EMD: UNA METODOLOGÍA DE RECUPERACIÓN PARA SISTEMAS DE LENGUAJE NATURAL BASADOS EN CASOS

A.Moreo [1] **J.L Castro**[1] **V.López**[1] **J.M Zurita**[1]

[1] Departamento de Ciencias de la Computación y la Inteligencia Artificial, Universidad de Granada, {moreo, castro, victor, zurita}@correo.ugr.es

Resumen

Muchas de las aplicaciones que trabajan con el Lenguaje Natural, como los Sistemas de FAQ retrieval, los Sistemas de Diálogo, los Asistentes Virtuales, etc. pueden ser modelados mediante un enfoque de Razonamiento Basado en Casos (CBR), transformando el problema del Question Answering al de buscar el caso de la base de casos más parecido a la pregunta introducida por el usuario. Gran parte de los modelos propuestos se valen de Palabras Clave para tal propósito. Sin embargo, la elección de este conjunto de palabras no es una tarea trivial, puesto que depende del conocimiento experto y posiblemente, las palabras que lo compongan no sean suficientes de cara a la diferenciación de casos.

En este trabajo proponemos un algoritmo que generaliza este enfoque aprendiendo automáticamente un conjunto de expresiones (expresiones diferenciadoras) que definen un criterio de comparación entre preguntas. De esta manera, un algoritmo de búsqueda puede utilizarlas de forma eficiente para recuperar el caso más parecido de la base de casos.

Keywords: Question Answering, FAQ retrieval, Case Retrieval

1 INTRODUCCIÓN

El problema del Question Answering cobra especial interés en la Soft Computing por su capacidad para dotar a los computadores de la habilidad de comprensión de preguntas en Lenguaje Natural (LN) y sus posibilidades en el marco de la recuperación automática de información. Sin embargo, el LN presenta numerosos problemas, discutidos en [1, 2], que es necesario abordar para alcanzar esta meta, entre los que destacan el de la ambigüedad y la desestructuración.

Generalmente se clasifican estos sistemas, atendiendo al enfoque que aplican sobre el conocimiento, en sistemas de Dominio Abierto (buscan la información en la Web) o de Dominio Cerrado (parten de un modelado delimitado del conocimiento). Nuestro algoritmo se enmarca dentro de los sistemas de Dominio Cerrado.

Una de las estrategias propuestas en este tipo de sistemas consiste en aplicar un modelo Basado en Casos, donde cada Caso se constituye por una expresión lingüística y cierta información adicional. En FALLQ [3] encontramos un ejemplo de este enfoque. El problema queda reducido entonces a distinguir cual de las sentencias de la base de casos es la sentencia más parecida a la introducida por el usuario y la respuesta devuelta es la asociada a dicho caso. El problema se conoce como *Case retrieval*.

Este enfoque es especialmente interesante en el problema del *FAQ retrieval* (recuperación de información automática en conjuntos de "Preguntas Frecuentes"), cuyas principales dificultades son comentadas en el trabajo de Sneiders [4]. Se han propuesto muchos enfoques en este área, algunos de ellos bajo la filosofía de clustering, como [5] y FAQ Finder [6] en el que se utiliza un modelo de espacio vectorial (VSM) para calcular el grado de similitud combinado con WordNet [12] para gestionar el criterio de matching. Otros trabajos se centran en aplicar técnicas sobre Plantillas, como en los sistemas de Sneiders. En la mayoría de estos sistemas, la dependencia con el conjunto de Palabras Clave (términos relevantes en un dominio particular de conocimiento) es muy fuerte.

El problema de esto es que el conjunto de palabras clave, que depende de conocimiento experto y que a menudo se selecciona a mano, no siempre es buen candidato frente a la distinción entre casos. Se han planteado diferentes metodologías para hacer frente a estos problemas. Podemos encontrar un ejemplo de ello en Auto-FAQ [8], en el que se aplican técnicas de procesamiento del lenguaje para mejorar la comparación entre términos clave. En el sistema de Sneiders, las palabras clave se clasifican en *obligatorias*, *opcionales* e *irrelevantes* de forma automática. Ask Jeeves [14] clasifica el conjunto de FAQs en 7 clases distintas para mejorar la búsqueda mediante palabras clave. Otros enfoques pretenden capturar la semántica de los casos elevando el nivel de representación del conocimiento por medio de ontologías, ayudando de esta forma al proceso de gestión del conocimiento en el problema del FAQ retrieval, como se discute en [9, 10]. Esta propuesta tiene como dificultad

añadida el hecho de que la construcción de ontologías sobre dominios grandes puede llegar a ser una tarea compleja.

