UNIVERSIDAD DE DEUSTO

Facultad de Informática

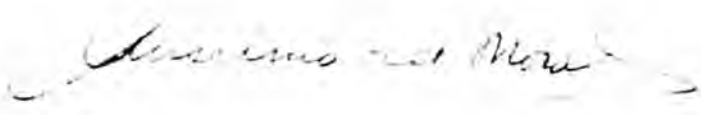
Efectividad y Dinamismo en la Recuperación Documental mediante Análisis Cluster

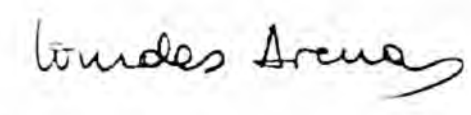
TESIS DOCTORAL presentada por Lourdes Arenas Alegría

DIRIGIDA por el Dr. Anselmo del Moral Bueno

El Director

El Doctorando





Marzo, 1991

MI AGRADECIMIENTO

Al Director de esta Tesis, ***Dr. Anselmo del Moral***, sin cuya ayuda no hubiera sido posible su realización.

A ***Mr. Les Klein***, Director Técnico de SYNERGY LTD. (Reino Unido), por los conocimientos que me transmitió durante mi estancia en Londres.

A mi padre, ***Benjamin Arenas***, por sus valiosos consejos.

A ***Javier García-Tejedor***, Director del departamento de Robótica y Visión Artificial de ROBOTIKER, por sus correcciones y sugerencias.

Al ***Programa COMETT de la C.E.E.***, que me concedió una beca en el curso 89-90, financiando parcialmente mi estancia en Inglaterra donde se llevó a cabo gran parte de la investigación.

A ***Beatriz Galan*** y a ***Iciar Rodríguez***, colaboradoras del departamento de Sistemas Informáticos y Computación de esta facultad, por la ayuda que me han prestado.

Y a todos aquellos que me han apoyado durante la elaboración de esta Tesis.

A Javier.

RESUMEN

El objetivo de esta Tesis es presentar un estudio detallado de un Sistema de Almacenamiento y Recuperación de la Información, referido a un centro documental donde se realizan investigaciones en alguna materia específica. Se analiza cada una de las partes que lo componen (entrada de la información; indexación y almacenamiento; interface con el usuario; búsqueda y recuperación; y salida de la información) diseñando algunas de ellas y explicando las diferentes posibilidades y tipos existentes de las otras.

También, se estudia la necesidad de utilizar algún tipo de clasificación en el sistema. Se propone un Algoritmo de Clasificación basado en la Teoría del Análisis Cluster, que nos proporciona una distribución estable y uniforme de los documentos en clases. Posteriormente se analiza la implantación del algoritmo propuesto en el sistema que se ha descrito.

ABSTRACT

The objetive of this Thesis is to introduce a detailed report of an Information Storage and Retrieval System, based on a documental centre where research is carried out on a particular scientific subject. Each component part is analyzed (input, indexing, storage, user interface, search and retrieval, output) designing some of them and explaining the different possibilities and existing types of the others.

This work also studies the need for a classification system, proposing a Classification Algorithm based on the Cluster Analysis Theory, that provides a stable and uniform distribution of the documents into classes. Finally the implementation of this algorithm in the system described is analyzed.

INDICE

CAPITULO 2: ESTADO DEL ARTE

CAPITULO 3: ESQUEMAS GRAFICOS DEL SISTEMA

CAPITULO 4: ENTRADA DEL DOCUMENTO

CAPITULO 5: INDEXACION DEL DOCUMENTO

CAPITULO 6: DICCIONARIO SEMANTICO O THESAURUS

CAPITULO 7: RECUPERACION

CAPITULO 8: SALIDA DE LA INFORMACION

CAPITULO 9: ALGORITMO DE CLUSTERING

ANEXO A: DOCUMENTACION DEL ALGORITMO

CAPITULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1. PROBLEMATICA HISTORICA DEL ALMACENAMIENTO DE DOCUMENTOS

Eliminar el soporte de papel de las oficinas, archivos, bibliotecas, etc., ha sido el sueño de los investigadores durante décadas. Dado el creciente aumento de éste, la cuestión de su almacenamiento se ha ido convirtiendo en un problema de difícil solución, que analizamos a continuación.

En España, por ejemplo, la ley obliga por razones de seguridad y en prevención de posibles problemas que pudieran surgir en el futuro, a guardar durante cinco años toda la documentación, en cualquier grande, mediana o pequeña empresa.

En el caso de un archivo, biblioteca ó centro documental, es fácilmente apreciable la enorme cantidad de información almacenada, además de que día a día crece la acumulación, puesto que en ella se producen muchas altas y sin embargo, casi ninguna baja.

Esto llega a convertirse en un auténtico problema: tal almacenamiento de información puede ocupar hasta habitaciones enteras. Mientras que la montaña de papel va creciendo, el sueño se hace cada vez más imposible. Más pronto o más tarde, *el papel desbordará la capacidad real de almacenamiento* dado que el conocimiento humano y las fuentes de información crecen sobre un 25% cada año [HAMP89].

También debemos considerar la dificultad que se deriva de tal acumulación de papel, en el momento de *encontrar una información en concreto*, lo cual puede suponer una búsqueda entre archivadores ó habitaciones enteras.

Hay otra cuestión igualmente problemática, que resulta de vital importancia para estos centros de almacenamiento: *la seguridad de la información*. Almacenando la información en papel existe un gran peligro de pérdida de información por distintas razones como pueden ser: robos, incendios, etc..

En un principio vamos a manejar el término "oficina" de manera general, para posteriormente definir el tipo de entidad concreta a la que nos vamos a referir.

1.2. SOLUCION INFORMATICA

La única solución a este problema de almacenamiento, consiste en almacenar la información de una forma más compacta y segura, y esto nos lleva directamente al ***uso del ordenador***.

La informática desde sus orígenes, ha intentado dar solución a este problema, almacenando en los distintos soportes, que con el tiempo han ido evolucionando, lo que antiguamente se guardaba en carpetas o armarios.

Afortunadamente, según va creciendo el volumen de papel, la capacidad de los ordenadores también crece, consiguiendo mantener el sueño vivo. Desde las configuraciones de ordenador con poca y cara memoria de los años 70, a las capacidades del orden de Gigabytes de hoy en día, la informática ha evolucionado mucho. Pero todavía le queda un amplio camino por recorrer.

Actualmente, comienza a ser habitual el término *Paperless Office* (Oficina sin papel). Es la oficina ideal, aquella en que todas las mañanas como primera tarea, todos los documentos, registros, cartas, etc., recibidos ó generados el día anterior se introducen de una forma u otra en el ordenador, desechándose posteriormente el documento físico. Esta, como su nombre indica, es una oficina sin

papel, en la que la preocupación por la necesidad de ampliación de espacio de almacenaje, se convierte en la preocupación por la necesidad de ampliación de memoria masiva en el ordenador.

El problema que surge inmediatamente (una vez que se dispone del sistema mecanizado), es el de los métodos a utilizar para la *localización y recuperación de un documento en concreto*. Es decir, una vez que se dispone del ordenador, además de estudiar la manera de introducir la información, también hay que analizar cómo y en dónde se debe almacenar, cómo se debe indexar (denominar dentro del ordenador) para su posterior localización, y por último cómo se va a realizar la búsqueda de una información concreta y qué mecanismos se van a utilizar para ello.

Esto indica que la mecanización de una empresa, oficina, biblioteca, etc., no sólo consiste en la entrada y almacenamiento de la información, si no también en un complejo proceso de indexación y mantenimiento, junto con las posteriores estrategias de búsqueda y extracción.

1.3. OBJETIVO Y ESTRUCTURA DE LA TESIS

1.3.1. OBJETIVOS

En esta tesis, diseñamos lo que puede ser un ***Sistema Informático de Almacenamiento y Recuperación de la Información*** que solvente los problemas antes descritos a todo centro de documentación donde la acumulación de papel se convierte en un problema importante. Es decir, un sistema que proporcione un ambiente de almacenamiento y recuperación efectivo y eficiente.

La "oficina" a la que nos vamos a referir en concreto, es un centro de almacenamiento, de la información tratada en investigaciones de áreas específicas, ya que el tratamiento de la información se realiza mediante la utilización de diccionarios de vocabulario controlado. Es obvio que, los centros documentales de ámbito extendido ó las bibliotecas, requieren otro tipo de tratamiento de la información.

En estos centros, se reciben diaria, semanal y mensualmente cientos de artículos y revistas especializadas en la materia objeto de estudio, y también se generan en ellos, con parecida frecuencia los llamados "reports" ó artículos, que bien simplemente se almacenan, ó se exportan para su publicación.

Por lo tanto, el documento con el que vamos a trabajar, va a ser de un tamaño, por lo general, de pocas páginas, y todas ellas conteniendo un texto narrativo sobre los avances y desarrollos del tema investigado en el centro ó de interés de los que en él trabajan.

Proponemos, en la sección 5.2, *un modelo de registro-índice para representar al documento* dentro del sistema; así como, en la sección 5.6, su *normalización en el sistema*, para que posteriormente pueda ser comparado con la petición de información del usuario.

Presentamos también, en la sección 7.2, un interface de usuario describiendo tanto *una forma estándar de query* que se adapta a la estructura del sistema, como *un algoritmo que realiza la transformación* de la petición de información del usuario en Lenguaje Natural a esta forma estándar. En la sección 7.4, se propone *la normalización del query en el sistema*, para finalizar planteando *el enfrentamiento entre el query y el documento* en la sección 7.6.

Proponemos un *Algoritmo de Clasificación* (sección 9.3), que usando la teoría del Análisis Cluster, genera clases de similaridad entre los documentos almacenados en el sistema, comprobando posteriormente cómo afecta y mejora su utilización a los resultados de la recuperación.

1.3.2. ESTRUCTURA DE LA TESIS

En el capítulo II, realizamos un análisis del *estado del arte* en el área de Almacenamiento y Recuperación de la Información.

En el capítulo III, presentamos los diferentes *esquemas gráficos* de un Sistema de Almacenamiento y Recuperación de la Información. En varias figuras reflejamos primero la concepción general, y posteriormente cada parte en concreto, detallando todos y cada uno de los procesos que en el sistema se llevan a cabo.

El capítulo IV, trata de la *entrada del documento en el ordenador*. Es decir, se analizan las diferentes formas existentes para introducir el documento, exponiendo las nuevas tecnologías del mercado. Hablamos de un equipo compuesto por *Scanner-OCR (Lector de Caracteres Opticos)*, que permite realizar la entrada del documento de una manera mucho más rápida y cómoda.

Como consecuencia de esto, se exponen los motivos de la necesidad de almacenamiento masivo, tal como *Almacenamiento Optico*.

El **capítulo V** trata sobre el ***almacenamiento y la indexación del documento***. Es decir, una vez que el documento se ha introducido en el ordenador, no está aún preparado para su recuperación. El documento ha de sufrir una serie de procesos de indexación, dándonos como resultado una denominación específica (***registro-índice***), para que luego pueda ser fácilmente clasificado, localizado y extraído mediante alguna función de recuperación.

En el capítulo VI, explicamos el ***Thesaurus como herramienta*** de trabajo. Su definición y estructura, así como su necesidad de uso. Este, es un diccionario semántico que almacena las llamadas ***categorías semánticas*** de las palabras, es decir, las relaciones de significado entre las palabras. Dada su importancia actual en todo Sistema de Almacenamiento y Recuperación de la Información, hemos creído necesario crear para su explicación, un capítulo a parte.

El capítulo VII, trata sobre el ***Interface hombre-máquina***. Primeramente, exponemos las diferentes formas de diálogo más utilizadas hoy en día, para posteriormente centrarnos en la explicación de un modelo de diálogo en Lenguaje Natural, dado que hemos considerado, que es la forma más general y más manejable por todo tipo de usuario.

Para concluir este capítulo, se introducen brevemente algunos ***Modelos de Recuperación*** existentes hoy en día, y la enumeración de sus ventajas e inconvenientes.

El capitulo VIII, trata de lo que resta en todo Sistema de Almacenamiento y Recuperación de la Información: la ***salida de los documentos*** seleccionados y recuperados mediante la función de recuperación expuesta en el anterior capítulo.

En el capítulo IX, proponemos nuestro ***Algoritmo de Generación Automática de Clases***, especialmente ideado y diseñado para un Sistema de Almacenamiento y Recuperación de la Información como el que hemos ido exponiendo, es decir, un sistema que almacene, indexe, y recupere los documentos de la manera explicada. Se explica su necesidad, desarrollo y aplicación.

En el capítulo X, exponemos los datos de entrada al documento y los ***resultados obtenidos*** en las diferentes pruebas.

En el capítulo XI, realizamos un ***estudio del algoritmo propuesto***, para posteriormente obtener conclusiones sobre su grado de aceptabilidad y complejidad.

El capítulo XII, es un ***análisis de conclusiones*** y propuestas de nuevas líneas de investigación.

CAPITULO 2

ESTADO DEL ARTE

Analizamos el estado del arte de los ***Sistemas de Almacenamiento y Recuperación de Información***, para cada una de las partes ó áreas en que se subdivide:

2.1. ALTERNATIVAS DE ENTRADA

Muchos de los sistemas actuales, soportan tanto *información estructurada*, es decir, orientada a campos, como *segmentos de texto no estructurados*.

Estos campos estructurados (con su formato de longitud fija y tipo de datos), frecuentemente almacenan información descriptiva y términos-índice manualmente asignados, y que han sido extraídos del texto del documento completo ó del resumen (abstract) del documento.

Estos sistemas emplean una entrada en tiempo real, mediante formatos de pantalla predefinidos, que generalmente caracterizan los niveles de campo mediante espacios en blanco para la entrada de los valores de datos.

Para facilitar la entrada en tiempo real, muchos sistemas incluyen un editor con las capacidades básicas de un procesador de textos que permiten la corrección de errores, inserción, borrado y movimiento de la información tecleada previamente. Este tipo de entrada se utiliza para información relativamente corta, como anotaciones, abstracts de documentos, etc..

Para información extensa, aunque posible, es poco práctica este tipo de entrada. La alternativa más utilizada es, la transferencia de la información desde otra fuente que pueda ser leída por la máquina.

Todos los Sistemas de Almacenamiento y Recuperación de la Información pueden aceptar documentos en forma de *ficheros ASCII generados por procesadores de texto*. Algunos de estos sistemas, aceptan solo los ficheros de documentos generados por procesadores de texto específicos, como WORDSTAR, WORDPERFECT u otros, en cuyo caso, se preserva el formato y la editabilidad del documento. Por ejemplo, el sistema [INFO DB+] del software Henco, acepta documentos generados por procesadores de la familia del VAX de DIGITAL [SAFF89].

Como alternativa, muchos de estos sistemas, pueden ya aceptar los *ficheros ASCII generados mediante el equipo SCANNER-OCR*.

También existen los sistemas que aceptan los ficheros de datos formateados por algún software específico como [dBase III] ó [dBase IV], [ASHTON-TATE], etc..

2.2. INDEXACION

La mayoría de los sistemas de hoy en día, realizan las *tareas de indexación de manera automática*, basándose en palabras que se derivan del texto del documento ó de campos de datos marcados como palabras clave cuando se generó el texto.

Como alternativa a la extracción de términos índice de manera automática, algunos sistemas como [INFO DB+] y [CAIRS] (también de DIGITAL), emplean la *indexación semi-automática* basada en un operador que marca las palabras contenidas en bloques de texto.

En lo relativo al tipo de índice, el índice sencillo (formado por una sola palabra), es el más utilizado; aunque por supuesto, también existen los sistemas que implementan frases-índice, como por ejemplo, el sistema [DOCU/MASTER].

Muchos sistemas utilizan estructuras de índice invertido, que consiste en listas de palabras ordenadas alfabéticamente con punteros a su localización en el texto,

posibilitando luego la visualización en tiempo real, de secciones alfabéticas de la lista, para facilitar la tarea de selección del término de búsqueda. En el sistema [CONCEPT FINDER], se puede requerir la visualización del término-índice más frecuentemente usado ó últimamente usado.

Existen diversos métodos para evitar las palabras carentes de significado como entradas de índice, es decir, artículos, preposiciones, adverbios etc.. Uno de ellos, el utilizado en el sistema [DOCU/MASTER], consiste en la utilización de las llamadas *stoplist* que no son más que listas que incluyen a todas estas palabras. El sistema [QL SEARCH] (desarrollado por QL Systems) utiliza *códigos especiales* para suprimir la indexación de palabras específicas. Otros sistemas soportan con este mismo objetivo, las *listas go ó keep*, que guardan las palabras que podrían ser usadas invariablemente como índices.

Muchos sistemas implementan un *diccionario semántico denominado THESAURUS*, que almacena las relaciones jerárquicas y de sinonimia entre palabras. También puede ser usado para relacionar los nombres con sus abreviaturas ó acrónimos como ocurre en el sistema [CAIRS].

2.3. INTERFACE DE USUARIO

De entre todas las posibilidades existentes en cuanto a la forma de realizar el interface entre el usuario y el ordenador, en un Sistema de Almacenamiento y Recuperación de la Información, tres son las más utilizadas:

- interface orientado a comandos
- interface orientado a menús
- interface en lenguaje natural

Los sistemas que disponen del *interface orientado a comandos*, requieren la memorización de las palabras clave que inicializan y controlan las operaciones específicas. Aunque se requieren manuales de operador, los lenguajes de comandos son generalmente fáciles de aprender y convenientes de usar.

Mientras que los operadores experimentados suelen preferir la entrada directa mediante comandos, otros sistemas emplean *menús, formatos de pantalla e instrucciones on-line* para guiar al usuario inexperto. El sistema [BRS/SEARCH] (desarrollado por BIBLIOGRAPHIC RETRIEVAL SERVICES) ofrece la posibilidad de elegir entre diferentes interface orientados a menus, mientras que, otros sistemas, como por ejemplo, [BASIS CONTEXT], permiten la optimización de menús para direccionar requerimientos de aplicaciones especiales.

Hay pocas diferencias entre los software orientados a pantallas y los orientados a comandos. Algunos sistemas combinan ambos modos de operación. Por ejemplo, algunos de los orientados principalmente a comandos, proporcionan pantallas solo para algunas tareas. Y existen también los sistemas que proporcionan la posibilidad de elegir el modo de interface. Por último, destacar que, el sistema [ASSASSIN] permite cambiar de un modo a otro de operación en cualquier punto de una sesión en tiempo real.

Como tercera alternativa, existen los sistemas, por ejemplo [STATUS], que soportan el *interface en lenguaje natural*, especialmente diseñados para usuarios novatos y sin ninguna experiencia con el ordenador. Estos, permiten la entrada de las especificaciones de búsqueda, en forma de sentencia en lenguaje natural.

Algunos sistemas ofrecen además, *modos de operación internacionales*, es decir, permiten al usuario elegir el idioma tanto en el modo comando como en el de pantalla. Por ejemplo, el sistema [TRIP] de Paralog ofrece la posibilidad de elegir el idioma entre el inglés y el sueco.

2.4. CAPACIDADES DE RECUPERACION

Hay Sistemas de Almacenamiento y Recuperación de la Información que se basan en el ***algebra booleana*** para llevar a cabo las tareas de recuperación. Es decir, se utilizan los operadores booleanos ***AND***, ***OR*** y ***NOT*** para unir los términos índice y formar especificaciones de búsqueda más complejas, mejorando claramente la calidad de la respuesta que el sistema proporciona al usuario.

El operador booleano ***AND*** recupera los documentos que satisfacen ambos términos índice (ó especificaciones de entrada), mientras que el operador booleano ***OR*** recupera todos los documentos que satisfacen alguna ó ambas especificaciones de entrada.

En lo que respecta al operador ***NOT***, se recuperan todos los documentos que no contengan el término-índice en cuestión.

Algunos sistemas, también soportan el operador exclusivo ***XOR***, que recupera los documentos que satisfacen, una y solo una, de las dos especificaciones de búsqueda; y el operador ***NAND***, que recupera los documentos que no satisfacen ninguna de las dos especificaciones de búsqueda indicadas.

En la mayoría de los sistemas, se pueden combinar más de un operador booleano en el query ó petición de búsqueda de información. En tales casos, se usan paréntesis para controlar la secuencia de ejecución de los operadores.

Como alternativa de estos sistemas booleanos, existen los sistemas de recuperación que se basan en la similaridad existente entre la petición de búsqueda de información y la definición (registro índice) de cada documento.

Caben destacar los sistemas que soportan *diccionarios semánticos ó thesaurus*, en donde queda relacionado cada término con sus sinónimos, sus términos asociados, sus términos más generales y específicos, como ocurre en el sistema [CAIRS]. De tal manera que en los sistemas que trabajan principalmente en lenguaje natural, el usuario no tiene por que conocer los términos concretos que se han utilizado en la descripción del documento.

También existe el sistema [TEXTBANK/PC] que, aunque no soporta el thesaurus, permite la creación de un *macro-comando* que utiliza el operador booleano *OR* que relaciona sinónimos para la búsqueda automática cada vez que se llama a la macro.

En cuanto a los sistemas que realizan el interface de usuario orientado a comandos ó a pantallas, algunos de ellos, facilitan la selección de términos de búsqueda mediante la visualización alfabética de los índices.

2.5. CARACTERISTICAS DE SALIDA

Como respuesta inicial algunos Sistemas de Almacenamiento y Recuperación de la Información visualizan el *número de documentos* que satisfacen la determinada petición de información. En algunos casos, también se indica el número de ocurrencias de los términos de búsqueda. Si se recuperan demasiados ó demasiado pocos documentos, se puede redefinir la petición de búsqueda (manual ó automáticamente) ó introducir especificaciones de búsqueda alternativas.

Hay sistemas que ofrecen la facilidad de *visualizar los segmentos del documento* que contienen los términos-índice o que satisfacen las especificaciones de recuperación.

Algunos sistemas utilizan algoritmos basados en la frecuencia de aparición de los términos de búsqueda, para *ordenar los documentos recuperados* por su relevancia ó importancia con respecto a la petición de información. El sistema [STAIRS] (de IBM) ofrece la posibilidad de elegir entre varios algoritmos de ordenación, y el sistema [PERSONAL LIBRARIAN] de Cucumber Information Systems ordena los documentos recuperados por su supuesta relevancia.

En muchos sistemas, los documentos asociados a una especificación particular de búsqueda, se agrupan en clases para su posterior modificación, recuperación, etc..

2.6. SEGURIDAD

Algunos Sistemas de Almacenamiento y Recuperación de la Información incluyen *accesos controlados por medio de palabras de paso* asignadas por el system manager ó por el encargado de desarrollar de la aplicación.

Las previsiones de seguridad más flexibles proporcionan accesos controlados vía palabra de paso a niveles de fichero, registro, campo.

En estos casos, el usuario ni siquiera sabe que hay información a la que no esta autorizado a acceder, eliminándose del registro los campos no utilizados, cuando éste es recuperado.

CAPITULO 3

ESQUEMAS GRAFICOS DEL SISTEMA

3.1. *INTRODUCCION*

La primera figura que presentamos (figura 3.1), muestra una representación del esquema general de un Sistema de Almacenamiento y Recuperación de la Información. Está dividido en tres partes:

1.- **ENTRADA:** Todo sistema tiene dos entradas de naturaleza bien distinta:

a.- **ENTRADA-1:** ***Entrada del usuario.*** Es decir, es la entrada de la petición de información ó query al sistema.

b.- **ENTRADA-2:** ***Entrada del documento.*** Es la llegada ó entrada del documento al sistema.

2.- **TRATAMIENTO:** La estructura del sistema de información, consiste en varios procesos:

a.- **PROCESO-1:** Cada vez que se produce en el sistema una ENTRADA-1, inmediatamente se realiza el PROCESO-1 ó ***adquisición del query***, que consiste en la identificación de la necesidad del usuario. Significa, el conocimiento por parte del sistema, de lo que el usuario necesita en ese momento concreto.

b.- **PROCESO-2:** De igual forma, cada vez que se produce una ENTRADA-2 en el sistema, inmediatamente se produce un PROCESO-2 de *adquisición del documento*. Consiste en, la captura del documento por parte del sistema.

c.- Los resultados de ambos procesos, se pueden considerar como las entradas a la estructura interna del sistema.

d.- **PROCESO-3:** La salida ó resultado de la parte más interna del Sistema de Almacenamiento y Recuperación de la Información, pasa por este proceso que podríamos denominar: ***puesta a disposición del usuario***. Es decir, los documentos recuperados son entregados al usuario.

3.- **SALIDA**: Como salida del ***Sistema de Almacenamiento y Recuperación de la Información***, obtenemos dos salidas bien diferentes:

a.- **SALIDA-1**: Se deriva conjuntamente de la ENTRADA-1 y del PROCESO-1 y es la obtención por parte del usuario del *conocimiento sobre los servicios disponibles* y sobre la forma de utilización del sistema.

b.- SALIDA-2: Se deriva del PROCESO-3, y consiste en la obtención por parte del usuario, *del documento físico*, a través de la impresora.

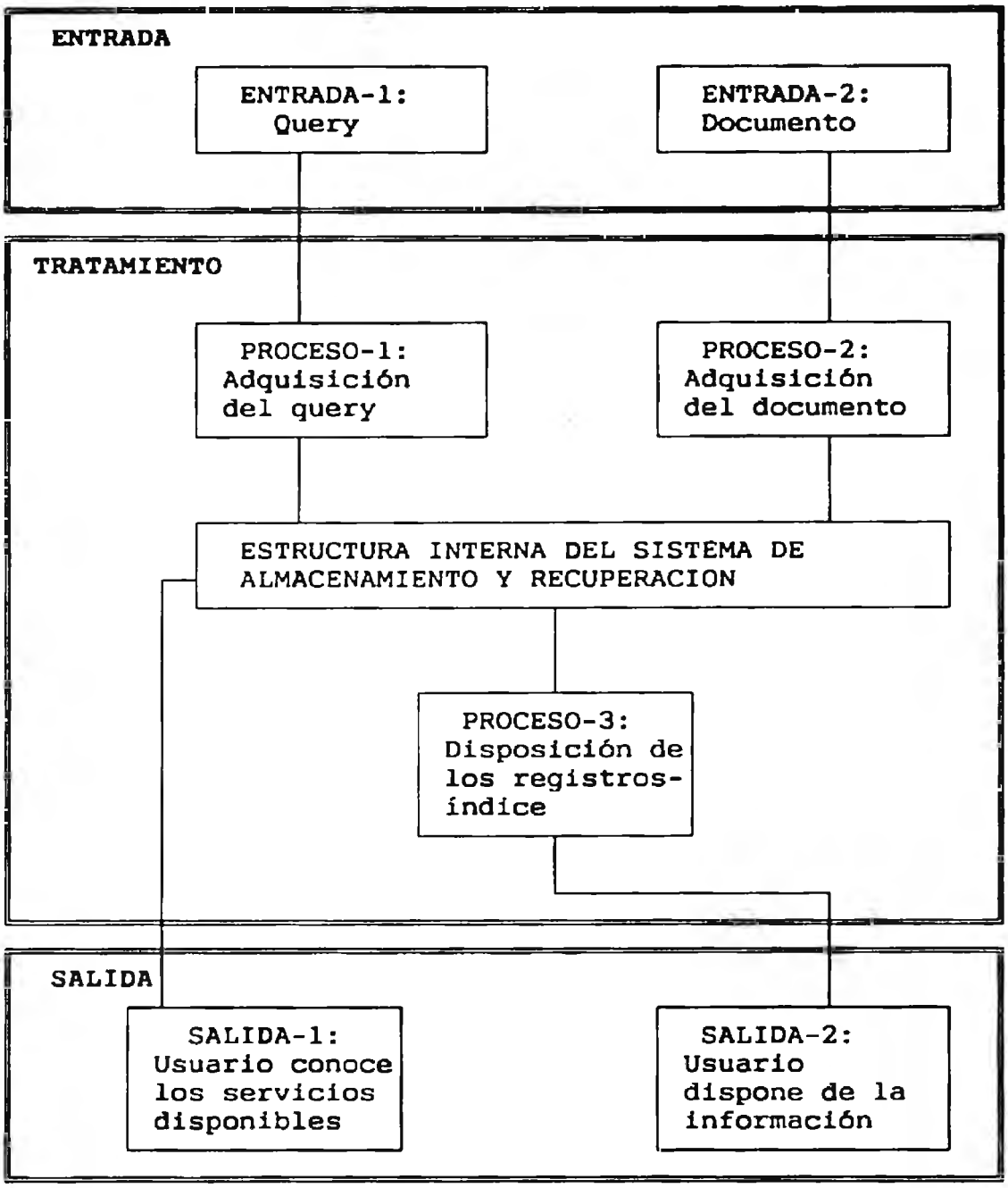

ENTRADA
ENTRADA-1:
Query
ENTRADA-2:
Documento
TRATAMIENTO
PROCESO-1:
Adquisición
del query
PROCESO-2:
Adquisición
del documento
ESTRUCTURA INTERNA DEL SISTEMA DE
ALMACENAMIENTO Y RECUPERACION
PROCESO-3:
Disposición de
los registros-
índice
SALIDA
SALIDA-1:
Usuario conoce
los servicios
disponibles
SALIDA-2:
Usuario
dispone de la
información

Figura 3.1

3.2. ENTRADA

En la figura 3.2 analizamos más detalladamente las dos entradas del Sistema de Almacenamiento y Recuperación de la Información.

1.- **ENTRADA-1**: Cada vez que un usuario tiene un problema y se acerca al sistema con la intención de que éste le de soluciones, se produce una entrada de éste tipo. Es decir, *el usuario realiza una petición de información*, que es lo que designamos por *query*.

2.- **ENTRADA-2**: Al inicio del día, se encuentran acumulados todos los documentos llegados y generados en el día anterior. Se revisan rápidamente y se envían a tres destinos diferentes:

a.- los documentos que no son lo suficientemente interesantes, se *desechan como entrada*.

b.- los documentos que van a entrar en el sistema *vía procesador*.

c.- los documentos que van a entrar en el sistema *vía scanner*.

