**UNIVERSIDAD DE CANTABRIA**

**Departamento de Matemática Aplicada y Ciencias de la Computación**

**Tesis doctoral**

# Procesamiento Paralelo y Distribuido aplicado al Almacenamiento y Recuperación de Información Documental

**Marta Elena Zorrilla Pantaleón**

**2001**

# Tesis Doctoral

para la obtención del grado de
Doctor Ingeniero de Telecomunicación

# Procesamiento Paralelo y Distribuido aplicado al Almacenamiento y Recuperación de Información Documental.


Realizada por:

Marta Elena Zorrilla Pantaleón

bajo la dirección de:

Dr. Eduardo Mora Monte

Dr. José Luis Crespo Fidalgo

Presentada en el

Departamento de Matemática Aplicada
y Ciencias de la Computación
de la
Universidad de Cantabria


Santander, 2001

*A mi familia*

## Agradecimientos

La autora quiere agradecer en estas líneas el apoyo recibido de D. Eduardo Mora Monte y D. José Luis Crespo Fidalgo durante el desarrollo de esta tesis, sin cuya colaboración no hubiera sido posible.

A Eduardo, agradecer su interés y preocupación por los que trabajamos con él y su asesoramiento y aliento en el trabajo diario.

Por otra parte, y centrándome en el desarrollo de este trabajo, agradecerle las aclaraciones y explicaciones recibidas sobre las técnicas existentes en el tratamiento de información, de las que tiene un amplio conocimiento. Tampoco debo olvidar sus siempre acertadas sugerencias fruto de su experiencia universitaria.

A José Luis, agradecerle las indicaciones, sugerencias y valiosas referencias dadas sobre los algoritmos de procesado paralelo y distribuido. Su vasto conocimiento del tema y su disposición ha sido de una ayuda incalculable para mi.

# Contenidos

# Introducción

# 1. Introducción

El importante avance tecnológico en las Telecomunicaciones y la Informática producido en los últimos tiempos ha dado lugar a que cualquier usuario pueda acceder a enormes cantidades de información de forma muy sencilla. Este es el propósito fundamental de Internet que es una de la mayores fuentes de información.

Esta importante capacidad de gestionar información de tipos tan diferentes y de temas tan diversos que la Red ofrece, ha dado lugar a nuevas necesidades de almacenamiento, búsqueda y visualización de la información.

Esta problemática no ha surgido en los años 90 sino que ya en los años 60, Gerard Salton [70,71] y sus discípulos comenzaron sus primeros pasos en este campo tratando de agilizar la gestión de catálogos de las bibliotecas.

Hasta hace pocos años, la mayor parte de la información se ha almacenado en Sistemas de Bases de Datos Relacionales, en los que la referencia a un documento se representa mediante una serie de campos estructurados, tales como autor, título, ISBN,... y las búsquedas se pueden realizar por medio de técnicas booleanas. Se sabe que estas técnicas sólo son adecuadas cuando el solicitante tiene suficiente información para realizar la búsqueda a partir de los campos que caracterizan al documento; así, sería difícil localizar una revista sin saber su título o ISBN.

La actual tecnología permite, por ejemplo, que los catálogos de las bibliotecas se amplíen con resúmenes e incluso versiones electrónicas completas de los artículos, libros, .... almacenándolos en sus propios sistemas de bases de datos.

Los Sistemas de Bases de Datos Relacionales puros no están concebidos para realizar búsquedas que permitan localizar textos a partir de las palabras que se encuentran en ellos, lo que se denomina búsqueda *full-text o búsqueda por contenido*. En general, son capaces de almacenar cualquier tipo de información en campos denominados Binarios (BLOBs – Binary Large Objects) pero el acceso a ellos debe realizarse a partir de otros campos que permitan localizar dicho objeto.

Paralelamente, desde hace años, se dispone de los denominados Sistemas de Almacenamiento y Recuperación de Información *(Information Retrieval Systems)* que permiten almacenar, buscar y mantener información, entendiendo por ésta textos, imágenes, vídeos, audios y otros objetos multimedia, los cuales, utilizan técnicas de búsqueda relativas a su contenido, que son específicas para cada tipo de información. Aunque la gama de posibilidades puede ser muy amplia, la información textual ha sido y continúa siendo la más estudiada.

Dado que la sociedad actual demanda un tratamiento global de la información, los fabricantes de Sistemas Gestores de Bases de Datos más conocidos como Informix, Oracle, Sybase,... han extendido sus productos relacionales para gestionar este nuevo tipo de información sin sacrificar la fiabilidad y escalabilidad de los mismos.

La elección de este tema para la presente tesis surgió como consecuencia de la participación de la autora en un proyecto de I+D que consistía en el diseño e implantación de un sistema de información que combinase las capacidades de un Gestor de Bases de Datos Relacional con un Gestor de Base de Datos Documental, antes de que existieran productos comerciales que dispusieran de ambas capacidades. Con este sistema de información, los usuarios podrían buscar los documentos a partir de los campos de su ficha o por el contenido de los mismos.

Al analizar Gestores de Bases de Datos existentes en aquel momento, se encontró que uno de los estudiados, hacía uso de redes neuronales artificiales para la indexación documental. Por ello, pareció una buena idea profundizar en este tipo de técnicas como alternativa a la de Índices Inversos que son el soporte de la mayoría de los productos comerciales dedicados a este fin.

Por último, la revolución que se está produciendo en los sistemas de información debido al fenómeno WWW y la aparición de los potentes buscadores como Altavista, Infoseek, Fulcrum, Topic, etc. que usan técnicas de IR (Information Retrieval) no hizo más que acrecentar la curiosidad de la autora.

Por estos motivos, esta tesis se centra en el tratamiento de la información textual, con objeto de permitir la búsqueda de documentos por su contenido.

Como se ha mencionado, la técnica más utilizada es la denominada de Índices Inversos y existen alternativas comerciales, con redes neuronales artificiales auto-organizativas dedicadas a clasificación. En este trabajo se pretende realizar propuestas basadas en otros tipos de redes neuronales y comparar su rendimiento respecto al que se obtiene con la técnica de Índices Inversos. En la evaluación comparativa de resultados obtenidos con ambas técnicas se han considerado los siguientes aspectos:

- El tiempo de indexación de la colección y el comportamiento de la estructura ante la incorporación de nuevos documentos y el borrado de los existentes.

  En este punto, se evalúa el tiempo que necesita el sistema en 'aprender' la matriz de términos x documentos de la colección a indexar. Al respecto, conviene adelantar que el tiempo de entrenamiento de la red, normalmente, será superior por precisar procesos de optimización.

  También se evalúa el coste de incorporación de nuevos documentos tanto en tiempo como en recursos de memoria. Ante esta situación, también se comprueba el grado de crecimiento del número de parámetros de la red cuando el número de documentos indexados es alto.

  Por otra parte, se calcula el coste que se produce al eliminar un documento del sistema. Conviene aquí observar que, en la estructura de red neuronal que se propone, basta con eliminar el procesador que representa al documento, junto con sus conexiones, mientras que, en Índices Inversos es preciso anular índices a varios niveles, lo que hace pensar en una cierta ventaja de la red neuronal respecto a la otra alternativa.

- Los recursos de almacenamiento.

  En este caso, se trata de comparar, de alguna manera, el número de parámetros que la red requiere, con el espacio necesario para la correspondiente indexación mediante Índices Inversos. Cabe mencionar, que la experiencia demuestra que en los sistemas basados en Índices Inversos, el espacio necesario para indexación, es aproximadamente el 80% del requerido para almacenar sólo la colección de documentos.

- La eficiencia de búsqueda.

La característica fundamentalmente a considerar en este aspecto es el tiempo de respuesta ofrecido por cada técnica ante las mismas consultas.

- La efectividad de la consulta.

En este punto, se trata de medir el grado de acierto de cada sistema al seleccionar los documentos que mejor responden a la pregunta realizada. Cabe mencionar que en este trabajo se está exigiendo el acierto en el 100% de los casos.

# Sistemas de Almacenamiento y Búsqueda de Información

# 2. Sistemas de Almacenamiento y Búsqueda de Información

*Este capítulo describen las características básicas de los Sistemas de Almacenamiento y Recuperación de Información, conocidos como Information Retrieval Systems (IR Systems). Entre ellas se incluyen su modo de funcionamiento y las técnicas de indexación comúnmente utilizadas. A continuación, se comentan las características más relevantes de los Sistemas IR comerciales de mayor difusión.*

En general, un Sistema de Almacenamiento y Recuperación de Información Documental es un sistema que permite indexar los documentos a partir de su contenido, para su posterior localización haciendo uso de nuevas posibilidades de búsqueda como son la búsqueda por palabras contenidas en el documento, por categorías predefinidas, por sinónimos, por proximidad de los términos de la búsqueda con las palabras del documento e incluso, consultas realizadas en lenguaje natural.

De una forma gráfica, ver Figura 1, puede decirse que un Sistema de Almacenamiento y Recuperación de Información Documental es una interfaz a una gran colección de documentos, donde el usuario, ante la necesidad de una determinada información, genera una consulta y la introduce al sistema. Éste localiza los documentos que mejor responden a la consulta formulada y se los presenta al usuario, generalmente, en forma de lista. El número de documentos seleccionados puede ser elevado si la consulta es demasiado general. Una solución, comúnmente utilizada, es la presentación de los documentos clasificados por orden de importancia. De este modo, el usuario sólo debe inspeccionar los documentos más relevantes para evaluar el resultado de la consulta. El juicio del usuario sobre los documentos que para él son más relevantes puede utilizarse para refinar la consulta (semi)-automáticamente (realimentación de relevancia).

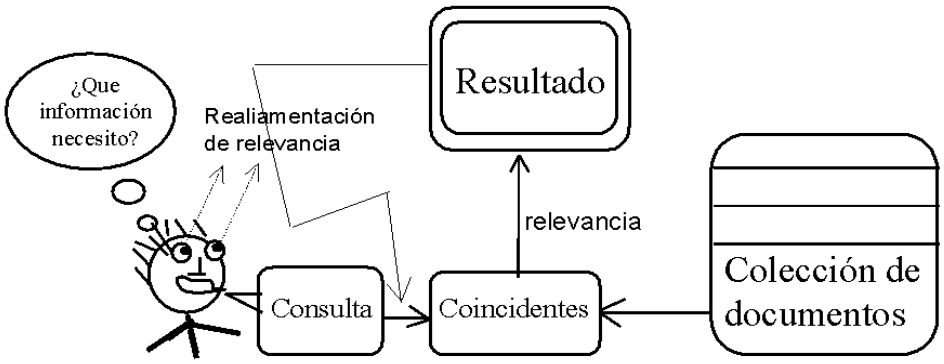


Figura 1. Esquema de un Sistema de Almacenamiento y Recuperación de Información.

### *2.1. Funciones de un sistema de IR*

Todo Sistema de IR contempla, al menos, dos tipos de procesos, que se representan en la Figura 2:

1. El proceso de indexación y almacenamiento de documentos.
2. El proceso de búsqueda y presentación al usuario del conjunto de documentos seleccionados.

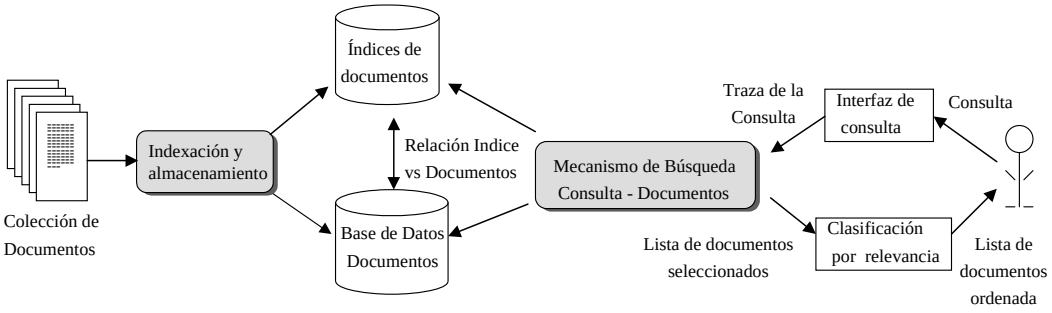


Figura 2. Proceso de Indexación y Búsqueda de documentos.

El proceso de Indexación normalmente requiere una fase de preprocesado del documento al que también se verá sometido la consulta que realice el usuario cuando haga uso del sistema.

Esta fase de preprocesado, ver Figura 3, consiste en partir del documento original para generar un documento texto (ASCII), en el que se encuentren exclusivamente las palabras a indexar. Si el documento original se encuentra en algún formato distinto al formato texto (por ejemplo word, excel, pdf...), previamente al preprocesado, deberá filtrarse por medio de conversores comerciales. En este preproceso, con el que se genera el fichero texto con las palabras a indexar, pueden realizarse las siguientes operaciones:

- Filtrado de palabras vacías (*stopwords*): consiste en la supresión del documento de aquellos términos que no son relevantes semánticamente. En esta categoría se incluyen los pronombres, artículos, preposiciones, conjunciones, adverbios, etc. Generalmente, el diccionario de palabras vacías se encuentra disponible para el usuario de forma que, él mismo, parametrice la lista de palabras que el sistema no debe considerar.

- Proceso de reducción por derivados (*stemming*): consiste en el análisis morfológico de las palabras para reducirlas a su raíz, por ejemplo verbos regulares e irregulares a su infinitivo, adverbios a su adjetivo, sustantivos plurales a su singular correspondiente, etc. La ventaja que ofrece este proceso es la reducción del tamaño de la base de datos de términos a indexar. Generalmente el "*stemmer*" se utiliza en la fase de búsquedas para la extensión de una consulta realizada por el usuario.

- Proceso de reducción por sinónimos (*Thesaurus*): Consiste en almacenar junto con cada término del diccionario de la BD, otros que se relacionan con el primero, bien por sinonimia, bien por relación jerárquica, de equivalencia u otra. Del mismo modo que el proceso de reducción por derivados, éste se utiliza más en la fase de búsquedas que en la propia indexación a no ser que el objetivo del sistema sea tener una base de términos reducida.

- Proceso de reducción por lista de términos a indexar: consiste en filtrar el documento según una lista de términos definida previamente (diccionario preestablecido), de forma que se obtenga un fichero con los términos que contiene de los del diccionario.

Una vez finalizado el filtrado del documento se guardará el documento original así como los términos que contiene el documento indexado.

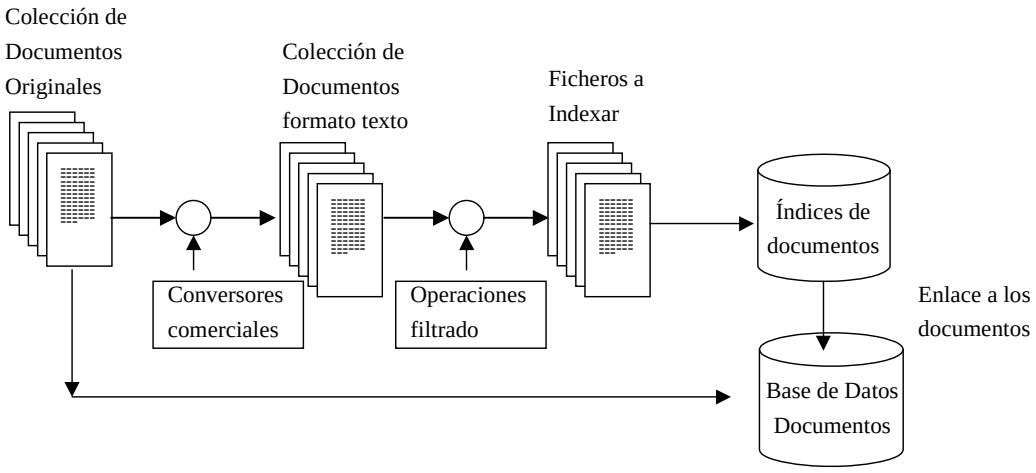


Figura 3. Preprocesado de documentos.

El proceso de búsqueda y recuperación de documentos consta, a su vez, de varias fases que se pueden resumir en:

- El filtrado de la consulta con los mismos criterios que los utilizados en el proceso de indexación,
- Realizar la operación de búsqueda de documentos que concuerden con la consulta realizada y,
- Presentar los documentos seleccionados al usuario ordenados por un criterio de relevancia.

La capacidad de búsqueda va a estar condicionada por la técnica de indexación que se utilice. Las más importantes se comentan en el apartado siguiente.

## *2.2. Técnicas de Indexación*

Las técnicas de indexación son muchas y diversas, así que su clasificación se puede realizar atendiendo a diversos criterios.

Uno de éstos puede basarse en la utilización de técnicas tradicionales y técnicas no tradicionales, según Faloustsos [27], entendiendo por estas últimas, aquellas que incluyen información semántica.

| | |
|---|---|
| **Técnicas tradicionales** | Full Text Scanning.<br>Signatures Files.<br>Índices inversos.<br>Modelo del Espacio Vector<br>Clustering probabilístico |
| **Técnicas no tradicionales** | Redes neuronales<br>Latent Semantic Indexing |

Clasificación según Faloustsos.

Otra clasificación, que se propone en esta tesis, puede fundamentarse en el conjunto de palabras con las que el sistema pueda trabajar. Surgen así dos grandes bloques según utilicen cualquier palabra (diccionario libre) o hagan uso sólo de un determinado conjunto (diccionario preestablecido). Una clasificación inicial es la que se presenta en la Figura 4.

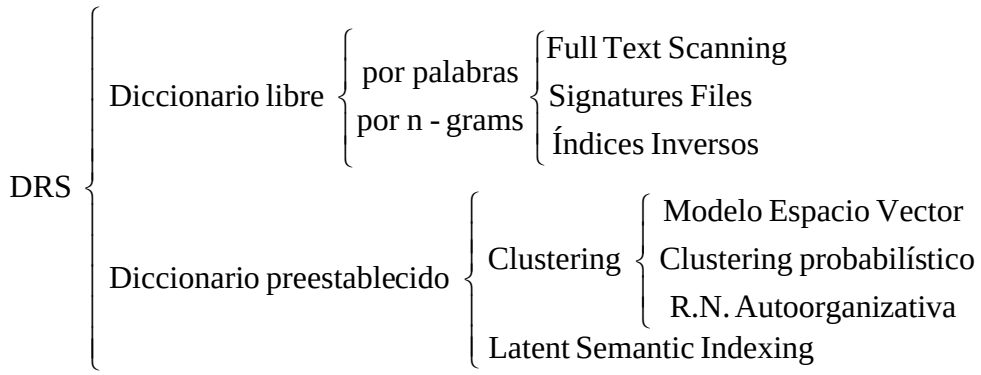


Figura 4. Clasificación de los Sistemas de Almacenamiento y Recuperación Documental.

En el caso de diccionario libre, a su vez, hay dos posibilidades: actuar sobre palabras de la lengua que se estén utilizando (por palabras), lo que lleva implícito un cierto mantenimiento de su significado; o actuar sobre bloques de caracteres de longitud fija, n, (por n-grams), con lo que, en cierto modo, se pierde su sentido dentro del texto. En este último caso, la cantidad de índices puede resultar relativamente reducida y, además, el sistema es independiente del idioma. En ambos casos, la técnica de indexación más implantada comercialmente es la de "Índices Inversos".

La otra alternativa consiste en delimitar previamente el juego de palabras a utilizar a un cierto conjunto no excesivamente grande (diccionario preestablecido) . En esta línea, fundamentalmente hay dos posibilidades: Clustering, mediante estadísticos (probabilísticos) o utilizando redes neuronales autoorganizativas y, Latent Semantic Indexing.

A continuación se describen brevemente las técnicas mencionadas. Estas técnicas no deben considerarse como herramientas aisladas sino que, frecuentemente, se combinan para beneficiarse de ventajas que se complementan.

### 2.2.1. Full Text Scanning

Esta técnica se puede considerar más un método de búsqueda que de indexación propiamente dicha, ya que no genera índices que se refieran a los términos presentes

en la colección. Esta técnica localiza los documentos que contienen un término concreto buscando en el contenido de cada uno de ellos. La única ventaja de este método es que no requiere espacio de disco extra para el almacenamiento de los índices. Por el contrario, el tiempo de respuesta en las búsquedas es muy alto.

### 2.2.2. Signatures Files

En este método la indexación consiste en crear un string binario (‘signature’) para cada documento o sección del mismo. Este string se forma mediante la composición con la operación OR de los códigos binarios de longitud fija de cada palabra o n-gram que forme parte del documento o sección, que se obtienen por medio de una función hash.

Cada uno de estos ‘signatures’ se almacena en otro fichero -‘signature file’- que es de menor tamaño que los ficheros originales y donde las búsquedas son más eficientes. La búsqueda consiste en convertir las palabras de la consulta en su 'signature' y comparar bit a bit con las 'signatures' de los documentos de la Base de Datos.

La desventaja que ofrece este método es el tiempo de respuesta cuando el ‘signature file’ es muy grande. Las ventajas son su sencillez en la implementación y la eficiencia en manejar inserciones.

Este método ha sido utilizado según Faloustsos [27] en bases de datos con pocos términos.

### 2.2.3. Índices Inversos

Este método es el seguido por la mayoría de los sistemas comerciales y consiste, ver Figura 5, en considerar los textos divididos en documentos, párrafos, frases y palabras. A partir de la entrada de textos, dinámicamente, se va generando un DICCIONARIO de palabras indexadas (utilizables en búsquedas). En él, se van incorporando las palabras nuevas, actualizando la cantidad de documentos en que se encuentra cada una y el número total de apariciones. Paralelamente, se construye una LISTA DE PALABRAS en la que, por cada aparición de una palabra se almacena el

número de orden de la palabra dentro de su frase, el de la frase dentro del párrafo, el del párrafo dentro del documento y el código dado al propio documento.

Las palabras en la lista se almacenan alfabéticamente en un fichero de índices y para cada una de ellas se tiene una lista de punteros a los documentos que identifican.

El fichero de índices se puede implementar por medio de árboles binarios, árboles binarios balanceados, mediante técnica hashing y variaciones o combinaciones de éstos.