El interés del problema de la comparación de preguntas como mecanismo previo para responderlas automáticamente es indiscutible, puesto que la forma más natural de obtención de información en los humanos se basa en el esquema de pregunta/respuesta. Es por esto que supone una fuerte motivación en el marco del E-Learning [7, 13].

En este artículo, proponemos un algoritmo para hallar automáticamente conjuntos de *Expresiones* que operan a modo de clasificadores sobre casos en torno a los cuales se construyen, permitiendo distinguirlos de forma eficiente. Dado que este conjunto de expresiones es muy grande, se pretenden seleccionar de forma automática aquellos más representativos.

El artículo está estructurado de la siguiente manera. En la sección 2 se plantea nuestro enfoque del problema. En la sección 3 definimos los conceptos relacionados con las expresiones y se explica el algoritmo de adquisición de los clasificadores. En el apartado 4 presentamos el algoritmo de búsqueda. Los apartados 5 y 6 están dedicados a los experimentos y conclusiones. Finalmente, la sección 7 recoge las propuestas de Trabajos Futuros.

2 ENFOQUE DEL PROBLEMA

2.1. SISTEMAS BASADOS EN LENGUAJE NATURAL

Este tipo de sistemas son entidades virtuales cuya finalidad es la de proporcionar información a consultas en las que tanto la pregunta como la respuesta se formula en lenguaje natural. Ejemplos de este tipo de sistemas son los Asistentes Virtuales, los FAQs inteligentes, los Puntos de Información turística inteligentes o los Sistemas de Diálogo.

Combinan diferentes técnicas de procesamiento de preguntas, búsqueda o inferencia de información y elaboración de la respuesta. La principal dificultad de estos sistemas estriba en la comprensión del lenguaje natural y en la representación del conocimiento.

2.2. ENFOQUE BASADO EN CASOS

En nuestra propuesta, se aborda el problema de la representación del conocimiento mediante un enfoque de Dominio Cerrado Basado en Casos. El razonamiento basado en casos es un modelo en el que se utiliza la experiencia precedente (*casos*) para resolver nuevos problemas por *similitud* y *adaptación*.

El proceso de comprensión consiste en *recuperar* el caso de la base de casos que mayor similitud guarde con la pregunta introducida por el usuario. El problema de la elaboración de la respuesta se reduce entonces a devolver la respuesta asociada a la pregunta seleccionada. Para ello propondremos una métrica de similitud entre preguntas, en un enfoque similar a [11], con la ventaja de que puede servir como mecanismo para simular la captación de la semántica de la pregunta.

Cada uno de los casos de nuestra base de casos se compone de un conjunto de expresiones lingüísticas (reformulaciones distintas de una misma pregunta) y un conjunto de índices que determinan dónde se ubica dentro de la estructura de la base de casos. Esta estructura es dependiente del dominio y la define el Ingeniero de Conocimiento. Cuanto más rico sea el conjunto de reformulaciones asociadas al caso, mejor quedará capturada su variedad sintáctica.

Las expresiones lingüísticas de los casos pueden estar descritas de forma *abstracta* utilizando variables que puedan referir a más de un objeto del dominio y que se instanciarán en la etapa de *adaptación*. Por motivos de espacio, obviaremos los *casos abstractos* y abordaremos el problema como si de un *Case retrieval* se tratase.

La etapa de *evaluación* de resultados y aprendizaje es realizada manualmente por un experto, que añade nuevos casos a la base de casos especificando el conjunto correcto de índices asociado.