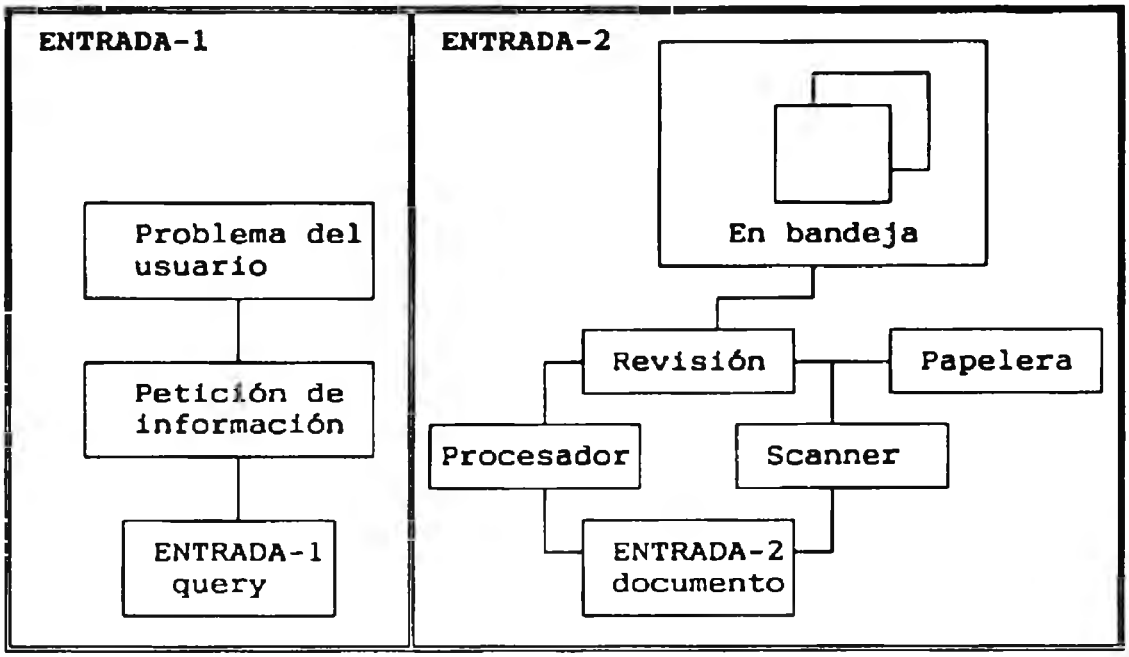

ENTRADA-1
Problema del usuario
Petición de información
ENTRADA-1 query
ENTRADA-2
En bandeja
Revisión
Papelera
Procesador
Scanner
ENTRADA-2 documento

Figura 3.2

3.3. *TRATAMIENTO O ESTRUCTURA INTERNA*

En la figura 3.3, analizamos al detalle la estructura interna del Sistema de Almacenamiento y Recuperación de la Información. Cada una de las dos diferentes entradas que se producen en el sistema, dan lugar a también diferentes procesos. Posteriormente, los resultados de estos procesos, se enfrentan y comparan en procesos comunes.

PROCESO-1.1: Nada más producirse una ENTRADA-1, se lleva a cabo el PROCESO-1.1, consistente en *transformar la necesidad del usuario* en una forma estándar comprensible por el sistema. Para ello será necesario usar los diferentes diccionarios disponibles.

PROCESO-1.2: El resultado del PROCESO-1.1, es decir, el query en forma estándar, se *almacena* en el sistema.

PROCESO-2.1: Nada más producirse una ENTRADA-2, se lleva cabo el PROCESO-2.1, consistente en la *indexación del documento*, obteniéndose un registro-índice para representarlo.

PROCESO-2.2: *Almacenamiento*. Se almacena tanto el resultado del PROCESO-2.1, es decir, el índice del documento, como del propio documento.

PROCESO-3: Se produce una comparación entre la representación del query en forma estándar y el registro-índice que representa al documento. Este enfrentamiento es la ***recuperación propiamente dicha***.

PROCESO-4: La comparación del PROCESO-3 produce la ***selección*** de unos documentos determinados.

PROCESO-5: ***Verificación de la satisfacibilidad*** de los documentos obtenidos en el PROCESO-4. Es decir, si el usuario encuentra que responden bien ó no a su necesidad de información.

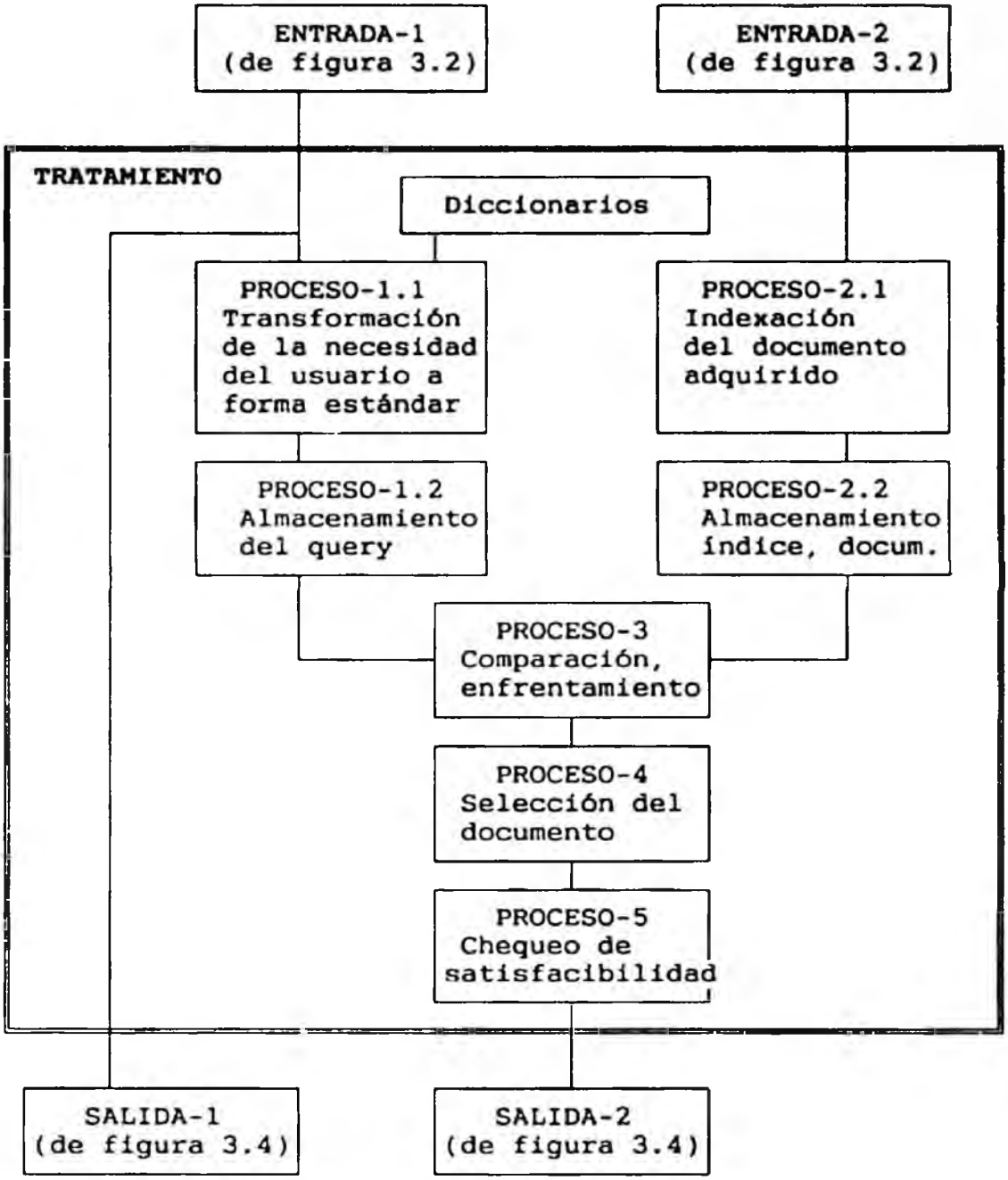

ENTRADA-1
(de figura 3.2)
ENTRADA-2
(de figura 3.2)
TRATAMIENTO
Diccionarios
PROCESO-1.1
Transformación
de la necesidad
del usuario a
forma estándar
PROCESO-2.1
Indexación
del documento
adquirido
PROCESO-1.2
Almacenamiento
del query
PROCESO-2.2
Almacenamiento
índice, docum.
PROCESO-3
Comparación,
enfrentamiento
PROCESO-4
Selección del
documento
PROCESO-5
Chequeo de
satisfacibilidad
SALIDA-1
(de figura 3.4)
SALIDA-2
(de figura 3.4)

Figura 3.3

3.4. SALIDA

Y en la figura 3.4, explicamos las dos salidas del Sistema de Almacenamiento y Recuperación de la Información.

SALIDA-1: Esta salida del sistema es, el conocimiento por parte del usuario tanto del funcionamiento, como de los servicios que ofrece el sistema.

SALIDA-2: Una vez obtenida la lista de registros-índice seleccionados (PROCESO-4) y comprobada su satisfacibilidad (PROCESO-5), se buscan los documentos en cuestión en la Base de Datos (mediante su localización dada por el índice) y posteriormente se *imprimen* para proporcionar al usuario *el documento físico en papel*.

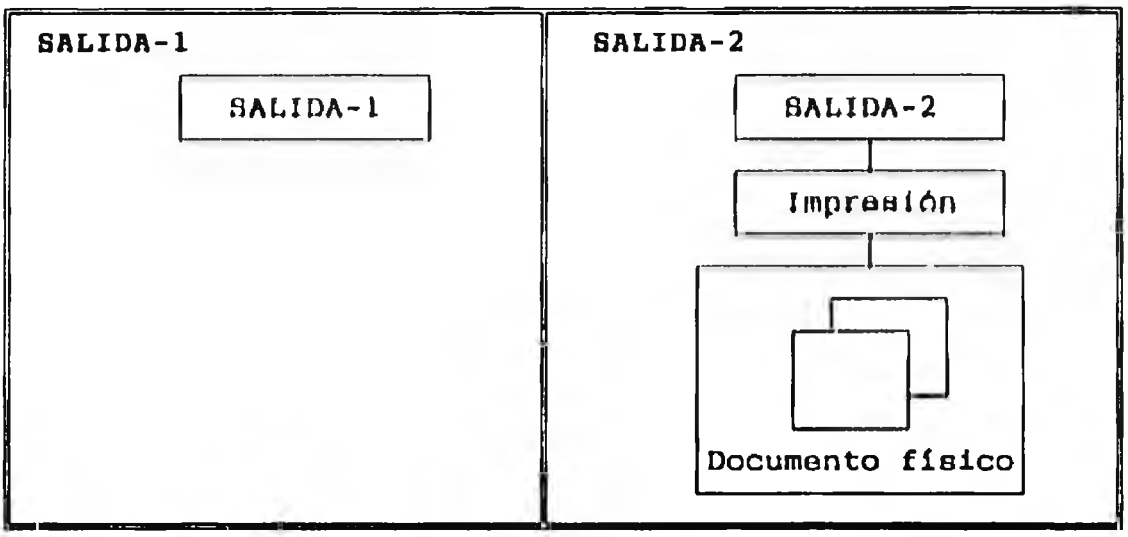


Figura 3.4

CAPITULO 4

ENTRADA DEL DOCUMENTO

4.1. FORMAS DE INTRODUCCION DEL DOCUMENTO

Anteriormente, la única posibilidad para la entrada de un documento en el ordenador, consistía en teclearlo usando algún procesador de textos, siendo esto muy lento. Supondría además, muchísimo trabajo en las oficinas y empresas actuales, en las que diariamente se reciben y generan gran cantidad de documentos de todo tipo. Existe una alternativa que, día a día se esta desarrollando y aplicando a más campos para aumentar la productividad. Hablamos de la tecnología de *SCANNER-OCR* (Optical Character Reader), que facilita la tarea de entrada del documento de forma mucho más rápida [COX89].

Previamente a la explicación detallada de la entrada vía scanner, vamos a incluir dos ventajas que ofrece la entrada convencional vía procesador, con respecto a la entrada vía scanner:

a.- Produce directamente el fichero de texto, con lo cual, se evita el uso del software OCR con sus posibles errores.

b.- El fichero de texto ocupa en memoria mucho menos espacio que la imagen digitalizada del documento.

4.2. ENTRADA VIA SCANNER-OCR

4.2.1. INTRODUCCION

Introduciendo el documento en el ordenador por medio del scanner, obtenemos una *imagen del documento*, es decir un mapa de bits. Después de esto, un apropiado software de OCR, lee dicho mapa de bits convirtiéndolo en un *fichero ASCII*.

El proceso de entrada, sería entonces el siguiente:

1.- *Situación del documento* en/dentro del scanner (dependiendo del modelo).

2.- *Activación del scanner* y proceso de digitalizado.

3.- *Obtención de la imagen* del documento en el disco.

4.- *Activación del software OCR*, que reconoce los patrones de puntos y les asigna el carácter correcto, hasta obtener el fichero completo.

5.- *Corrección de errores*. Podemos abrir el fichero de texto obtenido, con algún procesador y corregir los posibles errores que ha cometido el OCR en su conversión.

La mayoría de estos sistemas están por lo tanto, formados por un equipo hardware y software especializado [BIDA89]:

1º Proceso de digitalización llevado a cabo por un hardware no inteligente: mediante un scanner se obtiene una imagen ó mapa de bits del documento. Existen diversos tipos de scanners como analizaremos a continuación.

2º Proceso de transformación llevado a cabo por *un software inteligente*: A pesar de toda la efectividad y rapidez de este tipo de entrada, la imagen así obtenida, no dice nada sobre los contenidos del documento. Se hace necesario que el sistema OCR convierta el mapa de bits obtenido durante el digitalizado de la imagen, en un fichero de texto legible.

4.2.1.1. MODELOS DE SCANNER

Existen diversas categorías de scanner siendo las principales [COX89]:

a.- scanner de arriba (overhead)

b.- scanner plano (flatbed)

c.- scanner de introducción de hoja (sheet-fed)

Todos ellos producen como resultado un mapa de bits, aunque cada uno de ellos maneja el documento físico de diferentes maneras:

1.- En el scanner overhead, el documento se sitúa boca arriba en una plataforma y se lee mediante un dispositivo de lectura.

2.- En el modelo flatbed por el contrario, el documento se sitúa boca abajo en un plato de cristal y un sensor lo lee de una parte a otra.

3.- En el modelo sheet-fet, se introduce un borde del documento en el scanner, y poco a poco va pasando por el sensor.

4.2.1.2. TIPOS DE OCR

Los lectores de caracteres ópticos, fueron desarrollados para extraer inteligentemente la información de documentos. Principalmente existen dos tipos:

1º Los sistemas basados en el método estadístico ó de *Pattern Matching*, comparan el fichero con el conjunto de letras almacenado, es decir, se basan en la comparación bit a bit del carácter con los modelos almacenados en memoria. Es un proceso estadístico, no demasiado inteligente, ya que consiste en el recuento de coincidencias para elegir la mejor. Aunque es un método rápido, no trabaja con letras en cursiva, diferentes tamaños, ó cualquier tipo no similar a lo almacenado [COX89].

2º Hay otros sistemas basados en el reconocimiento de formas. Son los métodos sintácticos ó de *Extracción de Características*. Así, para reconocer una "A" mayúscula, basta comprobar la característica de todas las Aes: "dos barras verticales unidas por arriba y separadas por abajo, con una barra horizontal uniéndolas". Este método, es más lento, pero mucho más flexible que el de *Pattern Matching*, puesto que se puede usar para interpretar manuscritos. Dado que ningún carácter tiene más de 20 características diferentes, este método es bastante operativo [BIDA89].

En el mundo real, mucha información útil llega de manera no estándar. Es decir, los caracteres pueden ser de diferentes tamaños y formas. Hoy en día, existen OCR muy flexibles, conteniendo sofisticados algoritmos con la suficiente inteligencia como para interpretar la mayoría de los textos. Su arquitectura dispone de procesadores de alta velocidad.

4.2.2. RESOLUCION

La mayoría de los scanners para PC, son de una resolución de 300 d.p.i. (puntos por pulgada). Esto limita mucho su utilidad sobre todo para los documentos comprimidos ó para la escritura diminuta ya que el software OCR necesita alta resolución para poder distinguir los caracteres.

En documentos reducidos de letra pequeña, por ejemplo de 1/16 de pulgada de altura, un scanner de resolución de 300 d.p.i. proporciona menos de 20 x 20 puntos por cada letra, siendo esto insuficiente.

Ahora, con la gran difusión de impresoras laser y amplia gama de fonts (juego de caracteres), los documentos vienen con letras de tipo cursiva, negrita, múltiples columnas, y otras complicaciones. Esto significa que, los scanners necesitan alta resolución, y por lo tanto, el sistema de ordenador necesita más y más memoria y rápidos procesadores.

Como veremos posteriormente, una página digitalizada puede ocupar más de 1 Mb, mientras que la misma página introducido mediante el procesador de textos ocupa 1 Kb, es decir, 1/1000 de su espacio.

4.2.3. TIPOS DE INFORMACION

Esta tarea de digitalizado ó *conversión de la información legible por el hombre, en información legible por el ordenador*, se complica bastante por el hecho de que el documento puede contener distintos tipos de información [BIDA89].

a.- El tipo más sencillo de documento, es aquel que solo contiene información alfanumérica: mayúsculas, minúsculas, alfabeto, signos de puntuación, números y un conjunto de símbolos predefinidos.

Cada uno de estos 96 caracteres, se representa por un número en algún código (ASCII,...) que, fácilmente es capturado, almacenado y recuperado por el ordenador.

b.- Sin embargo, hay documentos que contienen gran cantidad de información proveniente de imágenes, como fotografías, dibujos, etc..

c.- Los documentos, también pueden contener firmas que posteriormente se necesiten para su verificación.

La necesidad de almacenar, tanto textos como imágenes, ha derivado en el desarrollo de scanners gráficos y diferentes tipos de OCR.

Un scanner gráfico es, básicamente una cámara electrónica que convierte una imagen en otra de tipo digital (mapa de bits), capaz de ser almacenada y recuperada en el ordenador. Estos dispositivos, leen muchos puntos diferentes en miles de líneas individuales disolviendo la imagen en el mapa de bits. Posteriormente, el ordenador puede almacenar, manipular, y reconstruir la imagen ó incluso los pixels en la imagen.

4.2.4. TIPOS ACTUALES DE SCANNERS-OCR

Básicamente existen 3 tipos de OCRs [BIDA89]:

1.- Lectores Inteligentes de Caracteres
2.- Lectores de Caracteres Opticos.
3.- Lectores de Formularios.

1.- Los SCANNER-OCR inteligentes son procesadores de documentos híbridos. Es decir, diferencian entre las áreas de gráficos y las áreas de texto de los documentos. En las áreas de gráficos, actúan como un scanner de gráficos, pero en las áreas de texto, reconocen los caracteres alfanuméricos.

2.- Los SCANNER-OCR normales son los que únicamente pueden leer ó interpretar textos ignorando los gráficos.

3.- Los lectores de formularios son los más especializados ya que pueden interpretar los formularios mecanografiados ó manuscritos.

4.2.5 ULTIMO DESARROLLO

El último y más revolucionario desarrollo en la tecnología OCR, ha sido la producción de un elemento del hardware llamado *Type-Reader-Card* (lector de tipos) [MORR89b].

La razón de que el OCR fuese lento en reconocer caracteres, era que solo podían ser rápidamente identificadas cierto tipo de letras. Incluso en fonts muy utilizados, la desaparición de una parte del carácter (una barra ...etc), podía hacer que el sistema diera lugar a error, requiriendo tiempo de verificación.

Este nuevo desarrollo emplea un principio totalmente diferente. Utiliza solo los primeros caracteres del documento para identificar el tipo de letra. Una vez reconocida, usa su información como base para comparar el resto del texto. Las suposiciones de bits perdidos en caracteres son entonces mucho más rápidas.

Una comparación realizada entre dos OCRs, uno con dicha tarjeta y otro sin ella, nos da resultados sorprendentes. El estándar, tarda en leer una página 180 segundos y produce una media de 50 errores, mientras que el que incorpora la tarjeta, tarda 30 segundos produciendo 6 errores al procesar la misma página [MORR89b].

4.3. NECESIDADES DE ALMACENAMIENTO

4.3.1. CALCULO DE LA OCUPACION DE MEMORIA

El espacio ocupado en memoria por la imagen del documento digitalizado, depende de tres factores:

a.- el tamaño del documento original: DIN A4, DIN A3...

b.- la resolución del scanner: número de puntos por pulgada.

c.- el algoritmo de compresión usado.

Caso 1. Cálculo del espacio ocupado por la imagen de un documento tamaño DIN A4, con un scanner de resolución de 200 b.p.i.

DIN A4: 210 mm x 297 mm = 8.26" x 11.69", redondeando valores: 9" x 12". Realizando los cálculos, (12 x 9) (pulgadas/A4) x (200 x 200) (puntos/pulgada), nos da 4.320.000 bits por página DIN A4. Y para calcular los bytes, 4.320.000/8 = 540.000 bytes/A4, que aproximadamente es 0.5 Mb.

Caso 2. Cálculo del espacio ocupado por la imagen de un documento tamaño DIN A3, con un scanner de resolución de 200 b.p.i.

DIN A3: 297 mm x 420 mm = 11.69" x 16.53", redondeando valores: 12" x 17". Realizando los cálculos, (12 x 17) (pulgadas/A3) x (200 x 200) (puntos/pulgada), nos da 8.160.000 bits por página DIN A3. Y para calcular los bytes, 8.160.000/8 = 1.020.000 bytes/A3, que aproximadamente es 1 Mb.

Como puede observarse, la cantidad de memoria necesaria se dispara. Existen diversos algoritmos de compresión que, consiguen reducir considerablemente el tamaño ocupado por la imagen en memoria. De entre ellos, el más usado es el que reduce el tamaño a la décima parte [KLEIN - SYNERGY, U.K.].

A4: 500.000 bytes ----------> 50.000 bytes ≈ 50 Kb.

Pero este tamaño aún es demasiado grande.

4.3.2. DISCO OPTICO. CARACTERISTICAS GENERALES

Para satisfacer estas necesidades, existe un tipo de almacenamiento masivo: ***ALMACENAMIENTO OPTICO*** (DISCO OPTICO). Un "jukebox" (dispositivo.que contiene grupo de discos) puede alcanzar capacidades de memoria del orden de Gigabytes [MITK89].

La tecnología óptica parece ser la ideal para aplicar en diferentes áreas [CHRI87]. Sus principales ventajas son el bajo coste y la gran capacidad de almacenamiento, así como la capacidad de largo período de almacenamiento (larga vida media del soporte), por lo que es particularmente adecuada para aplicaciones que requieren grandes capacidades.

La tecnología de los discos ópticos, tiene ciertas similaridades con la tecnología de los discos magnéticos [WARE85], [BELL83], [BELL84].

Los ópticos, están montados sobre un brazo móvil al igual que los discos magnéticos. Por lo tanto, existen retrasos similares en los tiempos de posicionamiento, tiempos de rotación y tiempos de transferencia.

Sin embargo, existen grandes diferencias de rendimiento, entre ambos tipos de dispositivos. La diferencia más importante en cuanto a recuperación, es que el mecanismo de lectura en los discos ópticos cuando está posicionado en un punto determinado, puede leer la información de las pistas cercanas a gran velocidad, sin tener que mover todo el ensamblaje óptico. Esto se realiza, moviendo espejos con inercia muy pequeña, que desvían la luz a estas pistas cercanas [WARE85], [BELL83], [BELL84].

Esto no es análogo a los discos magnéticos. Los cilindros en los discos magnéticos, están compuestos por pistas, para las que el retraso en activar las cabezas de lectura es muy pequeño en comparación con el retraso experimentado al mover el mecanismo de acceso entre diferentes cilindros. En este caso, el acceso a una pista dentro del mismo cilindro en el disco magnético, es similar al acceso de una pista dentro del mismo span (intervalo de disco) en un disco óptico.

La primera pista de un intervalo, puede ser cualquier pista del disco óptico, y los intervalos pueden solaparse, mientras que la primera pista de un cilindro es fija en el disco magnético, y los cilindros no pueden solaparse.

Esto, produce diferencias de rendimiento entre los discos ópticos y magnéticos: los tiempos de acceso y posicionamiento dentro del intervalo en el óptico y del

cilindro en el magnético son diferentes.

Cuando el tamaño del bloque es igual al tamaño del cilindro, el disco magnético, puede considerarse como un caso especial de un disco óptico con el tamaño del intervalo igual a un bloque.

Los costes estimados, vienen dados en función del número de veces que el mecanismo de acceso se mueve y de la distancia que tiene que recorrer para responder a un query dado.

Las diferencias principales con los tiempos de acceso normales, vienen dadas por la capacidad de los discos ópticos de leer datos de más de una pista cuando el mecanismo de acceso está en la misma posición; y también por la gran diferencia en la longitud de los documentos, que pueden hacer deseable en algunos casos el cruce en la limitación de pistas y en otros casos indeseable, pudiendo implicar dos ó más accesos a un bloque para recuperar un documento determinado.

Generalmente el tiempo de acceso al disco óptico, es superior que al disco magnético, debido a que los focos ópticos tienen más volumen que las diminutas cabezas de los discos magnéticos.

El almacenamiento óptico, tiene su origen en los primeros sistemas de video-disk, introducidos a finales de

los años 70, y en el éxito de los audio-compact-disk que le siguió [CHRI83], [CHRI84]

La sustitución de las antiguas cabezas de lectura basadas en lasers de gas, por los nuevos diodos laser de estado sólido, dio salida a los primeros CD-ROM (el tipo más sencillo de disco óptico), con capacidad de cientos de Mb.

La tecnología de almacenamiento óptico, ofrece una muy alta densidad de grabación, del orden de Mbits/cm^2, que por lo general, puede ser reproducido a bajo coste y alta velocidad.

La superficie del disco, se halla recubierta, de una capa protectora de plástico, que prolonga su esperanza de vida, elimina el requerimiento de ambientes superlimpios y lo hace más compacto y fácil de usar.

El relativamente grande espacio entre el cabezal de lectura y el soporte (del orden de milímetros), permite a la cabeza óptica, circunscribirse a puntos, eliminando el desgaste de la superficie, y haciendo imposible el aterrizaje de cabezas.

Finalmente, el disco óptico permite la integración de señales de video y audio en un medio directamente accesible desde el ordenador.

4.3.3. TIPOS DE DISCO OPTICO

El disco óptico, pude ser clasificado en tres categorías [MITK89]:

1.- SOLO LECTURA (Read-Only).

2.- ESCRITURA UNICA- LECTURAS MULTIPLES (Write-Once-Read-Many.

3.- MODIFICABLE (Ereseable/Rewriteable).

4.3.3.1. SOLO LECTURA

Es el tipo más sencillo dentro de la tecnología del disco óptico. El más conocido, es el CD-ROM, un compact disk con formato estándar. Toda la información se graba durante su elaboración, y el usuario no puede alterarla. Tiene una capacidad de hasta 550 Mb, y su esperanza media de vida es de 20 años. Es obvio que éste tipo de disco óptico, no es de nuestro interés, puesto que la información viene grabada de fábrica y es inalterable. Suele ser utilizado por editoriales para la distribución de series completas de publicaciones.

4.3.3.2. LECTURA UNICA- ESCRITURAS MULTIPLES

Este tipo de disco óptico, no tiene un formato estándar, y puede ser de diferentes tamaños. Como su nombre indica, la información solo puede ser grabada una vez por el usuario, y posteriormente ya no se puede borrar. La información se graba por medio de orificios en la superficie del disco. Esto hace que la grabación sea permanente. Tiene capacidad para almacenar más de 200 Mb en un disco simple, y su esperanza media de vida es por lo menos de 15 años. También es posible utilizar un "jukebox" (conjunto ó caja de discos), pudiéndose alcanzar capacidades de más de 1 Gb. Sus principales inconvenientes son la imposibilidad de reescritura, su precio y fragilidad.

Se suele utilizar para almacenar información voluminosa como Bases de Datos Documentales, ó aplicaciones que requieren un historial completo de actuación. Este tipo de disco óptico parece ser el más indicado para utilizar en un ambiente de recuperación como el que estamos estudiando.

4.3.3.3. DISCO MODIFICABLE

Como su nombre indica, este, es un disco modificable. Es el equivalente óptico al disco magnético estándar. Ofrece una capacidad de más de 200 Mb, y unos tiempos de respuesta muy parecidos a los de los discos convencionales. Su esperanza de vida esta entre los 10 y los 20 años. Este tipo de disco óptico es el más completo, pero ha comenzado a desarrollarse hace poco y su comercialización es muy reciente.

Las investigaciones para su desarrollo, se han centrado en tres diferentes alcances:

- cambio de material del soporte (de amorfo a cristalino)
- deformación del plástico
- combinación magnético-óptico

Este último avance es el más importante. De hecho, el disco modificable magnetico-óptico, apareció en 1988, pero todavía no se han establecido los estándares. Tiene el potencial de dominar el mercado del almacenamiento secundario.

Los desarrollos futuros, incluyen drivers más rápidos y cabezas ópticas más pequeñas, junto con técnicas más eficientes de codificación y corrección de errores.

CAPITULO 5

INDEXACION DEL DOCUMENTO

5.1. *OBJETOS A ALMACENAR*

Según se ha visto en el capítulo 4, las dos posibles entradas del documento al Sistema de Almacenamiento y Recuperación de la Información, dan resultados diferentes:

- la entrada vía procesador, da lugar a un fichero de textos ASCII.

- la entrada vía scanner, da lugar a una imagen, es decir, a un mapa de bits.

Decíamos también, que podíamos convertir el mapa de bits en un fichero de textos normal, mediante el sistema de OCR.

Sea cual sea la entrada empleada, una vez obtenido el fichero de textos ASCII, el Sistema de Almacenamiento y Recuperación de la Información, produce sobre él un proceso de indexación, dando como resultado un registro-índice, para su posterior identificación, localización y recuperación.

Por lo tanto, debemos tener en cuenta, tres diferentes objetos a almacenar:

- **Imagen del documento:** dado el tamaño que ocupa en memoria, se almacenará, en un disco óptico WORM.

- **Fichero de texto ASCII:** puede almacenarse en cualquier dispositivo magnético: disco duro, disquette..etc.

- **Registro-indice:** se almacenará en un dispositivo magnético cualquiera.

5.2. INDEXACION: DEFINICION DEL REGISTRO-INDICE

El registro-índice que definirá cada documento almacenado, suele tener un diseño estándar en cada Sistema de Almacenamiento y Recuperación de la Información. En esta tesis, describimos un posible formato de dicho registro-índice que se adecua al documento que hemos definido (ver sección 1.3) y al mecanismo de recuperación que describiremos a continuación. Esta formado por:

- un campo nombre-fichero: es el nombre del fichero de textos ASCII que contiene al documento. Es decir, es el puntero del registro-índice al fichero que contiene al documento.

- un campo de 15 referencias: son 15 campos que nos ayudan a localizar un determinado nombre-fichero ó un conjunto de nombres-fichero, que se ajustan ó responden a la necesidad de información expresada por el usuario.

Todos los registros-índice de los documentos que llegan al sistema y se indexan, quedan almacenados en un fichero que por ejemplo, llamaremos INDICE.DAT.

Estas 15 referencias están divididas en dos partes:

- **R0-R4:** son cinco referencias que guardan datos objetivos sobre el documento, por lo que las llamaremos **REFERENCIAS EXTERNAS.**

*R0: **Fecha de entrada***: Se toma del sistema el día de la introducción del documento.

*R1: **Código de identificación***: Se genera correlativamente.

*R2: **Tamaño***: Se calcula internamente, y expresa el tamaño del documento en memoria.

*R3: **Procedencia***: Ha de ser introducida por quién realice la entrada del documento, e indica el lugar (país, ciudad, empresa, etc.), de dónde procede el documento.

*R4: **Código de pertenencia***: Es un campo que solo puede tomar dos valores para indicar que, ó bien el documento se ha generado en el propio centro de investigación, ó bien es un documento recibido del exterior. Ha de ser también, rellenado por quién introduzca le documento.

- **R5-R14:** Son diez campos que contendrán diez descriptores con sus respectivos valores ó pesos. Es decir, serán los diez términos más relevantes del documento, los que mejor describan su contenido. Las llamaremos **REFERENCIAS INTERNAS.**

La tarea de indexación de un documento se divide por tanto, en dos subtareas principales:

1.- *Elección de los diez términos ó índices* que mejor describen los contenidos del documento.