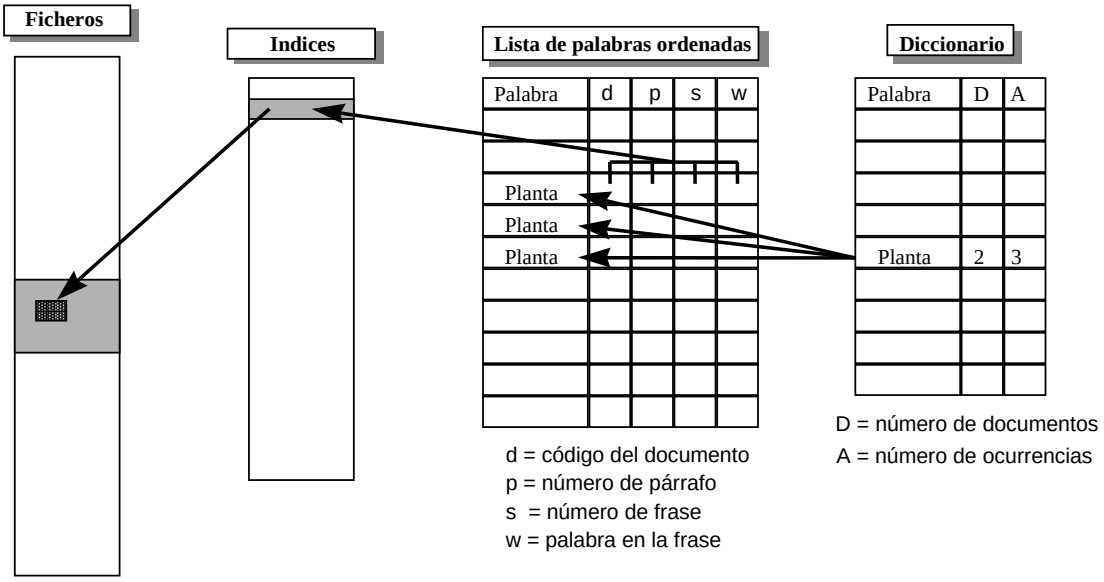


Figura 5. Técnica de Índices Inversos.

Las desventajas presentes en este método son el espacio extra necesario para almacenar los índices y el coste de actualización y reorganización de los mismos. Como ventajas presentan su fácil implementación y su sencillez para incorporar sinónimos, que se puede organizar como una lista dentro del diccionario.

### 2.2.4. Modelo del Espacio Vector

En el Modelo del Espacio Vector (VMS) el concepto fundamental es el Espacio de Documentos, ver Figura 6, cuyas dimensiones son los términos (índices) de la colección de documentos.

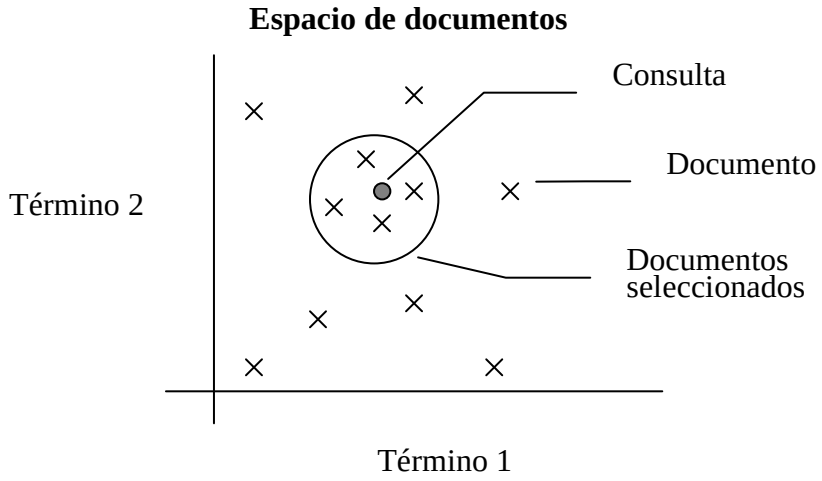


Figura 6. Modelo del Espacio Vector.

Cada documento se representa como un vector, o punto, en este espacio. Cada componente del vector toma un valor numérico que indica la relevancia de ese término concreto en él.

Existen varias funciones para el cálculo de relevancia propuestas [70]:

- Binaria: se representa con un 1 si se encuentra el término en el documento y con un 0 en caso contrario.
- Frecuencia del término: la relevancia del término *i* en el documento *k* toma el valor de la frecuencia, $FREQ_{ik,}$ del término en el documento.
- Frecuencia inversa: la relevancia toma el valor del cociente, $FREQ_{ik}$ / $DOCFREQ_{k,}$ entre la frecuencia del término ($FREQ_{ik}$) y el número de documentos que contienen el mismo ($DOCFREQ_{k}$).

Esta técnica de indexación requiere que los términos estén poco correlados para que pueda expandirse, propiamente, el espacio de documentos. Si el diccionario de términos es muy grande, se recurre a la técnica LSI (ver apartado 2.2.7) para obtener los términos principales de la colección de documentos.

Cuando el usuario solicita una consulta, ésta se representa en el espacio de documentos y la búsqueda se realiza por medio del cómputo de la distancia (generalmente distancia euclídea o del coseno) entre el vector consulta y los vectores de los documentos. El resultado se ofrece en forma de lista de documentos ordenados por similaridad, es decir, los más "próximos" a la consulta se presentarán los primeros.

Gerard Salton fue el precursor de la aplicación de VMS a los sistemas IR, desarrollando un sistema conocido por el nombre de SMART [71], en el que él y sus discípulos siguen trabajando en la actualidad.

### 2.2.5. Clustering probabilístico [20,21]

El clustering probabilístico también parte de la representación del documento como un vector, como en el caso del Espacio del Modelo Vector. La agilidad de este método se basa en comparar el vector de búsqueda con las clases de documentos en lugar de hacerlo con cada uno de los vectores individualmente. Por ello se requiere una clasificación previa de éstos. Este método ofrece dos ventajas: la primera, que la consulta tiene que ser comparada con un número limitado de "clusters" representativos de la colección de documentos lo que hará que el tiempo de búsqueda sea menor y, la segunda, que la búsqueda será más efectiva dado que los documentos contenidos en los clusters seleccionados serán los más parecidos (similares) a la consulta realizada.

Existe un gran número de algoritmos de clustering en el campo de la Estadística. El éxito del sistema dependerá de la elección del algoritmo que mejor conjunto de clusters cree respecto al coste computacional que requiera. Tradicionalmente los algoritmos más eficientes en la generación de clusters, como los métodos aglomerativos jerárquicos (hierarchic agglomerative methods), son los menos eficientes computacionalmente, ya que requieren calcular la matriz de similaridad entre todos los documentos de la colección. Cuando la colección de documentos es grande, de orden n, estos algoritmos llevan consigo una complejidad de $O(n^2)$ hasta $O(n^5)$ en tiempo y $O(n^2)$ en espacio de memoria.

La ventaja principal de estos métodos es que están perfectamente definidos y existe literatura disponible tanto en el ámbito teórico como práctico.

En este contexto, existen otros tipos de algoritmos, basados en partición, cuyo objetivo es la división de la colección en conjuntos planos de documentos más que en clases relacionadas jerárquicamente. Estos algoritmos tienen una complejidad en tiempo proporcional al tamaño de la colección.

*1. Algoritmo aglomerativo jerárquico .*

Estos algoritmos, generalmente, trabajan considerando iterativamente todas las parejas de clusters de la colección de documentos a clasificar y uniendo aquella pareja que tenga la mayor similaridad creándose un nodo del dendograma (árbol jerárquico). Ver Figura 7.

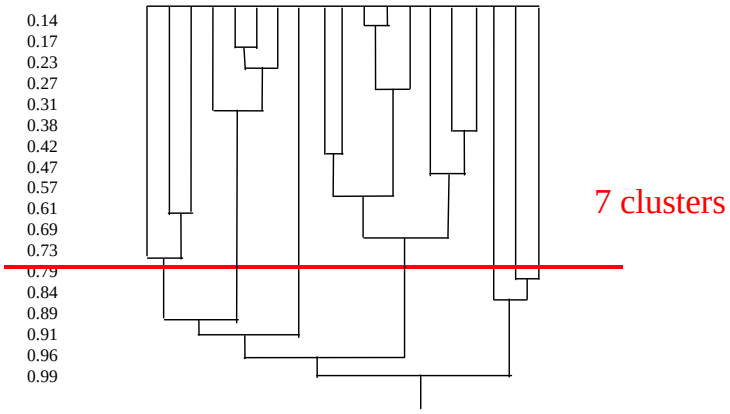


Figura 7. Clustering jerárquico. Dendograma.

Existen diversos algoritmos aglomerativos jerárquicos. La diferencia entre ellos se encuentran en la forma de cálculo de la similaridad cuando uno de los elementos de la pareja es el producto de una unión previa. Algunos de estos métodos son:

- Single Linkaje (SL). Se define la similaridad entre clusters como la máxima similaridad entre cualquier pareja de items, perteneciendo cada uno a un cluster distinto. Por tanto, cada item en un cluster será más similar al menos similar de su grupo que a cualquier otro de otro cluster. Este método ha sido el más

utilizado en sistemas de IR y tiende a producir grandes clusters con poca cohesión.

- Complete Linkaje (CL). En este método se considera la similaridad mínima en vez de la máxima por lo que cada item en un cluster es más similar al menos similar en el cluster que al menos similar de cualquier otro cluster. Este método tiende a generar un número grande de pequeños clusters.

- Group Average (GA). Este método toma, como magnitud a considerar para el agrupamiento, la similaridad promedio de los clusters.

Con uno de estos criterios se puede obtener el diagrama en el que se pueden localizar los documentos que respondan a la consulta realizada. Sorprendentemente, después de la investigación en este campo, Croft [20] ha encontrado que la mejor estrategia de búsqueda es utilizar los clusters del nivel más bajo.

*2. Algoritmo basados en partición.*

Estos algoritmos proceden de la siguiente manera, eligen un número de puntos semilla como centroides de los clusters (generalmente el número se corresponde al número de clusters final deseado) y se asignan los documentos al punto más cercano. Este algoritmo actúa en una única vuelta. Podría utilizarse sucesivas vueltas con el fin de refinar los centroides de los clusters. La diferencia principal entre estos algoritmos es el método utilizado para seleccionar los puntos iniciales. Algunas de estas opciones son:

- Seleccionar los primeros k documentos de la colección como semilla, donde k es el número de clusters a generar.

- Seleccionar aleatoriamente documentos de la colección como semillas.

- Dividir la colección en grupos y elegir como semilla los centroides de cada uno.

- Crear las semillas con un generador aleatorio.

- Elegir como semillas aquellos documentos que tienen parecido con otros documentos de la colección.

Los inconvenientes que presentan se pueden resumir en que estos métodos son sensibles a los parámetros iniciales y al orden de presentación de los documentos durante la clasificación.

### 2.2.6. Redes neuronales

Una alternativa al clustering por medio de estadísticos es la utilización de redes neuronales artificiales de aprendizaje competitivo. Existe un gran número de referencias que ofrecen los resultados obtenidos utilizando estas técnicas aplicadas a sistemas IR. En [75], Scholtes evalúa el rendimiento del método de redes autoorganizativas (SOM) según el algoritmo de Kohonen aplicado a una pequeña colección de documentos; en [44, 83] se compara la utilización del método de Kohonen con el método Growing Cell Structures de Fritzke; en [46,47,48] Teuvo Kohonen, describe el sistema WEBSOM, desarrollado por él y sus discípulos, basado en la utilización de su método y aplicado a grandes colecciones de documentos (superando el millón); y en [40], se compara el rendimiento de un sistema IR implementado con el método de Kohonen al que incorpora lógica difusa con respecto al modelo ART (Adaptive Resonance Theory).

El esquema más simple de una red de aprendizaje competitivo consta de una única capa de neuronas. Cada una de estas neuronas tienen un vector de conexión al conjunto de neuronas de entrada. Cuando se presenta un patrón a la entrada de la red, se evalúa la salida de cada neurona y aquella que tenga el valor más alto (gana la competición), cambia su vector de pesos en la dirección del vector de entrada. Después de presentar a la red un número suficiente de patrones, cada neurona responderá a un subconjunto del espacio de vectores de entrada. Por tanto, la respuesta de la red se puede ver como una categorización, clustering, del conjunto de vectores de entrada, donde el vector de pesos de cada neurona representa el prototipo del cluster.

El mapa autoorganizativo de Kohonen, Figura 8, es una variación del esquema de aprendizaje competitivo. Este consiste en una capa de neuronas emplazadas en una rejilla rectangular de dos dimensiones cuyo objetivo es obtener una clasificación de los patrones de entrada en la rejilla de forma que, patrones similares activen neuronas que están próximas entre sí y que neuronas próximas representen patrones similares. En estas condiciones, se dice que, la red preserva la topología del espacio de entradas en el espacio de la rejilla.

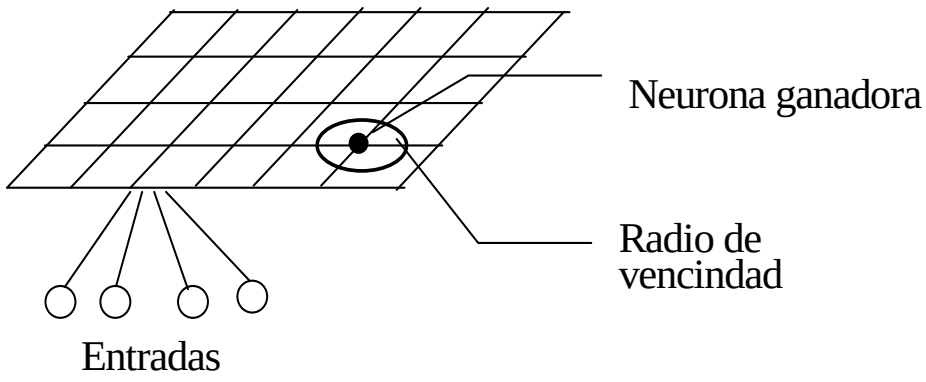


Figura 8. Mapa auto-organizativo de Kohonen.

Existen otros métodos que surgen como una variante al método de Kohonen. Estos son: el método Growing Cell Structures de Fritzke [30,31,33], el método Dinamic Cell Structures de Martinez [56] y el método de Blackmore y Mikkulaine [8].

La idea básica en estos métodos es la de comenzar con una pequeña red y adaptarla al espacio de entrada (patrones) incorporando y eliminando neuronas en las regiones donde es necesario. De este modo, una región densa del espacio de entradas se cubrirá con un número mayor de clusters que una región poco poblada. Estos métodos reflejan de forma más precisa la estructura cluster obtenida, a diferencia del método de Kohonen que la presenta en una rejilla de dos dimensiones. También son redes que tienen menor probabilidad de quedarse en mínimos locales ya que crecen ordenamente. Por otra parte, son muy sensibles a las variaciones de los parámetros iniciales por lo que las hace más inestables.

Otro algoritmo de red neuronal utilizada para la clasificación de documentos fue desarrollado por MacLeod y Robertson [55] utilizando redes del tipo ART (Adaptive Resonance Theory). Desarrollaron un algoritmo cuya complejidad se encuentra entre métodos aglomerativos jerárquicos y los algoritmos basados en partición pero con una efectividad comparable a los primeros.

### 2.2.7. Latent Semantic Indexing

Latent Semantic Indexing [6] es un método que trata de resolver el problema del emparejamiento léxico entre los términos de una búsqueda y los documentos disponibles, utilizando índices conceptuales en lugar de palabras individuales.

Este método asume que existe cierta asociación entre los términos utilizados en los documentos de una colección. Además considera que esta asociación puede ser estimada haciendo uso de técnicas estadísticas. En particular, hace uso de la descomposición de valores singulares (Singular-Value Descomposition) para extraer la relación entre los términos del diccionario y los documentos de la colección.

El método trabaja de la siguiente manera, ver Figura 9:

1. Se construye la matriz de términos x documentos $A=[a_{ij}]$ , donde $a_{ij}$ representa la frecuencia de cada término i en cada documento j.

2. La matriz A es factorizada en el producto de tres matrices haciendo uso de la descomposión de valores singulares, donde U representa los vectores de términos, V los vectores de documentos y Σ los valores singulares.

3. De esta descomposición se toman los k factores más relevantes, recogiéndose así la relación más importante entre términos y documentos y, al mismo tiempo, se elimina la variabilidad del uso de cada término en los mismos. En este espacio reducido, la asociación semántica entre documentos se obtiene en función de la frecuencia con la que ocurren los términos utilizados en un documento con relación al resto de los documentos de la colección. De modo similar, la asociación semántica entre términos se obtiene en función de la frecuencia con la que aparecen éstos en documentos de contenido similar

4. La consulta realizada por el usuario debe representarse como un vector k-dimensional y compararse con los documentos de la colección, generalmente con la medida de similaridad del coseno, para ofrecer el resultado en orden de relevancia. Una consulta se trata como un documento y se representa por medio de la ecuación (ec.1), donde q es el vector de términos obtenido de la consulta del usuario:

$$\hat{q} = q^t U_k \varepsilon_k^{-1} \quad \text{(ec. 1)}$$

5. Si la base de datos generada por LSI ya existe y se requiere incorporar nuevos términos y/o documentos, se tienen dos alternativas, o bien, se recalcula la descomposición de valores singulares y se elige un nuevo espacio k-dimensional, o bien, se añaden al espacio existente. Para llevar a cabo esto último, se debe representar el documento como un vector *d* cuyas componentes se corresponden con el peso de cada término en el documento y proyectarlo en el espacion k-dimensional según la ecuación (ec. 2). Una vez computado, se le añade a la matriz existente $U_k$. Del mismo modo se obrará en el caso de términos, pero aplicando la ecuación (ec. 3) e incorporándose el vector obtenido a la matriz $V_k$.

$$\hat{d} = d^t U_k \varepsilon_k^{-1} \quad \text{(ec.2)}$$

$$\hat{t} = t^t V_k \varepsilon_k^{-1} \quad \text{(ec.3)}$$

Este método resuelve algunas de las deficiencias de emparejamiento de términos que existe en los sistemas basados en palabras y, al mismo tiempo, suministra automáticamente una forma de obtener términos sinónimos sin necesidad de construir o actualizar un thesaurus. Además, como la dimensión del espacio semántico resultante, típicamente entre 100 y 300, es muy inferior al número de términos diferentes utilizados en los documentos de la colección, este método tiene la ventaja de requerir menor tiempo y menor cantidad de memoria para procesar las

consultas del usuario, no así para la obtención de la matriz de valores singulares cuyo coste computacional es alto [79].

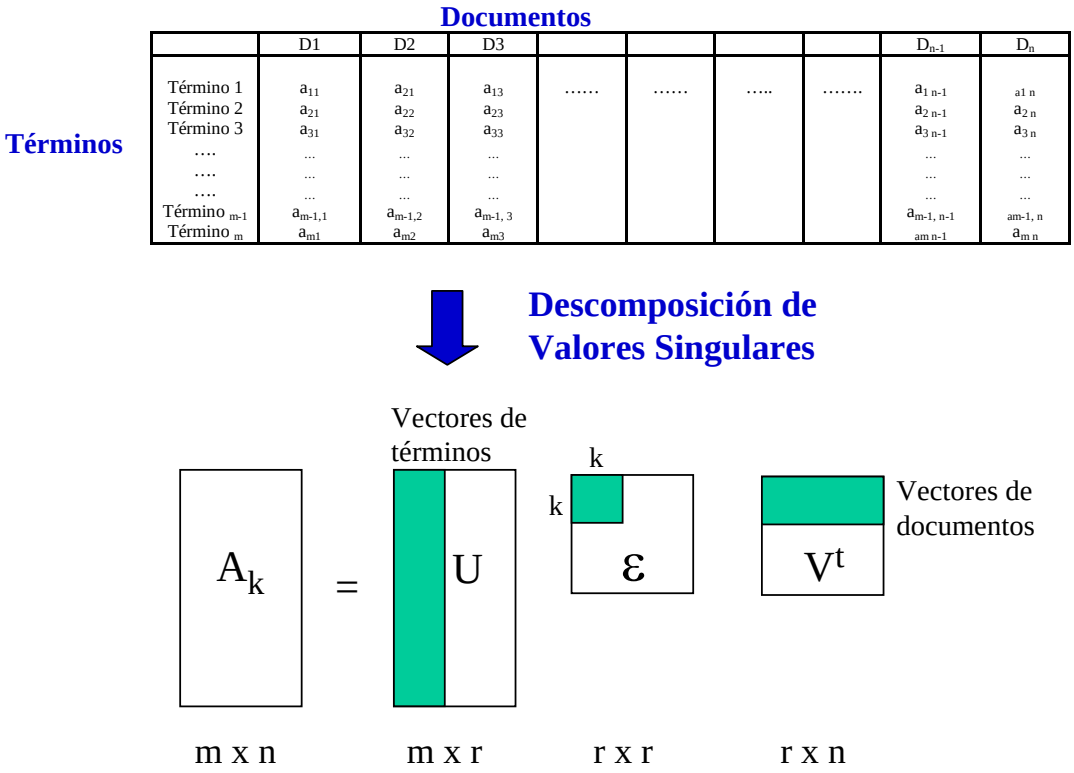


Figura 9. Método LSI.

### *2.3. Motores de Búsqueda e Indexación Comerciales*

El interés por los Sistemas de IR es tal que su estudio ha salido del entorno puramente académico llegando hasta el punto de que el DARPA (Defense Advanced Research Projects Agency) y la NIST (National Institute of Standards and Technology) patrocinan una conferencia denominada TREC (Text Retrieval Evaluation Conference). Ésta, en la actualidad, posee una base de datos estándar, con gigabits de información, que contiene sentencias de búsqueda y los resultados esperados para las mismas para que tanto los investigadores como empresas comerciales prueben sus sistemas.

En este apartado se presenta, de forma breve, las características más relevantes de los softwares comerciales más populares para el Almacenamiento y la Búsqueda de Información.

### 2.3.1. AltaVista Search (Altavista Internet Software)

**Arquitectura software:**

Está compuesto por 3 elementos fundamentales: el scooter, el indexador y el gestor de demandas. El scooter es un "robot" que recorre la red recogiendo las páginas HTML nuevas, modificadas o suprimidas. El scooter es configurable para no acceder a URLs no deseadas. El gestor de demandas recoge la petición del usuario y le muestra el resultado de la consulta.

Recursos: 13 ordenadores Turbolaser y un índice de más de 50GB gran parte en RAM.

**Arquitectura funcional:**

Los formatos de los documentos que puede explorar son el HTML y XML. Los formatos complejos se tratan a través de los filtros de la casa INSO.