En un dominio sobre información de Sistemas Inteligentes basados en LN, podemos encontrar el siguiente ejemplo, indexado a dos niveles, de la Tabla 1:

Tabla 1: Ejemplo de Base de Casos sobre SI de LN

Propiedad	Objeto	Reformulaciones
Ventajas	**AsistenteVirtual**	¿Cuáles son las ventajas de un Asistente Virtual?
	FAQ Inteligente	¿Cuáles son las ventajas de un FAQ inteligente?
Precio	**AsistenteVirtual**	¿Cuánto cuesta un Asistente Virtual?
		¿Cuánto vale un Asistente virtual?
	FAQ Inteligente	¿Cuánto cuesta un FAQ inteligente?
		¿Cuánto vale un FAQ inteligente?
Mantenimiento	**AsistenteVirtual**	¿Qué tipo de mantenimiento tienen los Asistentes Virtuales?
		¿Cómo se incrementa el conocimiento de los Asistentes Virtuales?
	FAQ Inteligente	¿Qué tipo de mantenimiento tienen los FAQs inteligentes?
		¿Cómo se incrementa el conocimiento de los FAQs inteligentes?

Esta estructura de casos se muestra en la Figura 1:

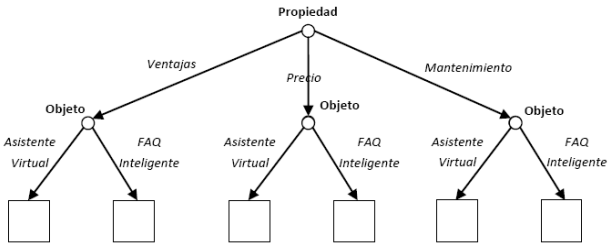

Figura 1: Estructura de la Base de Casos.

Estos índices podrían ser utilizados por un mecanismo de representación del conocimiento más potente como, por ejemplo, las Ontologías.

3 EXPRESIONES MINIMALES DIFERENCIADORAS

En muchos sistemas de Dominio Cerrado, se definen una serie de palabras clave, con la intención de captar la mayor relevancia de éstas frente al resto de términos del dominio. Nuestra intención es deducir conjuntos de palabras extraídos de los casos de forma que permitan distinguirlos al tiempo que representen una reducción semántica y generalicen a otras reformulaciones no contempladas. Existen algunas propuestas en la literatura para conseguir la misma finalidad, como la basada en técnicas estadísticas [15].

El algoritmo busca recubrimientos, que llamaremos expresiones, sobre las sentencias lingüísticas que componen los casos. Se seleccionarán aquellas expresiones que cumplan con las propiedades de diferenciación (subconjunto que únicamente está incluido en el caso del que se extrae) y de minimalidad (es el de menor tamaño).

Supongamos la base de casos a un nivel del ejemplo, con únicamente dos casos, que se muestra en la Tabla 2:

Tabla 2: Base de Casos sencilla.

Caso	Preguntas
Caso1	¿Cuál es el nombre del profesor?
Caso2	¿Cuál es el departamento del profesor?

En este ejemplo, podríamos tomar las expresiones e_1*={*nombre*}* y e_2*={*departamento*}* para diferenciar los casos. El símbolo '*' representa "*cualquier cadena de caracteres*". Sin embargo, si existiera un tercer caso "*¿Cuál es el nombre de la asignatura que imparte el profesor*?", entonces, e_1 dejaría de desambiguar al Caso1. Si redefiniéramos e_1*={*nombre*profesor*}* y añadiéramos e_3*={*nombre*asignatura*}* entonces las expresiones volverían a ser diferenciadoras.

No obstante, existen más expresiones para el Caso 3, como por ejemplo *{*asignatura*}* o *{*imparte*}* que podrían ser igualmente válidas, por lo que los conjuntos de expresiones asociadas a los casos contendrán varias expresiones. También podríamos tomar la expresión *{*nombre*asignatura*imparte*}*, pero esta expresión no es minimal y en la práctica considerar este tipo de expresiones puede conllevar un gasto computacional inaceptable. Existen otras posibles expresiones diferenciadoras como *{*la*}*, pero ésta no guarda información relevante y debería ser descartada.