2.- *Cálculo de los pesos de estos diez índices,* dado que todos ellos no describen al documento en la misma proporción ó medida.

Pero, podríamos añadir una tercera tarea que también es importante realizar para facilitar la posterior recuperación:

3.- *Normalización de los índices* que identifican al documento.

Una vez realizadas las dos primeras tareas de indexación, cada documento, queda representado por un vector que llamaremos V1 de diez dimensiones:

$$V1 = (t_1, w_1; t_2, w_2; \ldots\ldots; t_{10}, w_{10})$$

donde: t_i es el término ó índice i ($1 \leq i \leq 10$) que indexa al documento.

w_i es el peso del término t_i en el documento.

Por lo tanto, cada entrada correspondiente a un documento en el fichero de índices INDICE.DAT, está formada por tres campos:

1.- Campo de referencias externas dividido en cinco subcampos correspondientes a cada una de ellas.

2.- Campo de referencias internas, que tiene la forma del vector V1.

3.- Campo de nombre-fichero.

5.3. *TIPOS DE INDEXACION*

Esta tarea de elección de los diez términos que mejor describen al documento, se denomina *indexación*, y podemos clasificarla de tres maneras [SALT83]:

5.3.1. INDEXACION AUTOMATICA O MANUAL

Históricamente, la indexación se realizaba manualmente. Era labor de un experto en la materia de la cual trataban los documentos que se querían indexar. El elegía los términos más representativos del documento, es decir, los que mejor lo describían.

Pero hoy en día, en la gran mayoría de los sistemas, la indexación se realiza automáticamente, es decir, con el uso del ordenador.

Es obvio que nosotros nos centraremos en el estudio de una indexación automática.

5.3.2. INDEXACION DE TERMINOS CONTROLADOS O DE TERMINOS INCONTROLADOS

El uso de un vocabulario de indexación incontrolado, puede incluir toda la variedad del lenguaje natural, lo cual proporciona muchas posibilidades de ambigüedad ó error.

Sin embargo, un lenguaje de indexación controlado, garantiza una recuperación apropiada, pero para ello, se necesita algún intermediario como algún experto en la materia para indexar el documento y el query en los términos requeridos [DEWE88], [SALT83].

El vocabulario de indexación, siempre es menos controlado en la indexación automática que en la manual [SALT89].

5.3.3. INDEXACION POSTCOORDINADA O PRECOORDINADA

En la indexación postcoordinada, se utilizan en principio palabras individuales (términos-índice) para expresar los conceptos recogidos en el documento. Luego, estos términos-índice, pueden ser coordinados para formar descriptores más complejos cuando se formula el query.

En la indexación precoordinada, se utilizan como índices, frases que pueden incluir nombre y adjetivo etc..

EJEMPLO

En la indexación postcoordinada, los índices de un query podrían ser: I1: Información

I2: Recuperación

Y en la indexación precoordinada, el índice sería:

I1: Recuperación de la Información.

Como desventajas de la utilización de índices complejos, podríamos destacar la posible aparición de dos problemas:

1º Cuando se utilizan condiciones muy restringidas para construirlos (es decir, un criterio de frecuencia muy restringido y contextos muy limitados para reconocer las concurrencias entre los términos), entonces, se obtienen muy pocos índices y los resultados obtenidos difieren muy poco de los que se obtendrían con el uso de los índices sencillos.

2º Cuando el criterio de construcción de los índices complejos es más relajado, se pueden obtener buenos identificadores, pero a la vez también se obtienen muchos que no son útiles.

Pero la utilización de índices simples también conlleva dos desventajas:

1º Los índices simples fuera de contexto, pueden dar lugar a ambigüedades.

2º Los índices simples pueden resultar demasiado específicos ó generales para la indexación.

5.4. PRIMERA TAREA:

ELECCION DE LOS TERMINOS-INDICE

A continuación, se expone el *Algoritmo de Indexación Automática* propuesto por Salton [SALT83] para la elección de los índices ó términos que mejor describen los contenidos del documento, es decir, para la realización de la primera de las tareas en que se subdivide el proceso de indexación.

FASE 1: Identificación de las palabras individuales que constituyen el documento. Para ello, basta con identificar las cadenas de caracteres que ocurren entre:

- dos espacios
- un espacio y un signo de puntuación
- dos signos de puntuación

FASE 2: Cálculo de la frecuencia de cada palabra. Llamamos $FREQ_{ik}$, a la frecuencia de aparición de la palabra k en el documento i.

FASE 3: Cálculo de la frecuencia total de cada palabra. Llamamos $TOTFREQ_k$ a la frecuencia de la palabra k en toda la colección de documentos.

$$TOTFREQ_k = \sum_{i=1}^{n} FREQ_{ik} \qquad 5.1$$

donde n es el número de documentos de la colección.

FASE 4: Ordenación de las palabras en orden decreciente por su frecuencia total, $TOTFREQ_k$.

FASE 5: Elección de un umbral alto y eliminación de todas las palabras cuya frecuencia sea mayor a la del umbral. Existen en casi todos los idiomas unas 250 palabras que se incluyen en un diccionario denominado STOPLIST, que son las palabras que aparecen en todos los textos con muy alta frecuencia, y que no son nada significativas: preposiciones, artículos, etc..

FASE 6: Elección de un umbral bajo y eliminación de todas las palabras de menor frecuencia que dicho umbral. La desaparición de estos términos, al ocurrir tan infrecuentemente en la colección, no afecta a la función de recuperación del documento.

FASE 7: De entre el resto de las palabras de frecuencia media (nos hemos quedado con entre el 40% y el 50% de las palabras del texto), elegiremos como índices del documento, las diez de mayor frecuencia.

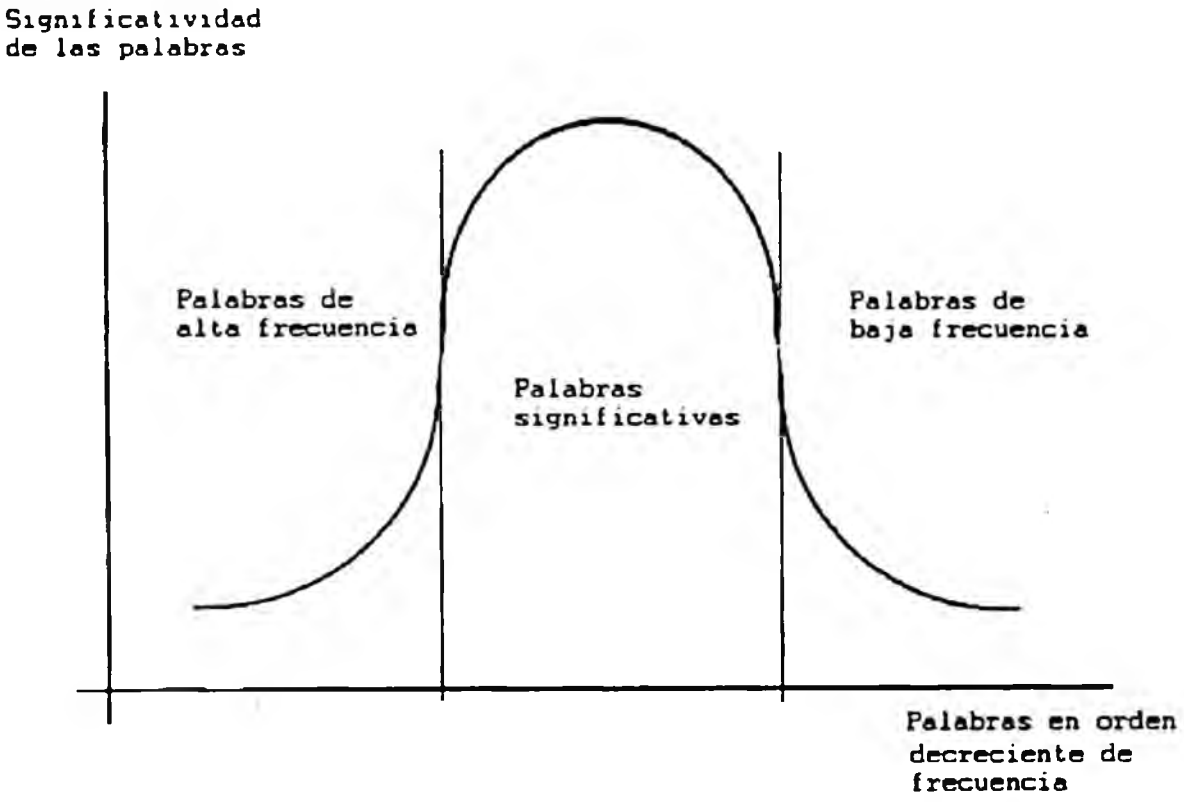


Figura 5.1

FASE 8: Utilización de un analizador morfológico [DEWE88], para la transformación de estas palabras a una forma canónica, mediante la identificación de:

- prefijos y sufijos
- singulares y plurales
- masculinos y femeninos

FASE 9: Identificación de las categorías semánticas en el Thesaurus de las palabras escogidas como índices, y elección del PT (PREFERRED TERM) ó término preferido. Además, en esta categoría semántica se incrementa en una unidad el valor de un contador que cuenta el número de documentos indexados por el término en cuestión.

Una vez aplicado el algoritmo a cada documento concreto, obtenemos sus diez referencias internas, es decir, queda finalizada la primera de las tres tareas a realizar en la indexación de un documento: la elección de los diez términos que mejor lo describen, y que serán utilizados como sus términos-índice.

5.5. SEGUNDA TAREA:

CALCULO DE LOS PESOS DE LOS INDICES

El principal objetivo de la función de obtención de los pesos de los índices, es proporcionar el máximo de efectividad en la recuperación [SALT88a], [LUCA88], es decir, el máximo acierto en la diferenciación entre los documentos relevantes y los no relevantes.

La efectividad, se mide principalmente por dos parámetros: ***Exhaustividad de Indexación*** y ***Especificidad de Término*** [SALT89]:

- *La exhaustividad de indexación*, refleja el grado en el cual se recogen todos los aspectos del tema del documento en el índice. Si la indexación es exhaustiva, se asignan muchos términos-índice.

- *La especificidad del término*, se refiere al grado de generalidad de los términos. Si los términos-índice asignados son muy generales, se recuperará mucha información útil, pero junto con otra que no lo es. Los términos generales, no pueden distinguir los documentos relevantes de los no relevantes; y los términos específicos recuperan bastante menos documentos, pero la mayoría de ellos son útiles al usuario.

Para juzgar el efecto de la exhaustividad de indexación y de la especificidad de los términos, nos valemos de dos factores importantes:

RECALL: Es la proporción de registros relevantes recuperados. Es decir, mediante este factor, se intenta que todos los registros relevantes (ó importantes para responder a la necesidad de información planteada por el usuario) sean recuperados. Por lo tanto, se mide por el ratio:

$$\text{RECALL} = \frac{\text{nº de reg. relevantes recuperados}}{\text{nº de reg. relevantes de la colección}}$$

PRECISION: Es la proporción de registros recuperados que son relevantes. Es decir, mediante este factor, se intenta que todos los registros no relevantes no sean recuperados. Se mide por el ratio:

$$\text{PRECISION} = \frac{\text{nº de reg. relevantes recuperados}}{\text{nº de reg. recuperados de la colección}}$$

El sistema ideal, será aquel en el que se concentren ambos factores: un alto RECALL, significando que todos los registros relevantes han de ser recuperados, y una alta PRECISION, significando que todos los registros que no son relevantes han de ser rechazados.

- Para alcanzar un alto nivel de RECALL, es necesario utilizar como índices, términos generales de alta frecuencia, es decir, que aparecen en muchos documentos de la colección, para de esta manera poder recuperar todos los documentos relevantes.

- Por el contrario, para conseguir una alta PRECISION, es necesario usar como índices, términos específicos, que sean capaces de aislar los registros relevantes del resto de la masa. Son términos que aparecen con alta frecuencia en pocos documentos y con nula en el resto. De ésta manera se consigue recuperar solo los registros relevantes y rechazar los no relevantes.

Como puede observarse, son tendencias contrarias, por lo que en la práctica, habrá que encontrar un compromiso entre ambas.

Estos requerimientos opuestos, hacen que el peso del índice esté formado por dos componentes derivados uno del factor de RECALL y el otro del factor de PRECISION.

1.- Los términos que son frecuentemente mencionados en documentos individuales son útiles para mejorar el nivel de RECALL, por lo que el primer componente para el cómputo del peso del índice será la *Frecuencia de Término* (FT) en el documento.

2.- Pero cuando los términos de alta frecuencia no están concentrados en pocos documentos, si no que prevalecen en toda la colección, todos los documentos tienden a ser recuperados y esto afecta al nivel de precisión. Luego, sería conveniente introducir en el cálculo del peso del índice, el componente ***Frecuencia Inversa del Documento*** (FID), que varía inversamente con el número de documentos n, a los que se asigna este término en la colección de N documentos. Para calcular este factor, nos dirigiremos al diccionario semántico ó thesaurus, donde en cada categoría semántica aparece el número total de documentos indexados por el documento en cuestión [LUCA88].

$$FID = \log (N / n) \qquad 5.2$$

Luego el peso de cualquier índice i para cualquier documento se obtiene como producto de ambos factores:

$$W_i = FT_i \times FID_i \qquad 5.3$$

es decir:

$$W_{ij} = FT_{ij} \times \log (N / n) \qquad 5.4$$

Una vez finalizada esta segunda tarea de indexación ya tenemos calculado el registro-índice completo que representará e identificará al documento dentro del Sistema de Almacenamiento y Recuperación de la Información.

En este momento, el registro-índice, puede ya almacenarse en el fichero de índices INDICE.DAT

5.6. *TERCERA TAREA:*

NORMALIZACION DE INDICES DE DOCUMENTO

Esta última fase del proceso de indexación, es importante por cuanto que los términos-índice que se han elegido (como resultado de las anteriores fases) para representar a los documentos, son diferentes y requieren una normalización.

Esto, significa que el vector V1 de 10 elementos que representa los 10 términos-índice ó referencias internas de cada documento, ha de ser transformado en otro vector que llamaremos V2, que representa ahora al documento por medio de M elementos, donde M es el número de términos que contiene el vocabulario del área de investigación del centro en estudio donde se quiere implantar el Sistema de Almacenamiento y Recuperación de la Información. Vamos a considarar un vocabulario de M = 50 palabras almacenado en un fichero que llamamos VOCAB.DAT.

Es decir, será para cada documento, la transformación de:

$$V1: (t_1,w_1; t_2,w_2;....., t_{10},w_{10})$$

donde: t_i es el término-índice número i elegido para indexar al documento. ($1 \leq i \leq 10$)

w_i es su peso ó valor en el documento en cuestión.

en:

$$V2: (t_1,w_1; t_2,w_2;....., t_M,w_M)$$

donde: t_i es el término-índice número i ($1 \leq i \leq M$) del vocabulario de referencia.

w_i es su peso ó valor en el documento en cuestión.

Como es obvio, cada documento, tendrá como máximo 10 elementos diferentes de 0 en V2 (aquellos que corresponden a los elementos del vector V1); y el resto, M-10 elementos serán iguales a 0.

Estos vectores V2 de representación de cada documento por medio de M elementos, según se van creando, se van almacenando en otro fichero de índices que para concretar, llamaremos DOC.DAT.

Cada vector V2, se guarda en el fichero junto a un campo que guardará el código de identificación (referencia externa R1) del documento respectivo.

En este momento, el sistema en lo que a sus datos se refiere, ya esta preparado para responder a una petición concreta del usuario.

CAPITULO 6

DICCIONARIO SEMANTICO

(THESAURUS)

6.1. DEFINICION

El thesaurus es un sistema estructurado de conceptos con todas las posibles relaciones jerárquicas y asociativas entre ellos, tal que para cada concepto, aparecen relacionados todos los términos que lo designan en las llamadas relaciones de sinonimia [SOER74].

El thesaurus es el diccionario semántico del sistema. Es decir, recoge todos los posibles conceptos del dominio de referencia, junto con sus relaciones de sinonimia y subordinación. Estas relaciones están almacenadas, según algunas teorías, como una red semántica en el cerebro humano, y el hombre hace uso de ellas continuamente aunque inconscientemente [SALT83].

Para determinar el apropiado nivel de exahustividad que debe poseer el thesaurus, se deben considerar los siguientes parámetros:

- El nivel de PRECISION y RECALL requeridos en los resultados de la búsqueda.

- El tiempo disponible para hacer la búsqueda.

- La frecuencia esperada de la petición de búsqueda.

- Tamaño de la colección.

- Técnicas de recuperación y dispositivos técnicos usados.

6.2. NECESIDAD DEL USO DEL THESAURUS

Dado que los conceptos dentro del Sistema de Almacenamiento y Recuperación de la Información solo pueden ser designados por un término (debido a la indexación controlada) para evitar errores y confusiones, debe de poder disponerse de alguna manera de todas las relaciones posibles entre conceptos y términos.

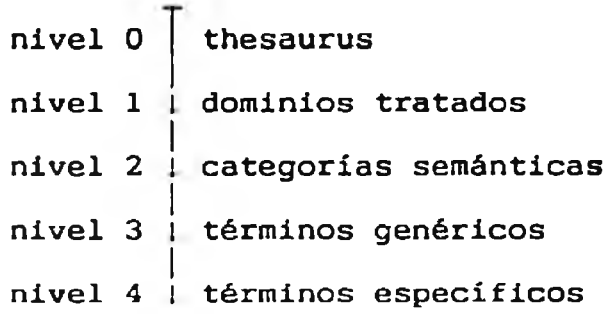
El thesaurus, será consultado principalmente en dos momentos diferentes:

1º Una vez indexado el documento, es decir, una vez extraídos los diez términos que mejor describen al documento, para cada uno de ellos, se consulta el thesaurus y se elige el término preferido (PT) de la categoría semántica en cuestión. Esto quedaba reflejado en la FASE 9 del algoritmo de indexación automática comentado.

2º De la misma manera, como veremos más adelante, también es necesario el uso del thesaurus, para la conversión de la petición de información expresada por el usuario en lenguaje natural, a una forma estándar entendible por nuestro sistema.

6.3. TIPOS DE TERMINOS Y ESTRUCTURA

El thesaurus tiene una estructura jerárquica que podemos representar de la siguiente manera [DEWE88]:



Nivel 0: THESAURUS: El thesaurus puede tratar diferentes dominios de vocabulario, aunque en nuestro caso, el dominio tratado será uno: el del área de investigación del centro a mecanizar.

Nivel 1: DOMINIOS: Dentro del dominio de referencia, las palabras se encuentran agrupadas en categorías semánticas ó de significado, recogiendo cada una de ellas todos los términos que designan ó están relacionados con un concepto.

Nivel 2: CATEGORIAS SEMANTICAS: Dentro de cada categoría semántica, podemos encontrar diferentes tipos de términos:

PT: TERMINO PREFERIDO (PREFERRED TERM): Es el término preferido, es decir, el término elegido de la categoría para representar el concepto en cuestión de una manera estándar, tanto en el documento como en el query del usuario.

BT: TERMINO GENERAL (BROADER TERM): Son los términos generales, es decir, los términos que dentro de la categoría, designan el mismo concepto, pero de una forma más general que el PT.

NT: TERMINO ESPECIFICO (NARROWER TERM): Son los términos específicos, es decir, los términos que dentro de la categoría, designan el mismo concepto, pero de una forma más especifica que el PT.

RT: **TERMINO RELACIONADO** (RELATED TERM): Son los términos relacionados, es decir, los términos que de alguna forma se relacionan con el concepto, y por lo tanto se engloban dentro de la misma categoría de significado.

SINONIMOS: Son los términos que expresan exactamente lo mismo que el PT.

Además, cada categoría semántica contiene un contador que almacena el número de documentos indexados por el término en cuestión, de tal manera que, cada vez

que un documento nuevo llega al sistema y se indexa, se incrementarán en una unidad todos los contadores de las categorías semánticas de los términos elegidos como índices de él.

CAPITULO 7

RECUPERACION

7.1. INTRODUCCION A LA RECUPERACION DOCUMENTAL

A finales de los años 50, fue Luhn [LUHN57] quien por primera vez sugirió que los sistemas de recuperación de textos, deberían ser diseñados basándose en la comparación entre los identificadores de contenido de los textos almacenados y de los queries ó peticiones de información del usuario [SALT88a].

Típicamente, ciertas palabras de los textos de los documentos y también de los queries, podrían ser usadas como identificadores de contenido.

Alternativamente, las representaciones de contenido, pueden también ser elegidas manualmente por expertos indexadores familiarizados con las aéreas consideradas y con los contenidos de la colección de documentos.

Sin embargo, hoy en día, estas tareas de indexación se realizan automáticamente con el uso del ordenador.

7.2. INTERFACE DE USUARIO

En los sistemas de recuperación actuales existen diferentes posibilidades en cuanto a la forma de diálogo entre el usuario y la máquina:

a.- Interface ó diálogo mediante menús ó pantallas.

b.- Interface orientado a comandos.

c.- Interface en Lenguaje Natural.

Vamos a centrarnos en el estudio de la tercera posibilidad: Interface en Lenguaje Natural. Y, ¿Por qué elegimos esta posibilidad?. Principalmente por dos razones [TAGU89]:

- La primera cuestión que nos debemos plantear es: "*¿Por qué el usuario se acerca al Sistema de Almacenamiento y Recuperación para buscar información?*".

El usuario habitualmente puede entrar en un estado anormal, denominado estado-ASK [BELK82]. Es decir, es un estado de duda, de desconocimiento que aparece porque se le ha planteado una necesidad de información y el desconoce absolutamente la respuesta que puede tener esta necesidad.

Entonces, él se acerca al sistema porque cree que en el sistema hay ó puede haber una serie de textos que le resolverán la duda y le ayudarán a salir del estado anormal de ASK en el que se encuentra.

Una vez que el usuario se ha acercado al sistema, puede ser ó puede no ser capaz de describir su necesidad de información en términos precisos ó estándar, comprensibles por el sistema de ordenador. Y en el peor de los casos, puede que hasta no sea capaz incluso de guiarse por menús ó por comandos.

Por lo tanto, vamos a considerar el caso más general, en el que el usuario desconoce totalmente la informática, por lo que realiza su petición de información en Lenguaje Natural.

- La segunda razón por la que vamos a realizar el interface en Lenguaje Natural, es para *aumentar la eficiencia del sistema*, ya que hace posible la formulación de cuestiones precisas que expresan exactamente las necesidades del usuario evitando interacciones.

En definitiva, nos parece mucho más interesante ahondar en una implementación del interface hombre-máquina en Lenguaje Natural.

Diremos que un *query en Lenguaje Natural* (NLQ), es "respondible" en nuestro sistema, si puede ser transformado en un formato estándar que denominaremos nuestro *Query Normal Form* (QNF) [SPIE88].

Se incluye un algoritmo que realiza ésta transformación, y posteriormente el Sistema de Almacenamiento y Recuperación de la Información podrá ya fácilmente interpretar la necesidad de información del usuario expresada en formato QNF.

7.2.1. INFORMACION DISPONIBLE

Antes de comenzar la exposición del modelo, necesitamos definir nuestro almacen de datos, es decir, la información de que disponemos, que se divide en dos partes:

1.- **BASE DE DATOS**: Esta formada por:

- *Documentos*: almacenados cada uno de ellos en un fichero de texto ASCII (ya se haya introducido vía procesador ó vía SCANNER-OCR), cuyo nombre aparecerá relacionado en el campo-puntero nombre-fichero del registro-indice.

- ***Registros-índice*** de estos documentos, almacenados en un fichero llamado INDICE.DAT, (que recordaremos, estaban divididos en tres partes: cinco referencias externas, diez referencias internas que junto con sus pesos ó valores se almacenaban en forma del vector V1, y un campo nombre-fichero refiriéndose al fichero ASCII que almacena al documento en cuestión).

- *Vectores V2* (de representación de los documentos por medio de M elementos) almacenados en el fichero DOC.DAT.

- *Vocabulario de términos-índice* que contiene todos los términos específicos del área de investigación, que se pueden elegir como índices. Es te vocabulario está almacenado en en fichero que llamaremos VOCAB.DAT.

2.- PARTE LINGÜISTICA DE LA BASE DE CONOCIMIENTOS:

Es la parte que contiene información sintáctica y semántica sobre el idioma castellano. Concretamente, en nuestro sistema, contamos con tres diccionarios:

- *Diccionario Sintáctico*: Especifica las categorías gramaticales ó sintácticas de las palabras. Es decir, este diccionario, contiene

todas las palabras del dominio de referencia del sistema, ya sean generales ó específicas, agrupadas por categorías sintácticas: sustantivos, verbos, artículos, adjetivos, etc..

- *Diccionario Semántico ó Thesaurus*: Especifica las categorías semánticas ó de significado de las palabras. Es decir, contiene todas las palabras del dominio de referencia del sistema, ya sean generales ó específicas, agrupadas por categorías semánticas.

- *Diccionario Propio*: Llamamos así a un diccionario que es exclusivo del Sistema de Almacenamiento y Recuperación de la Información que estamos exponiendo. Es un diccionario ideado y diseñado siguiendo la estructura del registro-índice que representa al documento. Está dividido en tres partes:

1.- RO (FECHA): Esta parte, contiene los nombres de los doce meses del año; los nombres de los siete días de la semana; una secuencia numérica del 01 al 31; y otra secuencia numérica anual comenzando en un año lo suficientemente antiguo, hasta el año actual.

2.- R1 (CODIGO): Esta parte contiene todos los códigos de identificación de todos los documentos. Es decir, cuando un documento nuevo llega al sistema y se le asigna un código secuencial, éste se almacena inmediatamente en éste diccionario.

3.- R3 (ORIGEN): Esta parte, es una lista de todos los lugares (países, ciudades,..etc) y de todos los organismos (empresas, centros de investigación, etc.), que estén relacionados con el centro de investigación en el cuál se se implementará el sistema en estudio.

Es importante hacer notar que la parte lingüística de la Base de Conocimientos debe ser, al menos parcialmente, dependiente de la Base de Datos, ya que al modificar la Base de Datos, habrán de ser igualmente modificados los tres diccionarios: sintáctico, semántico y propio. Y al introducir en la Base de Datos nuevos documentos que abarquen nuevas materias ó aéreas de conocimiento, habrán de ser creadas nuevas entradas en los diccionarios.

Para facilitar ó solventar parcialmente estos inconvenientes, los diccionarios sintáctico y semántico, podrían diseñarse divididos en dos partes:

- **Parte general:** conteniendo todas las palabras de nuestro dominio de referencia, pero que son de carácter general. Son las palabras que se usan habitualmente en cualquier sistema ó aplicación. Esta parte, es por tanto, independiente de la Base de Datos, pues no se ve afectada por los cambios en ella.

- **Parte específica:** conteniendo todas las palabras de nuestro dominio de referencia de carácter específico. Son las palabras que son características del área de conocimiento tratada. Esta parte, es por tanto, dependiente de la Base de Datos, pues será modificada al modificarse esta.

7.2.2. QUERY NORMAL FORM (QNF)

Vamos a definir ahora el QNF, ó lo que es lo mismo, el query estándar comprensible por el Sistema de Almacenamiento y Recuperación de la Información, resultante de la transformación llevada a cabo por el algoritmo que veremos a continuación, a partir de la petición de información introducida por el usuario.

Q: FEC, COD, TAM, ORI, IDE, $(q_1,w_1;\ q_2,w_2;\ldots\ldots q_{10},w_{10})$

R0	R1	R2	R3	R4	R5 ———R14
REFERENCIAS EXTERNAS					REFERENCIAS INTERNAS

donde: FEC: es la fecha de llegada del documento solicitado

COD: es el código del documento solicitado

TAM: es el tamaño del documento solicitado

ORI: es la indicación de la procedencia del documento solicitado.

IDE: es el identificativo de si el documento solicitado se originó en el propio centro ó no.

q_i $(1 \leq i \leq 10)$: es el término número i que describe al query ó a la petición de información.

w_i $(1 \leq i \leq 10)$: es el peso ó valor del término número i en el query.

7.2.3. ALGORITMO DE TRANSFORMACION

Podríamos utilizar el siguiente algoritmo para transformar el ***Natural Language Query*** (NLQ) en ***Query Normal Form*** (QNF), es decir, la petición de información introducida por el usuario, al estándar comprensible por el sistema. Este será desarrollado en trabajos posteriores.

FASE 1: ENTRADA Y PROCESO DEL QUERY

1.1.- **ENTRADA DEL QUERY:** El usuario introduce su solicitud de información en forma de sentencia del lenguaje natural.

1.2.- **IDENTIFICACION DE PALABRAS:** Identificación de todas las cadenas de caracteres existentes entre:

- dos espacios blancos
- un espacio blanco y un signo de puntuación
- dos signos de puntuación

1.3.- **CLASIFICACION-1:** Utilizando el Diccionario Propio, se identifican todas las palabras (si las hay) pertenecientes a las referencias R0, R1 y R3.

FASE 2: ANALISIS SINTACTICO

2.1.- **CLASIFICACION-2:** Utilizando el Diccionario Sintáctico, entre el resto de palabras, se identifican las categorías sintácticas: sustantivos, verbos, adjetivos, preposiciones, etc.

2.2.- **ELIMINACION-1:** Se desechan todas las palabras que no pertenezcan a la categoría del sustantivo, dado que no aportan nada a la semántica de la petición de información.

2.3.- **CLASIFICACION-3:** Utilizando el mismo diccionario, la lista de sustantivos, se divide ahora en dos sublistas: sustantivos generales y sustantivos específicos.

2.4.- **ELIMINACION-2:** Se elimina la lista de sustantivos generales, por no aportar nada al objetivo de la petición de información.

FASE 3: ANALISIS MORFOLOGICO

3.1.- **FORMA CANONICA:** Reducción de las palabras escogidas a su forma canónica. Para ello, el analizador morfológico, identificará:

- sufijos y prefijos
- plural y singular
- masculino y femenino

FASE 4: ANALISIS SEMANTICO

4.1.- **CLASIFICACION-4:** Utilizando el Diccionario Semántico, identificamos las categorías semánticas a las que pertenecen cada una de las palabras obtenidas del paso anterior.

4.2.- **ELECCION DEL PT:** Una vez identificadas dichas categorías, se elige de cada una de ellas el PT (Preferred Term).

FASE 5: TRANSFORMACION

Una vez que todas las palabras han sido identificadas y comprobadas en todos los diccionarios, esta lista se transforma en nuestro QNF, de la siguiente manera:

- las palabras que se han identificado en la CLASIFICACION-1, son asignadas a los campos correspondientes a las referencias externas R0, R1 y R3.

- las palabras que se han obtenido como resultado de la FASE 4, son asignadas a los campos correspondientes a los sucesivos términos q_i que componen las referencias internas del query.

FASE 6: VALIDACION

Antes de finalizar, conviene verificar si la estructura del query en QNF es correcta.

EJEMPLO DE TRANSFORMACION DE UN QUERY EN LENGUAJE NATURAL A SU FORMA ESTANDAR COMPRENSIBLE POR NUESTRO SISTEMA.

FASE 1: ENTRADA Y PROCESO DEL QUERY

1.1.- **ENTRADA DEL QUERY:** "Necesito toda la información que se recibió en Enero de 1990, sobre los avances en lenguajes de programación como Lisp ó Prolog, así como los avances en el área de demostración automática de teoremas en las lógicas borrosa y modal".