Modo de indexación: indexación por texto libre, incluyendo palabras vacías. Los documentos de más de 4Mb los fragmenta. Indexa las palabras teniendo en cuenta la referencia del documento (URL) y, también, puede guardar la posición de las palabras en el documento. La información escrita en la etiqueta META de los documentos HTML, se consideran palabras clave. Permite indexar documentos con estructura como las news (emisor, fecha, tema, resumen).

Modo de búsqueda: ofrece dos modos de búsqueda, simple y avanzada. Usa los mismos operadores: booleanos, caracteres comodín y de proximidad (NEAR). La diferencia está en que en la búsqueda simple, el análisis de pertinencia (estadístico) es implícito.

La pertinencia se calcula basándose en la presencia de las palabras en el título o al principio del documento, la frecuencia del término solicitado en el documento, la

poca frecuencia de las palabras en el índice y de la proximidad de las palabras de la pregunta en el documento.

<<Live Topic>> es un asistente, que suministra, para afinar las peticiones. Presenta una visión sintética del conjunto de respuestas construidas a partir de un análisis estadístico del contenido de los documentos y de una reagrupación en categorías de diversas palabras. Estas categorías están formadas por palabras que responden al contenido de los documentos encontrados.

### 2.3.2. ConText (Oracle)

**Arquitectura software:**

Está integrado dentro del propio motor de SGBDR en forma de procedimientos PL/SQL (parecido a ESQLC). Soporta juegos de caracteres de 8 y 16 bits en diferentes idiomas. El lenguaje que utiliza es una extensión del SQL recogido en el SQL3. Se pueden combinar peticiones de campos estructurados y no estructurados.

**Arquitectura funcional:**

Los documentos se pueden guardar en tablas relacionales o externamente, indicando su nombre y path de ubicación. Los formatos de los documentos que puede tratar son los suministrados por la casa Mastersolve.

Modo de indexación: permite indexación posicional del texto. Tiene lista de palabras vacías y no indexa letras acentuadas, sustituyen éstas por las no acentuadas. Dispone de indexación por palabras fonéticamente próximas. La técnica empleada es de Índices Inversos previo almacenamiento de las entradas del índice en las tablas del gestor.

Modo de búsqueda: ofrece búsqueda booleana, caracteres comodín, proximidad (NEAR), búsqueda fonéticas, búsqueda extensa (tolerante a la ortografía). Ofrece ponderación de los términos de búsqueda.

Integra la gestión de thesaurus de estructura conforme con la norma ISO 2788. Ofrece, pero solo en inglés, resumen automático (extrae las tres frases más significativas del texto) y clasificación lingüística a partir de un diccionario que

consiste en 1.000.000 de palabras agrupadas en 200.000 conceptos reagrupados en 2000 temas y casi 20.000 reglas lingüísticas.

### 2.3.3. ILEXI-QA (ERLI)

**Arquitectura software**:

ILEXI (Intelligent Linguistic Interface Query and Answer), es un conjunto de herrramientas destinadas a la búsqueda en lenguaje natural. Incluye un diccionario y una red semántica, un sistema de análisis gramatical, una herramienta de personalización de la red semántica y del vocabulario ILEXI-KA (Knowlodgen Architech), un módulo de administración y otro de interrogación.

**Arquitectura funcional:**

Lo incluyen ya los motores Search97 y Fulcrum, y también es posible su integración mediante un API.

A nivel funcional puede considerarse como una capa adicional al motor de indexación y búsqueda. Actúa en varias fases: primero, analiza la pregunta del usuario revisando los diccionarios y analizando su sintaxis. Seguidamente, analiza los parámetros destinados al motor para actuar sobre el cálculo de pertinencia. Después se envía al motor la petición y los parámetros generando la respuesta.

Efectúa análisis morfológico, sintáctico y semántico en base a tres diccionarios: diccionario general, temáticos y específicos de la empresa.

Modo de búsqueda: ofrece búsqueda booleana ponderada y configurable por el administrador.

Modo de indexación: Se tiene previsto integrar estas funciones lingüísticas a nivel de indexación.

### 2.3.4. RetrievalWare (Excalibur Technologies)

Excalibur desarrolla y comercializa dos productos: uno relativo a las técnicas de búsqueda de información por medio de redes neuronales aplicadas a la voz, a la

visión o al texto, resultante del análisis de sistemas biológicos y de su autoadaptación; y otro, relativo a la búsqueda de información por redes semánticas.

**Arquitectura software:**

RetrievalWare es un motor de arquitectura c/s, multiservidor y multibase. Está compuesto por diferentes módulos: seguridad, búsqueda, gestor de documentos (indexación de las novedades) e Introducción de los documentos (filtrado con filtros de MasterSolve).

**Arquitectura funcional:**

Modo de indexación: El proceso de indexación de texto sigue diversos procesos: la identificación de elementos en el texto (identificar las palabras reconociendo las fechas, nombres, números, etc.); seguidamente, se realiza un análisis morfológico y una búsqueda de expresiones idiomáticas; la tercera fase, se encarga de eliminar las palabras vacías y, la última, es la indexación textual y APRP (Adaptative Pattern Recognition Processing) basada en cadenas binarias.

La indexación textual utiliza la técnica de Índices Inversos mediante B-Trees, para lo que precisa en torno al 30% del espacio ocupado por los datos. En cambio el APRP solo requiere en torno al 5%.

La indexación por APRP codifica los términos en forma de notaciones binarias cuya longitud varía en función de lo que la red neuronal ya ha adquirido básicamente mediante la selección de las longitudes que se parecen a la que ha encontrado. Conceptualmente esto supone construir registros de índice sobre fragmentos de palabras de longitud variable. Se trata en síntesis, de una técnica de n-grams en la que el número n es variable y está optimizado en función de lo que la red neuronal haya adquirido como experiencia.

Modo de búsqueda: tiene dos modos de búsqueda: textual y multimedia.

- Modo de búsqueda textual. Permite búsqueda booleana, semántica y por reconocimiento de forma.

La búsqueda semántica se basa en el uso de diccionarios electrónicos que representan las redes semánticas durante la indexación y la búsqueda. La búsqueda

APRP es una tecnología basada en el reconocimiento de la forma mediante redes neuronales, que se aplican al código binario de los documentos. Es independiente del idioma.

- Modo de búsqueda multimedia: técnica de reconocimiento de formas fundamentadas sobre redes neuronales con posibilidad de optimizarlas según los tipos de documentos: iconográficos, fotografías, vídeos, huellas... Por ejemplo, en las imágenes se utilizan técnicas especiales de reconocimiento de contornos y de texturas (aplicando técnicas de análisis espectral); para vídeo, la construcción de muros de imágenes (se indexa la imagen, las diferencias y los objetos identificados).

### 2.3.5. Search97 (Verity Inc.)

**Arquitectura software:**

Arquitectura c/s sobre redes TCP/IP o en arquitectura distribuida de archivos compartidos. Trabaja con juego de caracteres 1 y 2 bytes.

**Arquitectura funcional:**

Search97 construye índices de búsqueda a partir de fuentes externas y de colecciones. Las colecciones reagrupan los índices de documentos con la misma estructura (lógica) y el mismo formato (físico): por ejemplo, documentos con los mismos campos a indexar. Las colecciones comprenden ficheros de estilos necesarios para la configuración del indexador así como ficheros de palabras vacías, diccionario de sinónimos, campos estructurados, reglas de optimización. Las colecciones se dividen en particiones limitadas cada una a 64.000 documentos. Esta fragmentación permite actualizar la indexación por partes y, por tanto, optimizar el rendimiento de la búsqueda. Una colección puede tener hasta 16 millones de documentos.

Los documentos indexados son filtrados por medio de los filtros de MasterSolve y los de KeyView.

Modo de indexación: utiliza la técnica de Índices Inversos y propone tres modos de indexación: texto completo, indexación de zonas e indexación estructurada. La

indexación por zonas, permite indexar fragmentos de documentos delimitados por etiquetas (lenguajes HTML, SGML, etc.). La indexación estructurada se refiere a la indexación por los campos definidos para representar al documento.

Modo de búsqueda: ofrece búsqueda booleana, por proximidad, búsqueda de expresiones en una frase o párrafo, búsqueda sobre fechas y datos numéricos, uso del comodín. También ofrece búsqueda fonética y búsqueda por raíces de términos. Además permite la búsqueda por conceptos. Este tipo de búsqueda se basa en la construcción de un árbol de conceptos que reagrupa jerárquicamente los términos o expresiones, asignando distintos pesos a las distintas ramas del árbol. Finalmente se ha integrado la solución iLEXI-QA para beneficiarse de las búsquedas lingüísticas.

### 2.3.6. SearchServer – Knowlegde Network (Fulcrum Technologies Inc.)

**Arquitectura software:**

Arquitectura c/s multiplataforma que funciona en entornos Unix, NT, Windows, OS/2 y Mac. Los datos utilizados por SearchServer para la indexación y la búsqueda se manipulan en tablas, en el sentido relacional del término.

**Arquitectura funcional:**

El catálogo (Tabla Data File) es una tabla que incluye, en columnas, los campos que caracterizan a los documentos fuente y en las filas, los registros de cada documento con la descripción del contenido bruto de cada campo. Los archivos de índice (Table Index File) son archivos secuenciales indexados que registran las palabras y su posición absoluta en el texto para cada una de las columnas definidas en el catálogo. La actualización de los índices se realiza en memoria caché mediante B-Trees.

Modo de indexación: En la indexación normal, los datos alfanuméricos se indexan por caracteres, aunque existe la indexación literal que actúa sobre la palabra completa. La información numérica y de fechas se indexa de forma numérica para aprovecharse de los operadores aritméticos. Cada índice utiliza un archivo de

palabras vacías. En las columnas del catálogo es donde se define de qué tipo es cada información (alfanumérica, numérica, fechas).

Modo de búsqueda: ofrece búsqueda booleana, caracteres comodín, proximidad (NEAR). Ofrece ponderación de los términos de búsqueda y las herramientas lingüísticas iLEXI-QA e INSO.

# Redes Neuronales Artificiales

# 3. Redes neuronales artificiales

*Este capítulo recoge una breve introducción a las redes neuronales explicándose sus conceptos básicos y características más relevantes y comunes, los métodos de ajuste más utilizados y los tipos de redes más representativos haciendo especial hincapié en las redes de clasificación.*

## *3.1. Introducción*

Las redes neuronales son estructuras computacionales que procesan información de forma distribuida y paralela por medio de un número de unidades, denominadas procesadores o neuronas, las cuales se comunican entre sí enviándose señales a través de las conexiones que las unen. En la figura 10, se puede observar la estructura básica de un procesador con la interpretación neurofisiológica de sus componentes, dado que, las redes neuronales se inspiran en la arquitectura del sistema nervioso.

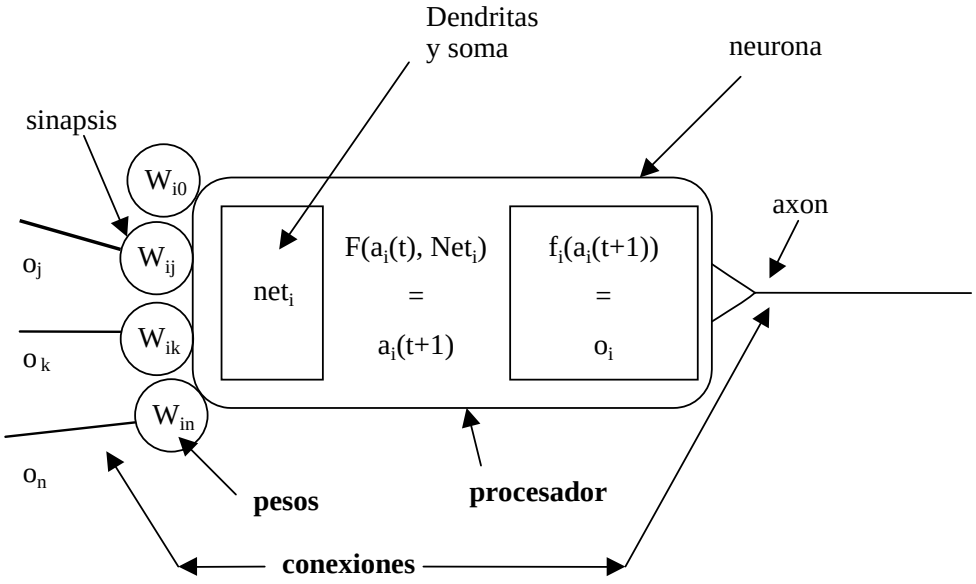


Figura 10.Componente básicos de un procesador $U_i$ y su analogía en la neurofisiología.

Una red neuronal está compuesta por un conjunto de procesadores. Las conexiones entre éstos determinan la arquitectura de la red. Generalmente la red está organizada en capas donde los procesadores de esas capas pueden realizar una determinada función como, por ejemplo, puede ser la capa de entrada y la capa de salida.

La tarea de procesado de una neurona es muy simple. El valor de entrada a una neurona se evalúa como una función de los pesos y las señales en sus conexiones de entrada. El caso más frecuente de esta función es el de la (ec. 4).

$$net_i = \sum_j w_{ij} \cdot o_j + w_{i0} \qquad \text{(ec.4)}$$

La activación de la neurona U en un tiempo t+1, $a(t+1)_i$, es función de la entrada y la activación previa de ésta, $F(net_i, a(t)_i)$, y la salida, $o(t)_i$, es función de la activación de la neurona es ese instante, $f(a(t+1))_i$). Ambas funciones, F y f, pueden ser lineales o no lineales. Normalmente se utilizan funciones continuas (sigmoidal) del tipo $f(x) =1/(1+e^{-ax})$, ya que son derivables y ésto facilita el cálculo en el aprendizaje. Generalmente, la función de activación F, es la función identidad por lo que el estado de activación de una neurona en t+1 coincide con el valor de la entrada de la misma. Según ésto, la salida de la neurona $o_i$ quedará según la expresión (ec. 5):

$$o_i(t+1) = f(net_i) = f(\sum_j w_{ij} \cdot o_j(t) + w_{i0}) \qquad \text{(ec.5)}$$

Para llevar a cabo la computación de la red, se presenta en la capa de entrada un vector de datos, éste activará los procesadores de la capa siguiente y así sucesivamente hasta llegar a la capa de salida donde se recogen los valores obtenidos. La transformación de la entrada en la salida está determinada por las funciones de activación y salida y los pesos de las conexiones. El proceso de cambio de estos pesos recibe el nombre de aprendizaje.

Existen dos procesos de aprendizaje, supervisado y no supervisado. En el aprendizaje supervisado se tiene un conjunto de vectores de entrada con su valor de salida correspondiente. El proceso actúa del siguiente modo, se presenta a la red los patrones de entrada, se calculan sus salidas y se las compara con el valor de salida esperado modificándose los pesos de las conexiones de forma que el patrón de salida de la red se ajuste al correcto. En el aprendizaje no supervisado no se tiene un

conjunto de respuestas correctas con lo que la red debe descubrir las regularidades, correlaciones y patrones explotando la redundancia de los datos de entrada. Ejemplos de aprendizaje supervisado es el backpropagation, y de aprendizaje no supervisado, denominado también autoorganizativos, son los derivados del principio Hebbiano.

Las redes neuronales son, en definitiva, una familia de sistemas de cómputo que permiten resolver, de otro modo, problemas tradicionales como son la aproximación a funciones, la categorización y el reconocimiento de patrones. Los mejores resultados que se han obtenido utilizando estos métodos se han dado en aquellas aplicaciones en las que se dispone de un gran volumen de ejemplos de entrenamiento pero las reglas para llegar a la solución de los mismos son difíciles de determinar, como por ejemplo: control de robots, reconocimiento de voz, reconocimiento de patrones, filtrado de señales, etc.

## *3.2. Características generales de las redes*

Desde el punto de vista computacional, las redes difieren de los algoritmos tradicionales en varias características, que se pueden resumir en las siguientes:

- Aprendizaje adaptativo. Las redes neuronales tienen capacidad para aprender a realizar tareas a partir de un entrenamiento o una experiencia inicial.

- Autoorganización. Las redes neuronales tienen capacidad para autoorganizar la información que reciben durante el aprendizaje y/o la operación.

- Tolerancia a fallos. En teoría, las redes neuronales son más tolerantes a los fallos en dos aspectos: uno, respecto a los datos, un pequeño cambio en la señal de entrada no afecta drásticamente en la salida; y, dos, respecto a la función que realiza la red, ya que aunque se destruya en parte, la red sigue operando si bien con cierta degradación. En la práctica, esta característica resulta poco relevante principalmente si las funciones de activación son discontinuas y/o la arquitectura está muy optimizada; pues, en el primer caso, pequeños cambios en las entradas pueden suponer cambios significativos en las salidas y, en el

segundo, la pérdida de algunos procesadores conduciría a un funcionamiento ineficiente.

## 3.3. Tipos de redes

En este apartado se comentan distintas clasificaciones de las redes según su topología, mecanismo de aprendizaje, tipo de asociación entre la información de entrada y salida, y por último, la forma de representación de estas informaciones.

Según su topología, disposición de neuronas en la red, se puede hablar de redes monocapa o multicapa. En las redes monocapa más útiles se establecen conexiones laterales entre las neuronas que constituyen la red. En la redes multicapa, las neuronas se agrupan en varios niveles realizando cada uno de ellos una función. Éstas últimas, a su vez, se pueden clasificar en redes con conexiones hacia adelante (feedforward) y redes con conexiones hacia adelante y hacia atrás (feedforward/feedback). Las redes feedforward/feedback se pueden considerar una variante de las monocapa.

Según el tipo de asociación entre la información de entrada y salida se puede hablar de redes heteroasociativas y redes asociativas. Una red heteroasociativa se considera a aquella que computa cierta función, que en la mayoría de los casos no podrá expresarse analíticamente, entre un conjunto de entradas y un conjunto de salidas, correspondiendo a cada posible entrada una determinada salida. Por otra parte, una red autoasociativa es una red cuya principal misión es reconstruir una determinada información de entrada que se presenta incompleta o distorsionada (le asocia el dato almacenado más parecido).

Según la representación de la información de entrada y salida de la red se pueden distinguir redes de variables continuas y redes de variables discretas.

## 3.4. Modelos de redes neuronales

En este apartado se presentan tipos concretos de redes, describiéndose su arquitectura, funcionamiento y aplicaciones donde se han utilizado.

### 3.4.1. Perceptrón.

En el año 1958, Rosenblatt creó la primera red neuronal artificial con capacidad para aprender a reconocer patrones sencillos y linealmente separables, que tomó el nombre de perceptrón. Años más tarde (1986), Rumelhart, Hinton y Willians popularizaron el método para que una red aprendiera la asociación que existe entre los patrones de entrada a la misma y las clases correspondientes utilizando varias capas de neuronas de tipo perceptrón, conocida como BACKPROPAGATION.

Básicamente, en este tipo de red, los procesadores se agrupan en capas y éstas están ordenadas, generalmente, de forma que los procesadores de cada una tomen la señal de las precedentes (alimentación hacia delante). Cada procesador calcula una función de la suma ponderada de sus entradas. Habitualmente esta función es la logística o la tangente hiperbólica.

El entrenamiento en este modelo de red es supervisado. Se basa en la generalización de la regla Delta de Widrow-Hoff o regla del mínimo error cuadrado medio (LMS) utilizada en los primeros modelos de red: ADALINE y MADELINE. El cambio de pesos para ajustar la red es proporcional a la delta o diferencia entre la salida deseada y la obtenida. Este método requiere que la función de activación sea continua y derivable.

Los problemas que presenta este tipo de entrenamiento radica en que se busca minimizar la función de error, lo que puede dar lugar a obtener mínimos locales o puntos estacionarios. Por otra parte, no existe ninguna regla para determinar el número de procesadores ni de capas necesarios para resolver cada problema concreto, solamente las pruebas propias y la experiencia de numerosos autores pueden ayudar.

Entre sus aplicaciones se encuentran: la codificación de información, el reconocimiento óptico de caracteres, el reconocimiento del lenguaje hablado, la compresión/descompresión de datos,...

### 3.4.2. Sistemas de memoria asociativa

En 1982, Hopfield desarrolló la primera red neuronal artificial autoasociativa, que recibió su nombre.

Básicamente este modelo de red consta de una sola capa de neuronas en la que cada neurona se encuentra conectada a todas las demás, pero no consigo misma. Además los pesos asociados entre pares de neuronas son simétricos.

El funcionamiento es el siguiente, durante el entrenamiento varios patrones diferentes son almacenados en la red, como si de una memoria se tratase. Posteriormente, si se presenta a la entrada alguna de las informaciones almacenadas, la red evoluciona hasta estabilizarse, ofreciendo en la salida la información almacenada, que coincide con la presentada en la entrada. Si, por el contrario, la información de entrada no coincide con ninguna de las almacenadas, por estar distorsionada o incompleta, la red evoluciona generando como salida la más parecida. El aprendizaje es no supervisado de tipo hebbiano.

Las limitaciones que ofrece este tipo de red se refieren a la cantidad limitada de datos que se pueden almacenar (para pocas informaciones se requiere un gran número de neuronas) y a la necesidad de que estos datos sean ortogonales entre sí, es decir, que sean lo suficientemente diferente entre ellos. A pesar de estas limitaciones, su éxito se encuentra en la fácil implementación en tecnología integrada VLSI.

Como aplicaciones de este sistema pueden citarse: el control de motores, la resolución de problemas de optimización, el reconocimiento de imágenes y voz.

### 3.4.3. Redes autoorganizativas

En 1982, Teuvo Kohonen desarrolló un modelo de red con capacidad para establecer relaciones, desconocidas previamente, entre conjuntos de datos basándose en el principio de formación de mapas topológicos. Se denomina autoorganizativo porque no se conoce a priori a qué grupo debe pertenecer cada punto de la muestra.

Este modelo tiene dos variantes, la denominada LVQ (Learning Vector Quantization) de tipo unidimensional y SOM (Self-Organizing Map) de tipo bidimensional.

El funcionamiento de la red es relativamente simple. Cuando se presenta a la entrada un patrón, cada una de las neuronas de la capa de salida la recibe a través de sus conexiones y la neurona que representa a la clase a la que pertenece dicho patrón será aquella que mayor valor tome. Además, como ante una entrada parecida se activa la misma neurona de salida u otra cercana a la anterior, se garantiza que las neuronas topológicamente próximas sean sensibles a entradas físicamente similares.