Vamos a definir una serie de conceptos que permitirán obtener un conjunto de expresiones útiles.

3.1. DEFINICIONES

3.1.1. Sentencia

Una sentencia S es una lista de palabras no necesariamente únicas. Las palabras pertenecen a una y sólo una de las clases de palabras: *Kw* (Palabras Clave), *Sw* (StopWords, conjunto no estándar de palabras que carecen de significado semántico) o *Cw* (Comunes o resto de palabras). W = { Kw ∪ Sw ∪ Cw } es una partición.

$$S=[w_1, w_2, w_3, \ldots, w_n],\ w_i \in W \quad (1)$$

Ejemplo: dime_{Cw} las_{Sw} ventajas_{Kw} de_{Sw} un_{Sw} Asistente_{Sw}

3.1.2. Expresión

Una *Expresión E sobre una Sentencia S* es una reducción de S compuesta por un subconjunto de sus palabras dispuestas en el mismo orden y que contiene, al menos, una palabra clave si en la sentencia S había al menos una. En el caso de que no hubiera ninguna palabra clave en la sentencia, entonces la única restricción impuesta sobre las palabras que la forman, es que no todas sean StopWords.

La notación "*E exp S*" representa que E es una expresión de S. El conjunto de todas las expresiones sobre una sentencia es E(S)={e / e Exp S}

3.1.3. Expresión Diferenciadora

Una *expresión diferenciadora e* es una expresión que distingue unívocamente la sentencia S de la que ha sido extraída del conjunto de sentencias B del resto de casos de la base de casos (pero no necesariamente de las reformulaciones del caso del que se extrae).

$$e\ Exp_{dif}\ \text{S wrt B} \leftrightarrow \begin{cases} e \in E(S) \\ \forall S' \in B, e \notin E(S') \end{cases} \quad (2)$$

El conjunto de todas las expresiones diferenciadoras de S con respecto a B es: ED(S,B) = {e / e Exp_{dif} S wrt B}. Las siglas *wrt* se corresponden con la expresión del inglés "*with respect to*".

3.1.4. Expresión Minimal Diferenciadora

Una *Expresión-Minimal-Diferenciadora* (MDE del inglés *Minimal-Differetiator-Expression*) es una Expresión-Diferenciadora no contenida en ninguna otra Expresión-Diferenciadora de la sentencia.

$$e\ MDE\ \text{S wrt B} \leftrightarrow \begin{cases} e \in \text{ED}(S,B) \\ \forall e' \in (\text{ED}(S,B) - \{e\}), e \not\subset e' \end{cases} \quad (3)$$

Nuestro objetivo, es calcular MDE(S,B)={e / e MDE S wrt B}

3.2. OBTENIENDO LAS EMDS

En esta sección, explicamos el algoritmo para obtener las EMDs. Previamente, definimos las etapas de preprocesamiento que se realizan sobre los casos. Por último, mostramos un ejemplo real de ejecución.

3.2.1. Preprocesamiento de la Base de Casos

El preprocesamiento de las sentencias de la base de casos se compone de tres fases. En una primera se eliminan tildes y marcas como los signos de puntuación y se normaliza el caso (todo en minúscula). En la segunda, se reducen *Ngramas* a *tokens*, con el objeto de evitar

interferencias semánticas entre las palabras individuales de conceptos nombrados por más de una palabra. Por ejemplo, "*inteligencia artificial*" podría reducirse a *inteligencia_artificial*. Con esto se consigue, además de reducir el número de palabras, evitar que aparezcan expresiones que contengan únicamente algunas de las palabras de un concepto. Este preprocesamiento se implementa mediante reemplazo por expresiones regulares. En la tercera y última fase, pretendemos capturar el tipo de la pregunta. Suponiendo que éste se deduce de la partícula interrogativa y el verbo más próximo, los unificamos en un token.
Por ejemplo, *"¿Qué es la Inteligencia Artificial?"* quedaría, despues de las tres fases anteriores, como *"que_es la inteligencia_artificial"*