1.2.- **IDENTIFICACION DE PALABRAS:** necesito, toda, la, información, que, se, recibió, en, enero, de, 1990,.....

1.3.- CLASIFICACION-1:

- Palabras pertenecientes a R0: enero, 1990
- Palabras pertenecientes a R1: no hay
- Palabras pertenecientes a R3: no hay

FASE 2: ANALISIS SINTACTICO

2.1.- CLASIFICACION-2:

- sustantivos: información, avances, lenguajes, programación, Lisp, Prolog, área, demostración, automática, teoremas, lógicas, borrosa, modal.
- verbos: necesito, recibió.

- artículos: la, los, el, las.
- preposiciones: en, de, sobre.
- conjunciones: que, e, y, ó.
- adverbios: toda, así, como.

2.1.- **ELIMINACION-1**: Solo se conserva la lista: información, avances, lenguajes, programación, Lisp, Prolog, área, demostración, automática, teoremas, lógicas, borrosa, modal.

2.2.- **CLASIFICACION-3**:

- generales: información, avances, lenguajes, programación, área, demcstración, automática.
- especificos: Lisp, Prolog, teoremas, lógicas, borrosa, modal.

2.3.- **ELIMINACION-2**: Solo se conserva la lista: Lisp, Prolog, teoremas, lógicas, borrosa, modal.

FASE 3: ANALISIS MORFOLOGICO

3.1.- **FORMA CANONICA**: Lisp, Prolog, teorema, lógico, borroso, modal.

FASE 4: ANALISIS SEMANTICO

4.1.- CATEGORIAS SEMANTICAS:

- Categoría Semántica de "Lisp"

El término preferido (PT) de la categoría es "Lisp".

- Categoría Semántica de "Prolog"

El término preferido (PT) de la categoría es "Prolog".

- Categoría Semántica de "teorema"

El término preferido (PT) es "teorema".

- Categoría Semántica de "lógico"

El término preferido (PT) es "lógico".

- Categoría Semántica de "borroso"

El término preferido es "difuso".

- Categoría Semántica de "modal".

4.2.- ELECCION DEL PT:

Lisp, Prolog, teorema, lógico, difuso, modal.

FASE 5: TRANSFORMACION

Enero,1990	R1	R2	R3	R4	VQ1: R5-R14

VQ1: Lisp,w_1; Prolog,w_2; teorema,w_3; lógico,w_4;
difuso,w_5; modal,w_6

7.3. CALCULO DE LOS PESOS DE LOS TERMINOS EN EL QUERY

Para el cálculo de los pesos de los términos en el query, debemos tener en cuenta idénticas consideraciones a las ya explicadas para el cálculo de los pesos de los términos en el documento [SALT88a] [LUCA88].

Es decir, aquí también, la función del peso vendrá dada como producto de dos factores:

$$W_i = FT_i \times FID_i \qquad 7.1$$

donde: FT_i: es la frecuencia del término q_i en el query. Es decir, es el número de apariciones del término en el texto del query. Es la ***Frecuencia de Término***.

FID_i: Es idéntico al ya explicado. Es la ***Frecuencia***

***Inversa del Documento*, (log (N/n)).**

Para cada término del query, en el cálculo de su factor FID, nos dirigiremos al diccionario semántico ó thesaurus, donde, en cada categoría semántica aparece el número total de documentos indexados por el concepto en cuestión [LUCA88].

La mayoría de los textos del query son relativamente cortos y además, generalmente, a los queries se les asigna menos términos que a los documentos, por lo tanto, la frecuencia de aparición de los términos del query rara vez excede de 1. Por lo tanto, el componente ***Frecuencia de Término*** (FT), puede ser eliminado en el cálculo del peso del término en el query [SALT89].

7.4. NORMALIZACION DE LOS INDICES DEL QUERY

Como resultado de las tareas anteriores: transformación del query, cálculo de pesos, etc., el query expresado en QNF, queda representado por medio de un vector de 10 elementos ó referencias internas, que designamos por VQ1.

Pero estos 10 términos elegidos para representar a cada query determinado, son todos ellos diferentes. Luego, idénticamente que en el caso de los documentos, los índices han de ser normalizados, para poder ser enfrentados con los índices de los documentos.

Esta normalización se lleva a cabo mediante la transformación del vector:

$$VQ1: (q_1, w_1; q_2, w_2; \ldots\ldots; q_{10}, w_{10})$$

donde: q_i es el término número i ($1 \leq i \leq 10$) elegido para indexar al query.

w_i es el peso ó valor del término i ($1 \leq i \leq 10$) en el query.

en el vector:

$$VQ2: (q_1, w_1; q_2, w_2; \ldots\ldots; q_M, w_M)$$

donde: q_i es el término i ($1 \leq i \leq M$) del vocabulario de referencia de M términos, almacenado en el fichero VOCAB.DAT.

w_i es el peso ó valor del término i ($1 \leq i \leq M$) del vocabulario, en el query.

Es obvio que, cada query, en el vector VQ2 sólo tendrá 10 elementos diferentes de 0 (aquellos que coincidan con los del vector VQ1), y el resto, M-10, serán iguales a 0.

Siguiendo con el ejemplo, el vector VQ1 se transformaría en un vector VQ2, que tendría, según los términos del vocabulario expuesto en la sección 10.2, todos los elementos iguales a 0 excepto: t_{16}= difuso, t_{26}= lisp, t_{29}= lógico, t_{31}= modal, t_{38}= prolog, t_{46}= teorema.

VQ2: $w_1=0$; ...; w_{16}; $w_{17}=0$; ...; w_{26}; $w_{27}=0$; $w_{28}=0$; w_{29}; $w_{30}=0$; w_{31}; $w_{32}=0$; ...; w_{38}; $w_{39}=0$; ...; w_{46}; $w_{47}=0$; ...; $w_{50}=0$.

7.5. FUNCION DE RECUPERACION

Una vez finalizadas las explicaciones de las sucesivas transformaciones que sufren las dos entradas de un Sistema de Almacenamiento y Recuperación de la Información (documento y petición de información por parte del usuario); y una vez que, ambas han quedado preparadas para ser enfrentadas y comparadas, exponemos ahora algunos de los diferentes modelos existentes de enfrentamiento. Es decir, los llamados ***Modelos ó Funciones de Recuperación***, que son los encargados de proporcionar una respuesta adecuada al usuario en base a la comparación query-documento.

7.5.1. MODELOS BOOLEANOS

En la teoría booleana, cada documento se representa mediante un conjunto de términos-índice que describen su contenido; y cada query, se representa mediante combinaciones booleanas de las variables correspondientes a estos términos-índice.

7.5.1.1. MODELO BOOLEANO SIN PESOS

En el modelo booleano sin pesos, sólo los documentos cuyas representaciones hagan verdad el valor de la expresión del query booleano, son procesados con valor 1, es decir, adquieren el valor booleano de 1 y por tanto, son incluidos en la salida ó conjunto recuperado de respuesta al query.

De acuerdo con esto, todos los documentos recuperados, aparecen con el mismo nivel de utilidad para satisfacer la necesidad de información del usuario, y además, dispuestos en cualquier orden.

La calidad de la salida proporcionada, puede ser satisfactoria en el caso en que el usuario sea muy experimentado, y sea capaz de presentar al ordenador su necesidad de información lo suficientemente bien definida por medio de la apropiada representación del query.

Pero incluso en esos casos, el sistema falla en la identificación de los documentos relevantes: generalmente, la salida no incluye todos los documentos relevantes, recuperándose por el contrario, muchos que no lo son.

REPRESENTACION DEL MODELO

Comenzamos a construir el sistema, definiendo las siguientes variables [WALL79]:

N: número de documentos almacenados en la Base de Datos

M: número de términos en el vocabulario

d_j: variable bocleana que puede tomar dos valores:

$d_j = 1$ —> el término j ha sido asignado, es decir, se utiliza como índice en el documento en cuestión.
$d_j = 0$ —> el término j no ha sido asignado a ese documento.

q_j: variable booleana que puede tomar dos valores:

$q_j = 1$ —> el término j ha sido asignado al query en estudio.
$q_j = 0$ —> el término j no ha sido asignado a ese query.

Por consiguiente, en el modelo booleano, un término concreto j, ó es asignado a un documento dado ($d_j = 1$), ó no lo es ($d_j = 0$), y de la misma manera, ó es asignado a un query dado ($q_j = 1$) ó no lo es ($q_j = 0$).

ESTRUCTURA DEL QUERY

Un query, sólo puede estar formado por expresiones booleanas bien construidas, que vienen dadas por las siguientes reglas:

```
EXP-BOOLEANA::= | <LIT> | (NOT<EXP-BOOLEANA>) |
                | (<EXP-BOOLEANA> AND <EXP-BOOLEANA>) |
                | (<EXP-BOOLEANA> OR <EXP-BOOLEANA>)
```

donde, el literal, puede ser un átomo ó un átomo negado:

```
LIT ::= | <ATOM> | (NOT<ATOM>)
```

y donde, el átomo puede ser cualquier término del vocabulario, añadiendo otros dos posibles términos: verdadero ó falso.

```
ATOM ::= | <TERM> | <TRUE> | <FALSE>
```

y donde TRUE es el término asignado a todos los documentos y FALSE es el término asignado a ningún documento en la Base de Datos. Por lo que, el query compuesto solo por el término TRUE, recuperaría todos los documentos; y por el contrario, el query compuesto solo por el término FALSE, no recuperaría ninguno.

Todo query bien construido, puede ser fácilmente manipulado para transformarse en dos diferentes representaciones canónicas, para poder calcular el valor del documento con respecto al query [SALT89]:

CNF: FORMA NORMAL CONJUNTIVA (Producto de sumas): Es un conjunto de cláusulas conectadas mediante la conectiva conjunción (AND).

$$\bigwedge_{i=1}^{p} \bigvee_{j=1}^{c_i} (L_{ij}) \qquad 7.2$$

DNF: FORMA NORMAL DISYUNTIVA (Suma de productos): Es un conjunto de frases conectadas mediante la conectiva disyunción (OR).

$$\bigvee_{i=1}^{t} \bigwedge_{k=1}^{K_i} (L_{ik}) \qquad 7.3$$

donde la cláusula es un conjunto de literales combinados por el operador lógico OR.

CLAUSULA::= | <LIT> | (<CLAUSULA>OR<LIT>)

y donde la frase es un conjunto de literales combinados por el operador lógico AND.

FRASE::= | <LIT> | (<FRASE>AND<LIT>)

7.5.1.2. MODELO BOOLEANO CON PESOS DIFUSOS

Como veíamos en el punto anterior, en el modelo booleano sin pesos, un término específico t_j, ó es asignado a un documento ó no lo es; y de la misma manera, es asignado a un query dado ó no lo es. Esta es una concepción muy estricta e inflexible.

Existe una alternativa, de permitir al indexador, ya sea manual ó automático, mayor flexibilidad en la asignación de los términos, por medio de los llamados *pesos*. Este método se basa en *las teorías difusas* [WALL79].

El *peso del término en el documento*, indica la importancia del término como buen descriptor del contenido del documento. Y de la misma manera, el *peso del término en el query*, nos indica la importancia del término como buen descriptor de los contenidos del query.

El uso de estos pesos difusos, proporciona dos importantes ventajas con respecto al modelo booleano sin pesos:

1.- Los pesos pueden ser usados para evaluar cada documento con respecto a su nivel de información en respuesta a un query dado. Para esto, se necesitan dos funciones:

a.- Función para generar la evaluación de cada documento con respecto a cada término en el query.

b.- Función para generar la evaluación total de cada documento con respecto al query (teniendo en cuenta todos los términos). Esta función se llama *Valor de Estado de Recuperación* (RSV).

Se necesitará un umbral, y solo se recuperarán aquellos documentos cuyo RSV excedan al valor del umbral.

2.- También se pueden utilizar los pesos para ordenar la salida recuperada. Es decir, la lista de documentos recuperados (cuyo RSV es superior al umbral), posteriormente se ordenará de mayor a menor RSV. De esta forma, el usuario, los obtiene ordenados por su importancia con respecto a su solicitud de información.

REPRESENTACION DEL MODELO.

Definimos las siguientes variables y conceptos:

N: número de documentos almacenados en la Base de Datos.

M. número de términos en el vocabulario.

Consideremos un subconjunto de documentos en la Base de Datos, que contiene a todos los documentos que tratan "sobre" el concepto representado por el término t_j. Esto da otro significado a la variable d_j.

d_j: es una variable que indica el grado de pertenencia de un documento dado al subconjunto borroso de los documentos que tratan "sobre" el término t_j.

$d_j = 1$ —> máxima importancia del término t_j en el documento en cuestión.

$d_j = 0$ —> mínima importancia del término t_j en el documento en cuestión.

q_j: es una variable que indica al grado de pertenencia ó la importancia de un query dado, en el conjunto borroso de todos los queries que pueden ser descritos usando el término t_j.

$q_j = 1$ —> máxima importancia del término t_j en el query en estudio.

$q_j = 0$ —> mínima importancia del término t_j en el query en estudio.

En este modelo difuso, ambas variables son continuas en el intervalo de valores [0,1]

ESTRUCTURA DEL QUERY

Es exactamente igual que para el modelo booleano sin pesos.

PROBLEMAS E INCONVENIENTES

Veamos los inconvenientes que supone la utilización de cualquiera de los dos modelos booleanos vistos, para posteriormente pasar a la explicación de otros modelos [SALT88b].

1.- En el modelo booleano sin pesos, no existe ninguna ordenación en la salida que nos de la importancia de los documentos recuperados con respecto al query y además los términos no tienen asignado ningún valor ó peso de acuerdo con su importancia relativa al query ó documento.

2.- En ninguno de los dos modelos hay forma de controlar el tamaño de la salida. Es decir, si el término índice es muy general, puede ocurrir que se recuperen infinidad de documentos, sin tener el usuario la posibilidad de obtener solo los más relevantes.

3.- En respuesta a un query tipo OR, los documentos que contienen todos los términos no son mejor considerados que aquellos que contienen solo alguno ó algunos de ellos.

4.- Como respuesta a un query tipo AND, los documentos que contienen un término son tan mal considerados como aquellos que no contienen ningún término.

Una vez expuestos ambos modelos, sus estructuras y sus inconvenientes, exponemos ahora, otro modelo resultante de otras investigaciones y que parece dar solución a estos inconvenientes: *modelos booleanos extendidos* (Extended Boolean Models).

7.5.2. MODELO VECTORIAL

REPRESENTACION DEL MODELO

Este modelo, fue propuesto y extensamente investigado por Salton [SALT83] y posteriormente por Wong y Ragharen [WONG84]. Es el modelo más fácil de usar y muchas veces es también el más productivo [SALT89].

Cada documento se representa por un vector de M elementos, donde cada elemento es un término del vocabulario ó dominio de referencia.

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{ij}, \ldots, d_{im})$$

donde, d_{ij} es el peso ó valor que refleja la importancia del término j en el documento i.

Las mismas consideraciones se aplican a la representación del query:

$$Q_i = (q_{i1}, q_{i2}, \ldots, q_{ij}, \ldots, q_{im})$$

donde, q_{ij} es el peso ó valor que indica la importancia del término j en el query i.

Cada uno de los M términos del vocabulario de referencia, se representa mediante un vector T de términos, y el *espacio vectorial* se define siempre que los M vectores sean *linealmente independientes* [SALT89].

Como consecuencia, tanto el documento como el query, se representan como combinación lineal de los M vectores de términos:

$$D_r = \sum_{i=1}^{m} d_{ri} \times T_i \qquad Q_s = \sum_{j=1}^{m} q_{sj} \times T_j \qquad 7.4$$

Una vez representados query y documento como vectores, podemos calcular para cada pareja query-documento, una medida llamada *Valor de Estado de Recuperación* (RSV), que mide la

similaridad entre ambos. Es un indicador del grado de similaridad entre una petición concreta de información por parte del usuario del sistema y un documento almacenado en la Base de Datos [RAGH89].

La decisión de recuperar un documento depende de su similaridad con el query. Existen gran cantidad de ***funciones de similaridad***, y la elección de una de ellas, depende de la estructura del sistema, representación de los datos, etc..

Por lo tanto, para un query dado Q, la función S de similaridad, produce un número real $S(Q,D_i)$ para cada documento i de la colección, donde el número mayor representa el mayor grado de cercanía ó semejanza.

Algunas medidas de similaridad son:

- Fórmula del coseno del ángulo formado por ambos vectores.

- Fórmula de la distancia entre los dos puntos en el espacio de M dimensiones.

- $S\ (D,Q) = \sum_{i,j=1}^{m} d_i \times q_j$

- ....etc.

Concretamente, en el ***Modelo Vectorial***, la similaridad entre el query y el documento viene dada por el producto:

$$X\ Y = |X||Y| \cos \alpha \qquad 7.5$$

donde | X | es la longitud de X

α es el ángulo formado por los vectores X e Y

Por lo tanto, la similaridad, nos vendría expresada por la siguiente fórmula:

$$D_i \times Q_s = \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} \times q_{sj} \times T_i \times T_j \qquad 7.6$$

Generalmente, los términos están incorrelacionados por lo que sus vectores son ortogonales ($T_i \times T_j = 0$, excepto cuando j = i, en cuyo caso, nos queda: $T_i \times T_j = 1$), quedándonos, entonces, la similaridad:

$$D_i \times Q_s = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \times q_{sj} \qquad 7.7$$

VENTAJAS DEL MODELO

Este modelo, ofrece las siguientes ventajas para implantaciones prácticas [SALT88b]:

1.- Permite la incorporación de pesos tanto en el query como en el documento, mostrando así, las distintas importancias de los términos en ambos.

2.- Proporciona la salida recuperada ordenada en decreciente por su similaridad con el query en estudio.

3.- El tamaño del conjunto recuperado, se puede adaptar a los requerimientos del usuario, recuperando sólo los primeros registros de la lista ordenada en decreciente por similaridad con el query.

4.- Facilita la adaptación del mecanismo sistema-usuario mediante el proceso de generación y redefinición de los queries.

7.6. ENFRENTAMIENTO QUERY-DOCUMENTO MEDIANTE LA FUNCION DE RECUPERACION

Vamos a estudiar los diferentes casos que pueden darse en el Sistema de Almacenamiento y Recuperación de la Información. Como veremos, el enfrentamiento, será diferente en cada uno de ellos, necesitándose un distinto número de accesos a también diferentes ficheros.

1.- Existen todas las referencias externas e internas: Esto indica que el usuario busca un documento en concreto del cual conoce todos sus datos. La recuperación, entonces, se hará directamente sobre el fichero INDICE.DAT, por la referencia externa R1 que es el código del documento. Una vez localizada la entrada correspondiente en este fichero, encontraremos en ella un campo nombre-fichero que nos remite al fichero de textos que contiene al documento buscado.

2.- Existen todas las referencias externas y ninguna interna: Este caso es exactamente igual al anterior por lo que se procede de la misma manera.

3.- Existen sólo las referencias internas: Este es el caso más común, dado que, por lo general, el usuario se acerca al sistema sin conocer el ó los documentos que

busca en concreto. El realizará la solicitud de información en base a los temas que necesita, pero sin especificar datos concretos de ningún documento. En este caso, la tarea de recuperación se realiza en dos pasos:

a.- Búsqueda en el fichero DOC.DAT, de la entrada ó entradas correspondientes a los documentos cuyos vectores V2 sean más similares al vector VQ2 del query en estudio.

b.- Cada entrada seleccionada en el fichero DOC.DAT, se relacionará con su correspondiente en el fichero INDICE.DAT (mediante el código R1). Como ya sabemos, ahora, el campo nombre-fichero nos remite al fichero de texto que almacena al documento buscado.

4.- Existen las referencias internas y parte de las referencias externas: El usuario, al acercarse al sistema, puede tener una ligera idea del documento que necesita, pero que no recuerde todos sus datos.

a.- Si la referencia externa R1 (código) es conocida, la recuperación se llevará a cabo de forma similar al caso 1.

b.- Pero, si por el contrario, esta referencia es desconocida, se procederá similarmente al caso 3.

CAPITULO 8

SALIDA DE LA INFORMACION

8.1 INTRODUCCION

En cuanto a la salida de la información u obtención del documento por parte del usuario, debemos distinguir dos etapas claramente diferenciadas:

1.- **SALIDA LOGICA**: Se refiere a la obtención en pantalla del conjunto de registros-índice recuperados como respuesta a un query dado. En este momento, el usuario, conoce la descripción (mediante las referencias externas e internas) y el nombre del fichero que contiene al documento (mediante el campo nombre-fichero), de todos los documentos recuperados como respuesta al query dado.

2.- **SALIDA FISICA**: Una vez que el usuario está satisfecho con los registros-índice que el sistema le ofrece, como respuesta al query formulado para satisfacer su necesidad de información, se produce la salida física, es decir, la obtención por impresora de cada uno de los ficheros que contienen a los documentos seleccionados.

8.2. SALIDA LOGICA

Como ya se ha mencionado previamente, llamamos salida lógica, a la lista de registros-indice que se obtienen por pantalla como respuesta a un query del usuario.

La evaluación de la salida lógica, juega un importante papel en la medición de la eficiencia y la efectividad de un modelo ó sistema de recuperación [RAGH89].

Existen métodos de evaluación investigados en el pasado. Un buen método de evaluación debe tener en consideración aspectos del proceso de recuperación [BOLL84], como son:

- Recursos usados por el sistema.

- Cantidad de tiempo y esfuerzo gastado por el usuario hasta obtener la información.

- Habilidad del sistema para recuperar los registros-índice útiles.

Es difícil obtener todos los parámetros de medición de estos aspectos. Incluso si disponemos de la información necesaria, también es difícil su combinación. Por consiguiente, es una práctica común, concentrar todos estos aspectos en la medida de la calidad de la salida lógica.

Existen aspectos que afectan a los resultados finales, es decir, a la salida lógica:

- el proceso de indexación

- la representación interna y el almacenamiento de la colección

- la estrategia de búsqueda

- el modelo de recuperación

En cuanto a los criterios de evaluación de la salida, podríamos enumerar: *RECALL, PRECISION, ESFUERZO, TIEMPO, FORMA DE REPRESENTACION.*

Entre todos ellos, los factores de RECALL y PRECISION han obtenido la máxima atención, y pueden ser considerados los más importantes.

Un Sistema de Almacenamiento y Recuperación de Información esta diseñado y construido como respuesta a una necesidad de almacenar y recuperar grandes masas de información. En la mayoría de los casos, cada vez que una petición particular de información se presenta en el sistema, los documentos de la colección pueden ser conceptualmente divididos en dos categorías:

- **documentos relevantes:** son aquellos documentos que serían juzgados por el usuario como interesantes para resolver su necesidad de información.

- **documentos no relevantes:** son aquellos documentos que serían juzgados por el usuario como no interesantes para su necesidad de información.

Desde el punto de vista del sistema, la efectividad del mismo puede ser medida como *la precisión y exactitud en identificar esta dicotomía.*

Desde el punto de vista del usuario, la efectividad del sistema queda reflejada en la siguiente frase:

" *Recuperar tantos registros-índice relevantes como sea posible y tan pocos registros-índice no relevantes como sea posible, en respuesta a un query dado*"

El primer criterio es lo que se conoce por RECALL, y el segundo por PRECISION.

Como puede observarse, ambas medidas de efectividad (para el sistema y para el usuario), son las mismas.

Tanto el factor de RECALL como el de PRECISION, se miden una vez que el sistema ha proporcionado una ordenación de los registros-índice en la salida lógica. Es decir, la ordenación

de los grados en que cada documento responde a la necesidad de información del usuario. Esto es lo que se llama ***Ordenación de la Salida*** (Ranking Output).

Por lo tanto, podemos decir que en la evaluación de la salida lógica, son importantes tres factores: ***ORDENACION DE LA SALIDA***, ***RECALL*** y ***PRECISION***.

8.2.1. ORDENACION DE LA SALIDA

Para conseguir el objetivo de que el sistema pueda localizar los registros-índice relevantes en respuesta a un query dado, se calcula un valor para cada par (V2,VQ2), formado por un documento y un query. Este valor se calcula mediante alguna de las funciones de recuperación anteriormente expuestas y se llama ***Valor de Estado de Recuperación*** (RSV).

Este valor RSV se utiliza para ordenar el conjunto de registros-índice recuperados, en orden descendente, por lo que en la salida lógica, los registros-índice de los documentos recuperados, se presentan al usuario ordenados de mayor a menor interés ó importancia con respecto a su necesidad de información.

Podemos distinguir dos tipos diferentes de ordenación según el RSV: ordenación lineal y ordenación débil.

ORDENACION LINEAL: En este caso, a cada registro-índice de la colección, se le asigna un RSV distinto por medio de la función de similaridad. Este tipo de ordenación, no plantea ningún problema.

ORDENACION DEBIL: Este caso se da cuando a más de un registro-índice se le asocia el mismo RSV. Podríamos decir entonces, que el sistema considera que dos ó más documentos responden idénticamente igual a la petición del usuario. En este caso, se debe introducir alguna noción probabilística para solventar el problema que surge al tener más de un documento asignado el mismo RSV.

8.2.2. PRECISION Y RECALL

Recordamos que el *factor de RECALL* se define como el ratio entre el número de documentos relevantes que son recuperados y el número total de documentos relevantes.

Y el *factor de PRECISION* se define como el ratio entre el número de documentos relevantes recuperados y el número total de documentos recuperados.

Ambos factores tienen intereses contrarios, en el sentido de que para aumentar el nivel de RECALL es preciso recuperar muchos registros (para recuperar todos los

relevantes), lo que también supone recuperar registros no relevantes, disminuyéndose así el nivel de PRECISION. Por lo tanto, hay que buscar un buen compromiso entre ambos factores.

En la exposición del sistema, ya se tuvieron en cuenta estos factores: recordamos que en la tarea de indexación, en el cálculo de los pesos de los términos, tanto en el documento como en el query, se empleaban los factores *Frecuencia de Término* y *Frecuencia Inversa de Documento*, que ayudaban a preservar los niveles aceptables de RECALL y PRECISION en la salida lógica obtenida.

Vemos ahora, como se calcula el nivel de RECALL y de PRECISION en una salida lógica obtenida como respuesta a un query dado.

Dada una ordenación de documentos, se debe especificar algún tipo de *Criterio de Parada* para el cálculo del RECALL y de la PRECISION.

En nuestro caso, el usuario es quien introduce el Criterio de Parada, siendo éste, *el número de documentos relevantes que desea recuperar*.

Sea r el número total de documentos relevantes que existen como respuesta a un query dado; y sea NR ($1 \leq NR \leq r$) el Criterio de Parada.

Existen r posibles niveles de Recall:

$$\text{RECALL} = \frac{1}{r}, \frac{2}{r}, \ldots, \frac{NR}{r}, \ldots, \frac{(r-1)}{r}, 1$$

mínimo RACALL — máximo RECALL

- mínimo RECALL (1/r): solo un documento relevante es recuperado.

- máximo RECALL (r/r): todos los documentos relevantes son recuperados.

Para cada nivel de RECALL, el nivel de PRECISION se calcula fácilmente como:

$$\text{PRECISION} = \frac{NR}{NR + NNR}$$

donde NNR es el número de documentos no relevantes que son recuperados entre los NR documentos deseados.

Los distintos niveles de precisión en este caso son:

$$\text{PREC.} = \frac{1}{1+NNR}, \frac{2}{2+NNR}, \ldots, \frac{NR}{NR+NNR}, \frac{(r-1)}{(r-1)+NNR}, \frac{r}{r+NNR}$$

8.3. SALIDA FISICA

Una vez que en pantalla disponemos ya de la lista definitiva de registros-índice recuperados en respuesta a una necesidad concreta de información, es decir, una vez que la salida lógica obtenida satisface al usuario, solo nos queda la obtención del documento físico en papel.

Para ello basta con dar la orden de impresión de los ficheros cuyos campos nombre-fichero aparecen en los registros-índice recuperados en la salida lógica.

CAPITULO 9

ALGORITMO DE CLUSTERING

9.1. NECESIDAD DEL CLUSTERING

El *Análisis Cluster* es una técnica que permite la *identificación de grupos, clases* ó *clusters*, es decir, de objetos similares en un espacio multidimensional [EL-H89].

La práctica de clasificar objetos de acuerdo con sus similaridades, es la base de muchas ciencias. La organización de los datos en grupos, es una de las formas más elementales de aprender y entender. El análisis cluster inicialmente, se desarrolló para aplicarse en las ciencias de la vida, pero posteriormente ha tenido diversas aplicaciones en muchos campos de la ciencia [SALT83], [CAN89].

El Análisis Cluster, es el estudio formal de los algoritmos y métodos para clasificar objetos. Estas clasificaciones, se realizan en base a la descripción de los objetos. Un objeto, se puede describir, ó por una serie de características, ó por sus relaciones con los demás objetos.

El objetivo del Análisis Cluster es simplemente, encontrar una organización conveniente y válida de los datos [JAIN88].

B.S. Everitt, nos da una buena definición del concepto de cluster [EVER74]: "***Un cluster es un conjunto de entidades que son parecidas ó similares, siendo las entidades de los clusters diferentes, no similares***".

Por lo tanto, las dos cuestiones cruciales consisten en, especificar "***qué similaridad hay entre los objetos***", y "***cómo medirla***".

Es importante destacar las ventajas que ofrecen las técnicas de clustering, sobre los procesos de clasificación manuales:

1º Los seres humanos somos capaces de obtener buenas clasificaciones cuando los objetos están descritos por medio de una, dos ó a lo sumo tres características. Pero cuando la lista de características asociadas a cada objeto es larga, un algoritmo de clustering, puede obtener las clases en un tiempo infinitamente inferior al utilizado por el ser humano.

2º Individuos diferentes, no siempre obtendrían la misma clasificación, puesto que la medida de similaridad dependería mucho de factores educacionales y culturales.

En el aérea de ***Recuperación de la Información***, el análisis cluster, se usa para clasificar, en el sentido de generar *clases ó clusters de documentos* [SALT89].

La clasificación generará clusters, conteniendo cada uno de ellos el conjunto de documentos juzgados como más similares.

Cuando las Bases de Datos en la que se hallan almacenados los documentos son muy grandes, como es el caso que estudiamos, el clustering, se hace totalmente necesario dado que nos proporciona las siguientes ventajas:

1.- Optimización del almacenamiento.

2.- Reducción del tiempo de respuesta.

3.- Reducción del número de comparaciones ó búsquedas que se han de efectuar para responder a una petición concreta del usuario.

Dado que los documentos similares han de ser todos relevantes para un determinado query, éste será primeramente comparado con el prototipo representante de cada clase ó cluster.

Una vez seleccionados los clusters relevantes para responder a una determinada demanda de información, sus documentos serán entonces, comparados con el query representativo de esta demanda, para una selección final y ordenación con respecto a su relevancia.

9.2. ANALISIS CLUSTER. TIPOS Y METODOS

Los algoritmos cluster son principalmente de dos tipos:

- Algoritmos jerárquicos.
- Algoritmos heurísticos.

Cada uno de los dos tipos, tiene sus apropiadas áreas de aplicación: los métodos jerárquicos se utilizan principalmente en ciencias biológicas, sociales y del comportamiento; mientras que los métodos heurísticos se usan en aplicaciones de ingeniería y en aplicaciones en las que entren en juego grandes Bases de Datos [SALT89].