El aprendizaje es no supervisado de tipo competitivo. Las neuronas de la capa de salida compiten por activarse y sólo una de ellas permanece activa ante una entrada. Se ajustan los pesos de las conexiones de la neurona que haya resultado vencedora. También se ajustan los pesos de las conexiones de las neuronas próximas a la vencedora pero en menor medida.

Las limitaciones que ofrece este modelo son la duración del proceso de aprendizaje y la imposibilidad de aprender nuevos datos sin tener que volver a repetir completamente el proceso de aprendizaje.

Esta red se caracteriza porque en sus parámetros de la red queda representada la geometría del problema (topología rectagular, hexagonal,...), característica que es explotada en el diseño de las interfaces de usuario.

Estos sistemas se han usado con éxito para reconocimiento de patrones (voz, texto, imágenes, señales...), compresión de imágenes, resolución de problemas de optimización y en sistemas de IR.

### 3.4.4. Redes estocásticas

Las redes estocásticas están compuestas por neuronas que tienen función de activación no determinista, cuya salida se obtiene de forma probabilística y utilizan mecanismos de aprendizaje también estocásticos.

Las redes estocásticas más conocidas son la Máquina de Boltzman y la Máquina de Cauchy. Ambas tienen la misma arquitectura y funcionamiento, la diferencia se encuentra en la distribución de probabilidad de las neuronas.

La utilización de esta red ha sido inferior a la red de Hopfield y Backpropagation sobre todo por su lentitud en aplicaciones relacionadas con el reconocimiento de voz e imágenes y problemas de optimización.

### 3.4.5. Modelo de Teoría de Resonancia Adaptativa (ART)

En el año 1986, Grossberg y Carpenter desarrollaron este modelo con el fin de que la red aprendiera nuevos patrones sin que perdiera la información previamente recogida.

Básicamente esta red consta de dos capas entre las que se establecen conexiones hacia delante y hacia atrás. Las neuronas de la capa de salida tienen conexiones autorecurrentes y se encuentran completamente conectadas entre sí a través de conexiones laterales de inhibición que permiten establecer la competición entre ellas.

El aprendizaje en el modelo ART es no supervisado de tipo competitivo, por lo que no se distingue de la etapa de entrenamiento y la de funcionamiento.

Cuando entra en funcionamiento una red ART se produce una búsqueda a través de los prototipos almacenados y asociados a cada clase o categoría, con la pretensión de encontrar uno lo suficientemente parecido a la información presentada a la entrada de la red. Si no existe esta categoría, se crea una nueva, de la cual es prototipo dicha información.

La limitación que presenta este tipo de red es su gran sensibilidad al tipo y orden de información aprendida y al parámetro que indica la creación de una nueva clase.

Estas redes se han utilizado para el reconocimiento de imágenes y el reconocimiento de señales analógicas.

### 3.4.6. RAM Networks

Existen otros tipos de redes neuronales conocidas por el nombre de "RAM-based", "n-tuple based" o "Weigthless Neural Networks" [4,54] que se han utilizado en tareas de reconocimiento de patrones, siguiendo la línea de las redes autoasociativas.

Estas redes constan de procesadores binarios en la entrada y en la salida y carecen de pesos en las conexiones entre nodos. Las funciones neuronales se almacenan en tablas de búsqueda, *"look-up tables"*, que son implementadas fácilmente en memoria RAM. En vez de ajustar pesos, el aprendizaje consiste en modificar los contenidos de esta tabla, proceso que es rápido y flexible.

La velocidad en el proceso de aprendizaje se debe a la existencia de independencia mutua entre los procesadores ya que un cambio en la entrada no afecta a los patrones ya aprendidos.

Por el contrario, estas redes no permiten la generalización (capacidad de dar salida correcta a patrones que no conocen) aunque de algún modo se puede implementar considerando la distancia Hamming entre los patrones existentes.

En [37], se presenta una evaluación comparativa entre los resultado con la técnica de Índices Inversos y la respuesta dada por una red neuronal sin pesos aplicado a Information Retrieval, en la que se concluye que la red, a pesar de tener un mayor coste de memoria, ofrece una velocidad de respuesta mayor, y ésta se ve incrementada cuantas más palabras a buscar se soliciten. En cuanto a la indexación de más documentos, es decir, el aprendizaje de la matriz términos x documentos, indica que podría resolverse, sin un gran coste de memoria, haciendo uso de almacenamiento mediante códigos superpuestos (superimposed storage) [27].

## *3.5. Redes para clasificación*

Existen diversas arquitecturas de red orientadas a tareas de clasificación como son el perceptrón multicapa (ver apartado 3.4.1), las redes competitivas (autoorganizativas) y las redes de respuesta radial.

En las redes competitivas los procesadores se agrupan formando clases. Para cada punto de entrada solo ofrecen una respuesta no nula, señalándose así la clase a la que pertenece.

Las redes de respuesta radial, a diferencia de las competitivas, ofrecen una salida continua. Todos los procesadores pueden tener respuesta, unos más alta y otros más baja. Para saber la clasificación de cada punto se asignan las categorías según las respuestas de los procesadores correspondientes, para cada entrada la categoría es la del procesador con respuesta mayor. Lo más habitual son redes de una sola capa, aunque es posible construir redes arbitrarias. La respuesta se evalúa mediante una función radial que suele ser del tipo (ec. 6) :

$$e^{-\sum \frac{(p_{1i}+y_i)^2}{p_{2i}^2}} \quad \text{(ec. 6)}$$

donde, $y_i$ es el valor de la entrada *i* en la neurona y $p_{1i}$ y $p_{2i}$ son los pesos en la conexión.

Estas redes guardan cierta similitud con los perceptrones de los que se diferencian en la función de activación (función de base radial) y en las operaciones que se realizan en las conexiones (suma en vez de producto).

Las redes de base radial pueden requerir más neuronas que una red del tipo perceptrón multicapa hacia delante con funciones de activación tagsig o logsig. Esto se debe a que las neuronas sigmoidales pueden tener salidas en una región más amplia del espacio de entradas, mientras que las neuronas radiales solo responden a una región pequeña.

### *3.6. Algoritmos constructivos*

Existen otras alternativas a las redes clásicas propuestas para clasificación, en las que su diseño se basa en encontrar la mejor arquitectura a través de la técnica de prueba y error, estos son los algoritmos constructivos de redes neuronales.

La característica de estos algoritmos es que construyen una red adaptada a cada problema. El primer algoritmo constructivo fue el algoritmo Tiling, de ahí surgieron el Cascade Correlation, Upstart, Offset y GAL entre otros.

Estos métodos, enfocados generalmente a resolver problemas de clases mútuamente excluyentes, construyen redes usando procesadores con salida continua. Para cada patrón de entrada, solo una unidad de salida se activa.

El proceso de aprendizaje suele ser un poco más largo en estos métodos ya que deben entrenar varias redes de distintos tamaños con el fin de encontrar un error mínimo global óptimo. Aunque este problema puede reducirse si sólo se ajusta un número de pesos pequeño en cada paso, como es el caso de Cascade Correlation que solo ajusta los pesos de la unidad oculta añadida con relación a la capa de entrada y salida.

En [65], se presentan varios algoritmos constructivos propuestos en la literatura para resolver problemas de 2 clases y su modificación para aprender a clasificar múltiples clases excluyentes. En [80], se presenta un algoritmo de aprendizaje adaptativo para tareas de clasificación de varias clases excluyentes con procesadores binarios. En [19], se encuentra descrita la red Hiperbandas, que no solo clasifica sino que además permite el solape entre clases, es decir, que un patrón de entrada puede pertenecer a varias clases.

### 3.6.1. Red Neuronal Hiperbandas

La Red Neuronal Hiperbandas [19], es una red que se construye ex profeso para clasificar un conjunto de datos de entrada sin interacción del usuario y con error cero. Esta red, dado que no tiene salidas exclusivas, es adecuada para situaciones donde los patrones de entrada se agrupan formando clases que pueden estar solapadas.

**Estructura de la Red**

La red, ver Figura 11, se compone de 4 capas: la capa de entrada, la capa de hiperbandas, la capa de subconjunto y la capa de salida.

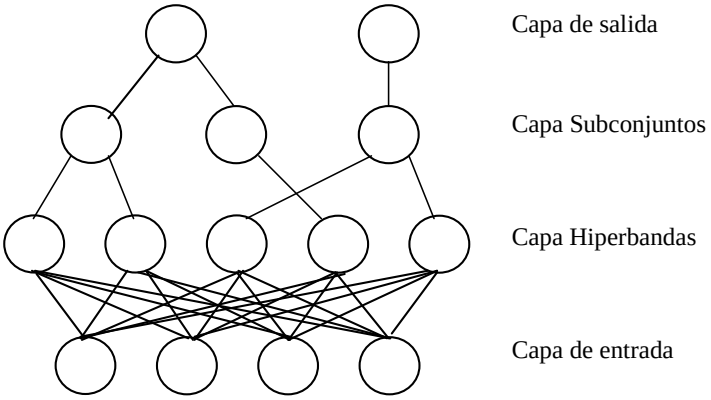


Figura 11.Arquitectura de Red Hiperbandas

La capa de entrada tiene tantos procesadores como variables de entrada tenga el problema a resolver, a los que se añade una unidad para considerar el término independiente de la ec.4.

La capa de hiperbandas consiste de un número variable de procesadores completamente conectado con cada unidad de la capa de entrada, con un único peso en cada conexión. La entrada de red de cada procesador es el producto escalar de sus pesos y los valores de entrada. La salida de cada procesador es el resultado de aplicar la función impulso rectangular.

La capa de subconjunto tiene también un número variable de procesadores que reciben conexiones de la capa de hiperbandas. Estas conexiones no tienen pesos. Cada procesador de esta capa recibe la entrada de varios, no todos, los procesadores de la capa de hiperbandas y produce como respuesta el mínimo de los valores de sus entradas.

La capa de salida tiene tantos procesadores como distintas clases tenga el problema. Cada unidad de salida recibe una o varias conexiones de la capa de subconjunto, tampoco estas conexiones tienen pesos y calcula el resultado como el máximo de los valores de sus entradas.

La justificación de esta arquitectura de red se fundamenta en que permite la interpretación de tipo geométrico que se resume a continuación.

Cada neurona de la capa de hiperbandas identifica a un hiperplano del espacio, cuya posición, queda identificada por la dirección del vector de pesos y su amplitud (anchura), por el módulo de dicho vector. Dado que en un espacio n-dimensional, cualquier subconjunto de puntos que es limitado con hiperplanos, puede obtenerse como la intersección de hiperbandas, esta intersección queda representada en la red por cada unidad de la capa de subconjuntos. Por último, cada unidad de la capa de salida representa la unión de diferentes subconjuntos.

**Algoritmo de construcción de la red**

Antes de describir el algoritmo de construcción de la red, se debe indicar que hay dos tipos de errores que la red puede producir:

- Error de tipo A, se tenga un 1 donde debiera ser 0.

- Error de tipo B, se tenga un 0 donde debiera ser 1.

Este proceso comienza sin unidades y procede de acuerdo a los siguientes pasos:

1. Se añade la primera unidad de salida y también una unidad en la segunda y en la tercera capa. Después de optimizar los pesos de la segunda capa, se comprueba si existen errores de tipo A.

2. Si los hay, una nueva unidad en la capa de hiperbandas se conecta a la unidad de la tercera capa y se ajustan sus parámetros. Si sigue habiendo errores se añadirán más unidades en la segunda capa, una a una. Después se comprobará si existen errores de tipo B.

3. Si se encuentran, se crea una nueva unidad en la tercera capa y se conecta a la unidad de salida y a una nueva unidad de la segunda capa. Se procede a la optimización y se comprueba si hay errores de tipo A repitiéndose el paso 2 si es necesario. En el caso de que queden errores de tipo B, se repite este paso de nuevo.

Para la próximas salidas el proceso es el mismo, con la diferencia de que antes de añadir unidades en las capas 2 y 3, se prueban las unidades creadas previamente y, si

el error se reduce, el proceso las toma. Otra posibilidad es la presentada en [19], donde las unidades previas son comparadas con una nueva unidad y se elige la mejor.

**Algoritmos de optimización**

Dado que la función de activación de la hiperbanda no es continua y que la geometría del problema es compleja se requieren algoritmos de optimización global. El ajuste de los parámetros de la capa de hiperbandas podría realizarse con cualquier algoritmo suficientemente potente, es decir, que puede estar sujeto a la evolución que tengan estos algoritmos en el panorama de la investigación internacional. Cara a una implementación concreta se propone la realización del ajuste en dos fases: un ajuste inicial, basado en el uso de un algoritmo genético y un ajuste más preciso después, basado en un algoritmo por trayectorias estocásticas [3].

## *3.7. Métodos de ajuste*

Los métodos de ajuste son un punto importante en la implementación de las redes neuronales. Originalmente se planteó como técnica de ajuste un descenso gradiente de la función de error según suma de cuadrados (ec. 7) :

$$E = \sum (y_i - \hat{y}_i)^2 \qquad \text{(ec. 7)}$$

Posteriormente se han propuesto otras (promedio de valores absolutos, entropía,...). El objetivo común es que la función de error se anule en el caso de predecir con exactitud la respuesta del sistema y sea positiva en cualquier otro punto, con mayor valor cuanto más lejos esté del mejor ajuste. La función de error elegida condiciona el resultado ya que, en general, el mínimo no será el mismo.

La función de error cuadrática es de uso común pero no resulta aconsejable cuando el ruido no es gaussiano o hay muchos valores erróneos en la muestra.

Los métodos de ajuste deben ser computacionalmente eficaces, deben converger, es decir, acercarse hacia un mínimo en forma estable y que este mínimo sea global, evitando valles locales que pueda tener la función objetivo. Los métodos de ajuste se pueden clasificar según:

- La amplitud de la búsqueda. Los algoritmos que realizan la búsqueda por todo el espacio de pesos se denominan globales, y aquellos que buscan el mínimo en las proximidades del punto inicial, se designan locales.

- El grado de derivadas que utiliza el algoritmo. Grado 0, no utilizan derivadas; grado 1, usan la primera derivada y, grado 2, utilizan la segunda. A mayor grado, el algoritmo es más veloz y complejo.

Existen diversos métodos de optimización de grado cero y búsqueda global, como son, el algoritmo de cristalización, algoritmos genéticos, algoritmos de perturbación, etc. que se pueden utilizar para la obtención preliminar de los pesos a partir de los cuales se aplique otro método de ajuste más eficaz (basado en derivadas).

A continuación se describen los métodos de ajuste más comúnmente utilizados en el ámbito de las redes neuronales.

### 3.7.1. Algoritmos genéticos

El algoritmo genético [17,57] se basa en la teoría de la evolución de las especies, según la cual, las poblaciones evolucionan en la naturaleza de acuerdo con los principios de la selección natural y la supervivencia de los más fuertes (postulados por Darwin).

El algoritmo comenzará con una población inicial de individuos (pesos), los cuales se seleccionarán, generalmente, de dos en dos para cruzarse y reproducirse. Una vez mutados constituirán la nueva generación de individuos. La selección de padres se efectuará al azar por medio de un procedimiento que favorezca la selección de los individuos mejor adaptados, aunque, de vez en cuando, se seleccionen también los menos adaptados (basado en la ruleta sesgada).

Si el algoritmo ha sido correctamente implementado, la población evolucionará a lo largo de generaciones sucesivas de tal manera que la adaptación media extendida entre todos los individuos de la población, así como la adaptación del mejor individuo se irán acercando hacia el óptimo global.

En forma de pseudocódigo, estos algoritmos actúan del siguiente modo:

Generar una población inicial

Computar la función de evaluación (adaptación) de cada individuo

MIENTRAS NO terminado HACER

PARA tamaño de la población HACER

Seleccionar dos individuos de la anterior generación para el cruce (según una probabilidad de selección proporcional a la función de evaluación del individuo).

Cruzar con cierta probabilidad los dos individuos obteniendo dos descendientes.

Mutar los dos descendientes con cierta probabilidad (generalmente pequeña, con objeto de incorporar nuevo material genético o recuperar individuos buenos perdidos durante la selección)

Computar la función de evaluación de los dos descendientes mutados.

Insertar los dos descendientes mutados en la nueva generación

SI la población ha convergido ENTONCES

terminado=TRUE

FIN

FIN PARA

FIN MIENTRAS

Cuando este método se aplica al ajuste de redes neuronales, la población de individuos corresponde al conjunto de pesos que minimizan el error global de la red.

### 3.7.2. Descenso de gradiente

Es un algoritmo de grado 1 y búsqueda local. Se basa en que la variación de cada peso para descender el error es más eficaz si es en la dirección del gradiente. Para cada peso se obtiene según (ec. 8)

$$\Delta p_i = -K \frac{\partial E}{\partial p_i} \quad \text{(ec. 8)}$$

donde K es una constante que establece un compromiso entre velocidad y estabilidad. Si el valor de K es muy bajo, el algoritmo tarda mucho tiempo en converger y si es muy alto, el algoritmo llega a desestabilizarse. Para evitar este inconveniente, surgió el método descenso de gradiente con momento en el que el incremento del peso se calcula según (ec. 9):

$$\Delta p_i = -K \frac{\partial E}{\partial p_i} + K_1 \Delta p_{i-1} \quad \text{(ec. 9)}$$

De esta forma si la diferencia entre el incremento de peso de una iteración y la iteración anterior supera un determinado factor los pesos no son actualizados. El momento permite a la red ignorar pequeñas características de la superficie del error, permitiéndole salir de un mínimo local.

Existe una variante de este método en el cual el factor K varía durante el proceso de entrenamiento, ofreciendo una convergencia más rápida.

### 3.7.3. Descenso de gradiente conjugado

Es un algoritmo de grado 1 y búsqueda local. Este algoritmo trata de solventar la poca eficiencia que presenta el descenso de gradiente al avanzar en direcciones ortogonales. Las direcciones conjugadas son suficientemente decrecientes a pesar de que no sean las del gradiente y no son ortogonales. En cada momento se conoce el gradiente, $g_i$, y la dirección a tomar $d_i$. La primera dirección es la del gradiente cambiado de signo. La siguiente dirección se calcula según (ec. 10)

$$d_i^{n+1} = -g_i^{n+1} + \beta d_i^n \quad \text{(ec. 10)}$$

$$\text{donde,} \quad \beta = \frac{\sum g_j^{n+1}(g_j^{n+1} - g_j^n)}{\sum d_j^n (g_j^{n+1} - g_j^n)}$$

Existen otras formas de calcular β, Fletcher-Reeves actualiza este parámetro según (ec. 11):

$$\beta = \frac{g_n^T g_n}{g_{n-1}^T g_n} \quad \text{(ec. 11)}$$

Y Polak-Ribiére según (ec. 12)

$$\beta = \frac{\Delta g_{n-1}^T g_n}{g_{n-1}^T g_n} \quad \text{(ec. 12)}$$

En cada paso hay una minimización unidimensional a lo largo de la dirección a tomar. En esta minimización se pueden aplicar varios algoritmos como son el de Newton, el de Brent o el de Charalambous.

### 3.7.4. Newton. Levenberg-Marquardt

El método de Newton es una alternativa a los métodos de gradiente conjugado para una optimización más rápida. Es un algoritmo local de grado 2. Toma una aproximación hasta la segunda derivada del desarrollo en serie del error y busca el mínimo a partir de esa expresión. Como la función no será de segundo orden, el proceso será iterativo. En cada iteración el vector incremento de pesos se calcula según (ec. 13)

$$\Delta p = -Hg \quad \text{(ec. 13)}$$

donde g es el vector gradiente y H la matriz hessiana de segundas derivadas. El vector incremento de pesos puede actuar como tal o como dirección a lo largo de la cual realizar una minimización.

Dado que el cálculo de las segundas derivadas es más difícil de obtener, existen variantes de este algoritmo en los que se realiza una aproximación de H. Estos son: el algoritmo Gauss-Newton y Levenberg-Marquardt.

En el primero, se aproxima H según (ec. 14):

$$H = J^T J \quad \text{(ec. 14)}$$

donde J es la matriz Jacobiana cuyos componentes son la derivada primera de los errores de la red respecto a los pesos.

En el algoritmo Levenberg-Marquardt, se aproxima H según (ec. 15):

$$H = J^T J + \mu I \quad \text{(ec. 15)}$$

donde μ es un escalar que cambia en cada iteración. Decrece si el error decrece y aumenta en caso contrario.

### 3.7.5. Métrica variable. Cuasi-Newton.

Este método se basa en el de Newton pero evita calcular las segundas derivadas y la inversión matricial, mediante la aproximación de la matriz inversa de la hessiana. Obtiene una dirección de incremento de los pesos, a lo largo de la cual realiza una busca unidimensional. Esta dirección se calcula según (ec. 16)

$$d = -\tilde{H}g \quad \text{(ec. 16)}$$

donde g es el vector gradiente y H, la aproximación a la inversa de la matriz hessiana. Una aproximación a la matriz hessiana se puede obtener según (ec. 17):

$$\begin{gathered} d = p^n - p^{n-1} \\ A = \frac{1}{d(g_n - g_{n-1})} \\ B = (g_n - g_{n-1})^T \tilde{H}^{n-1}(g_n - g_{n-1}) \\ \tilde{H}^n = \tilde{H}^{n-1} + A\left[(1+AB)dd^T - \tilde{H}^n(g_n - g_{n-1})d^T - d(g_n - g_{n-1})^T \tilde{H}^{n-1}\right] \end{gathered} \quad \text{(ec. 17)}$$

Para comenzar el algoritmo se toma el gradiente negativo como dirección inicial y una matriz diagonal próxima a la identidad se asignará a la aproximación de la hessiana.

En cada paso hay una minimización unidimensional a lo largo de la dirección a tomar. En esta minimización se pueden aplicar varios algoritmos como son el de Newton, el de Brent o el de Charalambous.

### 3.7.6. Criterios de parada

Los algoritmos de ajuste funcionan por iteraciones, por eso se requiere algún criterio que indique detener el proceso y dar el resultado por bueno. Algunos de éstos pueden ser:

- Detener el algoritmo cuando el progreso es muy pequeño.

- Llegar a cierto valor de la función de error.
- Que todos los puntos de la muestran  tengan su error por debajo de un umbral.
- Que el gradiente, en algoritmos de grado 1 ó 2, sea casi nulo.