3.2.2. Algoritmo

El algoritmo calcula el conjunto MDE(S,B) para todas las sentencias S con respecto a la base B de N casos. Cuenta con un Registro de MDEs calculadas hasta el momento. Este registro permite identificar si una Expresión Diferenciadora engloba a alguna otra ya calculada. Su implementación por medio de Conjuntos Hash hace esta tarea muy eficiente. Las MDEs calculadas para cada Sentencia se van almacenando en el Registro.
Si el conjunto de palabras clave es vacío, o la sentencia no contiene ninguna, entonces el algoritmo trata a todas sus palabras como clave. Por tanto, no es obligatorio definir un conjunto de palabras clave. Este es uno de los puntos fuertes del algoritmo.
El algoritmo utiliza una cola de expresiones candidatas. Para cumplir el requisito de que cada expresión contenga al menos una palabra clave, reordenamos el vector de palabras de la sentencia Q_i^j en un nuevo vector que llamamos KCQ_i^j y que sirve para inicializar la cola únicamente con palabras clave. El número de palabras clave de Q_i^j es z. Cada elemento de KCQ_i^j contiene un índice, consultado por la función $index_{KC}$ que se utiliza para reordenar la expresión al final del algoritmo y antes de incluirla en el Registro:

```
1  Proc MDE(Qi^j, B, R):
2   MDE:=Ø
3   OPENED:=Ø
4   For k=1 to z do:
5    Push(OPENED,{KC_Qij[k]})
6   endFor
7   While(OPENED≠Ø) do:
8    expr:=pop(OPENED)
9    If(CheckMDE(expr,R,B-{Qi}))
10    MDE:=MDE ∪ expr
11   Else
12    If(size(expr)<alpha_bound)
13     For t=index_KC(tail(expr))+1 to k
14      push(OPENED,expr ∪ {KC_Qij[t]})
15     endFor
16   enddo
17  return EMD
```

El factor de ramificación del árbol puede llegar a ser muy elevado, por ello, en la línea 12, podamos el árbol mediante un umbral *alpha* (que establecemos en 5), por lo que no se crearán expresiones con más de 5 palabras.
El procedimiento que calcula el conjunto MDE debe verificar si la expresión candidata está ya registrada en R o está incluida en alguna otra sentencia de otro caso. Si esto ocurre, entonces la expresión no es diferenciadora. *CheckMDE* implementa esta comprobación:

```
1  Proc CheckMDE(expr,R,[Q1,Q2,...,Qy])
2   expro=QuickShortindex(expr)
3   isMDE:=true
4   If(Registred(R,expro))
5    isMDE:=false
6   Else
7    For(i=1 to y)AND(isMDE) do
8     For(j=1 to size(Qi))AND(isMDE)do
9      If(IsSatisfied(expro,Qi^j))
10      isMDE:=false
11    endFor
12   endFor
13  return isMDE
```

3.2.2. Ejemplo

En la tabla 3 se muestra un ejemplo de las EMDs que se obtuvieron de un conjunto de reformulaciones asociadas al punto de información "*Ventajas de un Asistente Virtual*" en nuestros experimentos (sección 5).

Tabla 3: Ejemplo de EMDs obtenidas.

Reformulación	EMDs
¿Qué ventajas tiene un Asistente Virtual?	{*ventajas*}
¿Cuáles son las ventajas de usar un asistente?	{*cuales_son*usar*}, {*las*usar*}
¿Qué ventajas me aportaría comprar un AV?	{*que_aportaria*}, {*me*comprar*}
¿En qué nos puede ayudar un Asistente Virtual?	{*en_que_puede*}, {*ayudar*}
¿Qué aportan vuestros asistentes virtuales?	{*que_aportan*}
¿Qué es lo mejor de los AV?	{*que_es*mejor*}, {*lo*mejor*}
¿Qué supone para mi empresa un Asistente?	{*que_supone*}
¿Qué ventajas me aporta un Asistente Virtual?	{*que_aporta*}
¿Qué ventajas se pueden destacar en los Asistentes Virtuales?	{*destacar*}

4 MATCHING CON EMDS

Nuestro algoritmo de búsqueda opera sobre las EMDs, especialmente diseñadas para mejorar la búsqueda, y no directamente sobre los casos o sobre las palabras clave. El caso ganador, es aquel que maximiza una medida de matching o cercanía entre una sentencia introducida por el usuario y el conjunto de EMDs asociadas al caso. Definiremos primero formalmente las relaciones de inclusión entre sentencias y expresiones para luego describir la búsqueda.