9.2.1. METODOS JERARQUICOS

Los algoritmos de este tipo, generan primeramente, una lista completa de las similaridades de todos los pares de objetos a clasificar. Es decir, parten de la llamada ***Matriz de Similaridades***.

Posteriormente, la generación de los clusters puede realizarse de dos formas:

a.- **forma divisiva**: inicialmente, se entiende que toda la colección de objetos forma un gran cluster, que subsiguientemente se divide en clusters más pequeños en base a las similaridades de los objetos.

b.- **forma aglomerativa**: inicialmente, se entiende que cada objeto forma un cluster individual, y posteriormente se van combinando los objetos más similares en la misma clase.

Como se ha expuesto, estos métodos se basan en un conocimiento a priori de las similaridades entre los pares de objetos, por lo que son relativamente costosos de ejecutar.

Sin embargo, como ventaja, podríamos señalar, que producen un único conjunto ó única distribución de clusters, independientemente del orden de proceso de los objetos.

9.2.2. METODOS HEURISTICOS

Los algoritmos de este tipo, generan los clusters rápidamente y con relativo bajo coste de ejecución dado que no requieren el conocimiento a priori de las similaridades. Es decir, parten de la matriz inicial, llamada ***Matriz de Patrones*** que es la que contiene a todos los objetos, y todos sus valores en los descriptores.

En estos métodos, la cuestión radica en lo siguiente: Dados n objetos en un espacio de m dimensiones, determinar la clasificación de objetos en k clases ó clusters, tal que los objetos de un cluster son más similares entre ellos que con los objetos de otro cluster. Para ello, se debe elegir un criterio de clasificación. Un criterio global, podría ser el de elegir de alguna manera, un *prototipo* para cada cluster, e incluir cada objeto en el cluster a cuyo prototipo sea más similar [JAIN88].

Suelen tener dos importantes desventajas:

1º Por una parte, puede ocurrir que la distribución de los objetos en los clusters no sea uniforme, es decir, que haya clusters con muchos objetos, y otros con muy pocos.

2º Y por otra, se observa que en la mayoría de los casos,la clasificación no es estable, es decir, que

según el orden de proceso de los objetos, se obtienen diferentes clusters.

Por consiguiente, vemos que cada uno de los tipos presenta una serie de características deseables en todo algoritmo de clustering. Concretando, según F. Can [CAN89], las siguientes propiedades son importantes para determinar la aceptabilidad de un algoritmo clustering:

1.- *La composición de los clusters es independiente del orden de procesado de los objetos.*

2.- *El algoritmo debe ser fácil y poco costoso de ejecutar.*

3.- *El algoritmo debe ser capaz de manejar fácil y eficientemente los nuevos objetos que entren en la colección.*

4.- *La distribución de los objetos en los clusters ha de ser lo más uniforme posible.*

5.- *Los clusters no se deben ver afectados por pequeños errores hechos en la descripción del objeto.*

6.- *Los clusters producidos deben proporcionar un ambiente de recuperación efectivo y eficiente.*

9.3. ALGORITMO PROPUESTO

9.3.1. PRESENTACION DEL ALGORITMO

Siendo "N" el número de documentos almacenados en la Base de Datos, y "M" el número de términos existentes en el vocabulario que recoge a todo nuestro dominio de referencia, el sistema puede considerarse como un conjunto de N objetos en el espacio de M dimensiones ó coordenadas.

Es decir, para las funciones de clustering ó clasificación, nos servimos de la representación del documento en forma del vector V2 de M elementos:

$$V2 = (t_1, t_2, t_3, ..., t_M)$$

Este algoritmo desarrollado en código LISP (usando GCLISP en un PC 386) pretende generar *clases de similaridad* entre los documentos almacenados. Intenta agrupar en cada clase a los documentos más similares para que luego su recuperación sea más rápida y eficaz.

Es importante la distinción entre:

1º *Generación de los clusters*: Es el proceso que se realiza por primera vez, es decir, cuando se implanta el algoritmo de clustering en el Sistema de Almacenamiento

y Recuperación de la Información. Es cuando se procesarán todos los vectores V2 correspondientes a todos los documentos almacenados en ese momento en la Base de Datos del sistema.

2º ***Mantenimiento de los clusters***: Consiste en los sucesivos procesos de clasificación que se llevarán a cabo cada vez que un documento nuevo llega, se indexa y se almacena en el Sistema de Almacenamiento y Recuperación de la Información. En este momento, quedará almacenado en el cluster existente más similar a el.

Como resultado del proceso de clasificación, este algoritmo, obtiene el llamado ***Centroide*** *ó* ***Representante de Clase***. Será un documento ficticio representante de todos los documentos similares que forman cada clase.

9.3.2. SEUDO-CODIGO DEL ALGORITMO

```
**************************************************************************

BEGIN: CLASIFICACION
   leer R
   leer T
   leer DOC.DAT
   1 -> cont1, cont2, contcen1
   V2 -> objetos1(cont2)
   1 -> objetos2(cont2)

   DOWHILE no-fin DOC.DAT
        leer DOC.DAT
        cont2 = cont2 + 1
        V2 -> objetos1(cont2)
        0 -> objetos2(cont2)
   ENDDO
   objetos1(cont1) -> centros(contcen1)
   1 -> objetos2(cont1)
   Actualizar centros2 con cont1

   DOUNTIL cont1 = cont2
        1 -> contcen2
        cont1 = cont1 + 1

        DOUNTIL contcen2 > contcen1
                Calcular D = d (objetos1(cont1),
                                centros(contcen2))

                IF D \ R
                   THEN objetos2(cont1)= objetos2(cont1)+ 1
                        Actualizar centros2 con cont1
                        1 -> pil

                        DOWHILE pil = 1
                                Recalcular centro
                                Actualizar centros(contcen2)
                                centros2(contcen2) -> lista
                                0 -> cont3, pil

                                DOUNTIL fin-lista
                                   cont3 = cont3 + 1
                                   lista(cont3) -> cont4
                                   Calc. D = d(centros(contcen2),
                                               objetos1(cont4))

                                   IF D > R
                                        THEN objetos2(cont4) =
                                             objetos2(cont4)-1
                                             1 -> pil
                                             Actualizar centros2
                                                con cont4
                                   ENDIF

                                ENDDO

                        ENDDO

                   ELSE contcen2 = contcen2 + 1
                ENDIF

        ENDDO

        IF objetos2(cont1) = 0
                THEN contcen1 = conctan1 + 1
                        objetos1(cont1) -> centros(contcen1)
                        objetos2(cont1) = objetos2(cont1) + 1
                        Actualizar centros2
        ENDIF

   ENDDO

**************************************************************************
```

```
**********************************************************************

   1 —› pil

   DOWHILE pil = 1
         1 —› cont1, pil

         DOUNTIL cont1 = cont2
                  1 —› contcen2
                  centros2(contcen2) —› lista

                  DOUNTIL contcen2 › contcen1
                           1 —› cont3

                           DOUNTIL fin-lista

                                    IF lista(cont3) = cont1
                                       THEN centros(contcen2)
                                            —› acumcen1
                                    ENDIF
                                    cont3 = cont3 + 1

                           ENDDO
                           contcen2 = contcen2 + 1

                  ENDDO
                  Calcular D1 = d(objetos1(cont1), acumcen1)
                  1 —› contcen2

                  DOUNTIL contcen2 › contcen1
                           centros(contcen2) —› acumcen2
                           Calcular D2 = d(objetos1(cont1), acuacen2)

                           IF D2 ‹ D1
                              THEN Actualizar centros2
                                    Recalcular acumcen1 —› centros
                                    Recalcular acumcen2 —› centros
                                    pil = 1
                           ENDIF
                           contcen2 = contcen2 + 1

                  ENDDO

**********************************************************************

   1 -› contcen2, c1, c2, c3

   DOUNTIL contcen2 › contcen1
         1 -› cont3
         centros(cont3) -› lista

         IF lista › T
            THEN 1 -› cont4

                  DOUNTIL fin-lista
                           lista(cont4) -› val1
                           objetos1(val1) -› objaux1(cont4)
                           cont4 = cont4 + 1
                  ENDDO
                  1 -› cont4, contd

                  DOUNTIL fin-lista
                           1 -› cont5

                           DOUNTIL fin-lista
                                    objaux1(cont4) -› aux1
                                    objaux1(cont5) -› aux2
                                    calcular D = d(aux1,aux2)
                                             -›listdist(contd)
                                    contd = contd + 1
                                    cont5 = cont5 + 1

                           ENDDO
                           cont4 = cont4 + 1

                  ENDDO
                  0 -› mayor, 1 -› cont6
```

```
                DOUNTIL fin-listdis(contd)

                        IF listdis(cont6) > mayor
                           THEN listdis(cont6) -> mayor
                                posmay = cont6 - 1
                        ENDIF
                        cont6 = cont6 + 1

                ENDDO
                pos1 = (posmay / contd) + 1
                pos2 = (posmay - contd * pos1) + 1
                objaux1(pos1) -> valor1
                objaux1(pos2) -> valor2
                1 -> cont7

                DOUNTIL fin-lista
                        calcular D = d(objaux1(cont7),valor1)
                                        -> dist1
                        calcular D =d(objaux1(cont7),valor2)
                                        -> dist2

                        IF dist1 >dist2
                           THEN objaux1(cont7) -> sub1 (c1)
                                lista(cont7) -> raub1(c1)
                                c1 = c1 +1
                           ELSE objaux2(cont7) -> sub2 (c2)
                                lista(cont7) -> raub2(c2)
                                c2 = c2 +1
                        ENDIF
                        cont7 = cont7 + 1

                ENDDO
                1 -> c1, c2

                DOUNTIL fin-sub1
                        sub1(c1) -> centros(c3)
                        raub1(c1) -> centros2(c3)
                        c1 = c1 + 1
                        c3 = c3 + 1

                ENDDO

                DOUNTIL fin-sub2
                        sub2(c2) -> centros(c3)
                        raub2(c2) -> centros2(c3)
                        c2 = c2 + 1
                        c3 = c3 + 1
                ENDDO

           ELSE lista -> centros2(c3)
                centros(cont3) -> centros(c3
                c3 = c3 + 1
        ENDIF
        cont3 = cont3 + 1

   ENDDO

*************************************************************************

   1 -> cont1

   DOUNTIL cont1 = cont2
        centros(cont1) -> CLASES.DAT
        Escribir CLASES.DAT
        centros2(cont1) -> CLASES2.DAT
        Escribir CLASES.DAT
        cont1 = cont1 + 1
        Escribir PERTENENCIAS.DAT
   ENDDO

END

*************************************************************************
```

9.4. ADAPTACION DEL ALGORITMO AL SISTEMA DE ALMACENAMIENTO Y RECUPERACION EN ESTUDIO

Una vez expuesto el algoritmo de clustering, en cuánto a su planteamiento general y desarrollo, pasamos ahora a estudiar su acomodación en el Sistema de Almacenamiento y Recuperación de la Información que estamos analizando.

Recordemos la estructura de los documentos y queries:

1.- **DOCUMENTO**: Por cada documento almacenado en la Base de Datos, tenemos:

- Un registro índice almacenado en el fichero INDICE.DAT, que tiene cinco referencias externas y diez internas (estructuradas en forma de vector V1) y un campo nombre-fichero.

- Un vector V2 de M elementos, es decir, un elemento por cada término del vocabulario, almacenado en el fichero DOC.DAT.

2.- **CENTROIDE**. Recordaremos que una vez aplicado el algoritmo de clustering, todos los documentos quedan agrupados en clases. Cada clase, contiene a los documentos más similares, y está representada por un

documento ficticio llamado centroide. Por cada centroide, tenemos:

- Un vector VC2 de M elementos, almacenado en el fichero CLASES.DAT.

- Una entrada en el fichero CLASES2.DAT, que contiene los códigos de identificación de los documentos que forman parte de ella.

3.- **QUERY**. Recordamos también, que el query que había sido inicialmente introducido en Lenguaje Natural, ha sufrido una transformación a forma estándar (QNF). Posteriormente, y para adaptación al modelo vectorial de recuperación, la parte de las referencias internas del QNF, es transformada en un vector, llamado VQ2 de representación del query mediante M elementos.

La adecuación del algoritmo al Sistema de Almacenamiento y Recuperación de la Información objeto de nuestro estudio, se centra en dos cuestiones principales:

- Cálculo de similaridades.
- Obtención de la salida.

1.- CALCULO DE SIMILARIDADES

Esta cuestión es de máxima importancia, puesto que en ella radica la utilización de nuestro algoritmo de generación automática de clases.

Se calculan las similaridades entre el query en estudio y todos los centroides obtenidos por el algoritmo (dado que cada centroide es el representante de una clase que engloba a un conjunto de documentos similares). Para ello, se aplican las fórmulas de similaridad vistas en 7.5.2., entre:

- el vector de M elementos de representación VQ2 del query.

- el vector de M elementos de representación VC2 del centroide.

Como resultado, obtendremos:

- *una clase* (ó a lo sumo dos) *altamente similar* al query, puesto que cada centroide representa a todos los documentos más similares.

- *el resto de las clases, resultarán de bajísima ó nula similaridad* con el query, por lo que, todos los documentos que las componen pueden desecharse desde este mismo momento del conjunto recuperado.

2.- OBTENCION DE LA SALIDA

El conjunto de los documentos pertenecientes a esta clase hallada como la más similar, es el conjunto de documentos recuperados en una fase inicial.

Se aplican ahora las mismas fórmulas de similaridad (vistas en el apartado 7.5.2) entre el query y cada uno de los documentos componentes de la clase recuperada (vectores VQ2 y V2), de forma que se obtenga por pantalla la lista de estos documentos ordenada en decreciente por su similaridad con el query.

Así, el usuario puede quedarse con el número de documentos que desee, teniendo la seguridad de que los elegidos son los que mejor responden a su query.

Concretando, los accesos ó enfrentamientos necesarios en este caso, es decir, en el caso de usar el algoritmo de clustering, serían:

a.- *Búsqueda en el fichero CLASES.DAT*, de la entrada correspondiente a la clase cuyo centroide más se asemeje al query solicitado (enfrentamiento entre VQ2 y VC2).

b.- El identificativo de esta entrada obtenida, nos remitirá a *la entrada con el mismo identificativo en el fichero CLASES2.DAT*, en donde encontraremos los códigos

de identificación de los documentos que componen la clase.

c.- *Estos códigos de identificación, nos remitirán a las entradas correspondientes a los documentos pertenecientes a la clase en cuestión, situadas en el fichero DOC.DAT.*

d.- *Se calculan ahora las similaridades entre el query en estudio y cada uno de los documentos* pertenecientes a la clase seleccionada (vectores VQ2 y V2), para obtener la ordenación de los documentos de mayor a menor similaridad con el query.

e.- *Cada entrada seleccionada en el fichero DOC.DAT, se relacionará con su correspondiente en el fichero INDICE.DAT* (mediante el código R1), que nos remite al fichero de texto que almacena al documento buscado.

CAPITULO 10:

EJECUCION DEL ALGORITMO

10.1. AMBIENTE DE EXPERIMENTACION

Hemos experimentado nuestro algoritmo en una Base Documental de 30 unidades u objetos (N = 30) , con un vocabulario de 50 palabras sobre el tema de Inteligencia Artificial (M = 50).

Realizamos las siguientes pruebas para comprobar la aceptabilidad (sección 11.1.2) del algoritmo:

1º Ejecución del algoritmo procesando los documentos en el orden de llegada a la colección.

2º Ejecución del algoritmo procesando los documentos en otro orden diferente.

3º Ejecución del algoritmo procesando los documentos con algún error en su descripción.

Realizamos las siguientes pruebas para estudiar la complejidad (sección 11.2.3) del algoritmo:

1º Medición del tiempo de ejecución del algoritmo procesando 10 documentos.

2º Medición del tiempo de ejecución del algoritmo procesando 15 documentos.

3º Medición del tiempo de ejecución del algoritmo procesando 20 documentos.

4º Medición del tiempo de ejecución del algoritmo procesando 25 documentos.

5º Medición del tiempo de ejecución del algoritmo procesando 30 documentos.

10.2. DATOS DE ENTRADA

```
**************************************************************
Fichero INDICE.DAT, contiene los registros-índice de los 30
documentos de la Base de Datos. Es decir, 5 referencias
externas, 10 referencias internas y un nombre-fichero
**************************************************************

D1 = 25-05-64 * 01 * nb * Bilbao * 1 *
     ( T7=0.62; T12=0.72; T15=0.69; T28=0.9; T33=0.10;
       T34=0.487; T39=0.10; T47=0.83; T49=0.31; T50=0.53 )
     file1.dat

D2 = 01-12-69 * 02 * nb * Deusto * 0 *
     ( T5=0.49; T9=0.9; T17=0.37; T19=0.83; T22=0.27;
       T28=0.11; T39=0.57; T32=0.92; T47=0.62; T44=0.12 )
     file2.dat

D3 = 05-12-69 * 03 * nb * Galicia * 1 *
     ( T5=0.81; T10=0.13; T15=0.21; T20=0.40; T25=0.58;
       T30=0.65; T35=0.32; T40=0.28; T45=0.03; T50=0.99 )
     file3.dat

D4 = 10-09-70 * 04 * nb * Cambridge * 1 *
     ( T4=0.13; T14=0.54; T16=0.92; T26=0.82; T29=0.91;
       T31=0.84; T38=0.89; T41=0.52; T46=0.936; T50=0.48 )
     file4.dat

D5 = 28-12-70 * 05 * nb * Londres * 1 *
     ( T7=0.68; T12=0.731; T15=0.671; T25=0.863; T32=0.20;
       T34=0.421; T37=0.15; T46=0.33; T47=0.86; T48=0.61 )
     file5.dat

D6 = 08-01-71 * 06 * nb * Deusto * 0 *
     ( T7=0.63; T12=0.75; T15=0.63; T22=0.71; T27=0.15;
       T34=0.40; T38=0.21; T41=0.59; T42=0.43; T47=0.821 )
     file6.dat

D7 = 23-05-71 * 07 * nb * Barcelona * 1 *
     ( T5=0.48; T12=0.08; T17=0.35; T22=0.22; T26=0.89;
       T29=0.21; T39=0.55; T45=0.87; T47=0.66; T48=0.35 )
     file7.dat

D8 = 09-02-72 * 08 * nb * Bilbao * 1 *
     ( T8=0.31; T13=0.55; T14=0.42; T27=0.66; T33=0.62;
       T35=0.72; T40=0.78; T46=0.81; T48=0.89; T49=0.90 )
     file8.dat

D9 = 10-10-73 * 09 * nb * Mexico * 1 *
     ( T1=0.59; T6=0.21; T11=0.68; T16=0.99; T26=0.88;
       T29=0.92; T31=0.86; T38=0.89; T41=0.58; T46=0.91 )
     file9.dat
```

D10 = 04-12-73 * 10 * nb * Deusto * 0 *
(T_8=0.33; T_{13}=0.51; T_{16}=0.59; T_{27}=0.62; T_{34}=0.68;
T_{35}=0.79; T_{40}=0.75; T_{43}=0.88; T_{48}=0.87; T_{49}=0.95)
file10.dat

D11 = 14-02-74 * 11 * nb * Madrid * 1 *
(T_2=0.79; T_{10}=0.15; T_{15}=0.212; T_{20}=0.44; T_{23}=0.61;
T_{30}=0.65; T_{32}=0.42; T_{33}=0.19; T_{45}=0.93; T_{50}=0.95)
file11.dat

D12 = 30-09-74 * 12 * nb * Bruselas * 1 *
(T_5=0.43; T_{15}=0.7; T_{17}=0.32; T_{22}=0.27; T_{27}=0.78;
T_{33}=0.171; T_{39}=0.59; T_{42}=0.853; T_{46}=0.22; T_{47}=0.69)
file12.dat

D13 = 23-03-75 * 13 * nb * Lyon * 1 *
(T_8=0.39; T_9=0.46; T_{14}=0.61; T_{27}=0.69; T_{32}=0.72;
T_{35}=0.77; T_{40}=0.72; T_{45}=0.83; T_{48}=0.89; T_{49}=0.99)
file13.dat

D14 = 09-06-75 * 14 * nb * Valencia * 1 *
(T_1=0.19; T_7=0.07; T_{16}=0.96; T_{26}=0.83; T_{29}=0.91;
T_{31}=0.84; T_{38}=0.89; T_{41}=0.53; T_{46}=0.93; T_{50}=0.64)
file14.dat

D15 = 21-01-76 * 15 * nb * Bilbao * 1 *
(T_2=0.8; T_{10}=0.19; T_{15}=0.27; T_{20}=0.48; T_{24}=0.63;
T_{30}=0.61; T_{35}=0.45; T_{37}=0.23; T_{43}=0.15; T_{50}=0.9)
file15.dat

D16 = 07-03-76 * 16 * nb * Madrid * 1 *
(T_3=0.25; T_{13}=0.49; T_{16}=0.92; T_{26}=0.8; T_{29}=0.92;
T_{31}=0.86; T_{38}=0.89; T_{41}=0.521; T_{46}=0.932; T_{49}=0.71)
file16.dat

D17 = 27-11-76 * 17 * nb * Londres * 1 *
(T_7=0.67; T_{12}=0.749; T_{15}=0.682; T_{27}=0.12; T_{29}=0.17;
T_{34}=0.492; T_{36}=0.09; T_{46}=0.39; T_{47}=0.81; T_{48}=0.62)
file17.dat

D18 = 04-02-77 * 18 * nb * Bruselas * 1 *
(T_8=0.39; T_{10}=0.44; T_{16}=0.63; T_{27}=0.63; T_{31}=0.76;
T_{35}=0.73; T_{40}=0.76; T_{44}=0.85; T_{48}=0.87; T_{49}=0.98)
file18.dat

D19 = 08-05-77 * 19 * nb * Valencia * 1 *
(T_4=0.82; T_{10}=0.15; T_{15}=0.25; T_{20}=0.46; T_{24}=0.622;
T_{30}=0.687; T_{34}=0.5; T_{35}=0.36; T_{46}=0.24; T_{50}=0.92)
file19.dat

D20 = 27-05-77 * 20 * nb * Galicia * 1 *
(T_5=0.47; T_6=0.6; T_{17}=0.33; T_{22}=0.288; T_{29}=0.72;
T_{30}=0.194; T_{39}=0.57; T_{43}=0.862; T_{47}=0.696; T_{48}=0.268)
file20.dat

D21 = 29-04-78 * 21 * nb * Lyon * 1 *
(T_2=0.78; T_4=0.8; T_{10}=0.15; T_{15}=0.25; T_{20}=0.49;
T_{23}=0.63; T_{30}=0.6; T_{34}=0.06; T_{40}=0.25; T_{50}=0.91)
file21.dat

D22 = 13-08-78 * 22 * nb * Barcelona * 1 *
(T_5=0.45; T_9=0.93; T_{17}=0.335; T_{22}=0.26; T_{26}=0.09;
T_{33}=0.17; T_{39}=0.56; T_{42}=0.84; T_{45}=0.25; T_{47}=0.63)
file22.dat

D23 = 07-09-78 * 23 * nb * Salamanca * 1 *
(T_1=0.5; T_4=0.16; T_{13}=0.45; T_{16}=0.95; T_{26}=0.83;
T_{29}=0.91; T_{31}=0.84; T_{36}=0.75; T_{38}=0.84; T_{46}=0.94)
file23.dat

D24 = 28-03-85 * 24 * nb * Deusto * 0 *
(T_7=0.65; T_{12}=0.71; T_{15}=0.69; T_{22}=0.73; T_{28}=0.93;
T_{32}=0.21; T_{34}=0.45; T_{36}=0.09; T_{42}=0.44; T_{47}=0.85)
file24.dat

D25 = 23-08-86 * 25 * nb * Bilbao * 1 *
(T_8=0.37; T_9=0.47; T_{13}=0.53; T_{27}=0.635; T_{31}=0.78;
T_{33}=0.68; T_{35}=0.719; T_{40}=0.74; T_{48}=0.81; T_{49}=0.93)
file25.dat

D26 = 12-04-87 * 26 * nb * Valencia * 1 *
(T_4=0.81; T_{10}=0.10; T_{15}=0.21; T_{20}=0.42; T_{23}=0.65;
T_{24}=0.625; T_{30}=0.68; T_{34}=0.04; T_{37}=0.25; T_{50}=0.95)
file26.dat

D27 = 09-10-87 * 27 * nb * Londres * 1 *
(T_8=0.355; T_{10}=0.42; T_{12}=0.01; T_{27}=0.677; T_{32}=0.7;
T_{34}=0.6; T_{35}=0.77; T_{40}=0.79; T_{48}=0.83; T_{49}=0.9)
file27.dat

D28 = 05-03-89 * 28 * nb * Deusto * 0 *
(T_5=0.47; T_9=0.95; T_{17}=0.35; T_{22}=0.24; T_{29}=0.715;
T_{33}=0.18; T_{39}=0.585; T_{43}=0.65; T_{45}=0.29; T_{47}=0.68)
file28.dat

D29 = 01-04-90 * 29 * nb * Paris * 1 *
(T_1=0.51; T_7=0.06; T_{13}=0.48; T_{16}=0.93; T_{26}=0.85;
T_{29}=0.92; T_{31}=0.86; T_{37}=0.88; T_{38}=0.89; T_{46}=0.95)
file29.dat

D30 = 25-05-90 * 30 * nb * Bilbao * 1 *
(T_7=0.665; T_{12}=0.73; T_{15}=0.68; T_{22}=0.75; T_{28}=0.98;
T_{34}=0.44; T_{39}=0.11; T_{42}=0.45; T_{47}=0.832; T_{49}=0.33)
file30.dat

****

Fichero VOCAB.DAT, contiene el vocabulario de los 50 términos sobre el tema de Inteligencia Artificial, que pueden ser tomados como índices.

****

T_1 : ALGORITMO
T_2 : APRENDIZAJE
T_3 : ARBOL
T_4 : ARTIFICIAL
T_5 : ASOCIATIVO
T_6 : ATRIBUTO
T_7 : AUTOMATICO
T_8 : AXIOMA
T_9 : BUSQUEDA
T_{10} : CLAUSULA
T_{11} : CONDICIONAL
T_{12} : CONOCIMIENTO
T_{13} : CONSECUENTE
T_{14} : DEDUCCION
T_{15} : DEMOSTRACION
T_{16} : DIFUSO
T_{17} : ENCADENAMIENTO
T_{18} : EXISTENCIAL
T_{19} : EXPERTO
T_{20} : FRAME
T_{21} : HEURISTICA
T_{22} : HORN
T_{23} : INFERENCIA
T_{24} : INFORMATICA
T_{25} : INTELIGENCIA
T_{26} : LISP
T_{27} : LISTA
T_{28} : LISTADO
T_{29} : LOGICO
T_{30} : MAQUINA
T_{31} : MODAL
T_{32} : NATURAL
T_{33} : PARALELO
T_{34} : PARSING
T_{35} : PREDICADO
T_{36} : PROBLEMA
T_{37} : PROPOSICION
T_{38} : PROLOG
T_{39} : RAZONAMIENTO
T_{40} : REGLA
T_{41} : RECURSION
T_{42} : RESOLUCION
T_{43} : SEMANTICO
T_{44} : SIMBOLICO
T_{45} : SINTACTICO
T_{46} : TEOREMA
T_{47} : UNIFICADOR
T_{48} : UNIFORME
T_{49} : UNIVERSAL
T_{50} : VARIABLE

****
Fichero DOC.DAT, contiene los vectores V2 de los 30 documentos almacenados en la Base de Datos.
****

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
D1	0	0	0	0	0	0	.62	0	0	0
D2	0	0	0	0	.49	0	0	0	.9	0
D3	0	0	0	0	.81	0	0	0	0	.15
D4	0	0	0	.13	0	0	0	0	0	0
D5	0	0	0	0	0	0	.68	0	0	0
D6	0	0	0	0	0	0	.63	0	0	0
D7	0	0	0	0	.48	0	0	0	0	0
D8	0	0	0	0	0	0	0	.31	0	0
D9	.59	0	0	0	0	.21	0	0	0	0
D10	0	0	0	0	0	0	0	.33	0	0
D11	0	.79	0	0	0	0	0	0	0	.15
D12	0	0	0	0	.43	0	0	0	0	0
D13	0	0	0	0	0	0	0	.39	.46	0
D14	.59	0	0	0	0	0	.07	0	0	0
D15	0	.8	0	0	0	0	0	0	0	.19
D16	0	0	.25	0	0	0	0	0	0	0
D17	0	0	0	0	0	0	.67	0	0	0
D18	0	0	0	0	0	0	0	.35	0	.44
D19	0	0	0	.82	0	0	0	0	0	.15
D20	0	0	0	0	.47	.6	0	0	0	0
D21	0	.78	0	.8	0	0	0	0	0	.15
D22	0	0	0	0	.45	0	0	0	.93	0
D23	.5	0	0	.16	0	0	0	0	0	0
D24	0	0	0	0	0	0	.65	0	0	0
D25	0	0	0	0	0	0	0	.37	.47	0
D26	0	0	0	.81	0	0	0	0	0	.1
D27	0	0	0	0	0	0	0	.355	0	.42
D28	0	0	0	0	.47	0	0	0	.95	0
D29	.51	0	0	0	0	0	.06	0	0	0
D30	0	0	0	0	0	0	.665	0	0	0

	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
D1	0	.72	0	0	.69	0	0	0	0	0
D2	0	0	0	0	0	0	.37	0	.83	0
D3	0	0	0	0	.21	0	0	0	0	.4
D4	0	0	0	.54	0	.92	0	0	0	0
D5	0	.731	0	0	.671	0	0	0	0	0
D6	0	.75	0	0	.63	0	0	0	0	0
D7	0	.08	0	0	0	0	.35	0	0	0
D8	0	0	0	.42	.55	0	0	0	0	0
D9	.68	0	0	0	0	.99	0	0	0	0
D10	0	0	.51	0	0	.59	0	0	0	0
D11	0	0	0	0	.212	0	0	0	0	.44
D12	0	0	0	0	.7	0	.32	0	0	0
D13	0	0	0	.61	0	0	0	0	0	0
D14	0	0	0	0	0	.96	0	0	0	0
D15	0	0	0	0	.27	0	0	0	0	.48
D16	0	0	.49	0	0	.92	0	0	0	0
D17	0	.749	0	0	.682	0	0	0	0	0
D18	0	0	0	0	0	.63	0	0	0	0
D19	0	0	0	0	.25	0	0	0	0	.46
D20	0	0	0	0	0	0	.33	0	0	0
D21	0	0	0	0	.25	0	0	0	0	.49
D22	0	0	0	0	0	0	.335	0	0	0
D23	0	0	.45	0	0	.95	0	0	0	0
D24	0	.71	0	0	.69	0	0	0	0	0
D25	0	0	.53	0	0	0	0	0	0	0
D26	0	0	0	0	.21	0	0	0	0	.42
D27	0	.01	0	0	0	0	0	0	0	0
D28	0	0	0	0	0	0	.35	0	0	0
D29	0	0	.48	0	0	.93	0	0	0	0
D30	0	.73	0	0	.68	0	0	0	0	0

