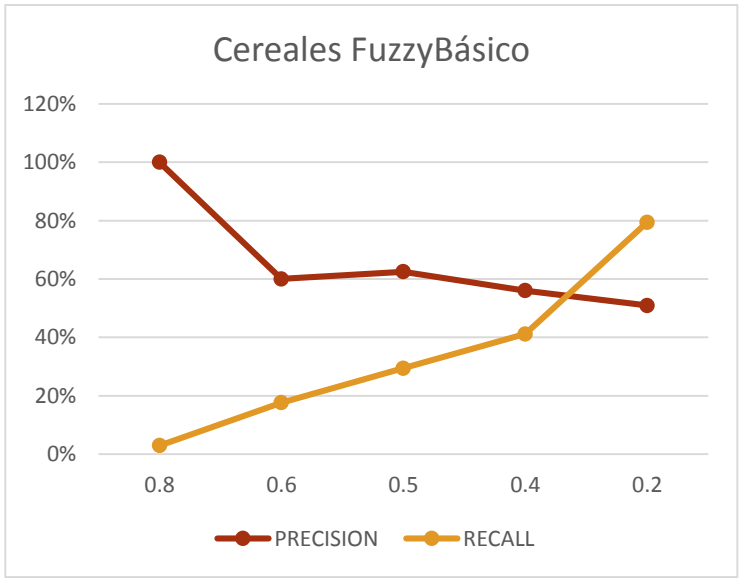


Figura 25. Cereales, comparación Levensthein/Fuzzy Básico

Como era de esperar los resultados en general son bastante peores. Tanto la precisión como el recall son menores en general (tan solo es mayor el recall de los cereales en los primeros umbrales). La distancia de Levensthein mide la distancia entre dos cadenas de texto en cuanto a caracteres, lo cual no funciona muy bien cuando lo importante son las palabras. Esto unido a que se está mediando la similitud del nombre con la del formato y la marca hace que los resultados bajen y se puedan observar problemas como los siguientes:

(Dist 1)(carbonell)(0.75 l) aceite de oliva virgen extra //
(Dist 2)(BORGES)(0.75 l) Aceite de oliva virgen extra (score 0,65476)

(Dist 1)(hojiblanca)(1 l) aceite de oliva virgen extra picual //
(Dist 2)(HOJIBLANCA)(1 l) Aceite de oliva virgen extra (score 0,5904)

(Dist 1)(milka)(125 g) chocolate extrafino con leche //
(Dist 2)(LINDT)(125 g) Chocolate extrafino con leche (score 0,65517)

Capítulo 5. Conclusiones

Como se ha podido observar en los resultados de las pruebas, se trata de un problema muy dependiente del dominio. Los datos de entrada condicionan en gran medida los resultados, provocando que en muchas ocasiones sea imposible proporcionar una solución automática.

La solución óptima no pasa por el uso de una técnica de detección de duplicados básica. Se deben unir y adaptar varias de estas técnicas para adecuarse a los datos concretos de que se disponen. Por eso es muy importante que el sistema pueda configurarse dependiendo de la sección. Como se han visto en resultados incluso los umbrales de aceptación (tanto superior como inferior) pueden variar en función de la categoría. Las características de los datos de la jerarquía de cereales, dan lugar a que el umbral con el que en otras jerarquías se pueden identificar idénticos de manera automática, no sea efectivo en dicho caso concreto.

Además de los distintos umbrales entre categorías se ha observado una cierta mejoría al aplicar ciertas transformaciones a los datos de entrada. Estas transformaciones, llevadas a cabo en un pre-proceso, también pueden depender de la categoría. Por ejemplo, se puede ver que el problema de "1º" o "1 º" tan solo se encontraba en aceites. Del mismo modo, la apreciación de que la palabra "avellana" tiene relevancia, se da en la jerarquía de chocolates.

Aun así, después de analizar los resultados, es fácil apreciar que se puede establecer un umbral general de 0.8 sobre el cual el sistema relativamente simple del "fuzzy lookup acotado" proporciona una precisión del 80% aproximadamente. Sin embargo, establecer un umbral inferior es más difícil, ya que hay idénticos con score muy bajo, 0,0125. Esto hace que la implantación de un sistema con las características del "fuzzy lookup acotado", se pueda implantar con rapidez y facilidad usando las herramientas que proporciona Microsoft. Así pues, el coste de crear un sistema básico que identifique duplicados de forma automática, no es muy elevado, y en pocos meses se puede implantar. Esto proporciona una base sobre la que ir mejorando y aplicando el conocimiento que se valla obteniendo de los datos con los que trabaja el sistema.

Queda claro con los resultados presentados en este proyecto que la intervención de un humano en el proceso de la detección de idénticos es inevitable. El sistema nunca consigue identificar todos los duplicados que existen y genera un gran rango de mapeos con incertidumbre (conjunto de incertidumbre) que necesita de la supervisión humana. Debida a esta necesidad de participación humana en el proceso de catalogación de idénticos, se propone un sistema que fusione la solución del "fuzzy lookup" con un sistema basado en reglas. Así pues el humano encargado de revisar el conjunto de incertidumbre aportara el conocimiento que valla adquiriendo al sistema en forma de reglas. De este modo, las mejoras que se han ido sucediendo en el capítulo 4 serían consecuencia de las reglas creadas por el componente humano del sistema.

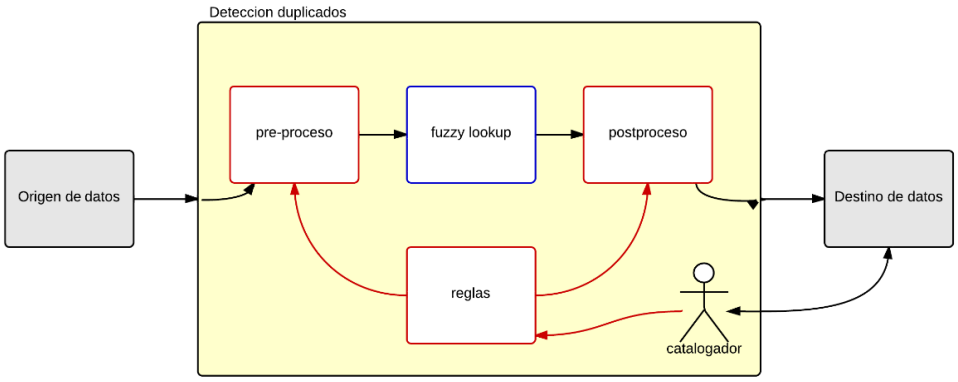


Figura 27. Sistema propuesto

Referencias

1. **Daggupati, Bhanupreeti.** Unsupervised Duplicate Detection (UDD) Of Query Results from Multiple Web Databases. Diciembre de 2011.

2. **Elmagarmid, Ahmed K., Ipeirotis, Panagiotis G. y Verykios, Vassilios S.** Duplicate Record Detection: A Survey. *IEEE Trans. Knowledge and Data Eng.* Enero de 2007. Vol. 19, 1.

3. **Levenshtein, V.I.** Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR.* 1965. Vol. 163, 4.

4. **Wagner, Robert A. y J.Fischer, Michael.** The String-to-String Correction Problem. *Journal of the ACM.* Enero de 1974. Vol. 29, 1.

5. **T.F., Smith y M.S., Waterman.** Identification of Common Molecular Subsequences. *J. Molecular Biology.* 1981. Vol. 147.

6. **Waterman, Michael S., Smith, Temple F. y Beyer, illiam A.** Some biological sequence metrics. *Advances in Mathematics .* 1976. Vol. 20, 4.

7. **Jaro, Matthew A.** UNIMATCH: A Record Linkage System. *User's Manual, U.S. Bureau of the Census, Washington, D.C.* 1976.

8. **Yates, R. B. y Neto, B. R.** Modern Information Retrieval. New York : ADDISON-WESLEY, 1999.

9. **E., Ukkonen.** Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science.* 1992. Vol. 92, 1, págs. 191-211.

10. **Fellegi, Ivan Peter y Sunter, Alan B.** A theory for record linkage. *Journal of the American Statiscal Association.* 1969. Vol. 64, 328.

11. **Moore, Andrew W.** Bayes Net Structure Learning. *CMU-CS.* 29 de Octubre de 2001.

12. **[En línea] http://www.autonlab.org/tutorials/svm.html.**

13**. Sarawagi, S. y Bhamidipaty, A.** Interactive deduplication using active learning. *KDD.* 2002.

14. **Bilenko, M. y Mooney, R.J.** Adaptive Duplicate Detection Using Learnable String Similarity Measures. *Proc. ACM SIGKDD.* 2003.

15. **Wang, Y. R. y Madnick, S. E.** The inter-database instance identification problem in integrating autonomous systems. *In Proc. fifth IEEE Int'l Conf. data Eng.* 1989.

16. **Becker, S y Plumbley, M.** Unsupervised neural network learning procedures for feature extraction and classification. *International Journal of Applied Intelligence.* 1996. Vol. 6.

17. **Chuan, Xiao, y otros.** Efficient Similarity Joins for Near Duplicate Detection. 28 de Abril de 2008.

18. **Surjait, Chaudhuri, y otros.** Robust and Efficient Fuzzy Match for Online Data Cleaning. *ACM SIGMOD.* Junio de 2003.

19. **DuDe. [En línea] http://www.hpi.uni-potsdam.de/naumann/projekte/dude_duplicate_detection.html.**

20. **Draisbach, Uwe y Naumann, Felix.** DuDe: The Duplicate Detection Toolkit. *VLDB.* Septiembre de 2010. Vol. 10.