4.1. NIVELES DE COMPARACIÓN Y GRADOS DE INCLUSIÓN

Una sentencia satisface a una expresión si la expresión cubre a la sentencia, es decir, si todos los términos que la forman, están en la sentencia junto con, posiblemente, algunos más. Por ejemplo, la sentencia *"¿Qué ventajas tiene un Asistente_Virtual?"*, satisface a la expresión *{*ventajas*Asistente_Virtual*}*. La medida de matching se basa en el grado de inclusión en que esto ocurre, y éste depende de la cercanía entre parejas de palabras y a su relación de orden en ambas listas de palabras.

Para comparar dos palabras w_1 y w_2 usamos un Comparador que devuelve $C(w_1,w_2)=Exacto$ si son la misma palabra (aceptando diferencias de género, número y caso y distintas conjugaciones para un mismo verbo), $C(w_1,w_2)=Derivado$ si son sinónimas o están relacionadas jerárquicamente (dos niveles de profundidad), o $C(w_1, w_2)=Diferentes$, en otro caso. Para implementarlo, hemos utilizado *WordNet* [13].

Contaremos con un predicado *orden(S,E)* que comprueba si las palabras *no Diferentes* contenidas en la sentencia S y la expresión E están dispuestas en el mismo orden.

De acuerdo a la proximidad de las palabras y al orden, la inclusión entre S y E puede ser *Exacta* ($S\ s_e E$), *Laxa* ($S\ s_l E$), *Desordenada* ($S\ s_d E$) o *Desordenada-Laxa* ($S\ s_{dl} E$), según:

$$S\ s_e E \leftrightarrow (\forall e \in E, \exists s \in S\ /\ C(s,e) = \{Exacto\}) \wedge orden(S,E) \quad (4)$$

$$S\ s_l E \leftrightarrow (\forall e \in E, \exists s \in S\ /\ C(s,e) = \{Exacto \vee Derivado\}) \wedge (\exists e \in E, \nexists s \in S\ /\ C(s,e) = \{Exacto\}) \wedge orden(S,E) \quad (5)$$

$$S\ s_d E \leftrightarrow (\forall e \in E, \exists s \in S\ /\ C(s,e) = \{Exacto\}) \wedge \neg orden(S,E) \quad (6)$$

$$S\ s_{dl} E \leftrightarrow (\forall e \in E, \exists s \in S\ /\ C(s,e) = \{Exacto \vee Derivado\}) \wedge (\exists e \in E, \nexists s \in S\ /\ C(s,e) = \{Exacto\}) \wedge \neg orden(S,E) \quad (7)$$

Cuyo valor de matching queda ponderado como sigue:

$$m(S,E) = \begin{cases} 1 & \text{if } S\ s_e\ E \\ 0.5 & \text{if } (S\ s_l\ E) \vee (S\ s_d\ E) \\ 0.25 & \text{if } S\ s_{dl}\ E \\ 0 & \text{in other case} \end{cases} \quad (8)$$

4.2. ALGORITMO DE BÚSQUEDA

El matching entre una sentencia S_u introducida por el usuario y un caso *C*, se calcula de forma aditiva sobre el valor de matching *m* de S_u con todas las expresiones del conjunto extraídas para cada reformulación incluida en *C* y que por simplificar llamaremos DME_C.