	T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}	T_{27}	T_{28}	T_{29}	T_{30}
D1	0	0	0	0	0	0	0	.9	0	0
D2	0	.22	0	0	0	0	0	.11	0	0
D3	0	0	0	0	.58	0	0	0	0	.65
D4	0	0	0	0	0	.22	0	0	.91	0
D5	0	0	0	0	0	0	0	.95	0	0
D6	0	.71	0	0	0	0	.15	.93	0	0
D7	0	.22	0	0	0	.08	0	0	.7	0
D8	0	0	0	0	0	0	.66	0	0	0
D9	0	0	0	0	0	.88	0	0	.92	0
D10	0	0	0	0	0	0	.62	0	0	0
D11	0	0	.61	0	0	0	0	0	0	.65
D12	0	.27	0	0	0	0	.78	0	0	0
D13	0	0	0	0	0	0	.69	0	0	0
D14	0	0	0	0	0	.83	0	0	.91	0
D15	0	0	0	.63	0	0	0	0	0	.61
D16	0	0	0	0	0	.8	0	0	.92	0
D17	0	0	0	0	0	0	.12	.97	0	0
D18	0	0	0	0	0	0	.63	0	0	0
D19	0	0	0	.62	0	0	0	0	0	.687
D20	0	.288	0	0	0	0	0	0	.72	.194
D21	0	0	.63	0	0	0	0	0	0	.6
D22	0	.26	0	0	0	.09	0	0	0	0
D23	0	0	0	0	0	.83	0	0	.91	0
D24	0	.73	0	0	0	0	0	.93	0	0
D25	0	0	0	0	0	0	.635	0	0	0
D26	0	0	.65	.625	0	0	0	0	0	.68
D27	0	0	0	0	0	0	.677	0	0	0
D28	0	.24	0	0	0	0	0	0	.715	0
D29	0	0	0	0	0	.85	0	0	.92	0
D30	0	.75	0	0	0	0	0	.98	0	0

	T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	T_{36}	T_{37}	T_{38}	T_{39}	T_{40}
D1	0	0	.1	.487	0	0	0	0	.1	0
D2	0	.92	0	0	0	0	0	0	.57	0
D3	0	0	0	0	.32	0	0	0	0	.28
D4	.84	0	0	0	0	0	0	.89	0	0
D5	0	.2	0	.421	0	0	.15	0	0	0
D6	0	0	0	.4	0	0	0	0	0	0
D7	0	0	0	0	0	0	0	0	.55	0
D8	0	0	.62	0	.72	0	0	0	0	.78
D9	.86	0	0	0	0	0	0	.89	0	0
D10	0	0	0	.68	.79	0	0	0	0	.75
D11	0	.42	.19	0	0	0	0	0	0	0
D12	0	0	.171	0	0	0	0	0	.59	0
D13	0	.72	0	0	.77	0	0	0	0	.72
D14	.84	0	0	0	0	0	0	.89	0	0
D15	0	0	0	0	.45	0	.23	0	0	0
D16	.86	0	0	0	0	0	0	.89	0	0
D17	0	0	0	.492	0	.09	0	0	0	0
D18	.76	0	0	0	.73	0	0	0	0	.76
D19	0	0	0	.05	.36	0	0	0	0	0
D20	0	0	0	0	0	0	0	0	.57	0
D21	0	0	0	.06	0	0	0	0	0	.25
D22	0	0	.17	0	0	0	0	0	.56	0
D23	.84	0	0	0	0	.75	0	.89	0	0
D24	0	.21	0	.45	0	.09	0	0	0	0
D25	.78	0	.68	0	.719	0	0	0	0	.74
D26	0	0	0	.04	0	0	.25	0	0	0
D27	0	.7	0	.6	.77	0	0	0	0	.79
D28	0	0	.18	0	0	0	0	0	.585	0
D29	.86	0	0	0	0	0	.88	.89	0	0
D30	0	0	0	.44	0	0	0	0	.11	0

	T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	T_{46}	T_{47}	T_{48}	T_{49}	T_{50}
D1	0	0	0	0	0	0	.83	0	.31	.53
D2	0	0	0	.12	0	0	.62	0	0	0
D3	0	0	0	0	.03	0	0	0	0	.99
D4	.52	0	0	0	0	.936	0	0	0	.48
D5	0	0	0	0	0	.33	.86	.61	0	0
D6	.59	.43	0	0	0	0	.821	0	0	0
D7	0	0	0	0	.27	0	.66	.35	0	0
D8	0	0	0	0	0	.81	0	.89	.9	0
D9	.58	0	0	0	0	.91	0	0	0	0
D10	0	0	.88	0	0	0	0	.87	.95	0
D11	0	0	0	0	.93	0	0	0	0	.95
D12	0	.853	0	0	0	.22	.69	0	0	0
D13	0	0	0	0	.83	0	0	.89	.99	0
D14	.53	0	0	0	0	.93	0	0	0	.64
D15	0	0	.15	0	0	0	0	0	0	.9
D16	.521	0	0	0	0	.932	0	0	.71	0
D17	0	0	0	0	0	.39	.81	.62	0	0
D18	0	0	0	.85	0	0	0	.87	.98	0
D19	0	0	0	0	0	.24	0	0	0	.92
D20	0	0	.62	0	0	0	.696	.268	0	0
D21	0	0	0	0	0	0	0	0	0	.91
D22	0	.84	0	0	.25	0	.63	0	0	0
D23	0	0	0	0	0	.94	0	0	0	0
D24	0	.44	0	0	0	0	.85	0	0	0
D25	0	0	0	0	0	0	0	.81	.93	0
D26	0	0	0	0	0	0	0	0	0	.95
D27	0	0	0	0	0	0	0	.83	.9	0
D28	0	0	.65	0	.29	0	.68	0	0	0
D29	0	0	0	0	0	.95	0	0	0	0
D30	0	.45	0	0	0	0	.832	0	.33	0

10.3. RESULTADOS OBTENIDOS

****
Clasificación obtenida procesando los documentos de la Base de Datos en el orden de llegada a la colección, con un radio igual a 2.
****

* ORDEN DE PROCESO DE LOS DOCUMENTOS *

D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25, D26, D27, D28, D29, D30.

* COORDENADAS DE LOS CENTROS DE LAS CLASES OBTENIDAS *

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
C1	0	0	0	0	.061	0	.559	0	0	0
C2	0	0	0	0	.472	.12	0	0	.556	0
C3	0	.395	0	.405	.135	0	0	0	0	.145
C4	.298	0	.041	.048	0	.035	.021	0	0	0
C5	0	0	0	0	0	0	0	.357	.155	.143

	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
C1	0	.612	0	0	.677	0	.045	0	0	0
C2	0	.015	0	0	0	0	.347	0	.166	0
C3	0	0	0	0	.233	0	0	0	0	.448
C4	.113	0	.236	.09	0	.945	0	0	0	0
C5	0	.001	.265	.171	0	.203	0	0	0	0

	T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}	T_{27}	T_{28}	T_{29}	T_{30}
C1	0	.351	0	0	.123	0	.149	.401	.024	0
C2	0	.255	0	0	0	.196	0	.022	.329	.038
C3	0	0	.315	.312	.096	0	0	0	0	.646
C4	0	0	0	0	0	.835	0	0	.915	0
C5	0	0	0	0	0	0	.652	0	0	0

	T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	T_{36}	T_{37}	T_{38}	T_{39}	T_{40}
C1	0	.058	.038	.384	0	.025	.021	.03	.114	0
C2	0	.184	.07	0	0	0	0	0	.567	0
C3	0	.07	.031	.1	.188	0	.08	0	0	.088
C4	.85	0	0	0	0	.125	.146	.881	0	0
C5	.256	.236	.216	.213	.749	0	0	0	0	.756

	T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	T_{46}	T_{47}	T_{48}	T_{49}	T_{50}
C1	.084	.31	0	0	0	.134	.813	.175	.091	.075
C2	0	.168	.302	.024	.282	0	.657	.123	0	0
C3	0	0	.025	0	.16	.04	0	0	0	.936
C2	.358	0	0	0	0	.932	0	0	.118	.186
C5	0	0	.146	.141	.138	.135	0	.859	.941	0

****

Clasificación obtenida procesando los documentos de la Base de Datos en diferente orden al de llegada, con un radio igual a 2.

****

* ORDEN DE PROCESO DE LOS DOCUMENTOS *

D25, D3, D23, D28, D29, D6, D1, D11, D16, D12, D5, D13, D8, D14, D7, D15, D19, D10, D20, D21, D2, D24, D26, D4, D27, D30, D17, D18, D22, D9.

* COORDENADAS DE LOS CENTROS DE LAS CLASES OBTENIDAS *

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
C1	0	0	0	0	.061	0	.559	0	0	0
C2	0	0	0	0	.472	.12	0	0	.556	0
C3	0	.395	0	.405	.135	0	0	0	0	.145
C4	.298	0	.041	.048	0	.035	.021	0	0	0
C5	0	0	0	0	0	0	0	.357	.155	.143

	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
C1	0	.612	0	0	.677	0	.045	0	0	0
C2	0	.015	0	0	0	0	.347	0	.166	0
C3	0	0	0	0	.233	0	0	0	0	.448
C4	.113	0	.236	.09	0	.945	0	0	0	0
C5	0	.001	.265	.171	0	.203	0	0	0	0

	T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}	T_{27}	T_{28}	T_{29}	T_{30}
C1	0	.351	0	0	.123	0	.149	.401	.024	0
C2	0	.255	0	0	0	.196	0	.022	.329	.038
C3	0	0	.315	.312	.096	0	0	0	0	.646
C4	0	0	0	0	0	.835	0	0	.915	0
C5	0	0	0	0	0	0	.652	0	0	0

	T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	T_{36}	T_{37}	T_{38}	T_{39}	T_{40}
C1	0	.058	.038	.384	0	.025	.021	.03	.114	0
C2	0	.184	.07	0	0	0	0	0	.567	0
C3	0	.07	.031	.1	.188	0	.08	0	0	.088
C4	.85	0	0	0	0	.125	.146	.881	0	0
C5	.256	.236	.216	.213	.749	0	0	0	0	.756

	T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	T_{46}	T_{47}	T_{48}	T_{49}	T_{50}
C1	.084	.31	0	0	0	.134	.813	.175	.091	.075
C2	0	.168	.302	.024	.282	0	.657	.123	0	0
C3	0	0	.025	0	.16	.04	0	0	0	.936
C2	.358	0	0	0	0	.932	0	0	.118	.186
C5	0	0	.146	.141	.138	.135	0	.859	.941	0

****
Clasificación obtenida procesando la colección con pequeños errores en los registros-índice de D3, D16, D25; y con un radio igual a 2.
****

* REGISTROS-INDICE DE LOS DOCUMENTOS D3, D16 Y D25 *

D3 = 05-12-69 * 03 * nb * Galicia * 0 *
(T_5=0.85; T_{10}=0.13; T_{15}=0.27; T_{20}=0.40; T_{25}=0.58;
T_{30}=0.65; T_{35}=0.32; T_{40}=0.28; T_{45}=0.03; T_{50}=0.99)
file3.dat

D16 = 07-03-76 * 16 * nb * Madrid * 1 *
(T_3=0.25; T_{13}=0.49; T_{16}= 0.92; T_{26}=0.8; T_{29}=0.9;
T_{31}=0.86; T_{38}=0.84; T_{41}=0.521; T_{46}=0.932; T_{49}= 0.71)
file16.dat

D25 = 23-08-86 * 25 * nb * Bilbao * 0 *
(T_8=0.37; T_9=0.47; T_{13}=0.53; T_{27}=0.635; T_{31}=0.75;
T_{33}=0.68; T_{35}=0.719; T_{40}=0.74; T_{48}=0.81; T_{49}=0.9)
file25.dat

* COORDENADAS DE LOS CENTROS DE LAS CLASES OBTENIDAS *

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
C1	0	0	0	0	.061	0	.559	0	0	0
C2	0	0	0	0	.472	.12	0	0	.556	0
C3	0	.395	0	.405	.141	0	0	0	0	.145
C4	.298	0	.041	.048	0	.035	.021	0	0	0
C5	0	0	0	0	0	0	0	.357	.155	.143

	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	T_{17}	T_{18}	T_{19}	T_{20}
C1	0	.612	0	0	.677	0	.045	0	0	0
C2	0	.015	0	0	0	0	.347	0	.166	0
C3	0	0	0	0	.243	0	0	0	0	.448
C4	.113	0	.236	.09	0	.945	0	0	0	0
C5	0	.001	.265	.171	0	.203	0	0	0	0

	T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}	T_{27}	T_{28}	T_{29}	T_{30}
C1	0	.351	0	0	.123	0	.149	.401	.024	0
C2	0	.255	0	0	0	.196	0	.022	.329	.038
C3	0	0	.315	.312	.096	0	0	0	0	.646
C4	0	0	0	0	0	.835	0	0	.911	0
C5	0	0	0	0	0	0	.652	0	0	0

	T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	T_{36}	T_{37}	T_{38}	T_{39}	T_{40}
C1	0	.058	.038	.384	0	.025	.021	.03	.114	0
C2	0	.184	.07	0	0	0	0	0	.567	0
C3	0	.07	.031	.1	.188	0	.08	0	0	.088
C4	.85	0	0	0	0	.125	.146	.881	0	0
C5	.251	.236	.216	.213	.749	0	0	0	0	.756

	T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	T_{46}	T_{47}	T_{48}	T_{49}	T_{50}
C1	.084	.31	0	0	0	.134	.813	.175	.091	.075
C2	0	.168	.302	.024	.282	0	.657	.123	0	0
C3	0	0	.025	0	.16	.04	0	0	0	.936
C2	.358	0	0	0	0	.932	0	0	.118	.186
C5	0	0	.146	.141	.138	.135	0	.859	.936	0

****

Fichero PERTENENCIAS.DAT, que contiene los factores de pertenencia de los documentos a las clases generadas.

****

	CLASE1	CLASE2	CLASE3	CLASE4	CLASE5
D1	+1.079	-0.003	-0.041	-0.874	-0.516
D2	-0.093	+0.774	-0.340	-0.952	-0.564
D3	-0.143	+0.029	+0.856	-0.759	-0.375
D4	-0.737	-0.554	-0.580	+1.268	-0.842
D5	+0.864	-0.046	-0.284	-0.854	-0.455
D6	+1.150	+0.102	-0.264	-0.767	-0.591
D7	+0.087	+0.812	-0.159	-0.484	-0.410
D8	-0.512	-0.527	-0.544	-0.811	+0.959
D9	-0.860	-0.658	-0.878	+1.196	-0.975
D10	-0.543	-0.520	-0.604	-0.964	+0.873
D11	-0.374	-0.195	+0.844	-0.913	-0.621
D12	+0.501	+0.466	-0.196	-0.803	-0.393
D13	-0.643	-0.398	-0.593	-1.184	+0.835
D14	-0.718	-0.554	-0.543	+1.381	-0.872
D15	-0.19	-0.148	+1.151	-0.759	-0.419
D16	-0.777	-0.609	-0.818	+1.18	-0.634
D17	+1.12	+0.123	-0.173	-0.656	-0.283
D18	-0.658	-0.631	-0.634	-0.781	+0.833
D19	-0.157	-0.23	+1.089	-0.781	-0.458
D20	+0.065	+0.874	-0.177	-0.603	-0.456
D21	-0.274	-0.256	+1.201	-0.826	-0.563
D22	+0.189	+1.046	-0.181	-0.792	-0.474
D23	-0.8	-0.597	-0.785	+1.154	-0.87
D24	+1.212	-0.026	-0.395	-1.017	-0.733
D25	-0.544	-0.407	-0.542	-0.843	+1.033
D26	-0.265	-0.234	+1.18	-0.749	-0.622
D27	-0.363	-0.404	-0.401	-1.049	+1.104
D28	-0.020	+1.226	-0.259	-0.652	-0.932
D29	-0.854	-0.667	-0.851	+1.07	-0.932
D30	+1.156	-0.065	-0.444	-1.04	-0.674

* IDENTIFICATIVOS DE LOS DOCUMENTOS PERTENECIENTES A CADA CLASE Y GRADO DE PERTENENCIA A ELLA *

CLASE 1

IDENTIFICATIVO	PERTENENCIA
D1	1.079
D5	0.864
D6	1.15
D12	0.501
D17	1.12
D24	1.212
D30	1.156

CLASE 2

IDENTIFICATIVO	PERTENENCIA
D2	0.774
D7	0.812
D20	0.874
D22	1.046
D28	1.226

CLASE 3

IDENTIFICATIVO	PERTENENCIA
D3	0.856
D11	0.844
D15	1.151
D19	1.089
D21	1.201
D26	1.18

CLASE 4

IDENTIFICATIVO	PERTENENCIA
D4	1.268
D9	1.196
D14	1.381
D16	1.18
D23	1.154
D29	1.07

CLASE 5

IDENTIFICATIVO	PERTENENCIA
D8	0.959
DI0	0.873
D13	0.835
D18	0.833
D25	1.033
D27	1.104

CAPITULO 11:

ACEPTABILIDAD

Y COMPLEJIDAD

11.1. ACEPTABILIDAD DEL ALGORITMO PROPUESTO

11.1.1. BASES DE COMPARACION

Es difícil realizar un estudio comparativo entre los diferentes algoritmos ó métodos clustering existentes. Milligan [MILL81], realizó un estudio comparativo de los algoritmos clustering previos a 1981, tratando de entender la seguridad de cada uno de ellos en proporcionar el adecuado número de clusters. Anderberg [ANDE73] proporcionó una discusión sobre los factores importantes a tener en cuenta a la hora de comparar algoritmos clustering. Dubes y Jain [DUBE76] aplicaron algunos de estos factores.

Pero es difícil recopilar todas las conclusiones alcanzadas en estos trabajos. No hay dos estudios que usen la misma metodología comparativa ó que incluyan todos los métodos clustering. No está claro que dos estudios diferentes usen la misma implementación de un algoritmo de clasificación.

Por lo tanto, no es factible una comparación teórica entre algoritmos clustering, dado que, ***no es posible modelarlos matemáticamente de forma que los modelos obtenidos puedan ser comparados*** [SALT89]. Algunos de los trabajos que han intentado proporcionar una lista de las características esenciales para medir la utilidad de un algoritmo clustering, han llegado a conclusiones contrarias.

Gower [GOWE67] en su estudio, llegó a la conclusión de que la dificultad a la hora de definir clusters radica en *la fuerte relación que hay entre el algoritmo clustering y el tipo de los clusters obtenidos*.

Hartigan [HART85] llegó a la conclusión de que, *diferentes métodos ó algoritmos de clasificación son buenos para diferentes propósitos, por lo que no se puede decir que unos sean mejores que otros*.

Otra estrategia de comparación de algoritmos clustering es la de preparar una lista de *criterios de aceptabilidad* tales como los propuestos por Fisher y Van Ness [FISH71]; por Rubin [RUBI67]; ó por Can [CAN 89]. La idea es proponer una propiedad p deseable y llamar al algoritmo ***p-admisible*** si cumple esta propiedad. Las propiedades, reflejan la manera en la que se forman los clusters, la estructura de los datos, y la sensitividad del algoritmo a los cambios en los datos ó en el tamaño deseado de los clusters.

Hemos elegido en ésta tesis, estudiar la aceptabilidad del algoritmo propuesto basándonos en las propiedades proporcionadas por Can [CAN89], ya expuestas en el capítulo anterior.

11.1.2. COMPROBACION DE LA ACEPTABILIDAD

A la vista de los resultados obtenidos y expuestos en el CAPITULO 10 de la tesis, fácilmente se puede comprobar que el algoritmo propuesto de clustering, cumple todas y cada una de las propiedades de aceptabilidad propuestas por Can [CAN89]:

1º *La Composición de los clusters generados es independiente del orden de proceso de los documentos.* Como puede observarse en los resultados obtenidos, la clasificación obtenida en dos ejecuciones diferentes del algoritmo es la misma, tratando cada una de ellas los documentos en diferente orden. Nuestro algoritmo, incluye un proceso llamado **ESTABILIDAD**, especialmente diseñado para cubrir esta característica, es decir, que *la clasificación de una Base de Datos sea la misma*, sea cual sea el orden de proceso de los documentos.

2º *El algoritmo es fácil y poco costoso de ejecutar.* Evidentemente, esta característica, depende directamente de la complejidad del algoritmo, que será analizada en la sección 11.2 del capítulo 11.

3º ***El algoritmo es capaz de manejar fácil y eficientemente los nuevos documentos que llegan a la colección.*** Para ello, se ha creado expresamente el proceso **CARGA**, cuya única finalidad es *tratar los nuevos documentos*, es decir, obtener y transformar sus índices, almacenarlos en los ficheros apropiados, y posteriormente incluirlos en el cluster más conveniente.

4º *La distribución de los documentos en los clusters es uniforme*. A la vista de los resultados incluidos en el capítulo 10, nuestro algoritmo produce una clasificación uniforme, es decir, los clusters están formados todos por más ó menos el mismo número de documentos, concretamente por 5, 6 y 7 documentos. Esto se ha conseguido, mediante *el control del tamaño del cluster*: el proceso **UNIFORMIDAD**, especialmente diseñado para ello, produce la subdivisión de los clusters cuando su tamaño exceda del tamaño medio deseado.

5º *Los clusters no se ven afectados por pequeños errores cometidos en la descripción del documento.* Esta característica, también queda demostrada en los resultados del capítulo 10. Simplemente, repetimos la ejecución *variando los registros-índice de algunos documentos*, en concreto, de los documentos D3, D16 y D25 y el algoritmo obtuvo los mismos clusters con pequeños cambios en sus coordenadas.

6º *Los clusters generados producen un ambiente de recuperación efectivo y eficiente.* Los documentos juzgados como más similares, han quedado agrupados en áreas comunes, por lo que la recuperación es más eficiente al limitarse la búsqueda entre los *documentos pertenecientes a los clusters más similares al query* dado.

11.2. ESTUDIO DE LA COMPLEJIDAD

La teoría de *Complejidad algorítmica*, se encarga de definir los criterios básicos para saber si un problema computable es "factible". Es decir, si tiene un algoritmo eficiente para su resolución, y en este caso, cuál es su grado de eficiencia [SAEZ87].

La complejidad, se mide por un orden de magnitud (de tiempo ó de espacio), expresado en forma abstracta como función del tamaño del problema. Decir "forma abstracta", significa, expresar el tiempo ó el espacio, como un número de pasos operativos elementales, al objeto de independizarlo de la velocidad concreta de la máquina ejecutora.

El tamaño de los datos de entrada, es el parámetro utilizado para medir la complejidad computacional. Por lo tanto, una vez fijado el problema y definida su entrada, su

complejidad dependerá de la clase (tiempo, espacio) y número de los recursos considerados (máquinas secuenciales, máquinas paralelas) y del algoritmo elegido.

11.2.1. TIPOS DE COMPLEJIDAD EN FUNCION DEL TAMAÑO DE LA ENTRADA

La forma de crecimiento de los recursos necesarios en la ejecución de un determinado algoritmo, se denota por O(f(n)), siendo n el tamaño de la entrada.

Así, las distintas complejidades (por ejemplo temporales) pueden ser:

- complejidad logarítmica: O (log n)
- complejidad lineal: O (n)
- complejidad polinómica: O (n^c)
- complejidad exponencial: O (c^n)

A grandes rasgos, los problemas computables, se clasifican en:

- *buenos ó eficientes*, si su complejidad es polinómica

- *malos ó complejos*, si su complejidad es exponencial

El problema cuyo algoritmo tiene una complejidad exponencial ($O(c^n)$), se convierte en intratable para pequeños tamaños de la entrada. No ocurre así con los algoritmos que tienen una complejidad lineal ($O(n)$) ó polinómica ($O(n^c)$). En los de complejidad polinómica, puede ocurrir que para tamaños grandes de entrada, se requiera un tiempo considerable [SAEZ87].

11.2.2. COMPROBACION DE LA COMPLEJIDAD

Como ya hemos indicado en la sección 10.1, realizamos diversas ejecuciones del algoritmo midiendo los tiempos. Los resultados han sido los siguientes:

- para n = 10 documentos, t = 22.3 segundos
- para n = 15 documentos, t = 41.3 segundos
- para n = 20 documentos, t = 65 segundos
- para n = 25 documentos, t = 98 segundos
- para n = 30 documentos, t = 148 segundos

Representamos ahora, en la siguiente gráfica la nube de puntos de los valores obtenidos junto con la recta que nos muestra su comportamiento lineal.

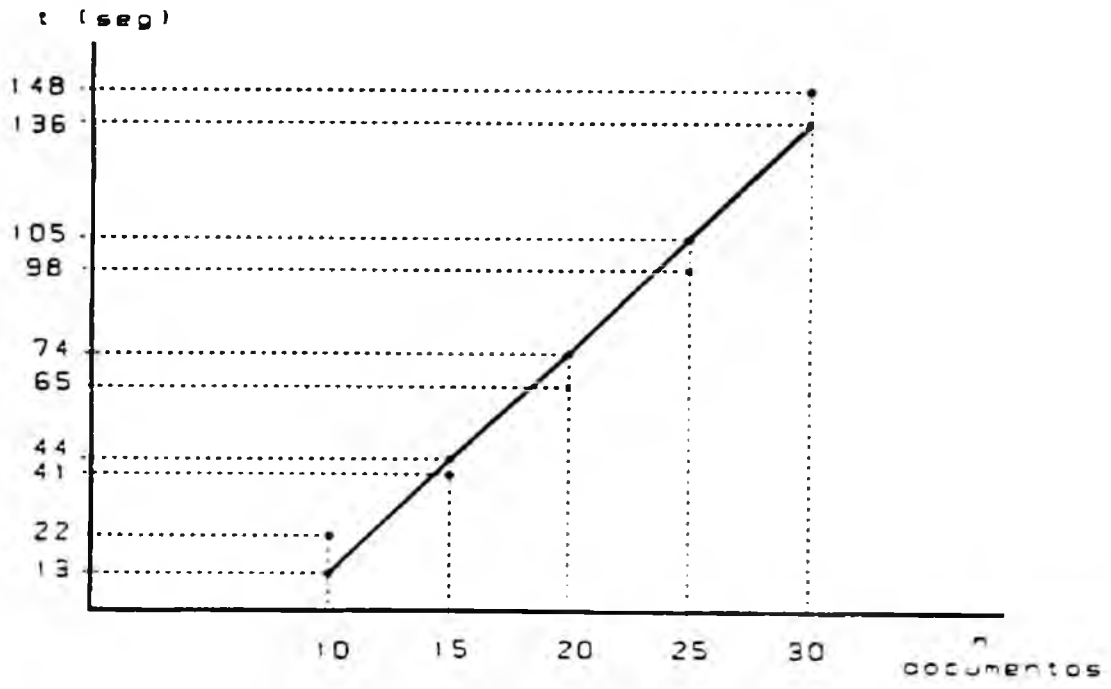


Figura 11.1

La ecuación de esta recta (y = ax + b), puede ser fácilmente calculada mediante el método estadístico de *la recta de regresión de y en x*. Es decir, las técnicas de regresión sirven para obtener la ecuación que nos da el comportamiento ó la tendencia de la nube de puntos. Esta, es la ecuación de una linea llamada linea de ajuste ó linea de regresión, y se debe ajustar lo más posible a la nube de puntos.

Las *técnicas de correlación* sirven para averiguar si el ajuste hallado es bueno ó malo.

Para ello, necesitamos la siguiente tabla:

X_i=n	Y_i=t	X_i*Y_i	X_i^2	Y_i^2
10	22.3	223.0	100	497.29
15	41.3	619.5	225	1705.69
20	65.0	1300.0	400	4225.00
25	98.0	2450.0	625	9604.00
30	148.0	4440.0	900	21904.00
100	374.6	9032.5	2250	37935.98

Figura 11.2

Las fórmulas que nos dan los valores de a y de b son:

$$a = \frac{\frac{\Sigma X_i Y_i}{n} - \frac{\Sigma X_i}{n}\frac{\Sigma Y_i}{n}}{\frac{\Sigma X_i^2}{n} - \frac{(\Sigma X_i)^2}{(n)^2}} = \frac{\frac{9032.5}{5} - \frac{100}{5}\frac{374.6}{5}}{\frac{2250}{5} - \frac{(100)^2}{(5)^2}}$$

operando, obtenemos: a = 6.162

$$b = \frac{\Sigma Y_i}{n} - a\frac{\Sigma X_i}{n} = \frac{374.6}{5} - 6.162\frac{100}{5}$$

operando, obtenemos: b = - 48.32

Por lo tanto, la ecuación de la recta que nos da el comportamiento lineal del tiempo de ejecución del algoritmo es:

$$y = 6.162\ x - 48.32$$

es decir:

$$t = 6.162\ n - 48.32$$

Ahora, mediante las técnicas de correlación, tenemos que comprobar si el ajuste lineal que hemcs realizado, es ó no es un buen ajuste. Para ello, empleamos el *coeficiente de correlación lineal* μ. Si este coeficiente toma valores próximos a "1" ó a "-1", indica que el ajuste realizado es un buen ajuste.

La fórmula para el cálculo de μ es:

$$\mu = \frac{S_{xy}}{S_x\ S_y}$$

siendo: $S_{xy} = \frac{\Sigma\ X_i\ Y_i}{n} - m_x\ m_y$

$$S_x^2 = \frac{\Sigma\ X_i^2}{n} - m_x^2$$

$$S_y^2 = \frac{\Sigma\ Y_i^2}{n} - m_y^2$$

$$m_x = \frac{\Sigma X_i}{n}$$

$$m_y = \frac{\Sigma Y_i}{n}$$

Sustituyendo los valores obtenidos y operando, obtenemos: $S_{xy} = 308.1$, $S_x = 7.071$, $S_y = 44.44$; por lo tanto: $\mu = 0.98$.

Podemos concluir que al ser μ muy cercano a "1", el ajuste lineal realizado es un buen ajuste, por lo que la nube de puntos tiene tendencia lineal.

CAPITULO 12:

CONCLUSIONES

12.1. CONCLUSIONES

En esta tesis, hemos descrito un ***Sistema de Almacenamiento y Recuperación de la Información,*** que proporciona un ambiente efectivo en todas y cada una de las partes que son importantes en el tratamiento de la información en un centro de investigación, es decir: entrada del documento, almacenamiento, indexación, recuperación y salida. Además, hemos diseñado algunas de sus partes, como son el modelo de registro-índice, la normalización de índics y el interface con el usuario.

El trabajo principalmente, ha consistido en el desarrollo e implementación de un algoritmo utilizando la ***teoría del Análisis Cluster,*** para clasificar la información, aumentando según se ha visto, la eficiencia del sistema descrito. Por todo ello, hemos llegado a concluir lo siguiente:

1º Los Sistemas de Almacenamiento y Recuperación de la Información que incorporan la posibilidad de clustering, *facilitan la recuperación* limitando el número de accesos. Esto se debe, a que cada query solo se compara primero con los centroides obtenidos en la clasificación y posteriormente con los documentos que componen la clase más similar a él.