# Sistema IR propuesto

# 4. Sistema IR propuesto

*Este capítulo describe las características y requisitos del Sistema IR propuesto, los datos utilizados y los resultados obtenidos haciendo uso de herramientas de propósito general y de programas propios desarrollados expresamente. Se comparan los resultados obtenidos con la red propuesta, con la respuesta que se consigue con la técnica de Índices Inversos tradicional y, finalmente, se presentan las conclusiones más relevantes.*

En el capítulo 2, se ha visto que la mayoría de las técnicas de indexación relacionadas con redes neuronales se basan en utilizar un diccionario preestablecido (entre 100 y 300 términos) y representar los documentos como un vector de términos cuyos componentes responden, en alguna medida, a la importancia de ese término en el documento respecto a la colección a indexar, obteniéndose una clasificación de los mismos.

La técnica de indexación que se propone consiste en una red neuronal artificial, con adecuadas características de clasificación, con el objetivo de identificar cada documento de la colección con un procesador de la capa de salida, en vez de crear clusters de documentos, técnica que tradicionalmente se ha venido utilizando.

La estructura de la red puede resumirse mediante el esquema de la Figura 12: de forma que, en la capa de entrada, se dispone de los procesadores de entrada necesarios para introducir cada término del diccionario, en formato binario y, como salida, se incluyen tantos procesadores como documentos tenga la colección.

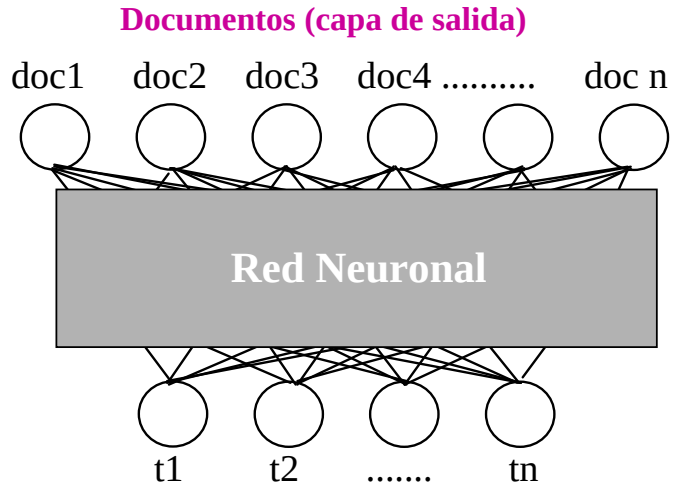


Figura 12.Red Neuronal propuesta.

### *4.1. Requisitos del Sistema.*

El sistema IR a implementar permite realizar las tareas que, a continuación, se citan:

- Indexar una colección de documentos especificada.
- Ampliar esta colección de documentos.
- Eliminar documentos existentes.
- Consultar documentos.

Los procesos de indexación y ampliación de la colección de documentos en primer lugar, requieren procesar los documentos suministrados al sistema y construir la matriz de entrenamiento (palabras x documentos) necesaria para la siguiente fase, la de aprendizaje de la red. En ésta, se realiza la fase de indexación propiamente dicha y consiste en el entrenamiento de la red para que aprenda los términos que presenta cada documento a incorporar.

El proceso de borrado de documentos consiste básicamente en la eliminación de la neurona de salida que identifique el documento junto con sus conexiones.

El proceso de consulta permite buscar documentos a partir de palabras del diccionario.

Además, no basta con que las operaciones del diccionario sean posibles, sino que se requiere que el algoritmo de indexación sea eficiente en tiempo y almacenamiento, es decir, ágil y compacto en el proceso de almacenar las asociaciones de términos x documentos. También, el algoritmo de búsqueda, deberá ser preciso y rápido en la respuesta.

### *4.2. Representación del diccionario*

Dado que la red neuronal no entiende directamente las palabras que componen un documento, se requiere realizar un proceso previo por el cual se extraigan los términos que componen cada documento y se codifiquen según un diccionario establecido. Dado que no había ninguna preferencia en la codificación de las palabras, se pensó en una codificación binaria, asignando a cada una de ellas, el número binario correspondiente a su posición alfabética en el diccionario previamente ordenado.

En caso de que se quisiera que el diccionario no tuviera un conjunto de palabras previamente establecido, sino que se construyera a partir de los propios documentos de la colección, se podría optar por alguna de las siguientes alternativas:

- Utilizar una técnica estadística que identifique los términos más relevantes por su frecuencia de aparición en la colección, por su especificidad u otra técnica de las propuestas en [70].

- Aplicar la técnica LSI (apartado 2.2.7) para obtener los términos conceptuales más relevantes de la colección.

- Hacer uso de una Base de Datos semántica que recoja las relaciones jerárquicas del lenguaje y con ella, construir una matriz simétrica con los términos de la colección, cuyo valor para cada pareja (x,y) sea proporcional a la distancia

dentro del árbol de relaciones que la base de datos establece, seleccionando los términos más próximos semánticamente [76].

## 4.3. Datos utilizados

La colección de documentos utilizada para el desarrollo de esta tesis corresponde a los artículos de la Sección I, II y III del Código Civil Español vigente en la fecha de publicación de esta tesis. Para trabajar de forma ágil, solventando así el problema de buscar documentos y su almacenamiento, cada artículo se ha considerado como un documento en formato ASCII. Así pues, se ha trabajado con un total de 295 documentos.

El diccionario confeccionado para el sistema se compone por un subconjunto de palabras extraídas del diccionario COES[1], de tal forma que entre el 60% y 75% de las palabras se encuentren en la colección de documentos.

## 4.4. Evolución del modelo propuesto.

Para la consecución del objetivo planteado se han analizado distintas arquitecturas de red neuronal. Primeramente, se trabajó con las clásicas redes neuronales utilizadas para clasificación como son las redes con Función de Base Radial (RBFs) y el Perceptrón Multicapa (MLPs). Dado que los resultados obtenidos con ellas no respondían a las expectativas previstas, finalmente se decidió hacer uso de algoritmos constructivos.

A continuación, se describen los diferentes modelos implementados y los resultados obtenidos, realizando una breve discusión de los mismos.

Para las diferentes pruebas se trabajó con distintos conjuntos de documentos y de términos que definían el diccionario. En la Tabla 1 se presenta la cuantificación de las diferentes colecciones utilizadas.

[1] **COES**. Diccionario español con más 53.000 términos desarrollado por Santiago Rodríguez y Jesús Carretero que pertenecen al Departamento de Arquitectura y Tecnología de Sistemas Informáticos de la Facultad de Informática de la Universidad Politécnica de Madrid.

| Nº ejemplo | Nº documentos | Nº términos |
|---|---|---|
| ejem_1 | 10 | 14 |
| ejem_2 | 60 | 30 |
| ejem_3 | 93 | 140 |
| ejem_4 | 194 | 140 |
| ejem_5 | 245 | 140 |
| ejem_6 | 295 | 140 |

Tabla 1. Colecciones de ejemplo.

### 4.4.1. Resultados con Redes de Respuesta Radial

Las primeras pruebas que se realizaron para conocer la viabilidad del proyecto, se llevaron a cabo con MATLAB. En particular, se hizo uso de las rutinas que suministra la toolbox Neural Networks para funciones de base radial, que son NEWRB y SIM.

Se comenzó con este tipo de función por ser una función de clasificación local que sería fácilmente interpretable.

La arquitectura de red neuronal de base radial suministrada por la toolbok consta de dos capas: una capa oculta de base radial y una capa de salida lineal que se utiliza para controlar el error mediante el ajuste de mínimos cuadrados.

La función NEWRB crea una red añadiendo iterativamente neuronas de base radial hasta que o bien, se alcance un determinado valor de error previamente especificado o bien, se alcance el número máximo de neuronas que se corresponde con el número de patrones de entrada. Existe un parámetro adicional, configurable, que es el parámetro sigma (ancho de la función radial) que determinará también el crecimiento de la red. La función SIM permite la ejecución de la red creada.

Se realizaron distintas pruebas, de las cuales se presenta aquí un breve resumen.

| **Consideraciones generales:** | | **Diccionario: 140 palabras**<br>**N º de Documentos: 20**<br>**Error : 0.0**<br>**N º de neuronas de entrada: 8** |
|---|---|---|
| Prueba #1 | sigma : 1 | N º neuronas radiales generadas: 140<br>Resultado: Aprende la muestra |

| **Consideraciones generales:** | | **Diccionario: 140 palabras**<br>**N º de Documentos: 20**<br>**Error : 0.02**<br>**N º de neuronas de entrada: 8** |
|---|---|---|
| Prueba #2 | sigma : 1 | N º neuronas radiales generadas: 139<br>Resultado: Aprende la muestra |
| Prueba #3 | sigma : 0.5 | N º neuronas radiales generadas: 81<br>Resultado: Aprende la muestra |
| Prueba #4 | sigma : 0.1 | N º neuronas radiales generadas: 81<br>Resultado: Aprende la muestra |

| **Consideraciones generales:** | | **Diccionario: 140 palabras**<br>**N º de Documentos: 93, 194**<br>**Error : 0.02**<br>**Nº de neuronas de entrada: 8** |
|---|---|---|
| Prueba #5 | sigma : 1 | Resultado: No consigue crear la red, problema de división por cero en el ajuste. |

A la vista de los resultados obtenidos se puede decir que:

- La red, con un número pequeño de documentos, es capaz de aprender la muestra con error cero. Cuando el tamaño de la colección aumenta, el proceso de ajuste falla y no consigue crear la red. Esto se debe a que la función lineal de la capa de salida no puede resolver la ecuación para el cálculo de los pesos, pues se produce "problemas" de división por cero.

- Al disminuir el parámetro sigma, el número de neuronas de la capa radial es menor, pero no suficientemente compacta. Se espera un número de neuronas en la capa oculta proporcional a la entrada para justificarse como alternativa a la técnica de Índices Inversos, ya que cuantos más procesadores ocultos sean necesarios, mayores requisitos de memoria tendrá el sistema.

Por lo que se puede concluir que la solución que ofrece este tipo de red solo es válida para un número pequeño de documentos. En este caso, además, la solución es trivial, pues crea tantos procesadores radiales como palabras distintas componen el diccionario. Por otra parte, se deduce que, la arquitectura de la red y su ajuste, no son válidos para las matrices de términos x documentos con las que se trabaja (colecciones de 93 y 194 documentos).

Antes de descartar las redes con respuesta radial, se hizo uso del software NODELIB, para construir un perceptrón multicapa con una capa oculta y respuesta radial. Después de realizar las mismas pruebas que con Matlab, se observó que la geometría de la función de error, con los datos que se trabajan, tiende demasiado fácilmente a mínimos degenerados. En estas pruebas, se utilizaron como función de error, la función de entropía y la función cuadrática.

### 4.4.2. Resultados con Perceptrón multicapa

Como consecuencia de los resultados obtenidos con las redes con función de base radial, se comenzó a trabajar con el perceptrón multicapa y la función de activación tangente hiperbólica.

Inicialmente, con programas de propósito general, como son el NODELIB, PDP++ y el SNNS, se diseñaron redes de tamaño pequeño para resolver el problema de 14 palabras y 10 documentos (ejem_1).

En estos casos se codificaron las palabras utilizando dos bits por dígito, considerando que, de este modo, se reforzaría el aprendizaje. Posteriormente se comprobó que los resultados eran comparables utilizando un solo bit por dígito.

A continuación se describe la arquitectura implementada y los resultados obtenidos, para este caso, con cada software:

**NODELIB**

a) Diseño de una Red con arquitectura 10-5-10 y función de error cuadrática.

   Resultado: El error promedio se estabiliza con un valor de 0,4 en 80 iteraciones. Falla en varias palabras. El método de ajuste utilizado fue cuasi-Newton con búsqueda de minimización lineal unidimensional por el algoritmo de Golden.

b) Diseño de una Red con arquitectura 10-5-10 y función de error cuadrática

   Resultado: El error promedio se estabiliza con un valor de 0.038 en 150 iteraciones. Falla también en varias palabras. El método de ajuste utilizado fue cuasi-Newton con búsqueda de minimización lineal unidimensional de paso fijo.

**PDP++**

a) Diseño de una Red con arquitectura 10-5-10 y función de error cuadrática

   Resultado: El ajuste se estabilizó en 700 iteraciones haciendo uso de pesos con inercia. Sólo hubo 1 fallo entre todos los de la muestra. Para el ajuste se utilizó el método de gradiente estocástico con permutación de muestra.

**PDP++**

b) Diseño de una Red con arquitectura 10-10-10 y función de error cuadrática

Resultado: El ajuste se estabilizó en 800 iteraciones. El error cae a cero, lo que quiere decir que los acierta todos. Se apreció que se ajusta mejor la red utilizando pesos con inercia.

c) Diseño de una Red con arquitectura 10-10-10 y función de error entropía

Resultado: La red falla 3 palabras en 150 iteraciones, reduciéndose a 1 en 1000.

**SNNS**

a) Diseño de una Red con arquitectura 10-5-10 y función de error cuadrática

Resultado: El ajuste se estabiliza en 100-200 iteraciones con un error promedio de 1.6. Falla bastantes palabras. El método de ajuste utilizado fue el gradiente estocástico con permutación de muestra.

b) Diseño de una Red con arquitectura 10-5-10 y función de error cuadrática

Resultado: El error alcanza el valor 0.1 en 200 iteraciones y no se reduce. Acierta toda la muestra. El método de ajuste utilizado fue el gradiente conjugado escalado.

A partir de las pruebas realizadas, se llegó a las siguientes conclusiones:

- Una red con 5 procesadores en la capa oculta puede llegar a aprender esta muestra sin error.
- El método empleado en la optimización puede tener una importancia definitiva.

- El mismo método de ajuste en distintos programas no ofrece el mismo resultado.
- La función de error utilizada no parece tener gran importancia.

Con objeto de poder actuar sobre el proceso de optimización, se abandonó el uso de herramientas software como las mencionadas y pasó a programar el perceptrón multicapa siguiendo la arquitectura propuesta. Se utilizaron dos métodos de ajuste, gradiente conjugado y Cuasi-Newton con minimización lineal unidimensional por el método Golden (mínimo por intervalos) y Brent (interpolación parabólica). Se trabajó con dos funciones de error: la cuadrática, por ser la más común, y la entropía, por tratarse de una función con respuesta entre 0 y 1, valores que toman los datos del problema objeto de esta tesis.

De las pruebas realizadas con el ejemplo anterior pero en este caso utilizando el programa desarrollado, se pudo concluir que:

- Puede considerarse que los resultados finales frecuentemente corresponden a mínimos locales.
- En una red de arquitectura 10 –5 –10 el error promedio todavía resulta algo elevado (> 0,2) aunque aproxima varios de los patrones perfectamente.
- Al aumentar el número de procesadores ocultos los resultados fueron más satisfactorios, siendo óptimos con 10 procesadores.
- El ajuste es más eficiente utilizando el método Cuasi-Newton y minimización unidimensional por el método de interpolación parabólica.
- El ajuste por el método de gradiente conjugado es más lento y ofrece peores resultados (error promedio de 0,5).
- El uso de la entropía como función de error no aporta mejores resultados que la función de error cuadrática.

- El resultado ofrecido por la red no es sensible a variaciones en la presentación de las palabras (orden por frecuencia en lugar de orden alfabético).

En la figura 13, para diferentes métodos de ajuste, se representa, gráficamente, la evolución, en la prueba ejem_1, del error cuadrático medio en relación al número de neuronas que tenga la red en la capa oculta.

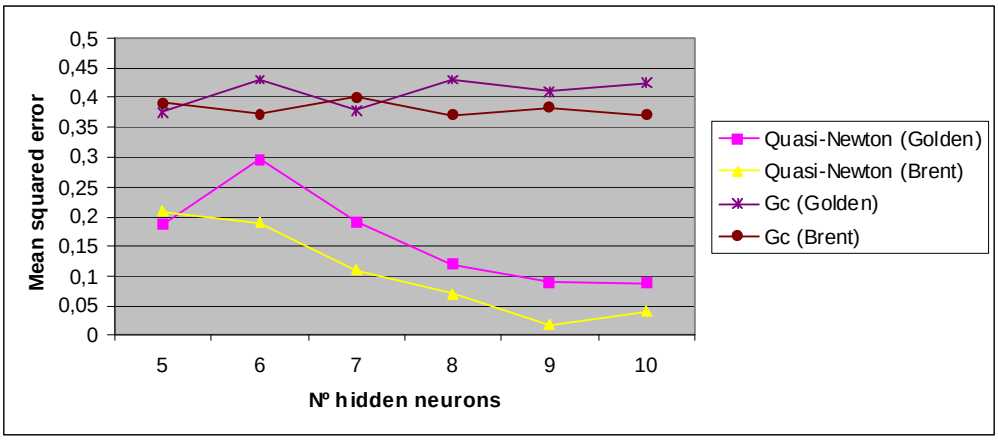


Figura 13.Comportamiento del error en relación al número de neuronas ocultas.

Para dar solución al tipo de problema planteado, se prosiguieron las pruebas haciendo uso de colecciones con un mayor número de documentos y un diccionario con más términos (ver Tabla 1).

Con el programa desarrollado, para el ejemplo 2 (30 palabras y 60 documentos), se implementó una red con una arquitectura de 10x20x60 procesadores, lo que implicaba 1480 pesos. El resultado obtenido no fue el esperado, pues la red no conseguía aprender la muestra. Para evitar este problema, se aumentó el nº de procesadores ocultos, y la red tampoco fue capaz de minimizar el error.

Por esta causa, para verificar que la respuesta del perceptrón implementado con el programa era fiable, se hicieron pruebas con las herramientas de propósito general ya mencionadas.

En la Tabla 2 se muestran resultados obtenidos haciendo uso de las herramientas PDP++ y SNNS que fueron similares a los que se obtuvieron con el programa confeccionado.

| *Ejemplo* | *Nº unidades ocultas* | *Error medio* | *Nº iteraciones* | *Resultado* |
|---|---|---|---|---|
| 1 | 5 | 0.003 | 400 | Aprende la muestra |
| 1 | 10 | 0.03 | 600 | Falla 1 de 10 |
| 2 | 20 | 0.5 | 1000 | Falla 5 de 30 |
| 2 | 40 | 0.91 | 1100 | Falla 7 de 30 |
| 3 | 16 | 0.66 | 1000 | Falla más de 40 sobre 93 |

Tabla 2. Resultados con Perceptrón Multicapa.

Dado que los resultados obtenidos fueron comparables, se concluyó que el problema residía en los datos. La geometría de éstos es compleja y conduce a mínimos locales no óptimos.

Además, los resultados muestran que un simple perceptrón requiere muchas unidades ocultas para resolver el problema. Dado que un requisito del sistema propuesto es que sea suficientemente compacto para justificarse como alternativa a la técnica de Índices Inversos, se descartó esta arquitectura.

### 4.4.3. Resultados con Cascade Correlation

Con objeto de conocer qué distribución de unidades ocultas requiere la red se hizo uso de algoritmos constructivos. Se trabajó con la implementación del algoritmo Cascade Correlation que suministra el programa SNNS. Se utilizaron diferentes métodos de entrenamiento: backprop momentum, quickprop y tacoma y los resultados no fueron positivos. Con los datos del ejemplo 3, la capa oculta llegaba a

crecer por encima de los 300 procesadores sin conseguir reducir el error promedio a valores inferiores a 3,8.

Se confirmó así que la selección de los procesadores ocultos por el criterio de máxima correlación entre la entrada y la salida no se adaptaba a las matrices de datos con las que se trabajaba. Las redes que resultaban tenían un número muy elevado de parámetros y el proceso de entrenamiento era muy largo. Por lo que esta arquitectura también se descartó.

### 4.4.4. Resultados con Red Hiperbandas

Dado que los resultados obtenidos hasta ese momento no eran satisfactorios respecto al objetivo planteado en esta tesis, se hizo necesario desarrollar una arquitectura de red cuyas características más relevantes fueran: la capacidad de aprender que un término puede encontrarse en varios documentos y el uso de un número reducido de parámetros. Esto llevó a trabajar con otro tipo de red, la red Hiperbandas [19].

#### *4.4.4.1. Creación y funcionamiento de la red.*

Después de probar la red con los distintos ejemplos (Tabla 1), se comprobó que la red que resultaba era muy compacta (pocas unidades ocultas) y que tenía la característica de aprender la matriz de términos x documentos con error cero, lo que confirmaba un primer punto de igualdad con respecto a la técnica de Índices Inversos, si bien esta última no requiere ningún proceso de optimización. Además, como, por definición de la red Hiperbandas, las capas de subconjunto y salida no tienen pesos, la memoria necesaria se reducía a almacenar únicamente los pesos de la capa de hiperbandas con la entrada.

En la Tabla 3 se presenta un resumen de los resultados obtenidos haciendo uso la red Hiperbandas, donde se ofrecen resultados promedios de las diversas ejecuciones realizadas.

Para implementar la red Hiperbandas se ha desarrollado un segundo programa, en lenguaje C/C++, y se ha trabajado sobre un Pentium II a 300 Mhz con 128 Mb RAM.

El comportamiento de la red se ha evaluado atendiendo a los siguientes parámetros:

1. El uso de memoria. Éste se calcula mediante la utilidad *sizeof()* del C/C++, que tiene en cuenta la estructura de datos necesaria para implementar la red Hiperbandas. En este caso, se compone de una estructura (struct de C) con 5 capas (témino independiente, entrada, hiperbandas, subconjunto y salida). Cada una de ellas consta, a su vez, de procesadores. Cada procesador tiene almacenado el valor a la entrada y a la salida del mismo, su identificación (ID del documento) y sus enlaces con los procesadores de la capa anterior. Cada enlace tiene el peso y la información del procesador que entra a él. Para más detalle ver la definición de la red en el Anexo I.

2. El tiempo de entrenamiento para almacenar la matriz de términos x documentos. Se calcula haciendo uso de la función *QueryPerformanceFrequency* de C++ encapsulada en una clase denominada *Duration*.

3. El tiempo de búsqueda de cada término en la colección. Se calcula haciendo uso de la clase *Duration* mencionada en el punto anterior.

| *Ejemplo* | *Arquitectura red capa 1 a capa 4* | *Tiempo de entrenamiento (seg.)* | *Memoria (bytes)* | *Tiempo de búsqueda (mseg.)* |
|---|---|---|---|---|
| Ejem_1 | 5-11-12-10 | 2.36 | 2620 | 0,08 |
| Ejem_2 | 6-17-21-60 | 7.56 | 5860 | 0,51 |
| Ejem_3 | 9-40-111-93 | 835.7 | 16820 | 3,35 |
| Ejem_4 | 9-46-134-194 | 908.94 | 22276 | 7,2 |
| Ejem_5 | 9-45-128-245 | 1352 | 23468 | 7,6 |
| Ejem_6 | 9-39-142-295 | 1641 | 24028 | 8,5 |

Tabla 3. Resultados con Red Hiperbandas.

En la arquitectura de la red de las tablas 3, 4, 5 y 6, el primer valor representa el número de procesadores de la capa de entrada incluyendo el término independiente.

A la vista de los resultados obtenidos, se puede concluir que:

- Cuanto mayor es la muestra, más compacta es la red, es decir, el nº de procesadores en la capa de hiperbandas no crece ni tan siquiera linealmente.
- El orden de presentación de patrones no influye de manera relevante en el resultado, las diferencias pueden considerarse efecto de la aleatoriedad del ajuste.
- La red aprende, por completo, la muestra. La red crece hasta obtener error cero para cada documento de la colección.
- El tiempo de entrenamiento no es excesivamente alto, considerando que se trabaja buscando mínimos globales y que se debe obtener error cero.
- El tiempo de búsqueda crece aproximadamente de forma lineal con el número de documentos de la colección.

Por estos motivos, se consideró que los resultados obtenidos al implementar el DRS propuesto con la red Hiperbandas eran aceptables, por lo que se prosiguió con la codificación del programa para incluir las opciones de incorporar nuevos documentos y borrar aquellos que el usuario seleccionase.

#### *4.4.4.2. Incorporación de nuevos documentos.*

Dado que se había modificado ligeramente la estructura de cada procesador, para incluir la identificación del documento al que hacen referencia, se realizaron de nuevo las pruebas con los ejemplos propuestos. Con el fin de comprobar que las hiperbandas y subconjuntos existentes no aumentaban y, por tanto, se constataba que el algoritmo de aprendizaje reutilizaba las bandas previamente creadas, a cada una de las redes generadas se le incorporaron nuevos documentos, aportando una colección idéntica a la que se utilizó para su generación. Además, se intentaba probar que la

indexación de una colección añadida con características similares no resultaba muy exigente en tiempo.

Los resultados obtenidos se presentan en la Tabla 4, con la letra *a* adjunta al número de ejemplo, se identifica el resultado obtenido para ese ejemplo en su ampliación.

| Ejemplo | Arquitectura red capa 1 a capa 4 | Tiempo de entrenamiento (seg.) | Memoria (bytes) | Tiempo de búsqueda (mseg.) |
|---|---|---|---|---|
| Ejem_1 | 5-11-12-10 | 1.855 | 2772 | 0.082 |
| Ejem_1a | 5-11-12-20 | + 0.025 | 3132 | 0.149 |
| Ejem_2 | 6-17-20-60 | 56.05 | 6240 | 0.52 |
| Ejem_2a | 6-17-22-120 | + 0.671 | 8472 | 1.01 |
| Ejem_3 | 9-49-122-93 | 897.74 | 20496 | 3.467 |
| Ejem_3a | 9-49-131-186 | + 248.5 | 24168 | 6.815 |
| Ejem_4 | 9-46-138-194 | 1264.476 | 23952 | 6.63 |
| Ejem_4a | 9-46-141-388 | + 442.23 | 31044 | 13.07 |
| Ejem_5 | 9-46-143-245 | 1342.34 | 25968 | 7.18 |
| Ejem_5a | 9-46-149-490 | + 540.45 | 35004 | 14.2 |
| Ejem_6 | 9-46-143-295 | 1512.40 | 27768 | 7.97 |
| Ejem_6a | 9-46-156-590 | + 570.01 | 38856 | 15.79 |

Tabla 4. Resultados con Hiperbandas al ampliar con la misma colección.

A la vista de los resultados indicados en la Tabla 4, se puede concluir que:

- Al incorporar una colección de documentos idéntica a la que ya tiene aprendida, el número de hiperbandas no crece, es decir, no aumenta el número

de pesos en la red, como se preveía inicialmente. Se aprecia que en algunas ocasiones, el número de subconjuntos sí se incrementa. Esto se debe a que, en la definición de la función de error de la red Hiperbandas, existe un factor que mide la dispersión de las entradas admitidas correctamente para errores de tipo A [19], que en determinados casos hace que utilizando un subconjunto nuevo el error sea menor que reutilizando uno que ya tiene aprendido. Dado que el hecho de tener este factor "*dispersión*", facilita el aprendizaje en tiempo y permite obtener una red más compacta y que, por otra parte, el consumo de memoria adicional no es relevante, se ha considerado un factor de diseño a estudiar con otras matrices de datos como línea de futuro.

- El coste del aprendizaje, en tiempo, dadas la características de reutilización de las hiperbandas y subconjuntos aprendidos, se ve reducido en aproximadamente una tercera parte del inicial. Lo que indica que se podría pensar en indexaciones on-line, siempre que la información a indexar sea similar a la ya aprendida. Por otra parte, no hay que olvidar, que este tiempo también se ve afectado por el número de documentos existentes en la red y el número de documentos nuevos a indexar.

- El coste de memoria, en estos ejemplos, se ve incrementado casi exclusivamente por el número de nuevos procesadores de salida.

- Se constata que el tiempo de búsqueda queda afectado directamente por el número de procesadores de salida. La única solución posible sería una implementación hardware o software masivamente paralela.

Con objeto de evaluar como afecta a la indexación el grado de similitud entre los documentos, se procedió a realizar la siguiente prueba.

A la red creada con el ejemplo 3, en primer lugar se le incorporó una colección de documentos que no tenía relación alguna con el diccionario establecido, col_a, (todo ceros en la matriz de términos x documentos); después, se le incorporó otra colección de documentos, la utilizada para su generación modificada en un 10 %, col_b; y, por

último, se indexó una colección de documentos para la cual la mayoría de los términos de la matriz de términos x documentos tomasen los valores contrarios a los de la matriz original, es decir, donde había 1 se cambió por 0 y viceversa, col_c.

La arquitectura de red y el tiempo necesario para su creación se presentan en la Tabla 5.

| *Ejemplo* | *Arquitectura red capa 1 a 4* | *Tiempo de indexación (mseg.)* |
|---|---|---|
| Ejem_3 | 9-43-119-93 | 837.41 |
| Ejem_3 + col_a | 9-43-119-186 | + 0.2975 |
| Ejem_3 + col_b | 9-43-140-186 | + 365.62 |
| Ejem_3 + col_c | 9-52-534-186 | + 1645.02 |

Tabla 5. Resultados de prueba de similitud.

A la vista de los resultados de la Tabla 5 se puede concluir que:

- El tiempo de indexación de nuevos documentos se ve afectado por el grado de similaridad que tenga la matriz de términos x documentos con relación a la colección ya indexada por la red. Si la matriz no aporta información nueva (col_a), el proceso es muy rápido; si la matriz tiene cambios pero éstos mantienen la geometría de los puntos de la matriz anterior, el proceso es ágil; mientras que si la matriz difiere por completo, se trata como una nueva indexación, la red no puede reutilizar la hiperbandas y subconjuntos ya aprendidos.

- La red, está orientada a trabajar con matrices con un mayor número de ceros (la función de activación de la capa hiperbandas responde a un pulso rectangular entre 0 y 1), por eso al indexar la matriz invertida requiere más procesadores ocultos. Desde el punto de vista operativo del sistema, se detecta que no se han elegido correctamente las palabras del diccionario, pues no permiten clasificar

claramente los documentos. Por otra parte, se comprueba que la red es capaz de indexar la colección, como ocurriría utilizando la técnica de Índices Inversos.

#### *4.4.4.3. Borrado de documentos.*

Se prosiguió con las pruebas de borrado de documentos, eliminándolos de uno en uno, para comprobar el tiempo que requiere cada una de estas operaciones y, si afectan al rendimiento de la red posteriormente. Se presentan los resultados de eliminar los mismos documentos en las dos colecciones más voluminosas, ejem_6 de 295 documentos y su versión ampliada, ejem_6a con 590 documentos.

En la Tabla 6 se recoge la arquitectura de la red, el documento a borrar con el número de enlaces que tiene el procesador que lo representa con la capa de subconjuntos y, el tiempo que requiere el sistema para borrarlo.

| *Ejemplo* | *Arquitectura red capa 1 a 4* | *Documento borrado (nº)* | *Tiempo de borrado (mseg.)* |
|---|---|---|---|
| Ejem_6 | 9-46-143-295 | doc.: 68 (14 enlaces) | 0.0871 |
| | | doc.: 115 ( 8 enlaces) | 0.0796 |
| | | doc.: 148 ( 0 enlaces) | 0.0427 |
| | | doc.: 295 ( 1 enlaces) | 0.0981 |
| Ejem_6a | 9-46-156-590 | doc.: 68 (14 enlaces) | 0.0913 |
| | | doc.: 115 ( 8 enlaces) | 0.0955 |
| | | doc.: 148 ( 0 enlaces) | 0.0427 |
| | | doc.: 295 ( 1 enlaces) | 0.1089 |
| | | doc.: 296 ( 1 enlaces) | 0.1089 |
| | | doc.: 460 ( 0 enlaces) | 0.1191 |
| | | doc.: 532 (14 enlaces) | 0.1827 |

Tabla 6. Resultados al borrar documentos.

A la vista de los resultados se puede concluir que:

- El tiempo de borrado para un mismo documento, en ambas colecciones, aumenta ligeramente (en torno a un 10%) debido al número de procesadores de salida, es decir, al tamaño de la colección.

- El tiempo de borrado se ve afectado por la posición que tiene el procesador que representa al documento en la estructura de datos definida, el número de enlaces que éste tenga con la capa de subconjunto y por el número de documentos de la colección.

- El borrado de documentos no repercute en el funcionamiento posterior de la red debido a las características del proceso de generación de la misma. Se ajusta la red para cada documento.

- La memoria liberada es proporcional al número de procesadores y enlaces eliminados. Cada procesador ocupa en memoria 36 bytes y cada enlace 24 bytes. La reutilización de esta memoria dependerá de la gestión propia del Sistema Operativo. Al respecto, cabe mencionar, que en la programación se ha utilizado el esquema estándar de punteros.

### 4.4.5. Comparación con Índices Inversos

Para poder llevar a cabo la comparación del Sistema de Almacenamiento y Recuperación Documental, haciendo uso de la red neuronal Hyperbandas descrita y la técnica de Índices Inversos, ha sido necesario desarrollar un tercer programa para recoger esta última.

Para implementar la técnica de Índices Inversos, la estructura de datos utilizada ha sido la de árboles binarios balanceados (B-Trees) [28]. Se ha elegido esta técnica porque, además de ser la más utilizada, minimiza el espacio de almacenamiento, permite una actualización de información sencilla y los tiempos de búsqueda son relativamente bajos.

La Figura 14 representa gráficamente la estructura de datos implementada. El árbol recoge en cada nodo un término del diccionario, junto con su clave, el número de veces que aparece el término en la colección y el número de documentos en los que se encuentra. Cada nodo apunta a un array con la identificación de los documentos que contienen el término.

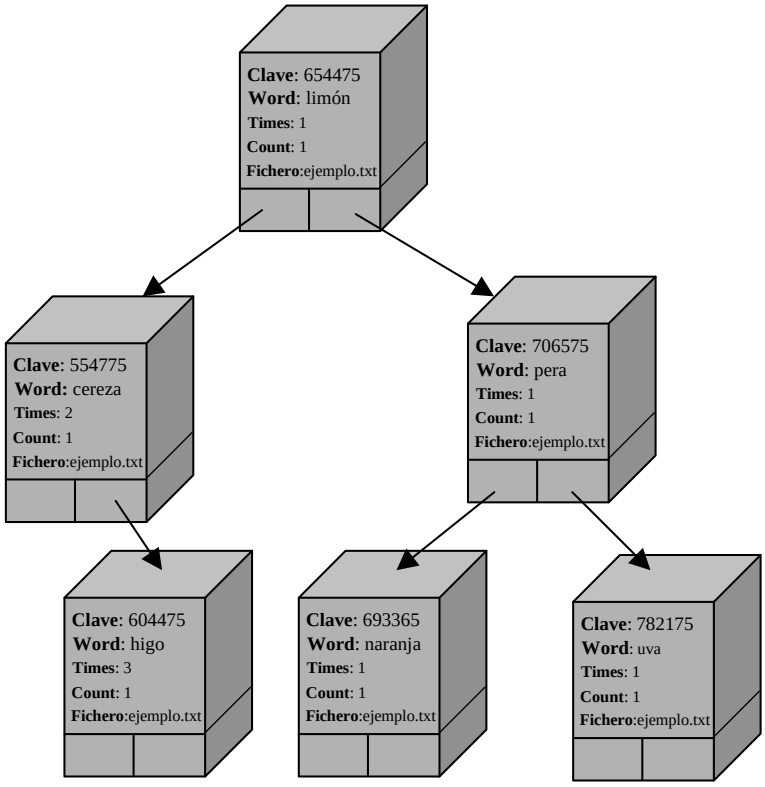


Figura 14.Estructura de datos B-Tree implementada.

#### *4.4.5.1. Indexación de documentos.*

Se efectuaron, con este programa, las mismas pruebas que previamente se habían llevado a cabo con la red Hiperbandas (Tabla 3). Se evaluaron los mismos parámetros con las mismas utilidades, estos son: tiempo de indexación, ocupación en disco y tiempo de búsqueda. Los resultados obtenidos haciendo uso de esta técnica de indexación se pueden ver en la Tabla 7.

| *Ejemplo* | *Tiempo de indexación (seg.)* | *Almacenamiento (bytes)* | *Tiempo de búsqueda (mseg.)* |
|---|---|---|---|
| ejem_1 | 0,06 | 760 | 2,21 |
| ejem_2 | 0,19 | 2160 | 2,30 |
| ejem_3 | 0,411 | 10192 | 2,11 |
| ejem_4 | 0,871 | 14548 | 2,19 |
| ejem_5 | 1,082 | 15952 | 2,11 |
| ejem_6 | 1,262 | 20008 | 2,25 |

Tabla 7. Resultados con B-Tree.

En la comparación de ambas estructuras de datos, Tabla 3 y Tabla 7, se puede concluir que:

- El DRS implementado con el B-Tree no requiere una fase de entrenamiento propiamente dicha, como la que se entiende en redes neuronales, sino que básicamente, construye la estructura a partir del diccionario establecido y actualiza la lista de documentos que tiene cada palabra. Esta es la razón por la que el tiempo es tan bajo en comparación con la técnica de Hiperbandas. Esta comparación se recoge gráficamente en la Figura 15.

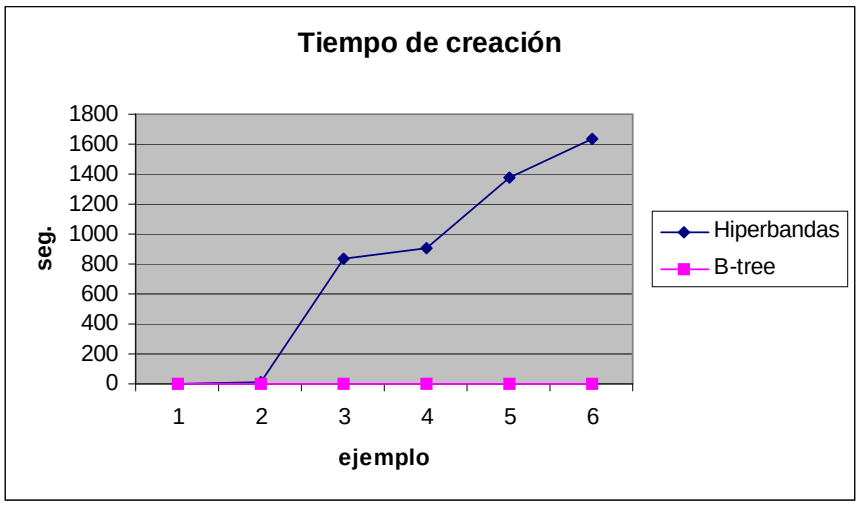


Figura 15.Comparación B-Tree – Hiperbandas respecto al tiempo de creación.

- En cuanto a la eficiencia de almacenamiento, aunque los datos numéricamente reflejan la estructura del B-Tree como mejor, se considera que son comparables, ya que la diferencia no es grande y siempre se podría recodificar la red para que fuera más compacta. Además, los algoritmos de optimización utilizados en la red Hiperbandas trabajan con precisión en coma flotante, mientras que en Índices Inversos trabajan con precisión simple. Esta comparación se recoge gráficamente en la Figura 16.

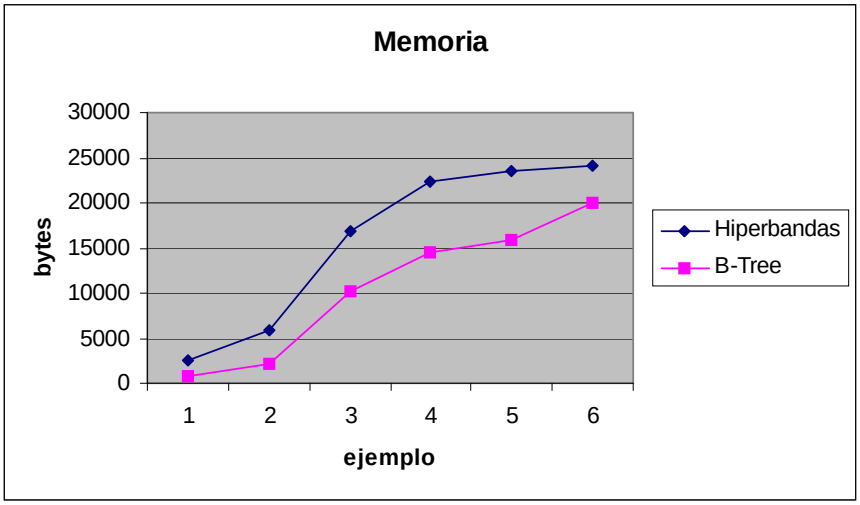


Figura 16.Comparación B-Tree – Hiperbandas respecto a la memoria utilizada.

- Por último, el tiempo de búsqueda en un árbol binario balanceado es proporcional al log(n), donde n, en este caso, es el número de términos del diccionario; mientras que, en la red Hiperbandas, se requiere la ejecución de las operaciones de la red, es decir, el tiempo es proporcional al número de procesadores y conexiones existentes. Por lo que esta última siempre estará en desventaja. No obstante, parece que el tiempo de respuesta para la red Hiperbandas puede ser aceptable. Esta comparación se recoge gráficamente en la Figura 17.

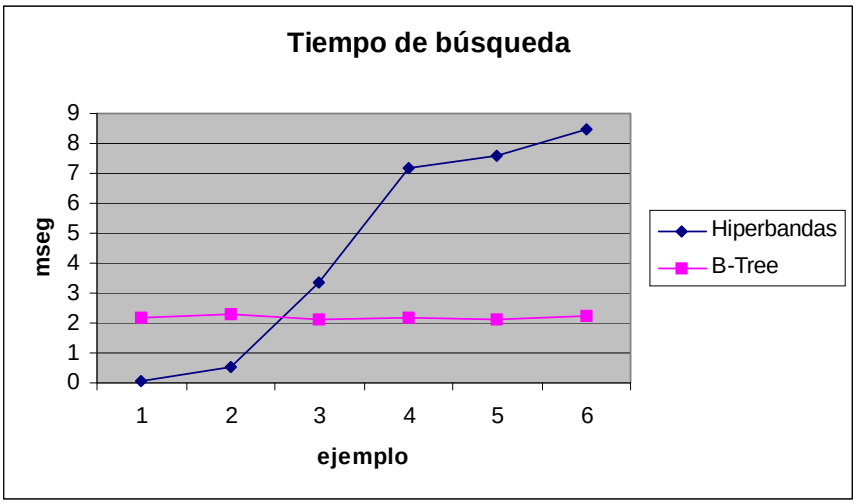


Figura 17.Comparación B-Tree – Hiperbandas respecto al tiempo de búsqueda.

#### *4.4.5.2. Incorporación y borrado de documentos.*

A continuación, con objeto de medir el comportamiento del B-Tree para incorporar nuevos documentos y borrar documentos existentes, se procedió a realizar las mismas pruebas que las llevadas a cabo con la red Hiperbandas. En la Tabla 8, se presenta los resultados correspondientes a Índices Inversos y su interpretación es la que se ha comentado para la Tabla 4.

Se prosiguió con las pruebas de borrado de documentos, eliminándose de uno en uno, para comprobar el tiempo que requiere esta operación. Del mismo modo que en el sistema implementado con la red Hiperbandas, se presenta el tiempo de borrado requerido por el B-tree para eliminar cada documento indicando el número de palabras del diccionario que contiene. Se presentan los resultados para las mismas colecciones y documentos de la Tabla 6. Los resultados obtenidos son los de la Tabla 9.

| Ejemplo | Tiempo de indexación (seg.) | Almacenamiento (bytes) | Tiempo de búsqueda (mseg.) |
|---|---|---|---|
| ejem_1 | 0,06 | 760 | 2,21 |
| ejem_1a | + 0.06 | 1060 | 2.19 |
| ejem_2 | 0,19 | 2160 | 2,30 |
| ejem_2a | + 0,185 | 3360 | 2,25 |
| ejem_3 | 0,411 | 10192 | 2,11 |
| ejem_3a | + 0,409 | 15808 | 2.23 |
| ejem_4 | 0,871 | 14548 | 2,19 |
| ejem_4a | + 0,877 | 24616 | 2,26 |
| ejem_5 | 1,082 | 15952 | 2,11 |
| ejem_5a | + 1,088 | 27424 | 2.24 |
| ejem_6 | 1,262 | 20008 | 2,25 |
| ejem_6a | + 1.267 | 35536 | 2,21 |

Tabla 8. Resultados al añadir documentos al B-Tree.

A la vista de los resultados mostrados en las Tablas 8 y 9 y comparados con los resultados presentados en las Tablas 4 y 6, se puede concluir que con la técnica de Índices Inversos:

- El tiempo requerido para incorporar más documentos al DRS, es proporcional al número de términos del diccionario que contengan los documentos a indexar. Como se trabaja con la misma colección de partida para añadirla al sistema, y el proceso es similar al de indexación, el tiempo necesario para la incorporación de éstos documentos es prácticamente igual al tiempo de la indexación inicial. Este tiempo es inferior al que requiere la red Hiperbandas.

- El incremento de espacio de almacenamiento, sólo se ve afectado por la referencia a nuevos documentos, cada referencia ocupa 12 bytes. En la red Hiperbandas se requieren 24 bytes debido al tipo de datos con los que se trabaja, que son de precisión en coma flotante.

- El tiempo de búsqueda no resulta muy afectado por el número de documentos dado que la estructura es independiente de la colección de documentos, salvo en lo que respecta a la referencia a éstos.

- El tiempo necesario para borrar un documento de la colección depende del número de términos en los que esté referenciado y del tamaño del árbol, pues es preciso recorrer el mismo para ver si en cada uno de sus nodos está referenciado el documento. Se aprecia que el tiempo de borrado de un documento depende muy poco del número de términos del diccionario que contenga, pues el tiempo empleado corresponde casi totalmente al que necesita para recorrer el árbol. Si se compara con la red Hiperbandas, el tiempo de esta última resulta ligeramente inferior cuando la red es pequeña, pero al crecer, se equipara e incluso, es superior.

Por otra parte, cuando se añaden documentos a una estructura creada previamente mediante la técnica de Índices Inversos, no hay ninguna diferencia por el hecho de que los nuevos documentos sean similares o no a los indexados previamente. Esto es debido a las características propias de la arquitectura implementada (Índices Inversos con diccionario preestablecido).

| *Ejemplo* | *Documento borrado* | *Tiempo de borrado (mseg.)* |
|---|---|---|
| Ejem_6 | doc.: 68 (6 palabras) | 0.103086 |
| | doc.: 115 ( 4 palabras) | 0.106438 |
| | doc.: 148 ( 6 palabras) | 0.100571 |
| | doc.: 295 ( 2 palabras) | 0.102248 |
| Ejem_6a | doc.: 68 (6 palabras) | 0.107276 |
| | doc.: 115 ( 4 palabras) | 0.109790 |
| | doc.: 148 ( 6 palabras) | 0.108952 |
| | doc.: 295 ( 2 palabras) | 0.108952 |
| | doc.: 296 ( 1 palabra ) | 0.104114 |
| | doc.: 460 ( 0 palabras ) | 0.109790 |
| | doc.: 532 ( 14 palabras ) | 0.107276 |

Tabla 9. Resultados al borrar documentos en el B-Tree.

### *4.5. Discusión*

De los resultados expuestos en los apartados 4.4.4 y 4.4.5 se puede concluir que se está cerca de encontrar una arquitectura de red neuronal de utilidad en el campo del Almacenamiento y Recuperación Documental con características equiparables a la técnica de Índices Inversos. Para ello, se debe mejorar, sobre todo, el tiempo de búsqueda ya que en tiempo de indexación la red siempre va a estar en clara desventaja, aunque podría estar más optimizado.

En relación a la efectividad de la búsqueda, ambas estructuras responden con total exactitud ya que en el presente trabajo no se han tenido en cuenta ningún juicio de relevancia. Además el tipo de red neuronal propuesto no proporciona características de generalización, pues solo es capaz de reconocer lo aprendido.

Ante la incorporación de nuevos documentos, la red Hiperbandas puede resultar beneficiada por la similitud de éstos con los que ya están en la colección, mientras que la técnica de Índices Inversos, por las características de su arquitectura, no se ve afectada por el parecido entre los documentos existentes y los que se incorporan.

En el caso del borrado, debido a la arquitectura de implementación de ambos sistemas, la red Hiperbandas está en una posición más ventajosa si el número de términos en el diccionario es alto y la colección de documentos pequeña, mientras que, en el caso contrario, el B-tree es el favorecido.

Respecto al espacio requerido para el almacenamiento de los índices, se considera que la red es suficientemente compacta pero, sus procesos de optimización requiere mayor precisión lo que conduce a un consumo de memoria algo mayor que el de Índices Inversos.

Se ha de tener en cuenta que en la implementación software de ambas estructuras se ha utilizado el esquema estándar de punteros y el tiempo se ha medido con una de las funciones que Visual C/C++ ofrece pero que dista de ser la opción más precisa. También, el compilador y la máquina donde se ejecuta (Pentium II con S.O. NT Workstation) influye en los resultados y, por ello, se deben considerar como promedios de un valor aleatorio con una cierta dispersión.

Además, siempre se puede encontrar una especificación hardware o software que se adapte mejor a la estructura de datos y potencie su funcionamiento, como puede ocurrir con la red Hiperbandas pues dadas sus características, la materialización hardware es sencilla ya que solo requiere de puertas AND y puertas OR para implementar la función de salida de la capa de subconjunto e hiperbandas respectivamente, lo que probablemente mejoraría sus tiempos de indexación y búsqueda.

# Conclusiones

# 5. Conclusiones.

En esta tesis se ha realizado un análisis exhaustivo de los llamados Sistemas de Almacenamiento y Recuperación Información (IRS), especialmente de los dedicados a la indexación y recuperación de documentos de texto por su contenido. Brevemente se ha descrito su funcionamiento general y se han resumido las técnicas más relevantes haciendo hincapié en sus características específicas. Para ello se han tenido en cuenta tanto las disponibles comercialmente como las que se utilizan en el ámbito académico y de investigación.

Por otra parte, con el objetivo de su utilización en sistemas IR, se ha experimentado en el campo de las redes neuronales, trabajando con distintas arquitecturas de red y métodos de ajuste conocidos. Para ello, se han utilizado distintos softwares de propósito general como SNNS, PDP++ y la toolbox Neural Network de Matlab, y también se han confeccionado programas para implementar las arquitecturas de perceptrón multicapa y red Hiperbandas orientadas al problema de la indexación documental.

Como objetivo de esta tesis se trata de utilizar redes neuronales con las que proponer alternativas a la técnica de Índices Inversos, que es la más difundida entre los sistemas IR. Esto quiere decir que la red que se proponga debe identificar cada documento y localizarlo a partir de las palabras que contiene. En este contexto, las técnicas utilizadas hasta ahora, se basan en la clusterización de documentos por medio de redes neuronales autoorganizativas, que no responden de la misma forma que la técnica de Índices Inversos.

Para la consecución del objetivo propuesto, se ha trabajado con redes neuronales con función de base radial, perceptrones multicapa con función de activación radial y tangente hiperbólica y con métodos constructivos como Cascade Correlation. Los resultados obtenidos con estas arquitecturas no han sido positivos, pues, o bien fracasaban en el ajuste, es decir, no aprendían la muestra, o bien, el número de parámetros necesarios (procesadores ocultos y pesos) era excesivamente alto, por lo que la memoria requerida era muy superior a su alternativa con Índices Inversos.

Por este motivo, se ha propuesto una nueva arquitectura, la de red Hiperbandas, cuyas características más relevantes se pueden resumir en que se trata de una red que se construye ex profeso para clasificar un conjunto de datos de entrada sin interacción del usuario, con error cero y adecuada para situaciones donde los patrones de entrada se agrupan formando clases que pueden estar solapadas.

Para verificar las prestaciones de esta nueva arquitectura, se han realizado comparaciones con el sistema usual basado en la técnica de Índices Inversos implementado mediante un B-Tree simple. Para llevar a cabo este análisis, se han examinado los parámetros que se utilizan generalmente para evaluar el rendimiento de un DRS, estos son: el tiempo de indexación de la colección y el comportamiento de la estructura ante la incorporación de nuevos documentos y ante el borrado de los existentes, el espacio de almacenamiento requerido, la eficiencia de búsqueda y la efectividad de la consulta.

De los resultados obtenidos en esta tesis puede concluirse que la alternativa propuesta se adapta bien a la indexación de colecciones temáticas, es decir, a trabajar con un diccionario preestablecido (keywords). Si bien es verdad que, no se puede decir, que esta red neuronal pueda suplantar a la técnica de Índices Inversos de la que existen multitud de variantes y muchos años de investigación, de hecho es la técnica comercialmente más implantada.

Por otra parte, se debe valorar que con esta tesis se presenta una alternativa basada en redes neuronales nunca vista anteriormente en el campo del Almacenamiento y Recuperación Documental, dado que la técnica más utilizada era la clusterización de documentos por medio de redes autoorganizativas.

También se debe resaltar que, por utilizar una codificación binaria, este sistema puede ser adaptado para indexar cualquier otro tipo de información, siempre que se tenga definido una conjunto de atributos por los que se los deba clasificar, como pueden ser fotos, películas, etc.

# Futuras líneas de investigación

# 6. Futuras líneas de investigación.

Cuando se intenta finalizar un trabajo de esta extensión siempre queda la sensación de que es mucho más lo que falta por hacer que lo que se ha hecho. En lo que sigue se mencionan algunas de las direcciones en las que se podría trabajar a partir de los resultados obtenidos en esta tesis.

En primer lugar, podrían diseñarse y desarrollar interfaces de usuario que proporcionen un utilización similar a las ofrecidas por los buscadores comerciales.

También se podría profundizar en la mejora del proceso de optimización de la red, con objeto de reducir el tiempo necesario para su creación. Simultáneamente, se podría progresar en la justificación de alguno de los parámetros utilizados en los algoritmos de ajuste. En este sentido, se podría trabajar con algoritmos orientados específicamente a datos binarios.

Por otra parte, podría intentarse mejorar la definición de la función de error de forma que permita una evaluación más acertada del mismo. Su objetivo sería el de minimizar el número de procesadores ocultos necesarios para aprender la muestra.

Dentro del área de Recuperación Documental propiamente dicha, se podrá avanzar en la integración de un módulo que permita la codificación semántica del diccionario por medio de una base de datos externa que recoja las relaciones semánticas existentes entre los términos. Esto exigiría modificar la estrategia de crecimiento de la red, ya que la capa de entrada debería ser de tamaño variable.

Finalmente, se podría probar mejor el comportamiento del sistema propuesto, trabajando con grandes colecciones de documentos sobre una arquitectura hardware más potente, utilizando incluso tecnología paralela y distribuida, con lo que se aprovecharía el potencial del paradigma empleado.

# Anexos

# Anexo I. Programas.

En el desarrollo de esta tesis, en lenguaje C/C++ haciendo uso de la herramienta Microsoft Visual C++ versión 5.0, se han realizado tres programas.

El primero, construye el diccionario predefinido a partir del fichero diccio.dat y codifica cada palabra en binario, realiza la fase de preprocesado de los documentos, genera la matriz de términos x documentos y, por último, implementa el sistema propuesto (Figura 12) con la arquitectura del Perceptrón Multicapa. Para la fase de optimización de la red se han codificado dos algoritmos de ajuste: el algoritmo gradiente conjugado y el algoritmo cuasi-Newton, ambos con la capacidad de utilizar minimización lineal o parabólica.

El segundo programa, recoge la construcción del Sistema Documental propuesto con la arquitectura de la red Hiperbandas según el algoritmo descrito en el apartado 3.6.1. El programa dispone de las siguientes opciones: crear la red para una colección de documentos, incorporar más documentos a la colección, borrar documentos existentes y consultar los mismos. Este programa hace uso de las matrices de datos generadas en el programa anterior.

Por último, en el tercer programa, se implementa el árbol binario balanceado, B-Tree y permite realizar la indexación, borrado y consulta de documentos.

### *Programa 1. Perceptrón multicapa*

Las principales funciones que incorpora el programa 1 son:

- Creación del diccionario binario a partir del fichero diccio.dat
- Generación de la matriz de entrenamiento con el conjunto de ficheros planos que se encuentren en el directorio col_doc.
- Creación de perceptrón multicapa.
- Ajuste de la red.

- Búsqueda documental (respuesta de la red )

A continuación se describe, a modo de pseudocódigo, la secuencia de operaciones que sigue el algoritmo del programa.

**Proceso "Creación del diccionario"**

Lectura del fichero diccio.dat (palabras ordenadas alfabéticamente)

Para cada palabra

Se elimina los acentos y se convierte a minúsculas

Se almacena junto con el valor binario de su posición en el fichero

**Proceso "Generación de matriz de entrenamiento"**

Para cada palabra en el diccionario

Abrir cada documento en el directorio col_doc

Si existe la palabra, se indicará con el valor 1

**Proceso "Creación de la red neuronal inicial"**

Crea tantos procesadores de entrada como longitud definida para la palabra del diccionario

Crea procesadores de la capa oculta

Crea procesadores de la capa de salida, uno por cada documento

Conecta procesadores de capa entrada a capa oculta

Conecta procesadores de capa oculta a salida

**Proceso "Ajuste inicial"**

Repite

Lee matriz de entrenamiento

Para cada fila de la matriz

Obtiene respuesta de la red

Calcula el error y las derivadas

Modifica la dirección del gradiente

hasta que llega el límite de iteraciones o el progreso sea pequeño

**Proceso "Búsqueda documental"**

Para cada palabra a buscar

Convierte palabra en binario

Respuesta de la red para esa palabra

Presentar al usuario los documentos que contienen la palabra

En lo que sigue, se presenta la estructura de datos con la que se construye el perceptrón multicapa.

```
enum conectividad {SPS, PPS, SSC}; //suma, producto,hibrida
enum tipo_capa {CONSTANTE, ENTRADA, OCULTA_1,
                        OCULTA_2,VACIA_1,VACIA_2, SALIDA};
struct enlace {
  double valor;         //valor suma o producto de los procesadores en enlace
  double peso;
  double peso0;                 //peso previo
  double derivada;
  double derivada0;             //derivada previo
  double direccion;             // algoritmo cuasi-newton
  enum conectividad conexion;  //  (0 aditiva, 1 multiplicativa
```

```
  int orden;                        /* numero de entradas de cada enlace.  */
  enum tipo_capa *entradas_capa;
  int *entradas_elemento;           /* elementos que entran a cada enlace*/
  };

struct elemento {
  double valor;                     //resultado despues de funcion de transferencia
  double sigma;                     //resultado antes de funcion (sumatorio)
  double c1;                        // derivada del error respecto a la suma de
                                  // conexiones de entrada de/dy*f'act
struct enlace* entrada;
  int numero_entradas;
  };

struct capa {
  struct elemento  *neurona;
  int numero_elementos;
  };

typedef struct capa *red[7];
typedef struct capa **redvar;

struct muestra {
  int numero_docs;
  int long_palabra;
  int tamano_dic;
  int* matriz;
  };  //matriz es tamano_dic * numero_docs (cada fila docs que tienen la palabra)
```

### *Programa 2. Red Hiperbandas*

Las principales funciones que incorpora el programa 2 son:

- Construcción de la red por ajuste a la matriz de términos x documentos presentada.
- Ampliación de la red creada previamente por ajuste a una nueva matriz de términos x documentos facilitada al programa.
- Borrado de documentos existentes.
- Búsqueda documental (respuesta de la red).

A continuación se describe, a modo de pseudocódigo, la secuencia de operaciones que sigue el algoritmo del programa 2.

**Proceso "Creación de la red"**

Lectura de la matriz de términos x documentos

Para cada documento

Se crea procesador de salida

Se ajusta la red hasta obtener error cero según algoritmo (apartado 3.6.1)

**Proceso "Ampliación de la red"**

Lectura de la red inicial

Lectura de la matriz de términos x documentos

Para cada nuevo documento

Se crea procesador de salida

Se ajusta la red hasta obtener error cero según algoritmo (apartado 3.6.1)

**Proceso "Borrado de documentos"**

Lectura del documento a borrar

Buscar procesador que lo representa

Borrar enlaces del procesador con procesadores de la capa de subconjuntos

**Proceso "Uso de la red"**

Lectura de las palabras a buscar en formato binario

Para cada palabra

Ejecutar la red

Escribir resultado

En lo que sigue, se presenta la estructura de datos con la que se construye la red Hiperbandas.

```
struct enlace {                 /* Conexión */
  double valor;                 /* Valor del procesador conectado */
  double peso;
  int entradas_capa;
  int entradas_elemento;        /* Elementos que entran a cada enlace */
};

struct elemento {               /* Procesador */
  double valor;                 /* Salida */
  double sigma;                 /* Resultado de operar todas las conexiones */
  double c1;                    /* auxiliar */
  int num_doc;          /* Id documento de la colección */
  struct enlace *entrada;       /* tabla de conexiones */
  int numero_entradas;
};
```

```
  struct capa {                         /* Capa */
    struct elemento *neurona;           /* tabla de procesadores */
    int numero_elementos;
    struct elemento **activos;          /*tabla de punteros a los que es necesario operar
    int nactivos;
  };

  struct defred {           /* Red */
    int ncapas;
    int *nproc;                     /* tabla con número de procesadores en cada capa */
  };

  struct rna {
    struct capa *grupo;
    int ncapas;
  };
  typedef struct rna *red;

  struct muestra {                  /* Muestra de datos */
    float *registro;                /* Los valores numéricos de las variables */
    long total_puntos;
};
```

### *Programa 3. Árbol binario balanceado.*

Las principales funciones que incorpora el programa 3 son:

- Crear el árbol binario con las palabras que componen el diccionario predefinido.
- Indexar la colección de documentos indicada.
- Buscar documentos por las palabras del diccionario.
- Borrar documentos de la colección.

A continuación se describe, a modo de pseudocódigo, la secuencia de operaciones que sigue el algoritmo del programa 3.

**Proceso "Crear árbol binario"**

Lectura del fichero que contiene el diccionario

Para cada palabra

Se crea un nodo del árbol

Se efectúa balanceo, si lo requiere

**Proceso "Indexar colección de documentos"**

Lectura de los documentos a indexar

Para cada documento

Se lee cada palabra

Se busca en el árbol

Si se encuentra, se hace referencia al documento en este nodo

**Proceso "Buscar documentos"**

Buscar palabra en el árbol

Presentar los documentos que la contienen

**Proceso "Borrar documentos"**

Lectura del documento a borrar

Buscar en cada nodo del árbol si se encuentra el documento

Si es así borrar referencia

En lo que sigue, se presenta la estructura de datos con la que se construye el árbol binario balanceado.

```
class tree {
   private:
  // Clases nodo/palabra que componen el árbol binario principal
  class node {
     private:
      node    *right;      // rama del árbol a la derecha
      node    *left;       // rama del árbol a la izquierda
     public:
      char    *word;      // palabra de cada nodo
      int times; // número de veces que aparece la palabra
      double    clave;    // clave numérica de cada palabra
      int level;              // Información sobre el balanceado
      int count;  // lleva la cuenta de los ficheros que contienen la palabra
      struct file_name fichero[MAX_FILES];
      node(void) { times = 1; clave = 0; count = 0; level = 1;}
     friend class tree;       // declaramos al árbol clase amiga para poder acceder a node
  };

  node *root1;
  node *Nullnode;
```

```
node *RotateWithLeftChild(node *K2)
{
    node *K1 = K2->left;
    K2->left = K1->right;
    K1->right = K2;
    return K1;
};

node *RotateWithRightChild(node *K1)
{
    node *K2 = K1->right;
    K1->right = K2->left;
    K2->left = K1;
    return K2;
};

// Buscamos palabra en el fichero
void filter_one(node *nodo_dato, char *word, double nummer, char *name);
// Metemos nuevo nodo en un árbol
void enter_two(node *&node, char *word, double nummer, char *name);
// Borra árbol
void borra_arbol_one(node * B);
// Sacamos por pantalla un simple nodo
void print_one(node *top);
// Calculo la altura del árbol (un subárbol)
int height_one(node *tip);
// Buscamos palabra en el fichero
void search_two(node *nodo_dato, char *word, double nummer);
// Balanceo
void skew (node *&T);
void split (node *&T);
// Función para el volcado de datos...
void PrintPreOrder(node *node_order,char *padre);
void plain_file_one(void);
// Borrar documento
void delete_one(node *node_order, char *padre, int num_id);
```

```
public:
  tree(void) {   Nullnode = new node;
            Nullnode->left = Nullnode->right = Nullnode;
            Nullnode->level = 0;
            root1 = Nullnode;}      //constructor
  // Añadimos nueva palabra al árbol de palabras vacias
        void enter2(char *word, double nummer, char *name) {
       enter_two(root1, word, nummer, name);
  }
  // Deshago el árbol para 'liberar memoria'
  void borra_arbol() {
       borra_arbol_one(root1);
  }
  // Sacamos por pantalla el árbol
  void print1(void) {
      print_one(root1);
  }
  //Función PrintPreOrder
  void volcado(void) {
       PrintPreOrder(root1,"***");
  }
  //Función borrar doc
  void borrar_doc(int num) {
       delete_one(root1,"***",num);
  }
  // Buscamos palabra en el árbol
  void filter(char *word, double nummer, char *name){
       filter_one(root1, word, nummer, name);
  }
  // Calculo la altura del árbol de palabras filtradas
  int height(void) {
       return height_one(root1);
  }
  // Buscamos palabra en el árbol de palabras filtradas
  void search2(char *word, double nummer){
       search_two(root1, word, nummer);
  }
```

```
    //reconstruccion del arbol de fichero a memoria
    void plain_file(void){
        plain_file_one();
    }
    double HacerClave(char *word, int truncado)
    {
        double n = 0;
        int indice;
        for (int i = 0; i<truncado && word[i] !='\0'; i++)//Ojo con truncado
        {
            indice = 0;
            if (( word[i] == 'á' ))       indice = 65;              //á       -96
            else if ( word[i] == 'é' ) indice = 69;      //é       -126
            else if ( word[i] == 'í') indice = 73;                 //í       -95
            else if ( word[i] == 'ó' ) indice = 80;      //ó       -94
            else if ( word[i] == 'ú' ) indice = 86;      //ú       -93
            else if ( word[i] == 'ñ' || word[i] == 'Ñ')  //ñ y Ñ  -92 -91
                indice = 79;
            else if ( (toupper(word[i])) >= 79 )
                indice = toupper(word[i])+1;          //al meter la ñ entre la 'n' y la 'o'
                                                      //desplazo las letras: o p q ... z en una
                                                      //unidad por lo de la ñ, pasa a 27
            else
                indice = toupper(word[i]);
            n += (  indice - 64  ) * pow(27,(truncado-1-i)) ;// OJO con truncado
            }
            return (n);
        }
struct file_name {          //Construyo una estructura que declaro dentro de la clase nodo
  char*nombre; //donde guardaré la info. sobre del nombre del fichero donde
  int   subtimes;           //aparece un término, así como el número de veces que lo hace
  int   num_id;             //localiza un fichero dentro de pal_en_doc
  };
```

# Anexo II. Diccionario.

En este anexo se presenta el diccionario de términos utilizado en la realización de esta tesis.

abandonar
abierto
abogar
aborda
abstención
acciones
acogimiento
administración
administrativo
adopción
adquirir
adquisición
agilizar
agresivo
agresor
anuncio
asilo
asociaciones
ausente
beneficios
bienes
byte
cabal
cambiar
capacidad
cardenal
cigarrillo
civil
concebido
concesión
conducta
conflicto
consejo
constatar
constitución
contencioso
contraer
corporaciones
deber
débito
decreto
defensa
denegación
denuncia
derecho
detalle
dieciocho
dinero
diplomático
discapacidad
domicilio
edad
efectos
emancipación
emancipado
emigrante
empleo

españa
estar
estatutos
europa
europeo
expediente
expirado
extranjeros
falsedad
familia
fértil
fraude
fundaciones
grado
guarda
guerra
hijo
horas
idioma
iglesia
incapacitado
industriales
instituciones
instrucciones
interesado
juicio
jurar
jurídicas
justicia
lengua
leyes
madre
materno
matrimonio
mayor
menor
mercantiles
ministro
muerte
municipio
nacimiento
nacionalidad
naturaleza
nazca
obligaciones
padre
particular
paternidad
patria
pena
perdida
personalidad
personas
plazo
poseer
potestad
presumen
presunción
primogénito
privado
provincia
recoger
refugio
región
regirán
registro
relaciones
representante
residencia
residente
rey
seno

| | | |
|---|---|---|
| sentencia | teoría | tutela |
| servicio | territorio | varón |
| sociedad | transmisión | viudo |
| suspensión | tratados | |

# Anexo III. Documentos.

En este anexo se presentan, a modo de ejemplo, los documentos que componen la colección más reducida, la denominada ejemplo 1.

**Artículo 100**

Fijada la pensión y las bases de su actualización en la sentencia de separación o de divorcio, sólo podrá ser modificada por alteraciones sustanciales en la fortuna de uno u otro cónyuge.

**Artículo 101**

El derecho a la pensión se extingue por el cese de la causa que lo motivó, por contraer el acreedor nuevo matrimonio o por vivir maritalmente con otra persona.

El derecho a la pensión no se extingue por el solo hecho de la muerte del deudor. No obstante, los herederos de éste podrán solicitar del Juez la reducción o supresión de aquélla, si el caudal hereditario no pudiera satisfacer las necesidades de la deuda o afectara a sus derechos en la legítima.

**Artículo 102**

Admitida la demanda de nulidad, separación o divorcio, se producen, por ministerio de la Ley, los efectos siguientes:

1. Los cónyuges podrán vivir separados y cesa la presunción de convivencia conyugal.

2. Quedan revocados los consentimientos y poderes que cualquiera de los cónyuges hubiera otorgado al otro.

Asimismo, salvo pacto en contrario, cesa la posibilidad de vincular los bienes privativos del otro cónyuge en el ejercicio de la potestad doméstica.

A estos efectos, cualquiera de las partes podrá instar la oportuna anotación en el Registro Civil y, en su caso, en los de la Propiedad y Mercantil.

**Artículo 103**

Admitida la demanda, el Juez, a falta de acuerdo de ambos cónyuges aprobado judicialmente, adoptará, con audiencia de éstos, las medidas siguientes:

1. Determinar, en interés de los hijos, con cuál de los cónyuges han de quedar los sujetos a la patria potestad de ambos y tomar las disposiciones apropiadas de acuerdo con lo establecido en este Código y en particular la forma en que el cónyuge apartado de los hijos podrá cumplir el deber de velar por éstos y el tiempo, modo y lugar en que podrá comunicar con ellos y tenerlos en su compañía.

Excepcionalmente, los hijos podrán ser encomendados a otra persona y, de no haberla, a una institución idónea, confiriéndoseles las funciones tutelares que ejercerán bajo la autoridad del Juez.

2. Determinar, teniendo en cuenta el interés familiar más necesitado de protección, cuál de los cónyuges ha de continuar en el uso de la vivienda familiar y asimismo, previo inventario, los bienes y objetos del ajuar que continúan en ésta y los que se ha de llevar el otro cónyuge, así como también las medidas cautelares convenientes para conservar el derecho de cada uno.

3. Fijar la contribución de cada cónyuge a las cargas del matrimonio, incluidas si procede las "litis expensas", establecer las bases para la actualización de cantidades y disponer las garantías, depósitos, retenciones u otras medidas cautelares convenientes, a fin de asegurar la efectividad de lo que por estos conceptos un cónyuge haya de abonar al otro.

Se considerará contribución a dichas cargas el trabajo que uno de los cónyuges dedicará a la atención de los hijos comunes sujetos a patria potestad.

4. Señalar, atendidas las circunstancias, los bienes gananciales o comunes que, previo inventario, se hayan de entregar a uno u otro cónyuge y las reglas que deban observar en la administración y disposición, así como en la obligatoria rendición de

cuentas sobre los bienes comunes o parte de ellos que reciban y los que adquieran en lo sucesivo.

5. Determinar, en su caso, el régimen de administración y disposición de aquellos bienes privativos que por capitulaciones o escritura pública estuvieran especialmente afectados a las cargas del matrimonio.

**Artículo 104**

El cónyuge que se proponga demandar la nulidad, separación o divorcio de su matrimonio puede solicitar los efectos y medidas a que se refieren los dos artículos anteriores.

Estos efectos y medidas sólo subsistirán si, dentro de los treinta días siguientes a contar desde que fueron inicialmente adoptados, se presenta la demanda ante el Juez o Tribunal competente.

**Artículo 105**

No incumple el deber de convivencia el cónyuge que sale del domicilio conyugal por una causa razonable y en el plazo de treinta días presenta la demanda o solicitud a que se refieren los artículos anteriores.

**Artículo 106**

Los efectos y medidas previstos en este capítulo terminan, en todo caso, cuando sean sustituidos por los de la sentencia estimatoria o se ponga fin al procedimiento de otro modo. La revocación de consentimientos y poderes se entiende definitiva.

**Artículo 107**

La separación y el divorcio se regirán por:

La ley nacional común de los cónyuges en el momento de la presentación de la demanda; a falta de nacionalidad común, por la ley de la residencia habitual del

matrimonio y, si los esposos tuvieran su residencia habitual en diferentes Estados, por la ley española, siempre que los tribunales españoles resulten competentes.

Las sentencias de separación y divorcio dictadas por Tribunales extranjeros producirán efectos en el ordenamiento español desde la fecha de su reconocimiento conforme a lo dispuesto en la Ley de Enjuiciamiento Civil.

**Artículo 108**

La filiación puede tener lugar por naturaleza y por adopción. La filiación por naturaleza puede ser matrimonial y no matrimonial. Es matrimonial cuando el padre y la madre están casados entre sí.

La filiación matrimonial y la no matrimonial, así como la adoptiva surten los mismos efectos, conforme a las disposiciones de este Código.

**Artículo 109**

La filiación determina los apellidos con arreglo a lo dispuesto en la ley.

Si la filiación está determinada por ambas líneas, el padre y la madre de común acuerdo podrán decidir el orden de transmisión de su respectivo primer apellido, antes de la inscripción registral. Si no se ejercita esta opción, regirá lo dispuesto en la ley.

El orden de apellidos inscrito para el mayor de los hijos regirá en las inscripciones de nacimiento posteriores de sus hermanos del mismo vínculo.

El hijo, al alcanzar la mayor edad, podrá solicitar que se altere el orden de los apellidos.

# Bibliografía

## Bibliografía

[1] Acevedo, F.; Zurdo, D.; Sicilia, A. *Buscadores de Internet*. Paraninfo. 1998.

[2] Ahrns,I.; Bruske, J.; Sommer, G. *On-Line Learning with Dynamic Cell Structures*. ICANN'95, Vol. 2 (1995) 141-146.

[3] Aluffi-Pentini, F.; Parisi, V. F.; Zirilli. *"Algorithm sigma, a stochastic integration global minimization algorithm*." A.C.M. Transactions On Mathematical Software, vol. 14, no. 4, pp. 366-388

[4] Austin, J.*Neural Associative Processing of Document Images*. Dpto. of Computer Science, University of York, UK.

[5] Blanzieri, E; Katenkamp, P. *Learning Radial Basis Function Networks On-line*. Machine Learning Proc 13TH Internation Conference (ICML96) , pp. 37-45.

[6] Berry, M.W; Dumais, S.T.; O'Brien, G.W. 1994. *Using Linear Algebra for Intelligent Information Retrieval*. Computer Science Department. University of Tennessee.

[7] Berlich, R.; Kunze, M. *A Comparation between the Performance of Feed Fordward Neural Networks and the Supervised Growing Neural Gas Algorithm.* Institut für Experimentalphysik, Germany.

[8] Blackmore, J.; Miikkulainen, R. *"Incremental Grid Growing: Encoding High Dimensional Structure into a Two-Dimensional Feature Map*", Proc. of the 1993 IEEE International Conference on Neural Networks, San Francisco, CA, pp.450-455.1993

[9] Brown, E.; Callan, J.; Croft, W.. *Fast Incremental Indexing for Full-Text Information Retrieval*. Proceedings of the 20th VLDB Conference. Santiago de Chile, 1994.

[10] Brown, E.; Callan, J.; Croft, W.; Moss, E. *Supporting Full-Text Information Retrieval with a Persistent Object Store*. University of Massachusetts.

[11] Bruske, J.; Sommer, G. *Dinamic Cell Structures.* Department of Cognitive Systems. Published in NISP 7, Proc. of the 1994 Conference (1995) 497-504.

[12] Can, F.; Drochak, F.. *Incremental Clustering for Dynamic Document Databases.* TH0307-9/90/0000/0061$01.00 IEEE. 1990.

[13] Chen, S.; Cowan, F.N.; Grant, P.M.. *Orthogonal Least Squares Learning Algorithm for Radial Basis Function Network*. IEEE Transactions on Neural Networks, Vol. 2, nº 2, March, 1991.

[14] Cocklin, J. *Hipertext: an introduction and survey*. Computer, September 1987, pag.17.

[15] *Computacion Neuronal*. Cursos de Verano. Universidad de Santiago de Compostela. 1995.

[16] Cottrell, G.W.; Munro, P.; Zipser, D. 1987. *Learning Internal Representations from Gray-Sacale Imagenes: an Example of Extensional Programming*. Proc. of th 5th Annual Conference of the Cognitive Science Society. Seattle. pp 462-473.

[17] Crespo, J.L. *Redes Neuronales Artificiales para Ingenieros.* 1996 ISBN: 84-605-5020-6.

[18] Crespo, J.L.1992. *Procesamiento Paralelo y Distribuido aplicado a la Simulación de Sistemas.* Tesis Doctoral. Universidad de Cantabria.

[19] Crespo, J.L. *Hyperband networks: description and automatic de*sign. Dpto. Matemática Aplicada y Ciencias de la Computación. Universidad de Cantabria. Preprint submitted to Elsevier Science. June, 2001.

[20] Croft, W.B. 1980. *A model of cluster searching based on classification*. Information Systems, Vol. 5, pp. 189-195.

[21] Cutting, D.R.; Karger, D. R.; Pedersen, J. O.; Tukey, J.W. June, 1992. *Scatter/Gather: a Cluster-based Apporach to Browsing Large Document Collections.* ACM SIGIR Conference, pp 318-329.

[22] Dawson,C.K.; O'Reilly, R.C.; McClelland, J.L. (2000). *PDP++ Software, version 2.0.*

[23] De Heer, T. 1982. *The Application of the Concept of Homeosemy to Natural Language Information Retrieval*. Annual Review of Information Processing and Management. Vol 18(5), pp. 229-236.

[24] Deerwester, S; Dumais, S.T.; Harshman, R.. 1990. *Indexing by Latent Semantic Analisys*. Journal of the American Society for Information Science, 41, pp. 391-407.

[25] Demuth, H.; Beale, M. Neural Network Toolboox for Use with MATLAB. User's Guide. The Math Works Inc.

[26] Dumais, S.T; Landauer, T.K.; Littman, M. L. 1996. *Automatic Cross-Linguistic Information Retrieval using Latent Semantic Indexing.*

[27] Faloutsos, C.; Oard, D. A *Survey Information Retrieval and Filtering Methods.* University of Maryland.

[28] Frakes, W.; Baeza-Yates, R. 1992. *Information Retrieval. Data structures & algorithms*. Prentice-Hall.

[29] Freeman, J. A.; Skapura, D. M. Neural Networks. Algorithms, Applications, And Programming Techniques.1991

[30] Fritzke,B. 1991. *Unsupervised Clustering With Growing Cell Structures*. Seatle, IJCNN'91.

[31] Fritzke,B. 1993. *Kohonen Feature Maps and Growing Cell Structures- a Performance Comparation*. Advances in Neural Information Processing Systems 5, Morgan Kaufmann, California.

[32] Fritzke,B. 1994. *Fast learning with incremental RBF Networks*. Neural Processing Letters, vol. 1, nº 1, pp. 2-5 5.

[33] Fritzke,B. 1995. *A Growing Neural Gas Network Learns Topologies*. Advances in Neural Information Processing Systems 7, MIT Press, Cambridge.

[34] Gerho, M.; Reiter, R.. *Information Retrieval Using Hybrid Multilayer Neural Networks*. Bellcore, NJ 08854.

[35] Hilera, José R.; Martínez, V. J. *Redes Neuronales Artificiales. Fundamentos, Modelos Y Aplicaciones*. Ra-Ma.1995

[36] Harman, D. *Overview of the Thrid Text Retrieval Conference* (TREC-3). 1995. NIST Special Publication 500-225.

[37] Hodge, V.J.; Austin, J. *An Evaluation of Standard Retrieval Algorithms and a Weighless Neural Approach*. Dpto. of Computer Science, University of York, UK.

[38] Honkela, T.; Pulkki,V.; K.; Kohonen, T. 1995. *Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map*. Proceedings of International Conference on Artificial Neural Networks, ICANN 95.

[39] Honkela, T.; Kaski, S.; Lagus, K.; Kohonen, T. 1996. *Newsgroup Explotarion with WEBSOM Method and Browsing Interface*. Helsinki Universitu of Techonology. Finland.

[40] Hui, S.C.; Goh, A. *Incorporating Fuzzy Logic with Neural Networks for Document Retrieval.* Pergamon PII:S0952-1976(96)00051-6.

[41] Ichikawa, Y.; Sawa, T. (1992). *"Neural network application for direct feedback controllers"*. IEEE Transactions on Neural. Networks, v. 3, n. 2, March.

[42] Keen, E. 1971. *Evaluation parameters in the SMART retrieval system in Automatic Document Processing.* Prentice-Hall. New Jersey.

[43] Ketelaere, B.; Moshou, D.; Coucke, P.; Baerdemaeker, J. 1996. *A hierarchical Self-Organizing Map for classification problems*. Dpto. of Agro-Engineering and Economics, Laboratory of Mechanical Engineering. Heverlee, Belgium.

[44] Köhle, M.; Merkl, D. 1996. *Visualizing Similarities in High Dimensional Input Spaces with a Growing and Splitting Neural Network*. Artificial Neural Networks - ICANN 96. pp 581-586.

[45] Kohonen, T.; Hynninen, J.; Kangas, J.; Laaksonen, J. 1996. *SOM_PAK: the Self-Organizing Map Program Package*. Laboratory of Computer and Information Science, Helsinki University of Technology.

[46] Kohonen, T. *Exploration of Very Large Databases by Self-Organizing Maps*. In Proccedings of ICNN'97, pp. PL1-PL6.

[47] Kohonen, T. *Self-Organization of Very Large Document Collections: State of Art*. In Proccedings of ICNN'98, pp. 65-74.

[48] Kohonen, T.; Kaski, S.; Lagus, K.; Honkela, V; Saarela, A.. *Self Organization of a Massive Document Collection*. IEEE Transactions on Neural Networks, Vol. 11, nº 1, May 2000.

[49] Kowalski, G. *Information Retrieval Systems: Theory and Implementation.* Kluwer Academic Publishers. ISBN: 0-7923-9926-9.

[50] Layaida, R.; Boughanem, M.; Caron, A. 1974. *Construction an Information Retrieval Systems with Neural Networks.* Database and Expert Systems Aplications 5th Internacional Conference.

[51] Lagus, K. *Generalizability of the WEBSOM Method to Document Collections of Various Types*. European Congress Intelligent Techniques and Soft. Computing (EUFIT'98), Vol. 1, pp. 210-214.

[52] Leloup, C. 1998. *Motores de Búsqueda e Indexación. Entornos cliente/servidor, Internet e Intranet*. Gestión 2000.

[53] Levis, D.D.; Schapire, R.E.; Callan, J.P.; Papka, R. *Training Algorithms for Linear Text Classifiers. Center for Intelligent Information Retrieval.* University or Massachusetts.

[54] Ludermir,T. B.; Carvalho, A.; Braga, P.A.; Souto, M.C.P. *Weightless Neural Models: A Review of Current and Past Works*. Neural Computing Surveys 2, 41-61. 1999. ISSN: 1093-7609.

[55] MacLeod, K.; Robertson, W.. *A Neural Algorithm for Document Clustering.* Information Processing and Management, Vol. 27(4), pp. 337-346.

[56] Martinetz, T.; Schulten, K. *Topology representing networks*. Neural Networks 7(3), 505-522. 1994.

[57] Masters, T. *Practical neural network recipes in C++*. Boston Academic Press, 1993. ISBN 0-124-79040-2

[58] Maybury, M. *Intelligent Multimedia Information Retrieval*. Aaai Press / The Mit Press. 1997.

[59] Movellan, J.R. *Effects of the Error Function on the Noise Resistance and Generalization Performance of Backpropagation*. Department of Psychology, Carnegie Mellon University. 1995.

[60] Munera, C.; Tena, J. *Codificación de la Información*. Universidad de Valladolid. 1997.

[61] Orzech, D. *Motores de Búsqueda Inteligente.* Datamation. Sept. 1998.

[62] Orr, M. J. L.. *Introduction to Radial Basis Function Networks.* Centre for Cognitive Science. University of Edimburgh. April, 1996.

[63] Orr, M. J. L.. *Matlab routines for subset selection and ridge regression in linear neural networks.* Technical Report, Institute for Adaptive and Neural Computation. University of Edimburgh. 1996.

[64] Orr, M. J. L.. *Recent Advances in Radial Basis Function Networks.* Institute for Adaptive and Neural Computation. University of Edimburgh. June, 1999.

[65] Parekh, R.; Yang, J.; Honavar, V. *Constructive Neural Network Learning Algorithms for Multi-Category Pattern Classification.* Department of Computer Science. Iowa State University. October, 1995.

[66] Ridder, D.; Tax, D.M.; Duin, R. P.. *An experimental comparison of one-class classification methods*. Dpto of Applied Physics. Delft University of Technology. The Netherlands.

[67] Riloff, E.; Lehnert, W. *Information Extraction as a Basis for High-Precision Text Classification.* ACM Transactions on Information Retrieval. July 1994, Vol.12, nº 3, pp 296-333.

[68] Ritter, H.; Kohonen,T. 1989. *Self-Organizing Semantical Maps*. Biological Cibernetics, Vol. 61, pp. 241-254.

[69] Rozmus, J. M. 1995. *Information Retrieval by self-organizang maps*. 16th National Online Meeting Proceedings. pp. 349-354.

[70] Salton, G.; McGill, M. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

[71] Salton, G.; Yang, C.S.; Wong, A.. 1974. *A Vector Space Model for Automatic Indexing*. Departament of Computer Science, Cornell University. TR 14-218.

[72] Sayood, K.; Kaufmann, M. *Introduction to Data Compression*. Publishers. 1996

[73] Schäuble, P. *Multimedia Information Retrieval. Content-Based Information Retrieval from Large Text and Audio Databases*. Kluwer Academic Publishers.1997

[74] Scholtes, J.C. 1991. *Unsupervised Learning and the Information Retrieval Problem*. IJCNN 91, International Joint Conference Neural Networks. vol. 1 , pp 95-100.

[75] Scholtes, J.C. 1993. *Neural Networks in Natural Language Processing and Information Retrieval*. Ph.D. Thesis, Universiteit van Amsterdam.

[76] Siolas, G.; d'Alchè-Buc, F. (2000). *"Support Vector Machines based on a Semantic Kernel for Text Categorization"*. Laboratoire d'Informatique de Paris, Université Pierre et Marie Curie. 0-7695-0619-4/00 IEEE.

[77] Smith, S.D.; Escobedo, R.; Anderson, M.; Caudell, P. *A Deployed Engineering Design Retrieval System Using Neural Networks*. IEEE Transactions on Neural Networks, Vol. 8, nº 4, July 1997.

[78] Syu,I; Lang, S.D. 1994. *A Competition Based Connectionist Model for Information Retrieval Using a Merged Thesaurus*. CIKM 94 proc. 3rd International Conference of Information and Knowlegde. Ref: TXHH012231.

[79] Syu,I; Lang, S.D.; Deo, N. 1996. *A Neural Network Model for Information Retrieval Using Latent Semantic Indexing*. IEEE 96 International Conference Neural Networks.

[80] Torres Moreno, J. M.; Gordon, M. B. *Efficient Adaptive Learning for Classification Tasks with Binary Units*. Départment de Recherche Fondamentale sur la Matière Condensée. CEA Grenoble. June, 1997.

[81] Von Altrock, C. *Fuzzy Logic & Neurofuzzy Applications Explain*. Prentice-Hall.

[82] Woods, W. A.. *Conceptual Indexing: A Better Way to Organize Knowledge*. Technical Report published by Sun Microsystems Laboratories. 1997.

[83] Zavrel, J. 1995. *An Experimental Study of Clustering and Browsing of Document Collections with Neural Networks*. Ph.D. Thesis, Universiteit van Amsterdam.

[84] Zell, A. et al. (1998). *SNNS, version 4.2*. University of Stuttgart.

[85] Zorrilla, M.; Crespo, J.L.; Mora, E.. *An Online Information Retrieval Systems by means of Artificial Neural Networks*. Lecture Notes in Computer Science, vol. 2178.