El valor semántico recogido en una Expresión, es tanto más preciso cuanto mayor sea el número de palabras que la componen. Por esta razón, ponderamos cada valor de matching por la longitud de la expresión.

$$M(S_u, C) = \sum_{e_i \in DME_C} (m(S_u, e_i) \cdot size(e_i)) \quad (9)$$

Por último, el algoritmo devuelve el caso que maximice la medida *M* en la base *B* de casos siempre y cuando éste supere un umbral *h* de fiabilidad. Es decir:

$$Search_{MDE}(S_u, B) = \begin{cases} C_i \in B\ /\ \left(M(S_u, C_i) \geq \max_{C_j \in B}\{M(S_u, C_j)\} \wedge M(S_u, C_i) \geq h\right) \\ \emptyset \text{ en otro caso} \end{cases} \quad (10)$$

La respuesta obtenida ante una pregunta, es la respuesta asociada al caso ganador, o la respuesta vacía si no se infirió ningún caso.

5 EXPERIMENTOS

En esta sección ponemos a prueba nuestro algoritmo enfrentándolo a una implementación del algoritmo de recuperación por conteo de palabras comunes. Si definiéramos un conjunto de palabras clave, entonces el algoritmo de conteo sería un algoritmo de búsqueda básico por palabras clave. Sin embargo, para evitar que los resultados dependan de la calidad de las palabras clave, este conjunto será vacío en ambos algoritmos. Utilizaremos la técnica *10-fold cross-validation*.

5.1. DATOS DE EXPERIMENTACIÓN Y MÉTRICA

Para probar el algoritmo, hemos tomado la base de casos de un Asistente Virtual de una empresa real y hemos establecido un conjunto de pruebas basándonos en los puntos de información del mismo. Partimos el conjunto de casos en dos: por un lado, el *conjunto estático* que utilizamos para aprender el lenguaje y que no intervino en la partición de lotes; por otro lado, tomamos un conjunto de 1500 preguntas realizadas de forma independiente que es el que se usó para crear los lotes.

Se han realizado 10 validaciones completas para tener resultados suficientes para contrastarlos con el *t*-test. En cada *10-fold cross-validation* la base de casos es divida en 10 subconjuntos independientes y excluyentes entre sí, y todos con idéntica distribución. Cada subconjunto se utiliza una única vez como conjunto de prueba quedando la unión del resto 9 como base. Los resultados se han evaluado mediante la métrica de $precisión = \frac{Nc}{Nc+Nf}$ (proporción de los casos clasificados correctamente, *Nc*, frente a todos los casos clasificados, *Nc+Nf*, donde *Nf* denota a los erróneamente clasificados).

5.2. RESULTADOS OBTENIDOS

Los resultados que hemos obtenido en media se muestran en la tabla 4, donde *MDE* representa la media de porcentaje de acierto de nuestro algoritmo, y la fila *M*, la media de porcentaje de acierto del algoritmo de conteo. Como puede verse la media de nuestro algoritmo es superior a la del algoritmo de conteo.

Tabla 4: Medias de resultados.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
MDE	90,30	90,57	91,00	91,18	90,36	90,84	90,54	91,49	90,32	91,10
M	70,70	70,89	69,26	68,99	69,75	69,48	70,27	69,71	69,49	68,97

	Media	Desv. Típ.	Error Típ. En Media
EMD	**90,774**	0,413	0,131
M	**69,786**	0,669	0,212

Con la intención de verificar que estos resultados son estadísticamente significativos calculamos el *t*-test al nivel de confianza del 95%, y obtuvimos un *p-value* inferior a 0,001. Esto demuestra que existen diferencias significativas en la precisión a favor de nuestro modelo respecto al algoritmo de conteo.

6 CONCLUSIONES

El conjunto de MDEs generadas automáticamente, prescindiendo de palabras clave, supera sustancialmente el porcentaje de acierto de los algoritmos de conteo de palabras clásicos. Al permitir que no se tenga que crear un conjunto de palabras clave, la etapa de modelado del conocimiento, en aquellos problemas en los que esto sea necesario, se agiliza notablemente.

La mejora de porcentaje de acierto que se consigue con el preprocesamiento de *Ngramas* y *Tipo de pregunta* no es estadísticamente mejor. Sin embargo, el tiempo de obtención de MDEs se reduce al bajar el índice de ramificación del algoritmo. Por otro lado, si bien es cierto que el algoritmo depende de las reglas de reducción de *Ngramas* que deben ser definidas a mano, también es cierto que esta dependencia es menor que la que tienen la mayoría de algoritmos con respecto a las palabras clave.