Refiriéndonos en concreto a la Base de Datos de 30 documentos utilizada para experimentar nuestro algoritmo, podemos observar lo siguiente:

a.- Si el Sistema de Almacenamiento y Recuperación de la Información a implantar, no incorporase la facilidad de clustering, el número de comparaciones a efectuar para dar respuesta a cada query introducido, sería igual a 30, es decir, habrían de realizarse tantas comparaciones como documentos almacenados. Cabe destacar que, entre ellas, muchas serían totalmente irrelevantes, puesto que algunos documentos pueden ser completamente diferentes al query.

b.- Sin embargo, si el Sistema de Almacenamiento y Recuperación de la Información a implantar, incorpora el algoritmo de clustering, el número de comparaciones se reduce considerablemente: primero se realiza una comparación entre el query y los

centroides de las clases generadas, y posteriormente, se compara el query con cada uno de los documentos pertenecientes a la clase recuperada como más similar. Observamos que, en nuestro caso concreto, en la primera fase, el número de comparaciones es 5, y en la segunda fase, puede ser 5, 6 ó 7. Por lo tanto, hemos reducido el número de 30 a 10, 11 ó 12, aumentándose así la eficiencia en la recuperación.

Volviendo al ejemplo del query expuesto en la sección 7.2.3. (suponemos ya calculados los pesos de sus términos-índice), podríamos comprobar que con solo 5 comparaciones con los centroides de las clases generadas, el sistema nos daría como respuesta la clase número cuatro (C4) por ser la más similar a él. Posteriormente, el query sería comparado con los documentos de esta clase (D4, D9, D14, D16, D23, D29) para presentar al usuario una salida ordenada.

2º Los Sistemas de Almacenamiento y Recuperación de la Información que incorporan la facilidad de clustering, *requieren más almacenamiento* que los sistemas que no tienen los documentos clasificados en clusters. Y esto es debido precisamente a los ficheros e índices auxiliares que se hacen necesarios para mantener la clasificación. Esta diferencia de memoria requerida, no

es de ninguna forma crucial por cuanto que los sistemas proporcionan suficiente almacenamiento para acomodar estas facilidades.

Esta pequeña desventaja, queda ampliamente compensada por el *aumento en la eficiencia* que proporcionan los sistemas que incorporan algún algoritmo clustering, medida principalmente por:

a.- Disminución del tiempo de respuesta al usuario (al disminuir el número de comparaciones).

b.- Mayor fiabilidad de la respuesta.

3º Por todo esto y a la vista de los resultados, podemos concluir que *el algoritmo propuesto es aceptable* al proporcionarnos una clasificación que es estable, uniforme, independiente de los pequeños errores cometidos en la indexación del documento, y además aumenta la eficiencia de la recuperación.

12.2. LINEAS FUTURAS DE INVESTIGACION

Por lo tanto, partiendo de estos resultados, se podría continuar investigando en las siguientes direcciones:

1º En el área de *indexación del documento*, se podría perfeccionar el algoritmo de indexación para reducir al mínimo el porcentaje de palabras significativas, con vistas a conseguir que los diez índices obtenidos sean con toda certeza los diez más adecuados.

2º En lo referente *al interface del usuario*, se podría desarrollar en programación el algoritmo propuesto de transformación del query en Lenguaje Natural al query en QNF (forma estándar comprensible por el sistema).

3º En cuanto a *la recuperación de la información*, se podría investigar en algún algoritmo de redefinición del query. De tal forma que si el query en QNF obtenido en la transformación desde el lenguaje natural, no proporcionara al usuario resultados lo suficientemente satisfactorios, este pudiera ser sometido a subsiguientes redefiniciones.

4º En el área de *clasificación*, se podría perfeccionar el algoritmo de clustering propuesto con el fin de mejorar su complejidad.

Anexo A

Documentación del Algoritmo

A.1. DATOS Y FICHEROS

Los siguientes ficheros son necesarios para la ejecución del algoritmo propuesto:

1.- **DOC.DAT**: Es el fichero que contiene los vectores **V2** de todos los documentos almacenados en la Base de Datos. Es decir, contiene **N** objetos (documentos) con **M** descriptores (términos-índice) cada uno de ellos. Cada vez que llega un documento nuevo al sistema y se indexa, queda almacenada una entrada correspondiente a él en forma de vector **V2** en este fichero.

2.- **CLASES.DAT**: Es el fichero que se obtiene como resultado de la clasificación. Tiene la misma estructura que el fichero DOC.DAT, pero ahora, aquí, hay una entrada por cada clase formada, almacenando el identificativo de clase más el vector **VC2** (que tiene la misma forma que **V2**) representante del documento imaginario ó centroide de la clase.

3.- **CLASES2.DAT**: Este fichero también se obtiene como resultado de la clasificación. Tiene una entrada por cada clase generada, almacenando cada una de ellas el identificativo de clase más todos los códigos de identificación (la referencia externa **R1**) de todos los documentos que han resultado pertenecientes a ella.

4.- **DISTANCIAS.DAT**: Es un fichero que tiene estructura de matriz, conteniendo cada elemento, la distancia de cada objeto a todos los demás.

5.- **PERTENENCIAS.DAT**: Es un fichero que tiene estructura de matriz, conteniendo cada elemento el grado de pertenencia de cada objeto a cada clase generada en el proceso de clasificación. Este grado de pertenencia viene dado por la diferencia entre el radio de clasificación y la distancia del objeto al centro de la clase.

6.- **VOCAB.DAT**: Es un fichero que contiene todo el vocabulario relacionado con el Sistema de Almacenamiento y Recuperación de la Información en el que se va a implantar el algoritmo de clustering.

A.2. VARIABLES Y LISTAS

1.- **OBJETOS1**: Es una lista de **N** sublistas (una por cada objeto almacenado en la Base de Datos), y cada sublista tiene **M** elementos (uno por cada peso w_i del vector **V2** que representa al documento mediante **M** elementos).

2.- **OBJETOS2**: Es una lista de N elementos (uno por cada objeto almacenado en la Base de Datos). Inicialmente, todos estos elementos tienen valor 0, y posteriormente, pueden tomar diferentes valores:

0: el objeto correspondiente a esa posición no pertenece a ninguna clase.

1: el objeto correspondiente pertenece a una clase.

2: el objeto correspondiente pertenece a dos clases.

.....etc.

3.- CENTROS: Es una lista que contiene tantas sublistas como clases se hayan generado. Cada sublista tiene **M** elementos, siendo estos, las M coordenadas del centro de la clase. Es la lista que posteriormente se volcará en el fichero CLASES.DAT.

4.- **CENTROS2:** Es una lista que contiene tantas sublistas como clases se hayan generado. Cada sublista tiene tantos elementos como objetos pertenezcan a la clase correspondiente. Y cada uno de estos elementos, almacena el identificativo del objeto correspondiente. Es la lista que posteriormente se volcará en el fichero CLASES2.DAT.

5.- **CONT1:** Es un contador de objetos procesados. Contiene en todo momento, el identificativo ó número del objeto en curso.

6.- **CONT2:** Es un contador que contiene en todo momento el número de objetos almacenados en la Base de Datos.

7.- **CONTCEN1:** Es un contador de clases creadas. Se incrementa en una unidad cada vez que se crea una clase nueva.

8.- **CONTCEN2:** Es un contador auxiliar que se inicializa con valor uno cada vez que se comienza a procesar un objeto nuevo. Sirve para tratar con cada objeto todas las clases existentes. Cuando alcanza el valor del contador CONTCEN1, significa que ya no existen más clases creadas.

9.- **RLIST:** Es una lista que contiene a los identificadores de todos los objetos de la colección.

Además de estos campos de trabajo que hemos descrito individualmente por considerarlos más importantes y de más uso, el algoritmo, utiliza otros muchos, que, dada su variedad y poco nivel de uso, no los describimos detalladamente.

A.3. PROCESOS

Debemos distinguir:

1º Procesos a llevar a cabo en la **generación de los clusters:** CREACION, CALCULO_RADIO, CLASIFICACION, ESTABILIDAD y UNIFORMIDAD.

2º Procesos a llevar a cabo en el **mantenimiento de los clusters:** CARGA, ESTABILIDAD y UNIFORMIDAD.

1.- **CREACION:** Es un proceso un tanto ajeno a la clasificación ya que consiste en la creación de los ficheros DOC.DAT y VOCAB.DAT. Es decir, también hemos programado en LISP, *el transbase de V1 a V2*. Podríamos dividirlo en dos subprocesos:

1.a.- Lectura del fichero **INDICE.DAT** y conversión de todos los vectores **V1** (representantes de los documentos almacenados en la Base de Datos, mediante sus diez términos-índice), en los vectores **V2** (representantes de todos los documentos, pero mediante M elementos).

1.b.- Almacenamiento de todos estos vectores **V2** en el fichero de documentos **DOC.DAT**.

1.c.- Creación del fichero **VOCAB.DAT.**

Luego, **CREACION** es un proceso que tiene como fichero de entrada a **INDICE.DAT**, y como ficheros de salida a **DOC.DAT y VOCAB.DAT.**

2.- **CALCULO_RADIO:** Es un proceso cuya finalidad es *calcular un radio inicial* que nos permita posteriormente realizar el primer proceso de clasificación, ya que inicialmente es difícil imaginar cuál puede ser el radio apropiado para definir el tamaño de las clases. Este proceso esta dividido en los siguientes subprocesos:

2.a.- Carga del fichero **DOC.DAT** en la lista de trabajo **OBJETOS1.**

2.b.- Cálculo de todas las distancias entre todos los objetos.

2.c.- Cálculo del radio inicial que vendrá dado por el sumatorio de todas las distancias dividido por el número de objetos.

2.d.- Almacenamiento de las distancias entre los objetos en el fichero **DISTANCIAS.DAT.**

Luego, este proceso, tiene como fichero de entrada, el fichero de documentos **DOC.DAT**, y como fichero de

salida, el fichero que almacena las distancias entre cada dos objetos ó documentos, **DISTANCIAS.DAT**.

3.- **CARGA**: Es un proceso que se realiza con cada documento, esto es, *cada vez que un nuevo documento llega al sistema* y queda indexado mediante el vector **V1** y almacenado en el fichero **INDICE.DAT**. Consiste en los siguientes subprocesos:

3.a.- Lectura del fichero **INDICE.DAT** y conversión del vector **V1** del nuevo documento, en el vector **V2**.

3.b.- Almacenamiento de la entrada correspondiente a este nuevo documento en el fichero **DOC.DAT**.

3.c.- Comparación del vector **V2** del nuevo documento con todos los vectores **VC2** de las clases existentes, e inclusión del mismo en la clase más similar cuyo centro diste del nuevo documento menos que el radio, con el posterior recálculo del centroide.

Luego, como fichero de entrada esta **INDICE.DAT** y como de salida **DOC.DAT**.

4.- CLASIFICACION: ***Es el proceso principal.*** **El algoritmo ideado para su ejecución, va tratando todos los documentos, y va generando tantas clases de similaridad como sean necesarias. Este proceso engloba a varios subprocesos:**

4.a.- Inicialización:

- Lectura del fichero **DOC.DAT** y carga de su contenido en la lista de trabajo **OBJETOS1.**
- Creación de la lista de trabajo **OBJETOS2**, y su inicialización con valor cero.
- Creación de la lista de trabajo **CENTROS2**, y su inicialización con valor cero.
- Inicialización de los contadores **CONT1** y **CONTCEN1** con valor uno.

4.b.- Ejecución de la clasificación:

El primer objeto que se procesa, pasa a formar un cluster por si solo, es decir, es el mismo, el centroide.

Cada subsiguiente objeto que se procesa, se compara con todos los clusters ya existentes (está claro que la primera vez, solo existirá uno), situándose en el primer cluster encontrado lo suficientemente similar.

Si el objeto en proceso, no es lo suficientemente similar a ningún cluster ya existente, formará un nuevo cluster por si solo.

Para comprobar si un objeto es lo "suficientemente similar" a un cluster, utilizamos las distancias euclidianas calculadas entre los vectores V2 de representación de los documentos, y los vectores VC2 de representación de los clusters, de tal forma que, se aceptará un objeto como perteneciente a la clase, si ésta distancia euclidiana calculada es menor a un radio dado.

En la primera ejecución del algoritmo, se utilizará el radio calculado en el proceso CALCULO_RADIO, y posteriormente, si no estamos satisfechos con los resultados obtenidos, se puede repetir el proceso variando el radio a nuestro gusto.

Cada vez que un objeto nuevo entra a formar parte de una clase ya existente, el centroide de esta ha de ser recalculado, lo que puede producir que algunos otros objetos hayan de salir de la clase por presentar ahora una distancia superior al radio establecido.

Como resultado de cada proceso de clasificación, puede ocurrir que haya solapamiento de clases, esto significa, que alguno ó algunos documentos pertenezcan a más de una clase. Esta información aparece indicada en la lista de trabajo OBJETOS2, permitiéndonos repetir la clasificación con un radio menor para disminuir el número de solapamientos.

5.- **ESTABILIDAD:** Es un proceso especialmente diseñado para proporcionar una *clasificación estable.* Esto significa que, sea cual sea el orden de proceso de los documentos de la Base de Datos, la clasificación obtenida es la misma. Este proceso, se subdivide en los siguientes subprocesos:

5.a.- Comparación de cada documento con todos los centros existentes. Si existe algún centro más cercano a él, que el centro de su propia clase, se producirá la salida del documento de la clase antigua, y la integración del mismo en la nueva.

5.b.- Recálculo de los dos centros: el de la clase antigua, y el de la clase nueva.

5.c.- Ambos subprocesos se ejecutan iterativamente, hasta que ningún objeto salga de su propia clase.

6.- **UNIFORMIDAD:** Es un proceso especialmente diseñado para preservar *la distribución uniforme de los documentos en las clases.* Es decir, para mantener un tamaño similar de las clase. Se compone de los siguientes subprocesos:

6.a.- Comprobación para cada clase de si su tamaño es superior al deseado.

6.b.- División de la clase cuyo tamaño supere al deseado en dos subclases, e inclusión de los documentos de la clase antigua en la subclase más similar.

6.c.- Ambos subprocesos se ejecutan iterativamente hasta que se compruebe que todas las clases son de tamaño inferior ó igual al deseado.

6.d.- Obtención de ficheros: Los resultados de la clasificación, quedan almacenados en tres ficheros:

- Fichero **CLASES.DAT**: Es el fichero de clases. Tiene una entrada por cada clase generada, con las **M** coordenadas del centroide representante de la clase (vector **V2** de representación).

- Fichero **CLASES2.DAT**: Es un fichero de clases que contiene información sobre los documentos pertenecientes a cada clase. Es decir, cada entrada contiene los códigos (**R1**) de identificación de los documentos que componen la clase.

- Fichero **PERTENENCIAS.DAT**: En este fichero quedan almacenadas las pertenencias, que se

calculan al final de la clasificación y nos dan, el grado de pertenencia de cada documento a cada una de las clases generadas.

En este proceso no hay ficheros de entrada, mientras que en salida, tenemos: **CLASES.DAT**, **CLASES2.DAT** y **PERTENENCIAS.DAT**.

Anexo B

Código LISP del Algoritmo

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso de CREACION sirve, como su nombre indica  ;;;;
;;;; para crear los ficheros de datos: INDICE.DAT, DOC.DAT;;;;
;;;; y VOCAB.DAT, que posteriormente se utilizarán como   ;;;;
;;;; entrada en los procesos de clasificación             ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;; La función CREACION es la principal del proceso, es decir,
;;;sirve para conectar todas las demás que componen el proceso:
;;;FICH, FICH2 y LEER.

(DEFUN CREACION ()
   (FICH)
   (FICH2)
   (FICH3))
```

```
;;; Esta función, crea el fichero INDICE.DAT, donde cada
;;;registro es una lista de sublistas, y cada campo es una de
;;;estas sublistas.

(DEFUN FICH ()
   (SETF INDICEP (OPEN "INDICE.DAT" :DIRECTION :OUTPUT)
         FIN 'X)
   (DO ()
       ((EQUAL FIN 'FF))
       (DO ((CONT 0 (1+ CONT)))(LISREG '()) (CONS VALOR LISREG)))
           ((> CONT 6) (SETF LISTREG (REVERSE LISREG)))
           (COND ((NOT (EQUAL CONT 5))
                  (FORMAT T "~%INTRODUCIR EL VALOR DE R'A  " CONT))
                 (T (FORMAT T "~%INTRODUCIR EL VALOR DEL VECTOR  ")))
           (SETF VALOR (READ)))
       (FORMAT INDICEP "~A ~%" LISTREG)
       (FORMAT T "~%QUIERE SEGUIR METIENDO DATOS (S/N)  ")
       (SETF CONTES (READ))
       (COND ((EQUAL CONTES 'S)(SETF FIN 'X))
             (T  (SETF FIN 'FF))))
   (CLOSE INDICEP))
```

```
;;; Esta función crea el fichero VOCAB.DAT que contiene el
;;;vocabulario completo de todas las palabras relacionadas con
;;;nuestro centro de investigación.

(DEFUN FICH2 ()
   (SETF VOCABP (OPEN "VOCAB.DAT" :DIRECTION :OUTPUT))
   (DO ((NUM 1 (1+ NUM)))
       ((> NUM 50))
       (FORMAT T "INTRODUZCA LOS TEMAS ")
       (SETF TEMA (READ))
       (FORMAT VOCABP "~A~%" TEMA))
   (CLOSE VOCABP))
```

```
;;; Esta función crea el fichero DOC.DAT a partir de INDICE.DAT
;;;y de VOCAB.DAT, para ello utiliza las funciones CARVOCAB,
;;;ENFRENT y CREDOC.

(DEFUN FICH3 ()
   (CARVOCAB)
   (SETF INDICEP (OPEN "INDICE.DAT" :DIRECTION :INPUT)
         DOCP (OPEN "DOC.DAT" :DIRECTION :OUTPUT)
         REGLIS (READ INDICEP NIL 'EOF))
   (DO ()
       ((EQUAL REGLIS 'EOF))
       (SETF N1 (CADR REGLIS)
             LISTA (SIXTH REGLIS))
       (ENFRENT TEMAS LISTA)
       (CREDOC N1 ULTLIST)
       (SETF REGLIS (READ INDICEP NIL 'EOF)))
   (CLOSE DOCP)
   (CLOSE INDICEP))



;;; Esta función vuelca el fichero VOCAB.DAT en una lista de
;;;trabajo.

(DEFUN CARVOCAB ()
   (SETF VOCABP (OPEN "VOCAB.DAT" :DIRECTION :INPUT)
         LISTA3 ())
   (DO ((NUM 1 (1+ NUM)))
       ((> NUM 50)(SETF TEMAS (REVERSE LISTA3)))
       (SETF ELEM (READ VOCABP)
             LISTA3 (CONS ELEM LISTA3)))
   (CLOSE VOCABP))



;;; Función que enfrenta los ficheros INDICE.DAT y VOCAB.DAT
;;;para obtener el fichero DOC.DAT. LLama ala función FUN.

(DEFUN ENFRENT (TEMAS LISTA)
    (SETF *PRINT-LEVEL* 120
          *PRINT-LENGTH* 120
          ULTLIST '()
          LISTA5 '())
    (DO ((LISTA2 TEMAS (CDR LISTA2)))
        ((NULL LISTA2))
        (SETF X 0)
        (DO ((LISTA3 LISTA (CDR LISTA3)))
            ((OR (NULL LISTA3)(= X 1)) (FUN1 X LISTA2 LISTA3))
            (SETF ELEM (CAR LISTA3))
            (IF (EQUAL ELEM (CAR LISTA2)) (SETF X 1))))
    (SETF ULTLIST (REVERSE LISTA5)))



;;; Esta función sirve para comprobar la función de salida del
bucle Do de la función anterior.

(DEFUN FUN1 (X LISTA2 LISTA3)
   (SETF NOM (CAR LISTA2)
         LISTA5 (CONS NOM LISTA5)
         NUM (CAR LISTA3))
   (IF (= X 1) (SETF LISTA5 (CONS NUM LISTA5))
               (SETF LISTA5 (CONS '0 LISTA5))))
```

;;; Esta función copia un registro en el fichero DOC.DAT

```
(DEFUN CREDOC (R1 ULTLIST)
   (SETF LISDOC '()
         LISDOC (CONS R1 LISDOC)
         LISDOC (CONS ULTLIST LISDOC)
         LISDOC (REVERSE LISDOC))
   (FORMAT DOCF " ~A ~% " LISDOC))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso CALCULO_RADIO, calcula un radio inicial  ;;;;
;;;;que sirva para realizar una primera clasificación con;;;;
;;;;un radio inicial apropiado. Posteriormente se puede  ;;;;
;;;;clasificación variando el radio a gusto.             ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;; Esta función, es la principal de este segundo proceso.
;;;sirve para conectar las funciones que componen el proceso.
;;;LLama a las funciones LIMPIAR, CALCULAR y GRABAR.
```

```
(DEFUN CALCULO_RADIO ()
   (SETF *PRINT-LEVEL* NIL
         *PRINT-LENGTH* NIL)
   (LIMPIAR)
   (CALCULAR)
   (CALDISP OBJV OBJETOSI RLIST))
```

```
;;; Esta función, sirve para cargar el contenido del fichero
;;;DOC.DAT en las listas de trabajo OBJETOSI y RLIST
```

```
(DEFUN CALCULAR ()
   (SETF DOCP (OPEN "DOC.DAT" :DIRECTION :INPUT)
         LECT (READ DOCP NIL  EOF)
         RLIST '()
         OBJV 0)
   (DO ((LIST () (CONS LIST2 LIST)))
       ((EQUAL LECT   EOF) (SETF OBJETOSI (REVERSE LIST)))
       (SETF RILIST (CAR LECT)
             RLIST (CONS RILIST RLIST)
             LISTA (CADR LECT)
             OBJV (1+ OBJV))
       (DO ((CONT 1 (1+ CONT))
            (LISTA1 () (CONS ELEM LISTA1))
            (LISTA2 (CDR LISTA) (CDDR LISTA2)))
           ((> CONT 50) (SETF LIST2 (REVERSE LISTA1)))
           (SETF ELEM (CAR LISTA2)))
       (SETF LECT (READ DOCP NIL 'EOF)))
       (SETF RLIST (REVERSE RLIST))
    (CLOSE DOCP))
```

```
;;; Esta función sirve para crear la matriz de distancias entre
;;;objetos, y para grabarla en el fichero DISTANCIAS.DAT. Para
;;;ello llama a la función DISTEUCLI.

(DEFUN CALDIST (OBJ OBJETOS1)
   (SETF R 0
         LISTA1 OBJETOS1
         LISTA2 OBJETOS1
         DISTANP (OPEN "DISTANCIAS.DAT" :DIRECTION :OUTPUT))
   (DO ((NUM1 1 (1+ NUM1)))
       ((> NUM1 OBJ) (FORMAT DISTANP "~%"))
       (DO ((NUM2 1 (1+ NUM2)))
           ((> NUM2 OBJ) (FORMAT DISTANP "~%"))
           (SETF DISTANCIA (DISTEUCLI (ACCESO NUM2 LISTA2)
                                      (ACCESO NUM1 LISTA1)))
           (FORMAT DISTANP " ~A " DISTANCIA)
           (COND ((OR (< NUM1 NUM2) (EQUAL NUM1 NUM2))
                                    (SETF R (+ R DISTANCIA))))))
   (SETF R (/ R OBJ))
   (FORMAT T "EL RADIO MAS ADECUADO ES :~A " R)
   (FORMAT DISTANP " %")
   (FORMAT DISTANP "~% FINAL DE FICHERO")
   (CLOSE DISTANP)
   (FORMAT T "~%PULSE UNA TECLA PARA CONTINUAR")
   (SETF A (READ)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso CLASIFICACION es el proceso principal del;;;;
;;;;algoritmo. Procesa a todos los objetos de la colección;;;
;;;;y obtiene las clases de similaridad necesarias;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;; La función principal del proceso, CLASIFICACION, sirve para
;;;conectar a las demás funciones componentes del proceso:
;;;LIMPIAR, CALCULAR y CLASIF.

(DEFUN CLASIFICACION ()
   (DECLARE (SPECIAL CONTCENT2 CONTCENT1 OBJETOS1
                     OBJETOS2 CENTROS CENTROS2 RLIST))
   (SETF *PRINT-LEVEL* 120
         *PRINT-LENGTH* 120)
   (LIMPIAR)
   (CALCULAR)
   (CLASIF))
```

```
;;; Esta es la función principal del proceso de clasificación.
;;;Conecta dos funciones básicas: PROYECTO y BUCLE.

(DEFUN CLASIF ()
  (DECLARE (SPECIAL CONTCENT2 CONTCENT1 OBJETOS1
                    OBJETOS2 CENTROS CENTROS2 RLIST))
  (PROYECTO)
  (BUCLE))
```

```
;;;  Función que inicializa las variables y listas, y
;;;posteriormente arranca el bucle iterativo que se repite para
;;;cada objeto de la colección. LLama a la función PRINCIPA.

(DEFUN PROYECTO ()
  (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RDO RLIST
                    OBJETOS1 OBJETOS2 CENTROS CENTROS2))
  (SETF OBJETOS2 ()
        CENTROS2 ()
        CENTROS ()
        CONTCENT1 1
        DISTANCIA 0)
  (DO ((N 1 (1+ N)))
      ((= N (1+ (LENGTH OBJETOS1))) NIL)
      (SETF OBJETOS2 (APPEND (LIST 0) OBJETOS2)))
  (PRINCIPA))
```

```
;;; Función que para cada objeto, arranca un bucle iterativo
;;;que se repite por cada clase ya creada. Llama a la
;;;función FUNCION. Al finalizar el proceso, llama a las
;;;funciones ESTAB y UNIF, que son procesos que deben
;;;ejecutarse inmediatamente después de la clasificación para
;;;asegurar que la clasificación obtenida sea estable y
;;;uniforme. Posteriormente, llama a las funciones SALIDA1,
;;;SALIDA2 y SALIDA3 para escribir los ficheros de datos.

(DEFUN PRINCIPA ()
  (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RET RDO RLIST
                    OBJETOS1 OBJETOS2 CENTROS CENTROS2 RADIO))
  (TERPRI)
  (FORMAT T "TECLEE EL RADIO: ")
  (SETF RADIO (READ))
  (TERPRI)
  (SETF CENTROS  (METER2 1 CENTROS (ACCESO 1 OBJETOS1))
        CENTROS2 (APPEND (LIST (ACCESO 1 RLIST)) CENTROS2)
        OBJETOS2 (METER2 1 OBJETOS2  1))
  (DO ((LISTA-ANTIGUA (CDR OBJETOS1) (CDR LISTA-ANTIGUA))
       (CONTI 2 (+ 1 CONTI)))
      ((NULL LISTA-ANTIGUA) LAMBDA ())
      (TERPRI)
      (ESTAB)
      (UNIF)
      (SETF CLAS (LENGTH CENTROS1))
      (SALIDA1)
      (SALIDA2)
      (SALIDA3)
      (SETF PARAMP (OPEN "PARAM.DAT" :DIRECTION: OUTPUT))
      (FORMAT PARAMP "~A ~%  A ~% A ~% A ~% OBJETOS2 CENTROS
                                           CENTROS2 RADIO)
      (CLOSE PARAMP))
  (FUNCION LISTA-ANTIGUA RADIO CONTI)))



;;; Esta función llama a FUNCION1 y FUNCION2, que representan
;;;las diferentes alternativas a ejecutar según haya que crear
;;;una clase nueva ó no.

(DEFUN FUNCION (LISTA-ANTIGUA RADIO CONT1)
   (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RET RDO JBJETOS1
                     OBJETOS2 CENTROS CENTROS2 RLIST))
  (SETF CONT2 1)
  (SETF RET 0)
  (SETF CONTCENT2 1)
  (DO ()
      ((= RET 1) NIL)
      (COND ((> CONTCENT2 CONTCENT1)(FUNCION1 CONT1 LISTA-ANTIGUA))
            (T (FUNCION2  LISTA-ANTIGUA CONT1 RADIO)))
      (SETF CONTCENT2 (1+ CONTCENT2)))))
```

```
;;; Esta función se ejecuta en el caso de que el objeto en
;;;curso no pertenezca a ninguna clase, creándose por tanto una
;;;nueva clase.

(DEFUN FUNCION1 (CONT1 LISTA-ANTIGUA)
   (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RET RDO
                     OBJETOS1 OBJETOS2 CENTROS CENTROS2 RLIST))
   (COND ((= 0 (ACCESO CONT1 OBJETOS2))
          (SETF CENTROS (APPEND CENTROS (LIST(CAR LISTA-ANTIGUA)))
                CONTCENT1 (1+ CONTCENT1)
                CENTROS2 (METER2 CONT1 CENTROS2 (ACCESO CONT1 RLIST))
                CENTROS2 (DELETE NIL CENTROS2)
                OBJETOS2 (METER2 CONT1 OBJETOS2
                                 (1+ (ACCESO CONT1 OBJETOS2)))
                RET 1))
         (T (SETF RET 1))))



;;; Esta función se ejecuta en el caso de que no haya que crear
;;;clase nueva. Comprueba si el objeto en curso pertenece a la
;;;clase en curso (llamando en este caso a la función
;;;FUNCIONE3) ó no.

(DEFUN FUNCION2 (LISTA-ANTIGUA CONT1 RADIO)
   (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RDO OBJETOS1
                     OBJETOS2 CENTROS CENTROS2 RLIST ACUM))
   (SETF DISTANCIA (DISTEUCLI (CAR LISTA-ANTIGUA)
                              (ACCESO CONTCENT2 CENTROS)))
   (COND ((> DISTANCIA RADIO) NIL)
         (T (FUNCION1 CONT1))))



;;; Función que sirve para añadir el objeto en curso en la
;;;clase en curso con el correspondiente recálculo del
;;;centroide de esta.

(DEFUN FUNCION3 (CONT1)
   (DECLARE (SPECIAL CONTCENT2 CONTCENT1 RDO OBJETOS1
                     OBJETOS2 CENTROS CENTROS2 RLIST ACUM))
   (SETF OBJETOS2 (METER2 CONT1 OBJETOS2
                  (1+ (ACCESO CONT1 OBJETOS2))))
   (SETF CENTROS2 (METER2 CONTCENT2 CENTROS2
                  (METER2 (+ 1 (LENGTH (ACCESO CONTCENT2 CENTROS2)))
                  (ACCESO CONTCENT2 CENTROS2)
                  (CAR (ACCESO CONT1 RLIST)))))
   (SETF CENTROS (METER2 CONTCENT2 CENTROS
                 (RECALCULAR (ACCESO CONTCENT2 CENTROS2) OBJETOS1)))
   (SETF PIL 1)
   (DO ()
       ((= PIL 0) NIL)
       (SETF PIL 0)
       (SETF RESU '())
       (DO ((LISTA (ACCESO CONTCENT2 CENTROS2) (CDR LISTA)))
           ((NULL LISTA) NIL)
           (SETF VAR (LIST (CAR LISTA))
                 CONT 0)
           (DO ((V RLIST (CDR V)))
               ((EQUAL VAR (CAR V))(SETF CONT (1+ CONT)))
               (SETF CONT (1+ CONT)))
           (SETF DIST (DISTEUCLI (ACCESO CONT OBJETOS1)
                                 (ACCESO CONTCENT2 CENTROS)))
           (COND ((> DIST RADIO) (SETF OBJETOS2 (METER2  CONT1 OBJETOS2
                                   (1- (ACCESO CONT1 OBJETOS2)))
                                 PIL 1))
```

```
          (T (SETF RESU (CONS (CAR LISTA) RESU)))))
(SETF CENTROS2 (METER2 CONTCENT2 CENTROS2 (REVERSE RESU)))
(COND ((= PIL 1) NIL)
      (T (SETF CENTROS (METER2 CONTCENT2 CENTROS
         (RECALCULAR (ACCESO CONTCENT2 CENTROS2) OBJETOS1)))))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso ESTABILIDAD ha sido especialmente creado ;;;;
;;;;para obtener una clasificación estable. Es decir, ga-;;;;
;;;;rantiza para cualquier orden de proceso de los objetos;;;
;;;;de la colección, las clases obtenidas son las mismas ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;; La función ESTAB arranca el proceso de ESTABILIDAD. Para
;;;cada objeto de la colección, comprueba si existe algún otro
;;;centro más similar a él. LLama a la función EJESTAB y a
;;;DISTEUCLI.
```

```
(DEFUN ESTAB ()
   (DECLARE (SPECIAL CENTROS CENTROS2 OBJETOS1
                     RLIST PIL CONTC CONTA CONTI IDCUR ))
   (SETF PIL '1)
   (DO ()
       ((EQUAL PIL 0))
       (SETF PIL '0)
       (SETF CONTA '1)
       (DO ((LISAUX CENTROS2 (CDR LISAUX)))
           ((OR (NULL LISAUX) (EQUAL PIL 1)))
           (SETF LISIDCUR (CAR LISAUX))
           (SETF CONTI '1)
           (DO ((LISIDAUX LISIDCUR (CDR LISIDAUX)))
               ((OR (NULL LISIDAUX) (EQUAL PIL 1)))
               (SETF IDCUR (CAR LISIDAUX))
               (SETF CONT '0)
               (DO ((V RLIST (CDR V)))
                   ((EQUAL (LIST IDCUR) (CAR V)) (SETF CONT (1+ CONT)))
                   (SETF CONT (1+ CONT)))
               (SETF DISPROP (DISTEUCLI (ACCESO CONTA CENTROS)
                                        (ACCESO CONT OBJETOS1)))
               (SETF CONTC '1)
               (DO ((CENAUX CENTROS (CDR CENAUX)))
                   ((OR (NULL CENAUX) (EQUAL PIL 1)))
                   (SETF CENCUR (CAR CENAUX))
                   (SETF DISTAN (DISTEUCLI (ACCESO CONT OBJETOS1) CENCUR))
                   (COND ((< DISTAN DISPROP) (EJESTAB)))
                   (SETF CONTC (1+ CONTC)))
               (SETF CONTI (1+ CONTI)))
            (SETF CONTA (1+ CONTA)))))
```

```
;;; Esta función, se ejecuta en el caso de que el objeto en
;;;curso pertenezca a otra clase, es decir, sea más similar a
;;;otra clase que a la que inicialmente había sido asignado.
;;;LLama a las funciones ANAID, SACID y RECAL2.
```

```
(DEFUN EJESTAB ()
   (DECLARE (SPECIAL IDCUR CENTROS CENTROS2 OBJETOS1
                     RLIST CONTA CONTC CONTI CEN2AUX2 CEN2AUX1))
   (SETF PIL '1)
   (SETF CEN2AUX2 '())
   (SETF CEN2AUX1 CENTROS2)
   (DO ((CONTAUX 1 (1+ CONTAUX)))
       ((NULL CEN2AUX1))
       (COND ((EQUAL CONTAUX CONTC) (ANAID))
             (T (COND ((EQUAL CONTAUX CONTA) (SACID))
                      (T (SETF CEN2AUX2 (CONS (CAR CEN2AUX1) CEN2AUX2))))))
       (SETF CEN2AUX1 (CDR CEN2AUX1)))
   (SETF CENTROS2 (REVERSE CEN2AUX2))
   (RECAL2))
```

```
;;; Esta función saca el objeto en curso de la clase antigua,
;;;es decir, de la clase a que había sido asignado
;;;anteriormente.

(DEFUN SACID ()
   (DECLARE (SPECIAL CEN2AUX1 CONTI CONTA CONTC CEN2AUX2))
   (DO ((CONTS 1 (1+ CONTS))
        (CEN2AUX3 (CAR CEN2AUX1) (CDR CEN2AUX3)))
       ((NULL CEN2AUX3))
       (COND ((EQUAL CONTS CONTI))
             (T (SETF CEN2AUX2 (CONS (CAR CEN2AUX3) CEN2AUX2)))))
   (SETF CEN2AUX2 (LIST CEN2AUX2)))



;;; Esta función introduce el objeto en curso en una lista
;;;auxiliar.

(DEFUN ANAID ()
   (DECLARE (SPECIAL CEN2AUX2 IDCUR CEN2AUX1))
   (SETF PPIL '0)
   (DO ((LISCE (CAR CEN2AUX1) (CDR LISCE)))
       ((NULL LISCE))
       (COND ((EQUAL IDCUR (CAR LISCE)) (SETF PPIL '1))))
   (COND ((EQUAL PPIL 0) (SETF CEN2AUX4 (CONS IDCUR (CAR CEN2AUX1))
                               CEN2AUX2 (CONS CEN2AUX4 CEN2AUX2)))
         (T (SETF CEN2AUX2 (CONS (CAR CEN2AUXI) CEN2AUX2)
                  OBJETOS2 (METER2 IDCUR OBJETOS2
                                   (1- (ACCESO IDCUR OBJETOS2)))))))



;;; Esta función sirve para recalcular dos centros: el de la
;;;clase antigua (es decir, de la que sale el objeto), y el de
;;;la clase nueva (es decir, a la que entra el objeto en
;;;curso).

(DEFUN RECAL2 ()
   (DECLARE (SPECIAL CENTROS2 OBJETOS1 CONTA CONTC CONTI CENTROS ))
   (SETF CENA (RECALCULAR (ACCESO CONTA CENTROS2) OBJETOS1))
   (SETF CENC (RECALCULAR (ACCESO CONTC CENTROS2) OBJETOSI))
   (SETF RECA '())
   (DO ((CENREC CENTROS (CDR CENREC))
        (CONTREC 1 (1+ CONTREC)))
       ((NULL CENREC))
       (COND ((EQUAL CONTREC CONTA) (SETF RECA (CONS CENA RECA)))
             (T (COND ((EQUAL CONTREC CONTC) (SETF RECA (CONS CENC RECA)))
                      (T (SETF RECA (CONS (CAR CENREC) RECA))))))))
   (SETF CENTROS (REVERSE RECA)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso UNIFORMIDAD ha sido especialmente diseñado;;;
;;;;para obtener una clasificación uniforme, es decir, que;;;
;;;;la distribución de los objetos en las clases sea uni-;;;;
;;;;forme.                                              ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;; La función UNIF arranca el proceso de UNIFORMIDAD. Solicita
;;;un tamaño umbral y comprueba si el tamaño de cada clase es
;;;superior al umbral. LLama a las funciones LIMPIAR y
;;;CONVERTIR.

(DEFUN UNIF ()
   (LIMPIAR)
   (DECLARE (SPECIAL LISDO OBJETOS1 OBJETOS2
                            CENTROS CENTROS2 RLIST NUEVOBJ CLISTA))
   (SETF PIL 1)
   (FORMAT T "INTRODUZCA EL UMBRAL : ")
   (SETF UMBRAL (READ))
   (DO ()
       ((EQUAL PIL 0) NIL)
       (SETF CEN  '()
             CEN2 '())
       (DO ((LISTA CENTROS2 (CDR LISTA))
             (LCENTRO CENTROS (CDR LCENTRO)))
            ((NULL LISTA) NIL)
            (SETF RLISCEN1 '()
                  RLISCEN2 '()
                  LISCEN1 '()
                  LISCEN2 '()
                  CLISTA (CAR LISTA)
                  CLCENTRO (CAR LCENTRO)
                  LON (LENGTH CLISTA))
            (COND ((> LON UMBRAL) (CONVERTIR CLISTA OBJETOS1))
                  (T (SETF CEN (CONS CLCENTRO CEN)
                           CEN2 (CONS CLISTA CEN2))))))
       (SETF PIL 0)
       (DO ((LISCEN2 CEN2 (CDR LISCEN2)))
            ((OR (NULL LISCEN2) (EQUAL PIL 1)))
            (SETF CLISCEN2 (CAR LISCEN2)
                  LON (LENGTH CLISCEN2))
            (COND ((> LON UMBRAL) (SETF PIL 1))
                  (T (SETF PIL 0))))
       (SETF CEN2 (REVERSE CEN2)
             CENTROS2 CEN2))
       (SETF CEN (REVERSE CEN)
             CENTROS CEN
             CENTROS2 (REVERSE CENTROS2)))
```

```
;;; Esta función se utiliza para acceder a las coordenadas de
;;;los objetos que pertenecen a la clase que hay que
;;;subdividir. LLama a la función SUBDIV.

(DEFUN CONVERTIR (CENTRO OBJETO)
   (DECLARE (SPECIAL LISDO OBJETOS1 OBJETOS2 CENTROS
                     CENTROS2 RLIST NUEVOBJ CLISTA))
   (SETF NUEVOBJ '())
   (DO ((LISTA CENTRO (CDR LISTA)))
       ((NULL LISTA) NIL)
       (SETF VAR (LIST (CAR LISTA))
```

```
          CONT 0)
      (DO ((V RLIST (CDR V)))
          ((EQUAL VAR (CAR V)) (SETF CONT (1+ CONT)))
          (SETF CONT (1+ CONT)))
      (SETF NUEVOBJ (CONS (ACCESO CONT OBJETOS1) NUEVOBJ)))
  (SETF NUEVOBJ (REVERSE NUEVOBJ))
  (SUBDIV))
```

;;; Esta función divide la clase en dos subclases, siendo sus
;;;centros iniciales, los objetos más disimilares. LLAma a las
;;;funciones DISTEUCLI y ESCOGER.

```
(DEFUN SUBDIV ()
  (DECLARE (SPECIAL LISDO OBJETOS1 OBJETOS2 CENTROS
                    CENTROS2 RLIST NUEVOBJ CLISTA))
  (SETF LONG (LENGTH NUEVOBJ)
        LONGCUA (* LONG LONG)
        LISDIST '())
  (DO ((N 1 (1+ N)))
      ((> N LONG))
      (DO ((M 1 (1+ M)))
          ((> M LONG))
          (SETF DIST (DISTEUCLI (ACCESO N NUEVOBJ)
                                (ACCESO M NUEVOBJ)))
          (SETF LISDIST (CONS DIST LISDIST))))
  (SETF LISDIST (REVERSE LISDIST)
        MAYOR 0)
  (DO ((LISDODIST LISDIST (CDR LISDODIST))
       (CONT 0 (1+ CONT))
       (J 1 (1+ J)))
      ((> J LONGCUA))
      (COND ((> (CAR LISDODIST) MAYOR) (SETF MAYOR (CAR LISDODIST)
                                             CONTA CONT))))
  (SETF DIVI (/ CONTA LONG))
  (SETF POS1 (1+ (TRUNCATE DIVI)))
  (SETF POS2 (1+ (REM CONTA LONG)))
  (SETF VALOR1 (ACCESO POS1 NUEVOBJ)
        VALOR2 (ACCESO POS2 NUEVOBJ)
        RLISCEN1 (CONS (ACCESO POS1 CLISTA) RLISCEN1)
        RLISCEN2 (CONS (ACCESO POS2 CLISTA) RLISCEN2))
  (SETF LISDO NUEVOBJ
        RLISDO CLISTA)
  (DO ((CONT 1 (1+ CONT)))
      ((NULL LISDO))
      (COND ((OR (EQUAL POS1 CONT) (EQUAL POS2 CONT)))
            (T (ESCOGER)))
      (SETF LISDO (CDR LISDO)
            RLISDO (CDR RLISDO)))
  (SETF CEN2 (CONS RLISCEN2 CEN2)
        CEN2 (CONS RLISCEN1 CEN2)
        CEN2 (REVERSE CEN2)
        CEN (CONS (RECALCULAR RLISCEN1 OBJETOS1) CEN)
        CEN (CONS (RECALCULAR RLISCEN2 OBJETOS1) CEN)))
```

```
;;; Esta función, introduce cada objeto de la clase antigua en
;;;su subclase más adecuada. LLama a la función DISTEUCLI.

(DEFUN ESCOGER ()
   (DECLARE (SPECIAL LISDO OBJETOS1 OBJETOS2 CENTROS
                     CENTROS2 RLIST NUEVOBJ CLISTA))
   (SETF DIST1 (DISTEUCLI (CAR LISDO) VALOR1)
         DIST2 (DISTEUCLI (CAR LISDO) VALOR2))
   (COND ((> DIST1 DIST2) (SETF RLISCEN2 (CONS (CAR RLISDO) RLISCEN2)))
         (T (SETF RLISCEN1 (CONS (CAR RLISDO) RLISCEN1))))
   (SETF RLISCEN2 (REVERSE RLISCEN2)
         RLISCEN1 (REVERSE RLISCEN1)))



;;; Función que se ejecuta una vez que se ha finalizado la
;;;clasificación y se han ofrecido los resultados al usuario.
;;;Ofrece la posibilidad de repetir la clasificación variando
;;;el radio, para obtener otras clasificacionas.

(DEFUN BUCLE ()
  (DECLARE (SPECIAL OBJETOS1 OBJETOS2 CENTROS CENTROS2 RLIST))
  (FORMAT T "~% DESEA REPETIR EL PROCESO(S/N)")
  (SETF OPC (READ))
  (COND ((EQUAL OPC 'S) (PROYECTO))
        ((EQUAL OPC 'N) NIL)
        (T (BUCLE))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Funciones auxiliares: las siguientes funciones son  ;;;;
;;;;necesarias en en varias ocasiones, es decir, son lla-;;;;
;;;;madas desde diferentes funciones.                     ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

;;; Esta función sirve para cambiar el elemento número N de la ;;;lista. LLama a las funciones ACCESO y METER2.

```
(DEFUN CAMBIAR (N ELEM LISTA)
   (SETF AUX (ACCESO N LISTA)
         AUX  (APPEND AUX  (LIST ELEM)))
   (METER2 N LISTA   AUX))
```

;;; Esta función sirve para acceder al elemento N de la lista.

```
(DEFUN ACCESO (N LISTA)
   (CAR (NTHCDR (1- N) LISTA)))
```

;;; Esta función sirve para intoducir el elemento ELEM en la ;;;posición número N de la lista.

```
(DEFUN METER2 (N LISTA ELEM)
   "Esta funcion sirve para introducir el elemento ELEM
    en la posicion N de la lista LISTA"
   (DECLARE (SPECIAL RDO))
   (SETF RDO '())
   (DO ((CONT 1 (1+ CONT))
        (LISTA-ANTIGUA LISTA (CDR LISTA-ANTIGUA)))
        ((AND (NULL LISTA-ANTIGUA)(>= CONT (1+ N))) RDO)
        (COND ((= CONT N ) (SETF RDO (APPEND RDO (LIST ELEM))))
              (T (SETF RDO (APPEND RDO (LIST (CAR LISTA-ANTIGUA))))))))
```

;;; Función que calcula la distancia euclidiana entre dos ;;;objetos. LLama a la función FUN

```
(DEFUN DISTEUCLI (LIST1 LIST2)
  (SETF ACUM 0)
  (DO ((LISTA1 LIST1 (CDR LISTA1))
       (LISTA2 LIST2 (CDR LISTA2)))
      ((NULL LISTA1) ACUM)
      (SETF RES (- (CAR LISTA2) (CAR LISTA1)))
      (COND ((> (ABS RES) 150) (FUN RES))
            (T (SETF CUA (EXPT RES 2))))
      (SETF ACUM (+ ACUM CUA)))
  (SETF ACUM (SQRT ACUM)))
```

```
(DEFUN FUN (RES)
  (SETF DIV (/ RES 100)
        EX  (EXPT DIV 2)
        CUA (* EX 10000)))
```

```
;;; Función para limpiar la pantalla

(DEFUN LIMPIAR ()
  (SEND *TERMINAL-IO* :CLEAR-SCREEN))
```

```
;;; Esta función sirve para recalcular el centro de una clase
;;;cuando un objeto nuevo entra en ella. LLama a la función
;;;ACCESO.

(DEFUN RECALCULAR (CENTRO OBJETO)
   (DECLARE (SPECIAL RLIST OBJETOS1 OBJETOS2 CENTROS CENTROS2))
   (SETF NUEVOBJ '())
   (DO ((LISTA CENTRO (CDR LISTA)))
       ((NULL LISTA) NIL)
       (SETF VAR (LIST (CAR LISTA)))
       (SETF CONT 0)
       (DO ((V RLIST (CDR V)))
           ((EQUAL VAR (CAR V)) (SETF CONT (1+ CONT)))
           (SETF CONT (1+ CONT)))
       (SETF NUEVOBJ (CONS (ACCESO CONT OBJETOS1) NUEVOBJ)))
   (SETF NUEVOBJ (REVERSE NUEVOBJ))
   (SETF P (LENGTH NUEVOBJ)
         Z (LENGTH (CAR NUEVOBJ))
         CENTRO '())
   (DO ((NZ 1 (1+ NZ)))
       ((> NZ Z) NIL)
       (SETF SUM 0)
       (DO ((NP 1 (1+ NP)))
           ((> NP P) (SETF COOR (/ SUM P)))
           (SETF SUM (+ SUM (ACCESO NZ (ACCESO NP NUEVOBJ)))))
       (SETF CENTRO (CONS COOR CENTRO)))
   (SETF CENTRO (REVERSE CENTRO)))
```

```
;;; Esta función sirve para escribir el fichero CLASES.DAT a
;;;partir de la lista CENTROS.

(DEFUN SALIDA1 ()
  (DECLARE (SPECIAL CENTROS))
  (SETF STR (OPEN "CLASES.DAT" :DIRECTION :OUTPUT))
    (FORMAT STR "~A" CENTROS)
    (CLOSE STR))
```

;;;Esta función sirve para escribir el fichero ;;;PERTENENCIAS.DAT. LLama a la función BON.

```
(DEFUN SALIDA2 (LISTA1 LISTA2)
   (DECLARE (SPECIAL RADIO))
   (LET ((PERTENP (OPEN "PERTENENCIAS.DAT" :DIRECTION :OUTPUT)))
     (DO ((NUM1 1 (1+ NUM1)))
         ((> NUM1 (LENGTH LISTA1)) (TERPRI))
         (DO ((NUM2 1 (1+ NUM2)))
             ((> NUM2 (LENGTH LISTA2)) (BON NUM1 LISTA1 PERTENP))
              (ELEG NUM1 NUM2 LISTA1 LISTA2)
              (SETF PERTMAT (- RADIO DISTANCIA))
              (FORMAT PERTENP " ~A " PERTMAT)))
     (FORMAT PERTENP "~%")
     (FORMAT PERTENP "~% FINAL DE FICHERO")
     (CLOSE PERTENP)))

(DEFUN BON (NUM1 LISTA1 PERTENP)
   (COND ((< NUM1 (LENGTH LISTA1)) (FORMAT T "~A ;"(1+ NUM1)))
         (T NIL)))
```

;;; Esta función sirve para escribir el fichero CLASES2.DAT a partir de la lista CENTROS2.

```
(DEFUN SALIDA3 ()
   (DECLARE (SPECIAL CENTROS2 NIVEL))
   (SETF CLAP (OPEN "CLASES2.DAT" :DIRECTION :OUTPUT))
   (FORMAT CLAP " ~A " CENTROS2)
   (CLOSE CLAP))



(DEFUN ELEG (NUM1 NUM2 LISTA1 LISTA2)
   (DECLARE (SPECIAL ACUM))
   (SETF DISTANCIA (DISTEUCLI (ACCESO NUM2 LISTA2)
                              (ACCESO NUM1 LISTA1))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; El proceso CARGA está diseñado para el tratamiento de;;;
;;;;los nuevos documentos que llegan a la colección una  ;;;;
;;;;vez que los clusters ya están formados. Es decir, se ;;;;
;;;;encarga del mantenimiento de los clusters.           ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;La función principal del proceso, CARGA, sirve para leer el
;;;registro-índice del nuevo documento desde el fichero
;;;INDICE.DAT, normalizarlo y grabarlo en el fichero DOC.DAT.
;;;LLama a las funciones INTRO, CALCULAR y LEER.
```

```
(DEFUN CARGA ()
   (SETF INDICEP (OPEN "INDICE.DAT" :DIRECTION :INPUT))
   (SETF AUXIP (OPEN "AUXI.DAT" :DIRECTION :OUTPUT))
   (SETF LEIN (READ INDICEP NIL 'EOF))
   (SETF CONT '0)
   (DO ()
       ((EQUAL LEIN  EOF))
       (SETF CONT (1+ CONT))
       (FORMAT AUXIP "~A ~% " LEIN)
       (SETF LEIN (READ INDICEP NIL  EOF)))
   (SETF FIN 'X)
   (SETF CONTADOR '0)
   (DO ()
       ((EQUAL FIN 'FF))
       (SETF CONTADOR (1+ CONTADOR))
       (DO ((CONTA 0 (1+ CONTA))
            (LISREG '() (CONS VALOR LISREG)))
           ((> CONTA 6) (SETF LISTREG (REVERSE LISREG)))
           (COND ((NOT (EQUAL CONTA 5))
                  (FORMAT T "~% INTRODUCIR EL VALOR DE R"A "~" CONTA))
                 (T (FORMAT T "~% INTRODUCIR EL VALOR DEL VECTOR ")))
           (SETF VALOR (READ)))
       (FORMAT AUXIP "~A ~% " LISTREG)
       (FORMAT T "~% QUIERE SEGUIR METIENDO DATOS (S/N)  ")
       (SETF CONTES (READ))
       (COND ((EQUAL CONTES 'S) (SETF FIN 'X))
             (T (SETF FIN 'FF))))
   (CLOSE INDICEP)
   (CLOSE AUXIP)
   (DELETE-FILE "INDICE.DAT")
   (RENAME-FILE "AUXI.DAT" "INDICE.DAT")
   (DELETE-FILE "AUXI.DAT")
   (LEER)
   (CALCULAR)
   (INTRO CONTADOR CONT))
```

```
;;; Esta función sirve para clasificar los nuevos documentos
;;; que llegan a la colección. LLama a la función FUNCION.

(DEFUN INTRO (CONTADOR CONT)
   (SETF PARAMP (OPEN "PARAM.DAT" :DIRECTION :INPUT))
   (SETF OBJETOS2 (READ PARAMP))
   (SETF CENTROS (READ PARAMP))
   (SETF CENTROS2 (READ PARAMP))
   (SETF CONTCENT1 (LENGTH CENTROS))
   (SETF OBJETOS2 (REVERSE OBJETOS2))
   (SETF RADIO (READ PARAMP))
   (CLOSE PARAMP)
   (DO ((N 1 (1+ N)))
       ((EQUAL N (1+ CONTADOR)))
       (SETF OBJETOS2 (CONS 0 OBJETOS2)))
   (SETF OBJETOS2 (REVERSE OBJETOS2))
   (SETF LISTA-ANTIGUA OBJETOS1)
   (DO ((N 0 (1+ N)))
       ((EQUAL N CONT))
       (SETF LISTA-ANTIGUA (CDR LISTA-ANTIGUA)))
   (DO ((LISAUX LISTA-ANTIGUA (CDR LISAUX))
        (CONT1 (1+ CONT) (1+ CONT1)))
       ((NULL LISAUX) LAMBDA ()
         (ESTAB)
         (UNIF)
         (TERPRI)
         (FORMAT T "OBJETOS1 ~A ~% OBJETOS2 ~A ~% CENTROS ~A ~% CENTROS2 ~A ~%
                                          " OBJETOS1 OBJETOS2 CENTROS CENTROS2)
         (SETF CLAS (LENGTH CENTROS))
         (SALIDA1)
         (MATRIZ)
         (CLASES2)
         (DELETE-FILE "PARAM.DAT")
         (SETF PARAMP (OPEN "PARAM.DAT" :DIRECTION :OUTPUT))
         (FORMAT PARAMP "~A ~% ~A ~% ~A ~% ~A ~%" OBJETOS2 CENTROS
                                                  CENTROS2 RADIO)
         (CLOSE PARAMP))
       (FUNCION LISAUX RADIO CONT1)))
```

BIBLIOGRAFIA

[ANDE73] ANDERBERG,M.R.. ***Cluster Analysis for Applications***. Academic Press, New York, 1973.

[BELK82] BELKIN,N.J.; ODDY,R.N.; BROOKS,H.M.. Ask for Information Retrieval.Part 1: Background and Theory. ***Journal of Documentation***. Vol.38, No.2, pp.61-71, 1982.

[BELL83] BELL,A.. Critical issues in hight density magnetic and optical storage. ***Proceedings of the SPIE***. Vol.382, Optical Storage, 1983.

[BELL84] BELL,A.; MARRELLO,V.. Magnetic and optical data storage: A comparison of the technological limits. ***Proceedings IEEE COMPCON***. New York, pp.512-517, spring 1984.

[BIDA89] BIDA,M.C.. Scanning, OCR Devices complement microfilm in solving data entry bottlenecks. ***International Journal of Micrographics and Optical Technology***. Vol.7, No.1, pp 25-28, 1989.

[BOLL84] BOLLMANN,P.. Two axioms for evaluation measures in information retrieval. *Proceedings of the 3d Joint BCS and ACM Symposium in Research and Development in Information Retrieval*. Cambridge, pp.233-245, 1984.

[CHRI83] CHRISTODOULAKIS,S.. Estimating block transfers and join sizes. *Proceedings of the ACM SIGMOD 83*. New York, 1983.

[CHRI84] CHRISTODOULAKIS,S.. Implications of certain assumptions in database performance evaluation. *ACM Transactions on Database Systems*. Vol.9, No.2, pp.163-186, June 1984.

[CHRI87] CHRISTODOULAKIS,S.. Analysis of retrieval performance for records and objects using optical disk technology. *ACM Transactions on Database Systems*. Vol.12, No.2, pp.137-169, June 1987.

[CAN 89] CAN,F.; OZKARAHAN,E.A.. Dynamic cluster maintenance. *Information Processing and Management*. Vol.25, No.3, pp.275-291, 1989.

[CORT88] CORTES,U.; MANERO,J.; ANDRES,B.. ***Recuperacion de informacion en fondos documentales mediante la utilizacion de la logica difusa de enunciados***. Facultad de Informática de Barcelona. Universidad Politécnica de Cataluña, 1988.

[COX 89] COX,T.. Scanners and OCR begin to come of age. ***Andrew Seybold's Outlook on Professional Computing***. Vol.7, No.8, pp.13-14, March 1989.

[DEWE88] DOWEZE,A.. ***Informática Documental***. Masson, 1988.

[DUBE76] DUBES,R.; JAIN,A.K.. Clustering techniques: the user's dilema. ***Pattern Recognition***. Vol.11, pp.247-260, 1976.

[EL-H89] EL-HAMDOUCHI,A.; WILLETT,P.. Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval. ***The Computer Journal***. Vol.32, No.3, pp.220-227, 1989.

[EVER74] EVERITT,B.S.. ***Cluster Analysis***. John Wiley & Sons, Inc., New York, 1974.

[FISH71] FISHER,L.; VAN NESS,J.W.. Admissible Clustering Procedures. *Biometrika*. Vol.58, pp.91-104.

[GOWE67] GOWER,J.C.. A Comparison of some Methods of Cluster Analysis. *Biometrics*. Vol.23, pp.623-637, 1967.

[HAMP89] HAMPSHIRE,N.. Mass Media. *Personal Computer World*. April,1989.

[HART85] HARTIGAN,J.A.. Statistical Theory in Clustering. *Journal of Classification*. Vol.2, pp.63-76, 1985.

[JAIN88] JAIN,A.K.; DUBES,R.C.. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[JARD71] JARDINE,N.; SIBSON,R.. *Mathematical Taxonomy*. John Wiley & Sons, New York, 1971.

[LUCA88] LUCARELLA,D.. A document retrieval system based on nearest neighbour searching. *Journal of Information Science*. Vol.14, pp.25-33, 1988.

[LUHN57] LUHN,H.P.. A statistical approach to the mechanized encoding and searching of literary. Information. *IBM Journal of Research and Development*. Vol.1, No.4, pp.309-317, October 1957.

[MILL81] MILLIGAN,G.W.. A Monte-Carlo study of 30 internal criterion measures for cluster-analysis. *Psychometrika*. Vol.46, pp.187-195, 1981.

[MITK89] MITKAS,P.A.; BERRA,P.B.; GUILFOYLE,P.S.. An optical system for full text search. *ACM*, pp.98-107, 1989.

[MORR89a] MORRIS,A.. So you want to buy an OCR?. *Construction Computing*. Spring 1989.

[MORR89b] MORRIS,A.. OCR Revisited. *Construction Computing*. Winter 1989/90.

[RADE88a] RADECKI,T.. Trends in research on information retrieval. The potential for improvements in conventional boolean retrieval systems. *Information Processing and Management*. Vol.24, No.3, pp.219-227, 1988.

[RADE88b] RADECKI,T.. Probabilistic methods for ranking output documents in conventional boolean retrieval systems. *Information Processing and Management*. Vol.24, No.3, pp.281-302, 1988.

[RAGH89] RAGHAVAN,V.V.; JUNG,G.S.; BOLLMANN,P.. A critical investigation of Recall and Precision as measures of retrieval system performance. *ACM Transactions on Information Systems*. Vol.7, No.3, pp.205-229, July 1989.

[RUBI67] RUBIN,J.. Optimal Classification into groups: an approach for solving the taxonomy problem. *Journal of Theorical Biology*. Vol.15, pp.103-144, 1967.

[SAEZ87] SAEZ VACAS,F.; FERNANDEZ,G.. *Fundamentos de Informática*. Alianza-Informatica, Madrid 1897.

[SAFF89] SAFFADY,W.. Text storage and retrieval.A state of the art survey. *Library Computer Systems and Computer Review*. Vol.11, No.1, pp.2-14,1989.

[SALT83] SALTON,G.; McGILL,M.J. *Introduction to modern information retrieval*. McGraw Hill Book Company, 1983.

[SALT88a] SALTON,G.; BUCKLEY,C.. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*. Vol.24, No.5, pp.513-523, 1988.

[SALT88b] SALTON,G.. A simple blueprint for automatic boolean query processing. *Information Processing and Management*. Vol.24, No.3, pp.269-280, 1988.

[SALT89] SALTON,G.. *Automatic Text Processing*. Addison-Wesley Publishing company, 1989.

[SOER74] SOERGEL,D.. *Indexing languages and thesauri: construction and maintence*. Melville Publishing Company, California 1974.

[SPIE88] SPIEGLER,I.; ELATA,S.. A priori analysis of natural language queries. *Information Processing and Management*. Vol.24, No.6, pp.619-631, 1988.

[TAGU89] TAGUE,J.; SHULTZ,R.. Evaluation of the user interface in an information retrieval system: a model. *Information Processing and Management*. Vol.25, No.4, pp.377-389, 1989.

[WALL79] WALLER,W.G.; KRAFT,H. A mathematical model of a weighted boolean retrieval system. *Information Processing and Management*. Vol.15, pp.235-245, 1979.

[WALT89] WALTER,G. Technology overview: optical disk based document management systems (OD/DMS). *Information Journal of Micrographics and Optical Technology*. Vol.7, No.1, pp 15-24, 1989.

[WARE85] WAREN,C.. Optical storage shines on the horizon. *Mini-Micro Systems*. pp.68-80, december1985.

[WONG84] WONG,S.K.M.; VAN RIJSBERGEN,C.J.; RAGHAVAN,V.V.. Vector space model of information retrieval. *Research and Development in Information Retrieval*. Cambridge University Press, 1984.