La escalabilidad del algoritmo, en términos de porcentaje de acierto es muy buena. Cuanto mayor sea el número de reformulaciones y de casos, mayor lo será el porcentaje de acierto, ya que las MDEs abstraerán con más precisión. En el límite, considerando una base de casos teórica donde el despliegue de reformulaciones fuese exhaustivo, los clasificadores que se obtendrían serían óptimos, llegando a contener únicamente aquellas palabras que realmente sirvieran para diferenciar unas preguntas de otras.

7 TRABAJOS FUTUROS

La búsqueda automatizada de clasificadores de preguntas que proponemos requiere de más estudio y pruebas. Como tarea futura se considerará un estudio frente a problemas con conjuntos de palabras clave definidas, para evitar que los resultados se vean sesgados por la degeneración de las propiedades del concepto de *Palabra Clave* al asumir que todas lo son.

Nuestros trabajos futuros se centrarán en buscar no solo aquellos elementos que nos informen de qué diferencia unas preguntas de otras sino buscar también aquellos por los que las reformulaciones de un mismo caso se parecen. Podremos utilizar esta información adicional para ampliar el conjunto de reformulaciones en una etapa previa al entrenamiento y capturar algunas relaciones sintácticas que inicialmente no eran accesibles. Para representar esta información utilizaremos *Gramáticas* que generalicen las reformulaciones de un caso y que nos permitan producir casos nuevos por combinación de estructuras sintácticas.

Por otro lado, pretendemos automatizar la creación de las expresiones regulares que dan soporte a la etapa de preprocesamiento de *Ngramas* mediante el análisis de frecuencias de términos consecutivos.

Referencias

[1] N. Nicolas; B. Kalina; A. Galia; M. Ruslan, IBM T.J. Watson Research Center, Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003

[2] Advances in Natural Language Processing 5th International Conference on NLP, FinTAL 2006, Proceedings

[3] M. Lenz, A. Hbner, and M. Kunze, “Question Answering with Textual CBR”, Proceedings of the International Conference on Flexible Query Answering Systems (Denmark), pages 236–247, 1998.

[4] E. Sneiders, “Automated FAQ answering: continued experience with shallow language understanding”, AAAI Fall Symposium, 1999, pp.97–107.

[5] Harksoo Kim, Jungyun Seo, “Cluster-Based FAQ Retrieval Using Latent Term Weights”, IEEE INTELLIGENT SYSTEMS,

[6] K. Hammond, R. Burke, C. Martin and S. Lytinen, “FAQ Finder: A case-based approach to knowledge navigation”, Proceedings of the 11th Conference on Artificial Intelligence for Applications,1995,pp.80–86.

[7] Deng-Yiv Chiu, Ya-Chen Pan and Wen-Chih Chang, “Using Rough Set Theory to Construct E-Learning FAQ Retrieval Infrastructure”

[8] S.D. Whitehead, “Auto-FAQ: An Experiment in Cyberspace Leveraging,” Computer Networks and ISDN Systems, vol. 28, nos. 1–2, 1995, pp. 137–146.

[9] S.Y. Yang, “How Does Ontology Help Information Management Processing,” WSEAS Transactions on Computers, 5(9), pp. 1843-1850, 2006.

[10] S.Y. Yang, “How Does Ontology Help User Query Processing for FAQ Services,” WSEAS Transactions on Information Science and Applications, 4(5), pp. 1121-1128, 2007.

[11] Wanpeng Song, Min Feng, Naijie Gu and Liu Wenyin, “Question Similarity Calculation for FAQ Answering”, Third International Conference on Semantics, Knowledge and Grid, pp. 298-301

[12] C. Fellbaum, WordNet: An Electronic Lexical Database, the MIT Press, 1998.

[13] C.S. Lin “The desing and implementation of network learning environment”, Information and Education Volume 67, pp.24-50.

[14] Ask Jeeves, http://www.ask.com

[15] Frans Coenen, Paul Leng, Robert Sanderson, and Yanbo J. Wang, “Statistical Identification of Key Phrases for Text Classification”, Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern