Universidad de Alcalá

Escuela Técnica Superior de Ingeniería Informática

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

NUEVA PROPUESTA EVOLUTIVA PARA EL AGRUPAMIENTO DE DOCUMENTOS EN SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

José Luis Castillo Sequera

2010

Universidad de Alcalá

Escuela Técnica Superior de Ingeniería Informática

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS DOCTORAL

NUEVA PROPUESTA EVOLUTIVA PARA EL AGRUPAMIENTO DE DOCUMENTOS EN SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

Autor:

José Luis Castillo Sequera

Ingeniero en Informática

2010

Directores:

Dr. León Atilano González Sotos
Dr. José Raúl Fernández del Castillo Diez

Esta tesis se la dedico de todo corazón a mi Madre Elena, y en especial a mi Padre Pedro Castillo, que descanse en paz, porque sin ellos no sería nada.

Agradecimientos

Tal vez sea ésta la única página de la presente tesis que escribo con seguridad y, desde luego, con absoluta satisfacción, ahora que por fin se adivina el final del túnel, llega el momento más emotivo, al menos para mí.

En primer lugar quiero agradecer muy especialmente a mis directores, León González Sotos y José Raúl Fernández del Castillo Diéz, por su apoyo y su confianza para que este trabajo haya llegado a ser una realidad, poco de este trabajo se habría realizado si no fuera por su dedicación y disposición.

Mi especial agradecimiento va para León González Sotos, mi Director de Tesis, cuya capacidad profesional y calidad humana me motivarán siempre. El me enseñó que es posible aprender algo más si se dispone de ilusión, también me dio algo que todos necesitamos, especialmente al empezar, respeto.

Tengo que agradecer el poder haber realizado esta tesis a varias personas, y tengo que empezar por supuesto por mi familia, mi madre y mis hermanas, los cuales me han animado a seguir adelante con su sola presencia a mi lado, perdonándome el tiempo que no les he podido dedicar y que me gustaría haberles dedicado, les doy las gracias por haberme querido y apoyado siempre.

Quiero agradecer a mi Madre toda la fuerza que me ha transmitido y el haber sabido perdonar el poco tiempo que les he podido dedicar en estos meses. Sin ella, sin su comprensión y todo el amor que me han dado, esto no habría sido posible.

Sólo mi hermana sabe lo enormemente agradecido que le estoy por simplemente estar a mi lado y haberme salvado de un abismo que no hace mucho se abría a mis pies. Gracias Charo, por haberme demostrado que la victoria no sabe igual si no se lucha.

Debo también mencionar que el trabajo que ha desembocado en esta tesis doctoral no hubiera sido posible sin el apoyo que he encontrado en el Departamento de Ciencias de la Computación de la Universidad de Alcalá desde la dirección al personal de administración y servicios, pasando por mis compañeros en las tareas docentes, muchas gracias a todos por su apoyo y confianza.

Por último y para que nadie se quede fuera de mis reconocimientos, agradecer a todas las personas que, desde el anonimato y también de forma desinteresada, también me han ayudado en la realización de esta tesis.

Gracias a todos por vuestra ayuda, tiempo y dedicación.

Alcalá de Henares, 25 de Julio de 2010

José Luis Castillo Sequera

Resumen

El conocimiento explicito de las organizaciones se encuentra recogido en colecciones documentales controladas, a disposición de sus usuarios. Cuándo el número de documentos es elevado se necesitan herramientas para organizar y mostrar los contenidos de la colección, que permitan y faciliten a los usuarios explorar la colección para conocer mejor su naturaleza y descubrir relaciones, patrones, tendencias, y otras características para poder así "comprender" la información.

La necesidad de usar conocimientos en los Sistemas de Recuperación de Información empujó a los investigadores a analizar los sistemas inteligentes que procuran incorporar y usar dichos conocimientos con la finalidad de optimizar el sistema. En la presente tesis, se muestra un Sistema Evolutivo (SEV), y los resultados obtenidos en la construcción de un sistema de esta naturaleza.

En este trabajo hacemos una aportación en el área de Recuperación de Información (RI), proponiendo el desarrollo de un nuevo sistema que, utilizando técnicas evolutivas, implemente un sistema de aprendizaje del tipo no supervisado, para agrupar los documentos de un Sistema de Recuperación de Información (SRI); en donde los grupos y el número de ellos son desconocidos a *priori* por el sistema. El criterio para realizar el agrupamiento de los documentos estará basado por la similitud y distancia de los documentos, formando así de esta manera grupos ó *clustering* de documentos afines, permitiendo así agrupar los documentos de un SRI de una manera aceptable, presentándose como una alternativa válida a los métodos de agrupamiento tradicionales, pudiéndose contrastar sus resultados experimentalmente con algunos de los métodos clásicos.

Los lexemas más relevantes de cada documento, obtenidos mediante la aplicación de técnicas de RI, permiten enriquecer la información asociada a los documentos de la colección y utilizarlos como valores de metadatos para el algoritmo evolutivo. De esta forma, el sistema funciona mediante una metodología de procesamiento de documentos que selecciona los lexemas de los documentos mediante criterios de recuperación de información.

Los resultados obtenidos demuestran la viabilidad de la construcción de una aplicación a gran escala de estas características, para integrarla en un sistema de gestión de conocimiento que tenga que manejar grandes colecciones documentales controladas.

Abstract

Explicit knowledge of the organizations is kept in highly controlled document collections, available to its users. A large document collection requieres tools to organize and reveal its content, that allow users to easily explore it, so as to better get to know its type and discover relations, patterns, trends and other features in order to “understand information”.

The need for expertise in Information Retrieval Systems pushed researchers to analize intelligent systems that seek to incorporate and use such knowledge in order to optimize the system. In this thesis, it is shown an evolutionary system (EVS), and the results obtained with the construction of a system of this nature.

In this paper we make a contribution in the field of Information Retrieval (IR), proposing the development of a new system using evolutionary techniques, implement a system for unsupervised learning type, to group documents in an Information Retrieval System (IRS) where their groups and number of are unknown *a priori* by the system. The criteria used to create document clusters will be based on the similarity and distance of the documents, thus forming groups or clusters of related documents, allowing document clustering of a IRS in an acceptable manner, presenting as a valid alternative to traditional clustering methods, being able to compare their experimental results with some traditional methods.

The most relevant lexemes of each document, obtained by applying IR techniques, to enrich the information associated with documents in the collection and use them as metadata values for the evolutionary algorithm. Thus, the system works through a document processing method which selects the lexemes of documents using information retrieval criteria.

The results prove the feasibility of building a large-scale application of this type in order to integrate it into a knowledge management system that needs to handle large controlled document collections

Tabla de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Objetivos y metodología

En este capítulo planteamos los objetivos y el método de esta investigación. Además se mencionan las principales aportaciones, que iremos detallando a lo largo del documento.

1.1. Introducción

En pocos años, la tecnología de la red de ámbito mundial —*World Wide Web* — se ha convertido en una herramienta universal para todo tipo de actividades culturales, profesionales y comerciales. Los avances tecnológicos de los últimos años han provocado un aumento exponencial de la información mediada por dicha red. El proceso de digitalización y la transformación de documentos que se está llevando a cabo son dos claros ejemplos de la revolución de la información, la cual ha permitido su acceso a un número ilimitado de usuarios.

En este ámbito, la Recuperación de Información (RI) se puede definir como el problema de la selección de información desde un mecanismo de almacenamiento en respuesta a consultas realizadas por el usuario [Frakes, Baeza Yates, 1992]. Es muy importante distinguirlo de la Recuperación de Datos, y mencionar las diferencias entre estos conceptos, como indica Van Rijsbergen [Van Rijsbergen, 1979] y se recoge en la siguiente tabla 1.1.

	Recuperación de Datos	**Recuperación de Información**
Correspondencia	Coincidencia exacta	Coincidencia parcial, mejor coincidencia
Inferencia	Deducción	Inducción
Modelo	Determinista	Probabilística
Clasificación	Una Categoría	Múltiples categorías
Lenguaje de consulta	Artificial	Natural
Especificación de la consulta	Completa	Incompleta
Elementos deseados	Coincidente	Relevante
Influencia de un error	Sensible	Insensible

Tabla 1.1: ¿Recuperación de Datos o de Información?

Los sistemas de recuperación de información (SRI) son una clase de sistemas de información que tratan con bases de datos compuestas por documentos, y procesan las consultas de los usuarios permitiéndoles acceder a la información relevante en un intervalo de tiempo apropiado. Estos sistemas fueron originalmente desarrollados en la década de los años 40 con la idea de auxiliar a los gestores de documentación científica.

En principio, un documento es un conjunto de datos de naturaleza textual, pero la evolución tecnológica ha propiciado la proliferación de documentos multimedia, añadiéndose al texto fotografías, ilustraciones gráficas, videos animados, audio, etc. Aunque la variedad, en cuanto a documentos se refiere, está aumentando tanto en soportes como en el carácter de su contenido, en este trabajo nos vamos a centrar solamente en aquellos documentos que tienen naturaleza textual.

Los SRI permiten la recuperación de la información, previamente almacenada por medio de la realización de una serie de consultas (*queries*) a los documentos contenidos en la base de datos. Estas preguntas son expresiones formales que representan las necesidades de información, y vienen expresadas por medio de un lenguaje de consulta.

La mayoría de los SRI comerciales se basan en el modelo booleano, el cual presenta algunas limitaciones. La principal de estas limitaciones es su criterio de recuperación binario, que es muy tajante y estricto por lo que se puede considerar un sistema de recuperación de datos más que de información. Debido a ello, se han diseñado otros paradigmas con el fin de extender este modo de recuperación y superar sus problemas, como por ejemplo, el modelo vectorial [Salton, 1971], el probabilístico y el modelo de RI difusa.

En los últimos años, se ha experimentado un interés creciente en la aplicación de técnicas de Inteligencia Artificial (IA) [Winston, 1994], [Russell Stuart, Norvig Peter,1996], y de Minería de Datos (Data Mining) [Berry Michael, 2004], [Berry Michael, et al, 2008] al campo de la RI con el propósito de subsanar las carencias de los SRI tradicionales. Un ejemplo de ello es el paradigma del Aprendizaje Automático [Mitchell Tom, 1997] basado en el diseño de sistemas con capacidad para adquirir conocimiento por si mismo y las técnicas avanzadas de Data Mining [Castro Felix, et al, 2007]: Redes Neuronales (Neural Networks), Algoritmos Evolutivos (AE), Métodos de Agrupamiento, Lógica Borrosa, Razonamiento Inductivo entre otros, con el fin de extraer diferentes tipos de conocimiento en la información que se procesa.

Los Algoritmos Evolutivos (AE) [Michalewicz, 1999] no son específicamente algoritmos de aprendizaje automático, pero ofrecen una metodología de búsqueda potente e independiente del dominio que puede ser aplicada a gran cantidad de tareas de aprendizaje [Grefenstette, 1995]. Debido a estas razones, la aplicación de los AE a la RI se ha incrementado en los últimos años.

Entre otros, los AE se han aplicado en la resolución de los siguientes problemas:

- Agrupamiento (clustering) de documentos y términos.
- Indización de documentos.
- Mejoras en la definición de consultas.
- Aprendizaje de funciones de similitud.

Centrándonos en el primer grupo, una de sus posibles aplicaciones aparece al tratar de resolver un problema básico al que se enfrenta un SRI: la necesidad de encontrar la categoría o agrupación que mejor describa a un documento.

La Agrupación de Documentos (*Document Clustering*) puede definirse como la tarea de separar documentos en grupos afines. El criterio de agrupamiento se basa en alguna similitud existente entre ellos.

Los términos clasificación "*classification*", y categorización o agrupamiento ("*categorization*" ó "*clustering*"), son utilizados con distinto significado [Lewis, 1991], [Maarek et. al, 2000]. El concepto de clasificación de documentos se refiere al problema de encontrar para cada documento la clase a la que pertenece [Yang, 1999], asumiendo que las clases están predefinidas y que se tienen documentos preclasificados para utilizar como ejemplos.

Un criterio de agrupamiento utilizado es dividir los documentos en una jerarquía de temas, un ejemplo de esto último son las categorías o grupos que presenta el buscador Yahoo ®. Un documento en particular se podría encontrar, por ejemplo, dentro de la categoría de: Tecnología ó de Informática ó de Internet. Hay una dificultad para encontrar el grupo que mejor describa a un documento, pues los documentos no tratan un sólo tema, y aunque lo hicieran, el enfoque con el que el tema es tratado puede hacer que el documento encuadre en otro grupo. La agrupación de documentos es una tarea compleja y subjetiva incluso para diferentes personas, que podrían asignar el mismo documento a grupos diferentes, cada una aplicando un criterio válido. [Macskassy et. al, 2001].

En la presente tesis, se estudia el agrupamiento de documentos, entendiéndose por esto al proceso de encontrar grupos dentro de una colección de documentos, basándose en criterios de similitud o de distancia entre los mismos, sin requerir un conocimiento *a priori* de otras características.

Se presenta el desarrollo de un Sistema Evolutivo (SEV) que utilizando técnicas evolutivas implementa un sistema de aprendizaje del tipo no supervisado, a fin de agrupar los documentos, donde los grupos son desconocidos *a priori* por el sistema. Una de las aportaciones del sistema propuesto se basa en la forma de representar a los individuos en el sistema, típica de la programación genética (PG). El método propuesto puede presentarse como una alternativa a los métodos de *clustering* tradicionales, tales como los jerárquicos y/o particionales basados en algoritmos conocidos como por ejemplo el *K-Means*.

La programación genética (PG) constituye una forma de computación evolutiva en la que los individuos de la población son programas, en lugar de cadenas de bits. Los programas usados en la programación genética suelen representarse mediante sus árboles sintácticos; cada llamada a una función se representa por un nodo en el árbol, y los argumentos de la función corresponden con los nodos hijos de éste.

Para aplicar programación genética a un dominio particular, es necesario que el usuario defina de antemano las funciones que se van a emplear. La programación genética realiza una búsqueda evolutiva en un espacio de programas que vienen descritos mediante sus árboles sintácticos, y al igual que los algoritmos genéticos, la programación genética itera sobre una población de individuos produciendo una nueva generación mediante el empleo de la selección, el cruce o la mutación.

Uno de los motivos para proponer en la presente tesis un sistema evolutivo con éste tipo de representación basado en PG, es que para resolver el problema de agrupamiento ("*Clustering*") se han llevado a cabo diferentes aproximaciones con AE, que se han aplicado con mayor o menor éxito, pero la mayoría de ellas usan el enfoque tradicional de Algoritmos Genéticos (AG), y muy pocas hasta donde sabemos están basadas en la representación de PG.

Por esta razón, el objetivo principal de esta memoria es proponer el desarrollo de un nuevo Sistema (SEV) que utilizando técnicas evolutivas agrupe documentos de forma no supervisada. El criterio para realizar el agrupamiento de los documentos estará basado en el criterio de similitud o de distancia de los documentos, formando así grupos ó *clusters* de documentos afines, presentándose así como una alternativa válida a los métodos de agrupamiento tradicionales y pudiéndose constatar sus resultados experimentalmente con algunos de los métodos clásicos.

Es de exigir que la nueva propuesta mejore la capacidad de agrupamiento de los documentos, como demostraremos con una experimentación práctica sobre diferentes conjuntos de documentos con el sistema evolutivo propuesto.

1.2. Planteamiento del problema

Las hipótesis de trabajo de las que partimos son las siguientes:

La necesidad de la agrupación automática de documentos

El objetivo de las técnicas de recuperación de información es poder resolver consultas en forma eficaz y eficiente. El criterio para evaluar la eficacia se basa en la relevancia de los resultados encontrados [Raghavan, Bollman, Jung, 1989], y la eficiencia viene dada por la rapidez con la cual se resuelve la consulta.

En el contexto de recuperación de información, Van Rijsbergen [Van Rijsbergen, 1979] formula la denominada "Hipótesis del Agrupamiento" (en inglés, "*Cluster Hipótesis*"). Básicamente la hipótesis del agrupamiento sostiene que "los documentos fuertemente asociados" tienden a ser relevantes para la misma consulta, lo cual ha sido verificado experimentalmente [Cutting et. al, 1992], [Schütze et. al, 1997], [Zamir, Oren, Etzioni, 1999]. Basándose en dicha hipótesis, la agrupación automática tiene en cuenta el contenido de los documentos para agruparlos, ya que documentos similares contendrán palabras (términos) similares.

La tarea de encontrar grupos de documentos con características comunes no sólo es compleja sino que además consume tiempo. Un bibliotecario que tratara de clasificar manualmente un documento tendría que leerlo para comprender su significado y luego asignarle una categoría utilizando su experiencia y sentido común. El costo y el tiempo asociados a este proceso ha incentivado la investigación de técnicas que permitan automatizar la tarea [Rüger, Gauch, 2000].

Debido al incremento en los volúmenes de información disponibles en forma electrónica y a la necesidad cada vez mayor de encontrar la información buscada en un tiempo mínimo, éstas técnicas de automatización mencionadas han estado recibiendo creciente atención [Maarek et al, 2000], [Strehl et al, 2000], [Wong, 2002], [Ye, 2003], [Berry Michael, et al, 2008], [Olson David, 2008].

Sus aplicaciones más importantes son:

- Mejorar el rendimiento de los motores de búsqueda de información, mediante la agrupación previa de todos los documentos disponibles [Faloutsos et al, 1996], [Zamir, Oren, Etzioni, 1999]. Antes de comenzar a resolver las consultas, el conjunto de documentos es separado en grupos. Luego, al resolver una consulta, no se examinan todos los documentos, sino que se busca el "grupo representante" que mejor responda a la consulta.

- Facilitar la revisión de resultados por parte del usuario final, agrupando los resultados luego de realizar la búsqueda [Cutting et al, 1992], [Allen et al,1993], [Maarek et al, 2000]. Cuando se realizan búsquedas sobre un gran volumen de documentos, la cantidad de resultados puede ser muy grande. Si la lista se presenta sin ningún tipo de procesamiento previo, el usuario del sistema se verá obligado a revisar todos los documentos, descartando aquellos que no son relevantes para él. Si los resultados se agrupan, los documentos del mismo grupo serán relevantes para una subconsulta más específica que la original; de esta forma, los documentos quedarán separados en grupos

temáticos. Así, con sólo revisar uno o dos documentos de cada grupo, el usuario podrá determinar cuál es el subtema al que responden todos los documentos del grupo, pudiendo descartar rápidamente los documentos irrelevantes para su búsqueda.

- Aplicaciones al campo de la Minería de Datos, que permita extraer conocimiento de los documentos procesados, a través del análisis de la información obtenida en los agrupamientos.

La disponibilidad de los Algoritmos Evolutivos y de la Recuperación de Información

Los distintos modelos computacionales que se han propuesto dentro de la Computación Evolutiva [Bäck, Fogel, Michalewicz, 1997], suelen recibir el nombre genérico de Algoritmos Evolutivos (AE) [Bäck, 1996]. Existen cuatro tipos de AE bien definidos que han servido como base a la mayoría de los trabajos desarrollados en el área: Los Algoritmos Genéticos (AG) [Goldberg, 1989], [Holland, 1975], las Estrategias de Evolución (EE) [Bäck, 1996], [Schwefel, 1995], la Programación Evolutiva (PE) [Fogel L, 1966] y la Programación Genética (PG) [Koza, 1992].

Un AE funciona manteniendo una población de posibles soluciones del problema a resolver, llevando a cabo una serie de alteraciones sobre las mismas, y efectuando una selección para determinar cuáles permanecen en generaciones futuras y cuáles son eliminadas. Aunque todos los modelos existentes siguen esta estructura general, existen algunas diferencias en cuanto al modo de ponerla en práctica.

Los AG se basan en operadores genéticos tales como el cruce y la mutación, los cuales son aplicados a individuos que codifican posibles soluciones de un problema, intentando de esta forma imitar los procesos de evolución existentes en la naturaleza. En cambio, las EE y la PE aplican transformaciones basadas en mutaciones efectuadas sobre los padres para obtener los hijos, lo que permite mantener la línea general de comportamiento del individuo en su descendencia. Finalmente, la PG codifica las soluciones al problema en forma de programas, habitualmente representados mediante estructuras arborescentes, y adapta dichas estructuras empleando operadores muy específicos.

Cada individuo de la población recibe un valor, basado en una medida de adaptación que representa su grado de adecuación al entorno. La selección hace uso de estos valores y se centra en los individuos que presentan mejor valor en dicha medida. Los operadores de cruce o mutación alteran la composición de dichos individuos, guiando heurísticamente la búsqueda a través del espacio. Aunque simples desde un punto de vista biológico, este tipo de algoritmos son suficientemente complejos como para proporcionar mecanismos de búsqueda adaptativos muy robustos. Los mismos procedimientos pueden aplicarse a problemas de distintos tipos, sin necesidad de hacer muchos cambios.

Debido a que, en los últimos años, se ha experimentado un interés creciente en la aplicación de técnicas basada en la Inteligencia Artificial (IA) al campo de la RI con el propósito de resolver distintos problemas específicos de esta última área. En concreto, el enfoque evolutivo basado en la evolución natural, constituye una alternativa interesante para el área de la RI.

Los AE ofrecen una metodología de búsqueda potente e independiente del dominio que puede ser aplicada a gran cantidad de tareas de aprendizaje. De hecho, los AE han sido muy utilizados en problemas de aprendizaje automático de reglas de producción a partir de conjunto de ejemplos [Grefenstette, 1995], [Cordon, Herrera, Hoffman, Magdalena, 2001]. Este amplio uso se debe a que, cuando los individuos de la población genética representan conocimiento, pueden ser tratados al mismo tiempo como datos que son manipulados por el AG y como código ejecutable que lleva a cabo cierta tarea [Michalewicz, 1999].

Debido a estas razones, la aplicación de los AE a distintos campos de la ciencia se ha incrementado considerablemente. Un claro ejemplo lo constituye, su aplicación al área de la RI, tal y como lo demuestra el gran número de publicaciones aparecidas en la literatura especializada.

Entre otros, los AE se han aplicado en la resolución de los siguientes problemas dentro del marco de la RI [Cordon, Herrera-Viedma, Zarco, 2003]:

- Agrupamiento (clustering) de documentos y términos [Robertson, Willet, 1994] y otras propuestas.

- Indización de documentos mediante el aprendizaje de los términos relevantes para describirlos [Gordon, 1991] y de sus pesos [Vrajitoru, 1998].

- Mejoras en la definición de consultas : Aprendizaje de los términos de la misma a partir de un conjunto de documentos relevantes para el usuario, de los pesos de los términos proporcionados previamente por el usuario [Robertson, Willet,1996], [Yang, Korfhage,1994] ó de la composición de la consulta completa aprendiendo los términos y los operadores Booléanos [Smith, Smith,1997], o los dos anteriores y los pesos de los términos [Kraft et. al, 1997].

- Aprendizaje de las funciones de similitud [Pathak, Gordon, Fan,2000], [Fan,Gordon, Pathak,1999].

La posibilidad de utilizar los Algoritmos Evolutivos para el agrupamiento de documentos

Las causas mencionadas anteriormente han motivado la búsqueda de otro tipo de técnicas que pudieran subsanar las limitaciones de los algoritmos clásicos, dado que el problema a tratar se puede considerar como un problema de optimización en un espacio finito de gran tamaño, se trata de un problema de optimización combinatoria NP completo. Por tanto, la utilización de las nuevas técnicas estocásticas de optimización provenientes de la IA (como por ejemplo los AG) parece totalmente justificada.

Del paradigma de Algoritmos Evolutivos (AE) se observa que lo que más se ha aplicado en el área de agrupamiento son los AG, habiendo casi muy poco de la aplicación de las otras técnicas de AE (PG, EE, etc) al área de agrupamiento.

Hasta donde alcanza nuestro conocimiento, una de las pocas aproximaciones que se ha llevado a cabo en el campo de la RI utilizando la PG, está orientada al aprendizaje de consultas booleanas extendidas de un modo automático a partir de un

conjunto de documentos relevantes para un usuario, llamada propuesta de Kraft y otros, comentada en [Zarco Carmen, 2003]. Dicho método se basa en un algoritmo de PG, el cual se caracteriza por su potencialidad, pero también por su dificultad a la hora de derivar coeficientes numéricos.

Otra investigación relacionada con este tema es el trabajo de tesis doctoral [Zarco Carmen, 2003], que aplica algoritmos evolutivos al aprendizaje automático de consultas booleanas extendidas para sistemas de recuperación de información difusos. En ella se indica un modelo de representación de consultas (que serán vistas como los individuos del proceso evolutivo) sobre el que se desarrolla el modelo de representación de configuraciones del conjunto de documentos estudiado en el presente trabajo.

De ahí, como punto de partida, aparece el desarrollo de un Sistema Evolutivo cuyos individuos (configuraciones de documentos) se representan en los términos característicos de la PG, utilizando operadores de AE orientados específicamente a mejorar el agrupamiento de documentos, con una función de adaptación que combine las medidas de distancia y/o similitud.

A partir de esta idea, desarrollaremos un Sistema del tipo Evolutivo para agrupar los documentos de una colección documental. Para finalmente, comparar los resultados obtenidos por nuestra propuesta, con otros métodos de agrupamiento conocidos orientados a realizar la misma tarea, y así de esta manera validar nuestra propuesta experimental.

1.3. Objetivos y aportaciones originales

Como se desprende de la sección anterior, el principal propósito de esta tesis es definir un modelo evolutivo que, aplicando técnicas de IA, permita representar agrupamientos aplicables al ámbito de los Sistemas de Recuperación de Información (SRI) para colecciones documentales. El modelo contemplará, además de los elementos necesarios para representar los documentos existentes, la información relevante que permita aplicar un aprendizaje del tipo no supervisado sobre el sistema.

Para alcanzar este objetivo general se plantean los siguientes sub-objetivos:

1 Obtener un modelo de representación de los documentos (en este caso los vectores documentales ó de características) utilizando un método de procesamiento basado en los conceptos de RI. El modelo debe permitir representar el contenido de los documentos del SRI de manera fiable; para poder realizar las pruebas experimentales con el sistema propuesto. Para ello se estudiarán los distintos modelos de RI haciendo una evaluación crítica de estos modelos, teniendo en cuenta sus limitaciones y posibles mejoras, con el fin de aportar posibles soluciones a los defectos que presenten.

2 Investigar la forma de representar los documentos para poder ser tratados por los algoritmos de computación evolutiva. El hecho de trabajar con una colección de documentos en español, añade interés a los resultados obtenidos, ya que la mayoría de las colecciones existentes están disponibles en el idioma inglés.

3 Aplicar los AE al campo de la RI. Los AE constituyen una buena alternativa para resolver algunos de los problemas existentes en la RI.

4 Proponer un modelo de Sistema Evolutivo (SEV) para el agrupamiento automático de documentos. Diseñaremos y desarrollaremos un sistema que utilizando técnicas evolutivas implemente un sistema de aprendizaje del tipo no supervisado, que permita agrupar documentos, donde los grupos y el número de ellos sean desconocidos a priori por el sistema, la evolución se basará en el criterio de similitud y distancia de los documentos.

5 Analizar experimentalmente el funcionamiento del SEV propuesto, empleando colecciones documentales estándares (utilizadas por numerosos investigadores del área) dentro del dominio de la agrupación automática de documentos, lo que garantizará su aplicabilidad a cualquier tipo de documentos. Así, de este modo, se trabajará con la colección clásica Reuters "21578", y con los editoriales del diario El Mundo de los años 2006 y 2007 para los experimentos con documentos en español.

Como aportaciones significativas que surgen como resultado de nuestro proceso de investigación, cabe destacar las siguientes:

- Estudio e implementación de nuevos operadores genéticos que permitan optimizar el proceso de agrupamiento de documentos.

- Estudio e implementación de una nueva función de adaptación que optimice los grupos creados, basado en las métricas de similitud y distancia de los documentos de la colección documental.

- Aplicación real del modelo. Al realizar las pruebas del sistema con una distribución estándar dentro de la comunidad de investigadores dedicados a la agrupación automática de documentos (Reuters 21578), y con los editoriales de los años 2006 y 2007 del diario El Mundo para documentos en español.

- Validación del modelo propuesto, mediante la comparación de los resultados con los conocidos existentes, y comparando sus resultados con otro método conocido de agrupación.

1.4. Justificación de la investigación

En la actualidad, la investigación acerca de las técnicas para descubrir conocimiento contenido nuevo, implícito, en las grandes colecciones documentales es objeto de interés. Cada vez un mayor número de empresas y particulares, almacena de alguna forma más o menos estructurada una creciente cantidad de documentos de todo tipo. La forma de agrupar los documentos de estas colecciones, es todavía un asunto que se encuentra insatisfactoriamente resuelto. Las propuestas de esta tesis se enmarcan pues, en un contexto de plena actualidad y pretende contribuir a enriquecer lo hecho para tratar este problema.

El modelo presentado en este trabajo de tesis puede ser incorporado, a cualquier sistema de minería de datos. Gran parte de los algoritmos y las técnicas utilizadas en el presente estudio han sido muy poco aplicados a colecciones de documentos en español. Los resultados obtenidos, pueden servir como base para futuros trabajos en este campo. En esta tesis se propone una metodología de procesamiento de documentos sobre las colecciones documentales, y un sistema evolutivo que se nutre de dicha metodología para agrupar documentos.

1.5. Método de trabajo

En esta sección se describe la planificación general del trabajo, así como el método de evaluación seguido.

1.5.1. Planificación general

La metodología seguida en esta investigación comprende las siguientes fases:

1. **Estudio del estado de la cuestión**: Se estudian, por un lado, las técnicas existentes para llevar a cabo agrupamiento de documentos, identificando las ventajas y desventajas de las mismas. y las características de cada una de ellas; y por otro lado, el aporte de la Inteligencia Artificial, y específicamente la Computación Evolutiva en el campo de los Sistemas de Recuperación de Información.

2. **Descripción del contexto del problema**: El problema se plantea en el marco de la utilización de una colección de datos documental que esté representada con vectores documentales, los cuales se pretende agrupar en base a ciertos criterios evolutivos. A fin de seleccionar de manera fiable la muestra de datos, se presenta la necesidad de que dichos documentos tengan un “procesamiento previo”. El enfoque del planteamiento parte de la consideración de las distintas carencias en la representación adecuada de los documentos, que se deben mejorar al aplicar técnicas de RI.

3. **Definición del modelo**: Se define de manera precisa el modelo planteado para representar los diferentes grupos de documentos que se pueden obtener en un SRI, partiendo exclusivamente del enfoque evolutivo que se propone. Más concretamente, se utiliza la computación evolutiva para definir el modelo del sistema propuesto. y luego derivar los agrupamientos basados en un *fitness* o función de adaptación que tenga en cuenta el concepto de maximizar la similitud de los documentos y minimizar la distancia de los mismos, componiendo ambas métricas en un nuevo operador (que será la función de adaptación), que tendrá que ir afinándose poco a poco para lograr el objetivo propuesto por el sistema.

4. **Validación del modelo**: La validación del modelo propuesto se realizará comparando los resultados obtenidos por el sistema al finalizar la evolución con otro método clásico de agrupamiento, específicamente el método llamado Kmeans y verificando el porcentaje de aciertos obtenidos entre ambos. Estudiamos los grupos de documentos que se obtengan, y definimos otras nuevas características del modelo para ajustar los parámetros del modelo.

5. **Formulación de conclusiones**: Se concluirá con un análisis del ámbito y grado de cobertura de los objetivos alcanzados. Las líneas futuras de investigación y posibles ampliaciones al trabajo realizado.

1.5.2. Estructura de la memoria

La presente memoria está compuesta por esta Introducción, una Parte Teórica y una Parte Experimental, así como un capítulo de Conclusiones y trabajos futuros en la que se incluyen las conclusiones generales de la misma y los posibles trabajos futuros a desarrollar.

El **primer capítulo** ha descrito el planteamiento, objetivos y contenidos de esta tesis.

El **segundo capítulo** constituye la **parte teórica** y está compuesto por **tres apartados** cuyos contenidos resumimos brevemente a continuación:

En el **primer apartado** vamos a repasar los conceptos básicos de la RI, analizaremos los sistemas que se emplean en este campo, que denominaremos SRI, analizando sus componentes principales. Estudiaremos los distintos modelos de recuperación que se han propuesto en la literatura, y la manera como se enfoca su evaluación, presentando luego el problema de la agrupación automática de documentos como una alternativa para mejorar la eficiencia de un SRI.

Describiremos su utilidad y las causas que han despertado su interés para resolverlo; y mostraremos la imposibilidad de encontrar la solución al problema de agrupación automática mediante una búsqueda exhaustiva, lo que finalmente lleva a la creación de algoritmos que encuentran soluciones aproximadas al problema.

Mostraremos el estado actual de los campos de estudio relacionados con esta tesis, definiendo formalmente el problema de la agrupación automática de documentos, el cual lo vincularemos con las ciencias de la "*Recuperación de Información*" y de la "*Minería de Datos*". Del campo de la Recuperación de Información toma los conceptos y métodos utilizados para el procesamiento de documentos. Estas técnicas son las que permiten transformar la información no estructurada que

contienen los documentos (llamados “documentos de texto libre”) a estructuras de datos manejables mediante algoritmos computacionales. Del campo de la Minería de Datos toma las técnicas que se ocupan de los problemas de agrupación de objetos.

Los algoritmos utilizados para la agrupación automática de documentos son adaptaciones de los que se utilizan para el problema más general de agrupación de objetos de cualquier tipo; mostraremos la ubicación de la agrupación automática de documentos dentro de esta área, y detallaremos los algoritmos utilizados actualmente para la agrupación de documentos.

El **segundo apartado** de esta memoria está dedicado al estudio de los AE. Para ello, repasaremos en primer lugar las características genéricas de esta familia de algoritmos, profundizando en los dos tipos más conocidos y empleados, los AG y la PG, que serán los considerados en este trabajo. Posteriormente, analizaremos los distintos enfoques evolutivos existentes para afrontar problemas de optimización que serán útiles para comprender nuestra propuesta que realizaremos en la parte experimental.

En el **tercer apartado** haremos una presentación de las aplicaciones de los AE en el área de la RI, proporcionando una taxonomía de las mismas (agrupamiento, indización, aprendizaje de consultas y aprendizaje de la función de similitud), describiendo varias técnicas de agrupamiento que se han efectuado utilizando algoritmos evolutivos. Analizaremos los trabajos previos de la aplicación de los AG a este problema.

La **parte experimental** la componen cuatro capítulos que describiremos a continuación:

El **tercer capítulo** se centrará en la descripción de un marco de trabajo genérico que permita diseñar el modelo de experimentación que aplicaremos para validar el sistema evolutivo propuesto. Describiremos las colecciones documentales y los clusters considerados, el proceso seguido para la indización de las mismas, y las condiciones consideradas para la experimentación que se han realizado, así como el método de procesamiento propuesto para obtener los vectores documentales de los documentos del SRI. Así de esta forma describiremos las pruebas que se realizaron para evaluar la efectividad de la solución propuesta, el conjunto de datos utilizados: La colección de datos utilizada Reuter 21578; y también nuestra colección documental en español, preparada con los editoriales del diario El Mundo. Detallaremos la metodología seguida en la experimentación, enumerando las variables que intervienen en los experimentos y los distintos tipos de pruebas realizadas para obtener los vectores documentales.

El **cuarto capítulo** se centrará en el estudio de la propuesta de la presente tesis, el diseño y desarrollo del Sistema Evolutivo propuesto, el cual recogerá para su concepción la representación de *PG***.** Describiremos de manera rigurosa cada uno de los componentes que lo conforman, así como el enfoque evolutivo que utilizaremos para representar el problema de agrupamiento de documentos en un SRI. Detallaremos los operadores de cruce y mutación que fueron diseñados específicamente para el sistema, así como la función de adaptación que usamos para evaluar la distancia y similitud entre los documentos. Veremos el porqué de su utilización, y su aplicación práctica con los vectores documentales obtenidos con el método desarrollado para procesar la colección documental.

El **quinto capítulo** efectuará un estudio global de la calidad de las propuestas realizadas en comparación con el algoritmo empleado como base, comparando los resultados obtenidos con otros métodos clásicos de agrupamiento. Haremos un análisis metodológico que permita encontrar los parámetros óptimos a utilizar para obtener los mejores resultados con el sistema propuesto. Específicamente, las pruebas se realizaran comparando nuestros resultados experimentales, bajo diferentes condiciones contra los resultados experimentales de otra técnica de agrupamiento, a fin de evaluar la capacidad de agrupamiento del sistema propuesto. En concreto, se estudiarán dos aspectos, la capacidad del sistema para encontrar buenos grupos de documentos y su robustez de los resultados aportados en las ejecuciones realizadas en cada caso.

Finalmente, el **sexto capítulo,** "Conclusiones y trabajos futuros" resumirá los resultados obtenidos en esta memoria, presentará algunos comentarios sobre los mismos y planteará posibles trabajos futuros que se pueden abordar en el área.

Capítulo 2

Estado actual de la investigación

El objetivo de este capítulo es presentar el estado del arte en el campo de la Recuperación de Información (RI) y los Algoritmos Evolutivos (AE), así como los trabajos y aplicaciones de los AE en el área de la RI.

Se analizarán los sistemas de información que se emplean en este campo, a lo que denominaremos SRI, mostrando sus componentes principales; y los distintos modelos de recuperación que se han propuesto en la literatura, presentando luego el problema de la agrupación automática de documentos como una alternativa para mejorar la eficiencia de un SRI, describiendo su utilidad y las causas que han despertado el interés para resolverlo. Posteriormente, estudiaremos los AE, repasando las características genéricas de esta familia de algoritmos, profundizando en los dos tipos más conocidos y empleados, los AG y la PG, que serán los considerados en este trabajo

Finalmente, haremos una presentación de las aplicaciones de los AE en el área de la RI, proporcionando una taxonomía de las mismas (agrupamiento, indización, aprendizaje de consultas y aprendizaje de la función de similitud), describiendo varias de las aplicaciones concretas que se han efectuado, analizando la problemática encontrada, y profundizando en las técnicas de agrupamiento utilizadas con algoritmos evolutivos. Analizaremos los trabajos previos que existen acerca de la aplicación de los AE a este problema; así como también las limitaciones que presentan los algoritmos actuales, relacionados con la forma en la que exploran el espacio de posibles soluciones.

APARTADO I: ESTUDIO SOBRE LA RECUPERACION DE INFORMACION

2.1. Introducción a los Sistemas de Recuperación de Información

Los avances tecnológicos de los últimos años han provocado un aumento exponencial de la cantidad de información producida y a gestionar. El proceso de digitalización y la transformación de documentos que se está llevando a cabo son dos claros ejemplos de la revolución de la información, la cual ha permitido su acceso a un número ilimitado de usuarios.

El uso masivo de las tecnologías y de los ordenadores está presente en todos los ámbitos de la vida, sobre todo en el trabajo, y hasta en el hogar donde cada vez es mayor el número de personas que no sólo tienen ordenador sino que poseen equipos multimedia con conexión a internet.

A ello habría que sumar la distribución de información mediante las llamadas *"autopistas de la información"*, y el coste cada vez menor de los medios de almacenamiento. Todo ello nos sitúa dentro de un entorno en desarrollo de información electrónica a la que se puede acceder por medios automáticos. Otro aspecto que tenemos que considerar es la diversificación de los medios, que trae consigo una mayor cantidad de información no normalizada, imagen, sonido, texto, etc.

La Recuperación de Información (RI) se puede definir como el problema de la selección de información desde un mecanismo de almacenamiento en respuestas a consultas realizadas por un usuario [Baeza-Yates, Ribeiro Neto, 1999].

Los Sistemas de Recuperación de Información (SRI) son una clase de sistemas de información que tratan con bases de datos compuestas por documentos y procesan las consultas de los usuarios permitiéndoles acceder a la información relevante en un intervalo de tiempo apropiado (véase la figura 2.1). Estos sistemas fueron originalmente desarrollados en la década de los años 40 con la idea de auxiliar a los gestores de la documentación científica.

En principio, un documento es un conjunto de datos de naturaleza textual, aunque la evolución tecnológica ha propiciado la proliferación de documentos multimedia, incorporándose al texto fotografías, ilustraciones gráficas, video animado, audio, etc. Aunque la variedad en cuanto a documentos se refiere está aumentado tanto en soportes como en el carácter de su contenido, nosotros nos centraremos en este trabajo en los que tienen naturaleza textual.

Un SRI permite la recuperación de la información, previamente almacenada, por medio de la realización de una serie de consultas (*queries*) a los documentos contenidos en la base de datos. Estas preguntas son sentencias formales de expresión de necesidades de información y suelen venir expresadas por medio de un lenguaje de consultas.

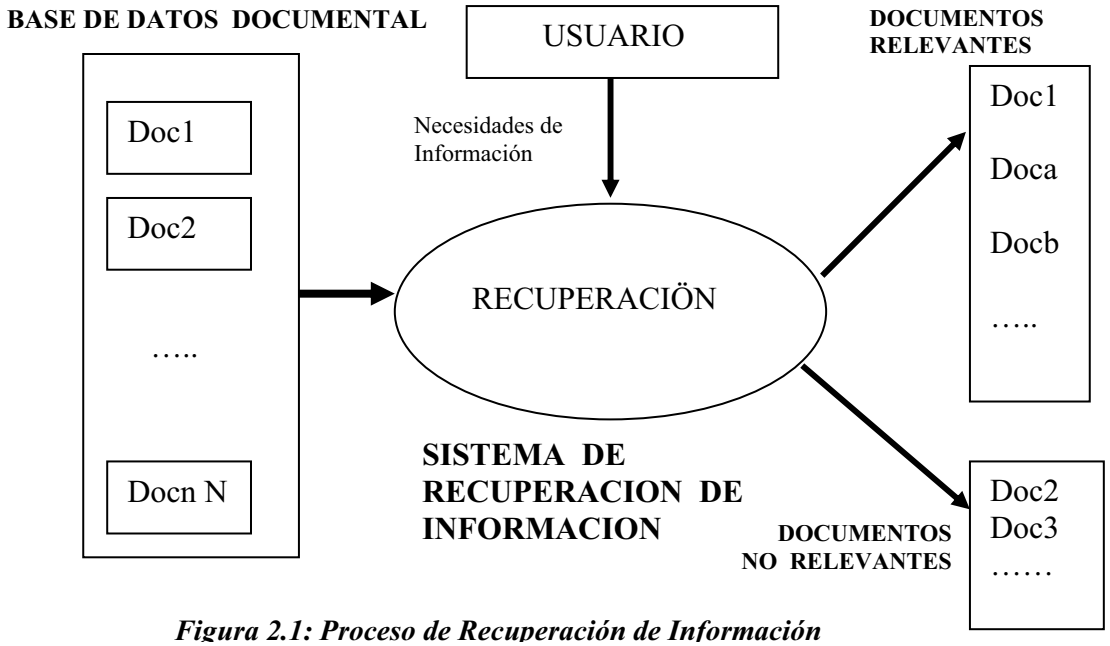

Figura 2.1: Proceso de Recuperación de Información

Un SRI debe soportar una serie de operaciones básicas sobre los documentos almacenados en el mismo, como son: introducción de nuevos documentos, modificación de los que ya estén almacenados y eliminación de los mismos. Debemos también contar con algún método de localización de los documentos para presentárselos posteriormente al usuario. Este proceso se resume en la figura 2.2.

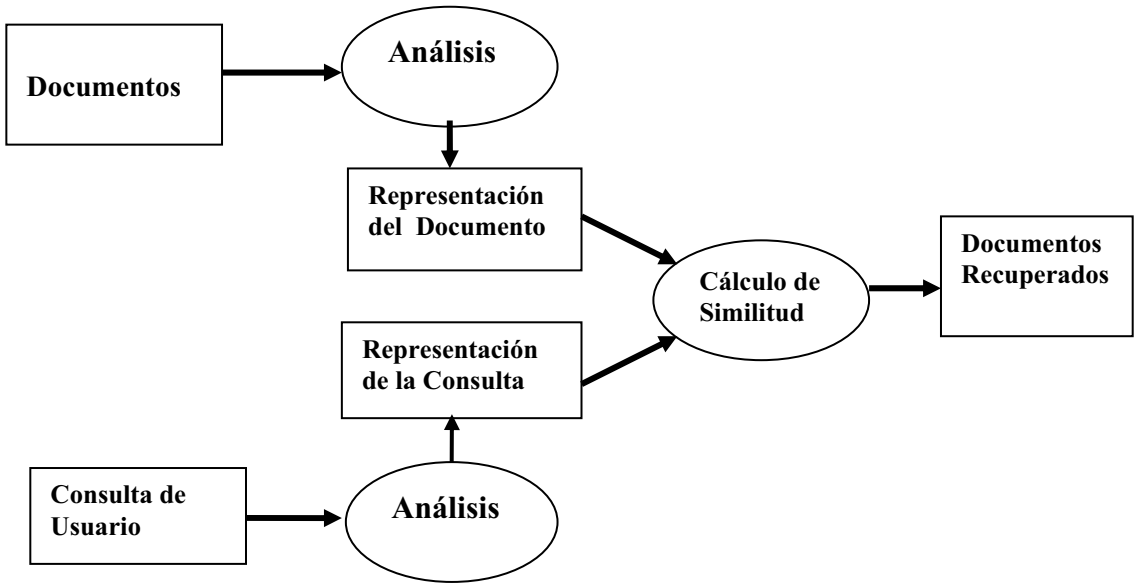

Figura 2.2: Operaciones para la recuperación de documentos

Los SRI implementan estas operaciones de varias formas distintas, lo que provoca una amplia diversidad en lo relacionado con la naturaleza de los mismos. Por tanto, para estudiarlos es necesario establecer en primer lugar una clasificación de estos sistemas. Para ello, empezaremos analizando cuáles son los componentes principales de un SRI.

2.2. Componentes de un Sistema de Recuperación de Información

Un SRI está compuesto por tres componentes principales: la base de datos documental, el subsistema de consultas y el mecanismo de emparejamiento ó evaluación (véase figura 2.3).

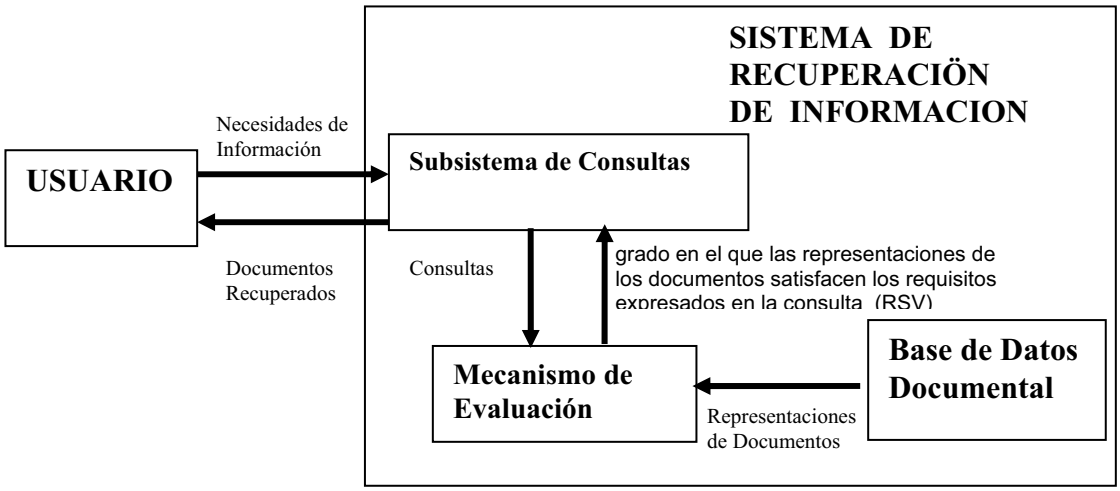

Figura 2.3: Composición genérica de un Sistema de Recuperación de Información

2.2.1. La base de datos documental

Un documento contiene datos de forma usualmente textual, aunque la evolución tecnológica ha propiciado la profusión de documentos multimedia, añadiendo al texto fotografías, ilustraciones gráficas, vídeos animados, audios, etc. Estos documentos no se introducen directamente en el SRI, sino que estarán representados por unos elementos llamados descriptores. La razón de usar estos descriptores es darle una mayor eficiencia al proceso de recuperación, permitiendo que el tiempo de búsqueda en ella sea mucho menor.

Por tanto, un documento se caracterizará de una serie de descriptores. Desde un punto de vista matemático, estos se agrupan mediante una tabla o matriz en la que cada columna indica las asignaciones de un determinado descriptor y cada línea o fila corresponde a un documento. En principio, en cada fila aparecen "*unos*" en las columnas relativas a los descriptores asignados al documento y "*ceros*" en las restantes, así cada documento estará representado por un vector de ceros y unos [Van Rijsbergen, 1979]. Se puede pensar que esta representación se podría mejorar introduciendo información ponderada sobre la asignación de un descriptor al documento, en lugar de simplemente 0 y 1. Como veremos a continuación, esta representación se tendría que hacer teniendo en cuenta todos los documentos y descriptores.

La información numérica de la asignación de un concepto a un documento puede tener diferentes significados dependiendo del modelo de recuperación que se trate. Por ejemplo, en el modelo de *Espacio Vectorial* [Salton, McGill, 1983] puede considerarse como el grado en el que ese descriptor describe el documento; mientras que en el modelo *Probabilístico* [Bookstein, 1983] se considera como la probabilidad de que el documento sea relevante para ese descriptor.

Podemos considerar una base documental *D*, compuesta por documentos d_i, indexada por un conjunto de *m* términos t_j, donde cada documento d_i contiene un número no especificado de términos de indización t_j. De esta forma sería posible representar cada documento como un vector perteneciente a un espacio *m-dimensional*, siendo *m* el número de términos de indización que forman el conjunto *T*:

$$d_i = (t_{i1}, t_{i2}, t_{i3}, \ldots\ldots\ldots t_{im})$$

Donde cada uno de los elementos t_{ij} puede representar la presencia o ausencia del término t_j en el documento d_i para el modelo de indización binaria, la relevancia del término t_j en el documento d_i para el modelo de espacio vectorial, o la probabilidad de que el documento d_i sea relevante al término t_j en el modelo probabilístico.

La indización (*proceso de construcción de los vectores documentales*) puede realizarse de forma manual, o automática. En este último caso, la base de datos documental comprende un módulo llamado módulo indizador que se encarga de generar automáticamente la representación de los documentos extrayendo los contenidos de información de los mismos.

La labor del módulo indizador consistirá en asociar automáticamente una representación a cada documento en función de los contenidos de información que tenga. Es decir, determinar los pesos de cada término en el vector documental, su función de indización o ponderación será:

$$F : D \; x \; T \rightarrow \{0, 1\}$$

La representación de cada vector tendrá *m* componentes. Los que estén referenciados en el documento tendrán un valor diferente de 0, mientras que los que no estén referenciados tendrán un valor nulo o 0. Es importante señalar que la indización juega un papel fundamental en la calidad de la recuperación puesto que en virtud de ella se definen los coeficientes correspondientes.

De este modo, para obtener estas representaciones se aplica un conjunto de procesos que permitan “CONSTRUIR LA BASE DOCUMENTAL”. Para ello, solemos partir de una información en estado puro del documento (*información textual*). Los documentos de tipo textual se pueden representar bien por una componente estructurada en campos (título, autor, resumen, palabras clave, etc.) o bien por una componente no estructurada, es decir, el texto literal. La representación textual de cada documento se basará normalmente de los términos de indización, los cuales son identificadores de los propios documentos.

Es evidente que en una colección de documentos aparecerán cientos o miles de palabras distintas, debido a que la dimensión del espacio vectorial de representación está dado por todas las palabras diferentes, cada vector contendrá cientos o miles de componentes (gran parte de ellas con valor igual a cero) [Zervas et al, 2000]. Esto es conocido como la “*maldición de la dimensionalidad*” (en inglés “*the curse of dimensionality*”) [Yang, Pedersen, 1997]

Por ello, es importante reducir dicha dimensionalidad adecuadamente, y para ello existen dos técnicas llamadas: La reducción de palabras a su raíz, y la remoción de los términos poco discriminantes, que en este trabajo en el capítulo 3 se aplicarán metódicamente.

- **Reducción de palabras a su raíz**

Para reducir el número de términos distintos con los que se trabaja, el primer paso es reducir todas las palabras a su raíz (en inglés "*stemming*". Por ejemplo, las palabras "medicina, "médico" y "medicinal" se reducen a la forma "medic". Esta técnica es llamada "*remoción de sufijos*" (en inglés "*suffix stripping*"). Si bien es posible que de esta manera se lleven a la misma raíz palabras que en realidad tienen significados distintos, en general las palabras que tienen la misma raíz refieren al mismo concepto, y la pérdida de precisión es compensada por la reducción de la dimensionalidad del espacio [Van Rijsbergen, 1979].

Las aplicaciones de agrupamiento de documentos usan técnicas de remoción de sufijos [Krovetz, 1993],[Zamir, Oren, Etzioni, 1998],[Steinbach, Karypis, Kumar, 2000], existiendo consenso en cuanto a la conveniencia de su aplicación. La técnica de remoción de sufijos más utilizada es la llamada *regla de Porter*" [Porter, 1980].

- **Remoción de términos poco discriminantes**

Otro método para reducir la dimensión del espacio de términos (complementario del anterior) es el de descartar los términos que se consideran poco discriminantes [Yang, 1999]. Un término es poco discriminante si el hecho de saber que ese término se encuentra en un documento nos dice poco (o nada, acerca del posible contenido o tema del documento).El ejemplo más simple de dichos términos son las preposiciones, artículos y toda otra palabra auxiliar que ayude a dar forma a las oraciones. Éstas palabras se consideran poco discriminantes en cualquier colección documental con la cual se trabaje, por lo que se utiliza una lista de palabras irrelevantes (en inglés, "*stop list*") y se descartan todas las palabras que aparecen en esa lista [Van Rijsbergen, 1979], [Strehl et. al, 2000].

Existen términos que pueden ser discriminantes dependiendo del conjunto de documentos con los que se trabaje, es por ello que estos términos podrían también variar dependiendo del idioma en el cual se encuentre la base documental; tal como hacemos en nuestro trabajo en la parte experimental.

En [Yang, 1999] se exponen diversas técnicas orientadas a detectar, en una colección de documentos, qué términos son irrelevantes para la tarea de agrupación. Algunas de ellas requieren un pequeño grupo de documentos previamente agrupados, para ser utilizados como un conjunto de entrenamiento, por lo que a veces su aplicación puede no ser posible. A continuación describimos cada una de ellas:

a) Umbral de frecuencia documental

Es el método más sencillo. Se calculan las frecuencias con que aparece cada término en la colección documental y los términos que no superan cierto umbral mínimo son descartados. Se asume que las palabras muy poco frecuentes no contienen información sobre la categoría a la que pertenece el documento, o que son términos de una importancia marginal.

b) Ganancia de información, Información mutua, Estadística X^2

Teniendo un conjunto de documentos ya agrupados, para cada término se calcula (en base a fórmulas probabilísticas) qué tan bueno es para predecir la categoría o

grupo a la que pertenece el documento. El puntaje más alto es obtenido por aquellos términos presentes en todos los documentos de una categoría y que no aparecen en ningún documento de las demás categorías. Los términos con puntaje más bajo (que son aquellos que aparecen por igual en documentos de todas las categorías) son descartados.

c) Fuerza del término

Usando un grupo de documentos de entrenamiento, se toman aquellos pares de documentos cuya semejanza excede un valor determinado. Para cada término, se cuenta la cantidad de veces que el mismo aparece en ambos documentos de cada par. En base a eso, se calcula la probabilidad de que el término esté en el segundo documento del par, sabiendo que está en el primero. Se asignan puntajes más altos cuanto mayor sea esa probabilidad, asumiendo que el término es descriptivo de la categoría de esos documentos.

El análisis comparativo [Yang, Pedersen, 1997] usando cada uno de los métodos de remoción de términos para diferentes algoritmos de agrupamiento, llega a la conclusión de que las técnicas de Umbral de frecuencia, Ganancia de información y Estadística X^2 obtienen resultados similares. Como ventaja, el método de Umbral de frecuencia no requiere documentos previamente clasificados.

2.2.2. El subsistema de consultas

Este subsistema está compuesto por la interfaz que permite al usuario formular sus consultas al SRI y por un analizador sintáctico que toma la consulta escrita por el usuario y la desglosa en sus partes integrantes. Para llevar a cabo esta tarea, incluye un lenguaje de consulta que recoge todas las reglas para generar consultas apropiadas y la metodología para seleccionar los documentos relevantes.

La interfaz ofrecerá facilidades al usuario a la hora de formular su consulta, ya que éste no tiene porqué saber exactamente el funcionamiento tanto externo como interno del sistema. También se ocupará de mostrar al usuario el resultado de su búsqueda, una vez procesada su consulta.

Lo más habitual es que los usuarios de SRI realicen sus peticiones basándose en la estructura de consultas booleanas (con operadores booleanos, es decir, Y, O, NO). La consulta que facilite el usuario no puede procesarse directamente en su forma original: esta ha de recibir un tratamiento previo que consiste en desglosar la consulta en sus componentes básicos, además de comprobar que corresponde con el formato que se espera de ella (es decir, que su composición es correcta y cuadra con las reglas del lenguaje de consulta). Esta comprobación se podrá llevar a cabo tanto a priori como a posteriori. Si se realiza a priori, el sistema directamente no permite al usuario ejecutar su consulta hasta que no esté en el formato correspondiente. Si la comprobación se realiza a posteriori, el sistema devolverá un mensaje de error.

El análisis de la consulta se llevará a cabo mediante un analizador sintáctico, que determinará si la consulta es correcta o no y la desglosará en sus componentes. Después se podrá llevar a cabo el proceso de *stemming* para obtener las raíces de los términos de consulta.

Finalmente la consulta se indexará o vectorizará y será enviada al mecanismo de evaluación para estudiar qué documentos se consideran relevantes para las necesidades de información que representa.

2.2.3. El mecanismo de evaluación

Llegados a este punto, tenemos una representación del contenido de los documentos en nuestra base documental y también una representación de las consultas que queremos realizar del subsistema de consulta. Lo que nos queda por resolver es la selección de los documentos que se consideren relevantes, de entre los documentos que forman la base documental, de acuerdo con los criterios de nuestra consulta.

De esto precisamente se encargará el mecanismo de evaluación que evalúa el grado en el que las representaciones de los documentos satisfacen los requisitos expresados en la consulta y recupera aquellos documentos que son relevantes a la misma. Este grado es lo que se denomina *RSV* (*Retrieval Status Value*). Principalmente existen dos modalidades de evaluación: sistemas que emparejan los documentos individualmente con la consulta, uno por uno; y otros que los emparejan en su conjunto [Frakes, Baeza Yates, 1992], para realizar este emparejamiento se utilizan las medidas de semejanza.

Una medida de semejanza se usa para poder evaluar el grado de similitud entre una consulta y los documentos de la base documental, y también entre documentos (en los agrupamientos) porque: "*cada documento debe ser lo mas similar a los documentos de su mismo grupo*". De esta forma es posible definir una medida cuantitativa de la similitud entre dos documentos o entre un documento y una consulta, medida basada en las analogías geométricas.

Las medidas más comunes son:

- Coeficiente de Jaccard:

$$RSV\ (d_i, d_j) = \frac{\sum_{k=1}^{m} t_{ik} \cdot t_{jk}}{\sum_{k=1}^{m} (t_{ik}^2 + t_{jk}^2 - t_{ik} \cdot t_{jk})} \qquad (2.1)$$

- Coeficiente del coseno:

$$RSV\ (d_i, d_j) = \frac{\sum_{k=1}^{m} t_{ik} \cdot t_{jk}}{\sqrt{\sum_{k=1}^{m} t_{ik}^2 \sum_{k=1}^{m} t_{jk}^2}} \qquad (2.2)$$

Ambas medidas definen el concepto de semejanza de consulta y documento (o entre documentos) por la cantidad de términos en común que contienen en relación al tamaño de los documentos.

También otra medida que es muy usada es:

- Distancia euclídea

$$RSV(d_i, d_j) = \sqrt{\sum_{k=1}^{m} (t_{ik} - t_{jk})^2} \qquad (2.3)$$

Al usar la distancia euclidea habitual, documentos y consultas serian tanto más similares cuanto menor fuera el valor del RSV.

2.3. Clasificación de los Sistemas de Recuperación de Información

Existen varios modelos de RI, cada uno tiene sus ventajas e inconvenientes, comentaremos varios de los modelos existentes y analizaremos los componentes que los forman.

2.3.1. El modelo booleano

Este modelo usa la teoría del álgebra de *Boole*. Esta trata con proposiciones, asociadas por medio de operaciones lógicas Y, O, NO, SI ….. ENTONCES, y que, por lo tanto permite cálculos de tipo algebraico. Los componentes principales de este modelo son:

2.3.1.1. Indización de documentos en el modelo booleano

Dentro de un sistema que siga el modelo booleano, los documentos se encuentran representados por conjuntos de palabras clave. La indización se realiza asociando en cada documento, un peso 0 ó 1 para cada término del índice según éste aparezca o no en el documento. 0 si el término no aparece en el documento y 1 si aparece aunque sea una sola vez. Las consultas se formulan mediante expresiones de palabras claves conectadas mediante los operadores lógicos mencionados.

El grado de similitud entre un documento y una consulta será también binario y un documento será relevante cuando su grado de similitud sea igual 1, de lo contrario el documento no tendrá ninguna relevancia en cuanto a la consulta. Por tanto, en el caso de los SRI booleanos, la función de indización sería una F:

$$F: D \times T \rightarrow \{0,1\}$$

2.3.1.2. El subsistema de consultas en el modelo booleano

Las consultas en este modelo se compondrán de expresiones booleanas formadas por elementos del conjunto de términos *T* combinados mediante los operadores lógicos *Y, O y NO*.

Un ejemplo sería: $(T1\ O\ T2)\ Y\ (T3\ Y\ NO\ T5)$

Este ejemplo se ilustra en la figura 2.4.

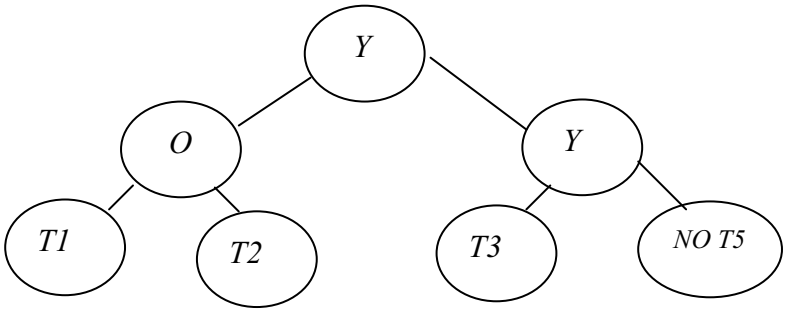

Figura 2.4: Representación de una consulta en forma de árbol

Cuando se ejecuta la consulta, el subsistema de consulta extraerá el *RSV* de cada documento, y decidirá qué conjunto de documentos es el que se considera relevante para dicha consulta. En este modelo, esta operación es muy sencilla ya que no existe graduación de la relevancia (el documento es totalmente relevante a la consulta o no lo es en absoluto). Por tanto los valores del *RSV* serán 0 o 1 y formarán el conjunto de documentos recuperados aquellos que tengan el *RSV* igual a 1.

2.3.1.3. El mecanismo de evaluación en el modelo booleano

El mecanismo de evaluación de este modelo emparejará la consulta *C* con los vectores de los documentos de la base documental para obtener, de este modo el *RSV* de cada uno de ellos. Para facilitar la evaluación de consultas Booleanas, que busca conjuntos de documentos relevantes de la base documental, se suele seguir un método de trabajo consistente en representar las consultas en forma de árbol, donde los términos índices serán las hojas (véase figura 2.4).

Para obtener el conjunto de documentos relevantes, se recorrerá el árbol de abajo a arriba, es decir, de las hojas a la raíz. Para ello nos situamos en una hoja y determinamos el conjunto de documentos relevantes para el término situado en ella, es decir, aquellos que tienen dicho término (o que no tengan en caso de negación).

Posteriormente, se va subiendo en el árbol aplicando la operación lógica correspondiente en cada nodo, resultando un conjunto de documentos (intersección de los conjuntos de abajo para el caso del Y, y unión de los conjuntos de abajo para el caso O), finalmente el conjunto de documentos devuelto por el sistema será el que quede contenido en el nodo raíz.

La ventaja del modelo booleano es que es un modelo muy simple, basado en el formalismo del Álgebra de Boole, lo que le da un marco teórico sólido. Su principal desventaja es la rigidez del criterio de recuperación binario, por lo que corresponde más propiamente a un sistema de recuperación de datos que a uno de información.

2.3.2. El modelo de espacio vectorial

Salton fue el primero en proponer los SRI basados en las estructuras de espacio vectorial a finales de los 60 dentro del marco del proyecto SMART [Salton 1971]. Partiendo de que se pueden representar los documentos como vectores de términos, los documentos podrán situarse en un espacio vectorial de *m* dimensiones, con tantas dimensiones como componentes tenga el vector.

Situado en ese espacio vectorial, cada documento se identifica con un punto determinado por sus coordenadas. Se crean así grupos de documentos que quedan próximos entre sí ó no en virtud de las características de cercanía de sus vectores representantes. Estos grupos están formados, en teoría por grupos de documentos que serían relevantes para la misma clase de necesidades de información.

En una base de datos documental organizada de esta manera, resulta muy rápido calcular la relevancia de un documento a una pregunta (su *RSV*), y es muy rápida también la ordenación por relevancia, ya que, de forma natural, los documentos ya están agrupados por su grado de semejanza. En la fase de la consulta, cuando se formula una pregunta, esta se formula como un elemento del espacio vectorial, y así aquellos documentos que quede más próximos a ella serán, en teoría, los más relevantes para la misma.

La representación de los documentos (y también de las consultas), se realiza mediante la asociación de un vector de pesos booleano (uno por cada término índice). Por ejemplo:

$$di = (t_{i1}, ti_2, t_{i3}, \ldots\ldots\ldots\ldots t_{im}).$$

El hecho de que tanto los documentos como las consultas tengan la misma representación dota al sistema de una gran potencialidad operativa.

2.3.2.1. Indización de documentos en el modelo vectorial

Sea $D = \{ d_1, d_2, \ldots..d_n \}$ el conjunto de documentos y $T = \{ t_1, t_2, \ldots.t_m\}$ el conjunto de términos índice. El mecanismo de indización de este modelo se presentará como una función:

$$F : D \; x \; T \rightarrow I$$

siendo habitual trabajar con una función de evaluación normalizada *F*, donde los vectores tengan como componentes valores reales en [0,1] que ponderan la relevancia del término en el documento. Una de las múltiples formas de definir la función *F* es mediante la frecuencia inversa del documento (IDF) [Salton, McGill, 1983]. Este método está ampliamente difundido como método para establecer los pesos w_{ij} de los términos en el modelo de espacio vectorial.

La idea es que cuando se analiza el texto que constituye un documento, tenemos una distribución de frecuencias de palabras, de modo que las palabras de muy alta frecuencia no suelen ser significativas, y el poder de resolución máximo se da en las palabras de frecuencia media. El poder de resolución de una palabra indica la importancia de esa palabra como término clave.

Si se admite que la importancia de un término en un texto particular es proporcional a su frecuencia de ocurrencia en ese texto, e inversamente proporcional al número total de textos en que aparece ese término, se llega al esquema de asignación de pesos IDF:

Si: n_j el número de documentos en los que aparece el término t_j.

N el número total de documentos de la base de datos.

f_{ij} el número de ocurrencias de t_j en el documento d_i.

Entonces, la representación de cada documento d_i , puede ser expresada por los pesos de cada uno de sus términos tal como: $d_i = (w_{i1}, ..., w_{im})$, *y* podemos elegir un w_{ij} del modo siguiente:

$$W_{ij} = f_{ij} \cdot \log\left(\frac{N}{n_j}\right) = \text{ peso del término } t_j \text{ en el documento } D_i$$

Esta expresión para los pesos, cumple lo que se espera de ella, si un término aparece en muchos documentos, n_j es grande, N/n_j pequeño, y $\log(N/n_j)$ pequeño, luego su peso sería menor que si aparece en pocos documentos, y cuantas más veces aparezca un término en un documento, mayor será el peso de ese término en el documento (debido al factor multiplicante f_{ij} en la expresión de w_{ij}).

Con este método se consigue un sistema para asignar pesos a los términos indexadores en una base de datos:

$$F(d_i, t_j) = f_{ij} . \log \frac{N}{n_j} \qquad (2.4)$$

El interés de la indización IDF estriba en que pondera la importancia de los términos en función de su aparición en el resto de los documentos de la base, además de su frecuencia de aparición en el documento actual.

2.3.2.2. El subsistema de consultas en el modelo vectorial

En este modelo tanto las consultas como los documentos tienen la misma representación, es decir, vectores m-dimensionales, donde m es el número de términos índice considerado. Cada una de los componentes del vector contiene un peso, el cual indica la importancia relativa del término concreto de la consulta o del documento. Este peso es un número real positivo que puede estar o no normalizado.

Cuando un usuario formula una pregunta, la mayoría de los pesos de la misma serán 0, con lo que bastará con proporcionar los términos con peso distinto de 0 para poder definirla. El sistema se encargará de representar la consulta completa en forma de vector m-dimensional de modo automático.

Un ejemplo de consulta para un SRI de Espacio Vectorial (SRI-EV) sería:

$$C = | \underset{t_1}{0} | \underset{t_2}{3} | \underset{t_3}{6} | \ldots\ldots\ldots | \underset{t_m}{1}$$

Una de las diferencias que existen entre este modelo y el booleano es que los términos individuales considerados en la consulta no están conectados por ningún operador (ni conjunción, ni disyunción, ni negación). La ventaja del modelo vectorial es que permite hacer correspondencias parciales es decir, ordena los resultados por grado de relevancia.

2.3.2.3. El mecanismo de evaluación en el modelo vectorial

El mecanismo de evaluación contrasta la consulta *C* con la representación (el vector) asociada a cada documento de la base $di \in D$**.**

Para obtener el grado de relevancia *RSV* del documento con respecto a la consulta *RSVi.* El *RSV* toma un valor real que será tanto mayor cuanto más similares sean documento y consulta.

Existen diferentes funciones para medir la similitud entre documentos y consultas, todas ellas están basadas en la analogía de las representaciones con los puntos del espacio m-*dimensional*, tal como se comentó en el apartado de medidas de semejanzas comentado previamente.

2.3.3. El modelo probabilístico

Este modelo mejora el rendimiento de los SRI por medio del uso de la información procedente de la distribución estadística de los términos en los documentos. La frecuencia de aparición de un término en un documento o conjunto de documentos podría considerarse un dato relevante a la hora de establecer una consulta a la base de datos documental.

El modelo probabilístico utiliza la teoría de la probabilidad para construir la función de búsqueda y para establecer su modo de uso [Fuhr, 1992] [Bookstein,1983].

La información utilizada para componer la función de búsqueda se obtiene del conocimiento de la distribución de los términos de indización a lo largo de la colección de documentos o de un subconjunto de ella.

2.3.3.1. Indización de documentos en el modelo probabilístico

En este modelo tanto las consultas como los documentos tienen la misma representación, es decir, son vectores m-dimensionales, donde m es el número de términos índices considerados. Cada uno de los componentes del vector contiene una probabilidad o peso, que indica la importancia relativa del término concreto de la consulta o del documento. Esta probabilidad es un número real positivo no mayor que 1.

2.3.3.2. El subsistema de consultas en el modelo probabilístico

Realmente, la función de la consulta usa una serie de pesos asociados a los términos de indización. La diferencia entre este modelo y el vectorial está en el modo de calcular el peso de los términos en la consulta.

El principio probabilístico indica que dada una consulta *q*, el modelo probabilístico trata de estimar la probabilidad que un usuario encuentre un documento d_j relevante usando una función del tipo probabilística, tal como: $P(i/D_r)$, que indique la probabilidad que el término i se encuentre en los documentos relevantes recuperados (D_r).

2.3.3.3. El mecanismo de evaluación en el modelo probabilístico

El mecanismo de evaluación del modelo probabilistico contrasta la consulta *C* con la representación asociada a cada documento de la base $di \in D$. Para obtener la probabilidad de relevancia *RSV* del documento con respecto a la consulta *RSVi***,** las medidas de similitud entre documentos y consultas, todas ellas están basadas en tener conocimiento de la distribución de los términos de indización a lo largo de la colección de documentos. La ventaja más clara de este modelo es que introduce el concepto de probabilidades. En cuanto a sus desventajas observamos que:

- No incorpora la noción de correlación entre términos.
- Es difícil determinar la asignación de las probabilidades iniciales; y
- No toma en cuenta las frecuencias de aparición de los términos.

2.3.4. El modelo booleano extendido

Cualquier SRI debe ser capaz de tratar con dos características inherentes al proceso de RI: la imprecisión y la subjetividad. Estos dos factores juegan un papel fundamental en los diferentes estados del procesamiento de la información. Los SRI booleanos no incorporan herramientas adecuadas para manejar las dos características anteriores. Debido a ello, los SRI basados en este modelo de recuperación presentan los siguientes problemas:

- Un término puede aparecer en un documento y ser más significativo en éste que en cualquier otro. No existen mecanismos para representar esta distinción en el modelo booleano.
- El conocimiento vago que el usuario tenga del tema sobre el que está preguntando queda desaprovechado. Si el usuario es un entendido, le gustaría tener la posibilidad de expresar en su consulta la importancia o relevancia que tengan unos términos sobre otros, a través del lenguaje de consulta. La incapacidad de realizar esta matización es una carencia grave del subsistema de consulta de los SRI booleanos.
- Por último, la recuperación será tajante: 1 si el término es relevante y 0 si no lo es. El RSV será 0 ó 1 sin permitir que exista una gradación en la recuperación que de cuenta la imprecisión

Pese a sus carencias, el modelo booleano es uno de los más extendidos en el ámbito comercial. Reconociendo este hecho, se han llevado a cabo varias extensiones sobre el mismo que permitan salvar algunas de las limitaciones. La teoría de conjuntos difusos [Zadeh, 1965] es uno de los instrumentos que se ha empleado como herramienta para tal propósito, especialmente por su capacidad para tratar la

imprecisión y la incertidumbre en el proceso de RI. Este hecho se debe fundamentalmente a dos razones principales:

- Es un marco formal diseñado para poder gestionar la imprecisión y la vaguedad.
- Facilita la definición de una prolongación del modelo booleano, de forma que los SRI basados en este último pueden modificarse sin tener que ser completamente rediseñados.

2.3.4.1. Indización en el modelo booleano extendido

La indización de los términos se llevará a cabo de modo análogo al modelo de espacio vectorial, pero permitiendo que un documento tenga asociado un peso para cada término, peso que indica el grado en que el documento se caracteriza por tal término.

Los pesos toman valor en el rango [0,1]. Se basará por tanto en una indización difusa donde una función de pertenencia *F* mostrará el grado en el que el conjunto de términos representa al documento. Dentro del marco difuso, los documentos se representarán como subconjuntos difusos del conjunto de términos índice, en los cuales el grado de pertenencia que liga un término a un documento expresa si el término describe el contenido del documento de manera significativa. El criterio que calcula el valor del grado de pertenencia será el número de apariciones del término dentro del documento.

Esto se podría interpretar como una función de pertenencia de un conjunto bidimensional [Klir, Yuan, 1995], [Zimmermann, 1996] que muestra el grado en el que cada documento *d* se relaciona a ese grupo de documentos que pertenecen al/los concepto/s representado/s por cada término *t.*

De esta forma, se podría asociar un conjunto difuso a cada documento y término como:

$$d_i = \left\{ \langle\, t,\, \mu_{di}(t) \,\rangle \;\middle/\; t \in T \right\}; \quad \mu_{di}(t) = F(d_i, t) \qquad (2.5)$$

$$t_j = \left\{ \langle\, d,\, \mu_{tj}(t) \,\rangle \;\middle/\; d \in D \right\}; \quad \mu_{tj}(d) = F(d, t_j)$$

2.3.4.2. El subsistema de consultas en el modelo booleano extendido

El *RSV* de los documentos será un valor perteneciente al intervalo [0,1]. Dicho valor mide en grado la relevancia parcial de cada documento y se ordenan los resultados en función a su valor. El conjunto final de documentos recuperados puede venir determinado por dos vías: bien proporcionando un umbral superior para el número de documentos recuperados lo que determina el umbral α resultante, ó bien definiendo un umbral inferior *α* para el grado de relevancia lo que limitará el número de documentos recuperados. Tomándolo de este segundo modo, el conjunto final de documentos recuperados sería:

$$R = \left\{ d \in D \;\middle/\; RSVq\,(d) \geq \alpha \right\} \qquad (2.6)$$

Como ya hemos comentado, una de las ventajas de aplicar estas extensiones a los SRI booleanos es que los documentos podrán ser ordenados según el grado de pertenencia. El usuario tendrá la ventaja de poder limitar el número de documentos en el conjunto recuperado especificando una cota superior para el número de documentos recuperados. Además podría limitar el tamaño de la recuperación devolviendo únicamente aquellos documentos que mejor satisfacen la consulta *q.*

2.3.4.3. El mecanismo de evaluación del modelo booleano extendido

La diferencia principal entre el subsistema de consulta propio del modelo booleano y el del modelo booleano extendido es la aparición de pesos y el hecho de que el resultado de la consulta sea un conjunto difuso definido sobre el espacio de representación de los documentos. Este modelo acarrea el consiguiente problema de la selección de documentos según la interpretación de los pesos.

El proceso de evaluación de la consulta se realiza desde abajo hacia arriba, empezando por los términos simples de la consulta. El primer paso consiste en combinar cada término individual con su peso asociado, obteniendo el *RSV* de cada documento para la consulta compuesta por ese único término y su peso. Esta operación se realiza mediante un operador que se denominará E *E(d, <t,w>),* cuya definición depende de la interpretación asociada a los pesos. Luego, se pasa a calcular el valor de la recuperación final como resultado de las combinaciones booleanas de las *E (d, <t,w>)* parciales.

Varios autores han reconocido que las semánticas de los pesos en la consulta deberían estar relacionadas con el concepto de "*importancia*" del término. Se han propuesto diferentes semánticas para los grados de pertenencia asociados con el término *t* en la definición de la consulta, tales como:

- La importancia relativa de *t,* de acuerdo con la semántica usada, que permite al usuario expresar la importancia de cada término en la consulta [Bookstein, 1980].
- El umbral para *t,* que considera los pesos como umbrales, premiando al documento cuyo grado de pertenencia para el término *t* sea mayor o igual que el grado de pertenencia del término en la consulta, pero permitiendo algún valor de coincidencia parcial cuando el grado de pertenencia del documento es menor que el umbral [Buell, Kraft, 1981].

2.4. Evaluación de los Sistemas de Recuperación de Información

Un SRI puede evaluarse empleando diversos criterios. Así, Frakes [Frakes, Baeza Yates, 1992] selecciona los tres siguientes como los más importantes:

- Ejecución eficiente (eficiencia).
- Almacenamiento correcto.
- Recuperación efectiva (efectividad).

La eficiencia en la ejecución se medirá por el tiempo que toma el sistema o una parte del mismo para llevar a cabo una operación. Este parámetro ha sido siempre una preocupación principal ante un SRI, especialmente desde que muchos de ellos son interactivos y un tiempo de recuperación excesivo disminuye su utilidad.

La eficacia del almacenamiento se medirá por el número de *bytes* que se precisan para almacenar los datos.

Tradicionalmente, se le ha dado mucha importancia a la efectividad de la recuperación, normalmente basada en la relevancia de los documentos recuperados a las necesidades reales de información del usuario, lo cual ha representado un problema ya que medir la relevancia es un proceso subjetivo, es decir, diferentes juicios personales asignarían diferentes valores de relevancia a un documento recuperado en respuesta a una búsqueda. Salton y McGill [Salton, McGill,1983] señalan que, además de los criterios anteriores que se centran principalmente en el punto de vista del diseñador del sistema, se debe considerar también el punto de vista del usuario cuyos criterios de evaluación no tienen porqué coincidir con los del primero.

Los seis criterios siguientes han sido identificados como los más importantes en lo que respecta a las características que un SRI debe ofrecer al usuario [Cleverdon, 1972]:

1. La exhaustividad o capacidad del sistema para presentar todos los documentos relevantes.

2. La precisión, o capacidad del sistema para presentar solamente documentos relevantes.

3. El esfuerzo, intelectual o físico, requerido por el usuario en la formulación de las consultas, en el manejo de la búsqueda y en el proceso de revisión de los resultados.

4. El intervalo de tiempo transcurrido entre que el sistema recibe la consulta del usuario y presenta las respuestas.

5. La forma de presentación de los resultados de la búsqueda, la cual influye en la utilidad para el usuario de la información recuperada.

6. La proporción en la que están incluidos en la recuperación todos los documentos relevantes del sistema ya conocidos por el usuario.

Teniendo en cuenta los distintos factores que se pueden considerar en el proceso de evaluación de un SRI, es importante destacar el hecho de que el propio concepto de evaluación puede verse desde dos perspectivas distintas. Por ello, existen dos grandes corrientes de investigación en evaluación de la RI, denominadas respectivamente la corriente algorítmica, basada en el modelo tradicional de evaluación y la corriente cognitiva. La primera tendencia (clásica) se centra en los algoritmos y en las estructuras de datos necesarias para optimizar la eficacia y la eficiencia de las búsquedas en base documentales. La segunda considera el papel del usuario y de las fuentes de conocimiento implicadas en la RI [Ingwersen, Willet, 1995].

La base del modelo cognitivo la constituyen los trabajos de Dervin y Nilan [Dervin, Nilan, 1986] que buscaban proponer una alternativa al modelo clásico de evaluación. Este modelo ha provocado un interés creciente por incorporar a los usuarios en el proceso de evaluación. Desde esta perspectiva, la evaluación se centra en la representación de los problemas de información, el comportamiento en las búsquedas y los componentes humanos de los SRI en situaciones reales, y se fundamenta en la psicología cognitiva y en las ciencias sociales. La búsqueda de información y la formulación de la necesidad de información se contemplan como procesos cognitivos del usuario, siendo el SRI y los intermediarios funcionales (como la interfaz del sistema) componentes fundamentales de este proceso.

El modelo cognitivo asume que la complejidad, puesta de manifiesta en la investigación orientada solamente a técnicas algorítmicas de RI, no es suficiente para proporcionar una panorámica completa del proceso de recuperación. Para lograrla, es necesario tener en cuenta las características del sistema, las situacionales del usuario y los elementos intermedios, el más importante de los cuales es la interfaz de usuario, mecanismo principal de enlace entre este último y el sistema [Ingwersen, Willet,1995]. En este marco se han propuesto distintas medidas de evaluación del SRI relacionadas con el concepto de relevancia basada en el usuario. Entre ellas se encuentran la proporción de cobertura o alcance ("*coverage ratio*"), definida como la fracción de documentos relevantes conocidos por el usuario que han sido recuperados, y la proporción de novedad ("*novelty ratio*"), que se define como la fracción de documentos relevantes recuperados que son desconocidos por el usuario [Korfhage,1997]. También se ha considerado la satisfacción del usuario como medida de la eficacia del SRI, aunque no se han logrado aún una forma adecuada para calcularla al tratarse de un criterio muy subjetivo.

Precisamente, esta última afirmación constituye la critica más relevante del modelo cognitivo, que al usar también otras medidas complementarias como beneficio, frustración, utilidad, etc, medidas que no son objetivas y que no evalúan directamente el sistema sino el efecto que este provoca en el usuario.

Por otro lado, en el modelo algorítmico se han propuesto múltiples medidas de efectividad de la RI, siendo las más empleadas y conocidas las *exhaustividad* y la *precisión* [Van Rijsbergen, 1979].

La exhaustividad es la proporción de documentos relevantes recuperados en una búsqueda determinada sobre el número de documentos relevantes para esa búsqueda en la base de datos, siendo su formula:

$$E = \frac{número \quad de \quad documentos \quad relevantes \quad recuperados}{número \quad de \quad documentos \quad relevantes} \qquad (2.7)$$

La precisión es la proporción de documentos relevantes recuperados sobre el número total de documentos recuperados, siendo su formula:

$$P = \frac{número \quad de \quad documentos \quad relevantes \quad recuperados}{número \quad de \quad documentos \quad recuperados} \qquad (2.8)$$

Para comparar la efectividad del SRI en los términos de exhaustividad y precisión, se han desarrollado métodos para evaluarlos de forma simultánea. De este modo, es habitual trabajar con un sistema de coordenadas en el que un eje es para exhaustividad y otro para la precisión. (figura 2.5).

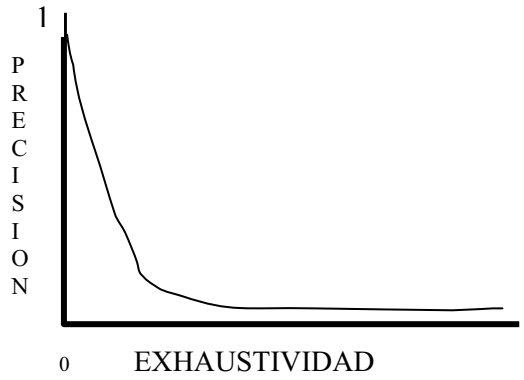

Figura 2.5: Precisión y Exhaustividad

Los puntos de exhaustividad-precisión están inversamente relacionados [Cleverdon,1972]; es decir, cuando la precisión sube, la exhaustividad normalmente baja y viceversa. Esto se debe a que, mientras que la precisión da importancia a la ausencia o "*no recuperación*" de documentos no relevantes, la exhaustividad se basa fundamentalmente en la recuperación de todos los documentos relevantes, aunque esto implique recuperación de documentos no relevantes.

Por tanto, el fin de la precisión es la ausencia de lo que en el ámbito documental llamamos "*ruido*", mientras que en la exhaustividad lo que se intenta evitar es el "*silencio*".

2.5. Alternativas para lograr una mejor recuperación: El agrupamiento

El objetivo de las técnicas de recuperación de información es resolver consultas de forma eficiente y eficaz. La eficiencia viene dada por la rapidez con la cual se resuelve la consulta, y la eficacia se evalúa por la relevancia de los resultados encontrados [Raghavan, Bollman, Jung, 1989].

Por muy completo que sea un SRI, por regla general frecuentemente presentan una carencia destacable en lo concerniente a la exhaustividad [Harman, 1986]. Los usuarios pueden generalmente recuperar algunos documentos relevantes como respuesta a sus consultas, pero casi nunca recuperan todos los documentos relevantes relacionados con las mismas. Existen casos en que esto no tiene mucha importancia para los usuarios, pero en aquellas ocasiones en las que la exhaustividad sea un parámetro crítico, debemos de poder recuperar la mayor cantidad de documentos relevantes.

En el contexto de recuperación de información, Van Rijsbergen [Van Rijsbergen, 1979] formuló la denominada "Hipótesis del Agrupamiento" ("*Cluster Hipótesis*"), sosteniendo que "*los documentos que estén fuertemente asociados*" resultarán relevantes para las mismas consultas. Aquí se supone que la agrupación automática se realiza teniendo en cuenta el contenido de los documentos, en el sentido de que documentos similares tendrán palabras (ó términos) similares.

Por ello, a la hora de resolver éste problema, importante, conviene que el propio SRI ofrezca al usuario la posibilidad de agrupar los elementos de la base documental antes o después de su consulta.

La agrupación automática de documentos se comenzó a investigar muy pronto dentro de la rama de la Recuperación de Información ("*Information Retrieval*"). En su forma más simple, las consultas están dirigidas a determinar qué documentos poseen determinadas palabras en su contenido. Por ejemplo, la consulta ***"buscador internet"*** podría tener por resultado todos los documentos que contengan las palabras *"buscador" e "internet"* .

Debido a que, el objetivo de las técnicas de recuperación de información es poder resolver consultas en forma eficaz y eficiente, en el ejemplo del párrafo anterior, una consulta eficaz debería retornar todos los documentos que trataran sobre la búsqueda de información en internet, aún si la palabra *"buscador"* no estuviera en el contenido de los mismos.

2.6. Agrupación automática de documentos

La tarea de encontrar grupos de documentos con características comunes no sólo es compleja, sino que además consume tiempo. Un bibliotecario que tratara de clasificar un documento tendría que leerlo para comprender su significado, para luego asignarle una categoría, utilizando su experiencia y sentido común. El costo y el que supone la agrupación de documentos ha motivado a la investigación de técnicas que permitan automatizar dicha tarea [Rüger, Gauch, 2000]. Debido al incremento en los volúmenes de información disponibles en forma electrónica, y a la necesidad cada vez mayor de encontrar la información buscada en un tiempo mínimo, éstas técnicas han estado recibiendo creciente atención [Zamir, Oren, Etzioni, 1999],[Strehl et al, 2000],[Maarek et al, 2000].

2.6.1. Aplicaciones del agrupamiento

La agrupación automática de documentos se investiga dentro del campo de recuperación de información por las razones antes expuestas.

Sus beneficios más inmediatos son:

- Mejorar el rendimiento de los motores de búsqueda de información mediante la agrupación previa de todos los documentos disponibles [Faloutsos et al, 1996],[Zamir, Oren, Etzioni, 1999],[Rüger, Gauch, 2000].

  Antes de comenzar a resolver las consultas, el conjunto de documentos es separado en grupos. Luego, en el procesado de una consulta, no se examinan todos los documentos, sino que se busca en el grupo *representante* que mejor responda a la consulta. Por la hipótesis del agrupamiento, el grupo encontrado estará compuesto de los documentos más relevantes para esa búsqueda.

- Facilitar la revisión de resultados por parte del usuario final; agrupando los resultados luego de realizar la búsqueda [Cutting et al, 1992],[Allen et al,1993],[Leousky, Croft, 1996],[Maarek et al, 2000].

  Cuando se realizan búsquedas sobre un gran volumen de documentos, la cantidad de resultados puede ser muy grande; si la lista se presenta sin ningún tipo de procesamiento previo, el usuario del sistema se verá obligado a revisar todos los documentos descartando aquellos que no sean relevantes para él. Si los resultados se agrupan, la hipótesis del agrupamiento indica que los documentos del mismo grupo serán relevantes para una subconsulta más específica que la original. De esta forma, los documentos quedarán separados en grupos temáticos. Así, con sólo revisar uno o dos documentos de cada grupo, el usuario podrá determinar cuál es el subtema al que responden todos los documentos del grupo, pudiendo descartar rápidamente los documentos irrelevantes para su búsqueda.

2.6.2. Naturaleza combinatoria del problema de agrupación

Una de las características principales del problema de la agrupación de documentos es su naturaleza combinatoria [Pentakalos, Menascé, Yesha, 1996]. [Liu, 1968] indica que la teoría combinatoria establece que la cantidad de maneras de agrupar *n* objetos en *K* grupos, está dada por:

$$S(n,k) = \frac{1}{K!}\sum_{i=0}^{k}(-1)^{i}\binom{k}{i}(k-i)^{n} \qquad (2.9)$$

En la práctica, se querría aplicar la agrupación automática, por lo menos, a colecciones con cientos de documentos. Supóngase un algoritmo que intente realizar todas las posibles agrupaciones para encontrar la mejor de ellas, este algoritmo tendría que ser muy rápido, si no se quiere que la respuesta demore demasiado, dado el volumen de posibilidades.

Por lo tanto, un algoritmo que evalúe cada una de las posibles agrupaciones no es aplicable en forma práctica a la agrupación automática de documentos, por lo que se han desarrollado algoritmos que hacen uso de heurísticas [Cutting et al, 1992], [Zamir, Oren, Etzioni, 1999], [Jain,Murty, Flinn, 1999], [Zhao, Karypis, 2001] para explorar una parte de estas posibles agrupaciones buscando una solución de calidad aceptable en función de diversos criterios, que darán lugar a diferentes técnicas.

2.7. Estado de arte de la agrupación de documentos

La agrupación automática de documentos se vincula con las disciplinas de la *Recuperación de Información* y de la *Minería de Datos*.

Del campo de la Recuperación de Información toma los conceptos y métodos utilizados para el procesamiento de documentos. Estas técnicas son las que permiten transformar la información no estructurada que contienen los documentos (de texto libre, ó *free text documents*) en estructuras de datos manejables mediante algoritmos computacionales.

Del campo de la Minería de Datos toma las técnicas que se ocupan de los problemas de agrupación de objetos. Los algoritmos utilizados para la agrupación automática de documentos son adaptaciones de los que se utilizan para el problema más general de agrupar objetos de cualquier tipo, detallaremos los algoritmos utilizados actualmente para agrupar en forma automática y mostraremos la ubicación de la agrupación automática de documentos dentro de esta área.

2.7.1. Agrupación de objetos

La agrupación de documentos es un tipo de problema perteneciente a la familia de problemas asociados a encontrar agrupamientos entre objetos de cualquier tipo. Si bien la agrupación de documentos tiene características particulares que surgen de las propiedades de éstos como objetos a agrupar, los principios generales coinciden con los que se aplican para agrupar cualquier otro tipo de elementos. Los algoritmos para la agrupación automática de documentos son los mismos que se utilizan para agrupar otros tipos de objetos, o adaptaciones de éstos [Qin He, 1996],[Fasulo, 1999], [Jain, Murty, Flinn, 1999].

Una definición del problema del agrupamiento de documentos, aplicable al agrupamiento de cualquier tipo de elementos, puede enunciarse [Zhao, Karypis, 2001] de la siguiente manera: Dado un conjunto *S* de *N* documentos, se quiere encontrar una partición $S_1, S_2, \ldots, S_k$, tal que cada uno de los *N* documentos se encuentre solamente en un grupo *Si*, y que cada documento sea más similar a los documentos de su mismo grupo que a los documentos asignados a los otros grupos.

2.7.2. Tipos de agrupación

Las formas de agrupación de objetos, tales como asignar clases predeterminadas a cada elemento ó agruparlos en forma significativa, son susceptibles de dividirse según el esquema de la figura 2.6 [Qin He, 1996]:

- No exclusivas: Un mismo objeto puede pertenecer a varias categorías o grupos.
- Exclusivas: Cada objeto pertenece solamente a una categoría, clase o grupo.

    a) Extrínsecas (supervisadas): Las clases a las que pertenecen los objetos están predefinidas, y se conocen ejemplos de cada una, ó algunos de los objetos ya están agrupados y son utilizados por el algoritmo para aprender a clasificar a los demás.

b) Intrínsecas (no supervisadas): La agrupación se realiza en base a las características propias de los objetos, sin conocimiento previo sobre las clases a las que pertenecerán.

   i. Jerárquicas: Los métodos jerárquicos consiguen la agrupación final mediante la separación (métodos divisivos) o la unión (métodos aglomerativos) de grupos de documentos. Así, estos métodos generan una estructura en forma de árbol en la que cada nivel representa una posible agrupación de los documentos.

   ii. Particionales (No jerárquicas): Los métodos particionales, o de optimización llegan a una única agrupación que optimiza un criterio predefinido o función objetivo, sin producir una serie de anidaciones de los grupos.

La agrupación automática de documentos se encuentra en la categoría *intrínseca*, ya que los criterios de agrupamiento se basan en la información contenida en los mismos que será utilizada para determinar sus similitudes.

En la figura 2.6 mostramos una clasificación de las técnicas existentes para agrupar cualquier tipo de objeto:

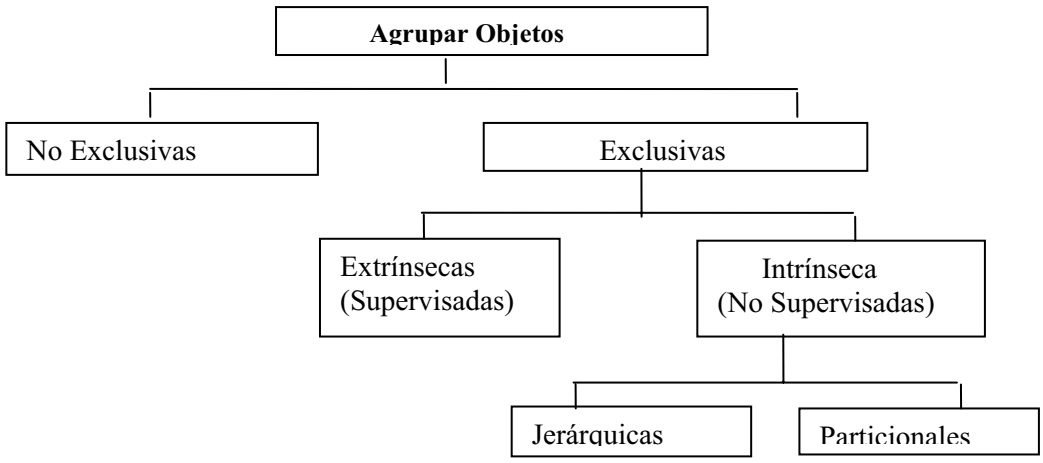

Figura 2.6: Formas de agrupar cualquier tipo de objeto

A continuación desarrollaremos formalmente los métodos tradicionales de agrupamiento del tipo exclusivo; para luego más adelante poder comparar nuestra propuesta con algunos de estos métodos clásicos.

2.7.3. Técnicas de agrupamiento exclusivas

Una *técnica de agrupamiento* exclusiva se puede definir como una técnica diseñada para realizar una agrupación asignando elementos a *grupos* de tal forma que cada grupo resulte más o menos homogéneo y distinto de los demás. Se suele emplear indistintamente los términos: grupo, agrupamiento, agrupación o el término inglés cluster, y también se suele hablar de *clustering* como sinonimo de técnica de agrupamiento. Intuitivamente, un conjunto de elementos se consideran un *grupo* si forman un conjunto homogéneo y diferenciado en algún sentido constatable.

El criterio de homogeneidad más simple está basado en la *distancia* (entendida como tal gracias a la representación vectorial de los documentos), se espera que la distancia entre los elementos de un mismo agrupamiento sea *significativamente* menor que la distancia entre elementos de agrupamientos diferentes. En definitiva, una técnica de agrupamiento debe encontrar y caracterizar, en primer lugar, los agrupamientos, para posteriormente etiquetar los patrones en base a éstos. Podemos esquematizar funcionalmente un método de agrupamiento como se indica en la figura 2.7 que a partir de un conjunto de *M* elementos no etiquetados: $\{X_i, i = 1, 2, .., M\}$ encuentra *K* agrupamientos $S_j, j = 1, 2, ..., K$.

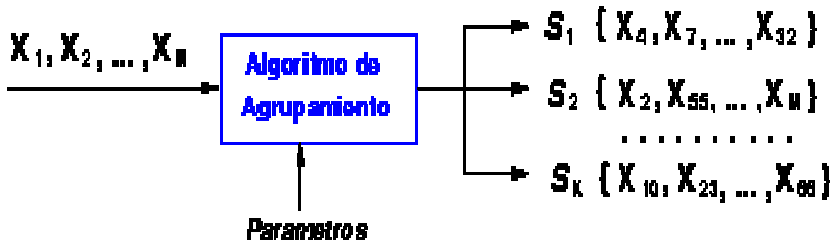

Figura 2.7: Esquema funcional de un algoritmo de agrupamiento exclusivo

Algunos algoritmos de agrupación requieren conocer previamente el número de agrupamientos a realizar (o un máximo) y en general, todos requieren unos parámetros específicos del algoritmo. Para hacer un agrupamiento se considera un conjunto P de patrones o criterios, cada uno de los cuales es característico de los agrupamientos, es decir:

Sea *M* el número de elementos $X_1, X_2, ..., X_M$ de *P*. Un *proceso de agrupamiento* consiste en buscar *K* grupos (subconjuntos de P) $S_1, S_2, ..., S_K$ tales que todo $X_i, i = 1, 2, ..., M$ pertenece a uno y solo uno de estos grupos, ósea:

$$S_1 \cup S_2 \cup \cdots \cup S_K = \{X_1, X_2, ..., X_M\} = P \qquad (2.10)$$
$$S_i \cap S_j = 0 \quad \forall i \neq j$$

es decir, $\{ S_1, S_2, \ldots\ldots S_k \}$ constituye una partición de P.

2.8. Tipos de agrupamiento exclusivos

Para la agrupación se dispone de dos posibles técnicas diferentes, una Supervisada (CS) y la otra la llamada No Supervisada (CNS), hay bastantes diferencias entre ambas técnicas, las dos más importantes son: el objetivo de la agrupación y la información de la que se dispone para llevarla a cabo.

Dado un conjunto de objetos, el objetivo de la CNS consiste en intentar establecer la estructura de subconjuntos subyacente para dicho conjunto; esto puede querer realizarse [Mirkin, 1996] por varios motivos: simplificación de un conjunto de datos, análisis de los mismos, predicción, extracción de conocimiento, etc. Por el contrario, la CS utilizando los objetos del conjunto pretende construir un sistema que asigne los objetos actuales del conjunto y objetos venideros a clases cuyo número viene prefijado de antemano.

En cuanto a la información que posee cada tipo de agrupación, la CNS únicamente dispone de los datos. Por el contrario la CS dispone del número de grupos y también de un ejemplo (ó muestras) para cada clase.

Nos centráremos en la CNS en la que los objetos vienen descritos por valores numéricos. En particular presentaremos los métodos que producen particiones, esto es: la clasificación particional y la clasificación jerárquica.

2.9. Agrupación particional

La CNS particional es la más habitual y la más extendida. Esta técnica trata de descubrir la posible estructura de clases en que pueda descomponerse un conjunto de datos. Es decir, dado un conjunto de datos Ω, se trata de dividir dicho conjunto en grupos $\{C_1, C_2, \ldots C_k\}$ de manera que cada grupo posea objetos que son parecidos entre sí, a la vez que diferentes de los objetos de los demás grupos en algún sentido significativo.

Formalmente casi todos los métodos de agrupación particional se basan en optimizar una función matemática *F.*

$$F: P_k(\Omega) \rightarrow R$$

Siendo en este caso $P_k(\Omega)$ el conjunto de las particiones del conjunto Ω en k subconjuntos. A dicha función F la llamaremos criterio de agrupación.

El hallar las clases subyacentes al conjunto no suele ser una tarea trivial y envuelve principalmente dos problemas:

- Hallar el número de clases que existen en el conjunto.
- Dado el número de clases anterior, repartir los objetos del conjunto en cada clase de forma que se optimice la función *F*.

Los dos problemas han sido resueltos principalmente mediante dos tipos de estrategias diferentes. La primera de las estrategias consiste en utilizar en una fase inicial un algoritmo que halle el número de grupos en el conjunto, para en una fase posterior utilizar otro algoritmo que dado el número de grupos asigne cada objeto a la clase más adecuada. La segunda estrategia consiste en utilizar un algoritmo que realice ambas cosas al mismo tiempo.

La mayoría de los métodos particionales trabajan en un sólo nivel, en el que se optimiza un agrupamiento. Estos métodos asumen que el valor de *k* (la cantidad de grupos) está definida de antemano [Qin He, 1996].

La estructura general de estos métodos se compone de los siguientes pasos [Han, Lamber, Tung, 2001]:

1) Seleccionar *k* puntos representantes (cada punto representa un grupo de la solución).

2) Asignar cada elemento al grupo del representante más cercano.

3) Actualizar los *k* puntos representantes de acuerdo a la composición de cada grupo.

4) Volver al punto 2)

Este ciclo se repite hasta que no sea posible mejorar el criterio de optimización. A continuación comentaremos las características de los principales algoritmos de este tipo.

Algoritmos para hallar los grupos dado el número de grupos

Comentaremos los tipos de algoritmos más utilizados para dividir un conjunto de datos en grupos siendo el número de éstos conocido de antemano. Casi todos los algoritmos mostrados utilizan el mismo criterio, suponen que cada objeto del conjunto está definido por *p* variables numéricas, de esta forma es posible considerar los objetos del conjunto como vectores en un espacio R^p.

El criterio utilizado por estos algoritmos consiste en determinar una función que dada una partición $\{C_1, C_2, \ldots C_k\}$, halla para cada grupo *Ci*, el punto medio de los objetos que forman dicho grupo (centro del cluster):

$$\bar{w}_j = \frac{1}{n_i} \sum_{i=1}^{n_i} x_j^i \quad j = 1, 2, \ldots .k \qquad (2.11)$$

Donde n_i es el número de objetos que se encuentran en el grupo C_i. A dichos puntos les llamaremos *centroides*. El criterio de agrupación (denotado por B) usa las sumas al cuadrado de las distancias entre los elementos de cada grupo y su centroide:

El concepto de centroide de un grupo de elementos que están representados vectorialmente es comúnmente utilizado en el contexto de los algoritmos de agrupamiento. Dado un grupo de vectores *S*, que contiene *h* elementos s_i, se define a su centroide C_s como el promedio de los vectores que componen el grupo:

$$C_s = \frac{\sum_{t=1}^{h} s_t}{h}, \qquad (2.12)$$

Cada componente del vector centroide es el promedio del valor de ese componente para los miembros del grupo [Steinbach, Karypis, Kumar, 2000],[Zhao, Karypis,2001].

El criterio de agrupación aquí vendrá denotado por B, siendo:

$$B(\{C_1, C_2, \ldots\ldots, C_k\}) = \sum_{j=1}^{k} \sum_{i=1}^{n_i} \left\| x_i^j - \bar{wj} \right\|^2 \qquad (2.13)$$

donde || || es la norma Euclidea.

Con este criterio se implementan el algoritmo llamado *Kmeans*, que comentaremos a continuación, que trata de minimizar la función *B*, y es del tipo particional, propuesto por [MacQueen, 1967], que dada una partición inicial en *k* clases, realizan cambios de objetos de un grupo a otro en caso de que dicho cambio reduzca el valor de la función *B*.

Un pseudocódigo para el *K-means* es el siguiente:

1. Dar una partición inicial
2. Siguiendo el orden de los datos realizar:
    a. Asignar el objeto *w* a la clase *C* que más reduzca el valor de la función criterio *B*.
    b. Modificar el centroide de la clase *C*.
3. Si algún objeto ha cambiado de clase ir al paso 2. En caso contrario terminar dando la clasificación obtenida.

Un aspecto clave en este algoritmo, es el de seleccionar bien los valores iniciales.

El método más frecuentemente utilizado para obtener los *k* puntos representantes iniciales es simplemente tomarlos al azar [Bradley, Fayyad, 1998]. Esta técnica es la más rápida y simple, pero también la más riesgosa, ya que los puntos elegidos pueden ser una mala representación de la colección de objetos. Cuando se utiliza esta técnica se debe ejecutar varias veces el algoritmo de agrupamiento para distintas selecciones aleatorias, tomando el mejor resultado y descartando el resto [Steinbach, Karypis, Kumar, 2000].

2.10. Agrupación jerárquica

La agrupación no supervisada jerárquica es otra técnica de CNS. Esta técnica, en vez de crear una única partición en el conjunto de datos, crea una sucesión encajada de particiones cuya estructura puede ser representada por medio de un árbol.

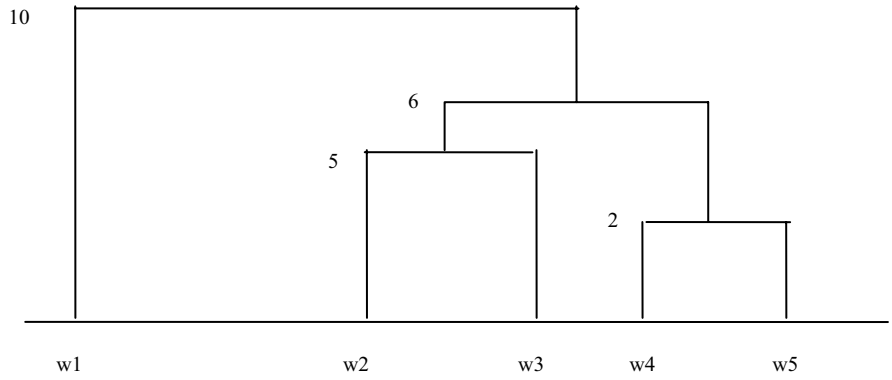

Figura 2.8: Ejemplo de una agrupación jerárquica

En la figura 2.8 se clasifica un conjunto con cinco objetos. Las hojas del árbol representan clases conteniendo los objetos a clasificar, cada nodo interno representa una clase que contiene a todos los objetos que se encuentran en la hoja de dicho nodo. La raíz representa a la clase que contiene a todos los objetos del conjunto. Al lado de cada nodo se encuentra un número que representa el *"grado"* al que se han juntado dichas clases y será llamado altura.

Es posible cortar el árbol de forma que se obtenga una clasificación para cada número *K (K=1,2,….n)* de clases diferentes. Así si cortásemos el árbol a un tercer nivel, como en el gráfico anterior, obtendríamos las siguientes clases (ver figura 2.9):

$$C_1 = \{w_1\},\ C_2 = \{w_2, w_3\},\ C_3 = \{w_4, w_5\}$$

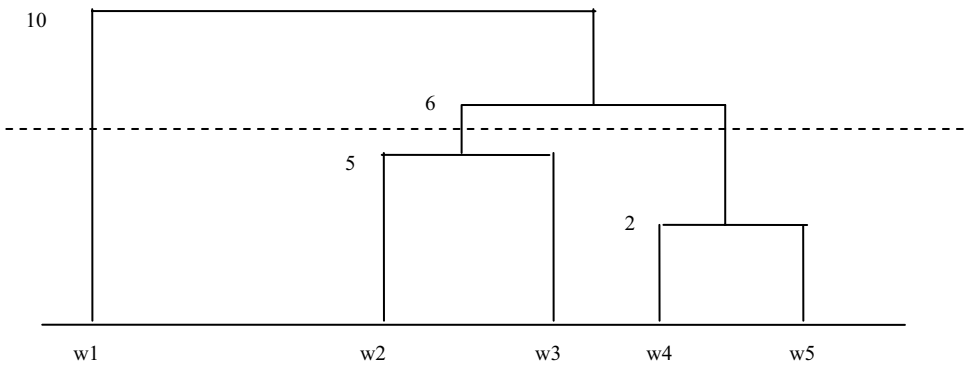

Figura 2.9: Corte en el árbol para obtener un número de clases

Los algoritmos jerárquicos se caracterizan por generar una estructura de árbol (llamada "*dendograma*"). Cada nivel es un agrupamiento posible de los objetos de la colección [Jain, Murty, Flinn, 1999], cada *nodo* del árbol es un grupo de elementos. La raíz del árbol se compone de un sólo grupo que contiene todos los elementos. Cada hoja del árbol es un grupo compuesto por un sólo elemento.

En los niveles intermedios, cada nodo del nivel n es dividido para formar sus hijos del nivel *n + 1*. Las figuras 2.10 y 2.11 ilustran estos conceptos mediante un ejemplo:

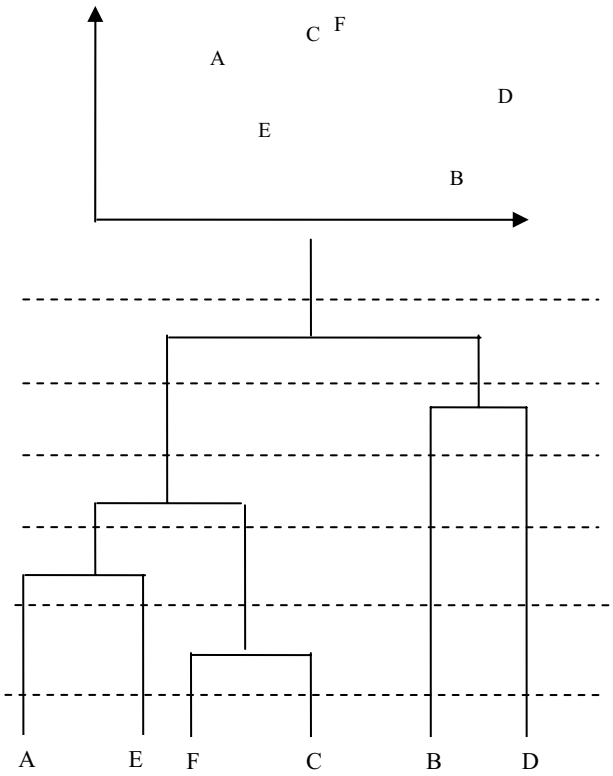

Figura 2.10: Posible colección de objetos en un espacio de 2 dimensiones

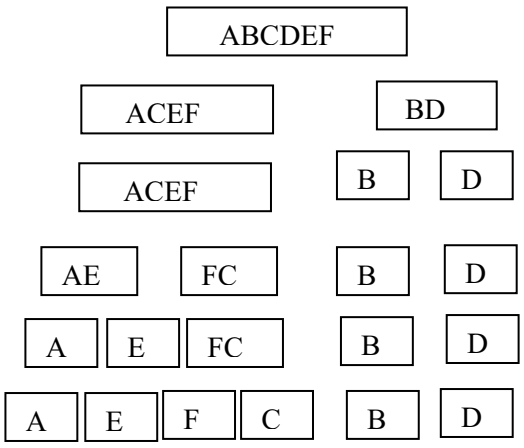

Figura 2.11: Dendograma y agrupamientos en cada nivel para la colección de objetos analizada

Los algoritmos de agrupamiento jerárquicos fueron uno de los primeros enfoques usados para los problemas de agrupación de documentos, y todavía se siguen utilizando debido a la forma simple e intuitiva en la que trabajan [Dash, Liu, 2001]. De acuerdo a la metodología que aplican para obtener el dendograma, los algoritmos jerárquicos pueden dividirse en aglomerativos ó divisivos [Han, Lamber, Tung 2001].

Los métodos aglomerativos parten de las hojas del árbol, ubicando a cada elemento en su propio grupo, y en cada paso buscan los pares de grupos más cercanos para juntarlos. Los divisivos, por su parte, hacen el camino inverso. Comenzando en la raíz, en cada paso seleccionan un grupo para dividirlo en dos, buscando que el agrupamiento resultante sea el mejor de acuerdo a un criterio predeterminado. El análisis necesario para pasar de un nivel a otro (decidir qué grupo dividir o cuales juntar) es más sencillo para los métodos aglomerativos [Dash, Liu, 2001], y esto hace que éstos sean más utilizados que los divisivos [Fasulo, 1999]. A continuación comentamos ambos algoritmos, dando más énfasis a los algoritmos aglomerativos.

Las distintas variantes de algoritmos jerárquicos aglomerativos difieren únicamente en la manera de determinar la semejanza entre los grupos al seleccionar los dos grupos más cercanos [Qin He, 1996],[Jain, Murty, Flinn, 1999],[Fasulo 1999]. En nuestro caso debe notarse la diferencia entre medidas de semejanza entre documentos y medidas de semejanza entre grupos de documentos. La similitud de dos grupos se calcula en base a los valores de semejanza existentes entre sus documentos, pero hay diversas posibilidades de entender la semejanza. Los distintos algoritmos jerárquicos aglomerativos se distinguen por la medida de semejanza entre grupos que utiliza cada uno. Así, por ejemplo en la figura 2.12 se presenta un espacio bidimensional en el cual se han colocado 4 grupos de objetos, los objetos de un mismo grupo se han representado mediante el mismo símbolo (x, *, # , +).

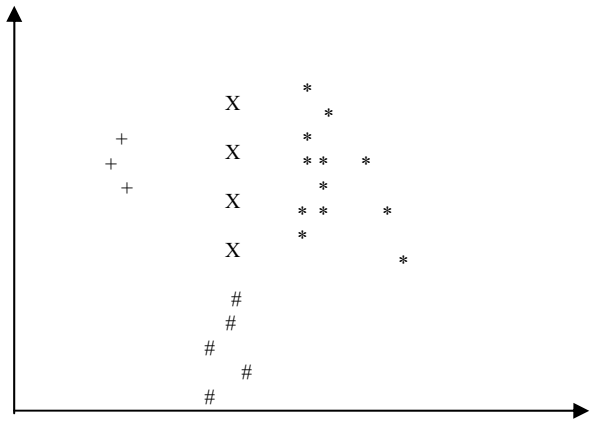

Figura 2.12: Cuatro grupos de objetos en un espacio de 2 dimensiones

Las figuras 2.13, 2.14 y 2.15 muestran cómo los métodos pueden diferir entre ellos al seleccionar cuáles son los grupos más semejantes. En las figuras se utiliza la distancia euclideana como medida de semejanza entre los documentos; la disposición presentada de los grupos se ha elegido especialmente para provocar las diferencias, si los grupos están suficientemente separados.

- **Enlace simple ("single link" ó mínimo)**

El método de enlace simple, también llamado "del vecino cercano" ("*nearest neighbour*"), calcula la semejanza entre dos grupos como la semejanza entre los dos elementos más cercanos de ambos (figura 2.13). Este método es eficaz cuando los grupos tienen formas irregulares, pero es muy sensible a la existencia de elementos dispersos que no forman parte de ningún grupo definido, llevando a la creación de grupos alargados compuestos de objetos disímiles.

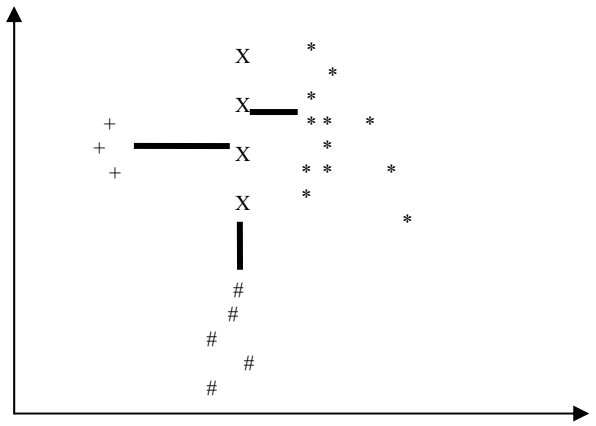

Figura 2.13: Distancias al grupo de las "X" según el método de enlace simple. (La distancia entre los grupos más cercanos está remarcada).

- **Enlace completo ("complete link" ó máximo)**

En el extremo opuesto del método de enlace simple se encuentra el método de enlace completo, que calcula la semejanza entre dos grupos usando la semejanza de los dos elementos más lejanos (figura 2.14). De esta manera, el método no sufre del efecto de "*encadenamiento*", y encuentra con eficacia grupos pequeños y compactos. Sin embargo, cuando los grupos no están bien definidos, puede llevar a la creación de grupos sin significado.

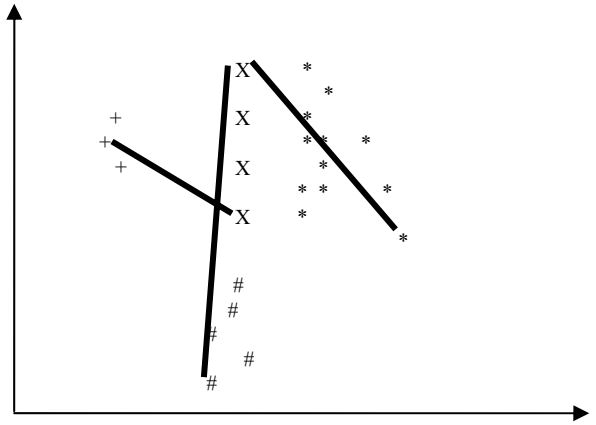

Figura 2.14: Distancias al grupo de las "X" según el método de enlace completo (La distancia entre los grupos más cercanos está remarcada).

- **Enlace promedio (“average link”)**

A mitad de camino entre los dos métodos anteriores, el algoritmo de enlace promedio define a la semejanza entre dos grupos como el promedio de las semejanzas de cada miembro de uno con cada miembro del otro (figura 2.15).

Al tomar propiedades de los métodos de enlace simple y completo, éste algoritmo obtiene resultados aceptables en un rango de situaciones más amplio [Cole, 1998].

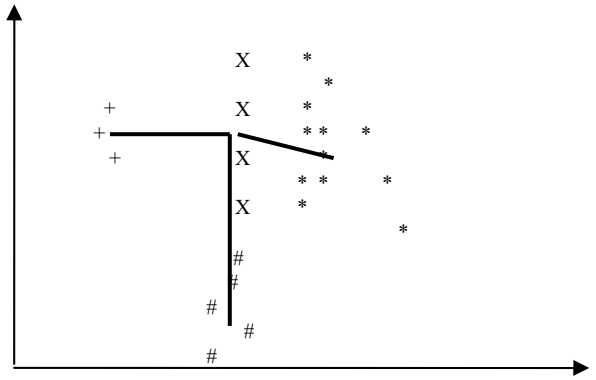

Figura 2.15: Distancias al grupo de las “X” según el método de enlace promedio (La distancia entre los grupos más cercanos está remarcada).

Los algoritmos divisivos se caracterizan por construir el árbol en dirección contraria a la utilizada por los aglomerativos. Comienzan con una sola clase en la que se encuentran todos los objetos del conjunto, y en cada paso del algoritmo dividen una clase en dos, hasta que cada clase consta de un solo objeto. Al igual que con los aglomerativos, es posible dar diferentes clasificaciones, en este caso, atendiéndonos a la forma de seleccionar la clase que va a ser dividida y la forma de dividir dicha clase.

Los algoritmos divisivos han sido mucho menos utilizados que los algoritmos aglomerativos debido en principio a su complejidad computacional y la consiguiente demanda de recursos informáticos.

En un primer paso de un algoritmo divisivo habría que mirar cuales son las posibles particiones del conjunto en dos, lo que hace un total de 2^{n-1}-1 comprobaciones, magnitud que crece exponencialmente con el tamaño, lo que es imposible realizarlo para conjuntos de un tamaño relativamente grande. Por esta razón los algoritmos divisivos han recogido menos estudios e interés por parte de los investigadores.

Para realizar la división se van pasando objetos de la clase a dividir a la nueva clase que se crea, la nueva clase comienza con el objeto más disimilar a todos los de la clase, y continúan uniéndose objetos siempre que su disimilaridad con los objetos que restan en la clase inicial sea mayor que la disimilaridad con la nueva clase. Para determinar cual es el objeto más disimilar a los de la clase, se utiliza alguna distancia entre un objeto y una clase.

Finalmente, considerando todos los métodos estudiados, podemos observar que los algoritmos de agrupación estudiados, tanto particionales como jerárquicos del tipo aglomerativo o divisivos, tienen un defecto: Son métodos de *optimización local*. En cada paso del algoritmo se elige la mejor opción, lo que significa que por su construcción, éste nos puede llevar hacia un óptimo local, con cualquier criterio que se esté utilizando. Es por ello por lo que tras examinar todos los conceptos revisados y comentados en este apartado, el presente trabajo propone un método de optimización global que usando criterios evolutivos tenga como resultado realizar un agrupamiento de documentos de manera no supervisada.

APARTADO II: ESTUDIO DE LOS ALGORITMOS EVOLUTIVOS

2.11. Introducción a los Algoritmos Evolutivos

Para situar a los Algoritmos Evolutivos en una perspectiva adecuada, deberíamos empezar por tener en cuenta una cantidad de técnicas de IA, entre cuyos objetivos se encuentre la optimización. En este conjunto de técnicas hay tres que son las más extendidas, éstas son:

- Enfriamiento Estadístico [Laarhoven Van, Aarts ,1987]
- Algoritmos Evolutivos [Bäck, 1996], [Rudolph, 1997]
- Búsqueda Tabú [Glower,1989, 1990]

Casi todos estos algoritmos están basados en simular principios naturales o procesos que se producen en la naturaleza. Así, en Enfriamiento Estadístico se intenta simular el proceso de "*annealing*" de un sólido ("*simulated annealing*"), es decir, el proceso por el cual un sólido es calentado hasta altas temperaturas, para dejarlo enfriar poco a poco de manera que acabe alcanzando el equilibrio. Los algoritmos evolutivos por su parte, tratan de simular la evolución de las especies de acuerdo con los postulados de Darwin. La búsqueda Tabú quizás sea la que más se aleje de esta analogía con la naturaleza, aunque sigue teniendo ciertas conexiones. Esta técnica de optimización hace uso de la memoria flexible como principal herramienta a la hora de llevar a cabo la búsqueda.

En este apartado repasaremos las características de la familia de los algoritmos evolutivos, inspirados en la teoría de la evolución de Darwin, profundizando en los dos tipos más conocidos y empleados, los Algoritmos Genéticos (AG) y la Programación Genética (PG), también analizaremos un algoritmo híbrido entre ambos.

Centrándonos en la Computación Evolutiva [Back, Fogel, Michalewicz, 1997], ella se basa en modelar procesos evolutivos para el diseño e implementación de sistemas que permitan resolver problemas mediante el ordenador. Los distintos modelos computacionales que se han propuesto dentro de esta filosofía suelen recibir el nombre genérico de Algoritmos Evolutivos (AE) [Bäck, 1996].

Existen cuatro tipos de AE bien definidos que han servido como base a la mayoría del trabajo desarrollado en el área:

i. Los Algoritmos Genéticos (AG) [Holland, 1975][Goldberg,1989].

ii. Las Estrategias de Evolución (EE) [Schwefel,1995].

iii. La Programación Evolutiva [Fogel L, 1966] y

iv. La Programación Genética (PG) [Koza, 1992].

Un AE trabaja manteniendo una población de posibles soluciones del problema a resolver, llevando a cabo una serie de alteraciones sobre la misma y efectuando una selección para determinar cuáles permanecen en generaciones futuras y cuáles son eliminadas. Aunque todos los modelos existentes siguen esta estructura general, existen algunas diferencias en cuanto al modo de ponerla en práctica. Los AG se basan en operaciones que tratan de modelar los operadores genéticos existentes en la naturaleza, como el cruce, y la mutación, los cuales son aplicados a las representaciones codificadas de los individuos que constituyen las posibles soluciones. En cambio, las EEs y la Programación Evolutiva aplican transformaciones basadas en la utilización exclusiva del operador de mutación para conseguir la nueva población, dejando de lado el operador de cruce lo que le permite mantener la línea general de comportamiento del individuo en su descendencia.

Finalmente, la PG considera las soluciones al problema en forma de programas, habitualmente codificados en una estructura de árbol, y adapta dichas estructuras empleando operadores muy específicos. Cada individuo de la población recibe un valor de una medida de adaptación que representa su grado de adecuación al entorno. La selección hace uso de estos valores y se centra en los individuos que presentan mejor valor de la misma. Los operadores de recombinación y/o mutación alteran la composición de dichos individuos, guiando heurísticamente la búsqueda a través del espacio. Aunque simples desde un punto de vista biológico, este tipo de algoritmos son suficientemente complejos para proporcionar mecanismos de búsqueda adaptativos muy robustos. Los mismos procedimientos pueden aplicarse a problemas de distintos tipos, sin necesidad de hacer muchos cambios [Fogel D, 1988].

2.12. Los Algoritmos Genéticos

Los AG fueron desarrollados por John Holland, junto a su equipo de investigación, en la universidad de Michigan en la década de 1970 [Holland, 1975], quien proporcionó los fundamentos teóricos de los mismos en su libro "*Adaptation in Natural and Artificial Systems*", aunque quién principalmente ha contribuido a su extensión y divulgación ha sido Goldberg con su libro [Goldberg ,1989]: "*Genetic Algorithms in search, optimization, and Machine Learning*". Desde la aparición del libro de Goldberg la literatura en el campo de los AG se ha incrementado de manera espectacular, existiendo en la actualidad gran cantidad de libros [Michalewicz, 1999], [Chambers L, 2001], [Bäck, Fogel, Michalewicz 1997] [Mitchell, 1996] que exponen los principios de este enfoque y gran parte de sus aplicaciones.

Los AG tratan de simular el comportamiento de las especies biológicas a lo largo del tiempo. Así como las poblaciones evolucionan por medio de la recombinación y por algunos cambios aleatorios que se producen en su material genético (mutación); de la misma manera funcionan los AG. Estos algoritmos mantienen en cada paso de su ejecución un conjunto de soluciones a un problema dado, las cuales se mezclan entre ellas (recombinación) dando lugar a nuevas soluciones. Las nuevas soluciones obtenidas son modificadas de manera aleatoria de tal forma que éstas tras la modificación pasan a formar parte de una nueva población.

Otro de los principios evolutivos: "*la supervivencia de los mejores adaptados*" también se aplica en los AG. Análogamente como en un entorno natural los individuos compiten por una cantidad de recursos escasos, y únicamente los más adaptados sobreviven y son capaces de reproducirse o lo hacen en mayor medida, de la misma forma en los AG, las mejores soluciones son transmitidas en cada generación del algoritmo, mezclándose posteriormente para producir nuevas soluciones.

Este paralelismo con la evolución de las especies hace que la nomenclatura utilizada en el campo de los AG sea muy parecida a la utilizada en el campo de la evolución. Por ejemplo, un conjunto de soluciones será llamado población, una solución en particular se llamará individuo o a veces cromosoma, y los componentes que forman los individuos se denominan genes; de esta forma un AG opera sobre una población compuesta de posibles soluciones al problema y cada elemento de la población se denomina "cromosoma". La forma en que los cromosomas codifican a la solución se denomina "*Representación*" (véase figura 2.16).

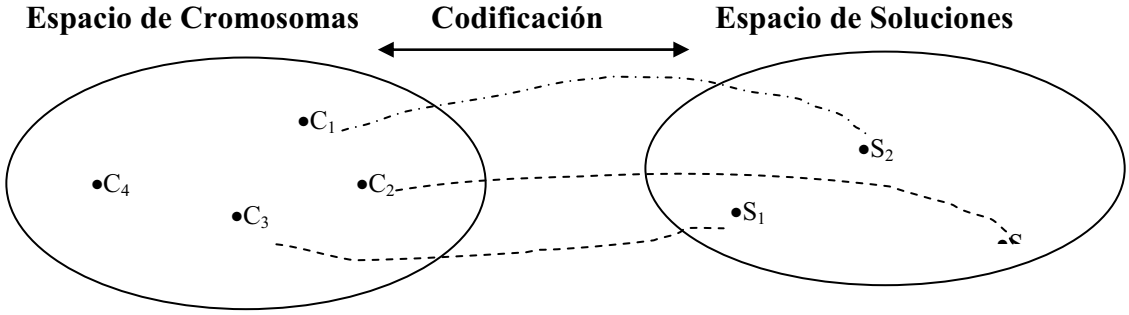

Figura 2.16: Representación de un Cromosoma: Cada cromosoma codifica una posible solución al problema

Desde su aparición, los AG han sido aplicados a gran cantidad de problemas prácticos de optimización donde se ha demostrado su eficacia. Son capaces de obtener buenas soluciones en tiempos de computación competitivos con el resto de algoritmos de optimización. Aunque para aquellos problemas para los que existan algoritmos específicos, normalmente estos funcionan mejor que los AG, es posible encontrar en la literatura situaciones, donde la creación de nuevos algoritmos híbridos entre AG y los algoritmos específicos, han podido mejorar los resultados existentes. Esto unido a la facilidad con la que es posible implementarlos ha llevado a la existencia de un crecimiento exponencial en las aplicaciones de los mismos.

Para su funcionamiento, el AG va creando nuevas "*generaciones*" a partir de una población inicial, cuyos individuos son cada vez mejores soluciones al problema. La creación de una nueva generación de individuos se produce aplicando a la generación anterior operadores genéticos.

La figura 2.17 representa el esquema de funcionamiento de un algoritmo genético. El proceso comienza seleccionando un número de cromosomas que formen parte de la población inicial. A continuación se evalúa la función de adaptación para estos individuos. La función de adaptación da una medida de la aptitud del cromosoma para sobrevivir en su entorno, debe estar definida de tal forma que los cromosomas que representen mejores soluciones tengan valores más altos de adaptación.

Los individuos más aptos se seleccionan en parejas para reproducirse. Entonces, la recombinación genera nuevos cromosomas que combinan características de ambos padres. Estos nuevos cromosomas reemplazan a los individuos con menores valores de adaptación. A continuación, algunos cromosomas son seleccionados al azar para ser mutados. La mutación consiste en aplicar un cambio aleatorio en su estructura. Luego, los nuevos cromosomas deben incorporarse a la población; estos cromosomas deben reemplazar a cromosomas ya existentes. Existen diferentes criterios que pueden utilizarse para elegir a los cromosomas que serán reemplazados. El ciclo de selección, recombinación y mutación se repite hasta que se cumple el criterio de terminación del algoritmo, momento en el cual el cromosoma mejor adaptado se devuelve como solución.

Existe una gran cantidad de problemas, entre los que se incluyen algunos de optimización, para los cuales no existe un algoritmo eficiente que los resuelva. En muchos de estos casos sería deseable disponer de métodos de resolución que proporcionen "*buenas*" soluciones si estas pueden ser obtenidas con rapidez. Los AG son algoritmos de búsqueda que permiten obtener buenas soluciones para muchos de estos problemas.

Así de esta forma, el funcionamiento de los algoritmos genéticos está basado en lo que se denomina la "*hipótesis de los bloques constructores*" [Goldberg, 1989]. Esta hipótesis requiere que los cromosomas se compongan de bloques significativos que codifiquen las características de la solución lo más independientemente posible.

El funcionamiento de un AG es bastante sencillo pero resulta eficaz en la búsqueda del óptimo, no estando limitado por restricciones en el espacio de búsqueda, se empieza de una población de candidatos a la solución óptima generada de forma aleatoria y evoluciona hacia la mejor solución en forma de selección. En cada generación las soluciones relativamente buenas son reproducidas y las soluciones relativamente malas mueren para ser reemplazadas por otras que presenten características de las buenas soluciones. Para evaluar en qué grado una solución es buena o mala para el problema tratado, se emplea una función de evaluación o función objetivo.

Como se ha dicho anteriormente, la terminología que se emplea en los AG está directamente importada de la Genética y de la Teoría de la Evolución de las Especies. Hablaremos de individuos de una población que serán denominados cromosomas y que están formados por unidades llamadas genes. Cada Gen podrá manifestarse de forma diferente, es decir, tomar distintos valores, que se conocen con el nombre de alelos o valores del rasgo.

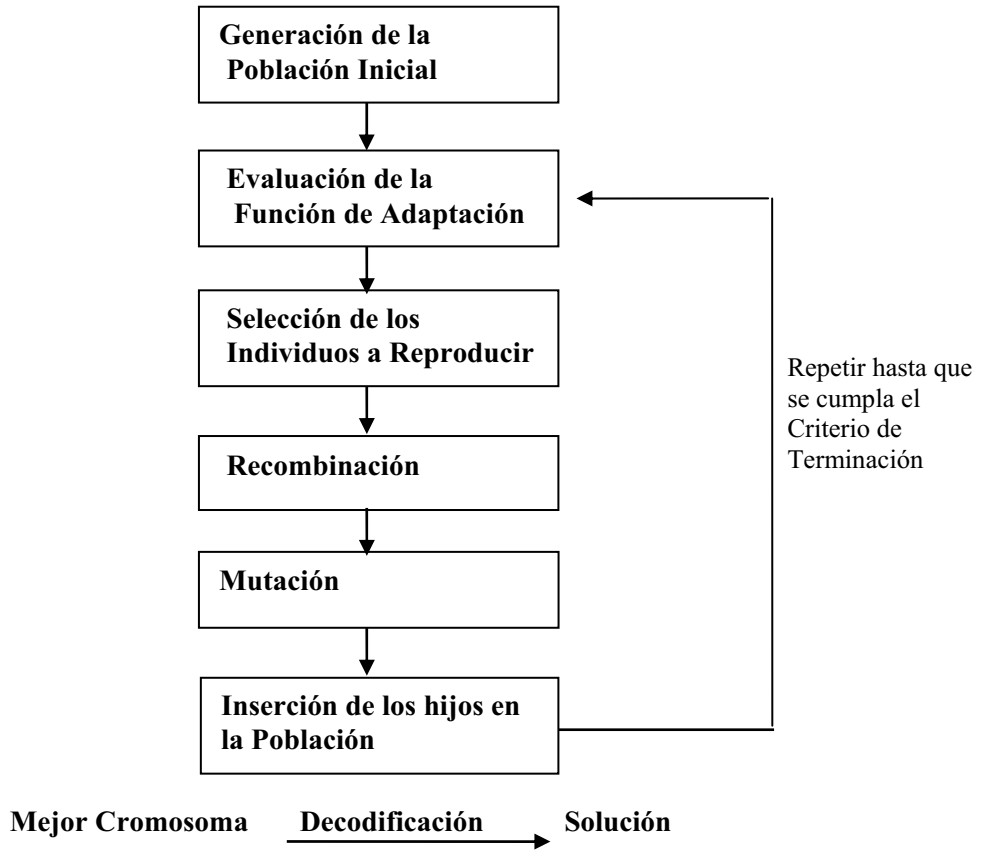

Figura 2.17: Esquema de funcionamiento del Algoritmo Genético

2.13. Funcionamiento básico de los Algoritmos Genéticos

Un AG simple mantiene en cada paso de su ejecución un conjunto de soluciones a un problema al que llamaremos población, cada elemento de la población, es decir, cada posible solución al problema será llamado individuo. En el caso del AG básico, las posibles soluciones del problema que se quiere optimizar vienen codificadas como vectores de ceros y unos. Cada individuo de la población es un vector de longitud *L* que codifica una solución. Se requiere una función *F*, llamada función objetivo o de adaptación, que mida el grado en que cada individuo resuelve el problema.

En términos evolutivos la función *F* mide el grado de adaptación del individuo al entorno. Si nos encontramos ante un problema de optimización normalmente la función *F* suele ser la propia función a optimizar.

Cada iteración del AG simple consiste en la realización de tres operaciones básicas: selección, recombinación y mutación. La primera de estas tres operaciones realiza una selección de dos individuos de la población, esta selección suele ser llevada a cabo de manera aleatoria pero proporcional al valor que la función objetivo *F* toma en cada individuo.

La segunda operación, es decir la recombinación, se efectúa con cierta probabilidad, esta operación consiste en la aplicación de un operador de cruce que trata de simular la mezcla genética que se produce en el mundo natural, mezclando los componentes de los dos individuos seleccionados.

En el caso del AG simple se suele utilizar lo que se llama operador de cruce basado en un punto.

La última de las operaciones consiste en aplicar un operador de mutación, en este caso el operador modifica el valor de algún gen de un individuo de acuerdo con una probabilidad. En el AG simple, generalmente, cada individuo de la población se representa mediante una cadena binaria de longitud fija que codifica los valores de las variables que intervienen en el problema, con lo cual las representaciones de los datos (cadenas de bits) y las operaciones sobre ellas pueden manipularse para generar cadenas de bits que estén bien adaptadas al problema a resolver. Como hemos comentado, los mecanismos fundamentales que aplican los AG operan sobre una población de individuos, generada inicialmente de forma aleatoria y cambian la población en cada iteración atendiendo a los tres pasos siguientes:

1. Evaluación de los individuos de la población.

2. Selección de un nuevo conjunto de individuos, sobre la base de su adaptación relativa.

3. Alteración de los individuos seleccionados para formar una nueva población a partir de los operadores de cruce y mutación.

Los individuos resultantes de las tres operaciones anteriores forman la siguiente población, repitiéndose el proceso hasta que se dé una cierta condición de parada.

Durante la iteración *t*, el AG mantiene una población *P(t)* de *M* soluciones $C_1^t, C_2^t ..... C_M^t$, en donde cada solución C_i^t se evalúa por la función *F*, de modo que $F(C_i^t)$ da una medida de conveniencia de la misma, en la siguiente iteración, *t+1*, se forma una nueva población en base a las operaciones 2 y 3.

Existen dos modelos básicos de evolución para un AG, que dependen del método de selección y reemplazamiento de solución considerado.

- El modelo generacional, también conocido como modelo clásico, en el que durante cada generación se crea una población completa con nuevos individuos mediante la selección de padres de la población anterior y la aplicación de los operadores genéticos sobre ellos. La nueva población reemplazará directamente a la antigua.

- El modelo estacionario, en el que durante cada generación se escoge un número más reducido de individuos, normalmente dos padres de la población (usando cualquier mecanismo de selección), y se aplican los operadores genéticos sobre ellos. Los dos nuevos cromosomas (o el único hijo) reemplazan a dos cromosomas de la población (normalmente a los dos peores).

Mostramos a continuación, los algoritmos de las dos estructuras nombradas anteriormente (generacional y estacionaria) para un AG:

Modelo Generacional:

```
Comienzo
t = 0
Inicializar P(0)
Evaluar P(0)
Mientras  que no se de la condición de parada hacer
     Comienzo
          t = t + 1
          Seleccionar P(t)  desde  P(t-1)
          Cruzar y Mutar P(t) con probabilidad  Pc y Pm
          respectivamente
          Evaluar P(t)
     Fin
Devolver el mejor individuo de P(t)
Fin
```

Modelo Estacionario:

```
Comienzo
t = 0
Inicializa P(0)
Evaluar P(0)
Mientras que no se de la condición de parada hacer
     Comienzo
          t = t + 1
          Seleccionar dos padres de P(t-1)
          Cruzar los dos padres para obtener dos hijos
          Mutar los dos hijos con probabilidad  Pm
          Añadir los dos descendientes a P(t)
          Eliminar los dos peores individuos de P(t)
     Fin
Devolver el mejor individuo de P(t)
Fin
```

Como se observa en ambos algoritmos, asociadas a los operadores de cruce y mutación existen la probabilidad de cruce P_c, y la probabilidad de mutación, P_m., la primera representa la probabilidad de aplicar el operador de cruce sobre parejas de cadenas obtenidas en la recombinación mientras que la segunda representa la probabilidad de aplicar el operador de mutación sobre los caracteres de cada cadena.

2.14. Representación del cromosoma en los Algoritmos Genéticos

El esquema de representación es el que define de qué forma se corresponden los cromosomas con las soluciones al problema. Para diseñar el esquema de representación, se buscan los parámetros que identifican a las soluciones, y luego se codifican estos parámetros dentro del cromosoma.

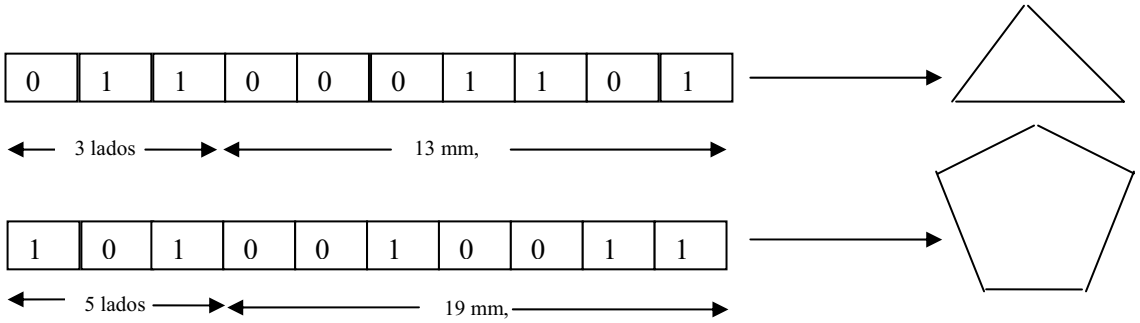

Figura 2.18: Ejemplo de un esquema de representación para polígonos regulares con un AG

La figura 2.18 muestra un posible esquema de representación para un problema que tiene como soluciones a los polígonos regulares. Los parámetros que identifican a cada solución son 2 (cantidad de lados y longitud del lado), y estos se codifican en el cromosoma en forma binaria. El cromosoma se compone de una cadena de 10 bits, en los que los primeros 3 son la cantidad de lados, y los siguientes 7 bits representan la longitud de los lados en milímetros.

El esquema de representación debería ser tal que exista al menos una posible codificación para cada una de las soluciones posibles. Las soluciones que no estén dentro del espacio de cromosomas no serán exploradas por el algoritmo genético. En el ejemplo de la figura 2.18, el algoritmo genético no explorará soluciones que se compongan por polígonos de más de 7 lados ni longitudes mayores a 127 milímetros, ya que con 3 y 7 bits pueden codificarse solamente números del 0 al 7, y del 0 al 127, respectivamente.

Por el mismo motivo (porque la búsqueda se hace sobre el espacio de cromosomas), es deseable que no haya redundancia en la representación; que cada solución sea representada por solamente un cromosoma. Si existen *K* cromosomas por cada solución, el espacio de búsqueda sobre el que opera el algoritmo genético es *K* veces más grande que el espacio de soluciones, haciendo más lento la evolución.

La representación ejemplificada en la figura 2.18 no tiene redundancia, cada polígono es representado sólo por un cromosoma, Un problema que puede presentarse es que haya cromosomas que no representan ninguna solución, por ejemplo en la figura, un cromosoma que tenga todos 0 en los primeros 3 ó en los últimos 7 bits no representa un polígono válido.

La codificación ejemplificada en la figura anterior se denomina "*binaria*", ya que cada posición del cromosoma contiene un bit. Esta es la representación clásica propuesta por los primeros autores y que todavía es utilizada ampliamente [Goldberg, 1989]; [Falkenauer, 1999], sin embargo, hay problemas para los cuales esta representación no es la más conveniente y se podría utilizar una representación mediante números reales.

2.15. Generación de la población inicial en AG

La población inicial es la principal fuente de material genético para el algoritmo genético. La población inicial debe contener cromosomas que estén bien dispersos por el espacio de soluciones, la manera más simple de cumplir con este objetivo es elegir cromosomas al azar.

El uso de una heurística [Diaz A, 1996] puede ayudar a generar una población inicial compuesta de soluciones de cierta calidad, ahorrando tiempo al proceso de evolución, siempre y cuando se pueda garantizar una dispersión suficiente por todo el espacio de soluciones, que permita empezar la búsqueda con garantías de no ignorar soluciones prometedoras.

2.16. La función de adaptación en AG

La función de adaptación cuantifica la aptitud de cada cromosoma como solución al problema. Determina su probabilidad de ser seleccionado para la fase de recombinación y poder pasar parte de su material genético a la siguiente generación. La función de adaptación hace evolucionar la población hacia cromosomas más aptos, por lo que una buena definición de esta función es fundamental para un correcto funcionamiento del algoritmo. La función debe asignar el valor más alto al cromosoma que representa la solución óptima al problema, soluciones cercanas a la óptima deben obtener valores altos, y la función debe ir decreciendo a medida que los cromosomas representan soluciones cada vez más alejadas de la óptima. Debido a que, el proceso de evolución tiende a retener el material genético de los cromosomas con valores altos de aptitud, una elección apropiada de la función de adaptación resultará mayor probabilidad de retener características de soluciones cercanas a la óptima.

Se debe tener en cuenta en el diseño de la función de adaptación "*la escala*" [Goldberg, 1989]. Cuando la aptitud de los elementos de la población es variada (al inicio del proceso de evolución), es común que la población contenga unos pocos cromosomas cercanos al óptimo, rodeados de cromosomas que representan soluciones mediocres. Los cromosomas más aptos obtendrán valores superiores al promedio para la función de adaptación, y el AG elegirá solamente a estos cromosomas para la recombinación, esto puede ocasionar lo que se denomina "*convergencia prematura*". Este problema puede paliarse aplicando una función de escala a la función de adaptación.

La función de *escala lineal* calcula la nueva función de adaptación *f'* a partir de la función de adaptación *f* usando la transformación lineal $f' = a.f + b$. Las constantes *"a"* y *"b"* se eligen de forma tal que el promedio de las dos funciones sea el mismo.

2.17. El mecanismo de selección en AG

La selección de los individuos que van a reproducirse de la población a partir de la antigua se realiza mediante un operador de selección. El operador de selección hace la elección basándose en los valores de adaptación de los individuos.

Existen distintos operadores de selección que pueden utilizarse [Miller, Goldbert, 1995], de los cuales comentaremos algunas de las mostradas en los trabajos de [Bäck , Schwefel, 1991] [Bäck, Fogel , Michalewicz,1997].

2.17.1. Modelo de muestreo aleatorio simple

Es el mecanismo clásico que propuso Holland [Holland, 1975], en este modelo se calcula la probabilidad de selección de forma proporcional al valor de la función de adaptación según la expresión:

$$P_{c_i} = \frac{F(C_i)}{\sum_{j=1}^{M} F(C_j)}, \quad i = 1,.....M \qquad (2.14)$$

Después se obtienen los *M* cromosomas, que van a configurar la nueva población por sorteo de los *M* individuos de acuerdo a dicha distribución de probabilidad. Este método llamado también el de la ruleta consiste en crear una ruleta en la que cada cromosoma tiene asignada una fracción proporcional a su aptitud. Por ejemplo supongamos que se tiene una población de 4 cromosomas cuyas aptitudes están dadas por los valores mostrados en la figura 2.19.

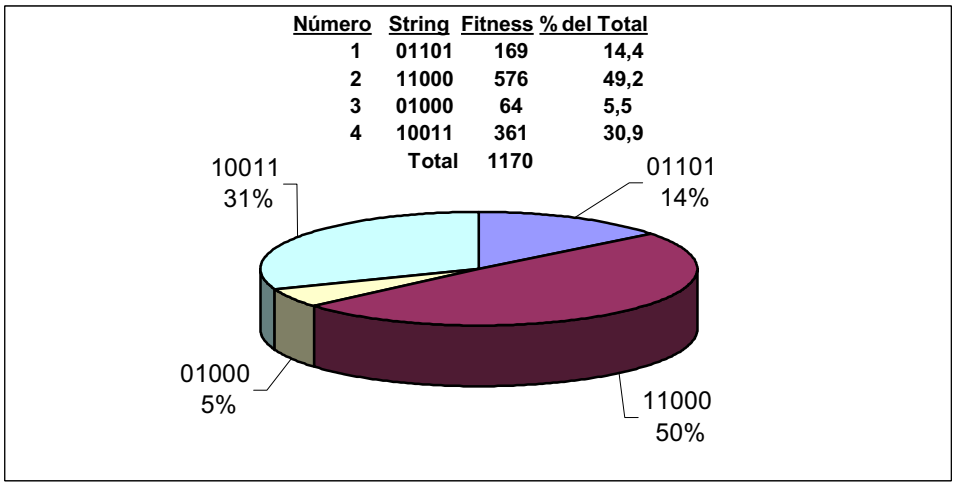

Figura 2.19: Ejemplo del método de la Ruleta

Con los porcentajes mostrados en la cuarta columna de la figura 2.19 podemos elaborar la ruleta de la figura. Esta ruleta se gira varias veces para determinar qué individuos se seleccionarán. Debido a que, a los individuos más aptos se les asigna un área mayor de la ruleta, se espera que sean seleccionados más veces que los menos aptos.

2.17.2. Muestreo universal estocástico

En 1987, John Baker presentó esta versión de selección como complemento final de otras que había desarrollado anteriormente [Chambers L, 2001]. El método se basa en el empleo de la misma ruleta considerada en el muestreo aleatorio simple, con un número de marcas igualmente espaciadas, equivalente a la longitud de la población.

La idea es bastante sencilla y efectiva, los puntos de recombinación están espaciados en las posiciones i/M Cada cromosoma $C_i(t)$ tiene asociada una amplitud:

Este tipo de muestreo llamado "*muestreo universal estocástico*", tiene en cuenta la ruleta sesgada, pero en lugar de realizar varios giros; utiliza un único giro para seleccionar todos los individuos. Para ello dispone de un sistema de marcadores igualmente espaciados cuya posición se elige aleatoriamente.

2.17.3. Selección por Torneo

La selección por torneo constituye un procedimiento de selección de padres muy extendido. La idea consiste en escoger al azar un número de individuos de la población, determinar el mejor tamaño del torneo, con o sin reemplazamiento, seleccionar el mejor individuo de este grupo, y repetir el proceso hasta que el número de individuos seleccionados coincida con el tamaño de la población. Habitualmente el tamaño del torneo es 2, denominado "*torneo binario*", que es el que proporciona mayor diversidad y menor presión selectiva.

Así, en este método se baraja la población y después se hace competir a los cromosomas que la integran en grupos de tamaño predefinido (normalmente compiten en parejas) en un torneo del que resultarán ganadores aquéllos que tengan valores de aptitud más altos. Si se efectúa un torneo binario (competencia por parejas) se utiliza una versión probabilística en la cual se permite la selección de individuos sin que necesariamente sean los mejores.

Es importante señalar, que cualquiera de los mecanismos de selección pueden ser completados por un modelo de selección elitista, basado en mantener un número determinado de los individuos mejor adaptados de la población anterior en la nueva población (la obtenida después de llevar a cabo el proceso de selección y de aplicar los operadores de cruce y mutación) [Goldberg, 1989],[Michalewicz,1999].

Con estos métodos se intenta evitar la selección de un súper individuo en las primeras generaciones, y evitar el problema de la convergencia prematura en el sistema.

2.18. El operador de cruce en AG

La recombinación se implementa por medio de un operador que es el encargado de transferir el material genético de una generación a la siguiente. Este operador confiere a la búsqueda de soluciones mediante algoritmos genéticos su característica más distintiva [Falkenauer, 1999]. A diferencia de otros métodos de optimización, los algoritmos genéticos no solamente exploran el entorno de las buenas soluciones, sino que recombinan sus partes para formar nuevas soluciones El objetivo de los operadores de recombinación es, partiendo de dos cromosomas padres, generar uno o más cromosomas hijos que hereden características de ambos padres, tal como se muestra en la figura 2.20. Se dice que en el hijo "*recombina*" las características de los padres. Si las características se traspasan en bloques significativos, se espera que un hijo que recombina características de buenas soluciones sea una buena solución, tal vez mejor que cualquiera de sus padres [Falkenauer, 1999].

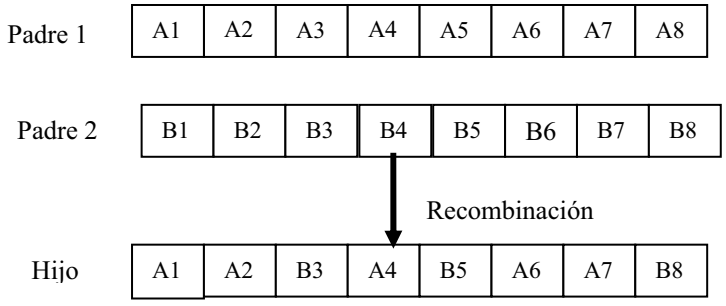

Figura 2.20: Ejemplo de recombinación, el hijo hereda características de ambos padres

Este proceso, no se suele aplicarse a todas las parejas de cromosomas de la población intermedia, sino que se lleva a cabo una selección aleatoria en función de una determinada probabilidad (probabilidad de cruce *Pc)*. Normalmente el cruce se maneja con un porcentaje que indica con qué frecuencia se efectuará, esto significa que no todas las parejas de cromosomas se cruzarán, sino que habrá algunas que pasarán intactas a la siguiente generación.

Existe una técnica, desarrollada hace algunos años, en la que el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie, y se mantienen intactos hasta que surge otro individuo mejor que él, que lo desplazará; dicha técnica es llamada *elitismo.*

La definición del operador de cruce depende directamente del tipo de representación empleada. Así por ejemplo, trabajando con el esquema de codificación binario, se suele emplear el clásico cruce simple en un punto, basado en seleccionar aleatoriamente un punto de cruce e intercambiar el código genético de los dos cromosomas padre a partir de dicho punto para formar los dos hijos, o el cruce multipunto que procede del mismo modo que el anterior pero trabajando sobre dos o más puntos de cruce.

El operador de cruce juega un papel fundamental en el AG, su tarea es la de explorar y explotar el espacio de búsqueda refinando las soluciones obtenidas mediante la combinación de las buenas características que se presenten. Los operadores de recombinación más típicos generan dos hijos a partir de dos padres. A continuación se describen los más utilizados.

2.18.1. Cruce monopunto

Este operador consiste en separar a los padres en dos partes para formar dos hijos intercambiando las partes de cada padre. Si los cromosomas se componen de *N* bloques, se elige un número al azar *c,* tal que *1* <= *c* < *N*, y luego se asigna al primer hijo los primeros *c* bloques del primer padre y los últimos *N* - *c* bloques del segundo padre; se procede en forma inversa para formar al segundo hijo.

Así en la figura 2.21 se muestra un ejemplo de la aplicación de éste operador.

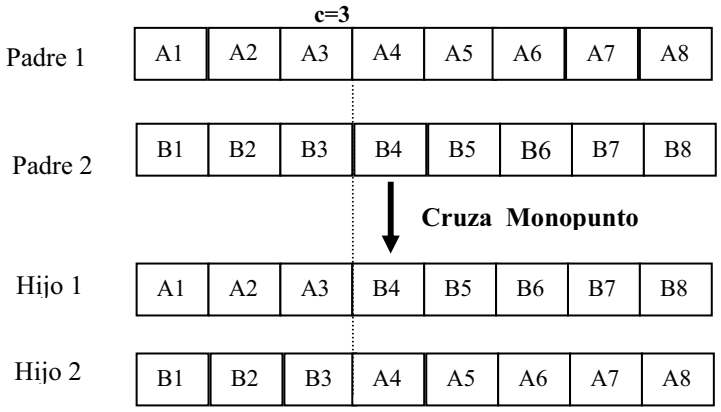

Figura 2.21: Ejemplo del método de Cruce Monopunto

2.18.2. Cruce multipunto

En el operador de cruza monopunto, el primer y último bloque de uno de los padres no pueden pasar juntos al hijo en ningún caso. En el operador de cruza

multipunto se avanza un paso más, quitando esta restricción. En la cruza multipunto, se eligen *M* puntos de corte al azar, y las secciones de cada padre se pasan a los hijos en forma alternada. La figura 2.22 muestra este procedimiento para el caso en que *M* es igual a 2.

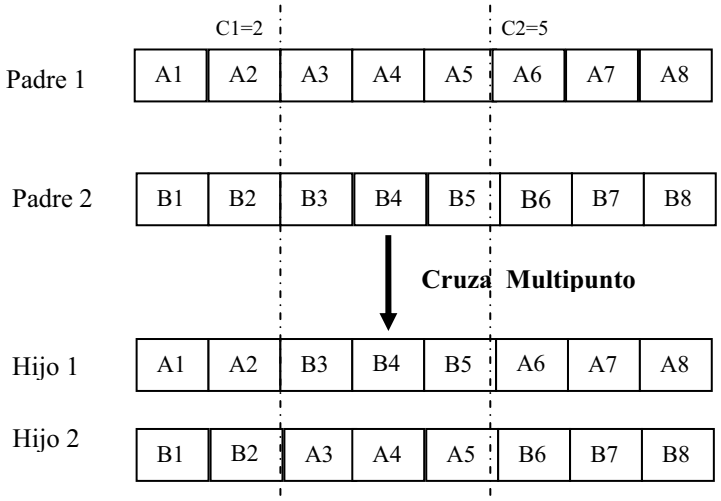

Figura 2.22: Ejemplo del método de Cruce Multipunto

2.18.3. Cruce uniforme

Si bien la cruza multipunto es más flexible que la monopunto, tiene algunos inconvenientes. En primer lugar, para valores impares de *M* impide que el primer y último bloque de los padres pasen juntos a los hijos. En segundo lugar, los bloques consecutivos tienen más posibilidades de pasar juntos que los que se encuentran más distanciados. Esto no es deseable (a menos que la codificación elegida haga que los bloques consecutivos representen características relacionadas)

El operador de cruza uniforme permite el intercambio de los bloques, tal que es independiente del orden que la codificación impuso a cada uno dentro del cromosoma. Para cada posición de los cromosomas hijos, se elige al azar cuál de los dos bloques de los cromosomas padres se copia en cada uno. Esta es una generalización del esquema de cruza multipunto donde la cantidad de puntos de corte *M* se elige al azar.

2.19. El operador de mutación en AG

Este segundo operador altera arbitrariamente uno o más genes del cromosoma seleccionado, con el propósito de aumentar la diversidad de la población. Todos los genes de los cromosomas existentes están sujetos a la posibilidad de mutar de acuerdo a una probabilidad de mutación P_m^{gen}. Sin embargo, es habitual que la probabilidad especificada sea la de mutación de cromosomas P_m^{crom}, y que después se pueda trabajar con la relación existente entre ambas: $P_m^{gen} = P_m^{crom} / número\ genes$

En este caso, la propiedad de búsqueda asociada al operador de mutación es la exploración, ya que la alteración aleatoria de uno de los componentes del código genético de un individuo, suele conllevar el salto a otra zona del espacio de búsqueda que puede resultar más prometedora. El operador de mutación clásicamente empleado en los AG con codificación binaria se basa en cambiar el valor del bit seleccionado para mutar por su complementario en el alfabeto binario.

2.20. Características principales de los Algoritmos Genéticos

La aplicación de un AG a un problema conlleva el análisis y diseño de los cinco componentes siguientes:

- Una representación genética de las posibles soluciones al problema.
- Un mecanismo de creación de la población inicial de soluciones.
- Una función de adaptación que asocie un valor de adecuación a cada cromosoma.
- Una serie de operadores genéticos que alteren la composición genética de los descendientes durante el proceso de recombinación, así como un mecanismo de selección de los padres que intervienen en dicho proceso.
- Una serie de valores para los parámetros del AG: tamaño de la población (M), probabilidades de aplicación de los operadores genéticos (Pc y Pm), etc.

Los AG han sido aplicados con éxito en una gran cantidad de áreas distintas; algunas de las razones de este éxito son las siguientes:

- Pueden resolver problemas muy complejos de una forma rápida y sencilla.
- Pueden enlazarse fácilmente con simulaciones y modelos existentes.
- Presentan gran facilidad para ser combinados con otras técnicas.

En resumen, los AG son muy robustos. Destacan por su buen comportamiento en problemas difíciles en los que el espacio de búsqueda es grande, discontinuo, complejo y poco conocido. Suelen proporcionar soluciones bastantes aceptables en un tiempo razonable.

Las aplicaciones de los AG ha crecido enormemente a lo largo de los últimos años en campos tan diversos como la optimización de funciones numéricas y la optimización combinatoria [Goldberg,1989],[Michalewicz,1999], la Inteligencia Artificial [Nils Nilsson, 2001], la Investigación Operativa [Nannen V, Smit K, Eiben E, 2006], las Redes Neuronales [Schaffer et. al,1992], los Sistemas Difusos [Pedrycz W,1997],[Cordon, Herrera, Hoffman, Magdalena, 2001], la Robótica [Davidor,1991], la Vida Artificial [Pedrycz W, 1998], el Aprendizaje Automático [De Jong, et. al.1993],[Janikov,1993], etc.

2.21. Análisis de las modificaciones y extensiones a los AG en trabajos previos

Aunque muchos de los aspectos que veremos a continuación están interrelacionados unos con otros, analizaremos cada componente por separado, apuntando en cada caso éste tipo de relaciones basados en los trabajos previos revisados en la bibliografía. Comenzaremos con aspectos que tienen que ver con la población.

2.21.1. Estudio y análisis del tamaño de la población

Evidentemente el tamaño de la población parece ser un aspecto crucial en el AG. Si por ejemplo utilizamos poblaciones pequeñas corremos el riesgo de no cubrir todo el espacio de búsqueda, sin embargo, utilizar grandes poblaciones puede suponer un esfuerzo computacional excesivo. En [Goldberg, 1989] se indica que el tamaño de la población óptima con individuos binarios crece en teoría exponencialmente con la longitud del individuo. Por lo que se haría imposible aplicarlo en problemas reales.

Luego, dando diferentes argumentos, afirma que en la práctica, un tamaño de población relativamente pequeño suele ser suficiente.

Otros estudios experimentales realizados por Alander [Alander, 1992] sugieren que un tamaño de población entre *l* y *2l* (l longitud del cromosoma) es suficiente en la mayoría de las aplicaciones prácticas. En [Reeves, 1993b] se trata el problema de la utilización de AG con poblaciones pequeñas, dando un valor mínimo que deben tener estas poblaciones para cumplir ciertas propiedades, mostrando que el uso de codificaciones no binarias, hace que dicho valor mínimo aumente en relación con el necesario para individuos binarios. Otros autores [Arabas, Michalewicz, Mulawka, 1994], proponen que el uso de tamaños de población variables es beneficioso para la búsqueda.

2.21.2. Estudio y análisis de la población inicial

En el AG simple hemos visto que la población inicial era generada de forma aleatoria. Sin embargo, es fácil pensar en otras posibilidades. En [Reeves, 1993b] se expone una forma de inicializar la población más conveniente que la aleatoria, en el caso de tener que trabajar con poblaciones pequeñas. También, en [Reeves, 1993a], se muestran utilizaciones de los AG, donde la población inicial esta formada por soluciones obtenidas mediante la aplicación de otro optimizador.

Aunque estas formas de inicializar la población, pueden ayudar al algoritmo a conseguir mejores soluciones en menor tiempo, también tiene el problema de que la utilización de muy buenos individuos iniciales puede conducir a la convergencia prematura. Este problema consiste en la existencia de un individuo cuyo valor de la función objetivo es mucho mejor que los demás (*superindividuo*), el cual en muchos casos es un óptimo local.

2.21.3. Estudio y análisis de la selección de individuos

En el algoritmo propuesto por Holland [Holland, 1975], la selección se lleva a cabo de manera proporcional al valor que cada individuo toma en la función objetivo. De esta forma, si denotamos I_i el *i-esimo* individuo de la población, la probabilidad de seleccionar dicho individuo viene dada por la ecuación 2.14.

La principal crítica recibida por la selección usada en el AG simple, es que este tipo de selección conlleva al problema de la convergencia prematura. Para subsanar este inconveniente surgió la selección basada en el rango, y otros métodos de selección.

La selección basada en el rango, consiste en que la probabilidad de seleccionar un individuo no dependa directamente del valor que toma la función objetivo en él, sino del lugar (rango) que ocupa dicho valor en el conjunto de la población De esta forma, si definimos $rg(F(I_i))$ como el rango de la función objetivo cuando los individuos de la población han sido ordenados de mayor a menor (el de menor función objetivo sería el primero y el de mayor el último, si por ejemplo se está maximizando). Entonces la probabilidad de selección es:

$$p_i = \frac{rg(F(I_i))}{n(n-1)/2} \qquad (2.15)$$

Utilizando las anteriores dos técnicas de selección, ocurre en muchos casos que el resultado real de la selección está muy lejos del resultado esperado. Para evitar este problema, De Jong ideó la "*selección de valor esperado*" [De Jong, 1975]. En esta técnica de selección se intenta forzar a que los individuos sean seleccionados tantas veces como su valor esperado. Esto se hace asignando a cada individuo un contador inicializado a $F(I_i)\big/\bar{F}$, donde $\bar{F}$ representa la media de la función objetivo en la población. Una vez que un individuo ha sido elegido, dicho contador decrece una cantidad **c**, c $\in$ (0.5, 1). El individuo no podrá ser seleccionado más veces cuando su contador tome un valor menor o igual que cero.

La técnica propuesta por Baker [Chambers L, 2001] llamada "*muestreo universal estocástico*", sigue la misma línea que las anteriores, tiene en cuenta la ruleta sesgada, pero en lugar de realizar varios giros; utiliza un único giro para seleccionar todos los individuos. Para ello dispone de un sistema de marcadores igualmente espaciados cuya posición se elige aleatoriamente.

En la figura 2.23 podemos ver un ejemplo de ello; en este caso el individuo *I1* ha sido seleccionado una vez, el individuo *I2* otra, *I3* ninguna, mientras que *I4* ha sido seleccionado dos veces.

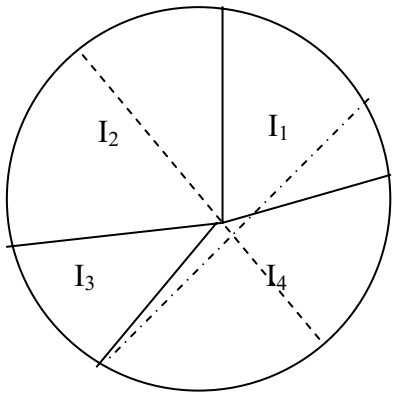

Figura 2.23: Ejemplo de Muestreo Universal

Finalmente, en la selección por torneo se escoge de manera aleatoria un conjunto de individuos de la población, y se efectúa dentro de ellos una competición decidiendo finalmente que individuo es seleccionado. En esta competición influye el valor de la función objetivo. Habitualmente el número de individuos elegidos para llevar a cabo el torneo es dos. Comparaciones entre diferentes tipos de selección pueden encontrase en Goldberg y Deb [Goldberg, Deb, 1991] y en [Bäck, 1994]. En el primer trabajo los autores apuntan que la selección basada en el rango y la selección por torneo obtienen mejores resultados. En el segundo trabajo, Bäck analiza distintos criterios de selección en cuanto a su presión selectiva sobre la población; esta presión selectiva la mide en función del número de iteraciones de selección necesarias para obtener una población compuesta por todos los individuos iguales al mejor individuo de la población inicial.

2.21.4. Estudio y análisis de la codificación de las soluciones

La codificación de las soluciones es otra cuestión importante en los AG. En un AG entendido en su forma más simple cada individuo está codificado como un vector de ceros y unos, sin embargo hay otras posibles codificaciones. Entre las más habituales formas de codificar una solución a la hora de utilizar AG tenemos:

- Vectores donde cada gen toma un valor elegido en un conjunto finito dado
- Vectores formados por varios valores de un conjunto finito de números dados.
- Vectores formados por genes que toman valores de números reales.

Cada uno de estas codificaciones, poseen sus operadores específicos. En la mayoría de los casos, los operadores expuestos en el apartado anterior no sirven para este tipo de codificaciones. La discusión en cuanto al tipo de codificación que sea más adecuada a un problema dado, ha traído durante mucho tiempo de cabeza a los investigadores, y no hay conclusiones categóricas en este sentido.

2.21.5. Estudio y análisis del Elitismo

De entrada, en teoría, en el AG simple, la nueva población compuesta por los hijos, sustituiría por completo a la generación anterior. Pero esto no se suele realizar así en la práctica. Habitualmente se utiliza lo que se ha venido a llamar "*elitismo*". El primer autor en proponer este concepto fue De Jong [De Jong, 1975] en su tesis doctoral. Para el autor un AG era elitista si el mejor individuo de la población en la generación *k* era introducido en la población en la generación *k+1*.

En [Eiben A, Marchiori A, Valko A, 2001] se comenta otro tipo de elitismo mediante "*operadores de reducción"*. Los Operadores de reducción se basan en utilizar tanto la población de los hijos como la de los padres para obtener la siguiente población. Un tipo de operador de reducción muy común es el operador de reducción elitista de grado 1. Este operador consiste en introducir en la generación *k+1* el mejor individuo del conjunto formado por la población en la generación *k* y los hijos.

Otra cuestión no comentada hasta el momento es el hecho de que en cada iteración del AG se generan *n* individuos. Este hecho ha sido modificado por algunos investigadores, que en vez de generar en cada paso *n* individuos, generan menos, de tal forma que la población en el siguiente paso del algoritmo está formada por algunos de los individuos que estaban en ella en el tiempo *k* y algunos de los hijos generados.

2.21.6. Estudio y análisis de la inserción de los hijos en la población

La reinserción de hijos consiste en incorporar los nuevos cromosomas en la población. Los métodos de reinserción son diferentes según la cantidad de cromosomas generados sea menor, igual o mayor que la cantidad de elementos existentes en la población; estos esquemas de inserción son los que analizaremos a continuación:

- **Se generan tantos cromosomas como elementos en la población**

El esquema denominado "*reinserción pura*", consiste en generar mediante la recombinación tantos hijos como elementos existen en la población, y reemplazar todos los cromosomas de la generación anterior por los nuevos [Goldberg, 1989]. Cada cromosoma vive exactamente una generación. Un inconveniente que presenta este método es que suele reemplazar buenas soluciones por soluciones de menor calidad.

- **Se generan más cromosomas que elementos en la población**

Este esquema es similar al de reinserción pura; se eligen los mejores cromosomas entre los que se generaron, y se eliminan los cromosomas sobrantes; luego, se reemplaza la población completa por la nueva generación.

- **Se generan menos cromosomas que elementos en la población**

Este esquema exige seleccionar entre los cromosomas de la población aquellos que se eliminarán, los métodos más utilizados para hacerlo son:

  - **Inserción uniforme.**

  Los cromosomas a ser reemplazados se eligen al azar entre los miembros de la población. Se corre el riesgo de eliminar buenas soluciones, ya que no se tiene en cuenta la aptitud de los cromosomas.

  - **Inserción elitista.**

  Se eligen los cromosomas menos aptos para ser reemplazados. Este método asegura que los mejores cromosomas pasarán siempre a la siguiente generación, pero puede restringir la amplitud de la búsqueda que realiza el algoritmo genético.

  - **Inserción por torneo invertido.**

  Se utiliza en combinación con el método de selección por torneo. Funciona exactamente igual que el método de selección por torneo pero seleccionando con probabilidad *p* al peor cromosoma del torneo para ser reemplazado, Tiene las mismas propiedades que el torneo, permitiendo regular la presión selectiva sin el uso de funciones de escala.

2.21.7. Estudio y análisis de los criterios de terminación del proceso de funcionamiento del Algoritmo Genético (AG)

El criterio de terminación del algoritmo genético es el encargado de definir el momento en el cual debe detenerse el ciclo de evolución y adoptar el cromosoma más apto como la solución encontrada por el algoritmo genético. Los criterios usados son:

- **Criterio de convergencia de identidad**

Este criterio consiste en detener al algoritmo genético cuando un determinado porcentaje de los cromosomas de la población representan a la misma solución. Los operadores del algoritmo genético tienden a preservar y difundir el material genético de los cromosomas más aptos, por lo que es de esperar que luego de un gran número de generaciones, alguna solución con gran valor de aptitud se imponga y domine la población.

- **Criterio de convergencia de aptitud**

Puede suceder que existan soluciones equivalentes o cuasi equivalentes a un problema, que obtengan valores de aptitud similares. En ese caso, es probable que no haya una solución que se imponga en la población (y el criterio de terminación por convergencia nunca se cumpla). Este criterio no espera a que la población se componga mayoritariamente de una sola solución, sino que finaliza la ejecución del

algoritmo cuando los valores de aptitud de un determinado porcentaje de las soluciones son iguales, o difieren en un pequeño porcentaje. Por ejemplo, cuando el 90% de las soluciones tenga valores de aptitud que no difieran en más de un 1.

- **Criterio de cantidad de generaciones**

Los métodos anteriores apuntan a esperar a que la evolución de la población llegue a su fin, cuando alguno de ellos se cumple, es probable que las soluciones no sigan mejorando mucho más, no importa cuantas generaciones más se ejecuten. Sin embargo, los algoritmos genéticos pueden necesitar un número de generaciones muy grande para llegar a la convergencia, dependiendo de las tasas de recombinación y mutación.

Utilizando cualquiera de los dos criterios anteriores no puede estimarse un número máximo de generaciones, ya que esto dependerá no solamente de los parámetros del algoritmo genético sino también del azar, esto puede ser un problema, sobre todo si se quieren comparar los tiempos de resolución del algoritmo genético con otros métodos [Estivill-Castro, 2000].

El criterio de terminación por cantidad de generaciones consiste simplemente en finalizar la ejecución una vez que ha transcurrido un número determinado de generaciones. Este método permite determinar con precisión los tiempos de ejecución del algoritmo a costa de detener la evolución sin la certeza de que las soluciones no seguirán mejorando.

2.21.8. Estudio y análisis de los operadores de cruce

En el AG simple hemos visto el operador de cruce más básico, esto es, el operador de cruce basado en un punto. Este operador se puede generalizar fácilmente como se ha comentado anteriormente. Así una primera generalización de este operador es el operador de cruce en múltiple puntos. En este caso se eligen varios puntos dentro de los cromosomas y se intercambian las subcadenas entre dichos puntos. En un primer trabajo realizado por De Jong [De Jong, 1975] se demostró que el operador de cruce basado en dos puntos obtenía mejores resultados que operadores de cruce basado en más puntos.

El operador de cruce que ha recibido mucha atención en los últimos años es el operador de cruce uniforme [Chambers L, 2001], donde el cruce es realizado a través de una mascara. Un esquema del cruce uniforme puede verse en la figura 2.24.

La mascara de cruce está indicando el cromosoma del que se tomará cada gen para formar los hijos; de esta forma si en la máscara aparece un 0 tomará el gen correspondiente del primer individuo y si aparece un 1 realizará lo contrario.

0 1 0 1 1 1 0 1 1 0 1 1 0	**Mascara de Cruce**
1 1 1 0 1 0 0 1 0 0 1 0 0	**Padre 1**
0 1 0 1 0 1 0 0 0 0 1 1 0	**Padre 2**
1 1 **1** 1 0 1 **0** 0 0 **0** 1 1 **0**	**Hijo**

Figura 2.24: Ejemplo de Cruce Uniforme

Una manera de generalizar este operador de cruce es no generando la máscara de forma aleatoria, sino que esta dependa del valor de la función objetivo del par de individuos a cruzar. Así de esta forma, se posibilitaría que el individuo hijo tuviera más genes provenientes del padre mejor adaptado. [Larrañaga, Poza, 1994].

2.21.9. Análisis de los operadores más usuales de tipo permutación

Dado que el tipo de individuos utilizados en los trabajos previos de CNS es en todos los casos el mismo: *permutaciones* (de números), y en nuestro estudio lo orientaremos a documentos; es importante analizar con detalle los diferentes tipos de operadores que se han venido utilizando en los trabajos previos.

El tipo de operadores diseñados para cromosomas formados por permutaciones de números, proviene principalmente de la aplicación de los AG al conocido problema del viajante. El problema del viajante consiste en lo siguiente: Un viajante tiene que visitar *n* ciudades y volver a la ciudad inicial, se trata de hallar el recorrido que, pasando por todas las ciudades una sola vez, minimice la distancia recorrida por el viajante. Este problema es un problema *NP-completo***.**

Cualquier solución al problema, es decir, cualquier ruta a través de las *n* ciudades puede expresarse como una permutación $X: (\pi_1\ \pi_2\ \pi_3\ \ldots\ldots\ \pi_n)$, donde cada término π_i representa la ciudad que se visita en el lugar i-esimo.

Es posible representar las soluciones para este problema de forma diferente. Un resumen bastante completo de la aplicación de los AG al problema del agente viajero puede encontrarse en [Larrañaga et. al, 1999]. Es evidente que los operadores clásicos de los AG, analizados anteriormente no pueden ser aplicados directamente a este tipo de individuos, ya que podrían conducirnos a recorridos no válidos, se repetirían ciudades, o no se pasarían por todas las ciudades. Esto conlleva a la creación de operadores específicos para este tipo de cromosoma.

Para ilustrar los operadores de cruce de manera más conveniente, supondremos en todos los casos que el cruce es llevado a cabo entre los siguientes dos individuos:

Padre 1: (2 4 6 7 1 8 5 3 9)

Padre 2: (1 5 9 3 4 7 8 2 6)

2.21.9.1. Operador de cruce basado en una correspondencia parcial (PMX)

El operador de cruce PMX fue introducido por Goldberg [Goldberg, 1989]. Este operador toma dos individuos padres, y devuelve dos hijos. Se basa en seleccionar dos subrutas que se intercambian entre los padres, el resto de la información se intercambia a continuación.

Para llevar a cabo este cruce, dados dos individuos, la selección del trozo de ruta que se intercambia se realiza estableciendo dos puntos de corte. Los puntos de corte son elegidos arbitrariamente dentro de los posibles puntos. Si suponemos que estos dos puntos han sido el tercero y el séptimo en nuestros individuos de ejemplo; tenemos:

Padre 1: (2 4 6 | 7 1 8 5 | 3 9)

Padre 2: (1 5 9 | 3 4 7 8 | 2 6)

Para crear los hijos se intercambian los trozos de subrutas, de manera que la subruta del primer padre pasa al segundo descendiente y al revés:

Hijo 1: (* * * | 3 4 7 8 | * *)

Hijo 2: (* * * | 7 1 8 5 | * *)

Para terminar de generar las rutas se rellena cada descendiente con las ciudades restantes del padre correspondiente. Si existe una ciudad que ya se encuentra en el hijo, dicha ciudad se sustituye a partir de una correspondencia que se establece entre las subrutas elegidas. En este caso la correspondencia es: 7 y 3, 1 y 4, 8 y 7, 5 y 8.

De esta forma vamos a rellenar el primer hijo. Comenzamos poniendo la primera ciudad del padre, la 2, la siguiente sería la 4, pero, esta ciudad ya está puesta, luego buscando en la lista de correspondencias, terminamos poniendo la ciudad que se corresponde con la 4, es la 1, si dicho número hubiese estado también puesto, seguiríamos buscando dentro de la lista de correspondencias. Así proseguiríamos hasta el final; y los descendientes que se obtienen son:

Hijo 1: (2 1 6 | 3 4 7 8 | 5 9)

Hijo 2: (4 3 9 | 7 1 8 5 | 2 6)

2.21.9.2. Operador de cruce basado en ciclos (CX)

Este operador diseñado a partir de dos individuos padres crea un solo hijo. Se trata de rellenar las posiciones del cromosoma hijo de forma alternativa, con el valor que en dicha posición tiene uno de los padres teniendo en cuenta algunos ciclos. Si tomamos los padres del ejemplo:

Padre 1: (2 4 6 7 1 8 5 3 9)

Padre 2: (1 5 9 3 4 7 8 2 6)

Comenzaríamos rellenando la primera posición del cromosoma hijo eligiendo aleatoriamente uno de los padres para copiar su posición, supongamos que elegimos el segundo padre, tenemos entonces:

Hijo: (1 * * * * * * * *)

Una vez elegido el 1 en el segundo padre, esto significa que el uno no puede volver a ser elegido en el padre alternativo que continúa, luego en la quinta posición necesariamente se elegirá el 4.

Hijo: (1 * * * 4 * * * *)

Esto significa que en la segunda posición estamos obligados a elegir la ciudad del segundo individuo padre alternativo que continúa, es decir 5:

Hijo: (1 5 * * 4 * * * *)

Podríamos continuar rellenando el individuo hijo de esta forma y obtendríamos:

Hijo: (1 5 * 3 4 7 8 2 *)

Una vez completado un ciclo, la primera posición no rellena hasta el momento sería completada nuevamente eligiendo al azar entre ambos padres. Suponiendo que elijo en este caso el segundo, se obtiene como hijo:

Hijo: (1 5 6 3 4 7 8 2 9)

2.21.9.3. Operador de cruce basado en el orden (OX1)

En [Davis, 1985] se propone el siguiente operador de cruce. Los nuevos individuos son construidos utilizando una subruta de uno de los padres y el orden de los individuos del otro padre. Si consideramos que los individuos padres, una vez seleccionadas las subrutas,son:

Padre 1: (2 4 6 | 7 1 8 | 5 3 9)

Padre 2: (1 5 9 | 3 4 7 | 8 2 6)

Copiamos para empezar las subrutas en los individuos hijos:

Hijo 1: (* * * | 7 1 8 | * * *)

Hijo 2: (* * * | 3 4 7 | * * *)

Para completar los individuos hijos comenzamos eligiendo por ejemplo el primero, para completar el individuo se comienza tras el segundo punto de corte. Las ciudades son puestas, en el orden en que estas ciudades están en el segundo padre, a partir de, también, el segundo punto de corte. La primera ciudad que está, a partir del segundo punto de corte en el padre 2 es la 8, sin embargo, ésta ya está puesta en el Hijo 1, por lo que vamos a la siguiente ciudad, esta es la 2. Ésta si que se sitúa obteniendo:

Hijo 1: (* * * | 7 1 8 | 2 * *)

La siguiente ciudad es la 6, poniéndola en el octavo lugar del Hijo 1. El orden en el Padre 1 se continúa comenzando por el primer gen. Finalmente, obtengo los individuos:

Hijo 1: (9 3 4 | 7 1 8 | 2 6 5)

Hijo 2: (6 1 8 | 3 4 7 | 5 9 2)

2.21.9.4. Operador de cruce basado en la alternancia de posiciones (AP)

El operador de cruce basado en la alternancia de posiciones aparece en los trabajos de Larrañaga y col [Larrañaga et. al, 1999]. El operador AP toma dos individuos padres y devuelve dos hijos, la forma de conseguir cada uno de los hijos consiste en rellenar las posiciones de los mismos de manera alternativa, eligiendo primero la primera ciudad de un padre, luego la primera del otro padre, luego la segunda del primer padre, la segunda del segundo padre, y así sucesivamente, pero sin que se repitan. Teniendo en cuenta nuevamente los individuos del ejemplo:

Padre 1: (2 4 6 7 1 8 5 3 9)

Padre 2: (1 5 9 3 4 7 8 2 6)

Si comenzamos eligiendo primero del primer padre obtendríamos el hijo:

Hijo 1 : (2 1 4 5 6 9 7 3 8)

y eligiendo la primera ciudad del segundo individuo se obtendría:

Hijo 2: (1 2 5 4 9 6 3 7 8)

2.21.9.5. Operador de mutación basado en el desplazamiento (DM)

Este operador es debido a Michalewicz [Michalewicz, 1999]. Se basa en elegir un subconjunto del individuo, el cual se inserta de forma aleatoria en otro lugar del resto del individuo. Si consideramos el siguiente individuo:

(1 2 5 4 9 6 3 7 8)

Para poder elegir un subconjunto se eligen dos puntos, dentro del individuo de manera aleatoria, supongamos que son el segundo y el quinto:

(1 2 | 5 4 9 | 6 3 7 8)

Dicho subconjunto, queda delimitado por los puntos elegidos, y se introduce en otro lugar del individuo, éste lugar se determina eligiendo de forma aleatoria una posición dentro de las ciudades restantes. Si la población elegida es la cuarta:

(1 2 6 3 | 7 8)

Obtenemos el individuo:

(1 2 6 3 5 4 9 7 8)

2.21.9.6. Operador de mutación basado en cambios (EM)

El operador EM [Banzhaf, 1990] es el más obvio de todos. Consiste en seleccionar a azar dos ciudades de la permutación e intercambiarlas. Considerando nuevamente el individuo:

(1 2 5 4 9 6 3 7 8)

Si elegimos al azar la cuarta y la sexta posición para modificar, tras la mutación se obtiene el individuo:

(1 2 5 6 9 4 3 7 8)

2.21.9.7. Operador de mutación basado en la inserción (ISM)

El operador ISM fue propuesto por Fogel [Fogel D, 1988] y por Michalewicz [Michalewicz, 1999]. Este operador de mutación se basa en escoger aleatoriamente una ciudad de la permutación para a continuación insertarla en otro lugar elegido al azar. Por ejemplo si partimos de:

(1 2 5 4 9 6 3 7 8)

y elegimos al azar la ciudad 9, para más tarde elegir el cuarto lugar para situarla, obtenemos el individuo:

(1 2 5 9 4 6 3 7 8)

2.21.9.8. Operador de mutación basado en la inversión simple (SIM)

El operador SIM [Holland, 1975] realiza una selección de dos lugares en la permutación para luego invertir el orden de las ciudades que se encuentran entre los dos lugares elegidos. Si el individuo y los puntos elegidos son:

(1 2 5 | 4 9 6 3 | 7 8)

El individuo que se obtiene es:

(1 2 5 | 3 6 9 4 | 7 8)

2.21.9.9. Operador de mutación basado en la inversión (IVM)

Este operador es una mezcla del operador DM y del operador anterior. Se elige una subruta y luego esta subruta se inserta en un lugar elegido aleatoriamente, pero en orden contrario. Si el individuo es:

(1 2 5 | 4 9 6 3 | 7 8)

y la subruta elegida es la que viene acotada por | entonces eligiendo el primer lugar para insertarla la subruta, obtendríamos el individuo:

(1 3 6 9 4 2 5 7 8)

2.21.9.10. Operador de mutación basado en el cambio (SM)

El operador SM se basa en seleccionar una subruta dentro de cada individuo y cambiar de forma aleatoria el orden de las ciudades dentro de la subruta. Si el individuo y la subruta elegida son:

(1 2 5 | 4 9 6 3 | 7 8)

Entonces un posible individuo resultante después de la aplicación del operador de mutación es:

(1 2 5 | 9 6 3 4 | 7 8).

2.21.10. Estudio y análisis de las probabilidades a utilizar

Dentro de los parámetros que utiliza el AG se encuentran las probabilidades de cruce P_c y de mutación P_m. El trato dado por los investigadores a cada uno de estos parámetros ha sido muy diferente. Mientras que la probabilidad de cruce ha recibido poca atención, y casi siempre en aplicaciones prácticas suele ser cercanos a 1, se ha dedicado gran esfuerzo, tanto desde un punto de vista práctico como teórico a establecer el valor óptimo de la probabilidad de mutación. Este hecho puede estar motivado por algunos trabajos como el de [Schaffer et. al, 1989], donde se afirma que, establecer un valor óptimo para la probabilidad de mutación es mucho más importante para la búsqueda que hacerlo para la probabilidad de cruce.

De Jong [De Jong, 1975] en su tesis doctoral recomienda una probabilidad de mutación que sea inversamente proporcional al tamaño del individuo, es decir $1/L^{-1}$. En [Schaffer et. al, 1989] por su parte se propone un valor que depende de la longitud del individuo y del tamaño de la población. Estudios más recientes se pueden encontrar en Hessser y Manner [Hesser, Manner, 1990], Susuki [Suzuki, 1995], Bäck [Bäck, 1996] y Rudolph [Rudolph, 1994].

Hay también trabajos relacionados con la probabilidad de mutación que se basan en modificar ésta, a medida que se lleva a cabo la búsqueda: Ackley [Ackley, 1987], Fogarty [Fogarty, 1989], Los autores de estos trabajos se basan en que la probabilidad de mutación debería tener en cuenta la diversidad de la población.

A continuación estudiaremos el otro enfoque, la Programación Genética (PG) que surgió como una evolución de los AG, cuando John Koza [Koza, 1992] propuso denominar PG a la inducción genética de programas, siendo entonces ella una forma de computación evolutiva en la que los individuos de la población son programas, en lugar de cadenas de bits.; y en la que se emplean los conceptos y los operadores de cruce y mutación analizados anteriormente.

2.22. Programación Genética

En la lucha por sobrevivir, los individuos mejor adaptados sobreviven más fácilmente, por tanto pueden reproducirse y transferir a sus descendientes las cualidades beneficiosas que les permitieron estar mejor adaptados, lo cual constituye como ya se ha dicho, el proceso de selección natural postulado por Charles Darwin en su libro " *El origen de las especies "*. Todos aquellos rasgos que faciliten la supervivencia del individuo tenderán a mantenerse, mientras que lo que constituya una debilidad para la adaptación tenderá a desaparecer, pues su poseedor no tendrá oportunidad de legarlo a su descendencia.

La PG es una forma de Computación Evolutiva en la que los individuos de la población evolutiva son programas de ordenador en vez de cadenas de bits. Al igual que en los AG, la población se selecciona siguiendo el principio darwiniano de supervivencia del más fuerte. Los conceptos utilizados aquí son los mismos que en AG.

Generalmente, los programas manipulados en PG están representados por árboles de "*parsing"* del programa. Cada llamada a función se representaría por un nodo en el árbol, y los argumentos de la función vendrían dados por sus nodos descendientes.

En la figura 2.25 podemos ver un ejemplo de esto.

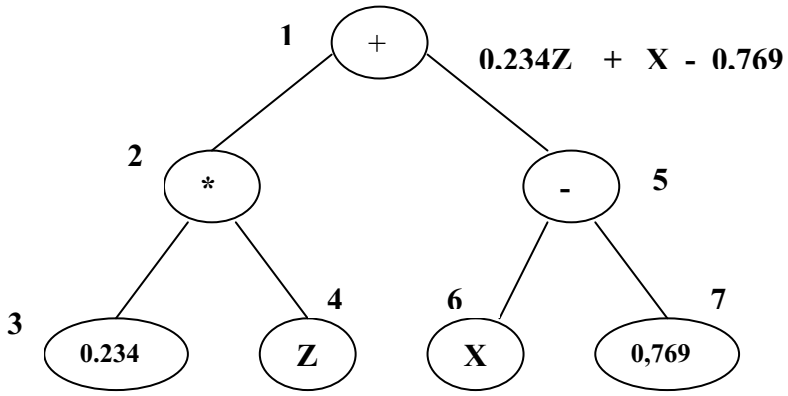

Figura 2.25: Ejemplo de representación de un programa en PG

En este caso, estamos representando para su posterior utilización con PG, por medio de un árbol la expresión:

$$0{,}234 * Z + X - 0{,}769$$

Para ello, deberemos definir previamente un conjunto de terminales y primitivas que nos permitan aplicar la búsqueda evolutiva sobre los programas que pueden ser descritos con dichos elementos.

De forma similar a los AG, un algoritmo de PG mantiene una población de individuos (programas representados por árboles), que en cada iteración producen una nueva generación por medio de selección, mutación y cruce. Las siguientes figuras muestran ejemplos de esta última operación genéticas aplicadas sobre programas.

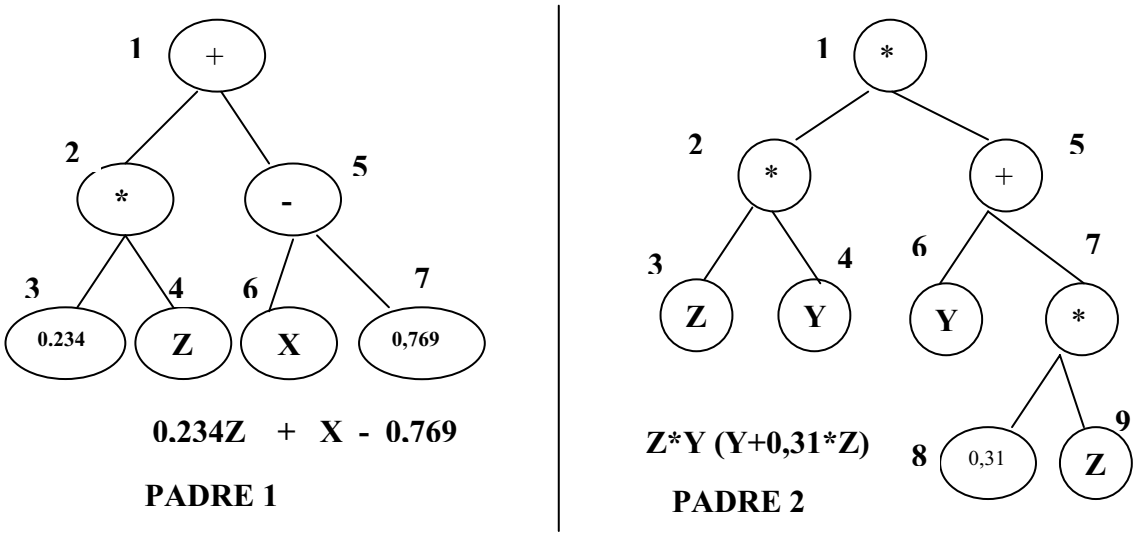

Figura 2.26: Ejemplo de dos árboles que van a ser los padres en el cruce con PG:

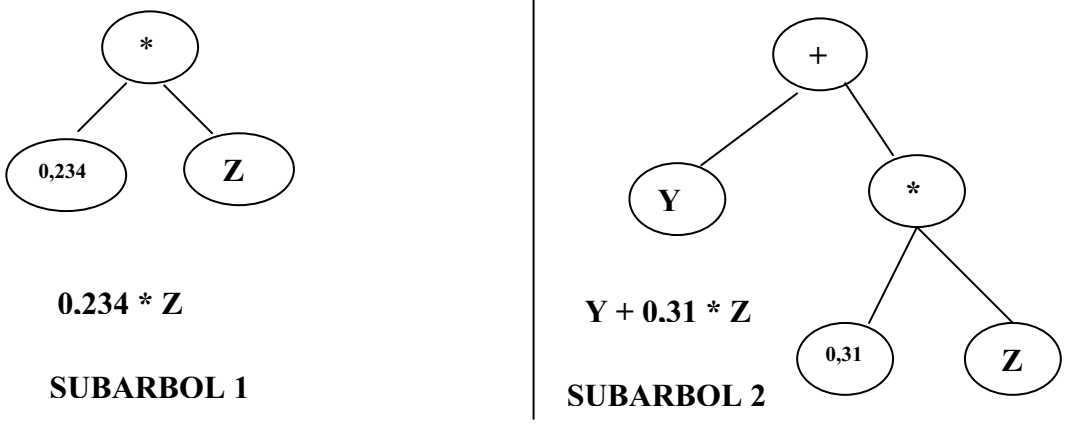

Figura 2.27: Vemos los subárboles que van a ser intercambiados en el cruce

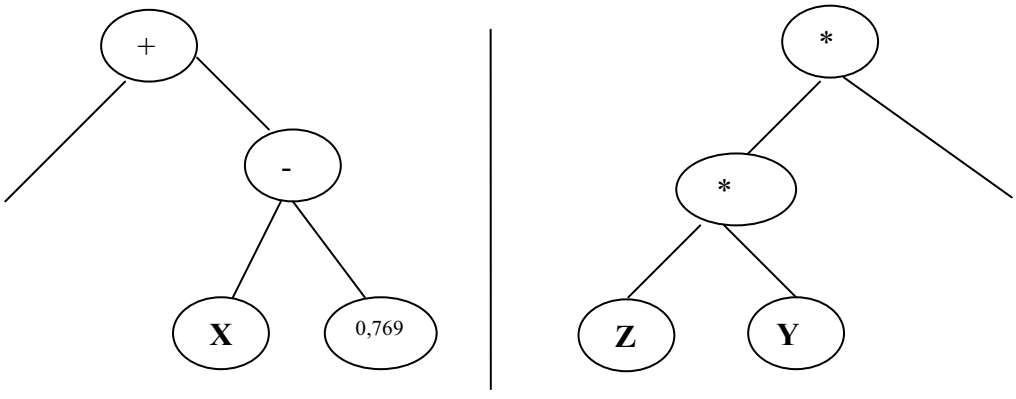

Figura 2.28: Vemos a los padres despojados de dichos subárboles para que sean intercambiados

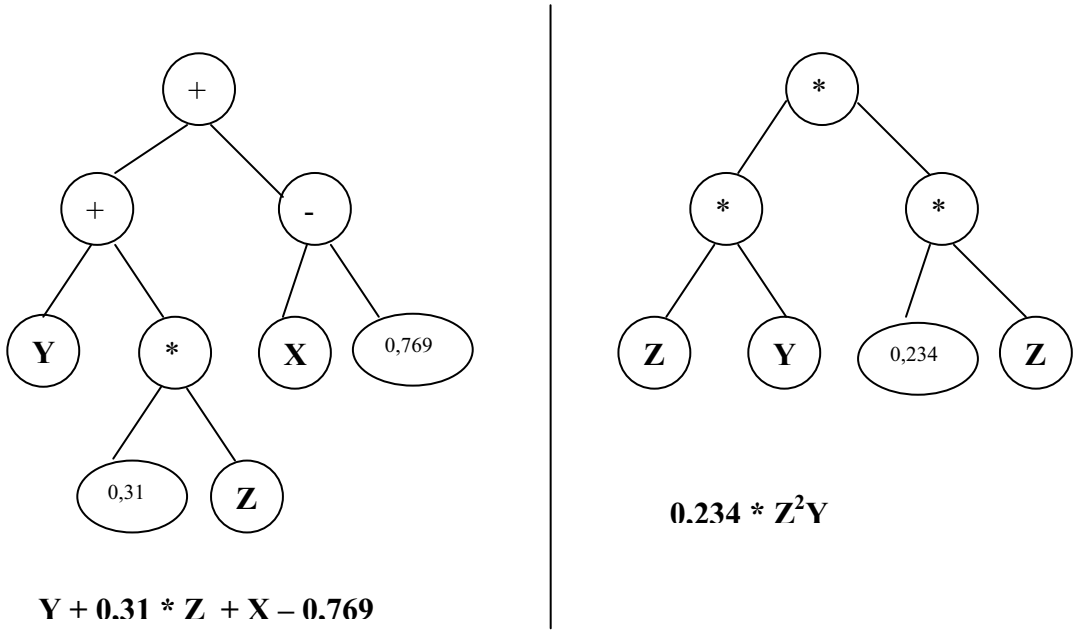

Figura 2.29: Ejemplo de los hijos generados con el intercambio de los subárboles

.

2.23. Requisitos para ejecutar un algoritmo de Programación Genética

Como ya se ha dicho, para la ejecución de un algoritmo de PG, es necesario definir previamente una serie de aspectos relativos al funcionamiento del propio algoritmo. Primero, definir el *conjunto de términos finales***,** esto es, las entradas del programa que debe resolver nuestro problema, y las *funciones primitivas*, que deben ser apropiadamente definidas: operadores aritméticos, lógicos, de programación, etc.

El conjunto de terminales más el conjunto de funciones primitivas deben cumplir el criterio de *suficiencia*, esto es, deben ser capaces de expresar la solución del problema. Por otro lado, cualquier función primitiva debe aceptar como argumento cualquier valor devuelto por cualquier función y cualquier valor que pueda ser usado como terminal. Esto es lo que se denomina *requisito de clausura*.

Debemos añadir una *medida de aptitud* que nos permita evaluar cómo de bien funciona cada programa de la población en el entorno del problema. Esta medida debe estar *completamente definida*, y debe ser capaz de evaluar cualquier programa de la población; así como los *parámetros de control de ejecución* que son varios, aunque básicamente se establecen el tamaño de la población inicial, el número de generaciones, el tanto por ciento de individuos que pasan directamente a la siguiente generación, etc.

Por último, el *criterio de finalización* debe indicar cuándo finalizar la ejecución del algoritmo de PG, lo cual puede ocurrir cuando transcurra el número de generaciones predeterminadas, o bien hallemos el programa que sea "perfecto" según la medida de aptitud que hemos proporcionado. La *designación del resultado* se hace generalmente

utilizando el criterio de "*mejor hasta el momento*", esto es, se elige el mejor programa de todos los obtenidos hasta la finalización de la ejecución.

La figura 2.30 muestra el diagrama de flujo que detalla el funcionamiento del paradigma de la PG.

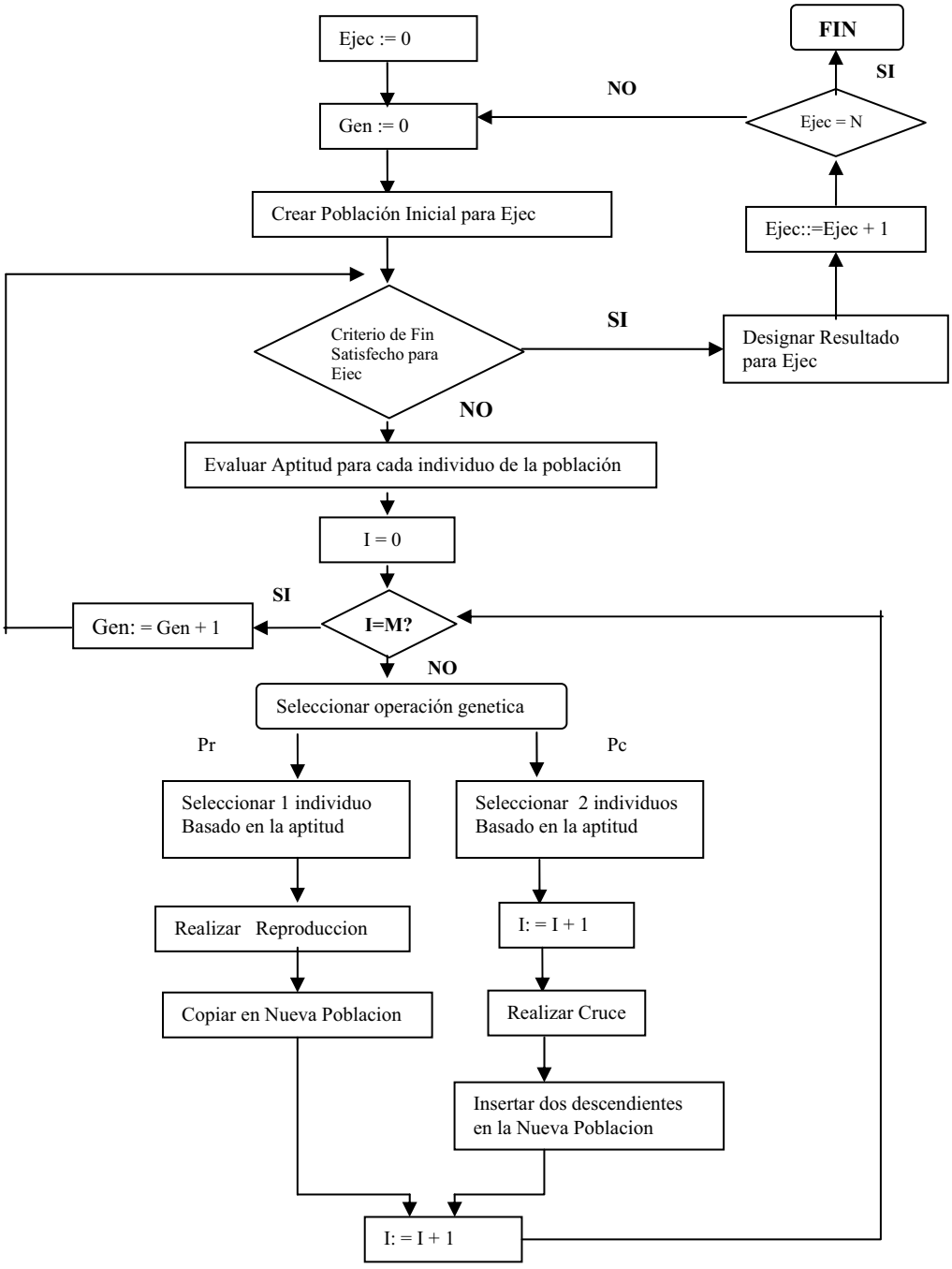

Figura 2.30: Diagrama de flujo del funcionamiento del paradigma de PG

2.24. Funciones definidas automáticamente (FDA)

Las Funciones Definidas Automáticamente, FDAs (en inglés, ADFs, "*Automatically Defined Functions*"), representan una evolución del concepto de lo que hemos denominado "*Programación Genética Clásica*".

La PG se "*limita*" a desarrollar programas secuenciales que combinan funciones primitivas y terminales para obtener un algoritmo que responda al problema planteado, las FDAs van más allá: además de generar esos programas, lo hace desarrollando funciones que evolucionan "*dinámicamente*" durante la ejecución del programa genético.

Así, éstas funciones en realidad no son sino nuevas formas de combinar los terminales y los nuevos operadores, pero con las ventajas que tiene la estructuración modular de un problema: reutilización, generalización, abstracción, menor coste computacional, etc. El estudio de esta evolución de la PG en [Koza, 1994] lleva a siete conclusiones principales sobre las FDAs:

1. Las FDAs permiten a la PG resolver un conjunto de problemas en una forma en que pueden ser interpretados como una descomposición del problema en subproblemas, una resolución y ensamblaje de los mismos, en una solución modular para el problema original. Se puede interpretar este proceso como una búsqueda de regularidades en el entorno del problema.
2. Las FDAs descubren y explotan las regularidades, simetrías, homogeneidades y modularidades de una forma muy distinta a la empleada por programadores humanos.
3. Para una gran variedad de problemas, la PG necesita menos tiempo de resolución usando FDAs que sin usarlas, siempre que la dificultad del problema sea superior a un cierto umbral característico del problema en cuanto a coste computacional.
4. Para una gran variedad de problemas, la PG con FDAs suele encontrar soluciones de menor tamaño que sin FDAs, siempre que la dificultad del problema sea superior a un cierto umbral.
5. Para los problemas estudiados, en que se incrementa progresivamente la dificultad de los mismos, el tamaño de las soluciones con FDAs se incrementa linealmente, mientras que sin FDAs lo hace exponencialmente.
6. Para los problemas estudiados en que se incrementa progresivamente la dificultad de los mismos, el esfuerzo computacional con FDAs se incrementa en menor medida que sin ellas.
7. La PG es capaz de, simultáneamente resolver un problema y evolucionar la estructura del programa.

2.25. Estructura de un programa con FDAs

La estructura de cualquier programa genético que use FDAs constará de una o varias ramas de definición de funciones y una rama de producción de resultados. Las ramas de definición de funciones contienen el nombre, la lista de argumentos y el cuerpo de la función que definen, mientras que la rama de producción de resultados

contiene únicamente el cuerpo del programa principal que llama a las funciones previamente definidas.

La figura 2.31 muestra un esquema de la estructura general de un programa genético con una sola FDA:

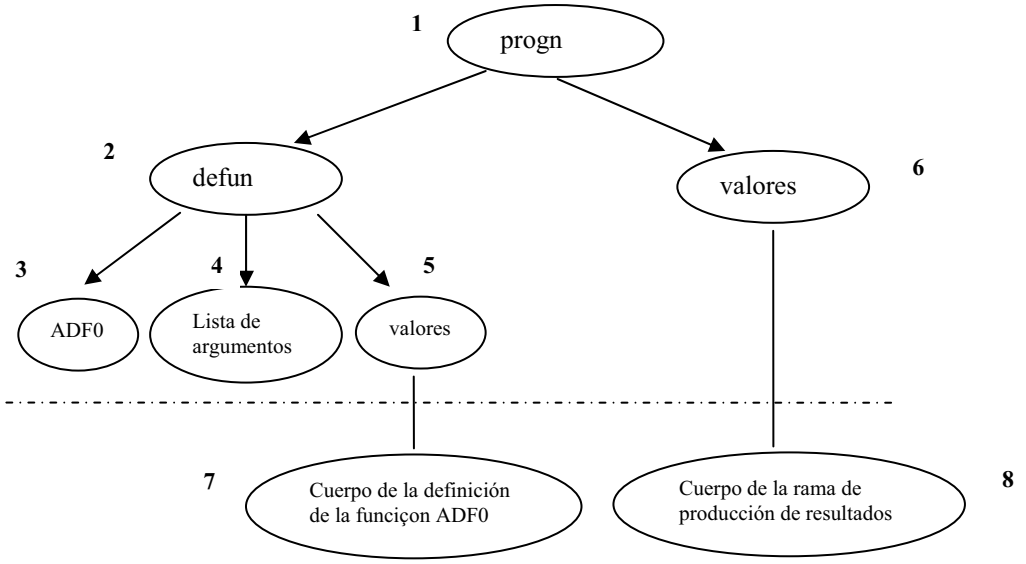

Figura 2.31: Esquema de la estructura general de un programa genético con una sola FDA

Los nodos que están por encima de la línea de puntos (nodos 1 a 6) son fijos, es decir, serán iguales para todos los programas generados genéticamente, mientras que los nodos que están por debajo de esta línea (nodos 7 y 8) son variables y variarán de individuo a individuo. De hecho, los nodos fijos no aparecen en la estructura interna de los programas genéticos, la cual consiste en una lista que agrupa a todos los nodos variables. Al aplicar el operador de cruce a programas con FDAs debemos preservar esta estructura general, y además los cruces deben producirse entre ramas del mismo tipo, es decir, se producirán cruces entre ramas de producción de resultados y entre ramas de definición de funciones con el mismo nombre.

Esto último se consigue asignando tipos a las distintas ramas, de forma que el cruce se produzca solamente entre ramas del mismo tipo. Esto se conoce como asignación de tipos a ramas (*branch typing*). En estos cruces, por tanto, primero se elige un punto de cruce del primer padre al azar y después se elige otro del segundo padre.

Las FDAs, al contrario que las funciones programadas por el ser humano, pueden mostrar un comportamiento aparentemente irracional, en el sentido de que pueden aparecer FDAs que realicen funciones que ya estén presentes en el conjunto de funciones del problema, FDAs que ignoren algunos de sus parámetros de entrada, FDAs que no sean llamadas en todo el programa, FDAs que simplemente devuelvan uno de sus argumentos o que devuelvan una constante, etc.

2.26. Aplicaciones de la Programación Genética

Koza [Koza,1991],[Koza,1992],[Koza,1994], [Koza, et al, 1999] y también [Angeline,1996] han recogido una gran cantidad de ejemplos de aplicación de la PG a problemas de aprendizaje (véase la tabla 2.1).

Tipo de Problema	Ejemplos
Control óptimo y modelos inversos	Centrado del péndulo invertido Problemas cinemáticos
Planificación de trayectoria	Hormiga artificial Robot que sigue una trayectoria
Aprendizaje de conceptos booleanos	Problema del multiplexador Funciones booleanas Sumador de dos bits
Regresión simbólica y descubrimiento empírico	Leyes de Kepler Modelos econométricos Series caóticas
Resolución de ecuaciones	Solución aproximada de ecuaciones diferenciales y ecuaciones funcionales Integración simbólica
Programación automática	Generador de números aleatorios Diseño de Redes Neuronales
Estrategias de juegos	Coevolución Evolución de estrategias de mercado
Clasificación e inducción de árboles de decisión	Inducción de clasificadores Extracción de características
Programación de autómatas celulares	Autómatas de una y dos dimensiones
Aprendizaje de consultas para SRI	Consultas booleanas y booleanas extendidas

Tabla 2.1: Ejemplos de aplicaciones de la PG según [Koza,1992] y otros

2.27. Componentes de un Sistema de Programación Genética

El diseño de un sistema de PG se basa en la elección de un lenguaje que permita definir las soluciones. Se emplean lenguajes muy sencillos, definidos específicamente para el problema que se desea resolver, aunque podría emplearse cualquier lenguaje e incluso el código máquina del procesador que se éste utilizando [Nordin, 1997].

```
Y -> EXPRESION

EXPRESION -> VARIABLE | CONSTANTE |
     + EXPRESION  EXPRESION |
     *  EXPRESION  EXPRESION

VARIABLE -> X-1 | X-2
CONSTANTE  -> número
```

```
En caso de una consulta Booleana sería:

     S -> EXPRESION
     EXPRESION  -> T (Terminos)  | EXPRESION Y EXPRESION |
                EXPRESION O EXPRESION | NO EXPRESION

     T ->  t1 | t2 | t3 …….| tn
```

En ambos casos, se construye un intérprete del lenguaje correspondiente, a partir del cual puede evaluarse un programa sobre un conjunto de casos de prueba. Cada programa se puntuará de acuerdo con su funcionamiento, y esta información se empleará en el AG para realizar la búsqueda de la cadena mas adecuada.

De la explicación anterior se deduce que en un sistema de PG intervienen los cuatro componentes siguientes:

- Un lenguaje L, definido mediante una gramática libre de contexto. La solución del problema será una cadena de este lenguaje.

- Un conjunto de casos de prueba para la tarea que se desea resolver. Por ejemplo, si se desea inducir una función de regresión sobre un conjunto de datos, se utilizará un conjunto de pares de valores de entrada y salida de éste.

- Una función de adaptación, que asociará cada cadena de **L** con un número (por ejemplo, el número de casos de prueba en que el programa funciona correctamente o, en nuestro caso, el número de documentos agrupados correctamente). Mediante la función de adaptación puede compararse soluciones del problema para decidir cuál de ellas es mejor. El valor de esta función se obtiene evaluando la solución en los casos de prueba, por lo que en la implementación de esta función está implícito un interprete del lenguaje **L**.

- Un AG capaz de hacer evolucionar una población de cadenas de **L** de acuerdo con la función de adaptación.

2.27.1. El esquema de representación de soluciones en PG

La estructura de un algoritmo de PG es la misma que la de un AG. En principio, la única característica diferenciadora es el esquema de representación de soluciones, que debe permitir representar cadenas del lenguaje.

Ahora bien, las definiciones de las operaciones de cruce y mutación han de estar especializadas para este tipo de representación. El cruce en uno o en dos puntos elegidos al azar, analizado anteriormente, no es directamente aplicable porque los descendientes no serían necesariamente cadenas del lenguaje, y lo mismo podría decirse de la mutación. Ambos problemas se resuelven utilizando una codificación intermedia, en forma de árbol, de los individuos. Si se define el cruce de dos cadenas mediante el intercambio de subárboles, y la mutación como el reemplazamiento de un subárbol por otro generado aleatoriamente, puede garantizarse que los descendientes resultantes de estas operaciones sean, a su vez, representaciones de cadenas del lenguaje.

En general, se empleará una de las siguientes codificaciones:

- Mediante los árboles sintácticos de las cadenas [Banzhaf et. al, 1998].
- Mediante sus árboles de análisis sintáctico [Geyer Schulz, 1995].

La representación que se emplea más frecuentemente en PG consiste en definir cada posible solución mediante un árbol y las reglas necesarias para evaluarlo, sin hacer referencia a un lenguaje libre de contexto [Banzhaf et. al, 1998]. Esta caracterización es un caso particular de una gramática.

En la figura 2.32 se muestra gráficamente, como ejemplo, la codificación de la cadena t1, t2, t3, t4 con respecto a la siguiente gramática:

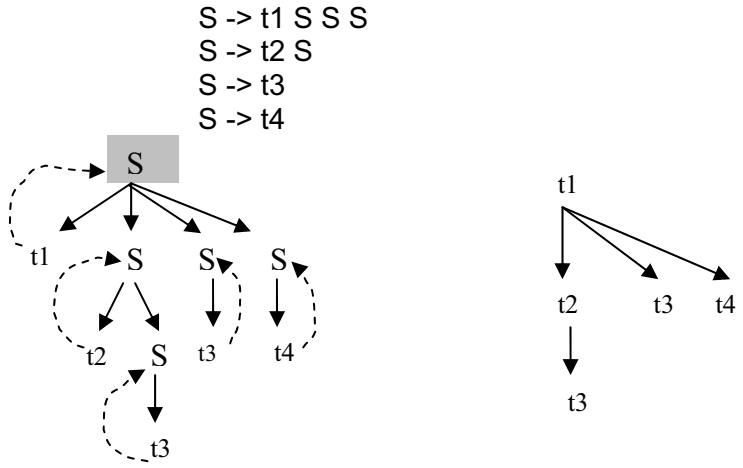

Figura 2.32: Relaciones entre los árboles de análisis y sintáctico en gramáticas no ambiguas y con un único símbolo no terminal

Por dar un ejemplo práctico, obsérvese que no hay diferencias entre definir un individuo como un árbol en el que los nodos terminales son las variables x1, x2 con números reales; y otra representación donde los nodos internos son los operadores suma y producto. Luego, se podrá definirlo como un árbol sintáctico de una de las cadenas de la gramática que sigue:

S -> x1 | x2 | número | + S S | * S S

La expresión (+ (* x1 x1) (* 0.37 (* x1 x2))) (los paréntesis han sido añadidos por claridad) se codificaría mediante el árbol de la figura 2.33, en el que todos los nodos del árbol son terminales de la gramática.

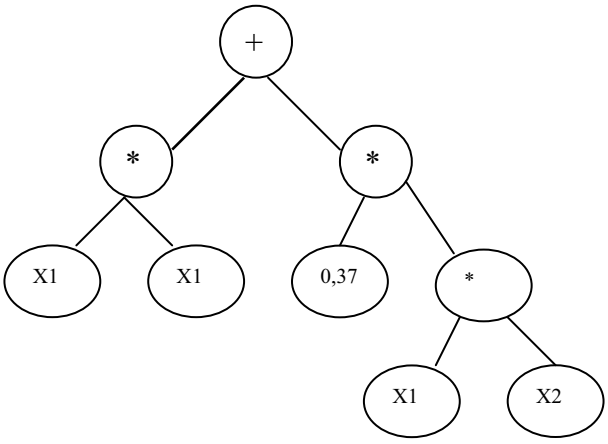

Figura 2.33: Ejemplo de Árbol sintáctico de la expresión: + * x1 x1 * 0,37 * x1 x2

2.27.2. La función de adaptación en PG

Con la representación basada en un árbol sintáctico, la evaluación recursiva de un individuo es inmediata. Se comienza por dar valores a los atributos de los nodos terminales del árbol sintáctico y se recorre el árbol, calculando los atributos de los nodos internos de acuerdo con las reglas semánticas correspondientes a su producción.

En el ejemplo anterior, cada nodo tendrá un atributo, al que llamaremos "valor". En los nodos terminales, éste tomará el valor de una de las variables de entrada o un número real, según la regla de producción, y en los restantes nodos el valor del atributo se calculará mediante las siguientes reglas (véase tabla 2.2):

Producción	**Regla Semántica**
S -> + S1 S2	S. Valor <- S1. valor + S2.valor
S -> * S1 S2	S. Valor <- S1. valor * S2.valor

Tabla 2.2: Reglas semánticas de producción del ejemplo de PG.

2.27.3. La generación de la población inicial en PG

La población inicial está compuesta por árboles sintácticos generados aleatoriamente. La construcción aleatoria de un árbol sintáctico se suele hacer de forma descendente, seleccionando al azar las producciones y aplicando las reglas semánticas definidas en cada caso para la construcción del individuo.

Siguiendo con el mismo ejemplo, se definirán en primer lugar las órdenes "*hazhoja*" y "*haznodo*". La orden "*hazhoja*" tiene como argumentos el tipo de nodo y el valor de su atributo, y la orden "*haznodo*" tiene como argumentos el tipo del nodo y las raíces de dos árboles.

Las reglas semánticas que generan la población inicial, serían (véase la tabla 2.3):

Num.	**Producción**	**Regla Semántica**
1	S -> X1	Hazhoja(x1,0)
2	S-> X2	Hazhoja(x2,0)
3	S-> número	Hazhoja(k,μ(-1,1))
4	S-> S1 + S2	Haznodo("+",S1,S2)
5	S-> S1 * S2	Haznodo("*", S1,S2)

Tabla 2.3: Reglas semánticas para generar la población inicial

Si el generador de números aleatorios generase la secuencia 4, 1, 5, 2, 1, se construirá el árbol sintáctico de la cadena +x1 * x2 x1

El valor del atributo en los nodos del tipo "x1" y "x2" es irrelevante, porque éste se cambiará cada vez que se evalúe el individuo. No ocurre lo mismo con el atributo de los nodos del tipo "*numero*", la expresión μ (-1,1) significa el valor aleatorio entre -1 y 1,

2.27.4. El operador de cruce en PG

La operación de cruce consiste en intercambiar subárboles entre dos árboles sintácticos (ver figura 2.34). Los dos subárboles se eligen de forma que la longitud final de las dos nuevas cadenas sea inferior al límite impuesto. Para seleccionar el punto de cruce, se elige aleatoriamente un arco del primer árbol.

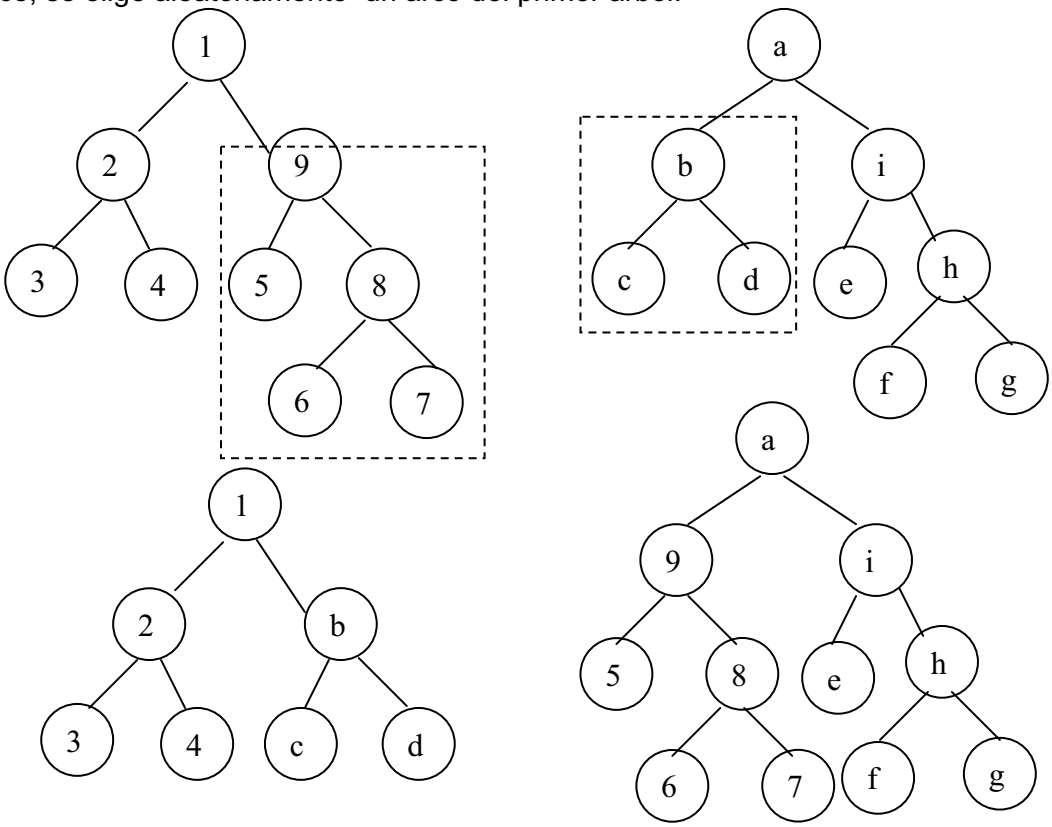

Figura 2.34: Cruce en PG, Los dos árboles padres intercambiar sus subárboles

2.27.5. El operador de mutación en PG

La operación de mutación consiste en reemplazar un subárbol por otro generado aleatoriamente de lo forma como se explicó anteriormente (ver figura 2.35) y de modo que la altura total del árbol obtenido no supere el máximo que se haya fijado.

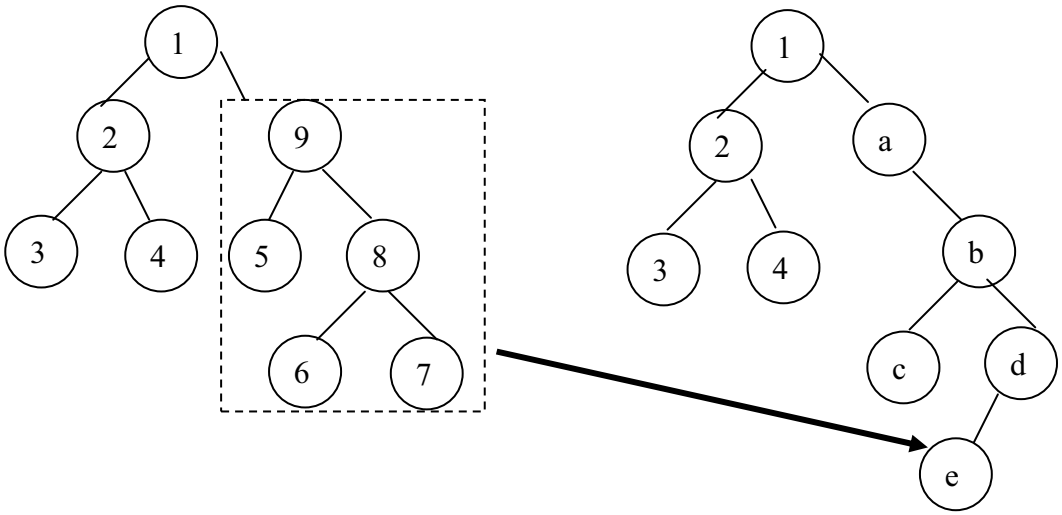

Figura 2.35: Mutación en PG: Consiste en reemplazar un subárbol por otro

Finalmente, luego de haber comentado en los apartados anteriores, todo el estado del arte referente a los Sistemas de Recuperación de Información y de los Algoritmos Evolutivos, concluiremos nuestra revisión del estado del arte comentando detalladamente en el siguiente apartado todas las aplicaciones estudiadas de los Algoritmos Evolutivos a los Sistemas de Recuperación de Información.

APARTADO III: APLICACIÓN DE LOS ALGORITMOS EVOLUTIVOS A LA RECUPERACION DE INFORMACION

Los AE constituyen una buena alternativa para resolver algunos de los problemas existentes en el ámbito de la RI. En este apartado se planteará y justificará la aplicación de esta familia de algoritmos en el agrupamiento de documentos y términos.

2.28. Algoritmos Evolutivos y Recuperación de Información

Como ya se ha dicho, en los últimos años se ha experimentado un interés creciente en la aplicación de técnicas basadas en la Inteligencia Artificial al campo de la RI con el propósito de resolver varios de los problemas considerados en el área. Así, el paradigma del Aprendizaje Automático [Mitchell Tom, 1997] basado en el diseño de sistemas que presenten la capacidad de adquirir conocimiento por si mismos, se ha aplicado al área de la RI [Cordon, Herrera-Viedma,Zarco, 2003].

Los AE no son específicamente algoritmos de aprendizaje automático, pero ofrecen una metodología de búsqueda potente e independiente del dominio que puede ser aplicada a gran cantidad de tareas de aprendizaje. De hecho, han sido muy utilizados en problemas de aprendizaje automático de reglas de producción a partir de conjuntos de ejemplos [Grefenstette, 1995],[Cordon, Herrera, Hoffman, Magdalena, 2001]. Este amplio uso se debe a que, cuando los individuos de la población genética representan conocimiento, pueden ser tratados al mismo tiempo como datos que son manipulados por el AG y como código ejecutables que lleva a cabo cierta tarea [Michalewicz, 1999].

Debido a estas razones, la aplicación de los AE a distintos campos de la ciencia se ha incrementado en los últimos años. Un claro ejemplo lo constituye el área de la RI, tal y como demuestra el gran número de publicaciones aparecidas en la literatura especializada. Entre otros, los AE se han aplicado en la resolución de los siguientes problemas dentro del marco de la RI [Cordon, Herrera-Viedma,Zarco, 2003].

- Agrupamiento (clustering) de documentos y términos [Gordon, 1991] [Robertson, Willet, 1994] y otras propuestas.
- Indización de documentos mediante el aprendizaje de los términos relevantes para describirlos comentado en [Zarco Carmen, 2003] y de sus pesos [Vrajitoru,1998].

- Mejoras en la definición de consultas: aprendizaje de los términos de la misma a partir de un conjunto de documentos relevantes para el usuario y de los pesos de los términos proporcionados previamente por el usuario [Yang, Korfhage,1994], [Robertson, Willet,1996], [Horng, Yeh, 2000], o de la composición de la consulta completa aprendiendo los términos y los operadores booleanos [Smith, Smith,1997], [Cordon, Herrera Viedma, Luque,2002], o los dos anteriores y los pesos de los términos [Kraft, et. al,1997].

- Aprendizaje de funciones de similitud para Sistemas de Recuperación de Información (SRI) basados en el modelo de espacio vectorial [Fan, Gordon, Pathak, 1999], [Fan, Gordon, Pathak, 2000], [Pathak, Gordon, Fan, 2000].

A continuación, realizaremos un breve estudio sobre el uso de los AE en el área de agrupamiento de documentos, analizando los enfoques que se han efectuado y comentando los resultados obtenidos, así como la problemática encontrada.

2.29. Algoritmos genéticos aplicados al agrupamiento particional

Como hemos visto en el primer apartado de este capítulo, los algoritmos de agrupación particional clásicos son algoritmos de optimización local cuyo resultado final depende de la partición inicial. En su mayoría necesitan de antemano el conocimiento del número de grupos. A todo lo anterior hay que añadir la imposibilidad de inspeccionar todo el espacio de búsqueda debido a su usualmente enorme tamaño.

Los problemas enumerados anteriormente han motivado la búsqueda de otro tipo de técnicas que pudieran subsanar las limitaciones de los algoritmos clásicos. Dado que el problema se puede considerar como de optimización en un espacio finito de gran tamaño, siendo en general *NP* completo, la utilización de las nuevas técnicas evolutivas de optimización provenientes de la IA se justifica plenamente.

Del paradigma de Algoritmos Evolutivos (AE) se observa que lo que más se ha aplicado en el área de agrupamiento es el uso de AG, habiendo casi muy poco de la aplicación de las otras técnicas de AE (PG, EE, etc) al área de agrupamiento. Se han aplicado al problema del agrupamiento particional otras técnicas de optimización combinatorias novedosa como el EE [Bäck, Fogel, Michalewicz 1997], o la búsqueda Tabú [Al-Sultan, 1995]; sin embargo, en todas estas aplicaciones se supone al igual que en los algoritmos clásicos que el número de grupos es conocido con antelación.

2.29.1. Revisión de aplicaciones de los AG al agrupamiento particional

Daremos a continuación un resumen de las diferentes aproximaciones que se han encontrado en la literatura de la aplicación de los AG al problema del agrupamiento particional. La literatura sobre este tema es bastante variada. A pesar de ello, hay bastantes aspectos que se repiten de unos artículos a otros, muchos de ellos, no tratan de encontrar los posibles grupos subyacentes a un conjunto de datos, sino más bien, de dividir un conjunto de datos en un número de grupos dado, de manera que se optimice una determinada función.

La primera cuestión que suele aparecer en el uso de los AG es la codificación de las posibles soluciones. Es decir, cómo vamos a codificar cada posible solución, de forma que estas representaciones puedan convertirse en individuos de nuestro AG, muchos de los problemas que se plantean a la hora de buscar una codificación en el ámbito del agrupamiento particional vienen expuestos en Bhuyan y col [Bhuyan et al, 1991].

Existen dos posibilidades; una de ellas consiste en tratar de buscar una codificación clásica, de manera que cada individuo se codifique como un vector binario (de ceros y unos) y de esta forma aprovechar los operadores de cruce y mutación tradicionales, los cuales han sido profundamente estudiados y analizados. El problema de esta aproximación radica en que la codificación y decodificación de cada individuo suele hacerse demasiado complicada.

La segunda opción para abordar el problema consiste en utilizar codificaciones no estándar, que se adecuen de manera simple a las soluciones del problema. La parte negativa de esta aproximación radica en la necesidad de diseñar operadores de cruce y mutación específicos, de forma que sean capaces de identificar trozos de los individuos asociados con una buena adaptación y puedan mezclarlos de manera conveniente. La elección entre ambas opciones está sin decidir, pues los artículos se reparten casi por igual entre ambas estrategias.

Para aclarar los diferentes conceptos, utilizaremos un pequeño ejemplo. Supondremos que se dispone de un conjunto de seis elementos:

{A, B, C, D, E, F}

Intentaremos dar un agrupamiento en tres grupos y usaremos en los ejemplos citados, los grupos: *{ A }, { B, E }, { C, D, F }.*

Describiremos los esquemas de representación y operadores que se han utilizado en los trabajos previos que aplicaron los algoritmos genéticos a problemas de agrupación automática. Dado que hay bastantes codificaciones que aparecen en más de un artículo, presentaremos un resumen de ellas, dividiéndolo en codificaciones binarias (0-1) y no binarias; comentando algunos de los artículos en particular. Para describir la mayoría de las codificaciones supondremos que tenemos un problema en el que se trata de dividir un conjunto de *"n"* objetos en *"k"* grupos diferentes.

2.29.1.1. Aproximaciones con codificaciones binarias (de ceros y unos)

Las codificaciones que han utilizado esta opción son:

- Bhuyan y col [Bhuyan et al, 1991] proponen una codificación basada en la teoría de grafos. Suponen que cada clasificación puede representarse mediante un grafo, donde cada objeto del conjunto de datos está representado por un vértice del grafo, y entre dos vértices existe una arista si los objetos se encuentran en la misma clase. Para los autores un individuo de la clasificación es un vector 0-1 de tamaño $\frac{n(n-1)}{2}$ (número máximo de aristas en un grafo no dirigido con *n* vértices), con cada gen representando una arista del grafo. Si un gen tiene un valor de 1 significa que dicha arista está en el grafo, y si no lo contrario. Así, la partición del ejemplo podría ser representada por el individuo:

  (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0)

- Alippi y Cucchiara [Alippi, Cucchiara, 1992] proponen una codificación de ceros y unos, en la que cada individuo es un vector de tamaño *"n por k"*. Cada individuo está dividido en *"k"* genes. Si el *j-esimo* bit del *i-esimo* gen toma un valor de *1*, esto significa que el *j-esimo objeto está en la i-esima clase*. Al mismo tiempo se tiene que dar que el *j-esimo bit* debe estar a *0* en los demás genes.

  Así con esta codificación nuestro ejemplo quedaría codificado como:

  (100000, 010010, 001101)

- Otra codificación 0-1 ha sido también propuesta por Cucchiara, comentada en [Cole, 1988], codificando cada partición mediante un vector de longitud :

  $$n * (| log_2 k | + 1).$$ (2.16)

  El individuo está dividido en *"n"* genes cada uno de longitud $[log_2 k] + 1$, cada uno de los cuales representa la clase a la que pertenece dicho objeto.

  La partición ejemplo se representaría como: *(00, 01, 10, 10, 01, 10*)

- Finalmente, para la representación del cromosoma se presenta una propuesta alternativa llamada representación matricial [Bezdeck et-al, 1994], explicada en [Cole, 1988], la cual consiste en un método que utiliza una matriz de *n x k* posiciones para representar un agrupamiento de *"n"* objetos y *"k"* grupos.

  Cada celda *i*, *j* de la matriz contiene el valor 1 si el elemento *"i"* se encuentra en el grupo *"j"*, o 0 en caso contrario. La partición del ejemplo se representaría de la siguiente forma:

1	0	0	0	0	0
0	1	0	0	1	0
0	0	1	1	0	1

  Así, si tenemos otra colección de 5 objetos, los siguientes cromosomas representan el agrupamiento que contiene los grupos {1, 2, 4} y {3,5} (véase tabla 2.4)
  :

1	1	0	1	0
0	0	1	0	1

Tabla 2.4: Representación matricial de un ejemplo de agrupamiento

Cabe mencionar, que en muchos casos, es imposible aplicar directamente los operadores clásicos para codificaciones de ceros y unos, pues su aplicación podría devolver agrupamientos con menor o mayor número de grupos del deseado, o devolver individuos sin sentido. Para resolver estos problemas, los autores diseñan operadores de cruce y mutación específicos para su codificación y/o modifican los individuos provenientes de la aplicación de los operadores clásicos.

2.29.1.2. Aproximaciones con codificaciones no binarias

Las codificaciones del tipo *no binaria* que se han propuesto para el problema de la agrupación particional son las siguientes:

- Bhuyan y col [Bhuyan et al, 1991] proponen una codificación donde cada individuo es un vector de longitud *"n"* y cada gen puede tomar un valor entre *1* y *"k"*. El valor que posee el individuo en el lugar *i-esimo* representa la clase a la que pertenece el objeto *i-esimo*.

  De esta forma, la partición de nuestro ejemplo estaría representada mediante el vector: *(1, 2, 3, 3, 2, 3)*.

  Esta misma codificación fue usada por Jones y Beltramo [Jones, Beltramo, 1991] llamándolo *"numeración de grupo"***.** De esta forma, el cromosoma representa un agrupamiento de *n* objetos con una cadena de *n* números enteros. En cada posición *i,* la cadena contiene el número de grupo en que se encuentra el objeto número *i*.

  Por ejemplo, para una colección de 5 objetos, el siguiente cromosoma representa un agrupamiento que contiene los grupos {1, 2, 4} y {3,5}

1	2	1	1	2

  En un agrupamiento de este tipo no importa el número de cada grupo, sino solamente cuáles objetos contiene cada uno, por lo que el siguiente cromosoma representa el mismo agrupamiento:

2	2	1	2	1

- También [Jones, Beltramo, 1991] propone una codificación especial para ciertos trabajos, en la que cada individuo es una permutación de los números: { 1, 2, ……n + k -1 }.

  Aquí, los números del *1* a *"n"* representan los objetos a agrupar y los números del *"n+1"* al *"n+k-1"* representan separadores entre clases. De esta forma nuestra partición ejemplo se representaría por medio del individuo:

  (1, 7, 2, 5, 8, 3, 4, 6)

  Donde los números *"7"* y *"8"* hacen de separadores entre diferentes particiones. Uno de los problemas de esta codificación es que los separadores no pueden aparecer en la posición inicial o final del individuo o dos juntos, ya que en este caso, tendríamos una clasificación con un número de clases menor que *"k"*.

  Este método llamado *"Permutación con separadores"* presenta un esquema en el que un agrupamiento de *"n"* objetos en *"k"* grupos se representa por una cadena de enteros en la que los valores de cada posición representan el número de objeto, y el grupo al que pertenece está dado por el orden en el que los objetos aparecen en la cadena. Los objetos que pertenecen al mismo grupo se encuentran juntos, y se utilizan los números *"(n + 1)"* al *"(n + k - 1)"***,** que no son válidos como números de objetos para indicar la separación entre los grupos.

Con esta representación, trabajando con la otra colección de 5 objetos de ejemplo, el siguiente cromosoma representa un agrupamiento que contiene los siguientes grupos: {1, 2, 4} y {3,5}

De esta forma, el agrupamiento del ejemplo anterior se podría codificar con cualquiera de los siguientes cromosomas (donde el número 6 actúa como separador):

1	2	4	**6**	3	5

3	5	**6**	1	2	4

4	1	2	**6**	3	5

En este trabajo, también se propone un operador de cruza basada en las aristas, este operador trabaja a nivel de agrupamiento, en lugar de hacerlo a nivel de cromosoma. Por lo tanto, no sufre el problema de la insensibilidad al contexto.

El algoritmo de cruza basada en las aristas sigue los siguientes pasos:

- Buscar las intersecciones no vacías de los grupos de cada padre. Estas intersecciones son los grupos del agrupamiento hijo.

- Juntar estas intersecciones al azar hasta que el hijo tenga *"k"* grupos.

Por ejemplo, si uno de los padres es

1	2	3	3	3	3	1

que contiene los grupos: { 1, 7 }, { 2 }, { 3, 4, 5, 6 }, y el otro es:

1	1	2	2	2	2	3

que contiene los grupos: *{ 1, 2 }, { 3, 4, 5, 6 }, { 7 }*

las intersecciones no vacías entre los grupos serán:

{ 1 }, { 2 }, { 3, 4, 5, 6 }, { 7 }

como son 4 intersecciones, y el hijo debe tener 3 grupos (al igual que los padres), deben juntarse 2 de las intersecciones. Puede verse que en ningún caso el grupo que es común entre los dos padres puede dividirse.

Si juntamos las intersecciones { 2 } y { 7 }, el hijo será:´

1	2	3	3	3	3	2

Sin embargo, si bien el operador es sensible al contexto, no aplica información específica del dominio para mejorar la calidad del hijo que está creando (no intenta juntar las intersecciones que contienen elementos más similares entre si).

- Finalmente, hay un método de permutaciones con búsquedas locales, que al igual que el método de permutación con separadores, utilizan una cadena de enteros, en el que cada valor representa el número de objeto, pero en este caso no se utiliza ningún separador. Un agrupamiento de *"n"* objetos en *"k"* grupos se representa mediante una cadena que contiene los *"n"* números de elementos ordenados adecuadamente. Cuando se utiliza esta codificación, los cromosomas no representan agrupamientos en forma directa, sino que debe emplearse algoritmos de búsqueda local para encontrar el agrupamiento representado. Dichos algoritmos deben encontrar un agrupamiento basado en un máximo local para algún criterio determinado.

  El más simple de estos algoritmos, denominado *"Permutación ambiciosa"* supone que los primeros *"k"* objetos son los centroides iniciales de los *"k"* grupos, y va asignando cada uno de los elementos siguientes al grupo con el centroide más cercano, igual a como funciona el algoritmo *k- means*.

2.29.1.3. Aproximaciones diferentes

Presentaremos una pequeña revisión de algunos artículos que difieren de los anteriores en aspectos importantes.

- Lucasius y col [Lucasius, Dane, Kateman, 1993] proponen una aproximación totalmente diferente al problema de la agrupación particional. Para construir una partición, los autores, se basan en las ideas de Kaufman y Rousseeuw [Kaufman y Rousseeuw, 1990] y buscan los *"k"* centroides mejores. Por lo tanto, convierten el problema de agrupación particional en uno de buscar un subconjunto óptimo de *"medoides"*. Para Lucasius y col [Lucasius, Dane, Kateman, 1993], un individuo es un vector de longitud *"k"***.** Cada gen toma un valor diferente entre los números { 1, 2, …n } que representa el centroide elegido. Por tanto, cada individuo se representa por un conjunto de centroides. El criterio de agrupación utiliza la distancia de Manhattan en vez de la Euclidea.

2.30. Algoritmos Evolutivos utilizados para el agrupamiento de documentos y términos

En lo que respecta a este segundo grupo, vamos a repasar dos enfoques distintos, el presentado por Robertson y Willet para el agrupamiento de términos equifrecuentes en una colección de documentos [Robertson, Willet, 1994], y la propuesta de Gordon para el agrupamiento de documentos cuyas descripciones están siendo adaptadas a lo largo del tiempo [Gordon, 1991].

2.30.1. Propuesta de Robertson y Willet

La idea básica que persigue el enfoque es formar grupos *"clusters"* de palabras de tal modo que la suma de las frecuencias de aparición de éstas en una colección de documentos sea aproximadamente igual en cada grupo. Este tipo de agrupamientos tienen varias aplicaciones interesantes, tales como la indización o la generación de signaturas.

El AG considerado realiza un agrupamiento por *división*, es decir, sin preservar el orden original de los términos; los autores proponen dos esquemas de representación distintos:

1. **Orden con separadores**: Cada individuo es un vector de dimensión *"N + n-1"*, donde "*N*" es el número de términos a agrupar y *"n"* el número de agrupamientos considerados. De este modo, los números enteros en el conjunto {1,2,….N} representan a los términos, y *"n-1"* genes notados con -1 son los separadores. Por ejemplo, con N=8 y n=4, el siguiente cromosoma:

   C = (1, 2, 3, -1, 4, 5, -1, 6, -1, 7, 8)

   representa los clusters:

   G1 = {1, 2, 3}, G2={4, 5}, G3={6}, G4= {7, 8}

   Puesto que con esta representación se pueden obtener individuos sin sentido cuando dos separadores están en posiciones consecutivas (lo que da lugar a que un cluster quede vacío), los autores proponen un mecanismo para solventar este problema.

2. **División-Asignación:** Los cromosomas son vectores de dimensión *"N"*. Cada posición esta asociada a un término, y el valor de la misma representa el cluster al que pertenece el término en cuestión. Así, con los valores de *"N"* y *"n"* anteriores, el cromosoma:

   C = (3, 2, 1, 4, 3, 3, 2, 1)

   representa la configuración de clusters:

   G1= {3, 8}, G2={2, 7}, G3={1, 5, 6}, G4= {4}

   La generación de la población inicial depende del esquema de representación.

En el primer caso, se construye el primer individuo, C1, tomando los términos en el orden en que aparecen en el fichero de frecuencias e insertando los separadores aleatoriamente o en las posiciones que hacen que todos los grupos tengan el mismo tamaño.

En el segundo caso, este primer individuo se construye asignando los *"n"* primeros términos, uno a cada grupo, y asociando sucesivamente los *"N-n"* restantes al grupo con menor suma de frecuencias. En ambos casos, el resto de individuos de la población son variaciones aleatorias del primero.

Por otro lado, el mecanismo de selección del AG no es habitual, cada operador tiene una probabilidad de aplicación asociada, y en cada paso, se escoge el operador a aplicar lanzando una ruleta. Una vez que se conoce éste, se escogen los padres necesarios para su aplicación. En el caso de la mutación, el único padre necesario se escoge girando la ruleta sobre la población. En cambio, para el cruce, el primer operador se escoge siempre a partir de la ruleta, pero el segundo puede escogerse bien a partir de ella o bien aleatoriamente dando la misma probabilidad de selección a todos los cromosomas.

En lo que respecta a la composición de los operadores en sí, el AG trabaja con dos operadores distintos de mutación, cada uno con una probabilidad de 0,25. En todos los casos, se escogen dos posiciones aleatorias y se efectúa algún cambio entre ellas.

Por ejemplo, dado el siguiente cromosoma:

C = (1, 2, 3 | 4 ,5 ,6 ,7 | 8)

Inversión: *C' = (1, 2, 3 | 7, 6, 5, 4 | 8)* Inversión de todos los genes.

Sublista aleatoria: *C'= (1, 2, 3 | 6, 4, 5, 7 | 8)* Selección aleatoria de los genes.

Posición: *C'= (1, 2, 3, | 7, 5, 6, 4 | 8)* Intercambio de los dos genes.

Los operadores de cruce también están basados en la representación de orden.

Para la primera representación, se escoge uno de los dos siguientes, que recibe una probabilidad de 0,5 en el algoritmo.

a) **Cruce ordenado**: Cada padre se asocia directamente a un hijo, y se genera aleatoriamente un patrón binario con la misma longitud que el cromosoma.

   Dicho patrón determina qué hijo recibe el gen del padre correspondiente en esa posición, el primero (valor 1 en el patrón) o el segundo (valor 0). Posteriormente, los huecos se rellenan con los genes restantes del padre asociado en el orden en que aparecen en el otro padre.

   Por ejemplo:

P = (0, 1, 1, 0, 1, 1, 0 ,0)	Paso 1	Paso 2
C1 = (1, 2, 3, 4, 5, 6, 7, 8)	*C1 ' = (- 2 3 – 5 6 - -)*	*C1'=(8, 2, 3, 4, 5, 6, 7, 1)*
C2 = (8, 6, 4, 2, 7, 5, 3, 1)	*C1' = (8 - - 2 - - 3 1)*	*C2'=(8, 4, 5, 2, 6, 7, 3, 1)*

b) **Cruce basado en posición**. Similar al anterior. En este caso, cuando el patrón muestra un 1, se copia el gen del padre 1 al hijo 2 y viceversa. Ante un 0, no se hace nada. Posteriormente, se rellenan los huecos del hijo con los elementos del otro padre, en el orden en que ocurren en el suyo.

Así, en el caso anterior, los descendientes serían:

P = (0, 1, 1, 0, 1, 1, 0, 0)	*Paso 1*	*Paso 2*
C1 = (1, 2, 3, 4, 5, 6, 8)	*C1' = (- 6 4 – 7 5 - -)*	*C1'= (1, 6, 4, 2, 7, 5, 3, 8)*
C2 = (8, 6, 4, 2, 7, 5, 3, 1)	*C2' = (- 2 3 – 5 6 - -)*	*C2'= (8, 2, 3, 4, 5, 6, 7, 1)*

Sin embargo, para la representación de división-asignación, los autores consideran el cruce simple en un punto y en dos puntos con los dos operadores anteriores. Para la función de adaptación, también se presentan dos propuestas: una medida de la entropía relativa, $H_R \in (0,1]$, donde el valor 1 muestra la equifrecuencia total en los clusters, y la medida de Pratt, $C \in [0,1]$, donde el valor optimo es el 0. De este modo, el AG afronta un problema de maximización en el primer caso y uno de minimización en el segundo.

Finalmente, en cuanto a los experimentos realizados, los autores trabajan con cinco conjuntos de datos distintos, cuatro de ellos procedentes de distintas colecciones de documentos en ingles y en turco, el último generado experimentalmente según una distribución de Zipf.

2.30.2. Propuesta de Gordon

La otra aplicación incluida en este grupo es bastante distinta a la anterior. En [Gordon, 1991] se analiza un problema frecuente cuando se trabaja con procesos de clustering en RI. Habitualmente, las técnicas de clustering se basan en patrones de similitud en las descripciones de los documentos, lo que da lugar a que documentos que no son relevantes para las mismas consultas pero que presentan descripciones parecidas se agrupen en los mismos clusters, en detrimento de la precisión del sistema.

Existen una serie de algoritmos de clustering que solucionan este problema en el campo de la RI, las denominadas técnicas de agrupamiento orientadas al usuario *("user-based clustering")*. En este caso, los documentos se agrupan por su relevancia y no solamente por su descripción, es decir, se busca formar grupos de documentos relevantes para las mismas consultas.

En realidad, en este trabajo, el autor [Gordon,1991], no propone un algoritmo concreto nuevo dentro de esta familia sino que propone como enfoque realizar un agrupamiento orientado al usuario haciendo uso de cualquier técnica clásica de clustering que agrupe los documentos por similitud de las descripciones.

La idea básica para ello es que el sistema adapte las descripciones de los documentos a lo largo del tiempo, de modo que documentos relevantes para las mismas consultas presenten descripciones similares y vaya realizando periódicamente un proceso de agrupamiento sobre las nuevas descripciones adaptadas. Para esta tarea, el autor propone un AG que está basado en asociar varias representaciones a cada documento.

De este modo, los experimentos que realiza en el trabajo, van orientados a demostrar la eficacia de la nueva propuesta. Para ello, Gordon trabaja con una base de datos documental preparada de antemano, e induce una serie de clusters que representan distintas materias a partir de las descripciones iniciales de los documentos. El objetivo perseguido es demostrar que, gracias al proceso adaptativo, las representaciones de documentos con relevancia similar, estarán más próximas en el espacio, facilitando el agrupamiento con técnicas clásicas de clustering.

En dichos experimentos, considera un número variable de documentos por cluster (4 y 6). Además, define un vecindario de documentos no relevantes para cada cluster. Para estudiar la calidad de los resultados, Gordon emplea una medida de la densidad relativa del cluster, resultante de dividir la varianza de las distancias de los documentos contenidos en el mismo entre la varianza de las distancias entre ellos y sus vecinos no relevantes.

Tras revisar en este capítulo el estado de arte relacionado con nuestra investigación, pasaremos en los siguientes capítulos a desarrollar la parte experimental de este trabajo.

Capítulo 3

Metodología de procesamiento de documentos

El objetivo de este capítulo es ofrecer la descripción de un marco de trabajo genérico que se utilizará en la experimentación para obtener los vectores característicos que sean representativos de la colección documental. La meta final de la tesis es validar el sistema evolutivo con dichos vectores y analizar su aplicabilidad. Para ello, describiremos las colecciones documentales y los clusters utilizados, las etapas seguidas para la indización de las mismas, las condiciones de experimentación que se han realizado, y detallaremos el *método de procesamiento* propuesto para obtener los vectores característicos de los documentos.

De esta forma, describiremos las pruebas de evaluación de la efectividad de la solución propuesta y el conjunto de datos utilizados, los cuáles, para que seán representativos, han sido extraídos de la colección Reuters 21578, reconocida como estándar dentro de la comunidad de investigadores dedicados a la categorización automática de documentos. Asimismo, describiremos una colección documental con documentos en español, preparada a partir de todos los editoriales del diario El Mundo del 2006 y 2007.

Detallaremos paso a paso la metodología seguida en la experimentación, enumerando las variables que intervienen en los experimentos y los distintos tipos de pruebas realizadas para obtener los vectores documentales. Finalmente, validaremos experimentalmente nuestra metodología de procesamiento comparándola con otros enfoques.

Antecedentes

A mediados de la década de los 60, tras una serie de pruebas preliminares que constituyeron el proyecto denominado Cranfield I, un grupo de investigadores de College of Aeronatics de Cranfield en el Reino Unido dirigidos por Cleverdon realizó una batería de pruebas para evaluar la calidad de treinta y tres sistemas de indización distintos en el proceso de RI [Cleverdon, 1972], considerando directamente la relevancia como medida de eficacia del SRI.

Para ello, diseñaron una colección compuesta por 1398 documentos sobre Aeronautica y compusieron una batería de 211 consultas generadas por autores de documentos seleccionados de la colección. Antes de realizar ninguna búsqueda, determinaron los documentos relevantes para cada consulta empleando un proceso en el que intervinieron, en primer lugar, estudiantes de Aeronáutica de Cranfield, y luego los propios autores de las consultas. De este modo, el experimento presentaba tres componentes principales:

- La colección de documentos de prueba.
- El conjunto de consultas.
- El conjunto de juicios de relevancia (valoraciones binarias que indicaban si el documento era juzgado como relevante para cada consulta)

Cada vez que se ejecutaba una consulta en el sistema, se determinaban los siguientes conjuntos de documentos:

a: Conjunto de documentos relevantes recuperados

b: Conjunto de documentos no relevantes recuperados.

c: Conjunto de documentos relevantes no recuperados.

d. Conjunto de documentos no relevantes no recuperados.

De este modo, los juicios de relevancia asociados a cada consulta se emplearon para medir el rendimiento de los mecanismos de indización considerados, haciendo uso por primera vez de los conceptos clásicos de precisión y exhaustividad (que definimos en la ecuación 2.7 y 2.8). De esta forma, ambas medidas se obtenían a partir de los conjuntos de documentos mencionados de la siguiente forma:

$$Precisión = \frac{a}{a+b}; \qquad Exhaustividad = \frac{a}{a+c}$$

Este experimento se conoció como Cranfield II, y sentó las bases de la evaluación algorítmica de SRI, pero el modelo tradicional de evaluación ha sido muy criticado, principalmente por las razones siguientes:

- La dificultad de obtención de los juicios de relevancia asociados a los documentos de la colección.
- El priorístico escalado de los resultados obtenidos en colecciones de pequeño tamaño a las colecciones documentales reales, de gran tamaño.
- El hecho de que las representaciones de los documentos se obtengan a partir de resúmenes de los mismos y no provengan del texto completo.

Sin embargo, a pesar de las dificultades a nivel teórico - práctico relacionadas con las medidas de exhaustividad y precisión, éstas ofrecen, al menos, características aceptables y homogéneas, y gozan de suficientes referentes como para permitir comparar el funcionamiento de los SRIs.

El problema de la escalabilidad de los resultados a grandes colecciones documentales, y la indización de documentos a texto completo va quedando progresivamente atenuado al aumentar el tamaño de las colecciones de test consideradas. Un ejemplo claro lo constituyen las colecciones TREC (Text Retrieval Conference) provenientes de una iniciativa liderada por Donna Harman a principios de los noventa en el NIST (Nacional Institute of Standards and Technology), en Maryland. La iniciativa consistió en el desarrollo desde 1992, de un congreso anual, denominado TREC (http://trec.nist.gov/) dedicado a la experimentación con grandes colecciones de prueba.

Sin embargo, el manejo de las colecciones TREC requieren una gran cantidad de recursos, lo que dificulta su manejo. Las consultas de TREC presentan entre ellas una superposición de resultados muy pequeña y no son muy útiles para investigar el impacto de las técnicas que se benefician de la información obtenida entre la consulta actual y las consultas pasadas del usuario. De esta forma, existe una gran cantidad de investigadores en el área que trabajan con colecciones de tamaño *medio* para las pruebas experimentales en el campo de los SRI. Así por ejemplo se tienen las siguientes colecciones (todas en inglés):

- ADI: documentos sobre Ciencias de la Información.
- CACM: artículos publicados en Communication of the ACM.
- INSPEC: resúmenes de electrónica, informática y física.
- REUTERS 21578: Noticias reales de la Agencia Reuters

3.1. La colección de documentos objeto del estudio

En este trabajo haremos uso de dos colecciones: la colección Reuters 21578, y debido a la falta de colecciones experimentales en español una colección documental en español que preparamos a partir de todos los *editoriales* del diario El Mundo del 2006 y 2007.

Hemos elegido la colección documental Reuters por los siguientes motivos:

- Para aplicar la metodología de procesamiento de documentos propuesta en este trabajo, a una colección clásica en el campo de la RI. Así de esta forma, podemos validar experimentalmente la calidad de los vectores característicos obtenidos en una colección real.
- Es una colección documental medianamente grande (compuesta por 21578 documentos), con muchas distribuciones variadas que permiten probar los resultados con colecciones documentales reales, de gran tamaño.
- Los documentos se encuentran en formato de texto completo.

3.1.1. La colección documental Reuters

La colección documental Reuters está conformada por noticias reales que aparecieron en cables de la agencia Reuters durante 1987. Los documentos fueron recopilados y categorizados manualmente por personal de la agencia y de la compañía Carnegie Group, Inc, en 1987. En 1990, la agencia entregó los documentos al Laboratorio de Recuperación de Información (*Information Retrieval Laboratory*) de la Universidad de Massachussets.

La colección se distribuyó bajo la denominación “Reuters 22173” desde 1991 hasta 1996. Es en ese año durante la conferencia ACM SIGIR un grupo de investigadores realizó un trabajo sobre esta colección con el objetivo de que los resultados de distintos trabajos que utilizaran la colección fueran más fáciles de comparar entre si. El resultado fue la distribución 21578, que es la que actualmente se utiliza en muchos trabajos de RI y específicamente en los de categorización automática de documentos con el fin de asegurar una metodología de prueba uniforme.

La colección se compone de 21578 documentos (cantidad que le da nombre a la misma), distribuidos en 22 archivos. Cada uno de estos archivos trata el mismo asunto, sin embargo cada documento tiene 5 posibles campos de categorización distintos: *Valor Bursátil, Organización, Persona, Lugar y Tema*. En cada campo, el documento puede tener un solo valor, varios o ninguno.

Esta colección de documentos ”Reuters 21578” [Lewis, 1997] se ha convertido en un estándar *de facto* dentro del dominio de la categorización automática de documentos, y es utilizada por numerosos autores de la materia [Steinbach, Karypis, Kumar, 2000],[Zhao, Karypis, 2001]. Por ello, parte de las pruebas experimentales de esta tesis, se han ejecutado utilizando esta colección.

En la figura 3.1 vemos un ejemplo de uno de los documentos de la colección.

Northeast Savings F.A.said its board adopted a shareholder rights plan designed to protect the company from coercive takeover tactics and bids not fair to all sharholders.

Under the plan, the board declared a dividend of one share purchase right for each of the Northeast common shares held of record as of November two, the company said.

Initially, the rights are not exerciseable, rights certificates are not distributed, and the rights automatically trade with Northeast's shares, the company said.

However, 20 days following the acquisition of 20 pct or more of Northeast's common shares or 20 days following the commencement of a tender offer for 30 pct or more of Northeast’s shares, the rights will become exerciseable and separate rights certificates will be distributed, the company said.

The rights will entitle holders of Northeast's common shares to purchase additional shares at an exercise price of 60 dlrs a share, the company said.

The company said that in the event of certain triggering events described in the rights plan, holders of the rights, other than an acquiring person, will be entitled to acquire Northeast's common shares having a market value of twice the then current exercise price of the rights. Also, in the event Northeast enters into certain business combination transactions, holders of the rights will be provided a right to acquire equity securities of the acquiring entity having a market value of twice the then-current exercise price of the rights, the company said. Northeast said it will be entitled to redeem the rights at one cent per right until the occurence of certain events.
Reuter

Figura 3.1: Uno de los documentos de la colección Reuters, luego de haberse filtrado las etiquetas y metadatos, en él se puede apreciar el contenido de la noticia (inglés)

Además se han atendido otros criterios de caracter metodológico que también llevan a utilizar esta colección documental en las pruebas experimentales, y las razones que justifican su uso fueron:

- La colección Reuters 21578 es una colección documental estándar para la prueba de algoritmos en el campo de los SRI, y específicamente en la categorización, por lo que los resultados obtenidos tendrán mucha más validez que si los experimentos se realizaran sobre un conjunto de datos recopilado sin seguir una metodología estándar.
- Debido a que la colección Reuters 21578 está distribuida en 22 archivos en formato SGML, el método utilizado para realizar los experimentos fue procesar cada una de las distribuciones de Reuters por separado, con el fin de poder analizar mejor los resultados del modelo de procesamiento en cada una de las distribuciones.
- Inicialmente, al subconjunto extraído de esta colección se le realiza un filtrado de aquellos documentos sin clasificar, o con más de una clasificación (que complican innecesariamente la evaluación de los resultados) a fin de poder posteriormente comparar y manipular los documentos de toda la colección con los métodos propuestos en el presente trabajo.

3.1.2. La colección documental en español

La colección documental en español, que se obtiene a partir de los editoriales del diario español El Mundo, publicados entre el 1 de enero de 2006 y el 31 de enero del 2007 tiene las siguientes características: se trata de documentos que tienen solamente titulo y contenido, y que carecen de cualquier otro tipo de metadato, como serian: autor, palabra clave, etc., exceptuando la fecha y el origen de la publicación. Dado el volumen de documentos procesados (en total 1402 editoriales), y debido a que están en español, esta colección documental constituye una buena colección experimental para validar las propuestas planteadas en esta tesis.

Como inconveniente, los documentos no estaban previamente caracterizados, por lo que tuvo que realizarse una clasificación manual de estos, con la finalidad de que los resultados que obtengamos nos sirvan para comparar las pruebas experimentales que realicemos más adelante con el AG.

Para poder llevar a cabo las medidas de aciertos de los diferentes algoritmos utilizados, se confeccionó un formulario de toma de datos, para que varias personas de formación universitaria, realizasen una clasificación manual y una selección de las frases más significativas en los documentos, sobre las que establecer las comparativas. Como tesauro para la clasificación se ha elegido el tesauro *Eurovoc* [Eurovoc, 2006], promovido por la Unión Europea, porque se trata de un tesauro plurilingüe que abarca todos los ámbitos de actividad de las Comunidades Europeas, y además se utiliza en los sistemas de documentación institucionales.

Un ejemplo de uno de los documentos contenidos en la colección en español, se muestra en la figura 3.2.

EDITORIAL

LOS PRECIOS SUBEN, LA COMPETENCIA DISMINUYE

La inflación sigue siendo el gran talón de Aquiles de nuestra economía. Los precios aumentaron un 0,2% en diciembre, pero la inflación acumulada en los doce meses del año pasado asciende al 3,7%, muy por encima de la previsión del 2% del Banco Central Europeo (BCE).

El fuerte alza de los combustibles en 2005 provocó un incremento del precio de los transportes del 6,2%, aunque también los alimentos frescos subieron un 5,2%. Ambos grupos son los principales responsables de ese 3,7% interanual, que sitúa a la economía española un punto y medio por encima de la media de la zona euro. A pesar del optimismo del secretario de Estado de Economía, que dijo ayer que la inflación va a empezar a descender en los próximos meses, la realidad es que las previsiones para enero son muy malas. En estas dos semanas de 2006, han subido la luz, el gas, el ferrocarril y el transporte público, el agua, las autopistas y otros muchos servicios.

No es extraña, pues, la reacción de numerosos colectivos que, ayer, realizaron fuertes críticas contra la incapacidad del Gobierno de controlar los precios. La CEOE habló de «merma de la competitividad», CCOO calificó las explicaciones del Gobierno de «excusas», UGT acusó a las grandes empresas y la Federación Nacional de Trabajadores Autónomos se lamentó de la pérdida de poder adquisitivo de sus afiliados. Todos tienen razón.

A pesar de haber dado un gran salto adelante en los últimos diez años, la economía española sigue sin recortar ese permanente diferencial con los países del núcleo duro de la UE, que oscila anualmente entre un punto y un punto y medio.

Hay muchas explicaciones para justificar por qué España tiene una inflación superior a la de esos países. Pero la esencial sigue siendo la falta de competencia en los servicios públicos, dominados por grandes compañías como Repsol, Gas Natural y Telefónica.

En estos dos años de Gobierno socialista, contra el discurso oficial de Zapatero y de ilustres asesores como Miguel Sebastián.lo que ha sucedido es que ha aumentado la concentración de poder económico y ha disminuido la competencia en sectores como las telecomunicaciones, la energía y el transporte.

Sin ir más lejos, los precios de Gas Natural, que controla más del 70% del negocio, crecieron el año pasado en casi un 20%, muy por encima del 11% que subieron los combustibles. Ello no se habría producido si existiera más competencia en el sector.

Pero el alza de los precios está causada también por los problemas de competitividad de las empresas españolas, reflejados en el apabullante déficit comercial del año pasado, el mayor del mundo en términos relativos.

Es el momento de afrontar un plan para fomentar la competitividad y de abordar reformas que incrementen la competencia. Los precios -y, por tanto, los ciudadanos- serán los primeros en notarlo.

Figura 3.2: Uno de los documentos de la colección de editoriales de El Mundo en texto plano, luego de la etapa del filtraje, en él se observa las caracteríticas del documento que se procesará (longitud de palabra, tamaño de texto, tipo, etc)

Los documentos que conforman ambas colecciones documentales, se han *procesado* en cada etapa utilizando técnicas de recuperación de información. Primero, se han removido de cada documento las palabras comunes, que no sirven para la categorización, utilizando un algoritmo de "*stoplist*", y luego, el resto de las palabras fueron llevadas a su raíz utilizando el algoritmo de Porter [Porter, 1980].

A continuación, describiremos detalladamente el modelo de "*procesamiento*" desarrollado para obtener los vectores característicos más representativos de las colecciones documentales, para que luego sean utilizados por el sistema evolutivo desarrollado.

3.2. Procesamiento documental

Como se ha expuesto anteriormente, una de las razones fundamentales para el uso de la colección documental Reuters, es que se ha convertido en un estándar de facto tras ser empleada por un gran número de investigadores en el ámbito de los SRI.

Dado de que la colección Reuters está en inglés, y que además disponemos de una colección documental en español, compuesta por las citadas editoriales del diario El Mundo, se hace necesario desarrollar cada una de las etapas del procesamiento documental en ambos idiomas.

Para comparar mejor la calidad de los agrupamientos se va a proceder a ensayar comparativamente el proceso de agrupamiento para aquellos documentos que en las colecciones tratadas figuren bajo una sola categoría, contrastando la categoría preexistente en la colección con nuestro método.

Por consiguiente, la primera etapa del procesamiento de la colección Reuters, fue seleccionar los documentos que tienen un único valor en el campo *"Tema"* (documentos que pertenecen a una sola categoría), y luego procesarlos a fin de obtener sus vectores característicos.

El procesamiento documental desarrollado en el presente trabajo está conformado por varias etapas, y parte de la información en estado bruto de los documentos (información textual) [Baeza-Yates, Ribeiro Neto, 1999]. Cada una de las etapas representa un tratamiento que se desarrolla sobre la colección Reuters 21578, y sobre la colección de editoriales del diario El Mundo. Todo el procesamiento tiene como objetivo final, el obtener los vectores documentales más representativos de cada colección documental [Castillo José Luis, Fernández del Castillo José R, León González, 2008].

Inicialmente extraemos del documento los términos, donde cada una de las palabras se comparará con una lista de palabras vacías ("*stoplist*") para eliminar las palabras que no tienen interés o carecen de significado propio. Después, las palabras sufren una lematización (*stemming*), para quedarnos con sus raíces. A continuación se realiza la selección de los términos relevantes sobre los que finalmente se aplica una función de ponderación para obtener los pesos asociados a cada término seleccionado.

Las etapas del procesamiento de las colecciones de documentos, se detalla a continuación. (véase figura 3.3). Cada una de ellas constituye en sí un proceso para tratar la información contenida en cada una de las colecciones de documentos que utilizamos.

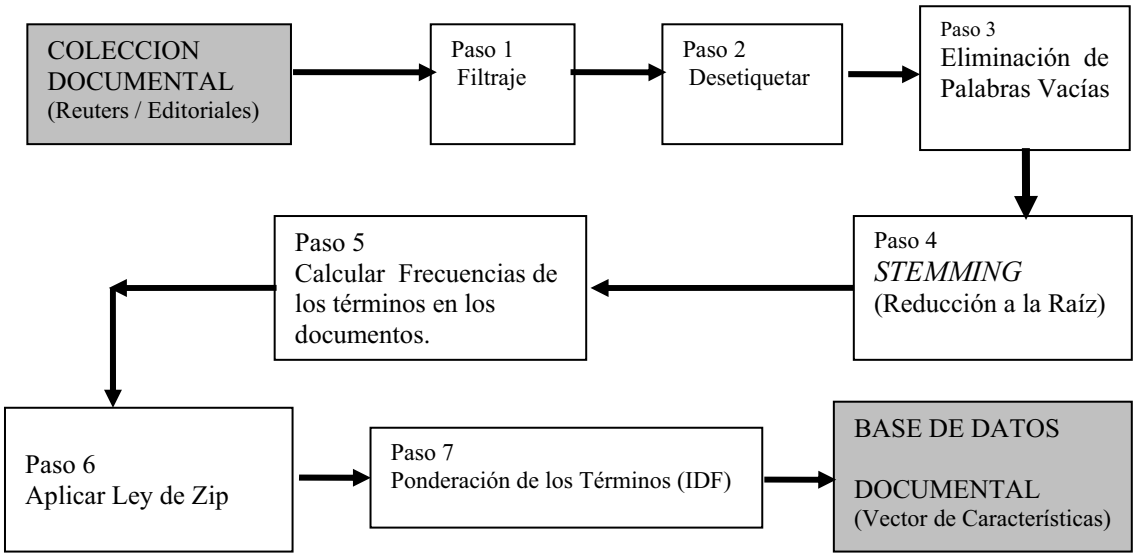

Figura 3.3: Etapas desarrolladas del proceso documental

Comentaremos detalladamente cada una de ellas en las secciones siguientes.

3.2.1. Filtraje

El primer paso es el denominado filtraje, cuyo principal objetivo es determinar los documentos que pertenecen a una sola categoría preexistente en la colección, para luego seleccionar el contenido textual de dicho documento. Este paso es necesario porque al usar la colección documental Reuters, ésta contiene muchos documentos sin agrupar, o con más de una agrupación que no utlizaremos en nuestra experimentación. Por ello, en esta etapa, como parte del filtraje, se descartan los documentos multicategóricos.

Debido a que, en la colección documental Reuters cada documento tiene 5 campos de categorización distintos: *Valor Bursátil, Organización, Persona, Lugar y Tema***;** y en cada campo, el documento puede tener un solo valor, varios o ninguno, se eligió el campo "Tema" por ser el más utilizado en las diferentes estudios que se han realizado con esta colección documental. Así, cada una de las categorías de la colección Reuters a la que un documento puede pertenecer está definida por la etiqueta *"Tema".* Así, si es que un documento pertenece a ninguna, una o varias categorías esta información la encontramos haciendo una revisión de la información que tengamos en esta etiqueta (etiqueta *"TOPICS"*).

Esta etapa fue realizada en toda la colección Reuters, obteniéndose las diferentes categorías a la que un documento puede pertenecer en la colección documental, y que mostramos detalladamente en la la tabla 3.1

acq	drachma	livestock	ringgit
alum	earn	lumber	rubber
austdlr	escudo	lupin	rupiah
austral	f-cattle	meal-feed	rye
barley	ffr	mexpeso	saudriyal
bfr	fishmeal	money-fx	sfr
bop	flaxseed	money-supply	ship
can	fuel	naphtha	silk
carcass	gas	nat-gas	silver
castor-meal	gnp	nickel	singdlr
castor-oil	gold	nkr	skr
castorseed	grain	nzdlr	sorghum
citruspulp	groundnut	oat	soy-meal
cocoa	groundnut-meal	oilseed	soy-oil
coconut	groundnut-oil	orange	soybean
coconut-oil	heat	palladium	stg
coffee	hk	palm-meal	strategic-
copper	hog	palm-oil	metal
copra-cake	housing	palmkernel	sugar
corn	income	peseta	sun-meal
corn-oil	instal-debt	pet-chem	sun-oil
cornglutenfeed	interest	platinum	sunseed
cotton	inventories	plywood	tapioca
cotton-meal	ipi	pork-belly	tea
cotton-oil	iron-steel	potato	tin
cottonseed	jet	propane	trade
cpi	jobs	rand	tung
cpu	l-cattle	rape-meal	tung-oil
crude	lead	rape-oil	veg-oil
cruzado	lei	rapeseed	wheat
dfl	lin-meal	red-bean	wool
dkr	lin-oil	reserves	wpi
dlr	linseed	retail	yen
dmk	lit	rice	zinc

Tabla 3.1: Relación de categorías para todos los documentos de la colección documental Reuters 21578.

Tras esta etapa se ha obtenido el conjunto de documentos que estaban asociados a una sola categoría (de las mencionadas en la tabla 3.1 para Reuters), tal como se aprecia en la tabla 3.2. Esta información la tratamos de tal forma que se obtengan los documentos agrupados por su categoría (número de documento y la categoría a la que pertenece) véase tabla 3.3.

Así por ejemplo, para la distribución 0 de la colección documental Reuters podría resultar una disposición inicial de los documentos como la que se muestra en la tabla 3.2, tras numerar cada documento.

Documento	**Categoría**
Documento 1	earn
Documento 2	acq
Documento 3	cocoa
Documento 4	earn
Documento 5	cocoa

……. y así sucesivamente.

Tabla 3.2: Disposición inicial de datos luego del proceso de filtraje.

Ordenando las categorías, obtenemos la tabla 3.3.

Documento	Categoría
Documento 2	acq
Documento 3	cocoa
Documento 5	cocoa
Documento 1	earn
Documento 4	earn

……. y así sucesivamente

Tabla 3.3: Disposición ordenada de datos luego del proceso de filtraje.

De esta manera, con esta información se puede formar los grupos de documentos, y se tendría la siguiente agrupación de documentos por ejemplo: (Tabla 3.4)

Categoria	Documentos
Grupo 1 (acq)	: Documento 2
Grupo 2(cocoa)	: Documento 3, Documento 5
Grupo 3 (earn)	: Documento 1, Documento 4.

Tabla 3.4: Disposición final de datos luego del proceso de filtraje.

Finalmente, de esta agrupación se deduce el número de grupos (el cual será el *"K"* a experimentar: K = Número de grupos). La tabla 3.4 constituye una información útil que servirá para poder evaluar la calidad del método evolutivo de agrupamiento que se propone en la presente tesis.

Tras el procesamiento de esta primera fase obtenemos los grupos reales de la colección Reuters 21578. La tabla 3.4 describe a que grupos (categorías) pertenece cada uno de los documentos (R0001.TXT, R0002.TXT, etc) de las distribuciones utilizadas.

Para preparar la colección documental en español, tuvimos que definir todas las categorías (grupos) y subcategorías, a las que puede ser asignado un documento de la colección. Para ello usamos el concepto de tesauro de descriptores, el cual constituye listas estructuradas de términos (conceptos), que representan de forma univoca el contenido conceptual de los documentos.

Mediante un tesauro puede realizarce una descripción detallada del contenido temático de los documentos. Las clasificaciones basadas en tesauros desarrolladas a lo largo del tiempo en las bibliotecas, constituyen la base de una familia de clasificadores automáticos que se utilizan para analizar el texto de los documentos y asignarles categorias.

En nuestro caso, utilizamos el tesauro Eurovoc [Eurovoc, 2006], que se describe más abajo, para la elección del conjunto de categorías a las que pueden ser asignados los documentos de la colección en español. Esta elección se justifica por la necesidad de disponer de un conjunto que abarque toda temática posible tratada en los documentos y que además, haya sido elaborado por expertos siguiendo criterios estrictos. De esta forma, preparamos una tabla análoga a la tabla 3.4 para cada uno de los documentos de la colección de los editoriales del diario El Mundo de los años 2006 y 2007 (1402 documentos) pero en este caso haciendo uso del tesauro Eurovoc.

3.2.1.1 Tesauro Eurovoc

Eurovoc [Eurovoc, 2006] es un tesauro plurilingüe que abarca todos los ámbitos de actividad de las Comunidades Europeas, y permite indexar los documentos en los sistemas de documentación de las instituciones allí radicadas.

La versión de Eurovoc que utilizaremos en este trabajo es la 4.2, que existe para las 17 lenguas oficiales de la Unión Europea (español, checo, danes, alemán, griego, inglés, francés, italiano, letón, lituano, húngaro, neerlandés, polaco, portugués, esloveno, finés y sueco). Es utilizada en la biblioteca del Parlamento Europeo y en otras instituciones comunitarias. Para su elaboración se han seguido estrictamente las pautas establecidas por la organización internacional de normalización para la creación de tesauros monolingües (ISO 2788-1986) y multilingües (ISO 5964-1985).

El tesauro es de libre acceso y está permitida la descarga de todo su contenido, lo cual representa una gran ventaja para su aprovechamiento por cualquier usuario. Cada uno de los documentos se imprimió en un formulario en el que en el anverso aparecía el editorial del documento, y en el reverso del formulario figuraba la tabla de categorías de Eurovoc.

El contenido temático se distribuye en 21 áreas que contienen descriptores sobre:

- Vida política.
- Relaciones internacionales
- Comunidades Europeas
- Derecho
- Vida Económica
- Intercambios económicos y comerciales
- Asuntos financieros
- Asuntos sociales
- Educación y comunicación
- Ciencia
- Empresa y competencia
- Trabajo y Empleo
- Transportes
- Medio Ambiente
- Agricultura, silvicultura y pesca
- Sector agroalimentario
- Producción, tecnología e investigación
- Energía
- Industria
- Geografía
- Organizaciones internacionales

A su vez cada uno de estos 21 grandes grupos se divide en subgrupos, formando un total de 127 subcategorías (véase figura 3.4)

VIDA POLITICA
0406 marco político
0411 partido político
0416 procedimiento electoral y sistema de votación
0421 Parlamento
0426 trabajos parlamentarios
0431 vida política y seguridad pública
0436 poder ejecutivo y administración pública

RELACIONES INTERNACIONALES
0806 política internacional
0811 política de cooperación
0816 equilibrio internacional
0821 defensa

COMUNIDADES EUROPEAS
1006 instituciones de la Unión Europea y función pública europea
1011 Derecho comunitario
1016 construcción europea
1021 finanzas comunitarias

DERECHO
1206 fuentes y ramas del Derecho
1211 Derecho civil
1216 Derecho penal
1221 justicia
1226 organización de la justicia
1231 Derecho internacional
1236 derechos y libertades

VIDA ECONÓMICA
1606 política económica
1611 crecimiento económico
1616 región y política regional
1621 estructura económica
1626 contabilidad nacional
1631 análisis económico

INTERCAMBIOS ECONÓMICOS Y COMERCIALES
2006 política comercial
2011 política arancelaria
2016 intercambios económicos
2021 comercio internacional
2026 consumo
2031 comercialización
2036 distribución

ASUNTOS FINANCIEROS
2406 relaciones monetarias
2411 economía monetaria
2416 instituciones financieras y crédito
2421 libre circulación de capitales
2426 financiación e inversión
2431 seguros
2436 hacienda pública y política presupuestaria
2441 presupuesto
2446 fiscalidad
2451 precios

ASUNTOS SOCIALES
2806 familia
2811 movimientos migratorios
2816 demografía y población
2821 marco social
2826 vida social
2831 cultura y religión
2836 protección social
2841 sanidad
2846 urbanismo y construcción

EDUCACIÓN Y COMUNICACIÓN
3206 educación
3211 enseñanza
3216 organización de la enseñanza
3221 documentación
3226 comunicación
3231 información y tratamiento de la información
3236 informática y tratamiento de datos

CIENCIA
3606 ciencias naturales y aplicadas
3611 humanidades

EMPRESA Y COMPETENCIA
4006 organización de la empresa
4011 tipos de empresa
4016 forma jurídica de la sociedad
4021 gestión administrativa
4026 gestión contable
4031 competencia

TRABAJO Y EMPLEO
4406 empleo
4411 mercado laboral
4416 condiciones y organización del trabajo
4421 administración y remuneración del personal
4426 relaciones laborales y Derecho del trabajo

TRANSPORTES
4806 política de transportes
4811 organización de los transportes
4816 transporte terrestre
4821 transporte marítimo y fluvial
4826 transporte aéreo y espacial

MEDIO AMBIENTE
5206 política del medio ambiente
5211 medio natural
5216 deterioro del medio ambiente

AGRICULTURA, SILVICULTURA Y PESCA
5606 política agraria
5611 producción y estructuras agrarias
5616 sistema de explotación agraria
5621 explotación agrícola de la tierra
5626 medio de producción agrícola
5631 actividad agropecuaria
5636 monte
5641 pesca

SECTOR AGROALIMENTARIO
6006 productos de origen vegetal
6011 productos de origen animal
6016 productos agrarios transformados
6021 bebidas y azúcares
6026 productos alimenticios
6031 industria agroalimentaria
6036 tecnología alimentaria

PRODUCCIÓN, TECNOLOGÍA E INVESTIGACIÓN
6406 producción
6411 tecnología y reglamentación técnica
6416 investigación y propiedad intelectual

ENERGÍA
6606 política energética
6611 industrias carbonera y minera
6616 industria petrolera
6621 industrias nuclear y eléctrica
6626 energía blanda

INDUSTRIA
6806 política y estructura industriales
6811 química
6816 metalurgia y siderurgia
6821 industria mecánica
6826 electrónica y electrotécnica
6831 construcción y obras públicas
6836 industria de la madera
6841 industria del cuero e industria textil
6846 industrias diversas

GEOGRAFÍA
7206 Europa y antigua Unión Soviética
7211 regiones de los Estados miembros de la UE
7216 América
7221 África
7226 Asia-Oceanía
7231 geografía económica
7236 geografía política
7241 países y territorios de ultramar

ORGANIZACIONES INTERNACIONALES
7606 Naciones Unidas
7611 organizaciones europeas
7616 organizaciones extraeuropeas
7621 organizaciones intergubernamentales
7626 organizaciones no gubernamentales

Figura 3.4: Grupos principales del Eurovoc, se aprecia que cada grupo se divide en subgrupos, formando un total de 127 subcategorías.

Podemos considerar, por ejemplo, que un documento caracterizado manualmente mediante los tres descriptores que mejor se ajusten a su contenido textual, pertenecerá a las categorías que engloben a dichos descriptores, sin tener en cuenta el orden de los mismos. Así, si estos descriptores pertenecen a una misma categoría, el documento será asignado a la categoría que los engloba. También puede ocurrir, que más de un descriptor o incluso todos, correspondan a diferentes categorías, lo que hará que dicho documento sea desechado.

Esta es una etapa realizada manualmente con los documentos de la colección documental en español. Para el caso de la colección Reuters 21578 no es necesario,

pues lo obtenemos automáticamente a partir del método de procesamiento comentado anteriormente. Todo ello nos servirá más adelante para poder contrastar los resultados obtenidos por el sistema evolutivo propuesto.

Continuamos describiendo las siguientes etapas de la metodología que proponemos.

3.2.2. Desetiquetar

La colección Reuters viene dada en formato SGML, e incluye muchas etiquetas en los documentos que no serán de utilidad para extraer la información que vamos a procesar. De la etapa anterior, tenemos un conjunto de documentos disponibles. En dichos documentos procederemos a eliminar todos los elementos (etiquetas, etc) distintos del mero texto documental.

Cada documento Reuters empieza con una etiqueta de la forma:

<REUTERS TOPICS=?? LEWISSPLIT=?? CGISPLIT=?? OLDID=?? NEWID=??>

donde **??** está rellenado de la forma correpondiente, y cada documento finaliza con una etiqueta de la forma:

</REUTERS>

Además, existen diferentes tipos de etiquetas que son utilizadas para delimitar elementos dentro de un documento. Estas etiquetas aparecen dentro del mismo, y siempre están colocadas en parejas abierto/cerrado. Así por ejemplo pueden tenerse las siguientes etiquetas:

- <DATE>, </DATE> [ONCE, SAMELINE]: Indica la fecha y hora del documento.
- <MKNOTE>, </MKNOTE> [VARIABLE] :Indica ciertas correcciones manuales que se han hecho sobre el corpus original de la colección documental Reuters.
- <TOPICS>, </TOPICS> [ONCE, SAMELINE]: Indica la lista de categorías TOPICS (Temas) de la colección Reuters. Si la categorías TOPICS está presente, cada una de ellas puede estar delimitada por la etiqueta <D> y </D>.
- <PLACES>, </PLACES> [ONCE, SAMELINE]: Mismo que <TOPICS> pero para categorías de PLACES (LUGAR).
- <PEOPLE>, </PEOPLE> [ONCE, SAMELINE]: Mismo que <TOPICS> pero para categorías de PEOPLE (PERSONAS).
- <ORGS>, </ORGS> [ONCE, SAMELINE]: Mismo que <TOPICS> pero para categorías de ORGS (ORGANIZACIÓN).
- <EXCHANGES>, </EXCHANGES> [ONCE, SAMELINE]: Mismo que <TOPICS> pero para categorías de EXCHANGES (VALOR BURSATIL).

- <COMPANIES>, </COMPANIES> [ONCE, SAMELINE]: Esta etiqueta siempre aparece adyacente a otra, indicando si hay categorías COMPANIES asignadas en la colección.
- <UNKNOWN>, </UNKNOWN> [VARIABLE]: Esta etiqueta indica si existe ruido y/o algún material misterioso en la colección documental.
- <TEXT>, </TEXT> [ONCE]: Delimita todo el material textual de cada documento.

Adicionalmente, existen etiquetas que delimitan a los elementos mas representativos de los documentos: (Tabla 3.5)

ETIQUETA	SIGNIFICADO
<AUTHOR>, </AUTHOR>:	Autor del documento
<DATELINE>, <DATELINE>:	Ubicación del documento, y día del año
<TITLE>, </TITLE>	Título del documento
<BODY>,</BODY>	El Texto principal del documento

Tabla 3.5: Etiquetas delimitadoras de la colección Reuters 21578

En cambio, la colección documental en español ha sido obtenida a partir de los editoriales del periódico El Mundo, y no contiene marcas o etiquetas que deban ser descartadas. Por ello, su tratamiento no requiere una etapa previa distinta de seleccionar el texto de cada documento.

Así, la finalidad del *Desetiquetado* es la de obtener el texto libre principal (*"Free Text")* de cada uno de los documentos, trabajando con las diferentes etiquetas descritas anteriormente para el caso de la colección Reuters.

De esta forma, delimitamos el contenido de los documentos que en pasos posteriores procesaremos. Esta extracción se realiza manteniendo la numeración secuencial del documento atribuida en la primera etapa del procesamiento. Se genera un nuevo fichero para cada uno de los documentos (sin ningún metadato) tomadas de cada colección documental, quedando éste sólamente en formato textual. Por ejemplo, para cada uno de los documentos de la colección Reuters obtenidos tras este etapa, tendríamos un conjunto de ficheros en formato texto tal como los que podemos ver en la figura 3.5.

F0001:TXT
Champions Products Inc said its
board of directors approved a two-for-one stock nsplit of its
common shares for shareholders of record as of April 1, 1987.
.....
Reuter
……
F0040.TXT
Northeast Savings F.A.said its board adopted a shareholder rights plan designed to protect the company from coercive takeover tactics and bids not fair to all sharholders.
.....
Reuter

.....
y así sucesivamente

Figura 3.5: Relación de ficheros obtenidos luego del Desetiquetado

3.2.3. Eliminación de palabras vacías (*stoplist*)

En la bibliografía revisada, hay varias formas de eliminar las palabras vacías, tales como:

- Examinar la salida del analizador léxico y eliminar aquellas que sean realmente vacías.
- Eliminar las palabras vacías, entendiendo por tales las presentes en una lista de palabras que se ha preparado previamente, y quedarnos con las palabras restantes (que no están en la lista).

En nuestro caso, hemos optado por eliminar las palabras vacías presentes en una lista de palabras preparadas de antemano, y quedarnos con las restantes. Para ello, hemos desarrollado un algoritmo de *"stoplist"* para cada idioma de las colecciones documentales (inglés y español), a fin de eliminar de cada uno de los documentos obtenidos tras el desetiquetado, aquellas palabras que no son relevantes, y que se muestrán en las tablas 3.6 y 3.8.

De esta forma, en esta etapa, se eliminan las palabras funcionales del lenguaje (artículos, preposiciones, etc.) cuyo valor como término para la indización y para la discriminación es muy bajo. La eliminación de estas palabras antes de la indización hace que ésta sea más rápida, y que se ahorre gran cantidad de espacio en los índices sin repercutir negativamente en la efectividad de la recuperación.

Para realizarla creamos un par de listas de palabras vacías (en inglés "*stoplist*") que nos ayudará a ir eliminando cada una de las palabras carentes de interés y hará que el funcionamiento del SRI sea mejor. Preparamos una lista de palabras vacías denominada *"En Inglés"* para procesar los documentos de la colección documental Reuters; y preparamos otra lista de palabras vacías denominada *"En Español"* para procesar la colección documental en español.

Así por ejemplo, la lista de palabras vacías en inglés, fue preparada explícitamente para que cumplir estas características, y se muestra en la tabla 3.6.

Tabla 3.6: Relación de palabras vacías utilizadas en inglés

a	came	four	if	my
about	can	from	important	myself
above	can't	full	in	n
across	cannot	fully	interest	necessary
after	case	further	interested	need
again	cases	furthered	interesting	needed
against	certain	furthering	interests	needing
almost	certainly	furthers	into	needn't
along	clear	g	is	needs
already	clearly	gave	isn't	never
also	come	general	it	new
although	could	generally	it's	newer
always	couldn't	get	its	newest
all	d	gets	itself	next
am	daren't	give	j	no
among	did	given	just	nobody
an	didn't	gives	k	non
and	differ	go	keep	noone
another	different	goes	keeps	nor
any	differently	going	kind	not
anybody	do	good	knew	nothing
anyone	does	goods	know	now
anything	doesn't	got	known	nowhere
anywhere	doing	great	knows	number
are	don't	greater	l	numbers
area	done	greatest	large	o
areas	down	group	largely	of
aren't	downed	grouped	last	off
around	downing	grouping	later	often
as	downs	groups	latest	old
ask	during	h	least	older
asked	e	had	less	oldest
asking	each	hadn't	let	on
asks	early	has	let's	once
at	either	hasn't	lets	one
away	end	have	like	only
b	ended	haven't	likely	open
back	ending	having	long	opened
backed	ends	he	longer	opening
backing	enough	he'd	longest	opens
backs	even	he'll	m	or
be	evenly	he's	made	order
became	ever	her	make	ordered
because	every	here	making	ordering
become	everybody	here's	man	orders
becomes	everyone	hers	many	other
been	everything	herself	may	others
before	everywhere	high	me	ought
began	f	higher	member	oughtn't
behind	face	highest	members	our
being	faces	him	men	ours
beings	fact	himself	might	ourselves
below	facts	his	mightn't	out
best	far	how	more	over
better	felt	how's	most	own
between	few	however	mostly	p
big	find	i	mr	part
both	finds	i'd	mrs	parted
but	first	i'll	much	parting
by	five	i'm	must	parts
c	for	i've	mustn't	per

perhaps	seen	the	u	while
place	sees	their	under	who
places	several	theirs	until	whole
point	shall	them	up	whom
pointed	shan't	themselves	upon	whos's
pointing	she	then	us	whose
points	she'd	there	use	why
possible	she'll	there's	used	why's
present	she's	therefore	uses	will
presented	should	these	v	with
presenting	shouldn't	they	very	within
presents	show	they'd	w	without
problem	showed	they'll	want	won't
problems	showing	they're	wanted	work
put	shows	they've	wanting	worked
puts	side	thing	wants	working
q	sides	things	was	works
quite	since	think	wasn't	would
r	small	thinks	way	wouldn't
rather	smaller	this	ways	x
really	smallest	those	we	y
reuter	so	though	we'd	year
right	some	thought	we'll	years
room	somebody	thoughts	we're	yet
rooms	someone	three	we've	you
s	something	through	well	you'd
said	somewhere	thus	wells	you'll
same	state	to	went	you're
saw	states	today	were	you've
say	still	together	weren't	young
says	such	too	what	younger
second	sure	took	what's	youngest
seconds	t	toward	when	your
see	take	turn	when's	yours
seem	taken	turned	where	yourself
seemed	than	turning	where's	yourselves
seeming	that	turns	whether	z
seems	that's	two	which	ñ

Tabla 3.6: Relación de palabras vacías utilizadas en inglés (continuación)

Inicialmente estas listas tratan de incluir las palabras que aparezcan con más frecuencia en el idioma en cuestión por ser elementos gramaticales del idioma (artículos, etc). Sin embargo, para su elaboración hemos tenido en cuenta matizaciones, ya que, por ejemplo, en lengua inglesa, los términos que más aparecen en la literatura son: “time”,”war”,”home”,”life”,”water” y “world”. Por lo tanto, si se siguiera esa regla tradicional sería muy difícil encontrar un documento cuyo título fuera por ejemplo “La Segunda Guerra Mundial” y que estuviéramos buscando por contenido de texto libre. En sentido contrario, también existen varias bases de datos especializadas que contienen términos realmente técnicos que resultarían completamente inútiles al público en general. Estos aspectos lo hemos considerado al momento de preparar y procesar la *stoplist*.

Por comodidad, y para no tener problemas en las etapas subsiguientes con las palabras obtenidas en esta fase del procesamiento, las palabras se pasan a minúsculas, eliminándose también las comas y los puntos del fichero de entrada. De esta forma, luego de aplicar el *stoplist* a la colección Reuters, tenemos el fichero de la forma como se muestra en la figura 3.6.

northeast savings board adopted shareholder rights plan designed protect company coercive takeover tactics bids
fair sharholders

plan board declared dividend share purchase northeast common shares held record november company

initially rights exerciseable rights certificates distributed rights automatically trade northeast's shares company

days following acquisition northeast's common shares days following commencement tender offer northeast's shares rights exerciseable separate rights certificates distributed company

rights entitle holders northeast's common shares purchase additional shares exercise price dlrs share company

Figura 3.6: Contenido del documento Reuters de la figura 3,1, luego del stoplist, en él se puede apreciar como queda dicho documento, luego de eliminarse todas las palabras en inglés de la lista de palabras vacías.

De esta forma, tras aplicar éste método, se obtienen las cantidades de documentos y de palabras diferentes que han pasado las etapas de *desetiquetar y stoplist* en cada una de las colecciones procesadas. En la tabla 3.7, vemos la cantidad de documentos, el total de palabras procesadas, el número total de palabras diferentes, el número de términos diferentes que quedan luego del stoplist, y el número de grupos que hay en cada distribución de la colección Reuters 21578, y en la colección de editoriales del diario "El Mundo", luego de realizar los experimentos.

Colección Documental	**Cantidad de Documentos**	**Total de Palabras Procesadas**	**Total palabras diferentes**	**Términos diferentes luego del *stoplist***	**Grupos (K)**
Reuters2-000.SGML	406	51069	12577	5113	31
Reuters2-001.SGML	436	49688	12358	4928	34
Reuters2-002.SGML	455	52160	13213	5357	27
Reuters2-003.SGML	450	58794	14246	5837	32
Reuters2-004.SGML	446	57692	13967	5545	33
Reuters2-005.SGML	498	65543	15270	5985	36
Reuters2-006.SGML	511	58807	14391	5612	38
Reuters2-007.SGML	475	53397	13249	5430	35
Reuters2-008.SGML	452	62372	14762	5905	32
Reuters2-009.SGML	457	59573	14452	5792	33
Reuters2-010.SGML	425	54784	13469	5477	31
Reuters2-011.SGML	485	59878	14418	5773	39
Reuters2-012.SGML	469	56022	13971	5782	37
Reuters2-013.SGML	199	26936	7781	3630	29
Reuters2-014.SGML	179	19747	6339	3081	27
Reuters2-015.SGML	441	47625	12640	4961	35
Reuters2-016.SGML	488	56441	13160	5188	39
Reuters2-017.SGML	398	65996	15408	6543	41
Reuters2-018.SGML	328	44589	11875	5274	39
Reuters2-019.SGML	316	39515	10579	4907	32
Reuters2-020.SGML	402	37752	9905	3781	26
Reuters2-021.SGML	273	26673	7651	3164	16
Colección El Mundo	1402	591708	63677	51406	-

Tabla 3.7: Cantidad de documentos, total de palabras procesadas, palabras diferentes, términos obtenidos luego del stoplist, y grupos filtrados de cada distribución Reuters 21578 y en la colección de editoriales del diario El Mundo.

Todo lo comentado previamente nos induce a concluir que la política que usemos para construir la "*stoplist*" cambiará dependiendo del idioma de los documentos, las características de los usuarios y del proceso de indización.

Así por ejemplo, en nuestro caso también hemos confeccionado el conjunto de palabras que tienen las características comentadas anteriormente, pero para el lenguaje español. Esta lista será de utilidad para nuestra colección de documentos en español cuando apliquemos el s*toplist*.

La tabla 3.8 muestra la lista preparada explícitamente para un s*toplist* en español.

Tabla 3.8: Relación de palabras vacías utilizadas en español

a	conocer	ellos	estemos	habida
añadio	considero	embargo	estén	habidas
aun	considera	en	estes	habido
actualmente	contra	encuentra	esteís	habidas
adelante	cosas	entonces	esto	habiendo
ademas	creo	entre	estos	habrá
afirmo	cual	era	estoy	habrán
agrego	cuales	eraís	estuve	habrás
ahi	cualquier	eramos	estuviera	habré
ahora	cuando	eran	estuviéramos	habreís
al	cuanto	eras	estuvieras	habremos
algun	cuatro	eres	estuvieron	habría
algo	cuenta	es	estuviese	habríamos
alguna	da	esa	estuviésemos	habrían
algunas	dado	esas	estuviesen	habrías
alguno	dan	ese	estuvimos	hace
algunos	dar	eso	estuviste	hacen
alrededor	de	esos	estuvo	hacer
ambos	debe	esta	ex	hacerlo
ante	deben	estan	existe	hacia
anterior	debido	está	existen	haciendo
antes	decir	estaba	explico	han
apenas	dejo	estabamos	expreso	hasta
aproximadam	del	estaban	fin	hay
ente	demas	estabas	fue	haya
aqui	dentro	estaís	fuera	hayaís
asi	desde	estamos	fueras	hayamos
aseguro	despues	estando	fuéramos	hayan
aunque	dice	estad	fueran	hayas
ayer	dicen	estada	fueron	he
bajo	dicho	estado	fuese	hecho
bien	dieron	estados	fuésemos	hemos
buen	diferente	estadas	fuesen	hicieron
buena	diferentes	estar	fueses	hizo
buenas	dijeron	estara	fuí	hoy
bueno	dijo	estarán	fuimos	hube
buenos	dio	estaras	fuiste	hubiera
como	donde	estaría	fuisteis	hubieramos
cada	dos	estaríamos	gran	hubieran
casi	durante	estarían	ha	hubieras
cerca	e	estarías	habéis	hubieron
cierto	ejemplo	estaré	haber	hubiese
cinco	el	estaremos	había	hubieses
comento	ella	estareís	habíamos	hubiésemos
como	ellas	estas	habían	hubiesen
con	ello	este	habías	hubimos

hubiste	nosotros	qué	solas	todos
hubo	nuestra	quedo	solo	total
igual	nuestras	queremos	solos	tras
incluso	nuestro	quien	somos	trata
indico	nuestros	quien	son	traves
informo	nueva	quienes	soy	tres
junto	nuevas	quiere	su	tu
la	nuevo	realizo	sus	tú
lado	nuevos	realizado	suya	tus
las	nunca	realizar	suyas	tuve
le	o	respecto	suyo	tuviera
les	ocho	si	suyos	tuviéramos
llego	os	solo	tal	tuvieran
lleva	otra	se	tambien	tuvieras
llevar	otras	señalo	tampoco	tuvieron
lo	otro	sea	tan	tuviese
los	otros	seaís	tanto	tuvieseis
luego	para	seamos	te	tuviesen
lugar	parece	sean	tendrá	tuviésemos
mas	parte	seas	tendrán	tuvieses
manera	partir	segun	tendrás	tuvimos
manifesto	pasada	segunda	tendré	tuviste
mayor	pasado	segundo	tendreís	tuvisteis
me	pero	seis	tendremos	tuvo
mediante	pesar	sentid	tendía	tuya
mejor	poca	sentida	tendríamos	tuyas
menciono	pocas	sentidas	tendrían	tuyo
menos	poco	sentido	tendrías	tuyos
mi	pocos	sentidos	tened	ultima
mía	podemos	ser	teneís	ultimas
mías	podra	sera	tenemos	ultimo
mío	podran	será	tener	ultimos
míos	podria	serán	tenga	un
mis	podrian	serás	tengaís	una
mientras	poner	seré	tengamos	unas
misma	por	sereís	tengan	uno
mismas	porque	seremos	tengas	unos
mismo	posible	sería	tengo	usted
mismos	proximo	seríamos	tenía	va
momento	proximos	serían	teniamos	vamos
mucha	primer	serías	tenian	van
muchas	primera	si	tenías	varias
mucho	primero	sido	tenida	varios
muchos	primeros	siempre	tenidas	veces
muy	principalment	siendo	tenido	ver
nada	e	siente	tenidos	vez
nadie	propia	siete	teniendo	vosotros
ni	propias	sigue	tercera	vosotras
ningun	propio	siguiente	ti	vuestra
ninguna	propios	sin	tiene	vuestras
ningunas	pudo	sino	tienen	vuestro
ninguno	pueda	sintiendo	tienes	vuestros
ningunos	puede	sobre	toda	y
no	pueden	sois	todas	ya
nos	pues	sola	todavia	yo
nosotras	que	solamente	todo	

Tabla 3.8: Relación de palabras vacías utilizadas en español (continuación)

Realizamos el mismo tratamiento para procesar el texto con los editoriales en español del diario El Mundo. De esta forma, luego de aplicar el *stoplist* a ésta colección en español, tenemos el fichero de la forma que se muestra en la figura 3.7.

Como se ha comentado anteriormente, las palabras que se han eliminado en esta etapa del procesamiento, están compuestos por preposiciones, artículos, etc., pero en este caso son del idioma español. Finalmente, después de esta etapa se tiene para la colección documental en español un conjunto de documentos con la misma numeración (secuencial), conteniendo términos que no pertenezcan a la lista de palabras vacias en español de la tabla 3.8.

En el figura 3.7, mostramos la forma en que queda el documento, luego de aplicar esta etapa del procesamiento, apreciándose claramente que el texto se reduce considerablemente.

precios suben competencia disminuye

inflación talón aquiles economía precios aumentaron diciembre inflación acumulada doce meses año asciende encima previsión banco central europeo bce

fuerte alza combustibles provocó incremento precio transportes también alimentos frescos subieron grupos principales responsables interanual sitúa economía española punto medio encima media zona euro optimismo secretario economía inflación empezar descender próximos meses realidad previsiones enero malas semanas subido luz gas ferrocarril transporte público agua autopistas servicios

extraña reacción numerosos colectivos realizaron fuertes críticas incapacidad gobierno controlar precios ceoe habló competitividad» ccoo calificó explicaciones gobierno ugt acusó grandes empresas federación nacional trabajadores autónomos lamentó pérdida poder adquisitivo afiliados razón

salto últimos diez años economía española recortar permanente diferencial países núcleo duro ue oscila anualmente punto punto medio

explicaciones justificar españa inflación superior países esencial falta competencia servicios públicos dominados grandes compañías repsol gas natural telefónica

años gobierno socialista discurso oficial zapatero ilustres asesores miguel sebastián.lo sucedido aumentado concentración poder económico disminuido competencia sectores telecomunicaciones energía transporte

ir más lejos precios gas natural controla más negocio crecieron año encima subieron combustibles producido existiera más competencia sector

alza precios causada también problemas competitividad empresas españolas reflejados apabullante déficit comercial año mundo términos relativos

afrontar plan fomentar competitividad abordar reformas incrementen competencia precios ciudadanos notarlo

Figura 3.7: Contenido del documento en español de la figura 3.2 luego del stoplist, en él se puede apreciar como queda dicho documento, luego de eliminarse todas las palabras en español de la lista de palabras vacías.

3.2.4. *Stemming* (Reducción a la raíz)

Llegados a este momento, tenemos una selección de palabras para la indización correcta del documento, aquellas que contienen significado o conceptos. Aún así las palabras presentan una elevada variabilidad morfológica que impide la inmediata identificación entre palabras y conceptos. Para resolver este problema necesitamos ser un poco más exigentes y restrictivos con el documento resultante, lo que luego permitirá mejorar el funcionamiento de una hipotética búsqueda de documentos en un SRI.

Así, el siguiente paso consiste en ofrecer al usuario la posibilidad de encontrar las variantes morfológicas de los términos. Procederemos por tanto a la reducción a la raíz de las palabras. Esta etapa se denomina lematización o reducción a raíces léxicas (en inglés *"stemming").*

Para realizar esta etapa aplicamos un *stemming*, adaptado a la naturaleza de la información que se procese, pues el método de *stemming* dejará de ser correcto tanto si las palabras se recortan en exceso como si no se recorta lo suficiente, porque provocaría "*ruido*" (recuperación de documentos no relevantes) o "*silencio*" (la no recuperación de documentos relevantes) de los documentos que se procesen.

Almacenando sólo las raíces de los términos, se puede llegar a reducir su dimensión considerablemente. La reducción de los términos puede realizarse bien durante la indización o bien en la propia búsqueda. La primera variante presenta la ventaja de ser más eficiente y ahorrar espacio, pero tiene la desventaja de perder información sobre los términos completos.

En nuestro caso hemos optado por aplicar el algoritmo de Porter [Porter,1980], tras adecuarlo a nuestra elección para procesar palabras en inglés y en español. De esta forma, tras esta fase obtendremos una lista de términos lematizados representativos de cada documento. Esta lista la podemos manejar computacionalmente con la metodología propuesta.

3.2.4.1. Algoritmo de Porter

El algoritmo de Porter [Porter,1980], se basa en una serie de reglas condición / acción. Tiene la finalidad de obtener la raíz de una palabra (lexema) y consta de una serie de pasos, en cada uno de los cuales se aplican sucesivamente una serie de reglas (reglas de paso) que van suprimiendo sufijos o prefijos de la palabra hasta que nos quedamos sólo con el lexema final.

Cada regla tiene:

- Un identificador de la regla.
- Un sufijo a identificar.
- El texto por el que se reemplaza el sufijo
- El tamaño del sufijo
- El tamaño del texto de reemplazo
- El tamaño mínimo de la raíz resultante luego de aplicar la regla.
- Una función de validación

Dado que un lexema es normalmente más corto que la palabra a la que corresponde, almacenar lexemas en lugar de palabras decrementa el tamaño del fichero de índice. Por ello, a partir de los programas existentes en varios lenguajes de programación, donde se implementan los algoritmos de lematización de Porter, hemos construido un lematizador para aplicarlo a los documentos (en inglés) de la colección Reuters. Tras aplicarlo con todas sus reglas, los textos de los documentos quedan como el que se muestra en la figura 3.8

northeast save board adopt sharehold right plan design protect
compani coerciv takeov tactic bid fair sharhold

plan board declare dividend share purchas northeast common share held record novemb compani

initi right exercis right certif distribut right automat trade northeast's share compani

day follow acquisit northeast's common share day follow commenc tender offer northeast's share right exercis separ right certif distribut compani

right entitl holder northeast's common share purchas addit share
exercis price dlr share compani

compani event trigger event describ right plan holder right acquir person
entitl acquir northeast's common share market valu twice current exercis price right event northeast enter busi combin transact holder right provid acquir equiti secur acquir entiti market valu twice then-current exercis price right compani northeast entitl redeem right cent occur event

Figura 3.8: Contenido del documento Reuters de la figura 3.1 luego del Stemmer, en él se puede apreciar los lexemas en inglés que hay en dicho documento.

Tras aplicar el proceso de *stemming* (aplicando el Porter en inglés) a la colección Reuters, se tienen los documentos y todos sus términos correspondientes en lexemas que corresponden a la raíz de los términos procesados.

Originalmente el algoritmo de Porter fue diseñado para la lengua inglesa. No obstante, debemos adaptar dicho algoritmo a nuestro idioma para procesar una colección de documentos en español. Existen adaptaciones del algoritmo de Porter que permiten procesar documentos en español. En general estas soluciones consisten en la utilización de similar algoritmo que el desarrollado para la lengua inglesa, con sufijos y reglas adaptadas al español, puestas a disposición del publico por sus autores bajo licencia GNU tales como Snowball [http://snowball.tartarus.org/]

Por ello, para nuestros experimentos de *stemmer* en español, hemos utilizado la herramienta multilingue *Snowball* por su versatilidad a muchos idiomas y por su portabilidad porque está desarrollado en lenguaje C. El Snowball es un stemmer que sigue un modelo basado en reglas lexicográficas definidas para el idioma Inglés y extendidas para muchos idiomas, en nuestro caso hemos usado las reglas adaptadas al español de. http://snowball.tartarus.org/algorithms/spanish/stemmer.html

Para la experimentación con el Snowball se utilizó como corpus, el conjunto de documentos en español procesado en el paso anterior. Se experimentó con cada uno de los documentos en formato textual. Los resultados fueron aceptables para los fines experimentales, porque se redujo considerablemente el tamaño de los mismos.

Luego de aplicar el stemmer a la colección en español, tendremos el fichero que se muestra en la figura 3.9.

preci sub competent disminu

inflación talón aquil economí preci aument diciembr inflación acumul doc mes año asciend encim previsión banc central europe bce

fuert alza combust provocó increment preci transport también aliment fresc sub grup principal respons interanual sitú economí español punt medi encim medi zon eur optim secretari economí inflación empez descend próximos mes realid prevision ener mal seman sub luz gas ferrocarril transport públic agu autop servici

extrañ reacción numer colect realiz fuert críticas incapac gobiern control preci ceo habló competitividad» cco calificó explic gobiern ugt acusó grand empres federación nacional trabaj autónom lamentó pérdid pod adquisit afili razón

salt ultim diez años economí español recort permanent diferencial país núcle dur

Figura 3.9: Contenido del documento en español de la figura 3.2 luego del Stemmer, en él se puede apreciar los lexemas en español que hay en dicho documento.

De esta forma, se obtiene una matriz de documentos versus términos (en forma de lexema) para cada uno de las distribuciones de la colección documental Reuters y para la colección del diario El Mundo, que reduce sustancialmente los términos, tal como mostramos en la tabla 3.9.

Colección Documental	Cantidad de Documentos	Términos Lematizados	% Reducción
Reuters2-000.SGML	406	3883	69,1
Reuters2-001.SGML	436	3729	69,8
Reuters2-002.SGML	455	4081	69,1
Reuters2-003.SGML	450	4413	69,0
Reuters2-004.SGML	446	4152	70,2
Reuters2-005.SGML	498	4472	70,7
Reuters2-006.SGML	511	4289	70,2
Reuters2-007.SGML	475	4065	69,3
Reuters2-008.SGML	452	4410	70,1
Reuters2-009.SGML	457	4367	69,7
Reuters2-010.SGML	425	4167	69,0
Reuters2-011.SGML	485	4367	69,7
Reuters2-012.SGML	469	4381	68,6
Reuters2-013.SGML	199	2807	63,9
Reuters2-014.SGML	179	2430	61,6
Reuters2-015.SGML	441	3753	70,3
Reuters2-016.SGML	488	3917	70,2
Reuters2-017.SGML	398	4885	68,3
Reuters2-018.SGML	328	4035	66,0
Reuters2-019.SGML	316	3757	64,4
Reuters2-020.SGML	402	2908	70,6
Reuters2-021.SGML	273	2527	66,9
Colección “El Mundo”	1402	30337	52,3

Tabla 3.9: Cantidad de documentos y terminos lematizados en cada distribución Reuters, y en la colección de editoriales del diario “El Mundo”, luego de las etapas de stoplist y stemmer, se muestra además el porcentaje de reducción

Como resultado de todo el proceso experimental, hemos reducido el tamaño del índice, en grados que varían dependiendo de la colección documental que utilicemos (inglés o español). De esta forma, tras aplicar esta etapa del procesamiento documental, logramos abreviar más cada uno de los documentos de cada colección, reduciendo considerablemente el número de sus términos, para a continuación poder seleccionar los términos que representen a la colección en las siguientes etapas del procesamiento.

3.2.5. Cálculo de la frecuencia de los términos en los documentos

En este paso, para todos los documentos de la colección, se calcula la frecuencia de los términos resultantes de las etapas anteriores, con la finalidad de conocer qué términos lematizados son más utilizados en cada uno de los documentos; y con esta información poder realizar luego una selección de aquellos términos (raíces) que sean más representativos en el documento.

Se dispuso un programa capaz de realizar el cálculo de estos valores para cada uno de los documentos. El resultado es una tabla con la estructura que se muestra en la tabla 3.10. en la que se presenta la frecuencia de cada uno de los términos lematizados. Se observa que T1' es la raíz del primer término obtenido, T2' la raíz del segundo término obtenido, y así sucesivamente con todos ellos.

	D1	**D2**	**……**	**……..**	**D_{406}**	**Total**
T1'	F_{11}	F_{12}	……	……..	$F_{1,406}$	S1
T2'	F_{21}	F_{22}	……	……..	$F_{2,406}$	S2
T3'	F_{31}	F_{32}	……	……..	$F_{3,406}$	S3
……	…..	…..	……	……..	……	…..
Tn'	F_{n1}	F_{n2}	……	……..	$F_{n,406}$	Sn

Tabla 3.10: Representación de las frecuencias de los términos lematizados para los vectores documentales de toda la colección.

En la tabla 3.10 se observa que *F11* es la frecuencia correspondiente del término lematizado 1 en el documento 1 (cuantas veces se repite *T1'* en el documento 1), y así sucesivamente con cada uno de los documentos; y teniendo *S1* como la suma total (cuantas veces se repite *T1'* en todos los documentos). De esta forma, hemos calculado todas las frecuencias de todos los términos lematizados con cada uno de los documentos de la colección experimental.

3.2.6. Utilización de la ley de Zipf

En este punto del proceso, teniendo todos los términos lematizados y conociendo la frecuencia de cada uno de ellos en cada documento, podemos iniciar el paso consistente en la selección de aquellos términos con mayor poder discriminatorio para proceder a su ponderación. Para ello nos basaremos de la ley de Zipf, o también conocida como la ley del mínimo esfuerzo.

Esta ley trata la frecuencia del uso de las palabras en cualquier lengua. Establece que dentro de un proceso de comunicación (sobre todo en el escrito), suelen utilizarse con mayor frecuencia un conjunto reducido de términos, limitando los autores el vocabulario con el que suelen expresar sus ideas.

Zipf [Zipf, 1949] afirmó que, en relación al esfuerzo medio necesario para alcanzar sus objetivos, el ser humano tiende a escoger aquellos procesos que resulten en el menor esfuerzo posible. Esto supone que la conducta humana seguiría el principio del mínimo esfuerzo, implicando que los seres humanos siempre actuán racionalmente y que sus patrones de conducta pueden ser analizados de acuerdo a ese principio.

Analizando estas observaciones Zipf formuló la ley de la frecuencia de palabras en un texto, que explicamos de la forma siguiente: Si hacemos una lista de las palabras del texto colocando en primer lugar la palabra que aparece con mayor frecuencia; en segundo la palabra con segundo valor de frecuencia, y así sucesivamente, al lugar en que quede una palabra en ese texto se denominará *rango de la palabra.* Así, por definición existe una relación entre la frecuencia de una palabra y su rango.

En efecto, mientras mayor sea el rango de una palabra, menor será la frecuencia con la que aparece en el texto. Esto es claro, ya que mientras mayor sea su rango, más abajo estará la palabra en la lista, lo que significa que menor será su frecuencia, la cual depende en forma inversa del rango (porque disminuye a medida que el rango aumenta). Si denotamos con la letra f la frecuencia y con la letra r al rango, entonces la relación matemática es que f depende de r como (1/r), lo que se expresa en la siguiente ecuación:

$$r * f = k \,. \qquad (3.1)$$

donde:

r = es el orden de la palabra en la lista ó rango.

f = es la frecuencia o el número de ocurrencias de esa palabra,

k = es una constante de proporcionalidad que depende del texto elegido y del idioma del mismo.

Esta expresión indica que en un texto o en una colección dada, la frecuencia de aparición de una palabra es inversamente proporcional a la posición que ocupa en el ranking o rango de frecuencia de palabras.

Así, Zipf afirmó que el aumento del rango implica una disminución de la frecuencia. Las palabras que sean demasiado frecuentes en los documentos de una determinada colección no aportan información y pueden ser eliminadas [Zipf, 1949]. De hecho, las palabras que exceden un umbral superior se consideran muy comunes, y aquellas que quedan por debajo de un umbral inferior se consideran raras y, por tanto, ninguna de ellas contribuye de forma significante a proporcionar información. Partiendo de esta hipótesis se puede establecer un umbral superior y otro inferior para la frecuencia de ocurrencia de un término, de tal modo que los términos que queden por encima del umbral superior o por debajo del umbral inferior se consideran términos poco relevantes para el documento [Moreiro González, 2002]**.**

Esto significa que dado un texto, si ordenamos sus palabras según su frecuencia de aparición, podemos identificar ciertos términos que pueden tipificar el asunto del texto. Para mostrar dichos términos es necesario identificar un determinado "punto de corte" y delimitar una zona de transición alrededor de dicho punto de corte (con la finalidad de seleccionar *n* palabras o términos situados en dicha zona como representativos del documento).

Existe una segunda ley de Zipf, enunciada por Goffman citado en [Pao, M.L, 1978] como comunicación personal y comentado en [Muñoz García, 1993]**,** [Moreiro González, 2002], que permite establecer un procedimiento para calcular la zona de transición aludida, determinando un número óptimo de frecuencia de aparición n alrededor del cuál se construye un intervalo de aceptación:.

Si llamamos: I_1 al número de palabras con frecuencia absoluta 1.

I_n al número de palabras con frecuencia absoluta *n*.

se verifica de modo aproximado que:

$$\frac{I_1}{I_n} = \frac{n \cdot (n+1)}{2}. \qquad (3.2)$$

Usando estas leyes según se comprueba en el artículo citado, el procedimiento permite eliminar los términos no relevantes de la colección documental basándose en el punto de corte, que se denominó punto de transición de Goffman [Goffman W,1975]. Según este procedimiento, la zona de transición se debe encontrar en la zona en que I_n, esto es, el número de palabras de frecuencia *n*, tiende a 1, porque esas palabras no serán ni demasiado frecuentes ni demasiado infrecuentes. Operando sobre la segunda ley de Zipf, sustituyendo I_n por 1, obtenemos:

$$n = \frac{-1 + \sqrt{1 + 8 \cdot I_1}}{2}. \qquad (3.3)$$

Así pues, hallando *n* (punto de transición), aparecen como términos indizadores aquellos que se hallen en su entorno, cuyo tamaño lo da el valor de un parámetro *d* (la zona de transición queda comprendida en el intervalo [n-d, n+d]). El valor de *d* es ajustable hasta obtener un número de términos que consideremos adecuado. El punto de transición es el punto donde las palabras de alta frecuencia comienzan a transformarse en baja frecuencia. [Urbizagastegui Ruben, 1995]

Hay dos formas de aplicar la ley de Zipf [Velasco, et al, 2004]. La primera calculando la zona de transición documento a documento, y la segunda calculando la zona de transición para el total de documentos que formen el corpus .

Al trabajar documento a documento aparece como principal problema el de la representatividad de la longitud de los documentos. Si los documentos son cortos, la zona de transición suele quedar muy desplazada hacia la derecha (hacia aquellos términos que presentan un número elevado de ocurrencias), con lo que el número de palabras ó términos vacíos que salvarán la etapa de filtrado y se incluirían como descriptores sería mayor del deseado, véase figura 3.10:

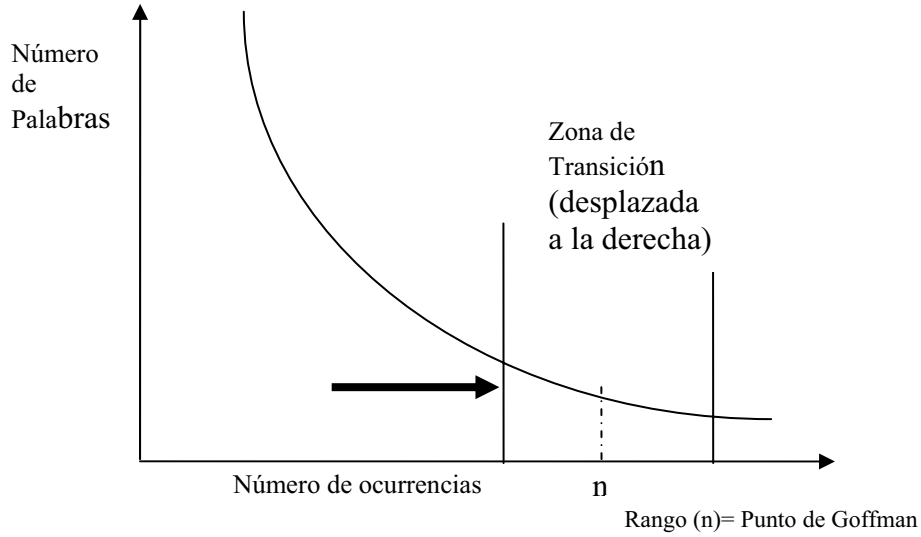

Figura 3.10: Zona de transición, delimitada para los documentos

Otro parámetro de funcionamiento se refiere al número de términos filtrados, concretamente el valor de *d* que hace que los términos aceptados tengan frecuencias de aparición próximas al punto de transición. Hay tres formas de definir dicho valor:

- Mediante un rango fijo: Tomando un valor absoluto. Podría usarse una función que tomase como parámetro el valor del punto de transición de Goffman (n) y el número de términos del documento.

- Mediante la elección de un número máximo de términos filtrados por documento: Debe definirse uno diferente para cada documento, ya que no todos los documentos tienen la misma representatividad, ni el mismo número de términos.

- Mediante el establecimiento de un porcentaje de términos filtrados por documento: A partir del número de términos diferentes, y en función de heurísticas obtenidas de la observación sobre el filtrado se podría establecer una función que determine el porcentaje óptimo de términos a filtrar.

En el caso de realizar la etapa de filtrado globalmente para el conjunto de todos los documentos, desaparece alguno de los problemas anteriores pero surgen otros. El primero que surge es el problema de los descriptores compuestos, pues se incrementa este número al tratarse globalmente la información. Todo ello implica que podemos identificar, los términos de indización para cualificar el contenido de textos. Sin embargo, hay pocos trabajos desarrollados como aplicación práctica de esta ley a los procesos de indización. La mayor parte de los artículos publicados se centran en prototipos académicos y están más preocupados por su validez matemática que en su aplicabilidad real en una situación empírica concreta.

En vista de todo esto, y teniendo en cuenta que es factible hacer un procesamiento automático aplicando la ley de Zipf para seleccionar los valores apropiados de indización de nuestra colección documental experimental, y poder así obtener aquellos términos que tengan un mayor poder discriminatorio en los documentos, aplicamos en esta etapa del procesamiento de los documentos esta ley, pero a diferencia de lo que propone Zipf, será aplicada sobre los términos lematizados de la colección documental.

En nuestra propuesta hemos optado por aplicar la ley de Zipf, usando un filtrado global con un rango fijo a partir de la zona de transición. Debido a que aplicamos la ley de Zipf con términos lematizados, el problema de los descriptores compuestos lo solucionamos, aún a costa de lematizar los documentos, pero nos compensa el esfuerzo, porque cuando se trabaja con raíces de palabras se reduce considerablemente el tamaño del documento.

Por lo tanto, podemos determinar los términos lematizados cuya frecuencia en la colección documental es igual a 1, y aplicar la ecuación 3.3. de la Ley de Zipf a fin de encontrar la frecuencia (de transición) y luego el punto de transición (punto de Goffman) para delimitar mejor la zona de transición. Es decir, para poder luego escoger aquellos términos que más nos interesen, y sean más relevantes de la colección. Una vez que hemos obtenido todos los términos con mayor poder discriminatorio, es decir, los más representativos, procederemos a la vectorización de los documentos. Esta etapa consiste en la construcción de vectores de dimensión igual al número de términos significativos que hemos seleccionado.

La colección documental podría representarse como una matriz, de términos lematizados donde cada fila de la matriz representa un documento y cada columna representa un término específico lematizado que ha sido seleccionado con nuestra método de procesamiento. Por lo tanto, el documento d_i se identificaran mediante una colección de términos lematizados extraidos de la tabla 3.10, donde los términos se han seleccionado usando un rango fijo desde la zona de transición encontrada a partir del punto de Goffman.

3.2.7. Ponderación de los vectores documentales

Tras haber seleccionado los términos más representativos para nuestros documentos, el paso siguiente es ponderar cada uno de los términos elegidos con el fin de poder tener una representación vectorial lo más coherente y fiable que nos permita, posteriormente, experimentar con el sistema evolutivo desarrollado. Es decir, en este paso se crean los vectores característicos, tomando como dimensión del vector, el número de los términos significativos escogidos, y ponderando cada uno de los términos lematizados.

Para ponderar los términos usamos el método IDF (Inverse Document Frecuency) ideada por Salton [Salton, McGill, 1983] que utiliza tanto la frecuencia de una palabra en el documento como el número de documentos de la base en los que aparece para calcular su poder de resolución. De esta forma se pretende estimar la capacidad de discriminación de un término, y está dada por una función decreciente con respecto al número de documentos de la base en los que aparece la palabra ó término en cuestión, de ahí su nombre “Inverted Document Frecuency” o IDF.

Si se admite que la importancia de un término en un texto particular es proporcional a la frecuencia de su aparición en ese texto, e inversamente proporcional al número total de textos en que aparece ese término [Muñoz García, 1993], llegaremos al esquema de asignación de pesos IDF: (comentada en el cápitulo 2 en la Ecuación 2.4)

De esta forma, en esta etapa del procesamiento, los térmnos seleccionados de la colección documental, lo ponderamos apropiadamente con el método IDF. Es decir, le damos un peso específico a dichos términos, y posteriormente construimos los vectores característicos que serán procesados por el sistema evolutivo, cuya dimensión dependerá del número de términos que fueron seleccionados y ponderados con el método IDF. La tabla 3.11 muestra la representación vectorial de cada uno de los documentos, donde los términos seleccionados se representan con T_i.

	T1	T2	……	……..	….	Tt
DOC_1	W_{11}	W_{12}	……	……..	……	$W_{1\,t}$
DOC_2	W_{21}	W_{22}	……	……..	……	$W_{2\,t}$
DOC_3	W_{31}	W_{32}	……	……..	……	$W_{3\,t}$
……	…..	…..	……	……..	……	…..
Doc n	W_{n1}	W_{n2}	……	……..	……	$W_{n\,t}$

Tabla 3.11: Representación final de los vectores característicos en la colección

Donde los valores w_{ij} corresponden a los pesos de cada uno de los términos lematizados que han sido seleccionados de los documentos de nuestra colección documental, que nos indicará el grado de importancia del término en cada documento.

3.2.8. Número de términos a seleccionar

El método de procesamiento comentado lo hemos denominado NZIPF, pero nos queda determinar cuantos términos lematizados son los más apropiados de seleccionar. Hemos realizado muchas pruebas experimentales, tomando como entrada diferentes muestras de documentos de cada una de las diferentes colecciones documentales..

Para cada una de las muestras hemos aplicado la metodología presentada y hemos optado por aplicar la ley de Zipf, usando un filtrado global con un rango fijo a partir de la zona de transición. Debido a que aplicamos la ley de Zipf con términos lematizados, el problema de los descriptores compuestos lo salvamos, aún a costa de realizar la lematización de los documentos, y realizar cada uno de los pasos descritos para el método NZIPF.

Por lo tanto, la dimensionalidad de los términos es fija en cada colección, pero cambia entre cada una de las diferentes colecciones procesadas, dependiendo del punto de Goffman que encontremos. Es decir, siempre seleccionamos el número de terminos a partir del punto de Goffman. Dicho número, en todos los casos es un número de muy alta dimensión (más de 3000 términos aproximadamente), lo que origina que los vectores característicos sean de una dimensión muy alta.

. En los siguientes cápitulos comprobamos experimentalmente las pruebas realizadas aplicando el sistema evolutivo propuesto y usando el algoritmo de agrupación del tipo supervisado llamado Kmeans. Hemos comprobado la validez del método, porque nos proporcionan resultados estables y aceptables. Además dichos resultados no varian sustancialmente al cambiar la colección experimental. [Castillo José Luis, Del Castillo José R, León González, 2009].

De esta forma, generamos los vectores característicos para realizar todos los experimentos del siguiente aspecto de esta tesis, osea, el desarrollo de un sistema evolutivo de agrupación de documentos, que es materia del siguiente capítulo.

Capítulo 4

Sistema Evolutivo

El objetivo de este capítulo es presentar el Sistema Evolutivo que proponemos concebido con técnicas de Computación Evolutiva, cuyo objetivo fundamental es el de realizar el agrupamiento de documentos de manera no supervisada. El criterio utilizado para realizar el agrupamiento de dichos documentos está basado en el uso de una función de *"fitness"* que utiliza tanto la *similitud como la distancia entre los documentos que conforman la base documental sobre la que opera el SRI*, midiendo la cercanía y afinidad existente entre estos documentos, lográndose conseguir de esta forma los grupos ó "*clusters*", con documentos afines, que podría presentar la colección documental. El sistema propuesto se presenta como una alternativa a los métodos de agrupamiento, tales como son los jerárquicos o los particionales.

4.1. Introducción

Durante la evolución de las especies, se han ido seleccionando las características que les hacen supervivientes. Las especies y los individuos que sobreviven son aquellos que están mejor adaptados al ambiente que les rodea, pudiendo de esta forma, reproducirse y transferir a sus descendendientes las cualidades que les permitieron estar mejor adaptados.

El proceso de selección natural postulado por Charles Darwin supone que en el curso de una evolución natural, en las especies se va seleccionando características que la hagan más aptas para sobrevivir, adaptandose mejor al medio y transmitiendo en mayor proporción de los individuos a sus descendientes las ventajas comparativas acumuladas.

Este paradigma puede ser útil como guía para desarrollar sistemas artificiales que de un modo progresivo se vayan encaminando a alguna aplicación. Así por ejemplo, el campo de la Computación Evolutiva considera los paradigmas de Programación Genética y de Algoritmos Genéticos inspirados en dicha idea.

La evolución, aplica el principio darwiniano de reproducción y supervivencia del más apto, utilizando la operación genética de recombinación sexual (cruce) para crear nuevos individuos. Análogamente, en todo sistema evolutivo artificial, la recombinación "sexual" entre dos representaciones estructuradas de información ("padres") permite crear nuevas representaciones estructuradas ("hijos"), con características similares a las de sus padres. Esto se realiza aplicando un operador de cruce que recombina partes elegidas al azar, intentando producir individuos acaso más aptos como resolución de un problema específico.

También, en todo proceso *evolutivo*, cada cierto tiempo aparecen individuos que han sufrido una alteración (mutación) de sus cromosomas debido a ciertos factores que les afecta positiva o negativamente. Este proceso se implementa también en el paradigma evolutivo utilizando un operador genético de mutación, y a resultas de la cual obtenemos individuos más o menos aptos, produciendo en el primer caso ventajas reproductivas que se transmiten a la especie.

En nuestro caso, se producen nuevos individuos, combinando o mutando características de los individuos, incorporando los nuevos a la población y eliminando de la misma los menos adaptados, de manera que las características de los mejores adaptados tienden a mantenerse y perfeccionarse (elitismo), pudiendo asi llegar a obtener soluciones adecuadas. Después de las operaciones de recombinación y mutación, la nueva población (población hija) reemplaza a la antigua población y el proceso se repite.

De esta forma, en los sistemas evolutivos, todos los rasgos que faciliten la supervivencia del individuo tenderán a mantenerse, mientras que lo que constituya una debilidad para la adaptación de la especie tenderá a desaparecer, pues su poseedor no tendrá oportunidad de legarlo a su descendencia.

Debido a que en los sistemas evolutivos la población se selecciona siguiendo el principio darwiniano ya explicado de supervivencia del más apto, éste paradigma resulta muy versátil y permite resolver problemas en contextos muy diferentes, como en particular al campo de la RI. Por ello, en esta tesis lo aplicaremos al agrupamiento de documentos.

4.2. Sistema Evolutivo para el agrupamiento de documentos de un sistema de recuperación de información

Como ya hemos dicho, tratamos de implementar un sistema no supervisado que permita agrupar de manera evolutiva los documentos de un SRI. Nuestro enfoque aplica una función de adaptación (*fitness*) que combina los conceptos de *distancia* y de *similitud* entre los documentos. El sistema propuesto estará conformado por los siguientes elementos y características:

- Individuos
- Tamaño de la población.
- Número de generaciones.
- Operadores de producción.
- Métodos de selección.
- Criterios evolutivos a considerar en el modelo propuesto.

A continuación, presentaremos detalladamente cada uno de los elementos:

4.2.1. Los individuos

La población está conformada por un conjunto de individuos, cada uno de los cuales adopta la estructura arborescente. Estas estructuras, están formadas por nodos. En nuestro sistema, existen nodos terminales y no terminales, conectados por aristas.

Los nodos terminales (nodos hojas) serán los documentos. Cada nodo terminal contiene la representación vectorial de un documento (representado por su vector característico, obtenido del procesamiento detallado en el capitulo 3). De esta forma, a cada documento de la colección le corresponde un único nodo terminal. Cada nodo no terminal contiene el resultado de la función de adaptación (*fitness*) de los dos nodos pendientes de ellos y las coordenadas del centroide de ambos.

Denominamos fitness global al fitness que se corresponde con el nodo no terminal de nivel superior (nodo raíz) y fitness parcial al obtenido en el resto de los nodos no terminales. Por lo tanto en la representación existen muchos *fitness* parciales que se tendrán que evaluar de manera recursiva, y finalmente un solo *fitness* (global) para cada individuo (cada árbol).

En la practica, codificamos cada una de estas estructuras mediante una cadena de cifras, en la que los ceros (0) denotan nodos no terminales y las subcadenas sin ceros representa a cada documento. En ella, cada documento aparece como un avatar (cadena) dispuesto entre los ceros, que en notación polaca describe al árbol.

Por ejemplo, podemos mostrar dos colecciones documentales, interpretadas a través de un recorrido en preorden, cuya representación mediante cadenas es:

- 000A50420a$ que se corresponde con los documentos [110,5,4,2,95,221] colocados en un determinado orden (véase figura 4.1).
- 000&Fh0300152 que se corresponde con los documentos [205,97,42,3,1,5,2] colocados en un determinado orden (véase figura 4.1).

Para la primera colección documental (primera cadena), el carácter “A” es un avatar que se corresponde con el documento 110, el carácter “5” con el documento 5, el carácter “4” con el documento 4, el carácter “2” con el documento 2, el carácter “a” con el documento 95, el carácter “$” con el documento 221:

Para la segunda colección documental (segunda cadena), el carácter “&” es un avatar que se corresponde con el documento 205, el carácter “F” con el documento 97, el carácter “h” con el documento 42, el carácter “3” con el documento 3, el carácter “1” con el documento 1, el carácter “5” con el documento 5 y el carácter “2” con el documento 2.

Cada una de las cadenas representan un individuo, cuya representación arborecente lo podemos mostrar en la gráfica 4.1.

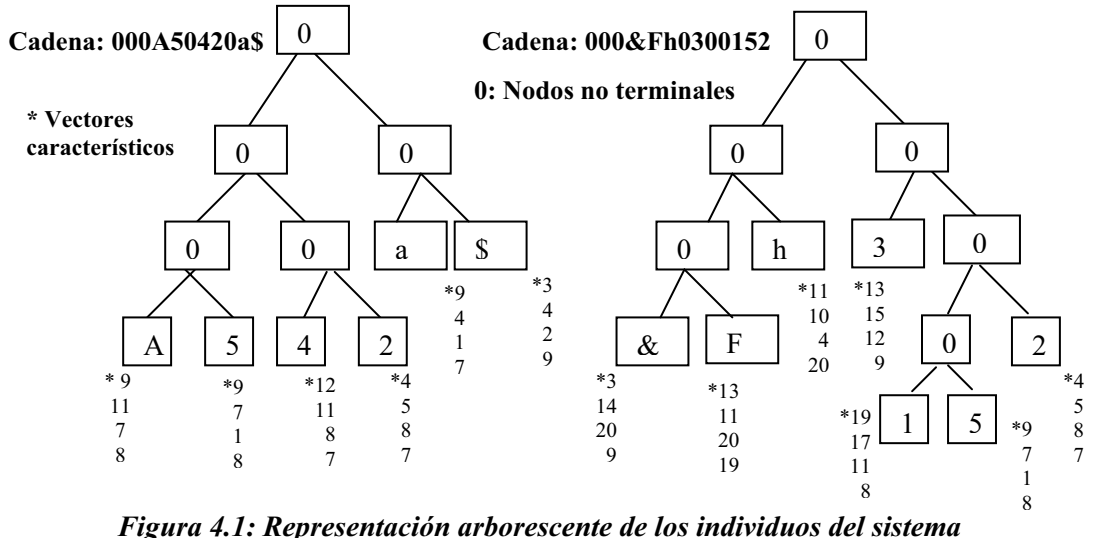

Figura 4.1: Representación arborescente de los individuos del sistema

Por ello, en nuestra propuesta, un individuo se considera como un cromosoma que adopta una longitud propia (dependiendo del número de documentos). El individuo está representado por una estructura de árbol binario con todos los documentos a agrupar en sus hojas. Se utiliza un árbol binario porqué las funciones del fitness (distancia, similitud) son medidas que operan sobre pares de vectores, y como se ha dicho anteriormente, cada documento está formado por un *vector (característico)* que está caracterizado por los pesos de los términos lematizados que fueron seleccionados aplicando la metodología de procesamiento de documentos explicada en el capítulo 3, también mostrados a modo de ejemplo en la gráfica 4.1.

Esta representación evolucionará aplicando los conceptos de la Computación Evolutiva hasta que los individuos encuentren los grupos "*Clusters*" más apropiados para todos los documentos de la colección documental del SRI. En el nodo raíz se almacenará el valor resultante de nuestra función de adaptación que mide la calidad del agrupamiento. Esta función está definida por el operador que se propone en la sección 4.2.6.3, basado en la distancia y en la similitud de los documentos del SRI.

Consideramos que la representación de los individuos es una de las aportaciones originales, que a diferencia de las otras propuestas hechas sobre AG las cuales hasta donde conocemos, utilizan *solamente* cadenas de longitud fija. De esta forma, nuestra representación interna se basa en cadenas cuya longitud depende del número de documentos que se desee procesar. Dichas cadenas son transformadas, por medio de reglas de producción en árboles no equilibrados que representan individuos de diferentes formas y tamaños, que podemos hacer evolucionar, dejando que sea la propia evolución la que decida cuál es la mejor de las configuraciones.

Hay una serie de restricciones que limitan la creación de dichos individuos. Así por ejemplo no se deberá crear en el conjunto inicial dos individuos iguales. Las reglas de producción que garanticen que se cumpla dicha condición, obligan a que la gramática de creación de los nodos de cada individuo se realice haciendo un recorrido del árbol en "*Preorden*" [Castillo José Luis, Fernández del Castillo José R, González León, 2008a]**.**

Otra novedad que se añade al sistema es la posibilidad de establecer el llamado "*umbral de complejidad"***,** asignado por el usuario, con el fin de que a cada individuo se le pueda limitar estructuralmente para que el sistema no consuma demasiado tiempo con él. Este umbral de complejidad se aplica para limitar la profundidad del árbol.

La población inicial estará conformada por el conjunto de individuos, generados al inicio en la llamada generación 0, tal como, por ejemplo, se puede apreciar en la figura 4.2 que muestra el proces, la estructura que adopta cada uno de los individuos y los valores que caracterizan a cada documento, que se recogen del procesamiento previo de los documentos que se comentó en el capítulo 3. En la gráfica a modo ilustrativo denotamos con "F" al fitness global y "f" al fitness parcial.

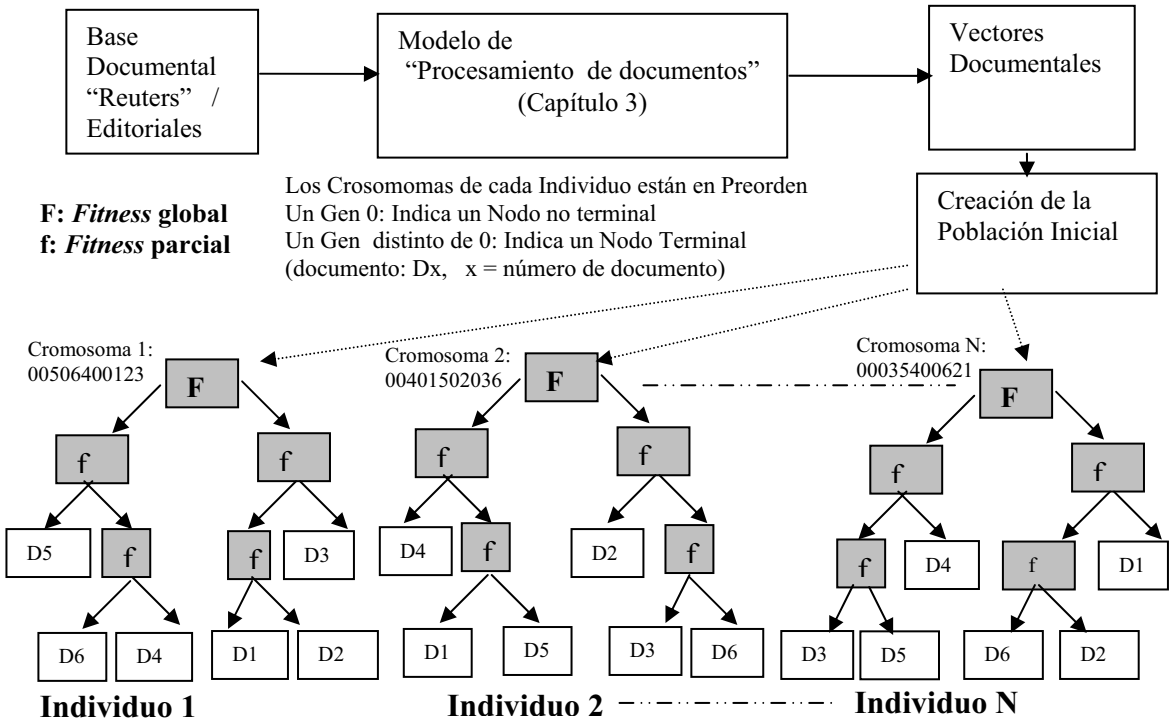

Figura 4.2: Población inicial de individuos del Sistema Evolutivo

4.2.2. Tamaño de la población

Cuanto mayor sea la población, más posibilidades y más variedad obtendremos durante la evolución. El tamaño de la población constituye un aspecto crucial en el sistema propuesto, pues si por ejemplo utilizamos poblaciones pequeñas corremos el riesgo de no cubrir todo el espacio de búsqueda, sin embargo utilizar grandes poblaciones puede suponer un esfuerzo computacional excesivo.

Los experimentos realizados comienzan con pocos individuos**,** tal como aconsejan los estudios experimentales previos, con la finalidad de poder comprobar si se evoluciona favorablemente. Posteriormente al observarse (experimentalmente) que el sistema converge correctamente, se incrementa el número de individuos, y se analizan los resultados obtenidos [Castillo José Luis, Fernández del Castillo José R, González León, 2008a]. [Castillo José Luis, Fernández del Castillo José R, León González, 2008b]

Para controlar el *Tamaño de la Población*, usamos la estrategia llamada *GAVAPS* (Genético de variación del tamaño de la población) propuesto por Michalewicz [Michalewicz, 1999], [Arabas, Michalewicz, Mulawka,1994] usando el concepto de edad y tiempo de vida. Cuando se crea la primera generación asignamos a todos los individuos una edad de cero, haciendo referencia al nacimiento del individuo, cada vez que llega una nueva generación la edad del individuo aumenta en uno. Al mismo tiempo que se crea un individuo se le asigna un tiempo de vida, que representa cuánto tiempo vivirá el individuo dentro de la población, cuando su edad llegue a este tiempo de vida morirá. El tiempo de vida depende del fitness en comparación con el promedio de la población; de este modo, si un individuo tiene mejor fitness tendrá mayor tiempo de vida, dándole mayor posibilidad de generar nuevos individuos con sus características.

Por ello, el tamaño de la población está formado en cada momento por lo individuos que tengan vida en el sistema en cada generación, con el antes ya comentado enfoque GAVAPS, más el resto de individuos que son generados aleatoriamente. De esta forma garantizamos la intensificación del óptimo en el espacio factible de la población, manteniéndolos mientras tengan un tiempo de vida; y garantizamos la diversidad generando al *azar los individuos restantes* en cada generación, para así explorar muchas regiones diferentes del espacio de búsqueda y lograr un cierto equilibrio entre intensificación y diversidad de las regiones factibles

4.2.3. El número de generaciones

La evolución se lleva a cabo modificando los individuos de la población con los operadores de cruce y mutación, a través de un cierto número de generaciones que definimos de antemano. En la primera generación se crean los individuos iniciales que mediante técnicas evolutivas y operadores genéticos permitirán obtener las siguientes generaciones de individuos que tendrán que evaluarse con el fin de determinar la evolución del sistema.

El sistema establece en principo un número máximo de generaciones a alcanzar; no obstante una vez alcanzado dicho máximo se puede dar la opción de continuar la evolución dependiendo del valor del *fitness* del AG.

En los experimentos realizados en el presente trabajo el valor de este parámetro se fijó en 4000 generaciones tal como lo comentamos en los experimentos que detallamos en la sección 5.6.

4.2.4. Los operadores de producción

Los operadores de producción se aplican en el momento de crear una generación a partir de la anterior (salvo en la generación 0). Los operadores pueden tomar uno o dos individuos y producir a partir de ellos nuevos individuos que poblarán la generación siguiente.

Para el sistema propuesto, se diseñan e implementan específicamente tanto el operador de mutación como el de cruce. Ambos operadores son dependientes de sendas tasas, que se asigna al sistema por el usuario. Dichas tasas permiten la utilización de los operadores. El individuo generado luego de aplicar el operador debe de cumplir dos requisitos. El primero, ser un individuo válido, es decir en sus nodos terminales (nodos hojas) debe de tener el avatar que representa a los documentos que se esté agrupando, y el segundo ser un individuo diferente (no repetido) dentro de la

población. Además solamente se incorpora a la población si su valor de *fitness* mejora al de sus padres

4.2.4.1. Operador de mutación

En el presente trabajo se desarrolla un operador de mutación sobre nodos. Este operador se implementa seleccionando un individuo de la población por el llamado método del torneo, descrito en el capítulo 2, y luego de manera aleatoria se selecciona un par de nodos de dicho individuo.(siempre y cuando alguno de ellos sea un nodo terminal). A continuación se intercambian dichos nodos, generándose así un nuevo individuo que podría tener una estructura arborescente diferente. El nuevo individuo tiene los mismos documentos de la base pero colocados en un orden distinto. Ejemplos de la utilización de este operador aparecen representados en las figuras 4.3 y 4.4 respectivamente.

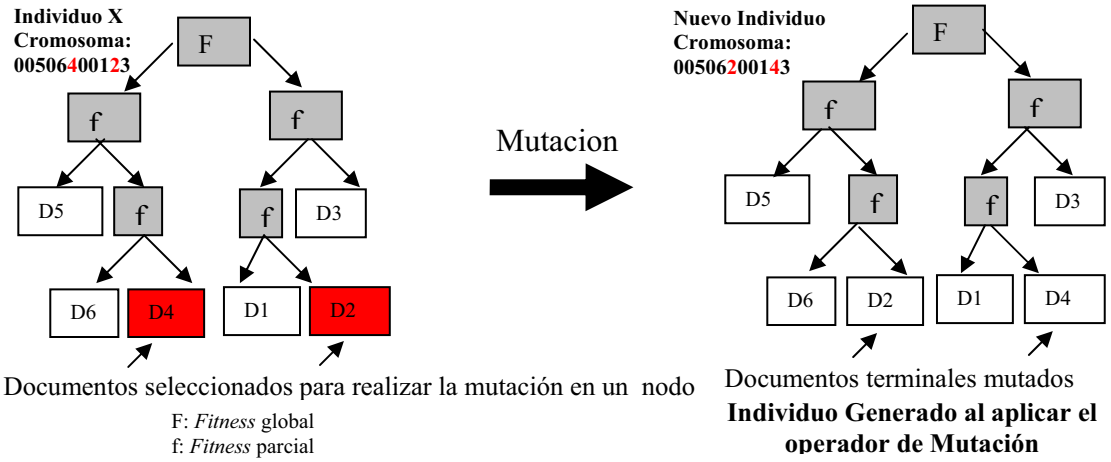

Figura 4.3 Uso del operador de mutación sobre nodos terminales de un individuo en la población genética.

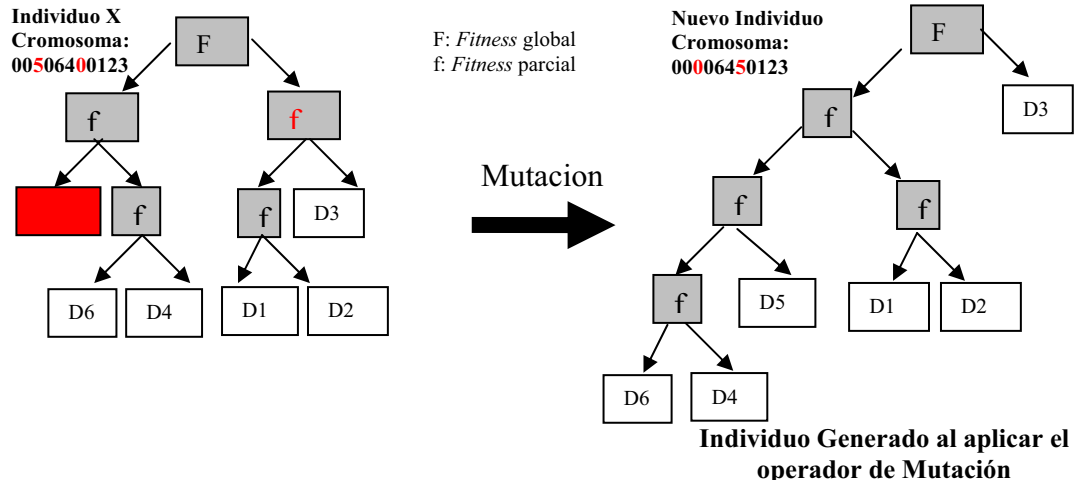

Figura 4.4 Uso del operador de mutación sobre nodo terminal y nodo no terminal de un individuo en la población genética.

En nuestra propuesta, tal como ya lo hemos comentado utilizamos este operador de mutación para generar y modificar nuevos individuos, intentando siempre mejorar (en *fitness* global) el individuo. Solamente mantenemos en la población el individuo mutado que mejora al individuo original. De esta forma, intentamos crear mayor diversidad en la población, mejorando las características del individuo mutado.

4.2.4.2. Operador de cruce

Para el operador de cruce desarrollamos un operador de cruce sobre la población de individuos basado en una *mascara de cruce*. Para ello, primero seleccionamos por el método del torneo dos individuos padres, luego elegimos de manera equiprobable (aleatoriamente) el cromosoma de uno de ellos, y dicho cromosoma es utilizado para definir la *mascara para realizar el cruce* [Castillo José Luis, Fernández del Castillo José R, González León, 2008a].

Para aplicar la mascara, tenemos que denominar *mascara de padre elegido* al cromosoma del individuo elegido aleatoriamente, y *mascara de padre no elegido* al otro individuo restante que no fue elegido.

El nuevo individuo tendrá en principio los avatares correspondiente al cromosoma de la *mascara de padre elegido.* Luego, el cruce se realiza recorriendo gen a gen los cromosomas de ambos padres, para ello se analizan los genes correspondientes a ambos cromosomas. Si en ambos genes de los cromosomas existe al menos un nodo no terminal (nodo 0) mantenemos el gen que corresponde a la *mascara del padre elegido* en el nuevo individuo hijo. Pero si encontramos nodos terminales (documentos) en ambos genes de los cromosomas padres, entonces seleccionamos el gen de la mascara del *padre no elegido*, y dicho gen es buscado e intercambiado en el nuevo individuo (*mascara)* con su correspondiente gen de la *mascara del padre no elegido*.

De esta forma, creamos un nuevo individuo “cruzando” los cromosomas de los padres seleccionados mediante éste tipo de operador. Podemos asegurar que en el cromosoma creado se mantienen las mismas características estructurales de los padres (generar un individuo con documentos no repetidos).

Así, por ejemplo si tenemos 5 documentos en la colección documental, y tenemos los siguientes cromosomas padres seleccionados:

Padre 1:	0	0	2	1	0	5	0	3	4
Padre 2:	0	0	0	5	3	0	2	1	4

Si por ejemplo, el segundo cromosoma (Padre 2) resulta ser elegido como *mascara de padre elegido, y* el primer cromosoma (Padre 1) resulta ser la *mascara del padre no elegido.*

El cromosoma creado luego de aplicar el operador de cruce propuesto sería:

0	0	0	1	3	0	2	5	4

En la figura 4.5 mostramos gráficamente el operador de cruce basado en mascara, usando la representación que corresponde al conjunto de documentos que hemos tomado previamente como ejemplo. Podemos ver que tras el cruce, se crea "*un nuevo individuo*", que es estructuralmente similar a sus padres y en el que los documentos representados no se repiten en su estructura.

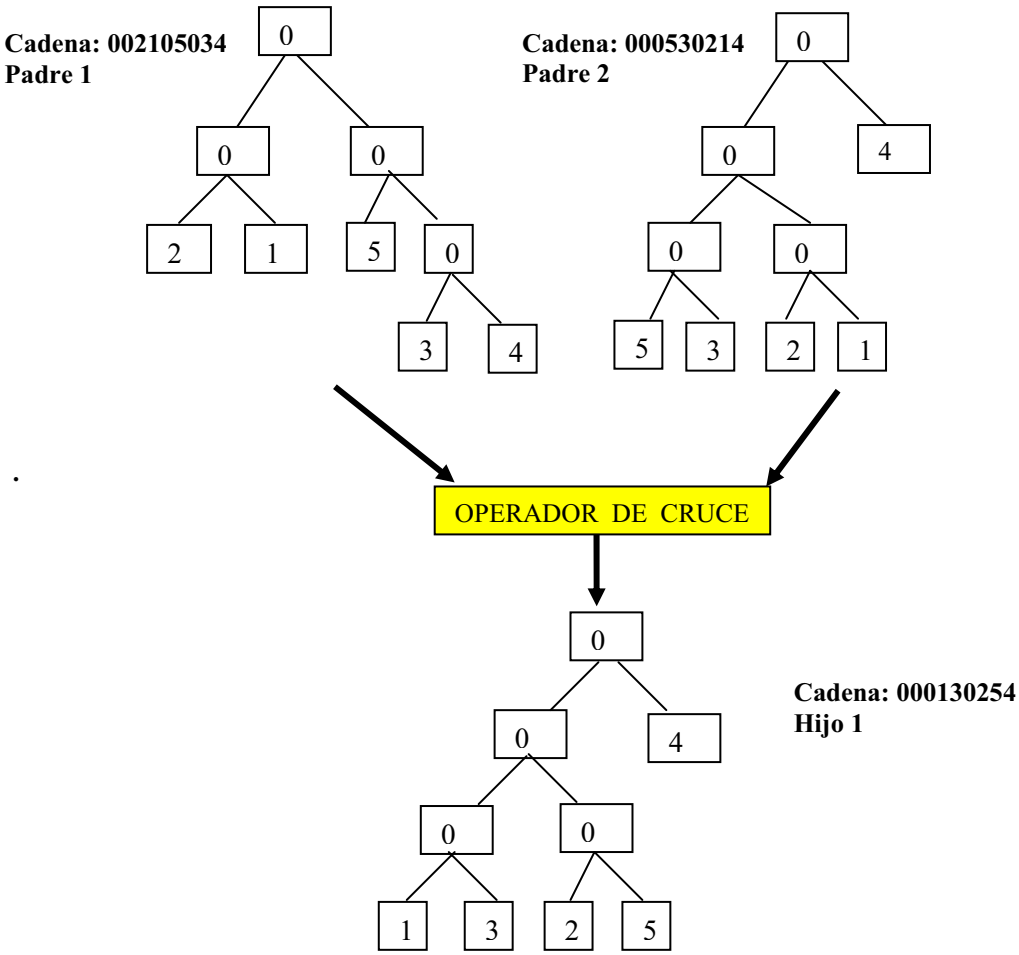

Figura 4.5: Operador de Cruce: Genera 1 nuevo Individuo basado en la mascara de cruce

En la parte experimental, sección 5.4.2, explicamos, el estudio realizado para fijar las tasas de los operadores del AG. En dichos estudios, cada uno de los operadores (tanto de mutación como de cruce), se aplicaron dependiendo de una tasa dada, que fue variando en función de los resultados que se obtenían. El rango que se utilizó para la tasa de cruce estaba entre 0,70 y 0.95, valores altos, orientados a aumentar la frecuencia de aplicación del operador. Finalmente, luego de los experimentos comentados en la sección 5.4.2, hemos optado por usar para el operador de cruce una tasa de 0.80, y haciendo caso a las recomendaciones de Schaffer que afirma que, un valor óptimo para la tasa de mutación es mucho más importante que la tasa de cruce [Schaffer et. al, 1989], se realizó en nuestros experimentos un análisis más detallado de la tasa del operador de mutación, dicho análisis lo establecimos en un amplio rango

desde 0,01 hasta 0,70, pero que estaban orientados a aplicar en diferentes situaciones el operador de mutación, y quedarnos finalmente con el valor de 0.03 para la tasa de mutación debido a que nos proporcionaba mejores resultados en la efectividad del AG, y nos ofrecía un mejor fitness medio entre las ejecuciones realizadas. En el capítulo 5 detallamos todos estos resultados y las pruebas experimentales realizadas.

4.2.5. Métodos de selección

Los métodos de selección son rutinas utilizadas para escoger a un individuo de entre todos los de la población. Son utilizados con los siguientes propósitos:

- Para elegir un individuo de la población.
- Para elegir un nodo del cromosoma.
- Para aplicar un operador durante el proceso de producción.
- Para realizar un intercambio cuando hay varias subpoblaciones (subárboles).

En nuestro sistema se utiliza mayoritariamente el método de selección por torneo, para elegir uno o dos individuos de la población. (un individuo en el caso de la mutación, y dos individuos en el caso del cruce). Para elegir los nodos a procesar con los operadores de producción usamos el método aleatorio (y también para elegir un individuo de dos padres previamente seleccionados por el método del torneo cuando se va a realizar un cruce).

Por lo tanto, en el sistema evolutivo aplicamos estas dos estrategias de selección:

- **La selección por torneo (Tournament):** Un conjunto de individuos es seleccionado de la población. De los individuos elegidos se selecciona el mejor de acuerdo al valor de su función de adaptación (*fitness global*). En nuestro sistema se utiliza el tamaño de torneo igual a dos (2).

- **Aleatorio (Random):** En el cual, la elección se realiza al azar con una probabilidad uniforme.

4.2.6. Factores a considerar en el modelo evolutivo propuesto

Existen seis factores a considerar en nuestro sistema que son importantes para su implementación, de la calidad de estos depende que el sistema realice mejor su propósito. Estos son:

- El conjunto de terminales.
- El conjunto de funciones primitivas.
- La medida de adaptación o *fitness*
- Los parámetros para controlar la ejecución
- Elitismo.
- Los criterios para terminar una ejecución.
- Criterio de selección de documentos iniciales

4.2.6.1. El conjunto de terminales

Es el conjunto de entradas al sistema, conjunto que en nuestro caso estará formado por los documentos a agrupar (representados por los vectores característicos obtenidos del procesamiento de los documentos de la base documental).

4.2.6.2. Funciones de similitud y de distancia

Es la colección de funciones necesarias requeridas por el sistema para solucionar el problema de agrupación de documentos. En nuestro caso, se propone que cada nodo no terminal contenga el *centroide* correspondiente a los documentos que dependan de dicho nodo (tal como comentamos en la sección 4.2.1) para luego, poder medir con dicho centroide las métricas basadas en las medidas de similitud y distancia entre los documentos que conforman el individuo.

Debido a que es necesario utilizar una medida o función que nos permita tener los documentos que sean más cercanos entre sí. En este trabajo se combinan dos métricas, las métricas que se utilizaran serán la medida de la distanciá euclidea para medir la distancia entre los documentos, y la métrica del coeficiente de Pearson No Centrado (tambien llamado coseno del ángulo de 2 vectores) para medir la similitud entre los vectores, que fueron comentadas en la sección 2.2.3.

De esta forma, la metrica de la *distancia euclídea* entre los documentos, está expresada por la siguiente formula aplicada para vectores documentales:

$$Distancia\ (d_i, d_j) = \sqrt{\sum_{k=1}^{m} (t_{ik} - t_{jk})^2} \qquad (4.1)$$

El coeficiente de Pearson No Centrado, (con *"m"* número de ejemplos a experimentar) está expresada por la siguiente formula aplicada a vectores documentales:

$$Similitud\ (d_i, d_j) = \frac{\sum_{k=1}^{m} t_{ik} \cdot t_{jk}}{\sqrt{\sum_{k=1}^{m} t_{ik}^2 \sum_{k=1}^{m} t_{jk}^2}} \qquad (4.2)$$

En nuestra propuesta, cada nodo no terminal del árbol tendrá el valor que corresponde a estas funciones de medida (para aquellos documentos que dependan directamente de dicho nodo y también de los niveles inferiores). Entonces, para formar la estructura de árbol del individuo a evaluar, en cada nodo debemos calcular, además de las coordenadas del *centroide, el valor de* las funciones métricas correspondiente a la Distancia Euclidea, y de la similitud del Coeficiente de Pearson No Centrado.de los dos nodos inferiores, tal como comentamos en la sección 4.2.1 cuando describimos a los individuos de la población.

4.2.6.3. La medida de adaptación o *fitness*

A la hora de crear una nueva generación, o finalizar el agrupamiento de documentos, hemos de poder discriminar los mejores individuos del resto. Esto lo conseguimos con la medida de fitness que nos da la bondad de un individuo. Cada individuo de una generación, y también cada nodo tiene asociado un valor de adaptación o *fitness*. Por ello, el *fitness* de cada individuo lo denominamos "*fitness* global" y el de cada nodo "*fitness* parcial", tal como comentamos en la sección 4.2.1.

Los factores que determinan la adaptación dependen en general del problema concreto al que se aplique el AG. Así, podemos considerar más aptos los individuos que tarden menos tiempo en resolver un problema, que utilicen el menor número de recursos, que sean más similares entre sí, etc.

Con respecto a nuestra propuesta se evalúa la función de adaptación (*fitnessl*) teniendo en cuenta dos criterios: la similitud de los documentos y la distancia entre ellos. De esta manera, el *fitness* propuesto intentará por un lado maximizar la similitud de los documentos y por otro minimizar las distancias entre ellos, siendo definido como el operador siguiente:

$$Fitness = \underset{\substack{i=1,...,N,\\ j=1,...,N,\\ i\neq j}}{MIN} \left[\alpha\ Distancia\ (d_i, d_j) + (1-\alpha) \frac{1}{Similitud\left(d_i, d_j\right)} \right] \qquad (4.3)$$

donde representamos con α al coeficiente que determina el peso que se le otorga bien al valor de la distancia (euclidea), bien al valor de la similitud (varia entre 0 y 1), es decir es el coeficiente de la combinación lineal de la distancia y el reciproco de la similitud, y donde como ya se ha dicho, cada d_i viene representado por su vector característico.

Como se ha comentado anteriormente, las funciones a utilizar para determinar la proximidad seran la distancia euclidea entre los documentos y para la similitud el llamado Correlación de Pearson No centrado (equivalente al coseno del ángulo de los dos vectores) [Castillo José Luis, Fernández del Castillo José R, González León, 2008a] [Castillo José Luis, Fernández del Castillo José R, León González, 2009a]

4.2.6.4. Parámetros para controlar la ejecución del experimento

Para optimizar los agrupamientos y controlar la ejecución del sistema se trabajó sobre los siguientes parámetros:

- Tasa de mutación/Tasa de cruce.
- Criterio de selección.

En la parte experimental (capítulo 5) justificamos el porqué se utilizó una tasa de mutación pequeña (0.03), para ello se realizaron un conjunto de experimentos con la finalidad de encontrar el máximo número de aciertos del AG Se experimentó con una tasa de mutación que variaba entre 0,01 y 0,70 por las razones explicadas en la sección 4.2.4 (operadores de producción) con el fin de analizar mejor el comportamiento del operador de mutación. De la misma forma, luego de muchos experimentos, se analizó la mejor tasa para el operador de cruce, la cual se fijó en 0.80, debido a que nos proporcionaba mejores resultados en promedio, y permitía aplicar con mucha frecuencia los operadores de cruce que se diseñaron para el sistema.

Además para aplicar el criterio de selección se experimentó mayoritariamente el método del torneo, con una tasa de torneo equivalente a 2.

4.2.6.5. Elitismo

En nuestra propuesta aplicamos en cada generación el concepto de "*Elitismo*"; basado en que el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie, y se mantiene intacto hasta que surja otro individuo mejor que él, que lo desplaza.

Como explicamos en la sección 4.2.2, sobre tamaño de la población, utilizamos la estrategia llamada GAVAPS (Genético de variación del tamaño de la población) usando el concepto de edad y tiempo de vida, de modo que si un individuo tiene mejor fitness tendrá mayor tiempo de vida. Por ello, con este enfoque mantenemos el *elitismo* en las siguientes generaciones, garantizando la intensificación del óptimo en el espacio factible, conservando en cada generación, mientras tengan un tiempo de vida adecuado, buenos individuos.

De esta forma, al menos, en cada generación siempre mantenemos el cromosoma del mejor individuo de la generación anterior.

4.2.6.6. El criterio de término ó finalización

En principio, en la naturaleza, el paradigma evolutivo es un proceso sin fin. En los sistemas artificiales, siendo prácticos, debemos establecer un criterio de término que sirva para finalizar el proceso evolutivo.

Si bien existen criterios que se utilizan en los experimentos, tales como el del número máximo de generaciones a experimentar o el de encontrar un individuo de la población que maximice (o minimice) su función adaptación *fitness.* En nuestro trabajo, luego de numerosos experimentos con diferentes números de generaciones tal como comentamos en la sección 4.2.3, establecemos un número de generaciones máximo. Este número se alcanza cuando observamos que el *fitness* global deja de evolucionar con los parámetros propuestos, condición que se cumple cuando evaluamos al menos 4000 generaciones en nuestro sistema.

En la figura 4.6, mostramos de manera esquemática los procesos que se han desarrollado para el modelo propuesto.

Al finalizar todo el proceso se identifica el mejor individuo (mejor cromosoma) que representará el mejor agrupamiento de toda la base documental, debido a valor obtenido por su fitness. En nuestro caso, intentamos siempre minimizar dicho fitness.

De estas forma, recorriendo el árbol en un orden preestablecido, en nuestro caso en *–preorden- obtenemos* los documentos y los grupos del SRI. Como se puede apreciar, al finalizar la evolución, en el nivel inferior de cada nodo del árbol se van agrupando los vectores documentales que corresponden a cada grupo de documentos, constituyendo todos los grupos resultantes al finalizar el proceso los diferentes "clusters" que se obtienen para toda la base documental del SRI.

El sistema evolutivo genera la población inicial, a partir de un conjunto de documentos normalizados [Castillo José Luis, Fernández del Castillo José R, González León, 2008a] y representados por el vector de características de cada documento. Éste será la entrada al sistema y ha sido previamente preprocesado siguiendo el procedimiento propuesto y descrito en el capítulo 3.

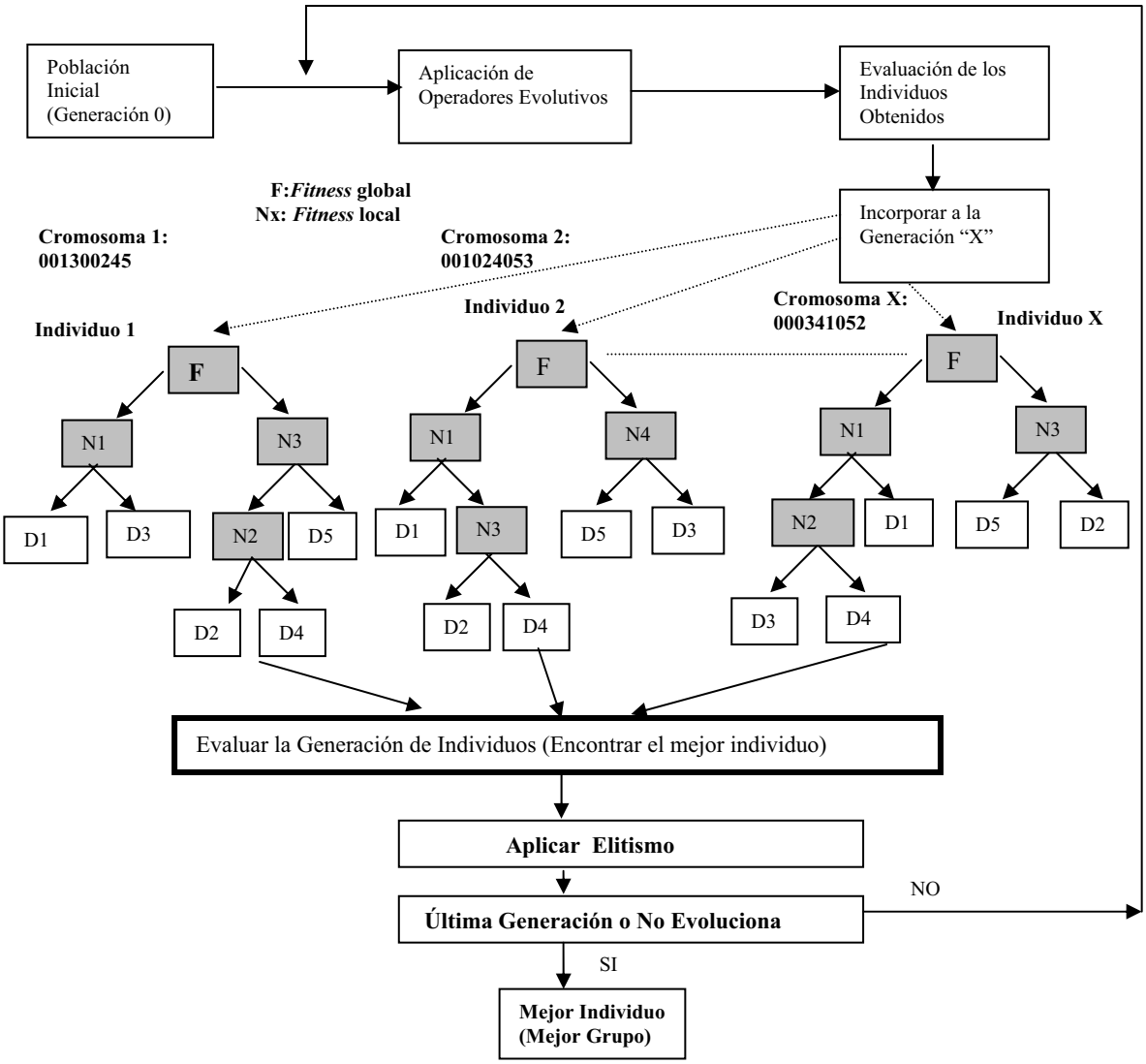

Figura 4.6 Procesos a realizar para el Sistema Evolutivo

4.2.6.7. Criterio de selección de documentos iniciales

Para poder elegir los documentos de la generación 0, establecemos una estrategia con el fin de crear un buen individuo dentro de la población. El objetivo es el de disponer de un conjunto de documentos que presenten algún tipo de relación entre ellos, evitando la presencia de grupos claramente diferenciados.

El proceso de selección está basado en el método de las relaciones completas descrito en [Kowalsky, 1998]. Para ilustrar mejor este preproceso, podemos citar la experimentación del siguiente caso, que parte de vectores de características que representan 5 documentos, donde cada uno de los documentos tiene 8 términos con los siguientes valores (véase como ejemplo la tabla 4.1).

	T1	T2	T3	T4	T5	T6	T7	T8
Doc1	0	4	0	0	0	2	1	3
Doc2	3	1	4	3	1	2	0	1
Doc3	3	0	0	0	3	0	3	0
Doc4	0	1	0	3	0	0	2	0
Doc5	2	2	2	3	1	4	0	2

Tabla 4.1:Ejemplo de un conjunto de datos de entrada (vector de características) mostrando los documentos que se desea seleccionar en la población inicial del algoritmo evolutivo

Aplicando la función de distancia propuesta (distancia euclidea) obtenemos los siguientes datos (ver tabla 4.2):

	Doc1	Doc2	Doc3	Doc4	Doc5
Doc1		7	7,14	5,66	5,29
Doc2	7		6,63	5,92	3,32
Doc3	7,14	6,63		5,39	7,14
Doc4	5,66	5,92	5,39		5,83
Doc5	5,29	3,32	7,14	5,83	

Tabla 4.2: Función distancia obtenida para el vector de características del ejemplo anterior, mostrando la relación existente entre los documentos

Luego podemos establecer un umbral de afinidad (el cual es asignado por el usuario), para seleccionar aquellos documentos que nos interesan de los vectores característicos. Así para nuestro ejemplo, usando un umbral de 6 (60 %) podemos obtener los siguientes vectores de características (ver tabla 4.3). De esta forma cada uno de los documentos que figuran con "1" estarán relacionados.

	Doc1	Doc2	Doc3	Doc4	Doc5
Doc1		0	0	1	1
Doc2	0		0	1	1
Doc3	0	0		1	0
Doc4	1	1	1		1
Doc5	1	1	0	1	

Tabla 4.3: Vectores característicos de los documentos luego de aplicar el umbral de afinidad

Un seudocódigo del algoritmo que establece el tipo de relación para crear el buen individuo es:

```
Comienzo

     Crear un individuo de la población aleatoriamente (cadena)

     Recorrer la cadena y extraer el numero de genes terminales
     encontrados que sean continuos, guardar el resultado en un
     vector de tamaño 2*n (subcadena)

     Recorrer de 1 a 2*n la subcadena

     Extraer avatar de gen terminal par e impar y etiquetarlo
     como un documento

     Si gen terminal impar tiene relacion con gen terminal par
     mantenerlo en la cadena en la misma posición (no cambiar)

     Si No

        Si gen terminal impar no tiene relacion con gen terminal
           par entonces Buscar elemento con quien tiene relacion
           gen terminal impar (buscar en tabla 4.3)

            Si elemento relacionado está no repetido reemplazar
            gen terminal par con dicho elemento (cambiar)

            Si No tiene relacion gen terminal impar en tabla 4.3
               o elemento relacionado esta repetido
               Buscar elemento con quien tiene relacion gen
               terminal par (buscar en tabla 4.3)

               Si elemento relacionado está no repetido
               Reemplazar gen terminal impar con dicho elemento
                                                    (cambiar)

             fin si

      fin si

     Restaurar avarares (cambiadod o nó) de la subcadena en la
     cadena original

     incorporar individuo (cadena) a la población inicial

Fin
```

Trabajamos, por lo tanto con un diagrama equivalente al mapa conceptual de la figura 4.7, que muestra la relación de los documentos, el cual serviría de guía para establecer los documentos que van a ser ubicados en los nodos terminales del individuo, pudiéndose generar de esta forma un buen individuo en la generación inicial que cumpla los objetivos planteados.

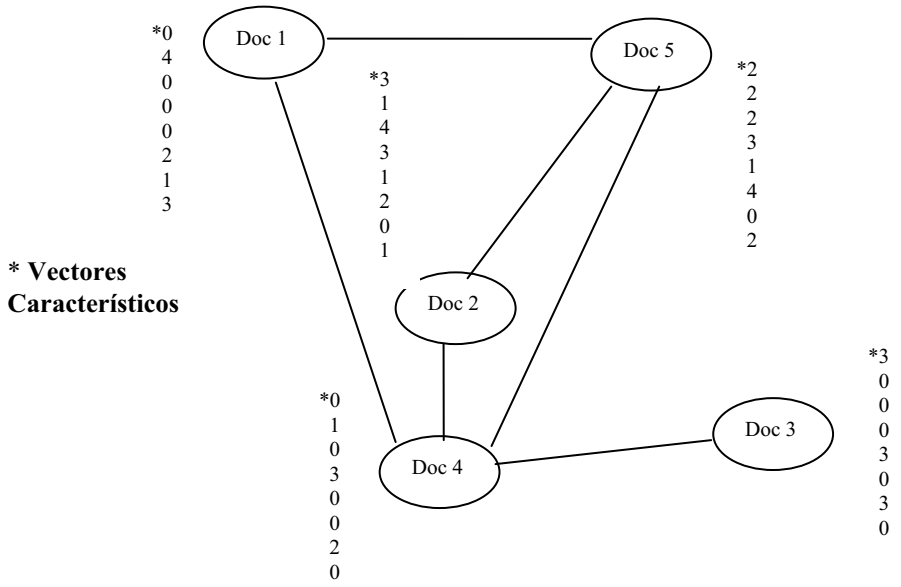

Figura 4.7: Relación de afinidad de los documentos seleccionados de la colección documental

Luego, el resto de la población (individuos) de la generación inicial se crean aleatoriamente, y en las siguientes generaciónes dicha población de individuos evoluciona con los operadores de cruce y mutación descrito en la sección 4.2.4, usando los criterios evolutivos que han sido descritos en este capítulo.

4.3. Especificación funcional del Sistema Evolutivo

El sistema que hemos desarrollado es una aplicación informática que ha sido concebida usando las técnicas de Computación Evolutiva, con la finalidad de poder realizar el agrupamiento no supervisado de los documentos en un SRI; éste sistema fue desarrollado en lenguaje C++. Se trata, pues de la implementación de un sistema informático modular y portable.

Para cumplir los objetivos marcados, se elaboró un modelado de requisitos, diseñando en un catálogo de requisitos las especificaciones funcionales (facilidades que ha de proporcionar el sistema y las restricciones a las que estará sometido) del presente trabajo. Hemos analizado y documentado las necesidades funcionales, que deberán ser soportadas por el sistema a desarrollar; y también establecido las prioridades a tener en cuenta, las cuales proporcionaran un punto de referencia para validar el sistema final. Todos estos puntos los comentamos a continuación.

4.3.1. Catalogo de requisitos

El catalogo de requisitos para el sistema desarrollado, que tiene como fin el de especificar las necesidades funcionales del sistema, tiene como componentes: El catalogo de requisitos funcionales del sistema, el catalogo de requisitos de seguridad, y el catalogo de requisitos de control. Cada uno de estos catalogos está detallado en los anexos a traves de su “identificador de requisitos” (véase tabla 4.4, 4.5 y 4.6), además hemos hecho una descripción de cada uno de sus elementos que lo detallamos en la tabla 4.7.

4.3.2. Descomposición en subsistemas

El sistema se descompone básicamente en cuatro subsistemas (más uno de control de acceso):

Un **subsistema de procesamiento documental**, desde el cual se llevan a cabo todas las técnicas metodológicas propuestas del procesamiento documental, con la finalidad de obtener los vectores característicos de todos los documentos del SRI.

Un **subsistema para crear la población inicial (Generación 0)**, desde el cual se crea el conjunto de individuos que conforman la población inicial, usando para ello el modelo de cromosoma propuesto en la sección 4.2.1, y utilizando un enfoque que permita colocar aleatoriamente todos los documentos del SRI , teniendo siempre individuos sin repetir en la estructura del árbol.

Un **subsistema para procesar el Sistema Evolutivo**, desde el cual se crean las siguientes generaciones de individuos del sistema, a partir de la población inicial, aplicando para ello el operador de *fitness* propuesto en la presente tesis, y los parámetros que describen y controlar el sistema (tasa de mutación, numero de generaciones, tamaño de población, elitismo, etc), así como los operadores de producción propuestos (cruce y mutación) que crean un individuo nuevo a partir de la generación actual.

Un **subsistema de gestión de resultados**, desde el cual se comparan los resultados obtenidos con nuestro sistema con el algoritmo supervisado de agrupamiento (*Kmeans*), tomando para ello el número de grupos que se procesan, e interactuando con el subsistema de procesamiento del sistema evolutivo, para generar los resultados.

En la figura 4.8, mostramos cada uno de los subsistemas desarrollados que se han descritos previamente.

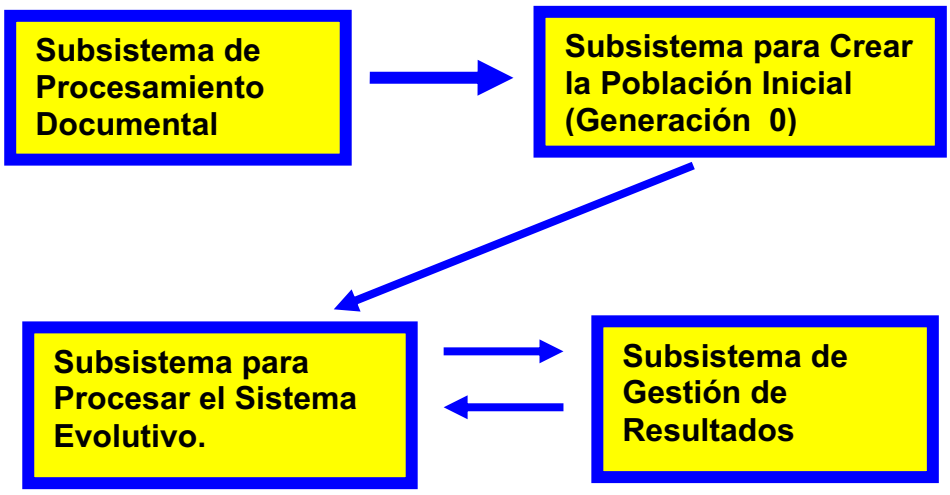

Figura 4.8: Descomposición de Subsistemas del. Sistema Genético

4.4. Modelo del comportamiento del Sistema Evolutivo

4.4.1. Diagrama de casos de uso

En la figura 4.9 mostramos el diagrama de caso de uso de la aplicación que hemos desarrollado para la presente tesis.

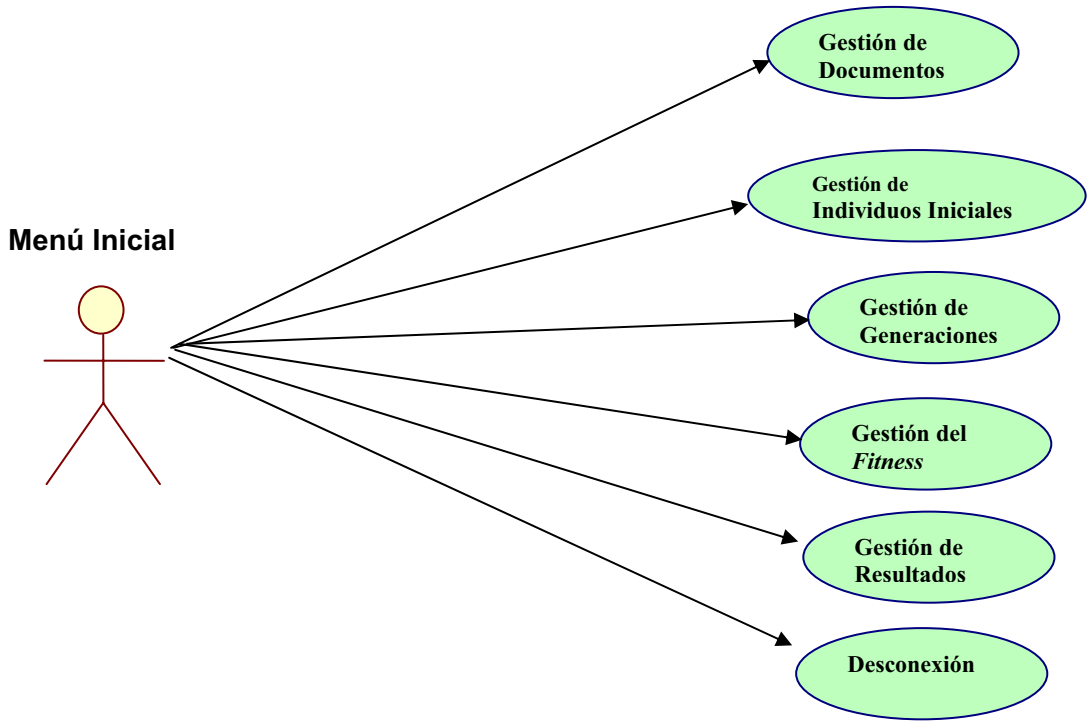

Figura 4.9: Diagrama de Casos de Uso del Sistema Genético

En el diagrama mostramos el comportamiento del sistema al afrontar una tarea o un proceso que gestionemos, y describimos toda la funcionalidad de la aplicación. Los casos de uso están representados por las elipses y los actores están representados por las figuras humanas. De esta forma, en nuestro sistema los actores podrían gestionar documentos, gestionar nuevos individuos de la población, gestionar las generaciones del algoritmo evolutivo, gestionar la evolución de la función de adaptación, gestionar los resultados del sistema, y desconectarse de la aplicación. El detalle y la descripción de cada uno de estos elementos lo presentamos en los anexos de la presente tesis.

En el siguiente capítulo, describiremos detalladamente los resultados experimentales obtenidos con el sistema evolutivo, todos los parámetros utilizados por el sistema, así como también desarrollaremos la descripción del marco de experimentación, y la muestra de datos (recogida de las colecciones Reuters y de la colección de los editoriales del diario El Mundo) que hemos utilizado, para luego comentar los grupos obtenidos con nuestra propuesta y realizar las comparaciones con otro método clásico de agrupamiento a fin de validar nuestra propuesta experimental.

Capítulo 5

Resultados obtenidos con el Sistema Evolutivo

Como hemos visto en el capítulo anterior, se ha desarrollado un sistema, basado en un algoritmo genético (AG) capaz de efectuar agrupaciones en conjuntos de documentos. En el presente capítulo, se presentan y discuten los resultados obtenidos de dicho sistema, al procesar tanto la colección documental *Reuters 21578*, como la colección documental obtenida de los editoriales en español de los años 2006 y 2007 del diario *El Mundo*.

En primer lugar, haremos una pequeña introducción planteando de forma más extendida el problema, detallaremos el conjunto de muestras de datos que utilizamos para las pruebas reales, y el entorno de experimentación de nuestro sistema. A continuación, explicaremos el procedimiento aplicado para la determinación de los valores óptimos para los diferentes parámetros del algoritmo, y fijaremos el método de evaluación de las soluciones obtenidas. Luego, realizaremos un análisis pormenorizado de los resultados de cada una de las muestras de documentos, centrándonos en el mejor resultado obtenido por nuestro algoritmo (*fitness* óptimo) y en el comportamiento de su valor medio (*fitness* medio). La utilidad del sistema propuesto en el presente trabajo queda demostrada a nuestro parecer, mediante la comparación de sus resultados con los obtenidos con un algoritmo de agrupamiento supervisado muy robusto, como es "*Kmeans*".

Finalmente, la calidad de nuestros resultados se contrastará con la agrupación real de cada colección documental, tratada en el capitulo 3.

5.1. Antecedentes

El sistema desarrollado utiliza como entrada los vectores característicos de los documentos procesados previamente con la metodología propuesta y desarrollada en el capítulo 3, sistema que se basa en un algoritmo evolutivo capaz de agrupar documentos atendiendo al conjunto de términos lematizados que caracterizan a cada documento, lo que permite incrementar su potencialidad y versatilidad.

En nuestro sistema, un individuo aparece, bien como un árbol que contiene los documentos en sus hojas y la proximidad entre ellos se hace patente al aparecer en ramas próximas. Dicho individuo es una cadena de símbolos que representa la sucesión de reglas de producción que generan dicho árbol. Lo hacemos evolucionar mediante la aplicación de una función de adaptación *fitness* que permitirá determinar la afinidad o cercanía existente entre los documentos. Se define una función de *fitness* a ser minimizada con el objetivo de agrupar por afinidad y similitud, mediante la aplicación de operadores genéticos sobre los individuos distintos.

.

Debido a que nuestro sistema es del tipo evolutivo, usa parámetros de ajuste que afectan el rendimiento del algoritmo. Por lo tanto tenemos que empezar definiendo el entorno de experimentación a aplicar, y comentar el conjunto de datos que utilizamos para las pruebas reales.

5.2. Conjunto de datos utilizado para las pruebas reales

Para las pruebas reales hemos utilizado documentos de la colección *Reuters 21578* [Lewis, 1997], tomando las distribuciones que más dispersión de datos presentan (distribución 21, distribución 2, distribución 20, distribución 8), y documentos de la colección de editoriales del diario El Mundo.

La razón de usar Reuters 21578, es que esta colección constituye uno de los estándares de *facto* dentro del dominio de la categorización automática de documentos, y como tal es utilizada por numerosos autores de la materia como *"piedra de toque"* [Steinbach, Karypis, Kumar, 2000], [Zhao, Karypis, 2001]. Esta colección se ha descrito, procesado, y comentado en el capítulo 3, está compuesta por 21578 documentos, distribuidos en 22 archivos, donde cada documento tiene 5 campos de categorización distintos llamados Valor Bursátil, Organización, Persona, Lugar y Tema.

Además hemos utilizado la colección en español de los editoriales del diario *El Mundo* de los años 2006 y 2007, la cual ha sido clasificada manualmente en la etapa de procesamiento documental.

De esta forma, de ambas colecciones hemos tomado muestras representativas de grupos de documentos que hemos tipificado según su tamaño en: "*muy pocos documentos, pocos documentos, muchos documentos y bastantes documentos*", cuándo evaluamos 20, 50, 80 y 150 documentos respectivamente, y algunos experimentos se han realizado con un número mayor de documentos. El objetivo fue procesar el AG con muestras de diferentes tamaños para verificar su funcionamiento.

Para constatar la validez de nuestro trabajo, hemos utilizado las distribuciones que presentan mayor dispersión. Previamente, en el Capitulo 3 se ha elaborado la tabla 3.7 en la que para la colección *Reuters 21578* se han detallando cada una de las distribuciones. Por lo tanto, disponemos del número de documentos de cada una de

ellas y la cantidad de grupos que contiene, siendo posible realizar la selección de documentos atendiendo a su tasa de dispersión, entendiéndose por tal al cociente entre la cantidad de documentos y el número de grupos de cada colección, dato que presentamos comparativamente en la tabla 5.1.

Distribución	Cantidad de Documentos	Número de Grupos (K)	Dispersión
Reuters2-000.SGML	406	31	13,09
Reuters2-001.SGML	436	34	12,82
Reuters2-002.SGML	**455**	**27**	**16,85**
Reuters2-003.SGML	450	32	14,06
Reuters2-004.SGML	446	33	13,51
Reuters2-005.SGML	498	36	13,83
Reuters2-006.SGML	511	38	13,44
Reuters2-007.SGML	475	35	13,57
Reuters2-008.SGML	**452**	**32**	**14,12**
Reuters2-009.SGML	457	33	13,84
Reuters2-010.SGML	425	31	13,70
Reuters2-011.SGML	485	39	12,43
Reuters2-012.SGML	469	37	12,67
Reuters2-013.SGML	199	29	6,86
Reuters2-014.SGML	179	27	6,62
Reuters2-015.SGML	441	35	12,60
Reuters2-016.SGML	488	39	12,51
Reuters2-017.SGML	398	41	9,70
Reuters2-018.SGML	328	39	8,41
Reuters2-019.SGML	316	32	9,87
Reuters2-020.SGML	**402**	**26**	**15,46**
Reuters2-021.SGML	**273**	**16**	**17,06**

Tabla 5.1: Tasas de dispersión de las distribuciones Reuters 21578

Así, en las pruebas reales, utilizamos la distribución Reuters 21, Reuter 2, Reuter 20 y Reuters 8, porque son las que contienen *la* mayor tasa de dispersión entre todas las distribuciones. Se han tomado muestras de documentos de estas colecciones (diferentes cantidades de documentos), y analizado el comportamiento del algoritmo a fin de maximizar el número de aciertos, el promedio de aciertos entre las diferentes pruebas, y finalmente confeccionar gráficos con la evolución del sistema.

Las muestras fueron tomadas seleccionando los documentos de entre dos categorías de cada colección documental, tomando aquellas categorías que contienen la mayor cantidad de documentos (categorías *Acq* y *Earn*)**.** De esta forma, todas las muestras tomadas para los experimentos de la distribución Reuters 21 se corresponden con los documentos que pertenecen a las categorías 1 y 7. Las muestras que corresponden a los documentos de la distribución Reuters 2, pertenecen a las categorías: 1 y 6, las muestras que pertenecen a la Distribución 20 corresponden a los documentos que pertenecen a las categorías 1 y 9 **,** y finalmente las muestras que son de la distribución 8 se corresponden con los documentos de las categorías 1 y 7. El criterio seguido fue siempre utilizar aquellas colecciones Reuters que presenten una mayor dispersión (en nuestro caso las cuatro primeras).

Para la colección de los editoriales del diario *El Mundo*, aplicamos el mismo criterio, utilizando las principales categorías que tiene el tesauro Eurovoc [Eurovoc, 2006]. Para ello fue necesario realizar previamente una clasificación manual en cada uno de los editoriales que componen la colección (1402 editoriales), según el método descrito en el capítulo 3, con la finalidad de comparar posteriormente los resultados del algoritmo con los datos reales.

La tabla 5.2 muestra ordenados alfabéticamente los resultados obtenidos con la colección de los editoriales. Se puede apreciar de qué forma están agrupados todos los 1402 editoriales de la colección en español que se procesaron para los años 2006 y 2007 según las categorías del tesauro Eurovoc.

Categoría Eurovoc	*Número de Documentos*
Agricultura, Silvicultura y Pesca	2
Asuntos Financieros	9
Asuntos Sociales	161
Ciencia	3
Comunidades Europeas	22
Derecho	302
Educación y Comunicación	30
Empresa y Competencia	8
Energía	20
Geografía	4
Industria	19
Intercambios Económicos y Comerciales	5
Medio Ambiente	11
Organizaciones Internacionales	9
Producción, Tecnología e Investigación	1
Relaciones Internacionales	219
Trabajo y Empleo	13
Transportes	18
Vida Económica	36
Vida Política	510

Tabla 5.2: Número de documentos de la colección de editoriales del periódico El Mundo agrupadas según categorías de Eurovoc, luego de la clasificación manual realizada en la etapa de procesamiento documental.

Así, para los experimentos realizados con la colección de editoriales del diario El Mundo, tomamos documentos de entre dos categorías (*Grupos a los que pertenece cada uno de los documentos de la colección de editoriales del diario El Mundo)*. En este caso, para realizar las pruebas usamos las categorías que presentan un mayor volumen de documentos (ver tabla 5.2), y seguimos el mismo criterio utilizado para la colección Reuters.

5.3. Entorno de experimentación

Dentro del entorno de experimentación debe de existir un usuario que proporcione los documentos que se quiera agrupar. El papel del usuario que aporta documentos estará representado por las muestras de documentos de cada colección documental representado por sus vectores de características.

La figura 5.1 representa el entorno documental que utilizamos para el desarrollo de las pruebas reales. Es importante destacar que, al contrario que en los algoritmos del tipo supervisado, en los que se necesita indicar como parámetro el número de grupos que se desea obtener, nuestro algoritmo evolucionará hasta encontrar la estructura más apropiada, formando él mismo los grupos. De esta forma, nuestro algoritmo formará los documentos de una manera no supervisada, demostrando que es posible el uso de un sistema evolutivo para la agrupación no supervisada de documentos como una alternativa a otras técnicas.

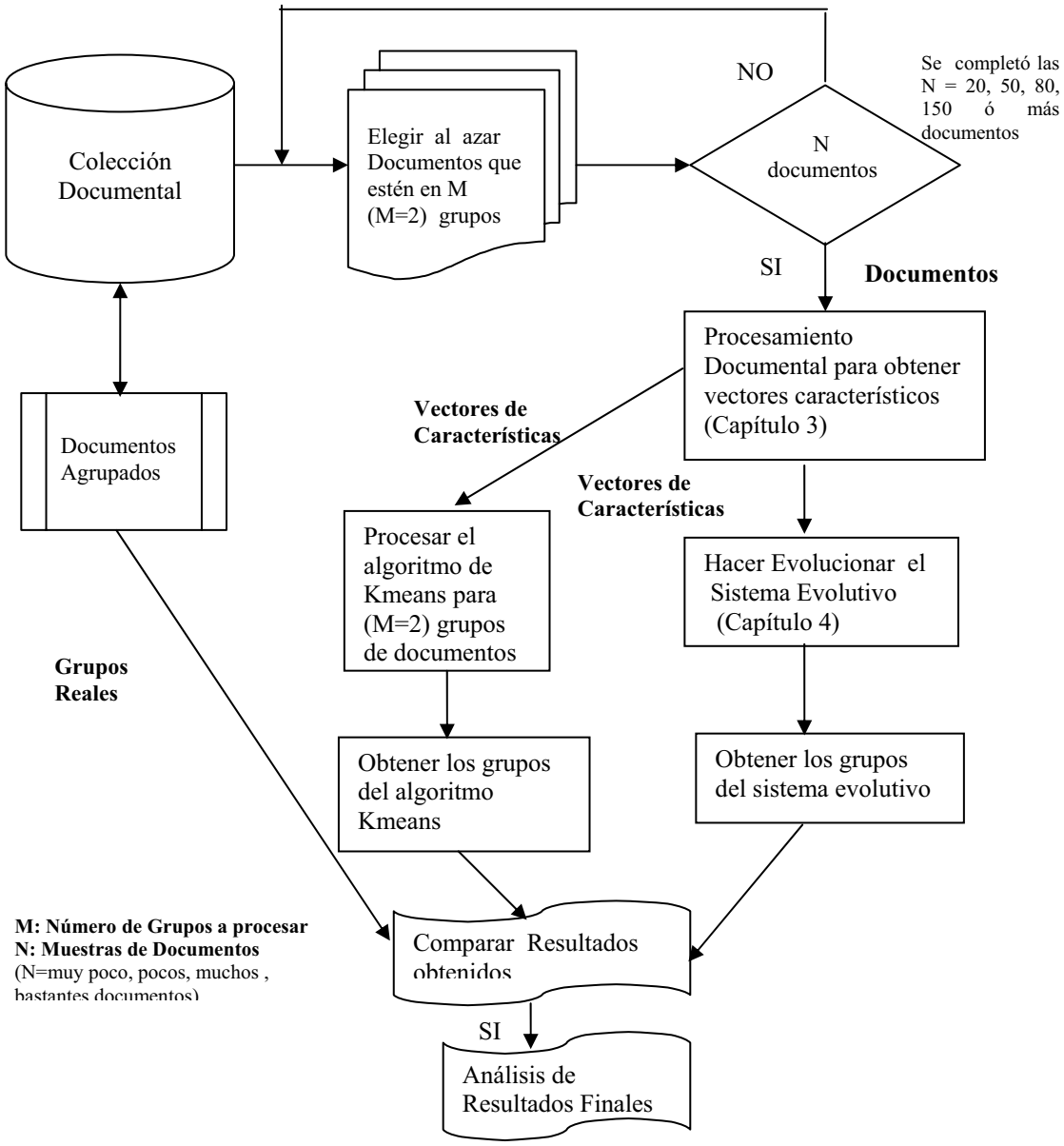

Figura 5.1: Entorno experimental empleado en las pruebas con el Sistema Evolutivo

A continuación vamos a analizar otros aspectos relacionados con la metodología experimental. Debido a la naturaleza de simulación del sistema evolutivo, su funcionamiento es pseudo aleatorio. Esto se traduce en la necesidad de realizar varias ejecuciones, con distintas semillas como entrada para el generador de números aleatorios, hasta alcanzar la solución óptima.

De este modo, los experimentos con el algoritmo evolutivo, se realizaron efectuando cinco ejecuciones sobre cada una de las muestras tomadas de las colecciones experimentales. La salida del experimento es el mejor *fitness* obtenido y la velocidad de convergencia ó la generación donde se encuentra el mejor *fitness*. Finalmente, como medida de la calidad del algoritmo se puede atender a la mejor solución obtenida y la robustez, es decir, la calidad promedio del algoritmo (promedio de los valores de las distintas soluciones obtenidas). En los experimentos previos que hemos realizado usando la colección Reuters hemos comprobado que nuestro AG funciona correctamente con diferentes muestras de documentos, y que el comportamiento del *fitness* medio en varias tiradas del algoritmo es correcto, tal como

lo podemos apreciar en la figura 5.2. Nótese en la figura la forma como evoluciona el *fitness* medio con cada una de las muestras de documentos procesadas.

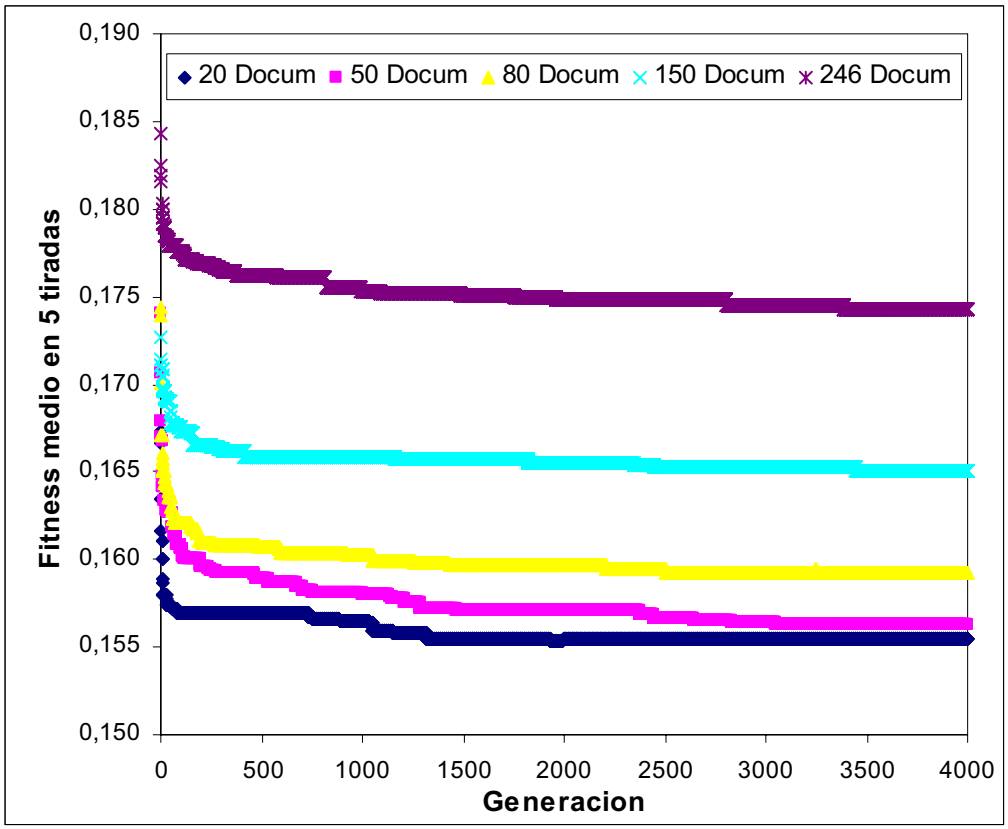

Figura 5.2: Evolución del Fitness del AG, para distintos números de documentos

En los experimentos dentro de nuestro entorno de trabajo, se utilizaron muestras de documentos recogidas al azar de entre "*muy pocos, pocos, muchos y bastantes*" documentos, con el requisito de que perteneciesen solamente a dos de las categorías de la distribución Reuters o de editoriales en español. Cada una de las muestras se procesó con las 5 semillas diferentes, y cada uno de los resultados se comparó con el método "*Kmeans*". [Castillo José Luis, Fernández del Castillo José R, León González, 2009a] [Castillo José Luis, Fernández del Castillo José R, León González, 2009b]

Por su tamaño, y por la influencia que durante los experimentos tienen las pequeñas variaciones de los parámetros en el comportamiento del algoritmo evolutivo, aparece como un factor crítico la elección de los valores de los parámetros a utilizar. Por ello para su elección hemos atendido a la sensibilidad de los posibles indicadores de rendimiento del sistema frente a las modificaciones del valor de alguno de ellos.

Específicamente atendimos a la evolución del número de aciertos (grupos correctos a los que pertenece cada documento dentro de la distribución), y la evolución de la función de adaptación "*fitness*" utilizada.

5.4. Parámetros del algoritmo evolutivo

Los algoritmos evolutivos poseen diversos parámetros, los cuales deben ser cuidadosamente elegidos para obtener un buen desempeño y evitar problemas tales como, por ejemplo, la convergencia prematura. En nuestro caso, después de muchas pruebas iniciales, hemos optado por escoger la mayor parte de los parámetros usualmente utilizados en el campo de la computación evolutiva, tal como lo explicamos en la sección 4.2, y analizar más detenidamente algunos de ellos que detallamos a continuación.

De esta forma, el tamaño de la población se ha fijado en 50 individuos en todos los experimentos realizados. En nuestro caso, la longitud l de nuestro cromosoma es igual a:

$$2 * ndoc - 1 \qquad (5.1)$$

siendo *ndoc* igual al número de documentos a agrupar.

En cuanto al tamaño de torneo (véase sección 4.2.5), se ha optado por el valor 2, ya que el torneo binario ha demostrado muy buen rendimiento en una gran cantidad de aplicaciones de los AEs [Bäck, 1994] [Goldberg, Deb, 1991].

Tras las pruebas con el AG, el número máximo de generaciones del sistema se ha fijado en 4000, tal como lo comentamos en la sección 4.2.3, con el fin de validar la pronta o tardía convergencia del algoritmo con los parámetros más comunes. Sin embargo éste parámetro podría variar, dependiendo de la convergencia del algoritmo; más aún el algoritmo evolutivo finaliza mucho antes, cuándo deja de evolucionar el *fitness*.

En lo que respecta al número de términos lematizados para representar los vectores característicos de cada uno de los documentos, lo hemos seleccionado usando el método de procesamiento NZIPF comentado en el capítulo 3 de la presente tesis.

Además, hemos establecido un límite llamado "umbral de profundidad" para los árboles que representan a los individuos (representación de los documentos), que depende del número de documentos a procesar, y que en el caso de "muy *pocos"* y "pocos" toma el valor de 7, y para el caso de "*muchos" y "bastantes"* toma el valor de 10, con la finalidad de que el individuo generado por el algoritmo no tenga una profundidad exagerada en comparación con la cantidad de documentos.

Para fijar el valor del coeficiente α que gobierna el peso que se otorga a los valores de la distancia y del inverso de la similitud entre documentos, realizamos un estudio detallado buscando el mayor número de aciertos del AG, y de esta forma obtener resultados más robustos. Por ello, realizamos un conjunto de pruebas, para encontrar el mejor valor, usando las dos distribuciones Reuters más dispersas con los parámetros más comunes del AG. Para encontrar dicho valor, realizamos los experimentos que detallamos a continuación:

5.4.1. Estudios para fijar el valor de α en el Algoritmo Evolutivo.

En primer lugar, utilizamos la distribución Reuters 21, por ser la que ofrece una mayor dispersión entre sus documentos, y aplicamos el AG, tomando muestras de 20, 50, 80, 150 y 250 documentos, haciendo variar el valor de α en cada una de las pruebas, y analizamos los resultados obtenidos por la ejecución del algoritmo con los parámetros más usuales, intentando siempre comprobar la efectividad del AG.

En la tabla 5.3, mostramos los resultados obtenidos para determinar el valor de α tomando las diferentes muestras de documentos examinadas. La tabla muestra el número de documentos procesados, el valor de α utilizado, la generación dónde converge el AG, el mejor *fitness* obtenido, el promedio de dicho *fitness* en la última meseta de la convergencia del AG, el número de aciertos del AG, y el porcentaje de efectividad del AG

Documentos	α	Generación	Mejor *Fitness*	Promedio del *Fitness* Medio Meseta Final	Aciertos	Efectividad (%)
20	0,75	1436	0,25291551	0,46489675	15	75,0
20	0,80	1592	0,20298477	0,47026890	16	80,0
20	0,85	2050	0,15255487	0,24504483	17	85,0
20	0,90	3694	0,15266796	0,25909582	17	85,0
20	0,95	1520	0,15319261	0,24596829	17	85,0
50	0,75	3476	0,25290429	0,28744261	35	70,0
50	0,80	3492	0,20285265	0,27862528	36	72,0
50	0,85	3355	0,15312467	0,29128428	36	72,0
50	0,90	2256	0,15318358	0,28347470	36	72,0
50	0,95	2222	0,15345986	0,27863789	36	72,0
80	0,75	3049	0,25704660	0,36871676	61	76,2
80	0,80	1371	0,20782096	0,33303315	61	76,2
80	0,85	2131	0,15784449	0,34447947	62	77,5
80	0,90	1649	0,15815252	0,32398087	62	77,5
80	0,95	2986	0,17796620	0,36009861	61	76,2
150	0,75	2279	0,26194273	0,29866150	91	60,6
150	0,80	1273	0,20636391	0,22933754	93	62,0
150	0,85	3257	0,15468909	0,27518240	94	62,6
150	0,90	1136	0,25482251	0,28218144	94	62,6
150	0,95	2452	0,25456480	0,26788158	91	60,6
250	0,75	3617	0,25754282	0,31144435	120	48,0
250	0,80	3274	0,20844638	0,25112189	121	48,4
250	0,85	3066	0,15805103	0,19299910	121	48,4
250	0,90	2343	0,20634355	0,20432140	121	48,4
250	0,95	2047	0,25541276	0,27844937	120	48,0

Tabla 5.3: Resultados de las pruebas realizadas con el AG, tomando diferentes muestras de documentos con la distribución 21 de la colección Reuters, a fin de determinar el mejor valor para α .

Para mostrar los resultados obtenidos, hemos definido el *factor de aciertos* de la siguiente forma:

$$a_i - a_0 \qquad (5.2)$$

donde a_i corresponde al número de aciertos medios obtenidos por el AG, y a_0 es el mínimo valor de aciertos medio obtenidos por dicho AG. De esta forma, en la figura 5.3, mostramos el factor de acierto frente a los valores de α. y podemos apreciar que

los mejores valores en cada una de las muestras se presentan cuando aplicamos un valor de α próximo al valor de 0,85. Estos resultados se producen porque el valor del mejor *fitness* toma el mínimo valor al utilizar dicho valor de α, a partir del cual la contribución de la métrica de la inversa de similitud no compensa el de la métrica de distancia, lo que origina que el valor del *fitness* no tienda a estabilizarse, tal y como lo apreciamos en la figura 5.4, que muestra el comportamiento del mejor *fitness* al variar los valores de α con diferentes muestras de documentos.

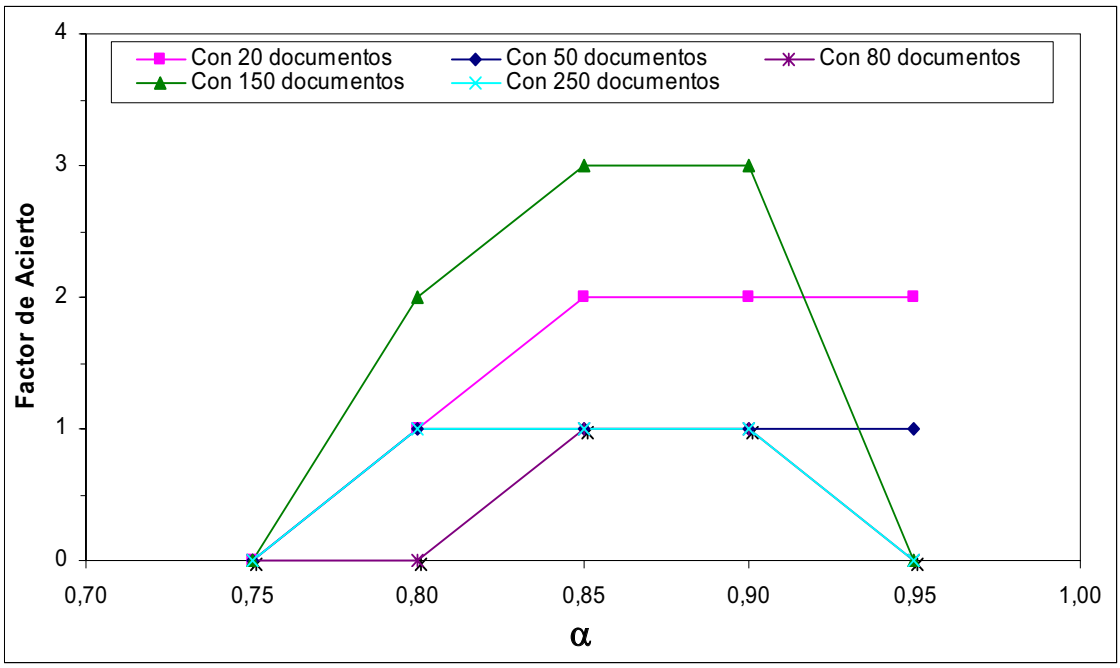

Figura 5.3: Dependencia del número de aciertos con el valor de α en el AG, tomando diferentes muestras de documentos de la distribución 21 de la colección Reuters.

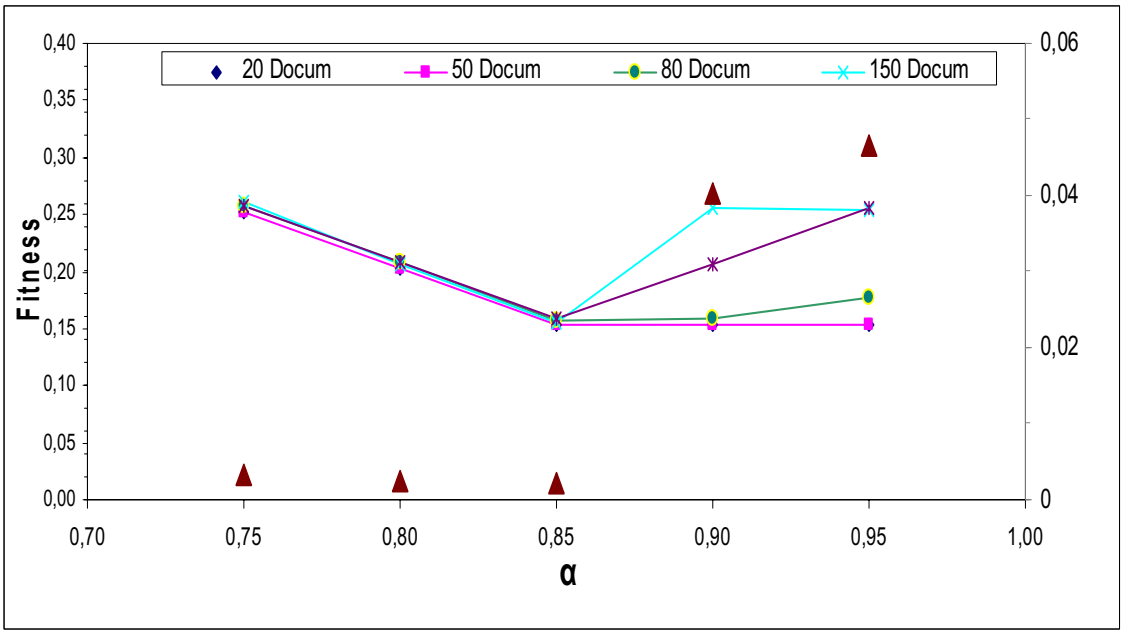

Figura 5.4: Mejores* Fitness frente a valores de *α para diferentes muestras de documentos de la colección Reuters: Distribución 21. Se puede apreciar que hay un incremento de la dispersión del fitness para valores superiores a 0,85, debido a la mayor contribución de la distancia euclídea que hace más insensible al Fitness para encontrar los agrupamientos.

Los resultados obtenidos, nos sugieren que un valor de α próximo al 0,85, nos proporciona mejores resultados, debido a que nos ofrece una mayor efectividad en lo que respecta al número de aciertos, y un mejor comportamiento del *fitness* del algoritmo. Para comprobar dichos resultados, realizamos las mismas pruebas con la siguiente distribución Reuters, que presenta mayor dispersión (distribución 2), siguiendo la misma metodología de trabajo y encontrando los resultados que comentamos a continuación.

La tabla 5.4, muestra los resultados obtenidos tomando diferentes muestras de documentos de la distribución 2, en dicha tabla la distribución de las columnas es la misma que la de la tabla 5.3.

Documentos	**α**	**Generación**	**Mejor *Fitness***	**Promedio del *Fitness* Medio Meseta Final**	**Aciertos**	**Efectividad (%)**
20	0,75	1882	0,25657612	0,46887308	17	85,0
20	0,80	2159	0,20512714	0,37527900	17	85,0
20	0,85	3018	0,15544757	0,28422983	18	90,0
20	0,90	977	0,15789137	0,30080442	18	90,0
20	0,95	667	0,25608850	0,31734338	17	85,0
50	0,75	3179	0,25751152	0,36179680	37	74,0
50	0,80	3188	0,20701337	0,28227682	42	84,0
50	0,85	2611	0,15551472	0,23235160	42	84,0
50	0,90	1344	0,16745702	0,26252360	42	84,0
50	0,95	3220	0,23627790	0,34559823	38	76,0
80	0,75	2154	0,25775189	0,31864548	68	85,0
80	0,80	2250	0,20772510	0,25126033	68	85,0
80	0,85	3475	0,15995932	0,20721564	69	86,2
80	0,90	2165	0,22811717	0,23927100	68	85,0
80	0,95	2457	0,26719844	0,38102333	67	83,7
150	0,75	1831	0,25513591	0,27398708	76	50,6
150	0,80	3344	0,20527565	0,22466592	77	51,3
150	0,85	2889	0,15546623	0,17053468	81	54,0
150	0,90	1697	0,20540662	0,21815454	77	51,3
150	0,95	939	0,24493634	0,26606182	76	50,6
250	0,75	3320	0,25454058	0,27878739	103	41,2
250	0,80	1040	0,20484433	0,22619679	105	42,0
250	0,85	1591	0,15386229	0,17511908	105	42,0
250	0,90	2346	0,30383620	0,31952528	99	39,6
250	0,95	2841	0,35419544	0,36668063	98	39,2

Tabla 5.4: Resultados de las pruebas realizadas con el AG, tomando diferentes muestras de documentos de la distribución 2 de la colección Reuters, a fin de determinar el mejor valor para α .

En la figura 5.5, mostramos el factor de acierto frente al valor de α para las diferentes muestras de documentos procesadas para la colección 2 de Reuters. Apreciamos un comportamiento que se asemeja al obtenido con la primera colección Reuters que se procesó, y que confirma un comportamiento del AG más satisfactorio, en lo que respecta al promedio de aciertos si se utiliza un valor de α cercano al 0,85.

Finalmente, en la figura 5.6 mostramos la evolución del valor del *fitness* para los diferentes valores de α, con distinto número de documentos. Se observa un valor mínimo a partir de α igual a 0,85. Podemos apreciar en la misma gráfica que, a partir

de dicho punto el *fitness* presenta mayor dispersión en sus valores. Se pone de manifiesto un comportamiento estable del *fitness* para aquellos valores de α inferiores a 0,85, a partir del cual la dispersión se desestabiliza, lo que se explica debido a que existe una mayor contribución por parte de la distancia euclidea a partir de dicho valor, que hace que el fitness sea más insensible para encontrar los agrupamientos de la colección.

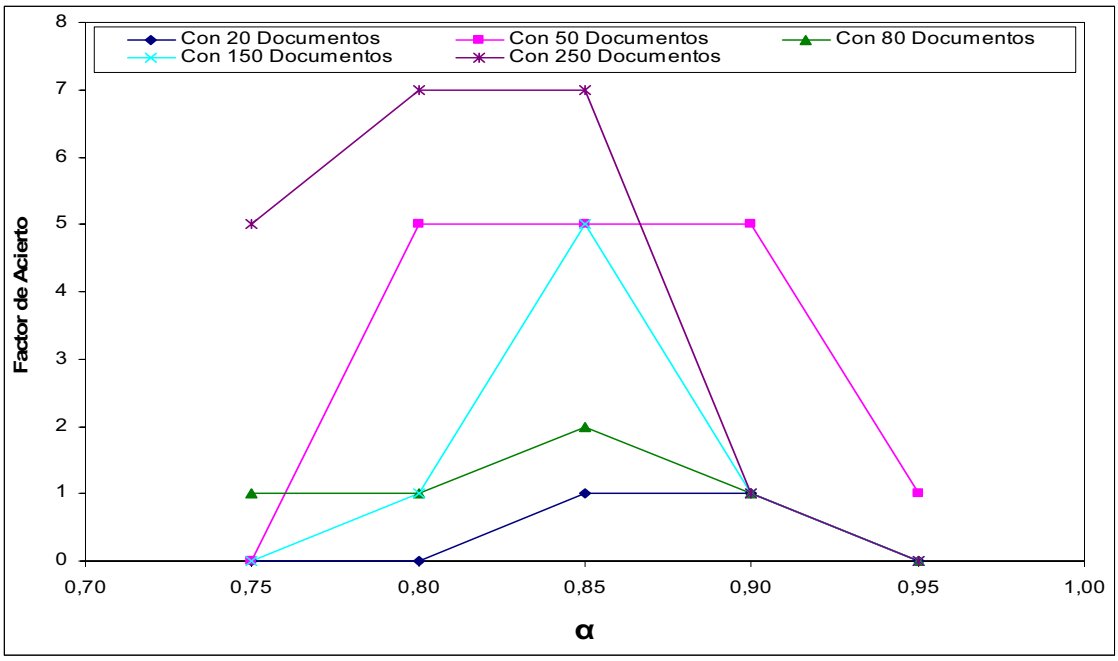

Figura 5.5: Factor de aciertos frente al valor de α en el AG, tomando diferentes muestras de documentos de la distribución 2 de la colección Reuters.

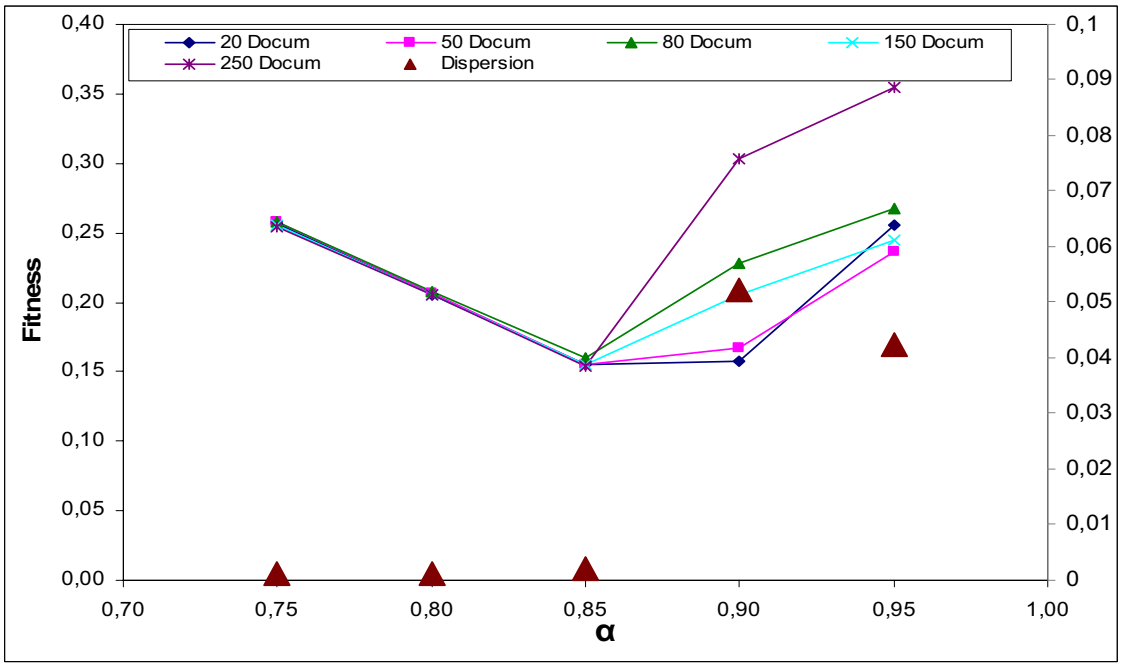

Figura 5.6: Mejores Fitness frente a valores de α para diferentes muestras de documentos de la colección Reuters: Distribución 2. Se puede apreciar que hay un incremento de la dispersión del fitness para valores superiores a 0,85, debido a la mayor contribución de la distancia euclídea que hace más insensible al Fitness para encontrar los agrupamientos.

De todas las pruebas realizadas con las diferentes colecciones, concluimos que el mejor valor de α para el funcionamiento del algoritmo es uno cercano a 0,85; que será empleado en los sucesivos experimentos.

Con ello queda nuestra función de *fitness* perfectamente caracterizada agregando en un único valor operadores tan distintos como son la distancia euclídea y el coeficiente de correlación de Pearson (similitud basada en el Coseno), comentada en la sección 4.2.6.3, y explicitada en la ecuación 4.3.

En la que α es el coeficiente de la combinación lineal de la distancia y el reciproco de la similitud: que tomará el valor de 0,85.

A continuación se va a hacer un estudio para determinar el valor de la tasa del operador de mutación, tomando de forma arbitraria para el operador de cruce, tasas con valores altos, a partir de 0,70, que permitan aplicar con mayor frecuencia el operador de cruce que usamos para el AG. De esta forma, hemos adoptado una estrategia que permite hacer un estudio detallado de la tasa del operador de mutación siguiendo las recomendaciones de Schaffer, que indicaba la importancia de tener un valor óptimo para la tasa de mutación es mucho más importante que la tasa de cruce [Eiben A, Hintending R, Michalewicz Z., 1999] [Schaffer et. al, 1989].

5.4.2. Estudios destinados a fijar el valor de la tasa del operador de mutación y la tasa del operador de cruce.

Empezamos realizando un análisis del comportamiento del sistema al variar el valor de la tasa del operador de mutación en un amplio rango para cubrir todas las posibles situaciones. Sobre las zonas que muestren indicios de incremento en la efectividad se realizarán experimentos con mayor detalle. En todos los casos se usarán valores altos del operador de cruce. Durante los experimentos se utilizan diferentes muestras de documentos tomadas de la distribución que tiene mayor dispersión en la colección documental Reuters. De esta forma, para la tasa del operador de mutación se analizó una gran variedad de valores: 0,01; 0,03; 0,05; 0,07; 0,1; 0,3; 0,5; 0.7; que nos permitía aplicar el operador de mutación del AG en diferentes circunstancias, y estudiar su comportamiento.

Para el estudio para la determinación del valor óptimo de la tasa del operador de cruce, se rastreó el intervalo que va desde 0,70 hasta 0,95; valores altos seleccionados para aplicar con más frecuencia el operador de cruce para el AG. Como índice de la calidad del valor del operador se atendió al número de aciertos del AG

De esta forma podemos apreciar en la tabla 5.5, los resultados obtenidos al empezar a evaluar el AG en el rango de valores estudiados, tomando muestras de 20 documentos. La tabla nos muestra en cada caso, el número de documentos procesados, la tasa de mutación, la tasa de cruce, la generación de convergencia, el valor del mejor *fitness* en la ejecución del AG, la Media del *fitness* medio en cada generación, el promedio del *fitness* medio de la meseta final donde converge el algoritmo, el número de aciertos que se obtiene con el AG y, finalmente, el porcentaje de efectividad obtenido por el algoritmo.

Doc.	TM	TC	Convergencia	Mejor *Fitness*	Promedio *Fitness* Medio Meseta Final	Aciertos	Efectividad (%)
20	0,01	0,70	3513	0,15435567	0,44181661	**17**	85
20	0,01	0,75	2713	0,15435567	0,35619737	**17**	85
20	0,01	**0,80**	**1114**	0,15435567	0,41293335	**17**	85
20	0,01	0,85	2611	0,15435567	0,36994836	**17**	85
20	0,01	0,90	2265	0,15435567	0,37426916	**17**	85
20	0,01	0,95	1396	0,15435567	0,36811227	**17**	85
20	0,03	0,70	3250	0,15430475	0,35640938	**17**	85
20	0,03	0,75	2800	0,15430475	0,32627855	**17**	85
20	0,03	0,80	1029	0,15435567	0,38475738	**17**	85
20	0,03	0,85	1044	0,15435567	0,32637615	**17**	85
20	0,03	0,90	767	0,15435567	0,38877133	**17**	85
20	0,03	0,95	646	0,15435567	0,38871117	**17**	85
20	0,05	0,70	1031	0,15594958	0,39170072	16	80
20	0,05	0,75	1344	0,15575242	0,38218165	16	80
20	0,05	**0,80**	1030	0,15435567	0,33035065	**17**	85
20	0,05	0,85	1298	0,15435567	0,31088238	17	85
20	0,05	0,90	1073	0,15435567	0,44963602	17	85
20	0,05	0,95	1020	0,15435567	0,39078092	17	85
20	0,07	0,70	1438	0,15599434	0,35937313	16	80
20	0,07	0,75	2080	0,15494680	0,35404563	16	80
20	0,07	**0,80**	**1402**	0,15265232	0,38218199	**17**	85
20	0,07	0,85	1240	0,15575242	0,38218247	16	80
20	0,07	0,90	1413	0,15435567	0,32992295	17	85
20	0,07	0,95	1144	0,15505367	0,32991810	16	80
20	0,10	0,70	1110	0,15591274	0,55032558	16	80
20	0,10	**0,80**	1440	0,15435567	0,38861757	**17**	85
20	0,10	0,90	1050	0,15435567	0,35685088	17	85
20	0,30	0,70	2522	0,15494680	0,43857683	16	80
20	0,30	**0,80**	1885	0,15435567	0,33282334	**17**	85
20	0,30	0,90	920	0,15503567	0,41753123	16	80
20	0,50	0,70	2162	0,15508141	0,35335700	16	80
20	0,50	**0,80**	1869	0,15435567	0,34037162	**17**	85
20	0,50	0,90	704	0,15525577	0,33829636	16	80
20	0,70	0,70	1046	0,15537822	0,31844010	16	80
20	0,70	**0,80**	1094	0,15435567	0,28384413	**17**	85
20	0,70	0,90	1069	0,15525567	0,28878307	16	80

Tabla 5.5: Resultados de las pruebas realizadas variando la tasa de mutación y la tasa de cruce, con muestras de 20 documentos de la colección documental.

Para poder validar los resultados obtenidos con la muestra de 20 documentos, a continuación calculamos el promedio de todos los aciertos obtenidos, a fin de garantizar la robustez del método de evaluación, lo cual lo reflejamos en la tabla 5.6.

Documentos	Tasa de mutación	Aciertos medios
20	0,01	**17,00**
20	0,03	**17,00**
20	0,05	16,67
20	0,07	16,33
20	0,10	16,67
20	0,30	16,33
20	0,50	16,33
20	0,70	16,33

Tabla 5.6: Resultados promedios del número de aciertos del AG con muestras de 20 documentos, variando la tasa de mutación.

Podemos apreciar en la gráfica 5.7 que trabajando con muestras de 20 documentos, obtenemos un mejor comportamiento en lo que respecta al número de aciertos del AG cuando disponemos una tasa muy baja para el operador de mutación (en este caso de 0,01 y 0,03). Nótese claramente donde se obtienen los mejores resultados, obteniéndose diferentes mesetas en la gráfica, siendo la mejor aquella que se corresponde con una tasa de mutación muy baja, y que los peores resultados en promedio se obtienen a medida que esta aumenta.

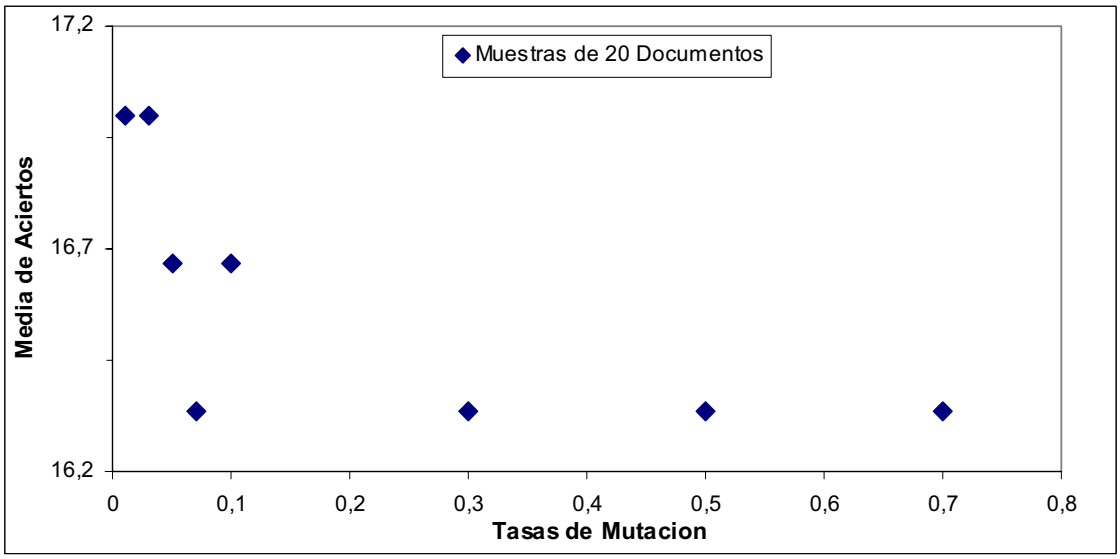

Figura 5.7: Promedio de los aciertos obtenidos para muestras de 20 documentos, al variar la tasa de mutación.

Para caracterizar el comportamiento de la tasa de mutación incrementamos el número de documentos de las muestras. En este caso utilizamos muestras de 80 documentos, y repetimos el mismo procedimiento aplicado anteriormente. Así mismo mediante el cálculo del promedio de los aciertos obtenidos con todas las muestras experimentales determinamos la robustez del algoritmo propuesto, cuyos valores son recogidos en la tabla 5.7.

Doc.	TM	TC	Convergencia	Mejor *Fitness*	Promedio *Fitness* Medio Meseta Final	Aciertos	Efectividad (%)
80	0,01	0,70	1197	0,16074207	0,21201974	49	61,2
80	0,01	**0,75**	2640	0,15930850	0,20950355	**51**	63,7
80	0,01	**0,80**	3151	0,15825307	0,20516351	**51**	63,7
80	0,01	**0,85**	1261	0,15910486	0,20060103	**51**	63,7
80	0,01	0,90	1704	0,15984718	0,21030326	50	62,5
80	0,01	0,95	3112	0,15960713	0,20319328	50	62,5
80	0,03	0,70	1023	0,15973501	0,20998906	49	61,2
80	0,03	**0,75**	1025	0,15946682	0,20885988	**51**	63,7
80	0,03	**0,80**	2500	0,15890940	0,21481804	**51**	63,7
80	0,03	**0,85**	3090	0,15951459	0,20694977	**51**	63,7
80	0,03	**0,90**	2631	0,15905178	0,21237022	**51**	63,7
80	0,03	0,95	3045	0,15971028	0,20584497	50	62,5
80	0,05	0,70	1019	0,16107313	0,20469933	47	58,7
80	0,05	0,75	2659	0,15970308	0,21307143	50	62,5
80	0,05	**0,80**	927	0,15887681	0,21223457	**51**	63,7
80	0,05	**0,85**	867	0,15919079	0,21421284	**51**	63,7
80	0,05	**0,90**	2638	0,15953038	0,20977618	**51**	63,7
80	0,05	0,95	829	0,15955894	0,21192493	50	62,5
80	0,07	0,70	3145	0,15985765	0,21373320	50	62,5
80	0,07	**0,75**	905	0,15883902	0,20602748	**51**	63,7
80	0,07	**0,80**	3486	0,15881098	0,20266374	**51**	63,7
80	0,07	0,85	2479	0,15988471	0,20321703	50	62,5
80	0,07	0,90	3639	0,15980052	0,20445449	50	62,5
80	0,07	0,95	692	0,15980766	0,20561340	50	62,5
80	0,10	0,70	2897	0,16069631	0,21170332	48	60,0
80	0,10	**0,80**	1554	0,15874341	0,20010583	**51**	63,7
80	0,10	**0,90**	1526	0,15936861	0,20260913	**51**	63,7
80	0,30	0,70	2842	0,16010743	0,20542835	49	61,2
80	0,30	**0,80**	1838	0,15882667	0,20689212	**51**	63,7
80	0,30	0,90	802	0,16003654	0,20348640	48	60,0
80	0,50	0,70	2087	0,16020106	0,20206286	49	61,2
80	0,50	**0,80**	3022	0,15852371	0,20302330	**51**	63,7
80	0,50	0,90	3317	0,16093935	0,20659338	49	61,2
80	0,70	0,70	3426	0,16087746	0,20293174	48	60,0
80	0,70	**0,80**	2050	0,15881654	0,20126015	**51**	63,7
80	0,70	0,90	2537	0,15952776	0,20153298	50	62,5

Tabla 5.7: Resultados de las pruebas realizadas variando la tasa de mutación, y la tasa de cruce con muestras de 80 documentos de la colección documental.

En la tabla 5,7, se muestran los valores medios de los aciertos para las diferentes tasas de mutación empleadas, dato que reflejamos extractado en la tabla 5.8, cuyos resultados se disponen gráficamente en la figura 5.8. En ella se aprecia de nuevo un mejor comportamiento (número de aciertos medio) cuando aplicamos una tasa de mutación muy baja. Así, para valores cuya tasa de mutación son inferiores a 0,1 se puede apreciar un incremento del índice de eficacia que se hace todavía más ostensible para la tasa de mutación igual a 0,03. Nótese en la gráfica, que el comportamiento medio de los aciertos es muy similar al de la muestra anterior, siendo mejor al aplicar una tasa de mutación menor a 0,1, y que aplicando tasas superiores a 0,1 decae sustancialmente la media de aciertos del algoritmo.

Documentos	Tasa de Mutación	Aciertos medios
80	0,01	50,33
80	**0,03**	**50,50**
80	0,05	50,00
80	0,07	50,33
80	0,10	50,00
80	0,30	49,33
80	0,50	49,67
80	0,70	49,67

Tabla 5.8: Resultados Medios del número de aciertos del AG con muestras de 80 documentos, para diferentes tasas de mutación.

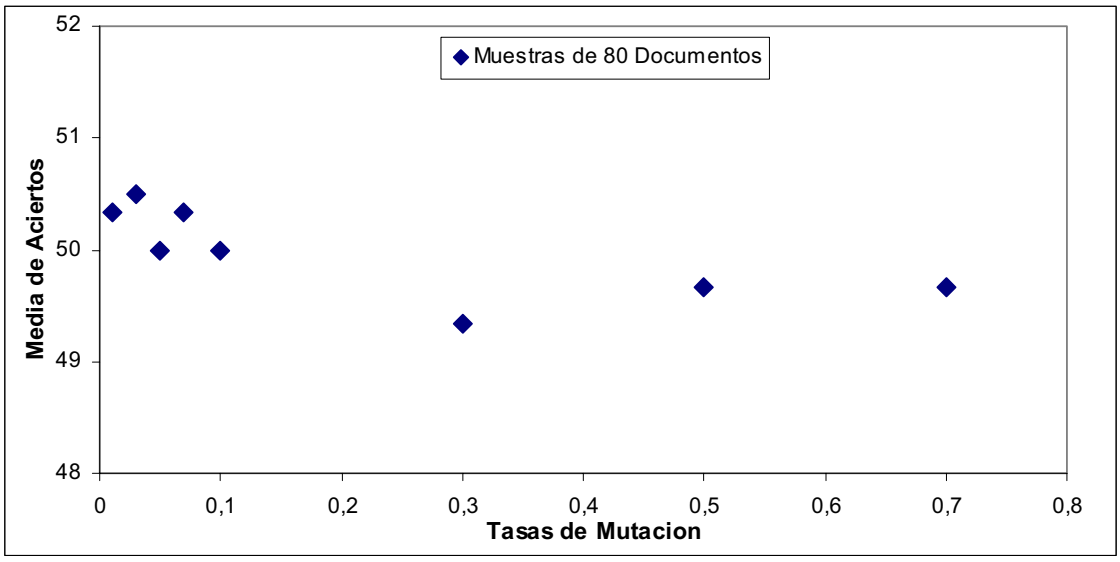

Figura 5.8: Promedio de los aciertos obtenidos para muestras de 80 documentos, al variar la tasa de mutación.

Para concluir, y tratando de corroborar los resultados anteriores, ampliamos la muestra de documentos. En éste caso tomamos muestras de bastantes documentos (150), y repetimos las pruebas experimentales. En la tabla 5.9, que adopta la misma estructura que las tablas anteriores, mostramos los resultados que obtuvimos con el AG con esta nueva muestra de documentos.

Doc.	TM	TC	Convergencia	Mejor *Fitness*	Promedio *Fitness* Medio Meseta Final	Aciertos	Efectividad (%)
150	0,01	**0,70**	1715	0,15785694	0,17403733	**79**	52,6
150	0,01	**0,75**	1682	0,15808014	0,17595423	**79**	52,6
150	0,01	**0,80**	2491	0,15890410	0,19911949	**79**	52,6
150	0,01	0,85	3138	0,16001647	0,17376193	77	51,3
150	0,01	0,90	661	0,16049940	0,17449949	77	51,3
150	0,01	0,95	2394	0,16003694	0,20301227	77	51,3
150	0,03	0,70	933	0,15773532	0,22178118	78	52,0
150	0,03	0,75	2303	0,15711131	0,17363894	79	52,6
150	0,03	**0,80**	2342	0,15601392	0,17495469	**80**	53,3
150	0,03	0,85	2950	0,15734472	0,17405530	79	52,6
150	0,03	0,90	945	0,15724355	0,17335366	79	52,6
150	0,03	0,95	3005	0,15700515	0,20953828	79	52,6
150	0,05	0,70	878	0,15828529	0,17414638	77	51,3
150	0,05	0,75	1086	0,15813713	0,17420443	78	52,0
150	0,05	**0,80**	1861	0,15724229	0,17461903	**79**	52,6
150	0,05	0,85	1175	0,15771947	0,17413689	78	52,0
150	0,05	0,90	1036	0,15774222	0,17540739	78	52,0
150	0,05	0,95	2916	0,15797834	0,21109289	78	52,0
150	0,07	0,70	2977	0,15897719	0,17479428	77	51,3
150	0,07	0,75	1019	0,15893285	0,17392619	77	51,3
150	0,07	**0,80**	3181	0,15771538	0,17346115	**78**	52,0
150	0,07	0,85	1633	0,15901125	0,17280352	76	50,6
150	0,07	0,90	2119	0,15934882	0,17341372	76	50,6
150	0,07	0,95	3396	0,15992226	0,17367093	76	50,6
150	0,10	0,70	1349	0,16512765	0,17293418	71	47,3
150	0,10	**0,80**	2687	0,15858357	0,17416110	**77**	51,3
150	0,10	0,90	2516	0,16590869	0,17162319	68	45,3
150	0,30	0,70	1330	0,16581335	0,17054837	71	47,3
150	0,30	**0,80**	2729	0,15737759	0,17084167	**79**	52,6
150	0,30	0,90	2817	0,15780166	0,17428955	78	52,0
150	0,50	0,70	986	0,16594233	0,17093260	71	47,3
150	0,50	**0,80**	901	0,15859682	0,17171996	**77**	51,3
150	0,50	0,90	1012	0,16576138	0,17133172	71	47,3
150	0,70	0,70	770	0,16509250	0,17145197	71	47,3
150	0,70	**0,80**	862	0,15842086	0,17020913	**77**	51,3
150	0,70	0,90	1177	0,16554335	0,16987958	70	46,6

Tabla 5.9: Resultados de las pruebas realizadas, variando la tasa de mutación, y la tasa de cruce con muestras de 150 documentos de la colección documental.

De la misma forma que en los experimentos anteriores, con el fin de constatar la robustez de las pruebas realizadas, calculamos el promedio de los aciertos obtenidos, información que mostramos en la tabla 5.10, y cuyos resultados se muestran en la figura 5.9. En ella se vuelve a poner de manifiesto un mejor comportamiento cuando se utiliza una baja tasa para el operador de mutación. Obsérvese que un valor de 0,03 nos proporciona mejores resultados, aún trabajando con una muestra con un número mucho mayor de documentos. Nótese, que el comportamiento de la gráfica tiene una

tendencia muy similar al de las anteriores, lo que de alguna forma corrobora la validez de los valores obtenidos en las pruebas experimentales.

Documentos	Tasa de mutación	Aciertos medios
150	0,01	78,00
150	0,03	**79,00**
150	0,05	78,00
150	0,07	76,67
150	0,10	72,00
150	0,30	76,00
150	0,50	73,00
150	0,70	72,67

Tabla 5.10: Resultados promedios del número de aciertos del AG con muestras de 150 documentos, para diferentes tasas de mutación.

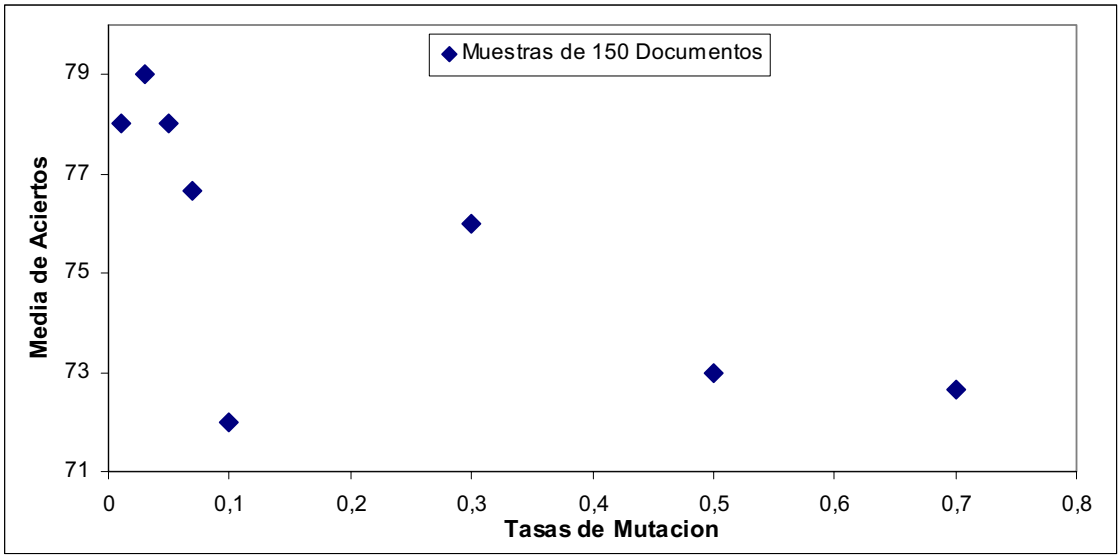

Figura 5.9: Promedio de aciertos obtenidos para muestras de 150 documentos (bastantes documentos), variando la tasa de mutación.

Finalmente los resultados de todas las pruebas realizadas aparecen resumidos en las gráficas 5.10 y 5.11, que muestran los valores del coeficiente de aciertos para las distintas tasas de mutación aplicadas. Mientras que la primera gráfica muestra todo el rango de variación utilizado para la tasa de mutación, en la segunda se muestran de forma ampliada los mismos datos para valores pequeños, de hasta 0,1. Podemos ver claramente que existe un mejor comportamiento, en promedio, para la tasa de mutación de 0,03, independientemente de la muestra de documentos que utilicemos. En la figura 5.12 se aprecia cómo, para diferentes muestras de documentos, el *fitness* medio da valores mínimos para una tasa de mutación de 0,03, deteriorándose la efectividad del AG cuando nos alejamos de este valor.

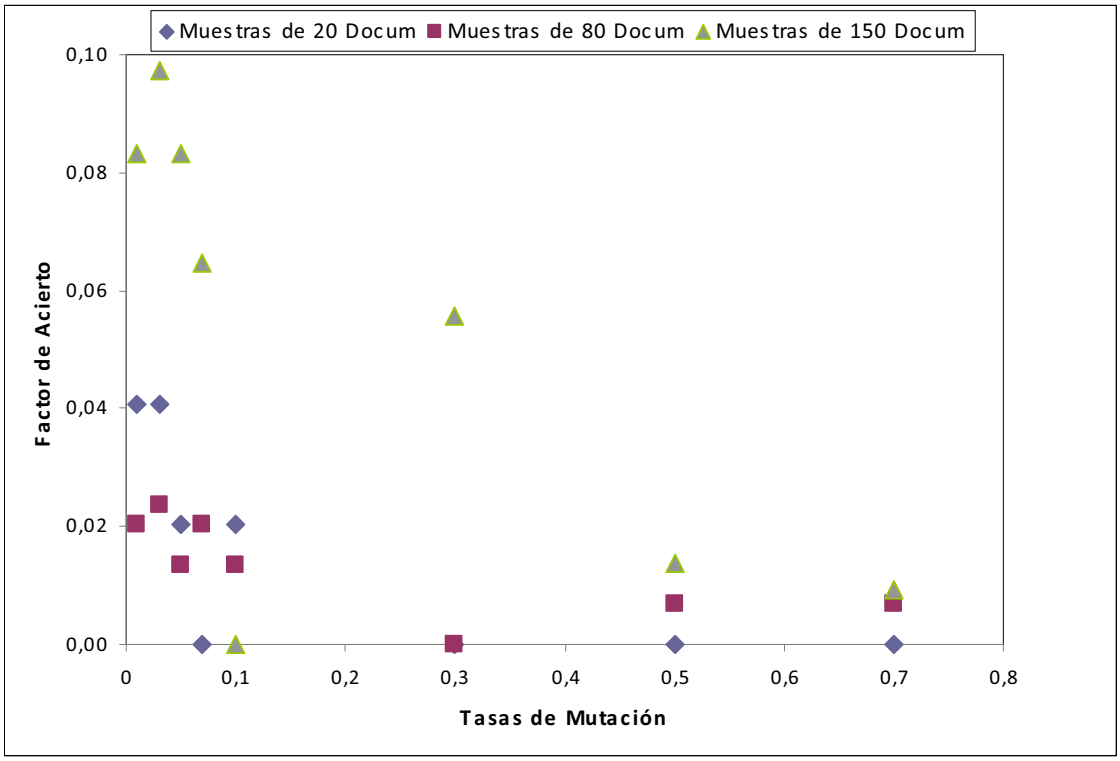

Figura 5.10: Factor de aciertos frente a la tasa de mutación para todas las muestras de documentos. Se aprecia un incremento para valores inferiores a 0,1.

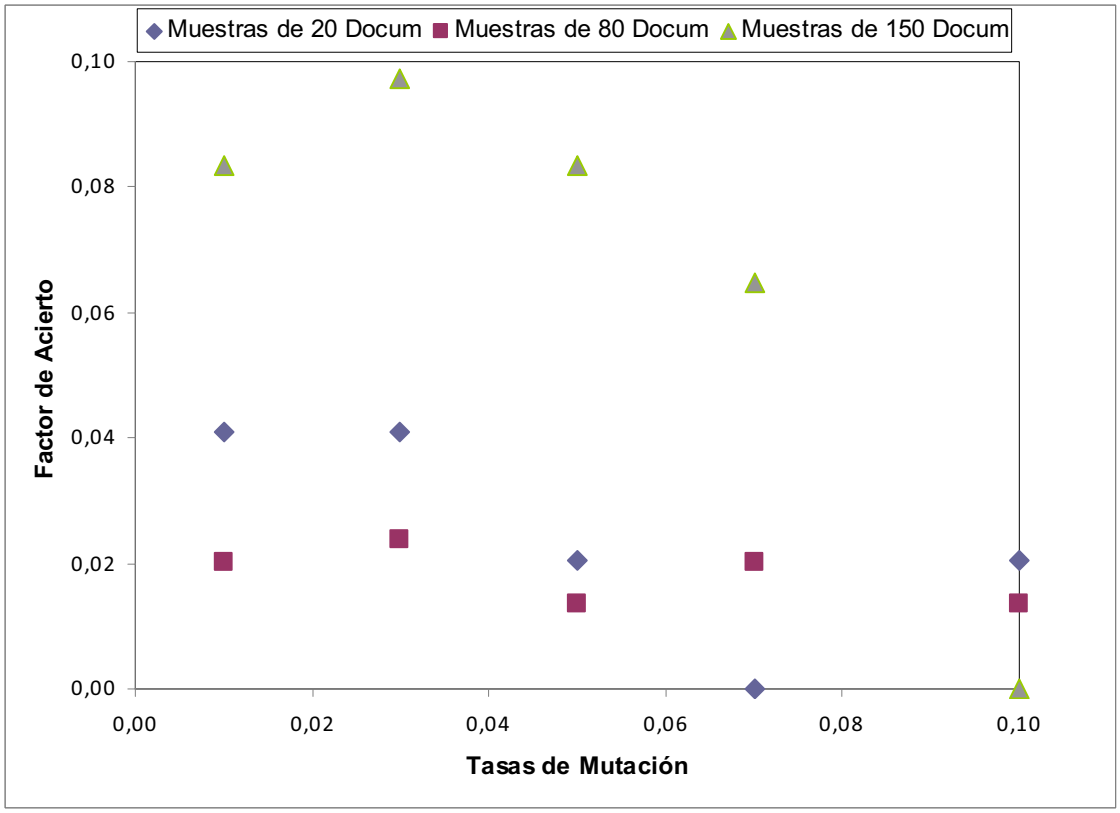

Figura 5.11:Factor de acierto frente a la tasa de mutación para todas las muestras de documentos procesadas, presentando de forma ampliada las tasas de mutación en el rango de 0,01 y 0,1.

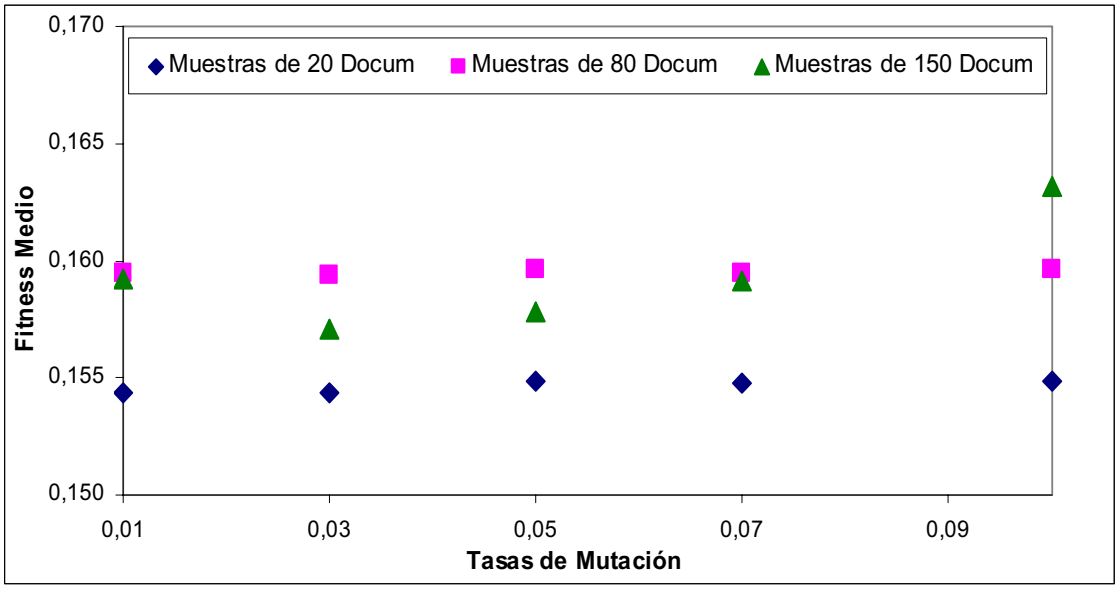

Figura 5.12: Fitness medio frente a las tasas de mutación del AG (entre 0 y 0,1) con diferentes muestras de documentos procesadas.

Habiéndose encontrado el valor óptimo de la tasa de mutación en 0,03, analizamos la evolución del *fitness* medio de nuestro AG, fijando dicha tasa y variando la tasa del operador de cruce a lo largo de todas las generaciones, cuya gráfica mostramos en la figura 5.13. Nótese en la gráfica, que existe una velocidad de convergencia del *fitness* más rápida al aplicar la tasa de cruce de 0,80. Podemos concluir que existe un mejor comportamiento del AG cuando se aplica dicha tasa de mutación.

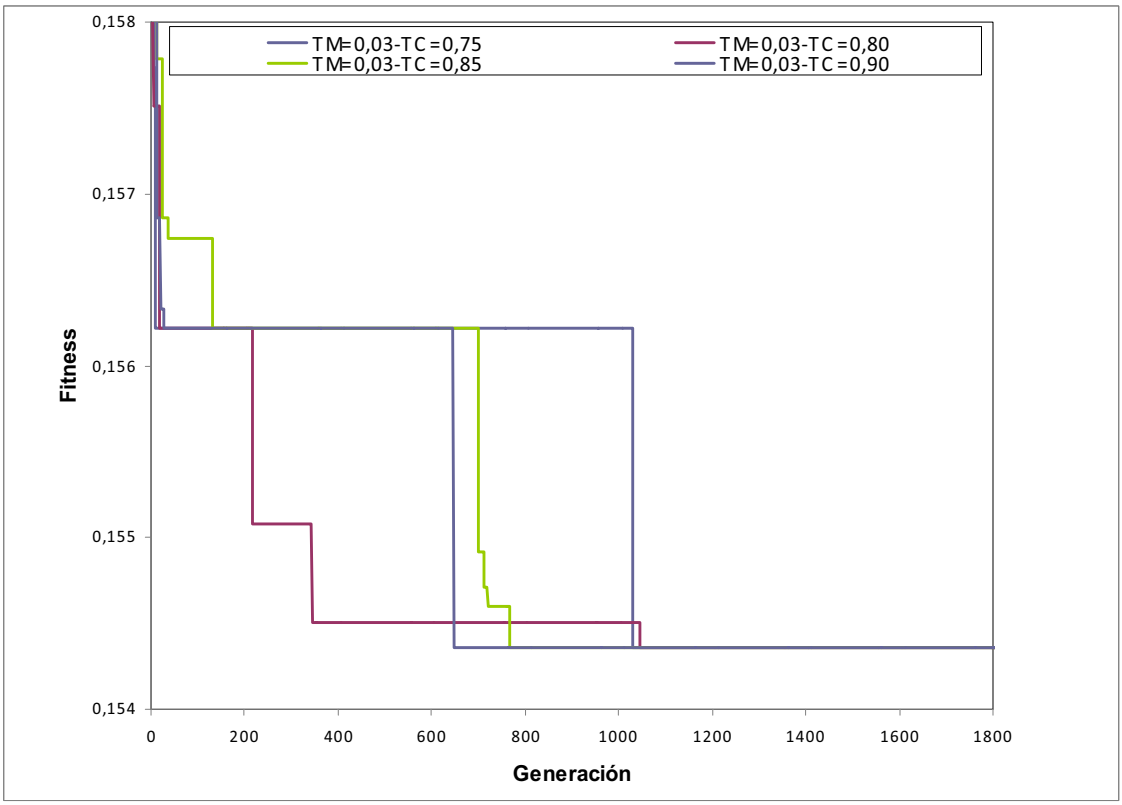

Figura 5.13: Evolución del fitness variando la tasa del operador de cruce del AG para una tasa de mutación fija igual a 0,03. Se aprecia que valores de la tasa de cruce cercanos a 0,80 mejoran la convergencia del AG.

La tabla 5.11, comprende los valores promedio de los aciertos obtenidos con el AG al variar la tasa del operador de cruce en cada una de las muestras de documentos.

Podemos analizar la incidencia del operador de cruce sobre los resultados finales. Las gráficas comparativas muestran el comportamiento de la tasa de cruce frente al promedio de aciertos de cada una de las muestras procesadas; figuras 5.14, 5.15 y 5.16, para las muestras de muy pocos (20), muchos (80) y bastantes documentos (150) respectivamente.

Documentos	Tasa de Cruce	Aciertos medios
20	0,70	16,25
20	0,75	16,50
20	0,80	**17,00**
20	0,85	16,75
20	0,90	16,62
20	0,95	16,75
80	0,70	48,62
80	0,75	50,75
80	0,80	**51,00**
80	0,85	50,75
80	0,90	50,00
80	0,95	50,00
150	0,70	74,38
150	0,75	78,25
150	0,80	**78,25**
150	0,85	77,50
150	0,90	74,62
150	0,95	77,50

Tabla 5.11: Resultados promedios del número de aciertos del AG con todas las muestras procesadas de la colección, variando la tasa de cruce.

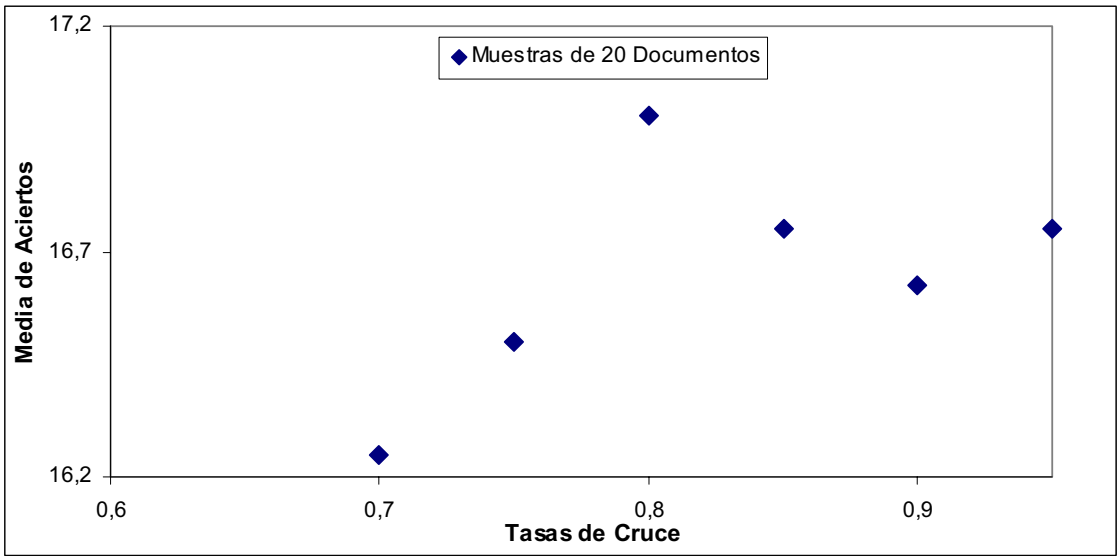

Figura 5.14: Promedio de aciertos al variar la tasa de cruce obtenidos con el AG para muestras de muy pocos documentos (20 documentos). La tasa de cruce próxima a 0,80 da los mejores resultados.

Contemplando los resultados obtenidos, podemos apreciar en la figura 5.14, que al procesar muestras de 20 documentos, resulta un mejor comportamiento (mayor tasa de aciertos en promedio) cuando se aplica una tasa de mutación de 0,80. Dicho comportamiento se manifiesta de manera gradual desde una tasa de 0,70 y llega a un punto máximo en 0,80, a partir del cual decae.

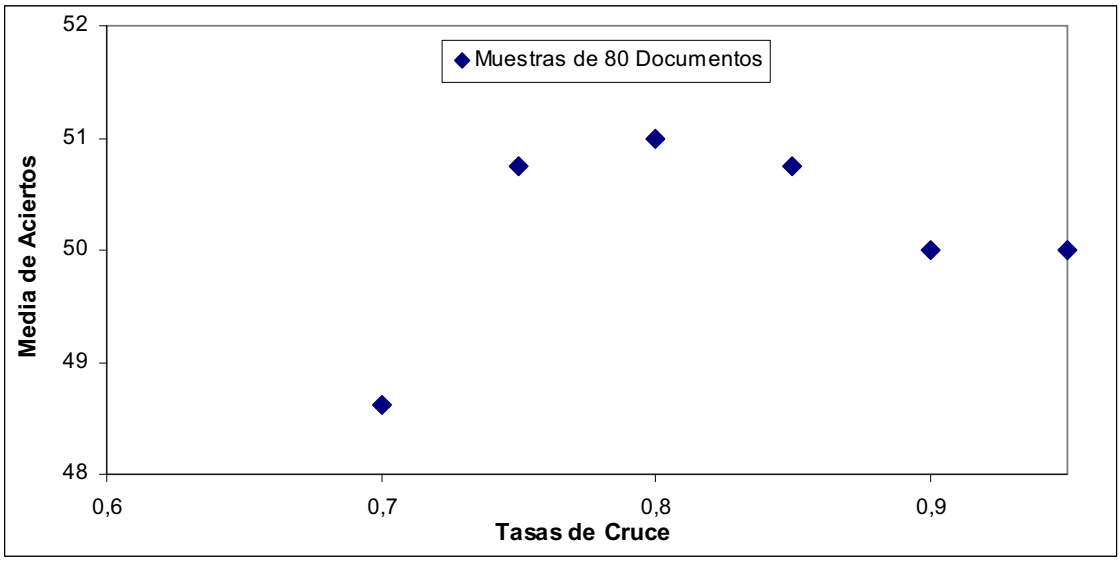

Figura 5.15: Promedio de aciertos al variar la tasa de cruce obtenidos con el AG, para muestras de muchos documentos (80 documentos). Tasa de cruce próxima a 0,80 da los mejores resultados.

Analizando, las figuras 5.15 y 5.16 se aprecia, para muestras de muchos (80) y bastantes (150) documentos que también aparece como el óptimo de la tasa de cruce el valor de 0.80.

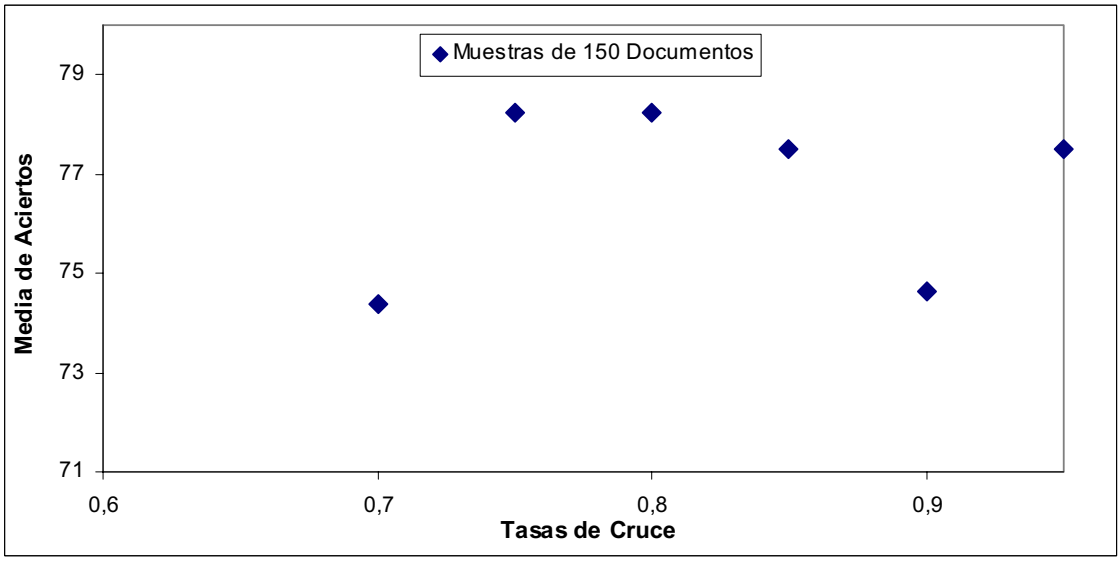

Figura 5.16: Promedio de aciertos al variar la tasa de cruce obtenidos con el AG, para muestras de bastantes documentos (150 documentos). Tasa de cruce próxima a 0,80 da los mejores resultados.

Para concluir, analizamos comparativamente en la gráfica 5.17, el factor de acierto del AG. Se aprecia claramente, un mejor comportamiento del AG cuando se utiliza una

tasa de 0,80 para el operador de cruce, independientemente de la muestra utilizada. Por lo tanto, dicho valor aparece como idóneo si deseamos maximizar la eficiencia del algoritmo, razón por la cual concluimos que es la tasa que nos ofrece mejores resultados.

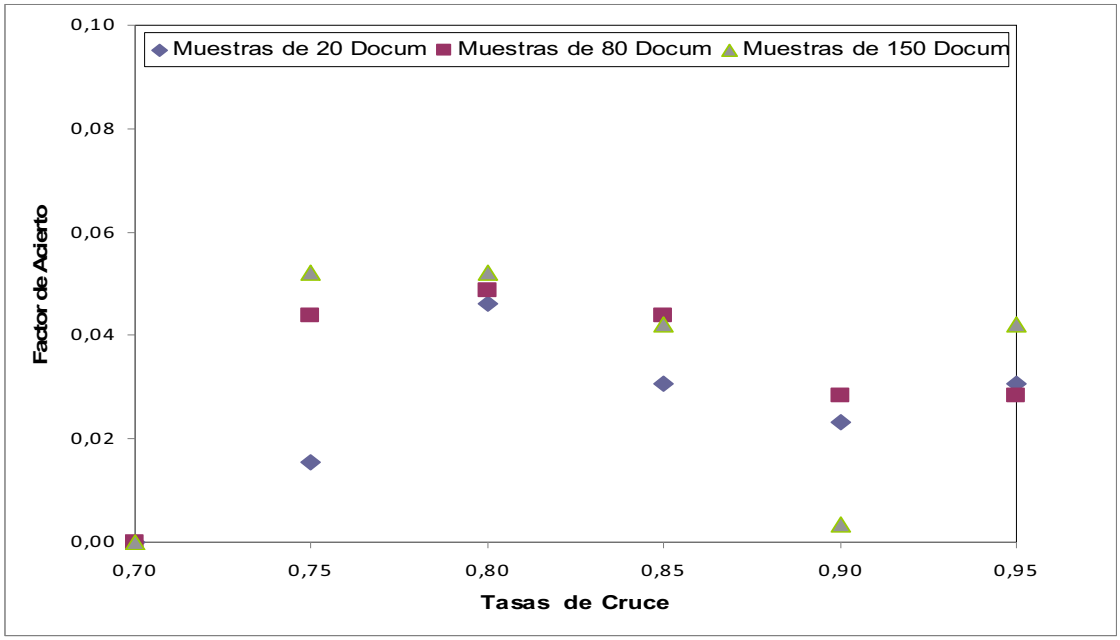

Figura 5.17: Factor de aciertos frente a tasas de cruce, con diferentes muestras de documentos procesadas .

Con el fin de confirmar las hipótesis anteriores, analizamos y comparamos el *fitness* medio obtenido en las diferentes pruebas resultantes de variar la tasa de cruce. Podemos apreciar en la figura 5.18, que el *fitness* mejora en promedio al aplicar una tasa de cruce equivalente a 0,80, indistintamente de la muestra de documentos que se utilice. Es decir, el *fitness* es menor si aplicamos dicha tasa de mutación. Este hecho coincide con que tengamos mejores resultados en la efectividad de nuestro AG, tal como se aprecia en la figura siguiente.

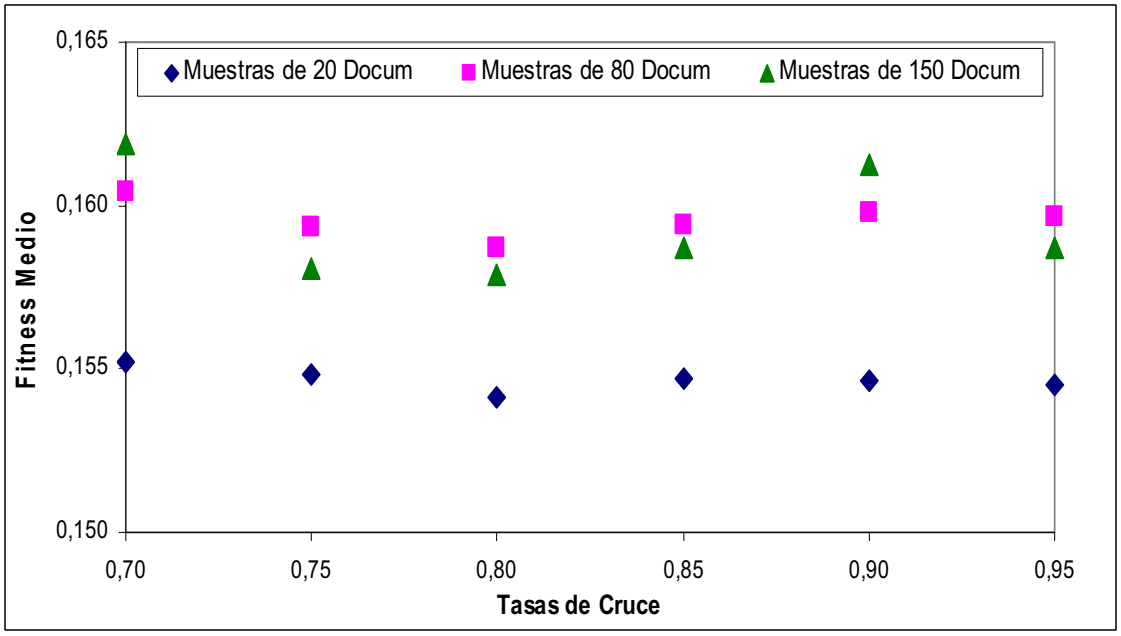

Figura 5.18: Fitness medio frente a las tasas de cruce, con diferentes muestras de documentos procesadas.

De todas las pruebas realizadas, podemos concluir que la tasa de cruce que nos proporciona mejores resultados es la tasa de 0,80, razón por la cual es la que utilizaremos para realizar las pruebas reales del AG con las diferentes colecciones documentales.

Para concluir, la tabla 5.12 recoge el conjunto de parámetros que definen el comportamiento del sistema experimental para nuestro AG.

Parámetros	**Valores**
Tamaño de la población (Número de árboles)	50
Número de evaluaciones (Generaciones)	4000 máximo
Tamaño del torneo	2
Tasa de Mutación (TM)	0,03
Tasa de Cruce (TC)	0,80
Cantidad de documentos	Muy Pocos (20), Pocos (50), Muchos (80), Bastantes documentos (150), Mas Documentos (250)
Número de Términos Lematizados	Dependiendo de la colección
Coeficientes α	0,85
Umbral de Profundidad	7 /10

Tabla 5.12: Parámetros considerados para el Sistema Evolutivo

Por último, remarquemos que utilizando todos los parámetros que hemos estudiado del AG la evolución es semejante a la que se tiene en los algoritmos del tipo no supervisado, donde el propio algoritmo va evolucionando y descubriendo los grupos de documentos afines sin ninguna otra información adicional que permita indicar a que categoría pertenece, ni se indica el número de grupos que deseamos conseguir, siendo el propio algoritmo él que lo va obteniendo a partir de la evolución.

5.5. Método de evaluación de los resultados

Para constratar los resultados, compararemos la agrupación de documentos realizada por nuestro sistema con los grupos realmente existentes, obtenidos de cada colección documental por el procedimiento introducido en el capítulo 3. De esta forma, nos centramos exclusivamente en calcular los aciertos del algoritmo, en comparación con los datos reales, conocidos de antemano para el proceso documental.

Con el objeto de comparar mejor los resultados los parámetros del sistema tomarán los valores presentados en la tabla 5.12 en todas las ejecuciones, y el sistema dispondrá de al menos 4000 generaciones para completar su evolución.

Además, todos los resultados son comparados con los obtenidos por otro algoritmo de agrupamiento del tipo supervisado en condiciones óptimas (*Kmeans*).

Para el análisis de los resultados, estudiaremos los siguientes aspectos:

a) **Efectividad del agrupamiento:** Es el indicador más importante en la comparación de los resultados al tratar la calidad del agrupamiento. Se analizarán los valores obtenidos para los “mejores aciertos” (los que devolvieron los mejores valores en la función de adaptación) y los valores promedio obtenidos en las cinco ejecuciones del algoritmo evolutivo

b) **Evolución del *Fitness*.** Analizamos la forma como evoluciona el *fitness* en cada una de las ejecuciones con cada una de las muestras de documentos procesadas, determinando su comportamiento y el número de aciertos que proporciona el algoritmo.

c) **Convergencia del algoritmo**: Analizaremos en qué ejecución el algoritmo evolutivo obtiene el mejor agrupamiento, es decir alcanza el mayor número de aciertos y se estabiliza en él. De este modo podremos determinar si el algoritmo se ha estancado en la evolución o si presenta una convergencia rápida. Para garantizar la robustez de las ejecuciones analizamos la convergencia media en todas las ejecuciones, si bien naturalmente serán más adecuados aquellos resultados que requieran menor tiempo de convergencia, siempre será deseable un agrupamiento de mejor calidad a pesar de que el AG necesite evolucionar más.

Todos los resultados obtenidos con el AG se obtuvieron aplicando los parámetros que figuran en la tabla 5.12. En los experimentos presentados se comparó en todo momento el comportamiento de nuestro algoritmo evolutivo con el del algoritmo base, el algoritmo "*Kmeans*", realizándose un análisis global de todos los resultados obtenidos con el objeto de determinar la robustez del sistema propuesto. Por ello todos los experimentos que evaluaron la calidad del algoritmo evolutivo se realizaron sabiéndose de antemano los resultados reales, variando en cada prueba el número de documentos a agrupar y tomándose grupos de "dos" categorías conocidas de la colección documental. De esta forma los resultados obtenidos se compararon siempre contra los datos reales. Además, en cada ejecución todos los documentos fueron procesados por el sistema evolutivo con semillas diferentes (un número de cinco) que dependían del tiempo de ejecución del sistema.

En lo que respecta al algoritmo "*Kmeans*", éste se ejecutó con las mismas muestras, pasándole como dato el número de grupos que se desea obtener, a fin de que el algoritmo agrupase los documentos lo mejor posible, para luego, también comparar mejor la eficacia del algoritmo propuesto contra sus resultados en condiciones idóneas.

5.6. Experimentos y análisis de los resultados del AG.

En esta sección, hacemos el análisis global del algoritmo evolutivo introducido en esta memoria para diferentes colecciones de datos, y utilizando todos los parámetros descritos en la sección anterior. En este análisis utilizaremos hasta cuatro distribuciones diferentes de la colección Reuters, que ofrezcan la mayor dispersión posible y además usaremos la colección de editoriales El Mundo de los años 2006 y 2007 preparada de antemano en la etapa de procesamiento documental tal como se comentó en la sección 5.2. Para ello, vamos a analizar los resultados con diferentes muestras de documentos, reagrupando los mejores resultados, los resultados promedio de cada muestra de documentos. Por ello, utilizaremos las muestras de documentos definidas anteriormente como "*muy pocas", "pocas", "muchas" y "bastantes*" documentos en cada una de las colecciones consideradas, y empezaremos nuestro análisis con la colección Reuters 21578.

Partiendo de estas muestras, hacemos un análisis del algoritmo, y finalmente, analizaremos su comportamiento global con ambas colecciones documentales, con el fin de conocer la mejor configuración del AG.

5.6.1. Mejores resultados con las muestras de documentos de la colección Reuters (colección en inglés)

En esta primera sección, analizaremos el comportamiento con las cuatro distribuciones de la colección Reuters que se han procesado con el AG. De este modo, las tablas siguientes (tablas 5.13, 5.14, 5.15 y 5.16) recogen el mejor resultado y el resultado promedio obtenido por el algoritmo con cada una de las muestras analizadas. Las tablas constan de tres columnas principales: algoritmo con muestras de documentos procesadas, mejor resultado y mejor promedio. A su vez las dos últimas se subdividen ambas en tres columnas respectivamente, mostrando el valor de *fitness*, la efectividad del algoritmo, y la convergencia del algoritmo para el mejor resultado individual; y las medias y desviaciones típicas de la función de adaptación, así como la convergencia media del algoritmo y el número de aciertos obtenido por el algoritmo de *Kmeans*.

Las tablas anteriores corresponden a los resultados resumidos de todos los experimentos [Castillo José Luis, Fernández del Castillo José R, León González, 2009b], el detalle de cada una de las pruebas realizadas con el AG en cada una de las distribuciones procesadas con la colección documental Reuters, lo podemos ver en las tablas (5.18 a 5.40) que presentamos en los anexos

Distribución 2 ***BD 1*** ***(Reuters)***	***Documentos de las Categorías: Acq y Earn*** ***Mejor Resultado***			***Mejor Promedio***			
Muestras documentos	***Fitness***	***Efectividad***	***Converg.***	***Fitness Medio***	***Desviación Fitness***	***Converg Media***	***Kmeans***
Muy pocos documentos (20 documentos)	0,155447570	*85% (18 aciertos)*	886	0,15545476	0,00000828	*1086*	*16,6*
Pocos Documentos (50 Documentos)	0,156223280	*94% (47aciertos)*	*3051*	0,15624280	0,00002329	*2641*	*45,8*
Muchos Documentos (80 Documentos)	0,159009400	*89% (71 aciertos)*	*2500*	0,15921181	0,00020587	*2246*	*67,8*
Bastantes Documentos (150 Documentos)	0,165013920	*77% (115 aciertos)*	*2342*	0,16508519	0,00007452	*2480*	*121,6*
Mas Documentos (246 Documentos)	0,174112100	*69% (170 aciertos)*	*2203*	0,17430502	0,00033602	*2059*	*202,8*

Tabla 5.13:Resultados comparativos del Sistema Evolutivo con diversas muestras de documentos, mostrando los mejores resultados obtenidos y los resultados promedios de las evaluaciones con la distribución 2 de la colección Reuters 21578.

Distribución 8 ***BD 2*** ***(Reuters)***	***Documentos de las Categorías: Acq y Earn*** ***Mejor Resultado***			***Mejor Promedio***			
Muestras documentos	***Fitness***	***Efectividad***	***Converg.***	***Fitness Medio***	***Desviación Fitness***	***Converg Media***	***Kmeans***
Muy pocos documentos (20 documentos)	0,151163560	*85% (17 aciertos)*	555	0,15116356	0,00000000	*679*	*15,8*
Pocos Documentos (50 Documentos)	0,154856500	*96% (48aciertos)*	*1615*	0,15485650	0,00000000	*1334*	*43,8*
Muchos Documentos (80 Documentos)	0,157073880	*85% (68 aciertos)*	*746*	0,15708362	0,00000898	*1360*	*66,2*
Bastantes Documentos (150 Documentos)	0,162035070	*69,3% (104 aciertos)*	*1989*	0,16242664	0,00033091	*2283*	*117,6*
Mas Documentos (188 Documentos)	0,163014600	*68,63% (129 aciertos)*	*2293*	0,16334198	0,00027325	*1773*	*140,6*

Tabla 5.14:Resultados comparativos del Sistema Evolutivo con diversas muestras de documentos, mostrando los mejores resultados obtenidos y los resultados promedios de las evaluaciones con la distribución 8 de la colección Reuters 21578.

Distribución 20 ***BD 3*** ***(Reuters)***	***Documentos de las Categorías: Acq y Earn*** ***Mejor Resultado***			***Mejor Promedio***			
Muestras documentos	***Fitness***	***Efectividad***	***Converg.***	***Fitness Medio***	***Desviación Fitness***	***Converg Media***	***Kmeans***
Muy pocos documentos (20 documentos)	0,153027060	*85% (17 aciertos)*	1092	0,15321980	0,00018398	*1108*	*16,8*
Pocos Documentos (50 Documentos)	0,156198620	*92% (46aciertos)*	*2173*	0,15666137	0,00030077	*2635*	*44,8*
Muchos Documentos (80 Documentos)	0,158069980	*81,25% (65 aciertos)*	*2196*	0,15810383	0,00001884	*1739*	*66,8*
Bastantes Documentos (108 Documentos)	0,159031080	*69.4% (75 aciertos)*	*1437*	0,15927630	0,00026701	*2636*	*82,2*

Tabla 5.15:Resultados comparativos del Sistema Evolutivo con diversas muestras de documentos, mostrando los mejores resultados obtenidos y los resultados promedios de las evaluaciones con la distribución 20 de la colección Reuters 21578.

Distribución 21 ***BD 4*** ***(Reuters)***	***Documentos de las Categorías: Acq y Earn*** ***Mejor Resultado***			***Mejor Promedio***			
Muestras documentos	***Fitness***	***Efectividad***	***Converg.***	***Fitness Medio***	***Desviación Fitness***	***Converg Media***	***Kmeans***
Muy pocos documentos (20 documentos)	0,152048900	*90% (18 aciertos)*	1163	0,15206069	0,00001601	*1165*	*17,8*
Pocos Documentos (50 Documentos)	0,153006650	*92% (46aciertos)*	*2079*	0,15304887	0,00004569	*2736*	*45,6*
Muchos Documentos (80 Documentos)	0,156029510	*81% (65 aciertos)*	*2787*	0,15637693	0,00025014	*2810*	*66,4*
Bastantes Documentos (132 Documentos)	0,157012180	*70,4% (93 aciertos)*	*3359*	0,15720766	0,00024132	*1980*	*98,6*

Tabla 5.16:Resultados comparativos del Sistema Evolutivo con diversas muestras de documentos, mostrando los mejores resultados obtenidos y los resultados promedios de las evaluaciones con la distribución 21 de la colección Reuters 21578.

Los datos mostrados permiten comparar la efectividad del AG, en porcentaje, para las diferentes muestras de documentos, la cual se refleja gráficamente en las figuras 5.19 al 5.22.

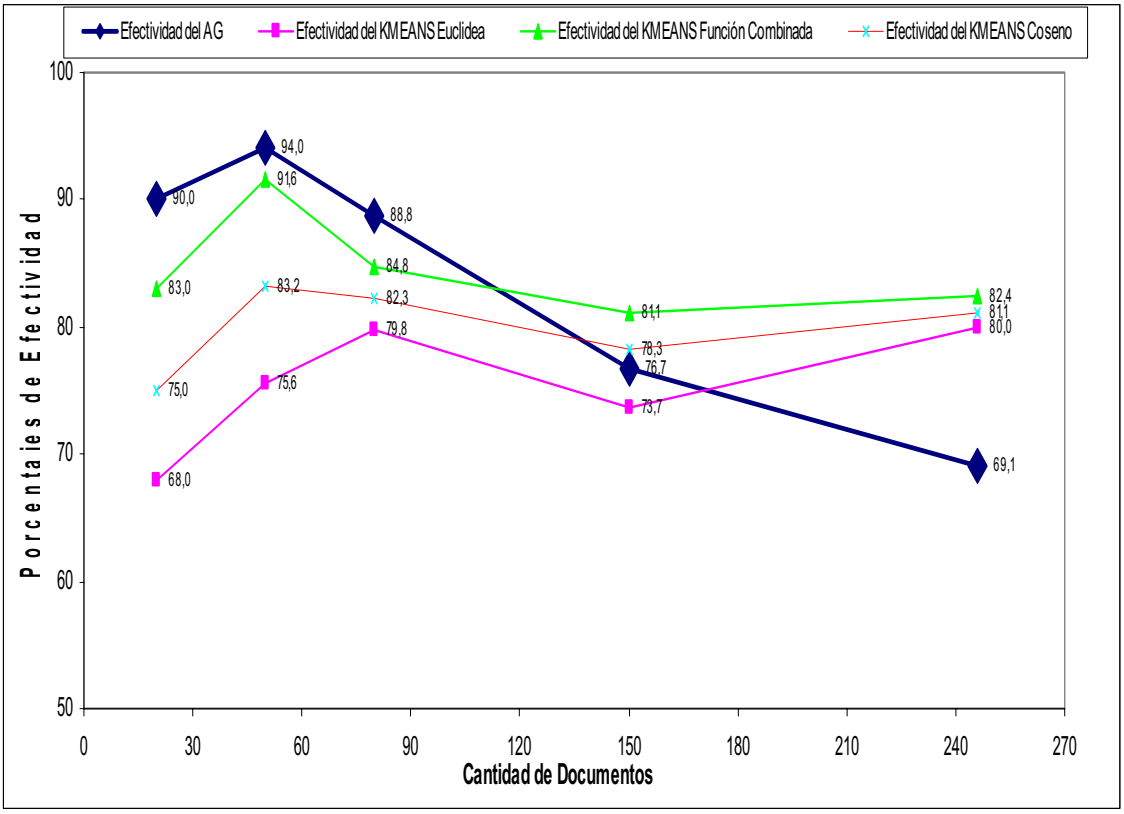

Figura 5.19: Efectividad del AG para diferentes muestras de documentos de la colección Reuters Distribución 2. Los resultados se comparan con los obtenidos por el algoritmo Kmeans (usando diferentes funciones). Se puede apreciar como para un número de documentos inferior a 120 el AG propuesto da mejores resultados que Kmeans.

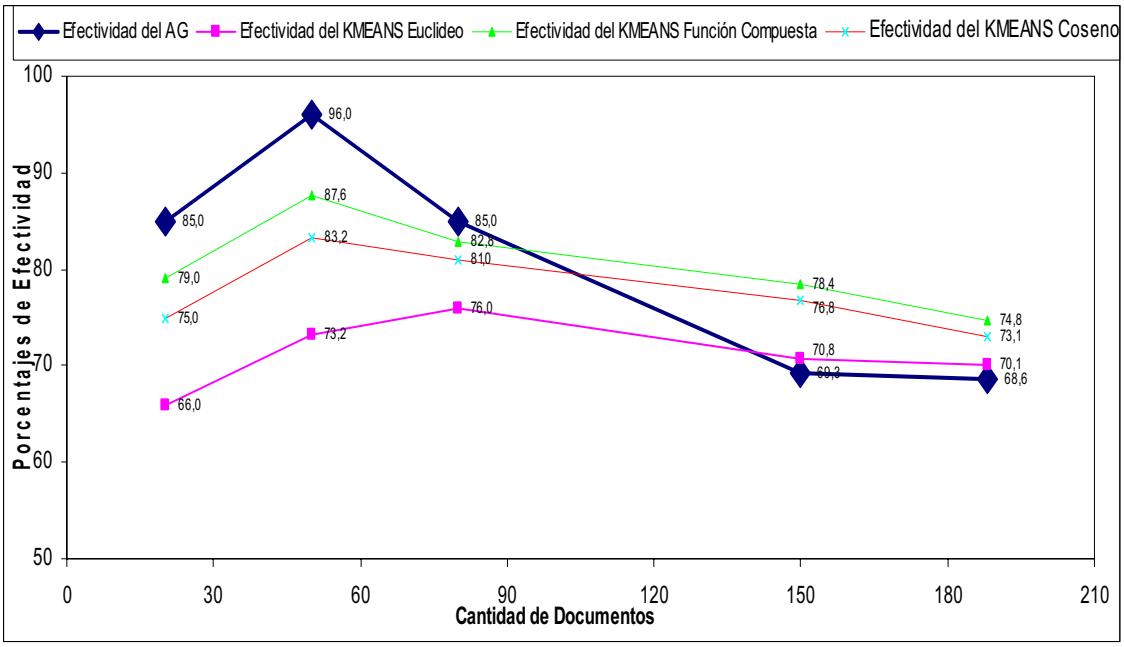

Figura 5.20: Efectividad del AG con diferentes muestras de documentos de la colección Reuters Distribución 8. Los resultados se comparan con los obtenidos por el algoritmo Kmeans (usando diferentes funciones). Se puede apreciar como para un número de documentos inferior a 100 el AG propuesto da mejores resultados que Kmeans.

Debido a que, la distribución 20 de la colección Reuters presentaba una menor cantidad de documentos, hemos realizado el procesamiento de dicha distribución usando las diferentes muestras de documentos de que constataba dicha colección, obteniendo en la mayoría de los casos mejores resultados que el algoritmo *Kmeans* , tal como lo apreciamos en la figura 5.21.

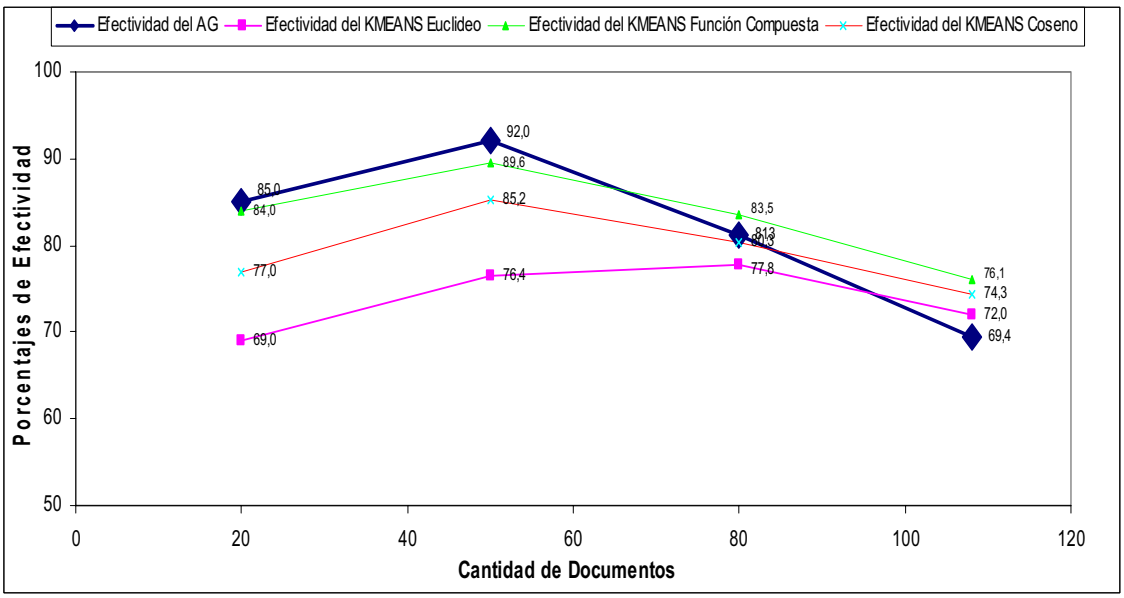

Figura 5.21: Efectividad del AG a nivel porcentual con diferentes muestras de documentos de la colección Reuters Distribución 20. Los resultados se comparan con los obtenidos por el algoritmo Kmeans (usando diferentes funciones). Se puede apreciar como para la mayoría de las muestras de documentos existentes en la colección(108 solamente),,el AG propuesto da mejores resultados que Kmeans.

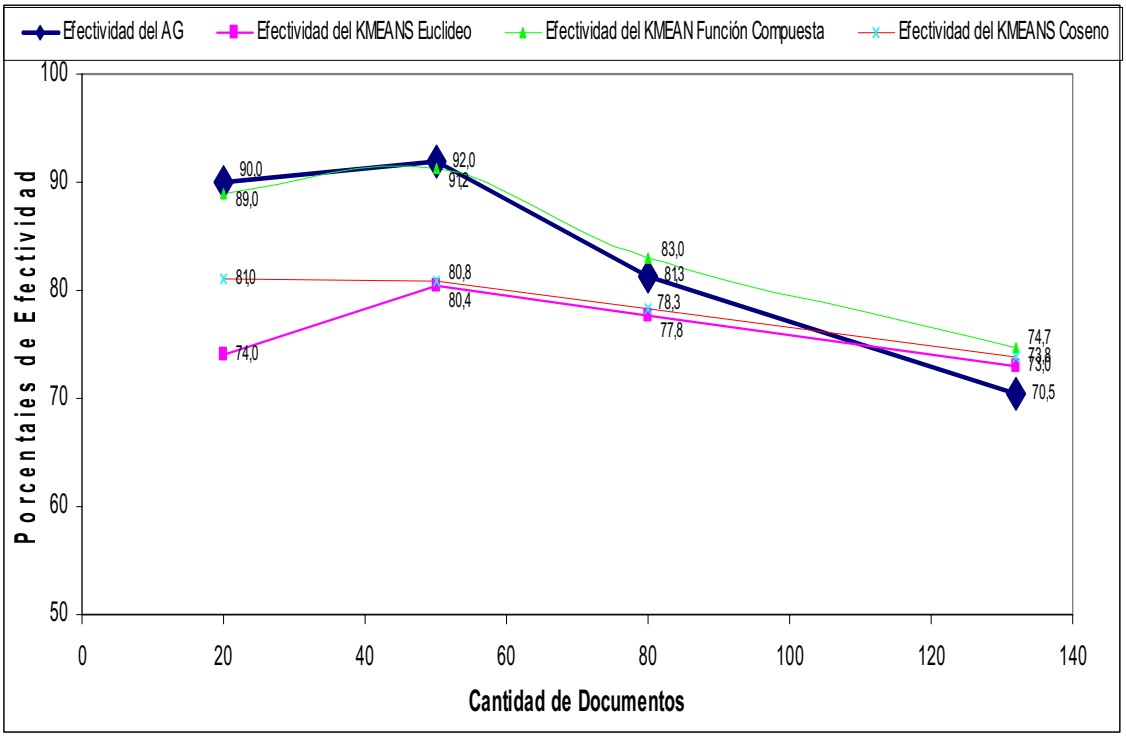

Figura 5.22:Efectividad del AG para diferentes muestras de documentos de la colección Reuters Distribución 21. Los resultados se comparan con los obtenidos por el algoritmo Kmeans (usando diferentes funciones). Se observa que la función compuesta da mejores, tanto con el algoritmo Kmeans como con el AG propuesto.

En las figuras anteriores, podemos apreciar la efectividad del AG comparada con la que nos proporcionan el algoritmo supervisado "*Kmeans*" (usando diferentes funciones de evaluación). Se observa que para las distribuciones Reuters que se procesaron la efectividad del algoritmo evolutivo es mucho mejor en la mayoría de los casos que el *Kmeans*.

Nótese que el comportamiento de las gráficas anteriores manifiesta una tendencia muy parecida para la mayoría de las distribuciones aunque en la distribución Reuters 21, con una mayor dispersión, los resultados no superan al *Kmeans tradicional cuando se procesan mas de 120 documentos.*

A la vista de estos resultados experimentales apreciamos que en general el comportamiento del algoritmo con nuestra función propuesta combinada es, en la mayoría de los casos, mejor que el algoritmo supervisado, lo que avala a nuestro algoritmo como una alternativa válida para agrupar muestras de documentos del orden de la centena, con la ventaja de realizarse de forma no supervisada. [Castillo José Luis, Fernández del Castillo José R, León González, 2009b]

Por otro lado, también hemos realizado un estudio de la convergencia del AG en cada una de las pruebas realizadas con las distribuciones de la colección Reuters, que se puede ver en las figuras 5.23 al 5.26, que muestran la convergencia de nuestro AG con cada una de las distribuciones Reuters procesadas.

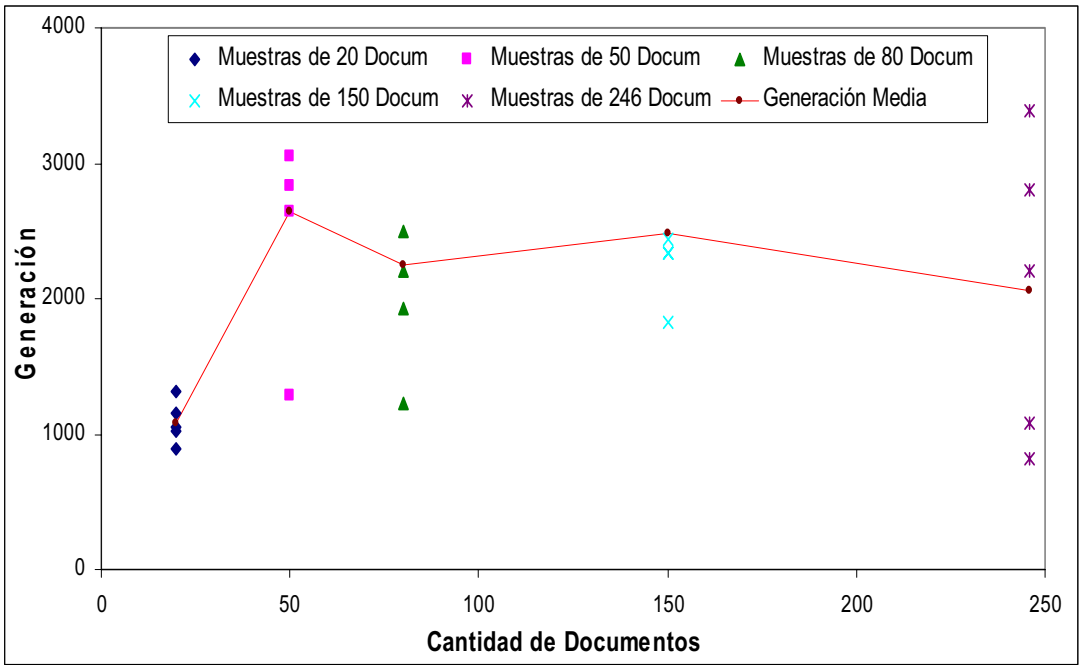

Figura 5.23:Convergencia del AG para diferentes muestras de documentos (en cada una de las pruebas realizadas) con la colección documental Reuters-Distribución 2, y su comparativa con su respectiva convergencia media del AG.

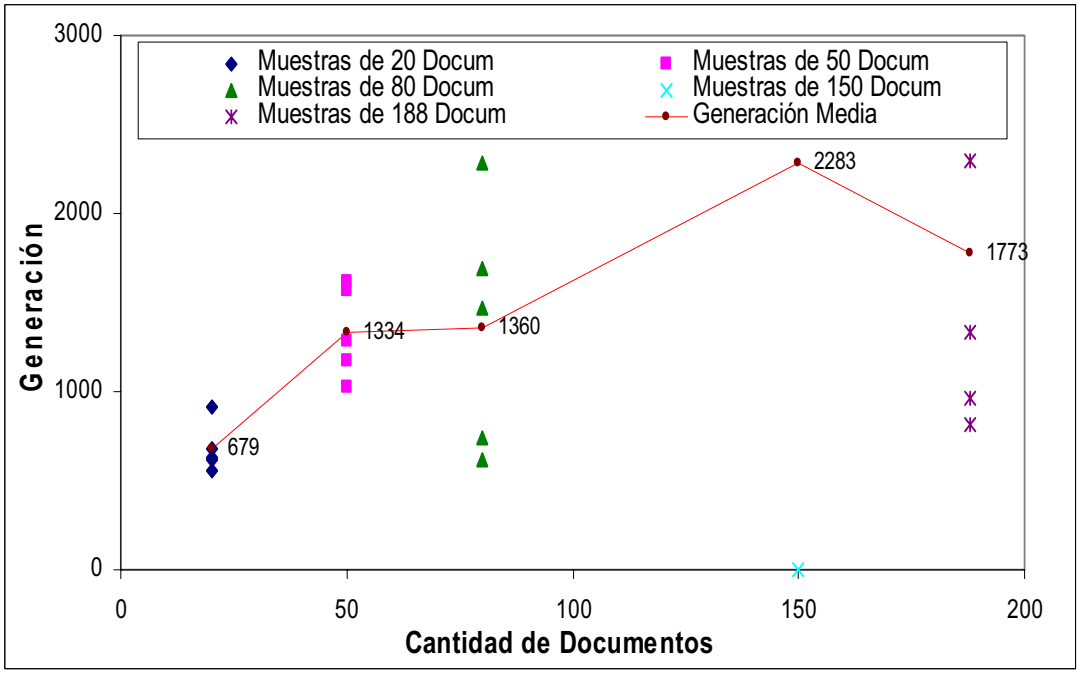

Figura 5.24: Convergencia del AG para diferentes muestras de documentos (en cada una de las pruebas realizadas) con la colección documental Reuters-Distribución 8 y su comparativa con su respectiva convergencia media del AG.

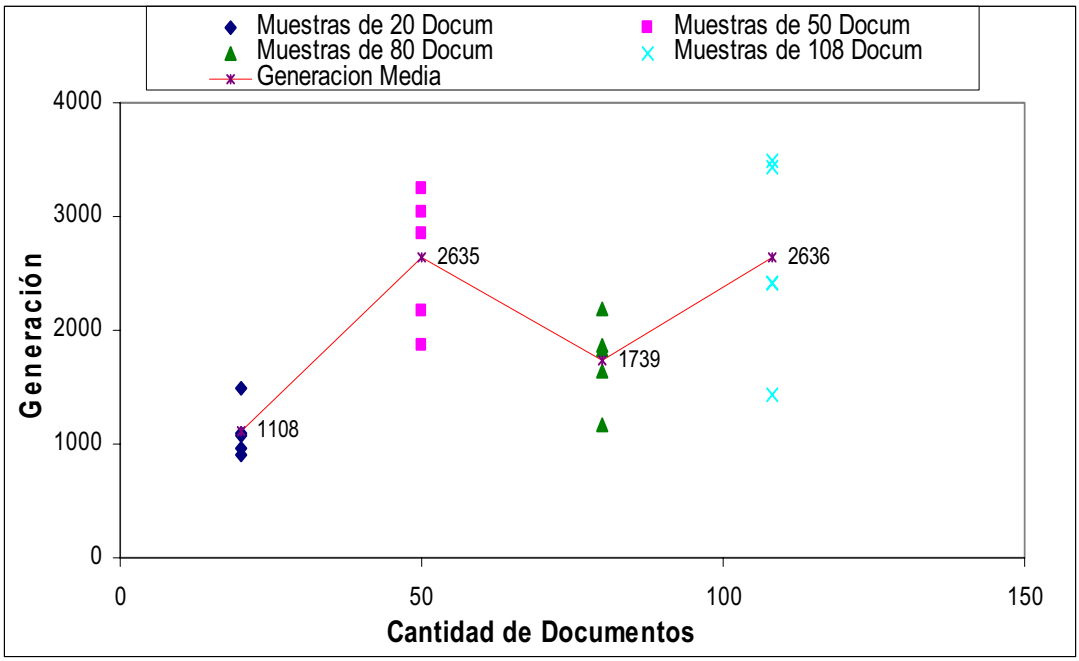

Figura 5.25:Convergencia del AG para diferentes muestras de documentos (en cada una de las pruebas realizadas) con la colección documental Reuters-Distribución 20, y su comparativa con su respectiva convergencia media del AG.

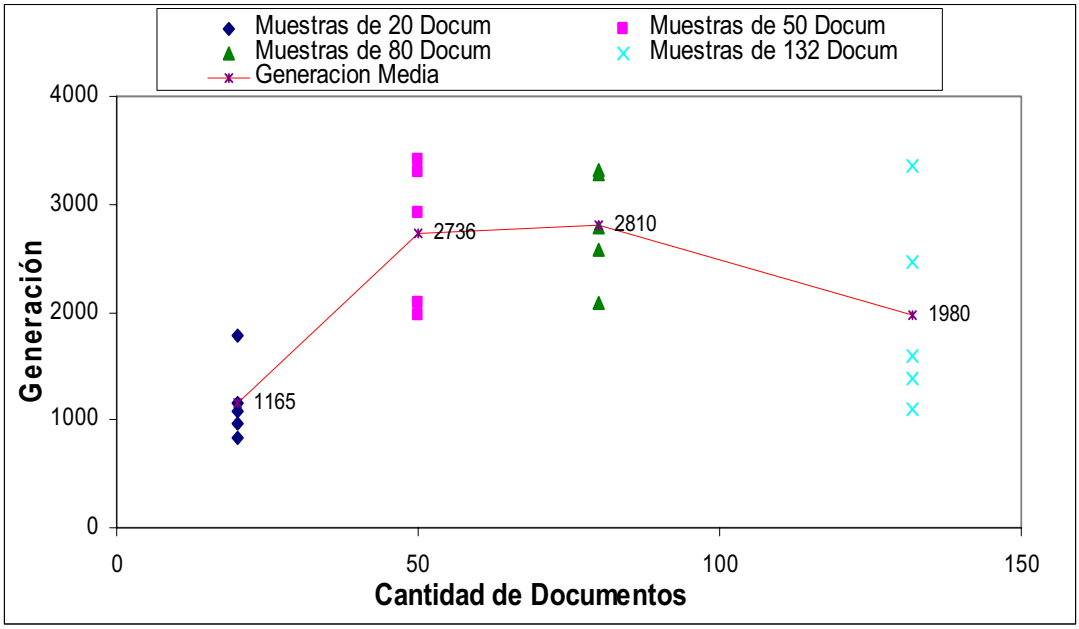

Figura 5.26:Convergencia del AG para diferentes muestras de documentos (en cada una de las pruebas realizadas) con la colección documental Reuters-Distribución 21, y su comparativa con su respectiva convergencia media del AG.

Finalmente, mostramos en la figura 5.27 el comportamiento de la convergencia media del algoritmo con todas las distribuciones Reuters procesadas, manifestándose un comportamiento medio estable y homogéneo para las diferentes muestras de documentos que se procesaron experimentalmente.

Podemos concluir, a la vista de los resultados obtenidos que, el AG, genera un número de aciertos alto, igual y muchas veces superior que el algoritmo del tipo supervisado *Kmeans*, lo que implica una mejor efectividad al procesar documentos. Verificamos experimentalmente que se logra los mejores resultados (mayor número de aciertos, y por lo tanto mayor efectividad) a medida que minimizamos el *fitness*, y esto ocurre experimentalmente independientemente de las muestras de documentos procesadas en todas las colecciones Reuters procesadas con los parámetros descritos en la tabla 5.12.

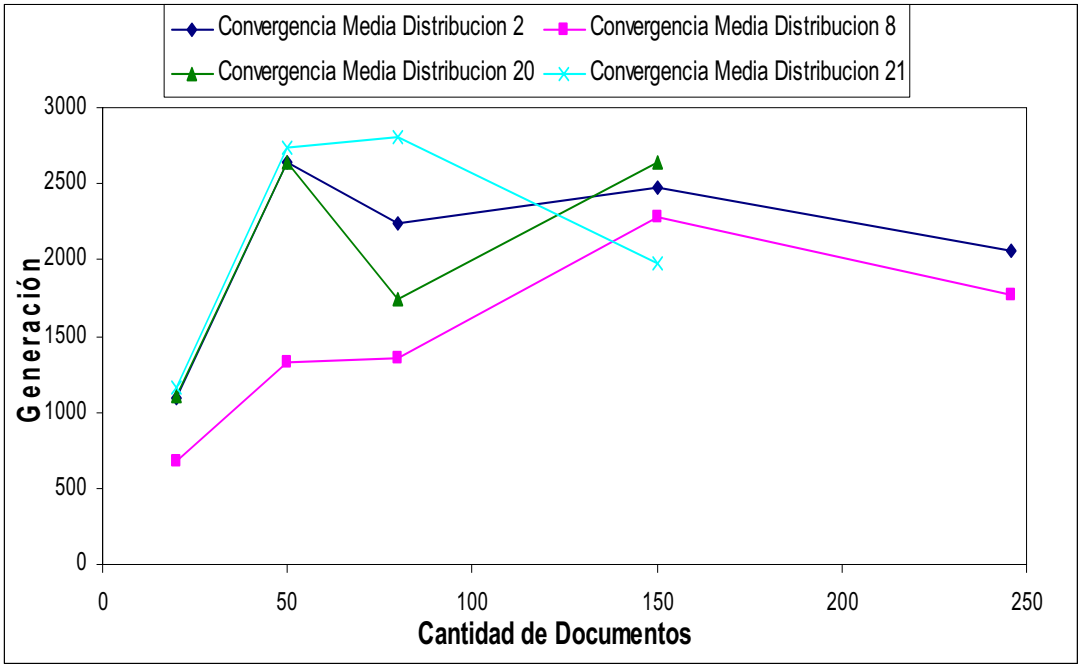

Figura 5.27: Convergencia media del AG en todas las distribuciones Reuters procesadas frente a todas las muestras de documentos.

Asimismo, debido a que usamos un algoritmo evolutivo, que se ejecuta con diferentes semillas y varias veces, también hemos analizado el comportamiento del *fitness* medio en todas las muestras con la finalidad de poder determinar la robustez del algoritmo y encontrar el mejor valor en promedio con cada muestra, lo que nos permitirá poder apreciar la tendencia que adopta el *fitness* medio en cada una de las distribuciones procesadas con esta colección documental.

De esta forma en la figura 5.28 mostramos la evolución del *fitness* medio en todas las distribuciones Reuters procesadas variando las muestras de documentos del AG. Podemos concluir que los mejores *fitness* en promedio presentan un comportamiento homogéneo independientemente de la distribución que se procese, lo que implica que el algoritmo evolutivo se comporta de una manera robusta con diferentes colecciones de datos, y que existe una tendencia exponencial en el *fitness* del algoritmo a medida que se procesa una mayor cantidad de documentos. Dicha tendencia se podría generalizar y daría motivo a proponer un nuevo *fitness* que tenga una relación directa con la cantidad de documentos que se desee procesar.

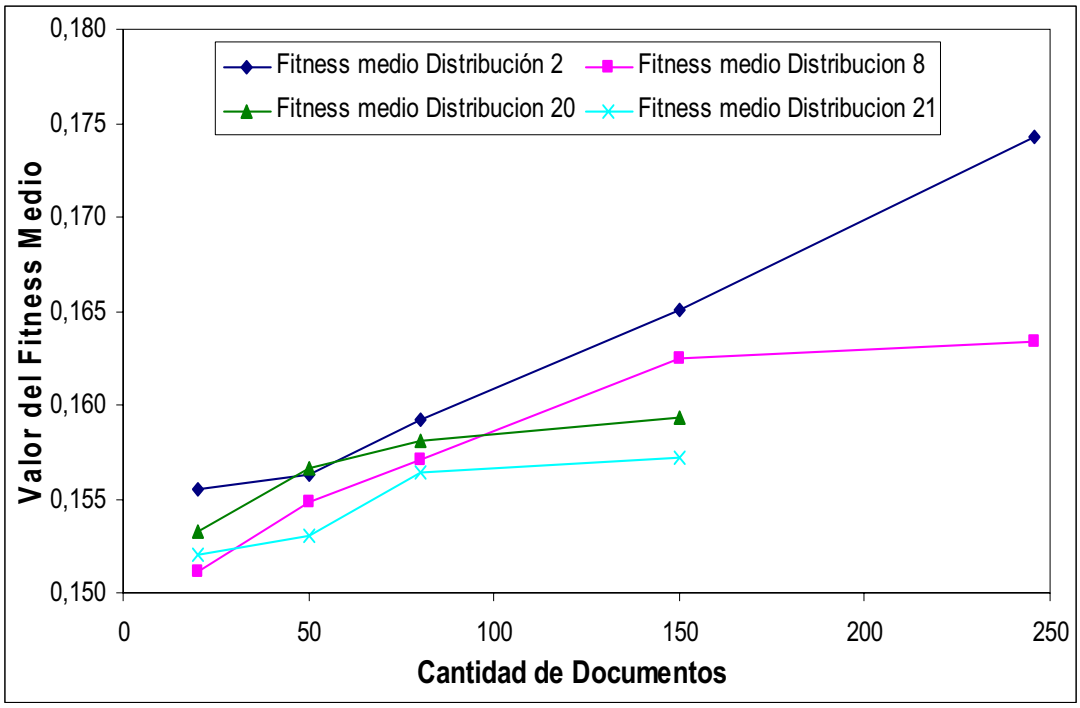

Figura 5.28: Fitness medio del AG en todas las distribuciones Reuters procesadas frente a las diferentes muestras de documentos.

Después de agrupar y mostrar todas las muestras de documentos procesadas por el AE con sus correspondientes mejores resultados individuales y promedios, estamos en disposición de efectuar una comparativa a nivel global.

A vista de los experimentos realizados, se puede apreciar que obtenemos muy buenos resultados al agrupar los documentos con el algoritmo, y que dentro de cada grupo, se van colocando cada vez más cerca los documentos por su similitud y su cercanía, debido a la naturaleza del *fitness* de nuestro AG.

Se observa además que la efectividad del algoritmo es mejor que la del algoritmo supervisado *Kmeans* en muchos de los casos, independientemente de las muestras tomadas, lo que demuestra que el algoritmo es eficiente y robusto en los resultados que proporciona. Aunque el porcentaje de eficiencia del algoritmo cae al aumentar la cantidad de documentos, eso no invalida en absoluto los resultados obtenidos, sino que reafirma más la robustez del algoritmo debido a que a mayor cantidad de documentos, hay una mayor probabilidad de que exista una mayor heterogeneidad entre los documentos, lo que perjudica indudablemente los resultados finales y hace más difícil aplicar cualquier método de agrupación. Por lo tanto podemos concluir que el algoritmo a través de la evolución constituye una alternativa válida y eficiente para el agrupamiento no supervisado de colecciones documentales, en este caso para colecciones en inglés.

Procederemos ahora a validar el algoritmo evolutivo cambiando el tipo de colección documental. Para ello mostraremos los resultados obtenidos usando los documentos en español procesados a partir de los editoriales del diario El Mundo.

5.6.2. Mejores resultados con las muestras de documentos de la colección de editoriales (colección en español)

En la tabla 5.17, realizamos el mismo tratamiento con la colección de documentos en español, preparada con las editoriales del diario El Mundo de los años 2006 y 2007, tomando muestras con distintas cantidades de documentos, en la tabla se muestra los resultados resumidos de todas las pruebas realizadas, y en los anexos, mostramos el detalle de cada una de las pruebas realizadas con esta nueva colección documental en español.

Documentos en Español ***BD (Editoriales)***	***Documentos de las Categorías: Vida Política y Derecho*** ***Mejor Resultado***			***Mejor Promedio***			
Muestras documentos	***Fitness***	***Efectividad***	***Converg.***	***Fitness Medio***	***Desviación Fitness***	***Converg Media***	***Kmeans***
Muy pocos documentos (20 documentos)	0,152444850	*85% (17 aciertos)*	1026	0,15265851	0,00012727	*2196*	*16,6*
Pocos Documentos (50 Documentos)	0,154187850	*96% (48aciertos)*	*2839*	0,15437233	0,00009278	*2678*	*44,2*
Muchos Documentos (80 Documentos)	0,155099730	*83,7% (67 aciertos)*	*3137*	0,15537314	0,00026143	*3085*	*66,8*
Bastantes Documentos (150 Documentos)	0,162184980	*78,6% (118 aciertos)*	*1244*	0,16249632	0,00027400	*1547*	*122,8*
Mas Documentos (238 Documentos)	0,172107860	*70,1% (167 aciertos)*	*3387*	0,17257683	0,00028775	*2021*	*179,6*

Tabla 5.17: Resultados comparativos del Sistema Evolutivo con diversas muestras de documentos, mostrando los mejores resultados obtenidos y los resultados promedios de las evaluaciones con la distribución de editoriales en español de los años 2006 y 2007 del diario El Mundo.

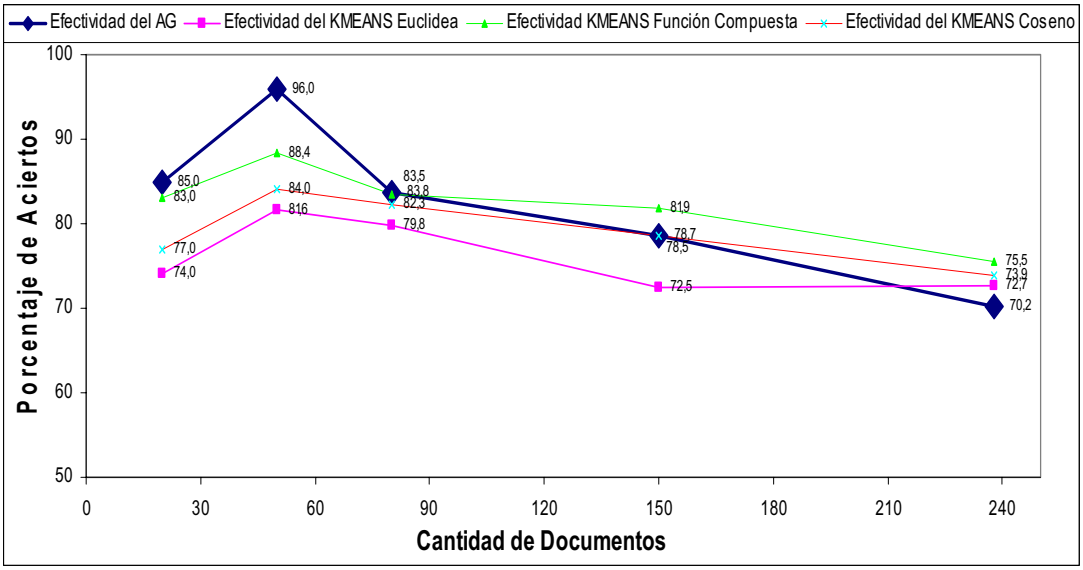

Figura 5.29: Efectividad del AG para diferentes muestras de documentos de la colección de editoriales del diario El Mundo. Se puede apreciar como para un número de documentos inferior a 150 el algoritmo propuesto da mejores resultados que Kmeans (usando diferentes funciones), siendo el comportamiento del AG similar al de la colección documental Reuters.

De todas las pruebas realizadas con este nuevo tipo de colección documental, podemos afirmar que, de forma general el algoritmo también genera aceptables soluciones (máximo número de aciertos y efectividad en la mayoría de los casos).

Además, los resultados en promedio son muy estables con la mayoría de las muestras de documentos, obteniéndose un comportamiento similar al obtenido cuándo se procesó la colección Reuters con el AG. Dicho comportamiento es mostrado en la figura 5.29.

En la figura 5.30 mostramos comparativamente el comportamiento de la convergencia media del algoritmo genético en cada una de las muestras procesadas de esta colección documental, y cada uno de los valores de convergencia obtenidos en cada una de las pruebas realizadas (5) con cada muestra. Podemos apreciar que la convergencia del algoritmo presenta algunos saltos cualitativos entre las diferentes muestras, alcanzando el mejor comportamiento (converge más rápido) al procesar muestras de 150 documentos.

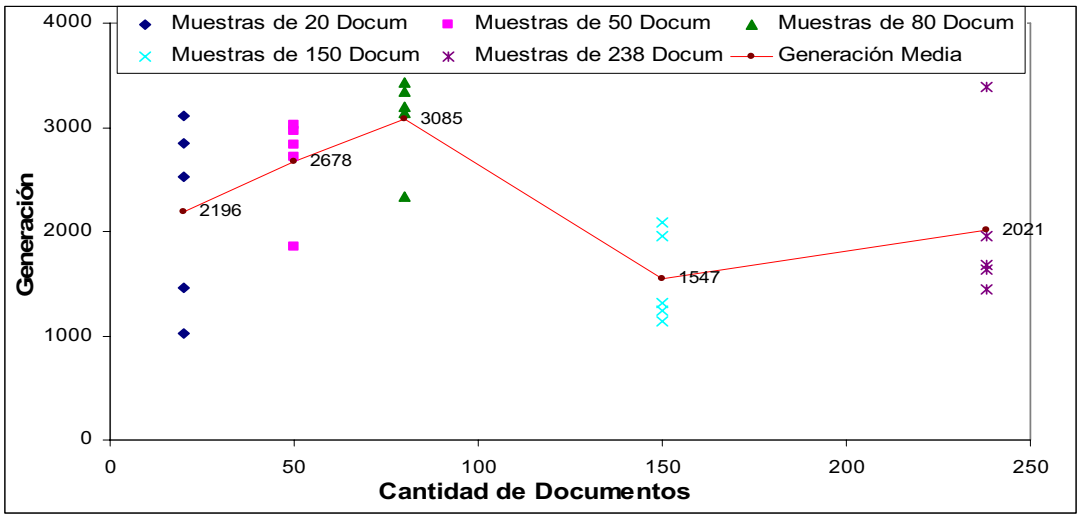

Figura 5.30:Convergencia del AG para diferentes muestras de documentos (en cada una de las pruebas realizadas)con la colección de editoriales del diario El Mundo y su comparativa con su respectiva convergencia media del AG.

De la misma forma que en los casos anteriores, y dado que realizamos muchas pruebas con cada una de las muestras de documentos de esta colección, con diferentes semillas en cada caso, y con el fin de poder comprobar la robustez del algoritmo, mostramos en la figura 5.31 el comportamiento de los *fitness* promedios obtenidos para las diferentes pruebas realizadas con cada una de las muestras de documentos de esta colección. Podemos comprobar que, el comportamiento de dichos *fitness* aumenta de una forma exponencial a medida que se procesa una mayor cantidad de documentos.

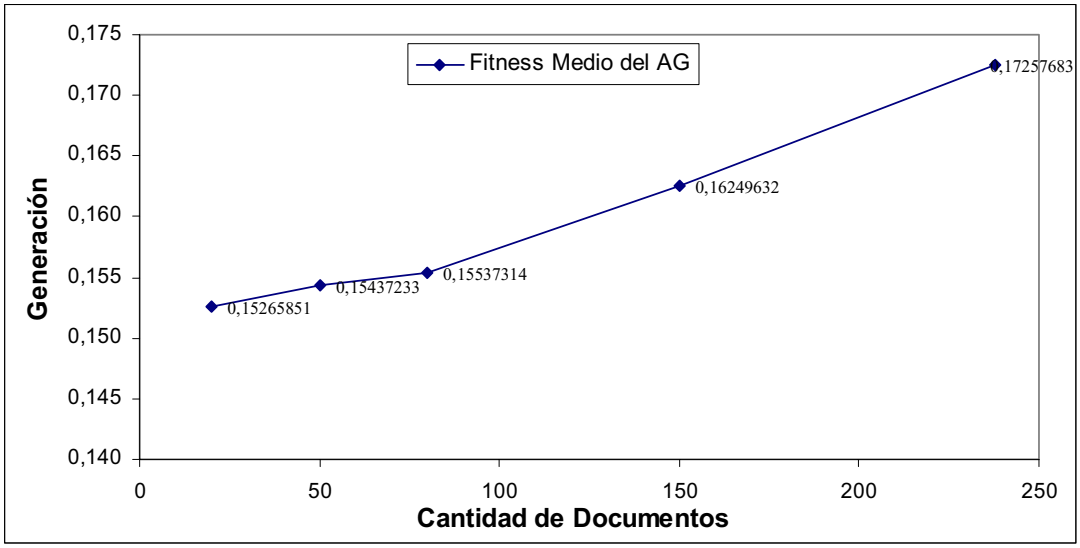

Figura 5.31: Comportamiento del Fitness medio a medida que se procesa más documentos con la colección de editoriales del diario El Mundo.

Podemos concluir, que los resultados de efectividad en esta colección difieren en muy poco con respecto a los obtenidos con la colección anterior. En general los resultados de efectividad del algoritmo son aceptables, y confirman la eficiencia y robustez del método. En cuanto a los mejores promedios se observa que de forma global hay una mayor dispersión en los valores obtenidos, a medida que se procesa una mayor cantidad de documentos, mucho mayor que cuándo se procesa los documentos de la colección Reuters, lo que indica que existen peculiaridades inherentes a esta colección debido probablemente a que tratamos documentos de índole periodístico que tienen un lenguaje muy particular y específico.

5.7. Conclusiones globales sobre el algoritmo evolutivo

En general nuestro algoritmo proporciona buenos resultados con ambas colecciones documentales. Podemos afirmar que al finalizar la evolución, y utilizando todos los parámetros estudiados y configurados para el algoritmo, ofrecemos una alta efectividad al realizar la agrupación de ambas colecciones documentales, siendo por lo tanto el algoritmo estable y robusto independientemente de la muestra ó colección documental utilizada.

Atendiendo a todos los resultados obtenidos, a fin de conocer la mejor opción a la hora de aplicar el algoritmo evolutivo con una cantidad de documentos determinada, podemos deducir que si lo que se desea es obtener el mejor resultado individual, es aconsejable utilizar la evolución del algoritmo con menores muestras de documentos (hasta 150), debido a que el *fitness* del algoritmo ofrece siempre una mayor efectividad de forma global al procesar dichas muestras.

En general el porcentaje de aciertos del AG siempre es muy elevado tanto en la colección documental Reuters, como en la colección en español de editoriales del 2006-2007 del diario El Mundo (habiendo mejor efectividad en la colección Reuters). La mayoría de los resultados usando el fitness combinado y el AG son tan buenos, o mejores, que los que se obtienen mediante el algoritmo supervisado *Kmeans*, como se ha demostrado con los diferentes experimentos sobre las muestras de documentos de ambos tipos de colecciones documentales. Esto permite concluir que el algoritmo evolutivo presentado constituye una alternativa válida para la agrupación de colecciones documentales de manera no supervisada.

Capítulo 6

Conclusiones y trabajos futuros

En este trabajo se ha propuesto una solución, haciendo uso de un sistema evolutivo, al problema de la agrupación de documentos. Se ha construido un sistema, que hace uso de un algoritmo genético que permite realizar la agrupación de documentos de forma no supervisada. El sistema evolutivo combina técnicas de Inteligencia Artificial, tales como computación evolutiva, para realizar la tarea de agrupación, y técnicas de Recuperación de Información, para realizar el procesamiento de los documentos.

Se ha introducido el uso de un método *NZIPF* con el fin de seleccionar los lexemas que representen las características de los vectores documentales que procesa el algoritmo genético. Para ello hemos realizado una metodología de procesamiento de documentos – comentada en el capítulo 3, sección 3.2.- basada en técnicas de *RI*.

Se ha analizado el comportamiento del algoritmo genético y hemos obtenido como resultado que este supera al conocido algoritmo supervisado *Kmeans,* tanto cuando se agrupan colecciones de documentos en inglés como en español.

Realizamos una adaptación de los *Algoritmos Genéticos*, usando operadores genéticos, que permitan realizar el agrupamiento de documentos, y hemos implementado una función *fitness,* que está basada en distancia y similitud entre documentos, para mejorar el proceso de selección de grupos de dichos documentos.

Otra aportación del sistema evolutivo se basa en la forma como se representan los individuos en el sistema, recogiendo conceptos evolutivos para su desarrollo.

De esta forma, el sistema propuesto constituye una alternativa a los métodos de *clustering* tradicionales, tales como los jerárquicos y/o particionales.

Validamos el sistema propuesto trabajando con colecciones documentales en inglés y en español. Para ello hemos usado las principales distribuciones de la colección documental Reuters 21578 – que constituye una piedra de toque usual dentro de la comunidad de investigadores dedicados a la agrupación de documentos -, y además hemos preparado y utilizado una colección con los editoriales de los años 2006 y 2007 del diario El Mundo para el caso de colecciones de documentos en español.

El trabajo realizado no sólo nos ha permitido obtener resultados que juzgamos muy interesantes, sino que también ha abierto nuevos caminos cuya futura exploración parece ser prometedora. A continuación, incluimos las conclusiones obtenidas, y una descripción de varias de estas posibles líneas de trabajo, cuya explotación queda propuesta para desarrollos futuros.

6.1. Conclusiones

A continuación, enumeraremos las conclusiones a las que nos ha llevado la investigación realizada en esta memoria de tesis.

- El método *NZIPF* funciona satisfactoriamente para la selección de los términos característicos de la colección documental. El número de aciertos que se obtiene, es muy aceptable, independientemente de la colección documental que se procese.

- El algoritmo genético usado es un algoritmo semi supervisado, que requiere el ajuste de algunos parámetros, sin necesitar ningún entrenamiento. La propia evolución determina el comportamiento del algoritmo, y proporciona los agrupamientos, comportándose de modo flexible al ajustarse las tasas de los operadores del algoritmo. En virtud de los resultados conseguidos, concluimos que los algoritmos semi supervisados pueden igualar e incluso superar a los algoritmos supervisados en estas tareas. Por lo tanto, se ha desarrollado un sistema no supervisado del tipo evolutivo que logra la precisión que proporcionan los algoritmos supervisados en la agrupación de colecciones documentales.

- Para la agrupación de documentos mediante un algoritmo genético se ha empleado la combinación de dos funciones de medida (distancia y similitud), para mejor caracterizar la relación existente entre los documentos, comprobándose su adecuado comportamiento.

- Hemos comprobado experimentalmente que un valor adecuado para el parámetro α, que interviene en la formula que combina linealmente las métricas de distancia e inversa de similitud de nuestro *fitness*, tiene que tomar valores próximos a 0,85. Superado dicho valor, a medida que nos alejamos, la contribución de ambas métricas va dejando de ser adecuada progresivamente.

- Se ha determinado valores para las tasas del operador de mutación *TM* y para las tasas del operador de cruce *TC* del algoritmo genético que maximizan la eficiencia del sistema. Para dicho algoritmo genético, utilizar una tasa de mutación equivalente a 0,03 conjuntamente con una tasa de cruce de 0,80, nos garantiza unos resultados que superan al algoritmo *Kmeans.* Podemos concluir que usar un enfoque evolutivo con unos parámetros adecuados, constituye una alternativa preferible a los métodos clásicos establecidos, tal como lo hemos comprobado experimentalmente con nuestro trabajo.

- El sistema ha sido estudiado en relación al número de documentos a procesar. Los experimentos muestran un buen funcionamiento para colecciones inferiores a 150 documentos, con número de aciertos superiores al de algoritmo *Kmeans*. De esta forma, hemos comprobado que la efectividad de nuestro algoritmo supera al *Kmeans* al procesar menos de 150 documentos, y se equipara a dicho algoritmo cuando procesamos mas de 150 documentos, siendo por lo tanto una alternativa a tener en cuenta para el agrupamiento de documentos, con la ventaja añadida que lo realizamos de forma no supervisada, dejando que sea la propia evolución la que realice el proceso. Además, se ha constatado que el *fitness* propuesto, adopta una tendencia exponencial a medida que aumentamos el número de documentos.

- El *fitness* utilizado por nuestro algoritmo genético, presenta muy poca dispersión en todas las pruebas realizadas con las diferentes muestras de documentos cuando utilizamos un valor de α menor a 0,85. A partir de dicho valor de α, existe una mayor contribución de la distancia euclídea, que origina una mayor dispersión en el *fitness*.

- En este trabajo se ha realizado una extensión de los algoritmos genéticos, incorporando nuevos operadores evolutivos a nuestro entender, para el cruce y para la mutación, lo que mejora el rendimiento del algoritmo en su tarea de agrupación. Considerando los resultados promedios obtenidos sobre todas las muestras, en cada una de las colecciones documentales utilizadas, podemos observar que se obtienen buenos resultados, y que es factible extender y diseñar nuevos operadores genéticos, orientados a múltiples problemas de optimización.

- Hemos demostrado lo eficiente que resulta el uso de los árboles binarios balanceados en la agrupación de documentos, aplicando un algoritmo genético que adopta dicha estructura, y que logra resultados que superan al algoritmo *Kmeans*.

Además, podemos mencionar, que debido al incremento en los volúmenes de información disponibles en forma electrónica y a la necesidad creciente de encontrar la información buscada en un tiempo mínimo, éste sistema tendría su aplicación para:

- Mejorar el rendimiento de los motores de búsqueda de información, mediante la agrupación previa de todos los documentos disponibles. Antes de comenzar a resolver las consultas, el conjunto de documentos podría ser separado en grupos afines por este sistema. A cada grupo de documentos se le asignaría un "representante de grupo". Luego, al resolver una consulta, no se examinarían todos los documentos, sino que se buscaría el "representante de grupo" que mejor responda a la consulta.

- Facilitar la revisión de consultas de documentos por parte del usuario final, agrupando los resultados luego de realizar la búsqueda. Cuando se realizan búsquedas sobre un gran volumen de documentos, la cantidad de resultados puede ser muy grande. Si la lista se presenta sin ningún tipo de procesamiento previo, el usuario del sistema se verá obligado a revisar todos los documentos, descartando aquellos que no son relevantes para él. Si los resultados se presentan agrupados jerárquicamente, los documentos del mismo grupo serán relevantes para una sub-consulta más específica que la original; de esta forma, los documentos podrían quedar separados en grupos temáticos, esta alternativa podría plantear problemas de rendimiento del algoritmo, pero tendría la gran ventaja de permitir al usuario que escogiera el nivel de agrupación que desea.

- Aplicar al campo de la Minería de Datos, permitiendo extraer conocimiento de los documentos procesados, a través de la agrupación de la información obtenida.

En relación con los resultados aportados en este trabajo se ha realizado ya las siguientes publicaciones:

- *''Methodology of Preprocessing of documents for Systems of Recovery of Information''*, [Castillo José Luis, Fernández del Castillo José R, León González, 2008] Proceedings of IADIS International Conference, Information System 2008. Algarve, Portugal Press- ISBN: 978-972-8924-57-7.

- *"Information Retrieval with Cluster Genetic"*, [Castillo José Luis, Fernández del Castillo José R, León González, 2008a] Proceedings of International Conference on Data Mining 2008. Amsterdam, The Netherlands, Press- ISBN: 978-972-8924-63-8.

- *"Agrupamiento de Documentos con Sistemas Evolutivos"*, [Castillo José Luis, Fernández del Castillo José R, León González, 2008b] Proceedings of IADIS International Conference Iberoamericana 2008, Lisboa, Portugal, Press- ISBN: 978-972-8924-72-0.

- *"Feature Reduction for Document Clustering with NZIPF Method",* [Castillo José Luis, Fernández del Castillo José R, León González, 2009] Proceedings of IADIS International Conference e-Society 2009, Barcelona, Spain, Press- ISBN: 978-972-8924-78-2.

- *"Group Method of Documentary Collections using Genetic Algorithms"* [Castillo José Luis, Fernández del Castillo José R, León González, 2009a] in Distributed Computing, Artificial Intelligence,Bioinformatics, Soft Computing, and Ambient Assisted Living. 10th International Work-Conference on Artificial Neural Network, IWANN 2009, Salamanca, Spain Proceedings Part II Springer, Press –ISBN:978-3-642-02480-1, LNCS 5518.

- *"Cluster of Reuters 21578 Collections using Genetic Algorithm and NZIPF method"*, [Castillo José Luis, Fernández del Castillo José R, León González, 2009b] Proceedings of International Conference on Data Mining 2009. Algarve, Portugal, Press- ISBN: 978-972-8924-88-1.

6.2. Líneas de trabajo futuras

A raíz de la investigación realizada, han quedado abiertas diferentes líneas de trabajo para un desarrollo futuro:

- Extensión de las técnicas evolutivas para agrupamiento de documentos considerando un orden difuso en la recuperación de los mismos. Puesto que los Sistemas de Recuperación de Información difusos admiten, al igual que el modelo vectorial, la posibilidad de devolver los resultados de la búsqueda ordenados por relevancia, podemos extender nuestras propuestas para derivar grupos que tengan en cuenta esta característica. Para ello, se pueden considerar funciones de *fitness* que tengan en cuenta dicho aspecto. Creemos, que con esta extensión, la eficacia de los grupos generados por nuestro algoritmo aumentaría significativamente.

- Uso de los Algoritmos Evolutivos multiobjetivo, para optimizar la función de adaptación. En nuestra opinión, el empleo de Algoritmos Evolutivos multiobjetivo en Recuperación de Información presenta una gran potencialidad puesto que el problema de la Recuperación de Información es bi-objetivo en sí al tener que optimizar dos criterios contradictorios, la precisión y la exhaustividad. Por tanto, creemos que el empleo de este tipo de Algoritmos Evolutivos en el campo de la Recuperación de Información podría proporcionar resultados muy prometedores.

- Mejorar la visualización de los contenidos cuándo se agrupe una gran colección documental, mostrando gráficamente los distintos niveles de una jerarquía, utilizando colores significativos asociados a cada metadato importante. La posibilidad de representar, en un único mapa, toda una colección formada por miles de documentos, e ir descendiendo en los diversos niveles jerárquicos que se hayan establecido, sin cambiar la forma de representación, proporcionaría una interfaz robusta, intuitiva y fácil de usar. Hay que tener en cuenta que, en la actualidad, grandes corporaciones están destinando muchos recursos en la búsqueda de sistemas de visualización para mostrar grandes colecciones de documentos, sin que hasta ahora se hayan conseguido unos resultados que conciten una satisfacción general.

REFERENCIAS BIBLIOGRÁFICAS

- [Ackley,1987] Ackley, D.H. *"A connectionist Machine for Genetic Hillclimbing"*. Kluwer Academic Press, 1987.

- [Alander, 1992] Alander. J.T. *"On optimal populations size of genetic algorithms"* en Proc CompEuro 92, págs 65-70. IEEE Computer Society Press, 1992.

- [Allipi, Cucchiara, 1992] Alippi C. y Cucchiara R. *"Cluster partitioning in image análisis classification: a genetic algorithm approach"*. En: Proc. Compeuro 92, págs. 139-144. IEEE Computer Society Press, 1992.

- [Allen et al,1993] Allen, R. B, Obry, P. y Littman, M. 1993. *"An interface for navigating clustered document sets returned by queries"*, Proceedings of the ACM Conference on Organizational Computing Systems.

- [Al-Sultan, 1995] Al-Sultan. K.S. *"A tabu search approach to the clustering problem. Patter Recognition"*, 28 (9):1443-1451, 1995.

- [Angeline, 1996] Angeline PJ. *"Genetic Programming's Continued Evolution"*. Angeline PJ, Kinnear KE, Eds. Advances in Genetic Programming Volume 2. MIT Press, 1996:1-20.

- [Arabas, Michalewicz, Mulawka,1994] Arabas J., Z. Michalewicz y J Mulawka. *"GAVaPs – a genetic algorithm with varying population size"*. Proceedings of the First IEEE Conference on Evolutionary Computation, Vol I, pags-73-79. IEEE Computer Society Press. 1994.

- [Bäck, 1994] Bäck T. Selective pressure in evolutionary algorithms: *"A characterization of selection mechanisms"*. En: DB. Fogel (ed), Proceedings of the First IEEE Conference on Evolutionary Computation, vol I, págs 5-62. IEEE Computer Society Press, 1994.

- [Bäck, 1996] Bäck T., *"Evolutionary Algorithms in theory and Practice"*, Oxford University Press, 1996.

- [Bäck, Fogel, Michalewicz 1997] Bäck T. Fogel DB, Michalewicz Z, Eds. *"Handbook of Evolutionary Computation"*. Institute of Physics Publishing and Oxford University Press, 1997.

- [Bäck, Schwefel, 1991] Bäck T, Schwefel HP. *"Extended Selection Mechanisms in Genetic Algorithms"*. Proceedings of the Fourth International Conference on Genetic Algorithms. San Diego: Morgan Kaufmann Publishers, 1991:92-9.

- [Baeza-Yates, Ribeiro Neto, 1999] Baeza-Yates R, Ribeiro-Neto B. *"Modern Information Retrieval"*. ACM Press Addison-Wesley, 1999. ISBN:0-201-39829-X .

- [Banzhaf, 1990] Banzhaf W. *"The 'molecular' travelling salesman problem"*. Biologica Cybernetics, 64:7-14, 1990.

- [Banzhaf et-al, 1998] Banzhaf W, Nordin P, Keiler R, Francone F. *"Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and its Applications"*, San Francisco:Morgan Kaufmann, 1998.

- [Berry Michael, 2004] Berry Michael, "*Survey of Text Mining – Clustering Classification and Retrieval "*, Springer – Verlag 2004. ISBN : 0-387-955-63-1

- [Berry Michael, et al, 2008] Berry Michael, Malu Castellano Editors: *"Survey of Text Mining II "*, Springer 2008. ISBN: 978-1-84800-045-2

- [Bezdeck et-al, 1994] Bezdeck J.C., Boggavaparu S. L.O. Hall y Bensaid A. *"Genetic algorithms guided clustering"*. En: D.B. Fogel (ed), Proceedings of the first IEEE Conference on Evolutionary Computation, volume I, págs. 34-40. IEEE Computer Society Press 1994 citado en ([Cole, 1998])

- [Bhuyan et-al, 1991] Bhuyan J.N., Raghavan V.V y Elayavalli V.K. *"Genetic Algorithms with an ordered representation"*. En: R. Belew y L.B. Booker (eds), proc. Of the Fourth International Conference on genetic Algorithms, págs 408-415. Morgan Kaufmann, 1991.

- [Bookstein, 1980] Bookstein A. Fuzzy Request: *"An approach to weighted Boolean Search"*. Journal of the American Society for Information Science and Technology 1980; 31:240.

- [Bookstein, 1983] Bookstein A. *"Outline of a General Probabilistic Retrieval Model"*. Journal of Documentation 1983; 39(2):63-72. (citado en [Zarco Carmen,2003]).

- [Bordogna, Carrara, Pasi, 1995] Bordogna G, Carrara P, Pasi G, *"Fuzzy Approaches to extend Boolean Information Retrieval"* Eds. Fuzzy sets and possibility theory in database management systems. Springer Verlag, 1995:231-74.

- [Bradley, Fayyad, 1998] Bradley, P. S. y Fayyad, U. M. 1998. *"Refining initial points for k- means clustering"*. In J. Shavlik, editor, Proceedings of the Fifteenth International Conference on Machine Learning (ICML ' 98, pages 91- 99, San Francisco, CA, 1998. Morgan Kaufmann.

- [Buell , Kraft, 1981] Buell DA, Kraft DH. *"Threshold values and Boolean Retrieval Systems"*. Information Processing and Management. 1981; 17:127-36. (citado en [Zarco Carmen,2003]).

- [Castillo José Luis, Fernández del Castillo José R, León González, 2008] *"Methodology of Preprocessing of documents for Systems of*

Recovery of Information", Proceedings of IADIS International Conference, Information System 2008. Algarve, Portugal 9-11 April 2008, 25(1):324-327. Press- ISBN: 978-972-8924-57-7.

- [Castillo José Luis, Fernández del Castillo José R, León González, 2008a] *"Information Retrieval with Cluster Genetic"*, Proceedings of IADIS International Conference on Data Mining 2008. Amsterdam, The Netherlands, July 2008,25(1):77-81.Press- ISBN: 978-972-8924-63-8.

- [Castillo José Luis, Fernández del Castillo José R, León González, 2008b] *"Agrupamiento de Documentos con Sistemas Evolutivos"*, Proceedings of IADIS International Conference Iberoamerica. Lisboa, Portugal, Diciembre 2008, 25(1):305-312. Press- ISBN: 978-972-8924-72-0.

- [Castillo José Luis, Fernández del Castillo José R, León González, 2009] *"Feature reduction for document clustering with NZIPF method"*, Proceedings of IADIS International Conference e-Society 2009. Barcelona, Spain, February 2009, 25(1):205-209. Press- ISBN: 978-972-8924-78-2.

- [Castillo José Luis, Fernández del Castillo José R, León González, 2009a] *"Group Method of Documentary Collections using Genetic Algorithms"* in Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. 10th International Work-Conference on Artificial Neural Network, IWANN Salamanca, Spain June 2009, 25(1):992-1000, Proceedings Part II Springer, Press – ISBN:978-3-642-02480-1, LNCS 5518.

- [Castillo José Luis, Fernández del Castillo José R, León González, 2009b] *"Cluster of Reuters 21578 Collections using Genetic Algorithm and NZIPF method"*, Proceedings of IADIS International Conference on Data Mining 2009. Algarve, Portugal, June 2009, 25(1):174-177, Press- ISBN: 978-972-8924-88-1.

- [Castro Felix, et al, 2007] Castro Felix, Vellido Alfredo, Nebor Angela, Mujica Francisco, "Applying Data Mining Techniques to e-Learning Problem" en Evolution of Teaching and Learning Paradigms in Intelligent Environment, [183-222]. Springer 2007. Press ISBN: 978-3-540-71973-1.

- [Chambers L, 2001] Chambers Lance *"The Practical Handbook of Genetic Algorithms Aplications"* Second Edition 2001 by Chapman & HALL /CRC ISBN :1-58488-2409-9.

- [Chowdhury, 1999] Chowdhury GG. (1999): Introduction to Modern Information Retrieval; Library Association Publishing. London ISBN:1-85604-318-5.

- [Cios et al, 2007] Cios Krzysztof, Pedrycz Witold, Swiniarski Roman, Lukasz Kurgan, *"Data Mining: A Knowledge Discovery Approach"*, Springer, 2007, ISBN-13: 978-0-387-33333-5.

- [Cleverdon, 1972] Cleverdon CW. *"On the inverse Relationship of Recall and Precision"*. Journal of Documentation 1972; 28:195-201. (citado en [Zarco Carmen, 2003]).

- [Cole, 1998] Cole, Rowena M. 1998. *"Clustering with Genetic Algorithms"*. Thesis for the degree of Master of Science, Department of Computer Science, University of Western Australia.

- [Cordon, Herrera, Hoffman, Magdalena, 2001] Cordon O, Herrera F, Hoffman F, Magdalena L. *"Genetic Fuzzy Systems- Evolutionary Tuning and Learning of Fuzzy Knowledge Bases"*. Word Scientific, 2001.

- [Cordón,Herrera-Viedma, Luque, 2002] Cordón O, Herrera-Viedma E, Luque M. *"Evolutionary Learning of Boolean Queries by Multiobjective Genetic Programming"*. Proceedings of the Parallel Problem Solving from Nature VII (PPSN-VII). Lecture Notes in Computer Science 2439. 2002: 710-9.

- [Cordon, Herrera-Viedma,Zarco, 2003] Cordón O, Herrera-Viedma E. Zarco Carmen *"A Review on the Application of Evolutionary Computation to Information Retrieval"* Dept. of Computer Science and A.I University of Granada, Puleva Food S.A.

- [Cutting et al, 1992] Cutting, Karger D, Pedersen D, y Tukey J. W. 1992. Scatter/ Gather: *"A cluster- based approach to browsing large document collections"*, Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval, Páginas 318- 29.

- [Dash , Liu, 2001] Dash, M, y Liu, H. 2001. *"Efficient Hierarchical Clustering Algorithms Using Partially Overlapping Partitions"*. Pacific- Asia Conference on Knowledge Discovery and Data Mining, páginas 495-506.

- [Davidor, 1991] Davidor Y. *"Genetic Algorithms and Robotics: a Heuristics Strategy for Optimization".* World Scientific, 1991.

- [Davis, 1985] Davis L. *"Applying adaptative algorithms to epistatic domains"*. En: proceedings of the International Joint Conference on Artificial Intelligence, págs. 162-164, 1985.

- [De Jong, 1975] De Jong, K.A. 1975. *"An analysis of the behavior of a class of genetic adaptive systems"*. Dissertation Abstracts International 36 10, 514B. University of Michigan, Microfilm No. 76- 9381.

- [De Jong et al, 1993] De Jong KA, Spears WM, Gordon DF. *"Using Genetic Algorithms for Concept Learning"*. Machine Learning 1993; 13:161-88.

- [Dervin, Nilan, 1986] Dervin B, Nilan M. *"Information Needs and Uses"*. Annual Review of Information Science and Technology 1986; 21:3-33. (citado en [Zarco Carmen,2003])

- [Diaz A, 1996] Diaz A. *"Optimization Heurística y Redes Neuronales"*. Editorial Paraninfo, 1996.

- [Eiben A, Marchiori A, Valko A, 2001] Eiben A.E, Marchiori E, Valkó V.A *"Evolutionary Algorithms with on the fly population size adjustement"* Univeristei Amsterdam.

- [Eiben A, Hintending R, Michalewicz Z., 1999] Eiben Agoston, Hinterding Robert, Michalewicz Zbigniew *"Parameter control in Evolutionary Algorithms"*, 1999, in IEEE Transactions on Evolutionary Computation Vol 3, N0 2 July 1999,

- [Estivill – Castro, 2000] Estivill- Castro, V. 2000. *"Hybrid Genetic Algorithms are Better for Spatial Clustering"*. Pacific Rim International Conference on Artificial Intelligence, pages 424- 434.

- [Eurovoc, 2006] *"Tesauro Eurovoc. Presentación alfabética permutada. Edición 4.2 – Lengua Española"*. ISSN 1725-426 Comunidades Europeas, 2006.

- [Falkenauer, 1999] Falkenauer E. 1999. *"Evolutionary Algorithms: Applying Genetic Algorithms to Real- World Problems"*. Springer, New York, Pag 65- 88. (citado en [Zarco Carmen,2003]).

- [Faloutsos et al, 1996] Faloutsos, Christos y Oard D. W. 1996. *"A survey of Information Retrieval and Filtering Methods"*. Technical Report CS-TR3514, Dept. of Computer Science, Univ. of Maryland.

- [Fan, Gordon, Pathak, 1999] Fan W, Gordon MD, Pathak P. *"Automatic Generation of Matching Functions by Genetic Programming for Effective Information Retrieval"*. Proceedings of the 1999 America's Conference on Information Systems. Milwaukee: 1999: 49-51.

- [Fan, Gordon , Pathak,2000] Fan W, Gordon MD, Pathak P. *"Personalization of Search Engine Services for Effective Retrieval and Knowledge Management"*. Proceedings of the 2000 International Conference on Information Systems (ICIS). Brisbane (Australias):2000: 20-34.

- [Fasulo, 1999] Fasulo, Daniel. 1999. *"An analysis of recent work on clustering algorithms"*. Technical Report # 01- 03- 12, Dept. of Computer Science & Engineering, University of Washington.

- [Fogarty, 1989] Fogarty T.C. *"Varying the probability of mutation in the genetic algorithm"*. En: J.D. Schaffer (ed.), Proceedings of Third International Conference on Genetic Algorithms, págs 104-109. Morgan Kaufmann, 1989.

- [Fogel D, 1988] Fogel. D.B *"An evolutionary approach to the travelling salesman problems"*. Biological Cybernetics, 60: 139-144, 1988.

- [Fogel L, 1966] Fogel L.J, *"Artificial Intelligence Through Simulated Evolution"*. John Willey and Sons, 1966.

- [Frakes , Baeza Yates, 1992] Frakes W. *"Introduction to Information Storage and retrieval Systems"*. Frakes W, Baeza-Yates R, Eds. Information Retrieval. Data Structures & Algorithms.Prentice Halls,1992:

- [Fuhr, 1992] Fuhr N. *"Probabilistic Models in Information Retrieval"*. Computer Journal 1992; 35(3):243-55.

- [Geyer-Schulz, 1995] Geyer-Schulz A. *"Fuzzy Rule-Based Expert Systems and Genetic machine Learning"* Physica-Verlag, 1995.

- [Glover, 1989] Glover F, *"Tabu Search-part I"*. ORSA Journal on Computing 1(3): 190-206, 1989.

- [Glover, 1990] Glover F. *"Tabu Search-part II"* ORSA Journal on Computing 2(1):4-32, 1990.

- [Goffman W, 1975] citado por Miranda Lee, Pao in *"Automatic text analysis based on transition phenomena of word occurences"*, como Comunicación Personal 1975.

- [Goldbert, 1989] Goldberg D.E. *"Genetic algorithms in search, optimization, and machine learning"*. Addison-Wesley, 1989.

- [Goldbert, Deb, 1991] Goldberg D.E. y Deb K. *"A comparative analysis of selection schemes used in genetic algorithms"*. En: G.J.E. Rawlins (ed.), Foundations of Genetic Algorithms, págs 69-93. Morgan Kaufmann, 1991.

- [Gordon, 1991] Gordon M.D. *"User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm"*. Journal of the American Society for Information Science 1991; 42(5):311-22.

- [Grefenstette, 1995] Grefenstette J.J. *"Genetic Algorithms for Machine Learning"*. Kluwer Academic Publishers, 1995.

- [Han, Lamber, Tung, 2001] Han, J, Kamber, M. y Tung, A. K. H. 2001. *"Spatial clustering methods in data mining: A survey. Geographic Data Mining and Knowledge Discovery"*, H. Miller and J. Han, editors, Taylor and Francis.

- [Harman, 1986] Harman D.W. *"An Experimental Study of Factors Important in Document Ranking"* ACM Conference on Research and Development in Information Retrieval. Pisa, Italia: 1986:186-93.

- [Hesser, Manner, 1990] .Hesser J. y Manner R. *"Towards an optimal mutation probability for genetic algorithms"*. En: Parallel Problem Solving from Nature. Lectures Notes in Computer Science, volume 496, págs. 23-32. Springer-Verlag, 1990.

- [Holland, 1975] Holland J.H. *"Adaptation in natural and artificial systems"*. The University of Michigan Press, Ann Arbor, MI, 1975.

- [Horng, Yeh, 2000] Horng J-T, Yeh C-C. *"Applying Genetic Algorithms to Query Optimization in Document Retrieval"*. Information Processing and Management 2000; 36:737-59.

- [Ingwersen, Willet, 1995] Ingwersen P, Willet P. *"An Introduction to Algorithmic and Cognitive Approaches for Information Retrieval"*. Libri 1995; 45(3-4):169-77.

- [Jain, Murty, Flinn, 1999] Jain A. K, Murty M. N, y Flinn, P.J. 1999. *"Data Clustering: A review"*. ACM Computing Surveys, Vol. 31, Nro 3, Septiembre 1999.

- [Janikov, 1993] Janikov C.Z. *"A Knowledge-Intensive Genetic Algorithm for Supervised Learning"*. Machine Learning 1993; 13:189-228.

- [Jones, Beltramo, 1990] Jones D.R. y Beltramo M.A.. *"Clustering with genetic algorithms"*. Informe Interno GMR-7156, Operating Sciences Department. General Motors Research Laboratories,1990.(citado [en Cole,1998]).

- [Jones, Beltramo, 1991] Jones D.R. y Beltramo M.A. *"Solving partitioning problems with genetic algorithms"*. En: R. Belew y L.B. Booker (eds), proc. Of the Fourth International Conference on genetic Algorithms, págs. 442-449. Morgan Kaufmann, 1991.

- [Kaufmann, Rousseeuw, 1990] Kaufmann, L. y Rousseeuw, Peter J. 1990. *"Finding Groups in data: An introduction to Cluster Analysis"*, John Wiley & Sons, Inc, NY.

- [Klir, Yuan, 1995] Klir G.J, Yuan B. *"Fuzzy Sets and Fuzzy Logic"*. Prentice Hall, 1995.

- [Korfhage, 1997] Korfhage R. *"Information Storage and Retrieval"*. New York: Wiley, 1997.

- [Kowalsky, 1998] Kowalsky Gerald (1998): Information Retrieval Systems Kluwer Academic Publishers. Isbn:0-7923-9899-8.

- [Koza, 1991] Koza J. *"Evolving a Computer Program to Generate Random Numbers Using the Genetic Programming Paradigm"*. Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, California: 1991:37-44.

- [Koza, 1992] Koza J. *"Genetic Programming: On the Programming Computers by Means of Natural Selection"*. Cambridge: MIT Press, 1992.

- [Koza, 1994] Koza J. *"Genetic Programming II: Automatic Discovery of Reusable Programs"*. Cambridge: MIT Press, 1994.

- [Koza, et al, 1999] Koza John, Bennet Forrest, Andre David, Keane Martin *"Genetic Programming III, darwinian invention and problem solving"*, 1999 Morgan Kauffman Publshers Inc, ISBN; 1-55860-543-6.

- [Kraft et al, 1997] Kraft D. Petry F.E, Buckles B.P, Sadasivan T. *"Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback."* Eds Genetic Algorithms and Fuzzy Logic Systems. 1997:155-73.

- [Krovetz, 1993] Krovetz, R. 1993. *"Viewing morphology as an inference process"*. In Proceedings of ACM- SIGIR93, pages 191- 203

- [Laarhoven Van, Aarts, 1987] Laarhoven P.J, Van M. y. Aarts E.H.L. *"Simulated Annealing: Theory and Applications."* D. Reidel Publishing Company, Dordretch, Holland, 1987.

- [Larrañaga, Poza, 1994] Larrañaga P. y Poza M.. *"Structure learning of Bayesian networks by genetic algorithms"*. En: E. Diday (ed.), New Approaches in Classification and Data Analysis, págs 300-307. springer-Verlag, 1994.

- [Larrañaga et al, 1999] Larrañaga P., Kuijpers C.M.H., Murga R.H., y Dizdarevic S.. *"Genetic Algorithms for the travelling salesman problem: A review of representations and operators"*. Artificial Intelligence Review, 1999.

- [Larose, 2006] Larose Daniel T. *"Data Mining Methods and Models"* A John Wiley & Sons, Inc. Publication, 2006. ISBN-13: 978-0-471-66656-1.

- [Leousky, Croft, 1996] Leousky, A. V. y Croft, W. B.1996. *"An evaluation of techniques for clustering search results"*, Technical Report IR- 76, Department of Computer Science, University of Massachusetts, Amherst.

- [Lewis, 1991] Lewis, D. 1991. *"Evaluating text categorization"*, Proceedings of the Speech and Natural Language Workshop, Asilomar, Morgan.

- [Lewis, 1997] Lewis, D. 1997. *"Reuters- 21578 text categorization test collection"*, (link consultado en junio 2009) http://www.daviddlewis.com/resources/testcollections/reuters21578/

- [Liu, 1968] Liu G. L. 1968. *"Introduction to combinatorial mathematics"*, McGraw Hill.

- [Lucasius, Dane, Kateman, 1993] Lucasius C.B., Dane A.D.y Kateman G. *"On k-medoid clustering of large data sets with the aid of a genetic algorithms: background, feasibility and comparison"*. Analytica Chimica Acta, 282:647-669, 1993.

- [Maarek et al, 2000] Maarek Y. S, Ben- Shaul, Israel Z. y Pelleg, Dan. 2000. *"Ephemeral Document Clustering for Web Applications"*. IBM Research Report RJ 10186.

- [MacQueen, 1967] MacQueen. J.B. *"Some methods for classification and analysis of multivariate observations"*. En: Proceedings of Fifh Berkeley Synposium, volume 2, págs. 281-297, 1967.

- [Macskassy et al, 2001] Macskassy, S. A, Banerjee, A, Davison, B. D, Hirsh, H. 2001. *"Human performance on clustering web pages"*. Technical Report DCS- TR- 355, Department of computer Science, Rutgers, State University of New Jersey.

- [Michalewicz, 1999] Michalewicz Z. *"Genetic Algorithms + Data Structures = Evolution Program"*. Berlin: Springer-Verlag, 1999 ISBN:3-540-60676-9 .

- [Miller, Goldberg, 1995] Miller B. L, Goldberg D. E. 1995. *"Genetic algorithms, Selection Schemes and the Varying Effects of Noise"*, IlliGAL report No. 95009.

- [Mirkin, 1996] Mirkin B. *"Mathematical Classification and Clustering"*. Kluwer Academic Publishers, 1996.

- [Mitchell, 1996] Mitchell M. *"An introduction to Genetic Algorithms"*. The MIT Press, Cambridge, Massacusetts, 1996.

- [Mitchell Tom, 1997] Mitchell Tom. *"Machine Learning"*. Mc-Graw Hill, 1997.

- [Moreiro González, 2002] Moreiro González José Antonio "*Aplicaciones al análisis del contenido provenientes de la teoría matemática de la información*", Anales de Documentación Número 5, 2002, pags 273-286.

- [Muñoz García, 1993] Muñoz García Alberto (1993): *"Técnicas de Recuperación de Información en Bases de Datos Documentales"*; Universidad de Salamanca, Cuaderno de ADAB 389-403.

- [Nannen V, Smit K, Eiben E, 2006] Nannen Volker, Smit S.K., Eiben E. "Costs and benefits of tuning parameters of evolutionary algorithms", Vrje Universitei Amsterdam.

- [Nils Nilsson, 2001] Nils Nilsson (2001): Inteligencia Artificial: Una Nueva Sintesis ,McGraw Hill. ISBN:1-55860-467-7

- [Nordin, 1997] Nordin P. *"Evolutionary Program Induction of Bynary Machine Code and its Aplication"*. Krehl-Verlag, 1997.

- [Olson David, 2008] Olson D. *"Advanced Data Mining Techniques"*, Springer 2008 ISBN:978-3-540-76916-3.

- [Pao M.L, 1977] Pao M.*"Automatic indexing based on Goffman transition of word occurrences"*. In American society for Information Science. Meeting (40th: 1977:Chicago). Information Management in the 1980's: proceedings of the ASIS annual meeting 1977, volume 14:40th annual meeting, Chicago. (citado in [Urbizagastegui Ruben, 1995] y [Moreiro González, 2002])

- [Pao M.L, 1978] Pao M.*"Automatic text analysis based on transition phenomena of word occurences"*. In journal of American society for Information Science Vol. 29, issue 3 1978 pag 121-124.

- [Pathak, Gordon, Fan, 2000] Pathak P, Gordon M, Fan W. *"Effective Information Retrieval using Genetic Algorithms based Matching Function Adaptation"*. Proceedings of the 33rd Hawaii International Conference on System Science (HICSS). Hawaii: 2000:122-5.

- [Pedrycz W, 1997] Pedrycz W. *"Fuzzy Evolutionary Computation"*. Kluwer Academia, 1997.

- [Pedrycz W, 1998] Pedrycz Witold (1998): Computational Intelligence CRC Press ISBN:0-8493-2643-5.

- [Pentakalos, Menascé, Yesha, 1996] Pentakalos, O, Menascé, D. y Yesha, Y. 1996. *"Automated Clustering- Based Workload Characterization"*. 5th NASA Goddard Mass Storage Systems and Technologies Conference.

- [Porter, 1980] Porter, M. F, 1980. *"An Algorithm for Suffix Stripping, Program"*, vol. 14, no. 3, 130- 137, 1980.

- [Qin He, 1996] Qin He, 1996. *"A review of clustering algorithms as applied in IR"*, UIUCLIS- 1996/ University of Illinois at Urbana- Champaign.

- [Raghavan, Bollman, Jung, 1989] Raghavan V, Bollmann P, y Jung G. 1989. *"A critical investigation of recall and precision as measures of retrieval system performance"*. ACM Transactions on Information Systems, 7(3: 205- 229.

- [Reeves, 1993a] Reeves C.R. *"Modern Heuristic Tecniques for Combinatorial Optimization"*. Blackwell Scientific Publications, 1993.

- [Reeves, 1993b] Reeves C.R. *"Using genetic algorithms with small populations"*. En: S. Forrest (ed). Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, 1993.

- [Robertson, Willet, 1994] Robertson A.M, Willet P. *"Generation of Equifrecuent Groups of Words using a Genetic Algorithms"*. Journal of Documentation 1994; 50(3):213-32.

- [Robertson, Willet, 1996] Robertson A.M, Willet P. *"An Upperbound to the Performance of Ranked-Output Searching: Optimal Weighting of Query Terms using a Genetic Algorithm"*. Journal of Documentation 1996; 52(4):405-20.

- [Rudolph, 1994] Rudolph G. *"Convergence analysis of canonical genetic algorithms"*. IEEE Transactions on neural networks,5(1):96-101, 1994.

- [Rudolph, 1997] Rudolph G, *"Convergence Properties of Evolutionary Algorithms"* Kovac, Hamburg, 1997.

- [Rüger, Gauch, 2000] Rüger S. y Gauch S. E. 2000. *"Feature reduction for document clustering and classification"*. Technical report, Computing Department, Sudmitted SIGIR.

- [Rusell Stuart, Norvig Peter, 1996] Rusell Stuart, Norvig Peter. *"Artificial Intelligence: A Modern Approach"*. Prentice Hall, 1996. ISBN:0-13-103805-2.

- [Salton, 1971] Salton G, Ed. The Smart Retrieval System. *"Experiments in Automatic Document Processing"*. Englewood Cliffs: Prentice-Hall, 1971.

- [Salton, McGill, 1983] Salton G, McGill M.J. *"An Introduction to Modern Information Retrieval"*. McGraw-Hill, 1983.

- [Schaffer et al , 1989] Schaffer J.D, Caruna R.A., Eshelman L.J. *"A study of control parameters affecting online performance of genetic algorithms for functions optimizations"*. En: J.D. Schaffer (ed.). Proceedings of Third International Conference on Genetic Algorithms, págs. 51-60. Morgan Kaufmann, 1989. citado en [Eiben A, Hintending R, Michalewicz Z., 1999].

- [Schaffer et al, 1992] Schaffer JD, Whitley D, Eshelman L.J. *"Combinations of Genetic Algorithms and Neural Network: A survey of the State of the Art"*. Proceedings of the COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks. 1992:1-37.

- [Shapiro, 2001] Shapiro Jonathan, *"Genetic Algorithms in Machine Learning"* in Machine Learning and its applications" Springer-Verlag Berlin 2001 pp:146-168, 2001. ISBN:3-540-42490-3.

- [Schwefel, 1995] Schwefel H.P. *"Evolution and Optimum Seeking"*. New York: John Wiley and Sons, 1995.

- [Schütze et al, 1997] Schütze, Hinrich y Silverstein Craig (1997) *"Projections for Efficient Document Clustering"*, in Proceedings of ACM/ SIGIR' 97, pp. 74- 81.

- [Smith, Smith, 1997] Smith M.P, Smith M. *"The Use of Genetic Programming to Build Boolean Queries for Text Retrieval through Relevance Feedback"*. Journal of Information Science 1997; 23(6):423-31.

- [Steinbach, Karypis, Kumar, 2000] Steinbach M, Karypis G, y Kumar V. *"A comparison of Document Clustering Techniques"*. Technical Report # 00- 034. University of Minnesota. In KDD Workshop on Text Mining.- 2000.

- [Strehl et al, 2000] Strehl A, Ghosh J. y Mooney R. *"Impact of Similarity Measures on Web- page Clustering"*. Workshop of Artificial Inteligence for Web Search. 2000.

- [Suzuki, 1995] Suzuki J. *"Analysis on simple genetic algorithms"*. IEEE Transactions on Systems, Man, and Cybernetics, 25(4):655-659, 1995.

- [Urbizagastegui Ruben, 1995] Urbizagástegui Ruben *"Las Posibilidades de la ley de Zipf en la indización automática"*, Universidad de California, Riverside USA CA 92521-5900

- [Van Rijsbergen, 1979] Van Rijsbergen C.J. *"Information Retrieval"*. Butterworth, London, 1979. Available at Computing Science. University of Glasgow.

- [Velasco, et al, 2004], Velasco I, Díaz J, Lloréis A *" Algoritmo de Filtrado Multi-término para la obtención de relaciones jerárquicas en la construcción automática de un tesauro*", Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica, 2004.

- [Vrajitoru, 1998] Vrajitoru D. *"Crossover Improvement for the Genetic Algorithms in Information Retrieval"*. Information Processing & Management 1998; 34(4):405-15.

- [Winston, 1994] Winston P.H. *"Inteligencia Artificial"*. Addison-Wesley Iberoamericana, 1994.

- [Wong, 2002] Wong Man Leung, Kwong S. *"Data Mining using grammar based Genetic Programming and Applications*" Kluwer Academic Publishers – Genetic Programming Series – Series Editor John Koza. ebook ISBN:0-306-47012-8.

- [Yang, Korfhage, 1994] Yang J.J, Korfhage R.R. *"Query Modification using Genetic Algorithms in Vector Space Models"*. International Journal of Expert Systems 1994; 7(2):165-91.

- [Yang, 1999] Yang Y. 1999. *"An evaluation of statistical approaches to text categorization"*. School of Computer Science, Carnegie Mellon University, Information Retrieval 69-90 CMU- CS- 97- 127. Information Retrieval 69-90 Kluwer Academic.

- [Yang, Pedersen, 1997] Yang Y. y Pedersen J, 1997. *"A comparative study on feature selection in text categorization"*. Proc. of the 14th International Conference on Machine Learning ICML97, páginas 412- 420.

- [Ye, 2003] Ye N., *"The Handbook of Data Mining"* Lawrence Erlbaum Associates, Publishers, London, ISBN: 0-8058-4081-8, 2003.

- [Zadeh, 1965] Zadeh LA. *"Fuzzy Sets. Information and Control"* 1965; 8:338-53.

- [Zamir, Oren, Etzioni, 1998] Zamir, Oren y Etzioni, Oren.1998. *"Web Document Clustering: A feasibility demonstration"*. Proceedings of ACM/ SIGIR' 98.

- [Zamir, Oren, Etzioni, 1999] Zamir, Oren y Etzioni, Oren. 1999. *"Grouper: A Dynamic Clustering Interface to Web Search Results"*. Proceedings of the Eighth International World Wide Web Conference, Computer Networks and ISDN Systems.

- [Zarco Carmen, 2003] Zarco Fernández, Carmen. Tesis Doctoral: "*Aplicación de los Algoritmos Evolutivos al Aprendizaje Automático de Consultas Booleanas Extendidas para Sistemas de Recuperación de Información Difusos".* Universidad de Granada, 2003.

- [Zervas et al, 2000] Zervas, Giorgio y Rüger, S. *"The curse of dimensionality and document clustering"*. Dept. of Computing, Imperial College, England 2000.

- [Zhao, Karypis, 2001] Zhao Y. y Karypis G, 2001 *"Criterion Functions for Document Clustering"*. Technical Report # 01- 40, Department of Computer Science, University of Minnesota.

- [Zhao, Karypis,2002] Zhao Y y Karypis G. *"Evaluation of Hierarchical Clustering algorithms for document datasets"* Technical Report #02-22. Department of Computer Science University of Mynnesota.

- [Zimmerman, 1996] Zimmermann H.J. *"Fuzzy Sets. Theory and its Applications"*. Kluwer Academic Press, 1996.

- [Zipf, 1949] Zipf G.K. *"Human Behavior and the Principle of Least Effort: An introduction to human ecology"*. Cambridge: Addison Wesley, 1949.(Citado en [Baeza-Yates, Ribeiro Neto, 1999]).

Anexos

En este apartado detallaremos los requisitos funcionales, de control y la descripción de cada uno de los elementos de la aplicación. También presentaremos los resultados obtenidos en cada una de las pruebas realizadas con el Sistema Genético desarrollado.

Anexo 1

Requisitos Funcionales, de Control y descripción de cada uno de los elementos del Sistema

En este anexo mostramos en tablas cada uno de los requisitos funcionales, los requisitos de seguridad, requisitos de control y la descripción de cada uno de los elementos del Sistema Evolutivo.

Tabla 4.4 Requisitos Funcionales del Sistema Evolutivo

Identificador de Requisito: RQ-F-001

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir quitar las marcas (en formato SGML) a cada distribución que se seleccione de la Base documental Reuters,

Identificador de Requisito: RQ-F-002

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Realizar el Proceso de Filtraje, a fin de delimitar los documentos para cada distribución de la base documental con la finalidad de tener documentos que pertenezcan a una sola categoría, para procesar solamente documentos de la base "Reuters" que pertenezcan a una sola categoría.

Identificador de Requisito: RQ-F-003

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar el proceso de "Zonning" a los documentos obtenidos del proceso de filtraje, extrayendo solamente el cuerpo del documento para cada uno de los documentos de cada distribución, a fin de delimitar el contenido de los mismos y luego procesar solamente las palabras que están contenidas en dicho cuerpo, obteniéndose el texto libre principal (Free Text) de cada uno.

Identificador de Requisito: RQ-F-004

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar el proceso de Stop List a cada una de los documentos de las distribuciones de la Base Documental, tratando de eliminar las palabras funcionales del lenguaje (artículos, preposiciones, etc.) cuyo valor como término para la indización y para la discriminación es muy bajo. Creando una lista de palabras vacías en inglés y en español (dependiendo del tipo de documento a procesar) que ayudará a ir eliminando cada una de las palabras carentes de interés en los documentos.

Identificador de Requisito: RQ-F-005

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar un proceso de **"stemming"** para reducir el tamaño de los ficheros índices. Almacenando sólo las raíces de los términos, aplicando el algoritmo de Porter a todos los documentos

Identificador de Requisito: RQ-F-006

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Calcular la frecuencia de los términos obtenidos para todos los documentos, para saber los términos que son más utilizados; desarrollando un modulo que calcule estos valores para cada uno de los documentos que se obtuvieron en los procesos anteriores.

Identificador de Requisito: RQ-F-007

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Seleccionar aquellos términos con mayor poder discriminatorio y proceder a su ponderación. Para ello implementar la ley de Zipf, (aplicando la variante propuesta por Goffman), y su posterior vectorización de los documentos, construyendo los vectores documentales con los términos significativos seleccionados.

Identificador de Requisito: RQ-F-008

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Ponderar y normalizar cada uno de los términos para tener una representación vectorial normalizada los más coherente y fiable de los documentos; aplicando para ello la asignación de pesos IDF.,que luego se normaliza con la formula: $w_{ij} = f_{ij} * (\log_2 N - \log_2 n_j + 1)$ Como resultado de este proceso obtenemos los vectores característicos para todos los documentos del SRI con todos los términos ponderados.

Identificador de Requisito: RQ-F-009

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios seleccionar la distribución Reuters que se desea procesar, para poder luego aplicar todos los procesos del subsistema de procesamiento sobre dicha distribución.

Identificador de Requisito: RQ-F-010

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios ver cada uno de los pasos del procesamiento documental, a fin de saber el correcto proceso de los mismos.

Identificador de Requisito: RQ-F-011

Subsistema:	Subsistema de Procesamiento Documental
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Almacenar en ficheros temporales cada uno de los diferentes resultados obtenidos en el procesamiento documental para su posterior tratamiento y revisión.

Identificador de Requisito: RQ-F-012

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Crear cada individuo de la población inicial de tal forma que cada individuo sea gramaticalmente diferente, hacer un recorrido del cromosoma generado para cada individuo en Preorden para insertar cada nodo en el árbol (de cada individuo) y fijar un umbral de profundidad para limitar el tamaño de los árboles creados

Identificador de Requisito: RQ-F-013

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios ver cada uno de los individuos creados, así como también la representación del cromosoma correspondiente a dicho individuo.

Identificador de Requisito: RQ-F-014

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Evaluar cada uno de los individuos creados, aplicando para ello la función de adaptación con los operadores definidos para el individuo. Este proceso se aplicará para todos los nodos función del cromosoma, siendo el nodo raíz el fitness global del individuo, para ello se deberá calcular los centroides correspondiente para cada nodo función del cromosoma y evaluar la función de distancia euclidea y la función de similitud del ángulo del coseno en cada nodo función desde los nodos inferiores del cromosoma hasta el nodo raíz del cromosoma.

Identificador de Requisito: RQ-F-015

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Comparar cada uno de los diferentes fitness globales de la población inicial, a fin de seleccionar a través del elitismo el mejor individuo (cromosoma) para las siguientes generaciones.

Identificador de Requisito: RQ-F-016

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Pasar toda la población inicial creada al siguiente modulo evolutivo del sistema a fin de poder aplicar los operadores y las técnicas genéticos en las siguientes generaciones.

Identificador de Requisito: RQ-F-017

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios ver cada uno de los pasos realizados en este módulo, así como también la probabilidad de mutación utilizada, probabilidad de cruce, la semilla utilizada para aplicar los números aleatorios, el tamaño de la población, el número de generaciones, y otros parámetros relacionados con este subsistema.

Identificador de Requisito: RQ-F-018

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir leer los datos obtenidos del Subsistema de Procesamiento Documental y poder utilizarlos como vectores característicos de entrada para representar los documentos del individuo creado.

Identificador de Requisito: RQ-F-019

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir al usuario poder parametrizar todos los parámetros del sistema a fin de evaluarlo fácilmente.

Identificador de Requisito: RQ-F-020

Subsistema:	Subsistema para Crear la Población Inicial (Generación 0)
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar el generador de números aleatorios al Subsistema a partir de la semilla establecida inicialmente, para utilizarlos durante el proceso de creación del subsistema.

Identificador de Requisito: RQ-F-021

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar el mejor individuo (elitismo) obtenido del Subsistema para mantenerlo en la siguiente generación.

Identificador de Requisito: RQ-F-022

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Ordenar por su fitness cada uno de los individuos creados en la población anterior para poder aplicar la selección del torneo.

Identificador de Requisito: RQ-F-023

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Seleccionar los Individuos de la población anterior, aplicando para ello el método de selección por torneo, utilizar inicialmente como tamaño de torneo el valor de 2.

Identificador de Requisito: RQ-F-024

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Crear la nueva población de individuos a partir de la selección por torneo utilizada y verificar que se cumplan la reglas de la gramática de individuos.

Identificador de Requisito: RQ-F-025

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar la tasa de probabilidad de mutación establecida para poder determinar si se va a aplicar el operador de mutación con aquel individuo repetido que ha sido seleccionado por el método del torneo.

Identificador de Requisito: RQ-F-026

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de mutación lo establece seleccionar de forma aleatoria dos nodos del cromosoma seleccionado previamente que contengan documentos diferentes con la finalidad de aplicar el operador de mutación a dichos nodos.

Identificador de Requisito: RQ-F-027

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de mutación lo establece luego de seleccionar dos nodos del cromosoma que tengan documentos diferentes aplicar la mutación transponiendo los nodos seleccionados.

Identificador de Requisito: RQ-F-028

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Verificar si el nuevo individuo mutado cumple las reglas de construcción establecidas, si es así mantenerlo en la nueva población, y generar los nuevos árboles correspondientes para incorporarlo a la nueva población en la generación actual.

Identificador de Requisito: RQ-F-029

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Aplicar la probabilidad de cruce establecida para poder establecer si se va a aplicar el operador de cruce con los individuos de la nueva generación que se está procesando actualmente.

Identificador de Requisito: RQ-F-030

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Fijar aleatoriamente un umbral de profundidad para poder limitar el número de cruces a realizar en la población, dicho umbral no debe ser superior al 30 % del tamaño de la población antes de aplicar el operador de cruce.

Identificador de Requisito: RQ-F-031

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece, generar un número aleatorio para poder determinar el número de veces que se aplicará los operadores de cruce sobre la población de individuos de la generación actual. Aplicar el operador de cruce Tipo 1 y el operador de cruce Tipo 2 por igual (cada uno al 50 %) dependiendo del número de veces obtenido al azar

Identificador de Requisito: RQ-F-032

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece, aplicar el Operador de Cruce Tipo 1 primero, Seleccionando dos individuos de la población actual de forma aleatoria para que realicen el papel de padres antes de aplicar el cruce. Este proceso repetirlo tantas veces como se haya fijado el umbral de profundidad para el número de cruces a realizar

Identificador de Requisito: RQ-F-033

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece seleccionar de forma aleatoria un individuo de los dos individuos elegidos previamente como padres para que realice el papel de cromosoma mascara antes de aplicar el cruce. Este proceso repetirlo tantas veces como se haya fijado el umbral de profundidad para el número de cruces a realizar

Identificador de Requisito: RQ-F-034

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece luego de seleccionar el cromosoma mascara de ambos padres aplicar el operador de cruce propuesto implementando el algoritmo de mascara y pivote en el cromosoma elegido como mascara. Este proceso repetirlo tantas veces como se haya fijado el umbral de profundidad a realizar

Identificador de Requisito: RQ-F-035

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Verificar si el nuevo individuo creado luego de aplicar el operador de cruce tipo 1 cumple las reglas de construcción establecidas, si es así mantenerlo en la nueva población, y generamos el nuevo árbol correspondiente para incorporarlo a la nueva población con dichas reglas. Este proceso repetirlo tantas veces como se haya fijado el umbral de profundidad para el número de cruces a realizar

Identificador de Requisito: RQ-F-036

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece, y luego de aplicar el Operador de Cruce Tipo 1, aplicar el operador de Cruce Tipo 2, Seleccionando dos individuos de la población actual de forma aleatoria para que realicen el papel de padres antes de aplicar el cruce. Este proceso repetirlo tantas veces como se haya fijado el umbral de profundidad para el número de cruces a realizar

Identificador de Requisito: RQ-F-037

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece, al aplicar el operador de cruce tipo 2, seleccionar de forma aleatoria dos nodos terminales del padre 1, y buscar dichos nodos en el padre 2 para realizar el intercambio de nodos entre los individuos

Identificador de Requisito: RQ-F-038

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Si la probabilidad de cruce lo establece luego de seleccionar los nodos terminales del padre 1 y luego de haber encontrado sus respectivos nodos en el Padre 2, realizar el cruce de ambos individuos, intercambiando los nodos seleccionados entre los dos individuos padres

Identificador de Requisito: RQ-F-039

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Verificar si los nuevos individuos creados luego de aplicar el operador de cruce tipo 2 cumple las reglas de construcción establecidas, si es así mantenerlo en la nueva población, y generamos el nuevo árbol correspondiente para incorporarlo a la nueva población con dichas reglas.

Identificador de Requisito: RQ-F-040

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Evaluar cada uno de los nuevos individuos generados, aplicando para ello la función de adaptación con los operadores definidos para los individuos. Este proceso se aplicará para todos los nodos función del cromosoma, siendo el nodo raíz el fitness global del individuo, para ello se deberá calcular recursivamente los centroides correspondiente en cada nodo función del cromosoma y evaluar nuevamente la función de distancia euclidea y la función de similitud del ángulo del coseno en cada nodo desde los nodos inferiores del cromosoma hasta el nodo raíz del cromosoma.

Identificador de Requisito: RQ-F-041

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios ver cada uno de los nuevos individuos creados y evaluados, así como también la representación del cromosoma correspondiente a dicho individuo.

Identificador de Requisito: RQ-F-042

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Comparar cada uno de los nuevos fitness globales de la actual población, a fin de mantener el elitismo del mejor individuo para las siguientes generaciones.

Identificador de Requisito: RQ-F-043

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Permitir a los usuarios ver cada uno de los pasos realizados en este módulo, y los operadores aplicados con este subsistema.

Identificador de Requisito: RQ-F-044

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Calcular el promedio de todos los fitness evaluados en la actual generación, para luego mostrarlo en los resultados

Identificador de Requisito: RQ-F-045

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar el fitness de aquel mejor individuo que haya sido encontrado, luego del procesamiento de todas las generaciones establecidas, o luego de haber alcanzado el mínimo global.

Identificador de Requisito: RQ-F-046

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Repetir el procesamiento de éste módulo hasta el número de generaciones establecido por el usuario o hasta que se encuentre un mínimo global entre todas las generaciones

Identificador de Requisito: RQ-F-047

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar en el fichero el fitness del mejor individuo que haya sido encontrado, luego del procesamiento de todas las generaciones.

Identificador de Requisito: RQ-F-048

Subsistema:	Subsistema para Procesar el Sistema Evolutivo
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar gráficamente aquel mejor individuo encontrado en la representación jerárquica propuesta (árboles binarios) que permita apreciar los documentos que han sido ordenados luego de aplicar el algoritmo del sistema.

Identificador de Requisito: RQ-F-049

Subsistema:	Subsistema de Resultados
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar todos los vectores de características de los documentos

Identificador de Requisito: RQ-F-050

Subsistema:	Subsistema de Resultados
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar los datos estadísticos referentes a los vectores documentales que se están procesando. Calcular la media y la mediana de toda la población.

Identificador de Requisito: RQ-F-051

Subsistema:	Subsistema de Resultados
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Mostrar los datos referentes a la distancia euclídea entre los documentos (Matriz de distancia euclidea) y la matriz de distancia del ángulo del coseno entre los términos procesados.

Identificador de Requisito: RQ-F-052

Subsistema:	Subsistema de Resultados
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Evaluar el algoritmo Kmeans con los vectores documentales y mostrar los grupos obtenidos, haciendo variar el valor de K .

Identificador de Requisito: RQ-F-053

Subsistema:	Subsistema de Resultados
Tipo de Requisito:	Funcional
Prioridad:	Alta
Descripción	Con los resultados obtenidos por el algoritmo Kmeans y con Nuestros resultados del sistema desarrollado, comparar los resultados, y ver los mejores grupos obtenidos con cada uno de los métodos.

Identificador de Requisito: RQ-F-054

Subsistema:	Todos
Tipo de Requisito:	Funcional
Prioridad:	Media
Descripción	Todas las pantallas de la aplicación tendrán el mismo diseño en función a la aplicación.

Identificador de Requisito: RQ-F-055

Subsistema:	Todos
Tipo de Requisito:	Funcional
Prioridad:	Media
Descripción	Todas las pantallas de la aplicación no serán muy pesadas en cuanto a tamaño, debido a la necesidad de rapidez del sistema.

Identificador de Requisito: RQ-F-056

Subsistema:	Todos
Tipo de Requisito:	Funcional
Prioridad:	Media
Descripción	Todas las opciones donde se tenga que seleccionar una opción, tendrán por defecto el introducido para entrar en el sistema

Identificador de Requisito: RQ-F-057

Subsistema:	Todos
Tipo de Requisito:	Funcional
Prioridad:	Media
D escripción	Desconectarse de la aplicación guardando los procesos realizados.

Tabla 4.5 Requisitos de Seguridad del Sistema Evolutivo

Identificador de Requisito: RQ-S-001

Subsistema:	Subsistema Menú Inicial
Tipo de Requisito:	Seguridad
Prioridad:	Baja
Descripción	Para poder acceder al sistema hay que introducir el nombre de usuario junto con una contraseña única almacenada en la base de datos del sistema.

Identificador de Requisito: RQ-S-002

Subsistema:	Todos
Tipo de Requisito:	Seguridad
Prioridad:	Alta
Descripción	El sistema no debe permitir que se introduzcan tipos de datos incorrectos en los distintos campos a rellenar. A saber, fechas incorrectas o con formato incorrecto,textos en campos numéricos, etc.

Tabla 4.6 Requisitos de Control del Sistema Evolutivo

Identificador de Requisito: RQ-C-001

Subsistema:	Todos
Tipo de Requisito:	Control y Portabilidad
Prioridad:	Alta
Descripción	Posibilidad de acceso al sistema, por parte del cliente, desde cualquier equipo, bajo cualquier plataforma de forma modular

Tabla 4.7. Descripción de los elementos del Sistema Evolutivo

En esta sección detallamos los actores involucrados en la aplicación, y la relación de cada uno de los casos de uso que han sido implementados en la presente tesis.

ACTORES

- Cualquier persona que se autentique en el sistema.

CASOS DE USO:

- **Gestión Documentos:** Permite realizar el procesamiento de los documentos del SRI con la metodología propuesta (Requisitos: RQ-F-001, RQ-F-002, RQ-F-003, RQ-F004, RQ-F005, RQ-F-006, RQ-F-007, RQ-F-008, RQ-F-009, RQ-F-010, RQ-F-011, RQ-F-054,RQ-F-055,RQ-F-056,RQ-F057)

- **Gestión de Individuos Iniciales:** Permite crear, evaluar, y modificar los individuos de la población inicial del sistema con la reglas de la gramática propuesta (Requisitos: RQ-F-012, RQ-F-013, RQ-F-014,RQ-F-015,RQ-F-016, RQ-F-017, RQ-F-018, RQ-F-019, RQ-F-020, RQ-F-0054, RQ-F-055, RQ-F-056, RQ-F057, RQ-S-001, RQ-S-002, RQ-C-001).

- **Gestión de Generaciones:** Permite visualizar, crear y modificar las diferentes generaciones de los individuos del sistema (Requisitos: RQ-F-021, RQ-F-022, RQ-F-023, RQ-F-024, RQ-F-025, RQ-F-026, RQ-F-027, RQ-F-028,RQ-F-029, RQ-F-030, RQ-F-031, RQ-F-032, RQ-F-033, RQ-F-034, RQ-F-035,RQ-F-036, RQ-F-037, RQ-F-038, RQ-F-039, RQ-F-040, RQ-F-041, RQ-F-042,RQ-F-043,RQ-F-054, RQ-F-055,RQ-F-056,RQ-F-057, RQ-S-001, RQ-S-002, RQ-C-001).

- **Gestión del Fitness:** Permite evaluar y gestionar el fitness de los individuos procesados en la aplicación (Requisitos: RQ-F-014, RQ-F-015, RQ-F-040, RQ-F-041, RQ-F-042, RQ-F-043, RQ-F-044, RQ-F-045, RQ-F-046, RQ-F-47, RQ-F048, RQ-F-054, RQ-F-055, RQ-F-056, RQ-F-057RQ-S-001, RQ-S-002, RQ-C-001)

- **Gestión de Resultados:** Permite gestionar los resultados con otro método de agrupamiento y comparar los resultados obtenidos por el sistema (Requisitos: RQ-F-049, RQ-F-050, RQ-F-051, RQ-F-052, RQ-F-053, RQ-F-054, RQ-F-055, RQ-F-056, RQ-F-057, RQ-S-001, RQ-S-002, RQ-C-001)

- **Desconexión:** permite la desconexión de la aplicación (RQ-F-057).

Anexo 2

RESULTADOS EXPERIMENTALES CON EL ALGORITMO GENÉTICO

En este apartado detallamos cada uno de los resultados experimentales que hemos obtenido con el algoritmo genético desarrollado en el presente trabajo. Presentamos cada una de las ejecuciones (test) con diferentes muestras de documentos, para cada una de las colecciones experimentadas, mostrando el mejor fitness obtenido, la generación que obtuvo el mejor fitness, la media del fitness medio en la ejecución del AG, el número de aciertos conseguido, el porcentaje de efectividad obtenido y su correspondiente promedio y desviación standard.

Tabla 5.18 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 2 tomando muestras de 20 documentos "Muy Pocos Documentos".

20 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,155455670	**0,155447570**	0,155447570	0,155470100	0,155452910	0,15545476	0,00000828
Generacion	1029	886	1151	1314	1050	1086	
Media Fitn medio	0,384950635	0,413259773	0,402602857	0,476514014	0,380609269		
Media Meseta	0,384757382	0,412310319	0,237581411	0,478276254	0,381715506		
Aciertos	18	18	18	18	18	18	
Efectividad %		90					

Tabla 5.19 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 2 tomando muestras de 50 documentos "Pocos Documentos".

50 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,156223280**	0,156284630	0,156226930	0,156252010	0,156227130	0,15624280	0,00002329
Generacion	3051	3390	2646	1288	2831	2641	
Media Fitn medio	0,298800179	0,353115820	0,237391335	0,249073108	0,266659410		
Media Meseta	0,299490939	0,363968012	0,236682567	0,247952728	0,263645633		
Aciertos	47	47	47	47	47	47	
Efectividad %	94						

Tabla 5.20 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 2 tomando muestras de 80 documentos "Muchos Documentos".

80 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,159009400**	0,159159790	0,159269340	0,159040500	0,159580010	0,15921181	0,00020587
Generacion	2500	3371	1233	2205	1920	2246	
Media Fitn medio	0,212428297	0,206307930	0,205906732	0,212037918	0,216960379		
Media Meseta	0,214818043	0,208615355	0,205768630	0,212349670	0,217050058		
Aciertos	71	71	71	71	71	71	
Efectividad %	89						

Tabla 5.21 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 2 tomando muestras de 150 documentos "Bastantes Documentos".

150 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,165013920**	0,165095550	0,165063620	0,165029710	0,165223150	0,16508519	0,00007452
Generacion	2342	2444	3446	2342	1824	2480	
Media Fitn medio	0,174864586	0,172703582	0,172874672	0,171668705	0,172647015		
Media Meseta	0,174954693	0,172766797	0,173335973	0,171652617	0,172586170		
Aciertos	115	115	115	115	115	115	
Efectividad %	77						

Tabla 5.22 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 2 tomando muestras de 246 documentos "Más Documentos".

246 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,174152830	0,174168700	**0,174112100**	0,174115790	0,174975670	0,17430502	0,00033602
Generacion	3386	816	2203	2805	1087	2059	
Media Fitn medio	0,176809912	0,175422141	0,179390566	0,175040223	0,174543789		
Media Meseta	0,178034899	0,175417242	0,179374445	0,175226405	0,174545080		
Aciertos	170	170	170	170	170	170	
Efectividad %			69				

Tabla 5.23 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 8 tomando muestras de 20 documentos "Muy Pocos Documentos".

20 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,151163560**	0,151163560	0,151163560	0,151163560	0,151163560	0,15116356	0,00000000
Generacion	555	674	618	915	634	679	
Media Fitn medio	0,160309372	0,160100739	0,159523223	0,160160068	0,159491626		
Media Meseta	0,160321046	0,160086505	0,159529258	0,160149238	0,159484242		
Aciertos	17	17	17	17	17	17	
Efectividad %	85						

Tabla 5.24 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 8 tomando muestras de 50 documentos "Pocos Documentos".

50 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,154856500**	0,154856500	0,154856500	0,154856500	0,154856500	0,15485650	0,00000000
Generacion	1615	1281	1574	1030	1168	1334	
Media Fitn medio	4,942009066	7,457933669	5,006444037	2,586181198	4,523546303		
Media Meseta	5,021880631	7,460688212	5,009128807	2,579231952	4,535479908		
Aciertos	48	48	48	48	48	48	
Efectividad %	96						

Tabla 5.25 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 8 tomando muestras de 80 documentos "Muchos Documentos".

80 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,157073880**	0,157082520	0,157093920	0,157093920	0,157073880	0,15708362	0,00000898
Generacion	746	1469	617	1686	2283	1360	
Media Fitn medio	0,559647656	0,493627847	0,460732301	0,512509851	2,383302473		
Media Meseta	0,558626061	0,487590109	0,460049907	0,516158886	2,398939935		
Aciertos	68	68	68	68	68	68	
Efectividad %	85						

Tabla 5.26 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 8 tomando muestras de 150 documentos "Bastantes Documentos".

150 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,162779010	0,162783320	**0,162035070**	0,162482850	0,162052970	0,16242664	0,00033091
Generacion	2634	2191	1989	2113	2489	2283	
Media Fitn medio	0,295702699	0,329326981	0,316686729	1,104882566	0,340786481		
Media Meseta	0,297215651	0,335067929	0,318031689	1,124243135	0,341447334		
Aciertos	104	104	104	104	104	104	
Efectividad %			69,3				

Tabla 5.27 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 8 tomando muestras de 188 documentos "Más Documentos".

188 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,163014600**	0,163412700	0,163803670	0,163123310	0,163355610	0,16334198	0,00027325
Generacion	2293	810	969	1328	3466	1773	
Media Fitn medio	0,304793933	0,350013653	0,305876405	0,312690000	0,338135395		
Media Meseta	0,303302353	0,349789773	0,305657812	0,311738118	0,369373886		
Aciertos	129	129	129	129	129	129	
Efectividad %	68,6						

Tabla 5.28 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 20 tomando muestras de 20 documentos "Muy Pocos Documentos".

20 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,153027060**	0,153086090	0,153149000	0,153295350	0,153541520	0,15321980	0,00018398
Generacion	1092	969	1073	1495	913	1108	
Media Fitn medio	0,242924859	0,258394993	0,222100675	0,296460970	0,257312553		
Media Meseta	0,243030243	0,258272633	0,221977593	0,297547730	0,256195976		
Aciertos	17	17	17	17	17	17	
Efectividad %	85						

Tabla 5.29 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 20 tomando muestras de 50 documentos "Pocos Documentos".

50 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,156919530	**0,156198620**	0,156973370	0,156792620	0,156422730	0,15666137	0,00030077
Generacion	2843	2173	3250	1866	3043	2635	
Media Fitn medio	0,248124160	0,282013626	0,270165977	0,277368454	0,252717127		
Media Meseta	0,248533644	0,283086562	0,267749904	0,278474788	0,253664094		
Aciertos	46	46	46	46	46	46	
Efectividad %	92						

Tabla 5.30 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 20 tomando muestras de 80 documentos "Muchos Documentos".

80 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,158105430	0,158115850	**0,158069980**	0,158102170	0,158125700	0,15810383	0,00001884
Generacion	1638	1867	2196	1824	1169	1739	
Media Fitn medio	0,192624337	0,207210299	0,198336720	0,210075523	0,191639942		
Media Meseta	0,192502569	0,207405665	0,198537974	0,209984963	0,191736077		
Aciertos	65	65	65	65	65	65	
Efectividad %			81,2				

Tabla 5.31 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 20 tomando muestras de 108 documentos "Bastantes Documentos".

108 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,159096850	0,159387760	0,159751230	**0,159031080**	0,159114570	0,15927630	0,00026701
Generacion	3432	2408	3494	1437	2410	2636	
Media Fitn medio	0,179448109	0,179623528	0,180856255	0,177984640	0,179050761		
Media Meseta	0,179738433	0,179344736	0,181128479	0,178117232	0,179040671		
Aciertos	75	75	75	75	75	75	
Efectividad %				69,4			

Tabla 5.32 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 21 tomando muestras de 20 documentos "Muy Pocos Documentos".

20 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,152054030	0,152054030	**0,152048900**	0,152092460	0,152054030	0,15206069	0,00001601
Generacion	966	1079	1163	836	1779	1165	
Media Fitn medio	0,604438247	1,324045760	0,613283977	0,311062364	0,571620607		
Media Meseta	0,606032184	1,331197226	0,613302981	0,313560461	0,574503228		
Aciertos	18	18	18	18	18	18	
Efectividad %			90				

Tabla 5.33 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 21 tomando muestras de 50 documentos "Pocos Documentos".

50 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,153023840	**0,153006650**	0,153030220	0,153136490	0,153047130	0,15304887	0,00004569
Generacion	3300	2079	3405	2924	1970	2736	
Media Fitn medio	0,192745874	0,197550227	0,195501116	0,215029330	0,255272464		
Media Meseta	0,192779710	0,197236007	0,195283332	0,213813054	0,256593762		
Aciertos	46	46	46	46	46	46	
Efectividad %		92					

Tabla 5.34 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 21 tomando muestras de 80 documentos "Muchos Documentos".

80 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,156666430	0,156512360	**0,156029510**	0,156129610	0,156546720	0,15637693	0,00025014
Generacion	2584	3286	2787	2082	3313	2810	
Media Fitn medio	0,201118306	0,193294485	0,226248087	0,193329380	0,198338202		
Media Meseta	0,201518850	0,194255816	0,225847524	0,193232779	0,197245454		
Aciertos	65	65	65	65	65	65	
Efectividad %			81				

Tabla 5.35 Resultados de los experimentos con el AG procesando la Colección Reuters 21578 - Distribución 21 tomando muestras de 132 documentos "Bastantes Documentos".

132 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,157158990	0,157014280	**0,157012180**	0,157184050	0,157668790	0,15720766	0,00024132
Generacion	1097	1381	3359	1595	2470	1980	
Media Fitn medio	0,177403176	0,179261940	0,180807121	0,178232537	0,177876818		
Media Meseta	0,177415779	0,179234114	0,180927336	0,178249306	0,178028819		
Aciertos	93	93	93	93	93	93	
Efectividad %			70,4				

Tabla 5.36 Resultados de los experimentos con el AG procesando la Colección de Editoriales del Diario "El Mundo - 2006-2007" tomando muestras de 20 documentos "Muy Pocos Documentos".

20 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,152578890	**0,152444850**	0,152752230	0,152745870	0,152770700	0,15265851	0,00012727
Generacion	3108	1026	2527	1463	2854	2196	
Media Fitn medio	0,236614796	0,247702996	0,204827944	0,305218411	0,223631271		
Media Meseta	0,236100169	0,247502704	0,203935277	0,306170001	0,221293563		
Aciertos	17	17	17	17	17	17	
Efectividad %		85					

Tabla 5.37 Resultados de los experimentos con el AG procesando la Colección de Editoriales del Diario "El Mundo - 2006-2007" tomando muestras de 50 documentos "Pocos Documentos".

50 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,154404620	0,154435720	0,154415110	**0,154187850**	0,154418360	0,15437233	0,00009278
Generacion	3029	2715	1849	2839	2957	2678	
Media Fitn medio	0,211850546	0,746881454	0,240409189	0,285422699	0,222410119		
Media Meseta	0,214093069	0,757644881	0,241090606	0,247126995	0,222842278		
Aciertos	48	48	48	48	48	48	
Efectividad %				96			

Tabla 5.38 Resultados de los experimentos con el AG procesando la Colección de Editoriales del Diario "El Mundo - 2006-2007" tomando muestras de 80 documentos "Muchos Documentos".

80 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	0,155710750	**0,155099730**	0,155277570	0,155121080	0,155656550	0,15537314	0,00026143
Generacion	3342	3137	2332	3425	3191	3085	
Media Fitn medio	0,199244986	0,205573474	0,201460970	0,195238601	0,202771725		
Media Meseta	0,199003388	0,204941967	0,203836031	0,193445123	0,204189879		
Aciertos	67	67	67	67	67	67	
Efectividad %		83,7					

Tabla 5.39 Resultados de los experimentos con el AG procesando la Colección de Editoriales del Diario "El Mundo - 2006-2007" tomando muestras de 150 documentos "Bastantes Documentos".

150 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,162184980**	0,162848880	0,162288290	0,162359540	0,162799890	0,16249632	0,00027400
Generacion	1244	1136	2090	1317	1949	1547	
Media Fitn medio	0,169958596	0,168619701	0,169612031	0,168309293	0,169380906		
Media Meseta	0,169971355	0,168601229	0,169552668	0,168265747	0,169379697		
Aciertos	118	118	118	118	118	118	
Efectividad %	78,6						

Tabla 5.40 Resultados de los experimentos con el AG procesando la Colección de Editoriales del Diario "El Mundo - 2006-2007" tomando muestras de 238 documentos "Más Documentos".

238 documentos	Test 1	Test 2	Test 3	Test 4	Test 5	Promedio	Desviación
Mejor Fitness	**0,172107860**	0,173012840	0,172542520	0,172594240	0,172626710	0,17257683	0,00028775
Generacion	3387	1635	1675	1963	1443	2021	
Media Fitn medio	0,176995345	0,173774925	0,175844937	0,171964579	0,174571195		
Media Meseta	0,177800944	0,173784496	0,175815206	0,171998665	0,174563243		
Aciertos	167	167	167	167	167	167	
Efectividad %	70,1						

APÉNDICE:

CLASES Y MÓDULOS PRINCIPALES DE LA APLICACIÓN

CLASES

```
class Adn
{
    //clase usada para generar los cromosomas
    private:
       int * datos;
       int ponf(int pos);              //pone un nodo funcion f y del que cuelgan 2 nodos
                                       //documentos en un array con estructura de arbol
       int maxf;

       void getP(unsigned char * buff,int n,int);      //funcion recursiva usada para
                                                       //  generar el orden que queremos

       void addChar(unsigned char *s,char c);     //agrega un caracter a una cadena

    public:
         Adn(int,int);         //funcion que genera un arbol ALEATORIO dentro de un
                               //  array de altura maxima altura
                               //hay un maximo de documentos dados por la altura maxima
         ~Adn();
         void getPreorden(unsigned char *buff);

};



class Arbol
{
    // Clase del Arbol documental
    private:

         double ucorrelation (int n, float *data1, float *data2);
         double distEuclidean (int length, float *v1, float *v2);
         void calculaCentroide(float *buff,float *d1,float *d2);

      public:
         void calcularFitness(float alfa);
         Arbol();
         Arbol(int valor,float vectordocum[],int nodo,int aux,int rr);
         ~Arbol();
         bool InsertarPreorden(int valor,float vectordocum[],int nodo,int aux,int rr);
         void Arbol::Ver(int n,int col, int lin);
         void ActualizaArbol(int aux);
         void Arbol::BuscaVer(int vv,int clase);

         //Datos

         int Dato;                          // documento del arbol
         bool hecho;// marca;               // Marca de ayuda para saber si se ha
                                            // procesado el gen (nodo o terminal)

         double distancia;                  // distancia calculada
         double similitud;                  // similitud calculada
         double fitness;                    // fitness del individuo
         int tiempovida;                    // Tiempo de vida del cromosoma
         int edad;                          // Edad del cromosoma
         int vive;                          // Flag de vive o no vive
         unsigned char genoma[TGENES];      // Declaracion de la estructura de la
                                            // poblacion
         int bien;                          // Bien agrupado
         int bien1;                         // Grupo 1
         int bien2;                         // Grupo 2

         int rama;                          // Indica a que Rama pertenece el Documento
         int numnodo;                       // Numero de Nodo en Preorden
         int nodopadre;                     // Indica el Numero de Nodo del Padre (en
                                            // preorden)
```

```
        double centroide[TERMINOS_MAX];         // centroide correspondiente de los
                                                //  terminos
        Arbol *Izq;                     // Hijo Izquierdo y Hijo Derecho
        Arbol *Der;
        float vectordocum[TERMINOS_MAX];        // Vector documental de cada
                                                // documento

};
```

*** GENERADOR DE NUMEROS ALEATORIOS */**

```
#ifndef __RANDOM_PPIO_H
#define __RANDOM_PPIO_H

/* Inicializa la semilla al valor 'x'.
  Solo debe llamarse a esta funcion una vez en todo el programa */
void Set_random (unsigned long x);

/* Devuelve el valor actual de la semilla */
unsigned long Get_random (void);

/* Genera un numero aleatorio real en el intervalo [0,1[
  (incluyendo el 0 pero sin incluir el 1) */
float Rand(void);

/* Genera un numero aleatorio entero en {low,...,high} */
int Randint(int low, int high);

/* Genera un numero aleatorio real en el intervalo [low,...,high[
  (incluyendo 'low' pero sin incluir 'high') */
float Randfloat(float low, float high);

#endif
```

// DEFINICIONES GENERALES DE LOS PARAMETROS A EXPERIMENTAR
// PARA EL SISTEMA GENETICO

```
#define PROB_MUTACION  0.03          // PROBABILIDAD DE MUTACION
                                     //   El Rango a experimentar es de:
                                     //  0.01,0.03..0.05..0,07..0.09...0.10,0.30, 0.50,...0,70

#define PROB_CRUCE        0.80       // PROBABILIDAD DE CRUCE
                                     //   El Rango a experimentar es de: 0.7,0.8,0.9

#define N_GENERACIONES  4000         // MAXIMO NUMERO DE GENERACIONES

#define TAM_TORNEO     2             // TAMAÑO DEL TORNEO

#define ALFA                0.85

#define MAXDOCUM                     // MAXIMO NUMERO DE DOCUMENTOS A PROCESAR


#define TGENES      305              // NUMERO DE GENES DEL CROMOSOMA

#define TERMINOS_MAX                 //  NUMERO DE TERMINOS  LEXEMAS
#define LONG_POBLACION    50         // TAMAÑO DE POBLACION
#define ALTURA_MAX        9          //  PROFUNDIDAD DEL ARBOL
                                     // MAXIMO DE DOCUMENTOS PARA ESA
                                     // ALTURA  sera 2 elevado a la N los documentos
#define COLECCION         21         // COLECCION DE DATOS A PROCESAR
#define GRUPOA            1          // GRUPO A
#define GRUPOB            7          // GRUPO B
```

MÓDULOS FUENTES PRINCPALES

ADN.CPP

```
#include "config.h"
#include "adn.h"
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <malloc.h>

int Adn::ponf(int pos)
{
   //pone un nodo funcion f y del que cuelgan 2 nodos documentos en un array con
   //estructura de arbol
   if (pos<=maxf) {
      datos[pos]='f';
      datos[pos*2]='d';
      datos[pos*2+1]='d';
      return 1;
   }
   return 0;
}

Adn::Adn( int docs,int altura)
 {
        //funcion que genera un arbol ALEATORIO dentro de un array de altura maxima
        //hay un maximo de documentos dados por la altura maxima

        int i,j,r;
        int n=(int)pow(2,altura-1)+1;

        if (docs*2>n) docs=((n-2)/2)+1;
       // if (docs>n) docs=((n-2)/2)+1;

        datos=new int[n];
        maxf =(int)pow(2,altura-2);

        memset(datos,0,sizeof(int)*n);

        ponf(1);
        int pdocs=2;

        while (docs>pdocs)                  //mientras queden documentos
        {
            r = rand()%(pdocs);             //de los documentos que hay puesto escogemos
                                            //uno aleatorio
            for (j=1,i=0;j<n;j++)
            {
               if (datos[j]=='d')
               {
                 if (i==r)
                 { //si estamos en el documento elegido, lo cambiamos por f
                        pdocs+=ponf(j);
                        break;
                 }
                 i++;
               }
            }

        }

        //for (int k=0;k<n;k++) printf ("%c",datos[k]);
}


Adn::~Adn() {
```

```
}
void Adn::addChar(unsigned char *s,char c)
{   //agrega un caracter a una cadena
    int l=strlen((const char *)s);
    s[l]=c;
    s[l+1]=0;

}
void Adn::getP(unsigned char *buff,int b,int d)
{
    //funcion recursiva usada para generar el orden que queremos
    addChar(buff,datos[d]);
    //  printf("%s\n",buff);

    if (datos[d] == 'd') return;
    getP(buff,b,d*2);
    getP(buff,b,d*2+1);

}
void Adn::getPreorden(unsigned char *buff)
{
    memset(buff,0,TGENES);
    getP(buff,0,1);
}
```

ARBOL.CPP

// Métodos del Arbol documental

```
#include "config.h"
#include "arbol.h"
#include <windows.h>
#include <conio.h>
#include <string.h>
#include <math.h>

double Arbol::ucorrelation (int n,float *data1, float *data2)
{
  // Rutina para calcular el coeficiente de correlacion
   double result = 0.;
 double denom1 = 0.;
 double denom2 = 0.;
 double aux;

   int ii;
    for (ii = 0; ii < n; ii++)
    {
      double term1 = data1[ii];
      double term2 = data2[ii];
        result += term1*term2;
        denom1 += term1*term1;
        denom2 += term2*term2;

      }
  if (denom1==0.) return 1.;
  if (denom2==0.) return 1.;

  //printf("\nresult %f, denom %f",result,sqrt(denom1*denom2));
  result = result / (sqrt(denom1*denom2));
  //printf("\nCorrelacion %f",result);

 return result;
}
/* ********************************************************************** */
```

```
double Arbol::distEuclidean (int length, float *v1, float *v2)
{
 // Rutina para calcular la distancia euclidea
 double dist = 0.0;
 double f;
 int ii;

  for (ii = 0; ii < length; ++ii)
    {
    //  printf("\nvalor 1: %f",v1[ii]);
    //  printf("\nvalor 2: %f",v2[ii]);
      f = v1[ii] - v2[ii];
    //  printf("\n f es: %f",f);
      dist += f * f;
    //  printf("\ndistancia1: %f,distanciasqr %f ",dist,sqrt(dist));
    }

  return sqrt (fabs(dist));
}

Arbol::Arbol() {
            Izq=NULL;
            Der=NULL;
            Dato=0;
            hecho=false;
}

Arbol::Arbol(int valor,float vector[],int nodo, int aux,int rr) {
            Izq=NULL;
            Der=NULL;
            Dato=valor;
            numnodo=nodo;
            nodopadre=aux;
            rama=rr;
            hecho=Dato!=0;
            memcpy(&vectordocum,vector,sizeof(vectordocum));
}
Arbol::~Arbol() {
            delete Izq;
            delete Der;
}

void Arbol::ActualizaArbol(int aux)
{
 // Actualiza la rama del arbol - rama=1 si es rama de la izquierda - rama = 2 si es rama
 // de la derecha
 // La actualizacion se realiza a partir del primer elemento en preorden ya sea de la Izq o
// de la Der.
 // Pasandole el parametro "aux" que indica la rama obtenida

 if (Dato) {   // Caso si es un documento
          if ((nodopadre==0) && (rama==1)) aux=1; // Nodopadre indica que estamos al
                                                  // principio de la rama Izq o Der
          if ((nodopadre==0) && (rama==2)) aux=2;
          rama=aux;
                          //  printf("D%d %d-%d->%d",Dato,numnodo,nodopadre,rama);
          }
 else  // Caso si es una Funcion
    {  if ((nodopadre==0) && (rama==0)) aux=0;   // Extrae para el primero solamente
       if ((nodopadre==0) && (rama==1)) aux=1;   // Nodopadre indica que estamos al
                                                 // principio de la rama Izq o Der
       if ((nodopadre==0) && (rama==2)) aux=2;
       rama=aux;
    //   printf("F %d-%d->%d",numnodo,nodopadre,rama);
    }

 if (Izq)
```

```
  {
  Izq->ActualizaArbol(aux);       // Llamadas recursivas para actualizar los nodos del
                                  // arbol por la Izq
  }

  if (Der)
  {
  Der->ActualizaArbol(aux);     // Llamadas recursivas para actualizar los nodos del
                                // arbol por la Der
  }
}

bool Arbol::InsertarPreorden(int valor,float vectordocum[],int nodo,int aux,int rr)
{
   int pp;
   // Inserta los nodos en el arbol en preorden
   if (Dato!=0)
             {
              return false; //es un nodo terminal (hoja)
             }
   if (Izq==0) { aux=nodo-1; if (aux==0) rr=1;
             Izq=new Arbol(valor,vectordocum,nodo,aux,rr);
               pp=aux;
             return true;
   }
   bool bOk;

   bOk = Izq->InsertarPreorden(valor,vectordocum,nodo,aux,rr);
   if (bOk) return true; //si ya lo inserto no seguimos

   if (Der==0) { aux=pp; if (aux==0) rr=2;
             Der=new Arbol(valor,vectordocum,nodo,aux,rr);
             return true;
   }

   bOk = Der->InsertarPreorden(valor,vectordocum,nodo,aux,rr);

   return bOk;

}

void Arbol::Ver(int n,int col, int lin)
{
 // Rutina recursiva para Imprimir el Arbol

   gotoxy(col,lin);

   textcolor(10);
   if (Dato)  printf("D%d",Dato);
   else printf("F");

   textcolor(LIGHTGRAY);

  if (Izq) {
          gotoxy(col-32/(2*n),lin+1);
          printf("%c",218);
          for (int j=0;j<(32/(2*n))-1;j++) printf("%c",196);
          printf("%c",193);
          Izq->Ver(n+1,col-32/(2*n),lin+2);
   }
   if (Der) {
          gotoxy(col,lin+1);
          printf("%c",193);
          for (int j=0;j<(32/(2*n))-1;j++) printf("%c",196);
          printf("%c",191);
          Der->Ver(n+1,col+32/(2*n),lin+2);
   }
```

```
  gotoxy(1,25);     // Pone el cursor al final de la pantalla
}

void Arbol::calculaCentroide(float *buff,float *d1,float *d2) {
 //calcula el centroide

  for (int j=0;j<TERMINOS_MAX;j++)
     {  buff[j]=(d1[j]+d2[j])/2;
       // buff[j]*=fitness;
     }

}

void Arbol::calcularFitness(float alfa) {
  //calcula el fitness

     float beta;

     beta=1-alfa;

     if (Dato) return;
     if (!(Izq->hecho && Der->hecho))
     {
         if (Izq) Izq->calcularFitness(alfa);
         if (Der) Der->calcularFitness(alfa);
     }
     distancia=distEuclidean(TERMINOS_MAX,Izq->vectordocum,Der->vectordocum);
     similitud=ucorrelation(TERMINOS_MAX,Izq->vectordocum,Der->vectordocum);
     fitness=alfa*(double)distancia+beta*(1/(double)similitud);
     calculaCentroide(vectordocum,Izq->vectordocum,Der->vectordocum);
     hecho=true;
}
```

RANDOM.CPP

```
/*  Rutina para generar semilla y numeros aleatorios
#include <stdio.h>
#include <stdlib.h>

#include <math.h>
#include <windows.h>
#include "random.h"
#include "conio.h"

unsigned long Seed = 0L;

#define MASK 2147483647
#define PRIME 65539
#define SCALE 0.4656612875e-9

void Set_random (unsigned long x)
// Inicializa la semilla al valor x.   Solo debe llamarse a esta funcion una vez en todo el
                                                                    //  programa
{
   Seed = (unsigned long) x;
}

unsigned long Get_random (void)
// Devuelve el valor actual de la semilla
{
   return Seed;
}


float Rand(void)
```

```
// Genera un numero aleatorio real en el intervalo [0,1]
//  (incluyendo el 0 pero sin incluir el 1)
{
   return (( Seed = ( (Seed * PRIME) & MASK) ) * SCALE );
}

int Randint(int low, int high)
// Genera un numero aleatorio entero en {low,...,high}
{
   return (int) (low + (high-(low)+1) * Rand());
}

float Randfloat(float low, float high)
// Genera un numero aleatorio real en el intervalo [low,...,high[
// (incluyendo 'low' pero sin incluir 'high')
{
   return (low + (high-(low))*Rand());
}
```

// GENETICO.CPP

```
#include <string.h>
#include <stdio.h>
#include <malloc.h>
#include <conio.h>
#include <math.h>
#include <limits>
#include <time.h>
#include <windows.h>

#include <vector>
#include <iostream>

#include "adn.h"
#include "config.h"
#include "arbol.h"
#include "random.h"
using namespace std;
int tdoc;                              // Numero total de documentos
int documentos;                        // documentos a procesar por el algoritmo
int termlexemas;                       // lexemas a procesar

struct Grupos
{
  int vv[MAXDOCUM];                    // Vector de todos los documentos
  int clase[MAXDOCUM];                 // Grupo de clase a la que pertenece
  int arbdocum[MAXDOCUM];              // Vector de Numero de documento
 }Grup;

int CalculoFicheros(int colum,int tipo);        // declaracion para Calcular el numero de documentos
int terminosdiferentes(char terminosdato[16]);  // declaracion para Calcular el numero de terminos lematizados a procesar
int cadenaAentero (char *cadena);


FILE *fsalida;                         // Fichero para la salida de datos con los mejores resultados de cada çcion
double pf1;                            // pf1 para calcular la media del fitness en cada generacion
int numero;                            // Apuntador de individuos generados
int seed2=1234567890;                  // Semilla fija para Randint  // ojo se cambia luego semilla con fecha del sistema para los
documentos
int seed;
float datax[MAXDOCUM][3000];           // Declaracion de la matriz de datos que recibirá los terminos de cada documento

Arbol * menor;                         // El menor de cada generacion

int *Torneo;

int indices[LONG_POBLACION];           // Reserva de memoria para el vector en el que se almacena los indices de los
individuos de la poblacion antigua seleccionada

Arbol Mbosquefitness[1];               // El Mejor arbol en cada generaciones  (temporal, se va reemplazando)
```

```
Arbol MMbosquefitness[1];                    // El Mejor arbol en todas las generaciones

Arbol bosque [LONG_POBLACION];               // Estructura que contiene todos los arboles generados en la poblacion inicial

unsigned char Newpopulation[LONG_POBLACION][TGENES];     //Declaracion Auxiliar de la Nueva estructura de la
poblacion-para usarla en la nueva generacion
double Newbosquefitness[LONG_POBLACION];     // Auxiliar de Fitness para los cruces
double Newbosquedistancia[LONG_POBLACION];
double Newbosquesimilitud[LONG_POBLACION];
unsigned char Newaux[TGENES];                // Auxiliar para el hijo creado luego del cruce
//
unsigned char population [LONG_POBLACION][TGENES];       //Declaracion de la estructura de la poblacion

double bosquefitness[LONG_POBLACION];        // Los Fitness en cada generacion (auxiliar) ?
double bosquedistancia[LONG_POBLACION];
double bosquesimilitud[LONG_POBLACION];

float alfa=ALFA;
int buenos=0,buenp,sumabuenp;

int jm1,jm2,jm3,jm4,jc1,jc2,jc3,jc4;     // Contadores para mutacion y cruce

void Arbol::BuscaVer(int vv,int clase)
{
 // Rutina recursiva para Buscar en el Arbol
  if (Dato==vv)       // Si lo encuentra
     {
      printf("Dato D%d Nodo %d Nodo padre %d Rama %d",Dato,numnodo,nodopadre,rama);
      if (rama==clase)
      {
       buenp=1;
       printf("\n Bien Agrupado %d",buenp);
       }
       else
       {
        buenp=0;
        printf("\n Mal Agrupado %d",buenp);
        }
       // return buenp;
     }

  if (Izq) {
          Izq->BuscaVer(vv,clase);
          }

   if (Der) {
          Der->BuscaVer(vv,clase);
          }

}

void retarda(int n)
{
  for (int i=0;i<=n;i++)
  {
  }
}

int encuentra(int t,struct Grupos Grup)
 /* busqueda secuencial para verificar si encuentra el documento generado */
 /* retorna -1 si es que no lo encuentra y un numero si lo encuentra (incluido 0) */
 {
   for(int a=0;a<MAXDOCUM;a++)
   {
       if(Grup.vv[a]==t)
        {
          return (a);
          break;
        }
    }
       return (-1);
 }


int SacaDocum(int tipo,int valor)
```

```
{
 // Leemos del fichero VectoresR.txt /VectoresT.txt de los datos ... toda la data entera
 // Sacamos el Numero documento(numdocum) correspondiente al documento que queremos analizar (valor)
 // numdocum corresponde al numero que tenemos representado en el arbol

  FILE *fichero;
  char salto;
  int wdocx;          //  Documento a analizar convertido a entero

   // Rellenando la matriz desde el Fichero de Preprocesamiento
    if (tipo==0)
    {   // Fichero de Reuters de datos
        if ((fichero = fopen("VectoresNR.LXM","r")) == NULL)
        {
        printf("ERROR al abrir el archivo VectoresR.LXM");    //Abrimos el fichero para escritura
        exit(1);
        }
    }
    else
    {   // Fichero de Texto de Datos
        if ((fichero = fopen("VectoresNT.LXM","r")) == NULL)
        {
        printf("ERROR al abrir el archivo VectoresT.LXM");   //Abrimos el fichero para escritura
        exit(1);
       }
    }

   int numdocum;        // Numero correlativo en el arbol
   char nombdocum[10];  // se usa para leer la primera columna de documento (numero de documento)
   salto=' ';
   for (int i=0; i<documentos; i++)
   {
     fscanf(fichero,"%3d %s",&numdocum,&nombdocum);   // Leemos la primera columna, el numero de documento
     char s2[5] = "R.\n\t";
     char *taux;
     // extraemos el termino
     taux = strtok(nombdocum, s2 );    // Primera llamada => Primer token
       wdocx=cadenaAentero(taux);                // Convertimos a entero los documentos que contienen el lexema

      if (wdocx==valor)
        {
         return numdocum;
        }
       for (int j=0; j<termlexemas; j++)
       {
        fscanf(fichero,"%f",&datax[i][j]);
        fscanf(fichero,"%c",&salto);
        }
      fscanf(fichero,"\n");
    }

    fclose(fichero);
}

void SeleccionaDatos(int tipo,int tdoc,int termlexemas)
{
  // Seleccionamos los documentos de la distribucion segun grupos

  vector <int> v1;       // Vector de cluster 1
  vector <int> v2;       // Vector de cluster 2

  int contav1=0,contav2=0,x1,x2,x3;          // Contadores y auxiliares para leer datos del fichero

  FILE *sel;

  if ((sel = fopen("docfintotal.TXT","r")) == NULL)
     {
      printf("ERROR al abrir el archivo de referencia..");    //Abrimos el fichero para escritura
      system("pause");
      exit(1);
      }

  for (int i=0; i < tdoc; i++)                    // Leemos el total de documentos (tdoc) de la distribucion para sacar grupos
  {
     fscanf(sel, "%3d\t%3d\t%3d",&x1,&x2,&x3);       // leemos el numero de documento y el numero de grupo y  secuencia
     fscanf(sel,"\n");
```

```
  if ((x2==GRUPOA) && (contav1 <MAXDOCUM/2))
     {
      contav1++;
      v1.push_back(x1);          // Asigna el numero de documento al vector v1 del grupo A
     }
   if ((x2==GRUPOB) && (contav2 <MAXDOCUM/2))
     {
      contav2++;
      v2.push_back(x1);          // Asigna el numero de documento al vector v2 del grupo B
     }

  }  // fin del for i de leer todo los documentos

  fclose(sel);

  printf("\n\n Hay %d del Grupo A y %d del GRUPO B",contav1,contav2);    //Error si hay pocos documentos en el grupo A

    if (contav1 <MAXDOCUM/2)
    {
    printf("\n\nPocos documentos del GRUPO A ...");    //Error si hay pocos documentos en el grupo A
    system("pause");
    exit(1);
    }
    if (contav2 <MAXDOCUM/2)
    {
    printf("\n\nPocos documentos del GRUPO B ...");    //Error si hay pocos documentos en el grupo B
    system("pause");
    exit(1);
    }

    int contav=contav1+contav2;

    if ((contav < MAXDOCUM))
      { printf("\n\nNo hay tantos documentos para realizar el agrupamiento.. Hay %d del Grupo A y %d del GRUPO
B",contav1,contav2);    //Error si hay pocos documentos en el grupo A
       system("pause");
       exit(1);
      }

   int xmax,w;              // Auxiliares para meter los grupos

    xmax=MAXDOCUM/2;

   for (int i=0; i<xmax; i++)              // En el vector V asignamos los xmax DOCUM de v1
    { Grup.vv[i]=v1[i];                    //     Asignamos en VV1 el primer grupo
     Grup.clase[i]=1;                     // Clase 1
    }

   if (MAXDOCUM<=80)                        // w se actualiza con la MAXDOCUM/2 si son menos de 80 docum
   w=MAXDOCUM/2;
   else  w=contav1;                        // de lo contrario w se actualiza a partir de contav1

   for (int i=w; i < MAXDOCUM; i++)          // Asignamos en V los w siguientes documentos de v2
    { Grup.vv[i]=v2[i-w];                  // Asignamos en VV2 el segundo grupo
     Grup.clase[i]=2;                     // Clase 2
    }

 FILE *fichero,*fsalida,*fzipt,*fzipn;
 char salto;

  if (tipo==0)
   {   // Fichero de Reuters de datos
      if ((fichero = fopen("VectoresR.LXM","r")) == NULL)
      {
      printf("ERROR al abrir el archivo VectoresR.LXM");    //Abrimos el fichero para lectura
      exit(1);
      }
      //
      if ((fsalida=fopen("VectoresNR.LXM","w"))==NULL)
      {
       printf("ERROR al crear el fichero de Vectores Nuevo");    //Abrimos el fichero para escritura de los documentos seleccionados
       exit(1);
      }
      if ((fzipt=fopen("DatosR.TXT","r"))==NULL)
      {
```

```
     printf("ERROR al crear el fichero de Vectores Zipf ");    //Abrimos el fichero para escritura de los documentos seleccionados
     exit(1);
     }
    if ((fzipn=fopen("DatosNR.TXT","w"))==NULL)
    {
     printf("ERROR al crear el fichero de Vectores Zipf Nuevo");    //Abrimos el fichero para escritura de los documentos
     exit(1);
    }
 }
 else
 {   // Fichero de Texto de Datos
    if ((fichero = fopen("VectoresT.LXM","r")) == NULL)
    {
    printf("ERROR al abrir el archivo VectoresT.LXM");   //Abrimos el fichero para lectura
    exit(1);
     }
    if ((fsalida=fopen("VectoresNT.LXM","w"))==NULL)
    {
     printf("ERROR al crear el fichero de Vectores Nuevo");
     exit(1);
    }
    if ((fzipt=fopen("DatosT.TXT","r"))==NULL)
    {
     printf("ERROR al crear el fichero de Vectores Zipf Nuevo");    //Abrimos el fichero para escritura de los documentos
     exit(1);
    }
    if ((fzipn=fopen("DatosNT.TXT","w"))==NULL)
    {
     printf("ERROR al crear el fichero de Vectores Zipf Nuevo");    //Abrimos el fichero para escritura de los documentos
       exit(1);
    }
 }

float dataz[termlexemas];       // Declaracion de la matriz auxiliar de datos que recibirá los terminos de cada documento
char nombdocum[10];             // se usa para leer la primera columna de documento
int ndocum;
char plinea[34];
fgets(plinea,33,fichero);       // se usa para leer la cabecera del fichero, que dice el num de filas y columnas

salto=' ';
int num=0;
for (int i=0; i<tdoc; i++)      // Leemos del fichero general todos los documentos para seleccionarlo segun grupos
{
   fscanf(fichero,"%s %c",&nombdocum,&salto);   // Leemos la primera columna, el numero de documento
   for (int j=0; j<termlexemas; j++)
   {
    fscanf(fichero,"%f",&dataz[j]);
    fscanf(fichero,"%c",&salto);
   }
   fscanf(fichero,"\n");

    if (encuentra(i+1,Grup)!=-1)        // Si lo encuentra (diferente de 0) existe en el vector V y son documentos
       {  num++;                        // de los grupos seleccionados, luego lo grabamos en el nuevo fichero
        fprintf(fsalida,"%3d %s %c",num,nombdocum,salto);
        for (int j=0; j<termlexemas; j++)
        {
         fprintf(fsalida,"%f",dataz[j]);
         fprintf(fsalida,"%c",salto);
        }
        fprintf(fsalida,"\n");
       }
 }

salto=' ';
num=0;
for (int i=0; i<tdoc; i++)      // Leemos del fichero zipt todos los documentos para seleccionarlo segun grupos
{
   fscanf(fzipt,"%3d %c",&ndocum,&salto);   // Leemos la primera columna, el numero de documento

   for (int j=0; j<TERMINOS_MAX; j++)
   {
    fscanf(fzipt,"%f",&dataz[j]);
    fscanf(fzipt,"%c",&salto);
   }

   fscanf(fzipt,"\n");
```

```
      if (encuentra(i+1,Grup)!=-1)           // Si lo encuentra (diferente de 0) existe en el vector V y son documentos
         {  num++;                           // de los grupos seleccionados, luego lo grabamos en el nuevo fichero
          fprintf(fzipn,"%3d %3d %c",num,ndocum,salto);

          for (int j=0; j<TERMINOS_MAX; j++)
          {
           fprintf(fzipn,"%f",dataz[j]);
           fprintf(fzipn,"%c",salto);
          }
          fprintf(fzipn,"\n");
         }
   }

   fclose(fichero);
   fclose(fsalida);
   fclose(fzipt);
   fclose(fzipn);

    for (int i=0;i<MAXDOCUM;i++)
    {
     printf("\nDocumento a Agrupar %d de Clase %d ",Grup.vv[i],Grup.clase[i]);
     int qdocum=SacaDocum(tipo,Grup.vv[i]);       // Sacamos el numero de documento (del arbol) correspondiente al documento a
evaluar
      Grup.arbdocum[i]=qdocum;
      printf(" representado en el arbol como %d",Grup.arbdocum[i]);
     }
  }

void MatrizGenetico(int gen,int seed,int n_genes,double pf1)
// Creamos el fichero de resultados del genetico
{
 char salto=' ';

  if (gen==0)
   {
     if ((fsalida=fopen("MatrizDatosXXX.txt","w"))==NULL)
      {
       printf("ERROR al crear el fichero archivo.txt");
       exit(1);
       }
      fprintf(fsalida,"---------------------------------------- RESULTADOS DEL SISTEMA GENETICO ----------------------------------------
--\n\n");
      fprintf(fsalida,"Generacion; SEMILLA  ;DOCUM;PMUTAC;PCRUCE;TORNEO;TERM;POBLACION; FITNESS  ;
DISTANCIA ; SIMILITUD ; INVERSA-SIM            \n");
      fprintf(fsalida,"-------------------------------------------------------------------------------------------------------------------\n");
   }

     Mbosquefitness[0].bien=sumabuenp;

    fprintf(fsalida,"%10d;%7d;%5d;%6.3f;%6.2f;%4d  ;%4d;%9d;%8.8f; %8.8f;
%8.8f;%8.8f;",gen,seed,MAXDOCUM,PROB_MUTACION,PROB_CRUCE,TAM_TORNEO,TERMINOS_MAX,LONG_POBL
ACION,Mbosquefitness[0].fitness,Mbosquefitness[0].distancia,Mbosquefitness[0].similitud,1 /(Mbosquefitness[0].similitud));

     for (int j=0;j<n_genes;j++)
     fprintf(fsalida,"%2d",Mbosquefitness[0].genoma[j]);
     fprintf(fsalida,";%3c",salto);
     fprintf(fsalida,"%3d;%8.8f", Mbosquefitness[0].bien,pf1);

     fprintf(fsalida,";%2d;%2d;%2d;%2d",jm1,jm2,jm3,jm4);

     fprintf(fsalida,";%2d;%2d;%2d;%2d",jc1,jc2,jc3,jc4);

     fprintf(fsalida,"\n");
     printf("\n Paso 4");
}

int busca(unsigned char t,int n_genes,unsigned char numarray[])
 /* busqueda secuencial para verificar si se repite el numero generado */
 /* retorna 0 si es que no lo encuentra (no esta repetido) y diferente de 0 si lo encuentra */
 // de esta forma comprobamos si el documento no se repita en el cromosoma
 {

   for(int a=0;a<n_genes;a++)
   {
     if(numarray[a]==t)
```

```
      {
        return (a);
        break;
      }
  }
    return (0);
}
```

void Cruce_Individuos1(int generacion,int n_genes,int indices[LONG_POBLACION],double bosquefitness[],int gana[2])

```
/* Operador de Cruce de Individuos (Genera 1 nuevo individuo basado en el nuevo operador de mascara diseñado) */
{
 int i1,i2,p,elegido,noelegido,pos,npos;
 unsigned char vnoelegido,velegido;
 unsigned char Newpop[TGENES];

 int marca[n_genes];

    for (int i=0;i<n_genes;i++)     // Marcamos todas las posiciones del elemento a crear con 0 (no procesado)
     marca[i]=0;
    // i1 indica el primer  individuo, i2 indica el segundo individuo, i1 e i2 individuos diferentes
        i1=gana[0];          // Asignamos los elementos ganadores del torneo
        i2=gana[1];

        // Mostramos los cromosomas Padres seleccionados
      /*
        printf("Cruce --> Cromosoma:%d 1er Padre; ",i1+1);
        for (int v=0;v<n_genes;v++)
           printf("%2d",population[indices[i1]][v]);
        printf("; Fitness: %.8f;\n",bosquefitness[i1]);
        printf("Cruce --> Cromosoma:%d 2do Padre; ",i2+1);
        for (int v=0;v<n_genes;v++)
            printf("%2d",population[indices[i2]][v]);
        printf("; Fitness: %.8f; ",bosquefitness[i2]);
       */
        p=Randint(1,2);             // Individuo Elegido como Mascara
  ///      printf("Elegido:%d\n",p);   // Mostramos cual es el Padre elegido como Mascara del cruce
         if (p==1)                //  Elegimos los indices a usar del padre
               {  // Elegido indice de primer padre
               noelegido=indices[i1];
               elegido=indices[i2];
               }
              else
               {  // Elegido indice de segundo padre
               noelegido=indices[i2];
               elegido=indices[i1];
               }
        for (int v=0;v<n_genes;v++)
        {
          if ((population[indices[i2]][v] !=0) && (population[indices[i1]][v]!=0) && (marca[v]==0) )
           {   // Si valor de los padres son documentos (diferentes de 0) y no procesado (marca =0 )

               vnoelegido=population[noelegido][v];   // Elegido el valor del Padre elegido y del Padre NoElegido
               velegido=population[elegido][v];

                 for (int m=0;m<n_genes;m++)               // En Newpop metemos los valores del padre elegido
                 Newpop[m]=population[elegido][m];

                   pos=busca(vnoelegido,n_genes,Newpop);   // Buscamos el correspondiente Vnoelegido en Newpop

                 // printf("elegido %d;vnoelegido %d, pos %d, v %d\n",velegido,vnoelegido,pos,v);
                  if ((pos!= 0) && (marca[pos]==0))      // Si lo encuentra y no procesado (marca=0))
                  {
                    npos=busca(vnoelegido,n_genes,Newaux); // Buscamos vnoelegido en NewAux (si ya esta intercambiado)
                   //   printf("npos %d",npos);
                    if (npos == 0)                    // (SI NO ESTA: Retorna 0 si no lo encuentra)
                     { Newaux[v]=vnoelegido;
                       Newaux[pos]=velegido;
                       marca[v]=1;
                       marca[pos]=1;
                   //    printf("Hizo intercambio gen %d; se marco: y pos %d\n",v,pos);
                      }
                     else
                      {
                   //    printf("no hizo intercambio esta en new gen %d;se marco\n",v);
```

```
                    Newaux[v]=population[elegido][v];
                    marca[v]=1;
                   }
                 }       // fin del if principal (population[generacion][elegido][k]== vnoelegido) && (marca[k]==0)
                 else
                 {
                 //  printf("no hizo intercambio ya procesado gen %d; se marco\n",v);
                   Newaux[v]=population[elegido][v];
                   marca[v]=1;
                   }

            } // Fin del Entonces: Si valor de los padres son documentos (diferentes de 0) y no procesado (marca =0 )
           else
           {
             if (marca[v]==0)
              {
               Newaux[v]=population[elegido][v];
               marca[v]=1;
              // printf("Puso la mascara gen %d; se marco\n",v);
               }
              // else
              //  printf("ya procesado gen %d\n",v);
            }  // fin de population

        }    // Fin del for de v a n_genes

  }

void Mutacion(int i,int n_genes)
/* Operador de Mutacion */
{
 int p,j,j1;
 unsigned char aux2;
 Set_random(seed2);                    // Fija la semilla seed con semilla 2 para Randint y Randfloat


     // Se determina del cromosoma, el gen que corresponden a la posicion que se va a mutar
     do
     {
      do                               // Buscamos solamente terminales (documentos del arbol)
      {
      j=Randint(1,n_genes-1);                 // j determina el gen a mutar del individuo
      } while (population[i][j]==0);

      do                               // Buscamos solamente terminales (documentos del arbol)
      {
       j1=Randint(1,n_genes-1);               // j1 determina el otro gen a mutar
      } while (population[i][j1]==0);
     } while (j==j1);                      // Repite mientras sean iguales para coger terminales diferentes

    aux2=population[i][j1];
  /*
     printf("El Cromosoma del Individuo: %d Antes de mutar en %d %d ",i+1,j,j1);
     printf("es:");
        for (int v=0;v<n_genes;v++)
         printf("%2d",population[i][v]);
     printf(";\n");
  */
     // Intercambiamos los elementos
      population[i][j1]=population[i][j];

      population[i][j]=aux2;

   /*         printf("El New Individuo: %d Mutado en Gen %d %d es: ---------->",i+1,j,j1);

         for (int v=0;v<n_genes;v++)
          printf("%2d",population[i][v]);
     printf(";\n");
  */
   }

void quicksort (double A[],int test[],int nrows)
{
  // Ordenamos los fitness y el indice para acceder al cromosoma
  // El método de ordenación es el de la burbuja
   double temp1;
```

```
  int temp2;

  for(int i=0; i<nrows-1; i++)
  {
   for(int j=i+1; j<nrows; j++)
   {
     if (A[i]>A[j])
     {
           temp1 = A[i];
           A[i] = A[j];
           A[j] = temp1;
           temp2 = test[i];
           test[i] = test[j];
           test[j] = temp2;
     }
   }
  }

}

void EvaluaIndividuo(int i,int generacion,int n_genes,Arbol bosque [LONG_POBLACION])
{
  // Evaluamos el fitness del individuo, lo insertamos en el arbol y lo mostramos

    float vectordocum[termlexemas];              // Vector auxiliar a partir de los terminos

    int datodoc;
    int j=1;  //nos saltamos el primer gen pues ya esta en la estructura


     do    // Generamos los nodos para el arbol a partir del cromosoma generado
     {
      datodoc=population[i][j];      // Retornamos el documento seleccionado (de cadena -char- a numero)

      for (int h=0;h<termlexemas;h++)
        {  if (datodoc)                        // Buscamos el vector documental asociado al documento
           {  //printf("%02d %02d\n",datodoc-1,h);
              vectordocum[h]= datax[datodoc-1][h];
           }
        }   // Fin del For


       bosque[i].InsertarPreorden(datodoc,vectordocum,j,0,0);   // Insertamos el Documento y su vector de terminos en el arbol
       j++;

      }while (j<=n_genes);  // Fin de generacion de nodos del arbol

      bosque[i].ActualizaArbol(0);

      bosque[i].calcularFitness(alfa);                    //Calcula el fitness

      char *oper;

      system("cls");
      textcolor(14);
      if (generacion==0)
        printf("Generacion: %d, Arbol #%02d: ",generacion,i+1);   // Generacion inicial
      else
        printf("Nueva Poblacion en la Generacion %d, Arbol #%02d: ",generacion,i+1); // Nuevos Crom.. Sgte generacion

     for (int v=0;v<n_genes;v++)
         printf("%2d",population[i][v]);              //Imprime los Genes del Individuo
     printf("; Fitness: %8.8f\n",bosque[i].fitness);    //Imprime el Fitness calculado del individuo

     textcolor(LIGHTGRAY);
     if (generacion==0)
     {  bosque[i].Ver(1,40,6);                        //Imprime el Arbol generado en pantalla del individuo
      // system("pause");
     }
}

void Ordena_Torneo (int generacion,int n_genes,double bosquefitness[])

// Ordenamos los cromosomas para aplicar el torneo
{
```

```
 for (int i=0;i<LONG_POBLACION;i++)          // Declaramos los indices de manera secuencial
  indices[i]=i;

 /* Reserva de memoria para la estructura que almacena los indices de los individuos seleccionados para competir en el torneo */
  Torneo = (int *) calloc ((unsigned)TAM_TORNEO,sizeof(int));

 if (Torneo == NULL)
  { printf("Error en la reserva de memoria de la estructura Torneo");
   abort();
   }

 quicksort(bosquefitness,indices,LONG_POBLACION);    // Ordenamos el fitness de los arboles de la poblacion de menor a mayor

   /* textcolor(14);
   printf("Ordenamos por Fitness los Cromosomas de la Generacion %d,para aplicar el Torneo\n",generacion);
  textcolor(LIGHTGRAY);
   for (int b=0;b<LONG_POBLACION;b++)
   { printf("Orden %02d: ",b+1);
    for (int v=0;v<n_genes;v++)
        {
         printf("%2d",population[indices[b]][v]);   //imprime los genes ordenados - usa el indice para obtener el gen
        }
       printf("; Fitness: %.8f %d\n",bosquefitness[b],indices[b]);    // Imprime el fitness para hacer el torneo
    }
    retarda(90);
  */
}

int Seleccion_Torneo (int generacion,int n_genes,double bosquefitness[])
// Selecciona la poblacion intermedia por la tecnica del Torneo
{
 int k, repetido, mejor_torneo,ganador;

 // Se procede a la seleccion de los cromosomas segun el metodo del Torneo
 /* textcolor(14);
 printf("Seleccionamos los cromosomas de la Poblacion por el Metodo de Torneo\n");
 textcolor(LIGHTGRAY);
 */
   Torneo[0] = Randint(0,LONG_POBLACION-1);
   mejor_torneo=Torneo[0];
   for (int j=1;j<TAM_TORNEO;j++)
    {
     do
     {
       Torneo[j] = Randint(0,LONG_POBLACION-1);
       repetido=0; k=0;
       while ((k<j) && (!repetido))

          if (Torneo[j]==Torneo[k])
             repetido=1;
          else
             k++;

       }
       while (repetido);
     // Se toma como referencia que se tiene ordenado de menor fitness a mayor fitness si minimizamos
     // Y de Mayor Fitness a Menor Fitness si Maximizamos  - El que este en la posicion mas baja siempre será el mejor

     if (Torneo[j]<mejor_torneo)
      mejor_torneo=Torneo[j];

     } // fin del for de j

   /*
    printf("Mejor Torneo --> Cromosoma %2d Seleccionado:",mejor_torneo+1);
    for (int v=0;v<n_genes;v++)
        printf("%2d",population[indices[mejor_torneo]][v]);

    printf("; Fitness: %.8f\n",bosquefitness[mejor_torneo]);        // Imprime el fitness
   */

    ganador=mejor_torneo;                          // Ganador del Torneo

    return ganador;

 }
```

```
void RealizaMutaciones(int generacion,int n_genes)
{
  float vectordocum[termlexemas];              // Vector auxiliar a partir de los terminos

  int datodoc;

  system("cls");
  jm1=0,jm2=0,jm3=0,jm4=0;
  // Empezamos a realizar todas la mutaciones
 for (int ii=0; ii<LONG_POBLACION;ii++)
 {
   //printf("Cromosoma entrada %02d: ",ii+1);
   for (int v=0;v<n_genes;v++)
   {  Newpopulation [ii][v]=population[ii][v]; // Guardamos la poblacion de cada individuo para ver si luego de la mutacion mejora
   //   printf("%2d",population[ii][v]);       //imprime los genes de la poblacion anterior
    }
   Newbosquefitness[ii]=bosquefitness[ii];
   Newbosquedistancia[ii]=bosquedistancia[ii];
   Newbosquesimilitud[ii]=bosquesimilitud[ii];

   // printf("; Fitness: %.8f \n",bosquefitness[ii]);

    float mu_next=Randfloat(0,1);

    if ( mu_next <=PROB_MUTACION )            // Aplicamos la mutacion
    {
     jm1++;
     textcolor(13);
     Mutacion(ii,n_genes);

     Arbol *bosqueM;             // Estructura que contendra todos los nuevos arboles luego de la mutacion
     try
     {
        bosqueM=new Arbol();

        int j=1;  //nos saltamos el primer gen pues ya esta en la estructura

        do    // Generamos los nodos para el arbol a partir del cromosoma generado
        {
           datodoc=population[ii][j];      // Retornamos el documento seleccionado (de cadena -char- a numero)
           for (int h=0;h<termlexemas;h++)
            {  if (datodoc)                          // Buscamos el vector documental asociado al documento
                  vectordocum[h]= datax[datodoc-1][h];
             }   // Fin del For
         bosqueM->InsertarPreorden(datodoc,vectordocum,j,0,0);   // Insertamos el Documento y su vector de terminos en el arbol

         j++;

        } while (j<=n_genes);  // Fin de generacion de nodos del arbol

        bosqueM->ActualizaArbol(0);
        bosqueM->calcularFitness(alfa);                    //Calcula el fitness

        system("cls");
        printf("Generacion: %d, MUTACION del Arbol #%02d: ",generacion,ii+1);

        for (int v=0;v<n_genes;v++)
          printf("%2d",population[ii][v]);             //Imprime los Genes del Individuo
        printf("; Fitness 1: %8.8f\n",bosqueM->fitness);  //Imprime el Fitness calculado del individuo

       // bosqueM->Ver(1,40,6);                         //Imprime el Arbol generado en pantalla del individuo

       if (bosqueM->fitness < bosquefitness[ii])         // Comprobamos si el cromosoma mutado mejora el fitness viejo
         {
           jm2++;
           Newbosquefitness[ii]=bosqueM->fitness;  // Si lo mejora pues lo asignamos
           Newbosquedistancia[ii]=bosqueM->distancia;  // Si lo mejora pues lo asignamos
           Newbosquesimilitud[ii]=bosqueM->similitud;  // Si lo mejora pues lo asignamos
           for (int v=0;v<n_genes;v++)
             Newpopulation [ii][v]=population[ii][v];

         }
        else jm3++;                          // Se rechaza la mutacion
        delete bosqueM;
```

```
      } //
      catch(bad_alloc e)
      {
            cout << "Insuficiente espacio de memoria para la Mutacion\n";
            getchar();
       }

     } // fin de la probabilidad de mutacion

    else
        jm4++;              // No se hace la mutacion

  }   // Fin del lazo del ii  (mutaciones)

}

void RealizaCruces(int generacion,int n_genes)
{
     float vectordocum[termlexemas];              // Vector auxiliar a partir de los terminos

      int datodoc;

      jc1=0,jc2=0,jc3=0,jc4=0;                  // Contadores para las operaciones de cruce

      for (int ii=0; ii<LONG_POBLACION;ii++)
      {
       float cruce1_next=Randfloat(0,1);

       if ( cruce1_next <= PROB_CRUCE )          // Aplicamos el operador de cruce 1   // ojooooo
       {
        jc1++;                                   // Veces que se aplica el cruce 1
        textcolor(15);
        Ordena_Torneo(generacion,n_genes,bosquefitness);   // Ordenamos para aplicar el Torneo a fin de seleccionar padres
        int gana[2];
        do     // Aplicamos el torneo para elegir 2 individuos diferentes
        {
        gana[0]=Seleccion_Torneo(generacion,n_genes,bosquefitness);   // Aplicamos el Torneo
        gana[1]=Seleccion_Torneo(generacion,n_genes,bosquefitness);
         } while(gana[0]==gana[1]);    // Seleccionamos dos individuos diferentes para hacer el cruce
         free (Torneo);
         for (int v=0;v<n_genes;v++)           // Inicializamos a 0 el vector para cada operador de cruce 1
           Newaux[v]=0;
         textcolor(11);
         Cruce_Individuos1(generacion,n_genes,indices,bosquefitness,gana);  //  Se aplica el cruce 1 entre individuos

         //printf("Cruce: Creado Cromosoma ----> ");  // Mostramos el individuo creado despues de cruzar

          for (int v=0;v<n_genes;v++)
          {
          //    printf("%2d",Newaux[v]);
            population[indices[gana[0]]][v]=Newaux[v];     // Nuevo Individuo asignamos al 1er padre
          }

          // Insertar el nuevo individuo en el arbol para calcular su fitness

         Arbol *bosqueC=0;   // Estructura que contendra todos los nuevos arboles luego del cruce
         try
         {
          bosqueC = new Arbol();

          int i=indices[gana[0]];                  // Ojo asigna primer padre para reemplazar
          int ic=indices[gana[1]];                  //  Segundo padre

          int j=1;  //nos saltamos el primer gen pues ya esta en la estructura

          do    // Generamos los nodos para el arbol a partir del cromosoma generado
          {
            datodoc=population[i][j];       // Retornamos el documento seleccionado (de cadena -char- a numero)
            for (int h=0;h<termlexemas;h++)
            {  if (datodoc)                        // Buscamos el vector documental asociado al documento
                {  //printf("%02d %02d\n",datodoc-1,h);
                  vectordocum[h]= datax[datodoc-1][h];
                }
             }   // Fin del For

           bosqueC->InsertarPreorden(datodoc,vectordocum,j,0,0);   // Insertamos el Documento y su vector de terminos en el arbol
```

```
            j++;

           }while (j<=n_genes);  // Fin de generacion de nodos del arbol

           bosqueC->ActualizaArbol(0);
           bosqueC->calcularFitness(alfa);                    //Calcula el fitness

           system("cls");
           printf("Generacion: %d, CRUCE %d del Arbol #%02d y #%02d: ",generacion,ii,i+1,ic+1);

           for (int v=0;v<n_genes;v++)
            printf("%2d",population[i][v]);              //Imprime los Genes del Individuo
           printf("; Fitness 1: %8.8f\n",bosqueC->fitness);    //Imprime el Fitness calculado del individuo
           textcolor(LIGHTGRAY);

          //  bosqueC->Ver(1,40,6);                        //Imprime el Arbol generado en pantalla del individuo

           if (bosqueC->fitness < Newbosquefitness[indices[gana[0]]])       // Comprobamos si el cromosoma mejora el fitness viejo
            {
            jc2++;
            Newbosquefitness[indices[gana[0]]]=bosqueC->fitness;
            Newbosquedistancia[indices[gana[0]]]=bosqueC->distancia;
            Newbosquesimilitud[indices[gana[0]]]=bosqueC->similitud;
            for (int r=0;r<n_genes;r++)
             Newpopulation [indices[gana[0]]][r]=population[indices[gana[0]]][r];
           }  // Fin del if
          else
             jc3++;                                  //  No mejora al padre 1

          delete  bosqueC;


         }   /// Fin del Try

         catch(bad_alloc e)
         {
            cout << "Insuficiente espacio de memoria para el Cruce\n";
            getchar();
         }

        } // Fin del if de la probabilidad de cruce

        else  jc4++;                                 //  No se hace el cruce

       }  // Fin del ii      Fin de los cruces de tipo 1

}

void ModiNuevaGeneracion(int generacion,int n_genes)
// Realizamos muchas mutaciones y cruces sobre la poblacion actual
{
    RealizaMutaciones(generacion,n_genes);
    RealizaCruces(generacion,n_genes);

    // Restauramos los valores iniciales de los cromosomas anteriores
      for (int t=0; t<LONG_POBLACION;t++)
      {
       bosquefitness[t]=Newbosquefitness[t];
       bosquedistancia[t]=Newbosquedistancia[t];
       bosquesimilitud[t]=Newbosquesimilitud[t];

       bosque[t].fitness=bosquefitness[t];
       bosque[t].distancia=bosquedistancia[t];
       bosque[t].similitud=bosquesimilitud[t];
       }
 }

void lectura_datoszip(int tipo)
{
 // Leemos del fichero DatoR.txt /DatosT.txt de los datos -- Datos seleccionado por el Zipf
 // Resultados almacenados en la matriz "datax"
 // Datos R: tiene todo el fichero entero con los NZIPF terminos seleccionados (40)
 // Datos NR: tiene solo los documentos seleccionados con los NZIPF terminos seleccionados (40)

  FILE *fichero;
```

```
 char salto;
 int ndocum;      // Numero de documento en la coleccion
 int numdocum;    // Numero correlativo en el arbol

  // Rellenando la matriz desde el Fichero de Preprocesamiento
   if (tipo==0)
   {   // Fichero de Reuters de datos
       if ((fichero = fopen("DatosNR.TXT","r")) == NULL)
       {
       printf("ERROR al abrir el archivo DatosNR.TXT");    //Abrimos el fichero para escritura
       exit(1);
       }
   }
   else
   {   // Fichero de Texto de Datos
       if ((fichero = fopen("DatosNT.TXT","r")) == NULL)
       {
       printf("ERROR al abrir el archivo DatosNT.TXT");   //Abrimos el fichero para escritura
       exit(1);
      }
   }

  salto=' ';
  for (int i=0; i<documentos; i++)
  {
     fscanf(fichero,"%3d %3d %c",&numdocum,&ndocum,&salto);   // Leemos la primera columna, el numero de documento
     for (int j=0; j<termlexemas; j++)
     {
      fscanf(fichero,"%f",&datax[i][j]);
      fscanf(fichero,"%c",&salto);
      }
    fscanf(fichero,"\n");
  }
  fclose(fichero);

}

void lectura_grupos(int tipo)
{
 // Leemos del fichero VectoresR.txt /VectoresT.txt de los datos ... toda la data entera
 // Resultados almacenados en la matriz "datax"

  FILE *fichero;
  char salto;

   // Rellenando la matriz desde el Fichero de Preprocesamiento
    if (tipo==0)
    {   // Fichero de Reuters de datos
        if ((fichero = fopen("VectoresNR.LXM","r")) == NULL)
        {
        printf("ERROR al abrir el archivo VectoresR.LXM");    //Abrimos el fichero para escritura
        exit(1);
        }
    }
    else
    {   // Fichero de Texto de Datos
        if ((fichero = fopen("VectoresNT.LXM","r")) == NULL)
        {
        printf("ERROR al abrir el archivo VectoresT.LXM");   //Abrimos el fichero para escritura
        exit(1);
       }
    }

   int numdocum;        // Numero correlativo en el arbol
   char nombdocum[10];  // se usa para leer la primera columna de documento (numero de documento)
   salto=' ';
   for (int i=0; i<documentos; i++)
   {
      fscanf(fichero,"%3d %s",&numdocum,&nombdocum);   // Leemos la primera columna, el numero de documento
      for (int j=0; j<termlexemas; j++)
      {
       fscanf(fichero,"%f",&datax[i][j]);
       fscanf(fichero,"%c",&salto);
       }
     fscanf(fichero,"\n");
   }
```

```
  fclose(fichero);
}

void generaCromosoma(unsigned char *buff)
//Genera un cromosoma aleatoriamente valido para la altura pedida
//el cronosoma podra generar un arbol con una altura maxima ALTURA_MAX y numero de DOCUMENTOS
{
    int doc[documentos];
    for (int j=0;j<documentos;j++) doc[j]=j+1;    // Inicializa el vector con los documentos (de forma secuencial - ascendente)
    int pos;
    for (int j=0;j<documentos;j++)
    {
       pos = rand()%(documentos-1);     // Coge aleatoriamente una posicion
       int bak = doc[pos];              // Almacena temporalmente el valor (inicial) de dicha posicion
       doc[pos]=doc[j];                 // Reemplaza la posicion por el contenido del indice actual (secuencial) j
       doc[j]=bak;                    // Reemplaza el valor del indice actual j por la variable temporal almacenada
      //  printf("pos: %d,doc[pos]:%d,j %d, doc[j]:%d\n",pos,doc[pos],j,doc[j]);
      //  system("pause");
    }

    Adn * adn = new Adn(documentos,ALTURA_MAX);

    adn->getPreorden(buff);

    int i,j;
    for (i=0,j=0;i<documentos*2-1;i++)
    {
       if (buff[i] == 'f') buff[i]=0;
       else buff[i] = doc[j++];
    }

    free(adn);
}

void Mostrar(int n_genes,int generacion,unsigned char population [LONG_POBLACION][TGENES],Arbol bosque
[LONG_POBLACION],int tipo)
// Mostramos todos los Genes de la Poblacion y sus fitness obtenidos
{
      float vectd[termlexemas];              // Vector auxiliar a partir de los terminos
      int datafin,p;
      numero=1;
      int doc=(n_genes+1)/2;           // Numero de documentos
      menor=&bosque[0];
      pf1=0.0;                       // Inicializamos el promedio de fitness de cada generacion
      textcolor(14);
    ///   printf("Resumen de todos los Cromosomas creados en la Generacion %d y sus Fitness calculados\n",generacion);
      for (int b=0;b<LONG_POBLACION;b++)
      {
       if (menor->fitness > bosque[b].fitness)
        {    menor=&bosque[b];      //buscamos el menor fitness de toda la poblacion
             numero=b+1;
        }

       /*  printf("Arbol#%02d: ",b+1);
         for (int v=0;v<n_genes;v++)
         {  // textcolor(3+v%2);
         printf("%2d",population[b][v]);        //Imprime los genes
         }

         printf("\nXXX b %d; Fitness: %.8f\n",b,bosque[b].fitness);       //Imprime el fitness
       */
         bosquefitness[b]=bosque[b].fitness;
         bosquedistancia[b]=bosque[b].distancia;
         bosquesimilitud[b]=bosque[b].similitud;
         pf1+=bosque[b].fitness;                    // Acumula los valores de fitness obtenidos por cada individuo
      }   // Fin del for

     // Pintamos el menor fitness de toda la poblacion
      printf("Mejor Individuo en Generacion %d, Arbol Menor #%d, Fitness: %8.8f",generacion,numero,menor->fitness);

        menor=&bosque[numero-1];  // Asignamos en menor el Mejor individuo (mejor fitness)
        Mbosquefitness[0].fitness=bosque[numero-1].fitness;  // Asignamos el mejor fitness de la generacion
        Mbosquefitness[0].distancia=bosque[numero-1].distancia;  // Asignamos la distancia de la generacion
        Mbosquefitness[0].similitud=bosque[numero-1].similitud;

        for (int s=0;s<n_genes;s++)
```

```
        Mbosquefitness[0].genoma[s]=population[numero-1][s];  // Asignamos el mejor cromosoma de la generacion

       Mbosquefitness[0].bien=bosque[numero-1].bien;

       pf1=pf1/LONG_POBLACION;                          // Media de los Fitness en la Generacion evaluada
       double ax =(bosque[numero-1].fitness/pf1);       // Asignamos el tiempo total de vida del cromosoma
       Mbosquefitness[0].tiempovida= int (ceil(ax));    // que sera en proporcion del fitness y el promedio
                                          // se redondes hacia arriba el resultado obtenido (al menos 1)

       Mbosquefitness[0].edad = 0;                      //  Se inicializa la edad del cromosoma

       Mbosquefitness[0].vive = 1;                      //  Se asigna vida al cromosoma para la siguiente generacion

       if ((Mbosquefitness[0].fitness < MMbosquefitness[0].fitness) || (generacion==0))   // Verificamos siempre el mejor fitness
       {
        MMbosquefitness[0].fitness=Mbosquefitness[0].fitness;       // Asignamos el mejor fitness de las generaciones
          for (int s=0;s<n_genes;s++)
           MMbosquefitness[0].genoma[s]=Mbosquefitness[0].genoma[s]; // Asignamos el mejor cromosoma de las generaciones

           // Insertamos el mejor genoma encontrado en el arbol para luego poder mostrarlo graficamente
           p=1; //nos saltamos el primer gen pues ya esta en la estructura
           do    // Generamos los nodos del mejor arbol del algoritmo para pintarlo
           {
            datafin=MMbosquefitness[0].genoma[p];   // Retornamos el documento seleccionado (de cadena -char- a numero)
            for (int h=0;h<termlexemas;h++)
             {  if (datafin)                        // Buscamos el vector documental asociado al documento
                vectd[h]= datax[datafin-1][h];
             }   // Fin del For  -datafin- "Numero del documento" - vectordocum - "vector de terminos"

            MMbosquefitness[0].InsertarPreorden(datafin,vectd,p,0,0);   // Insertamos el Documento y su vector de terminos en el
                                                                        // arbol
            p++;
           }while (p<=n_genes);  // Fin de generar nodos del arbol

          //  Fin de insersion del mejor genoma
          MMbosquefitness[0].ActualizaArbol(0);
          for (int i=0;i<MAXDOCUM;i++)
           printf("\ni %d, valor %d, clase %d",i+1,Grup.vv[i],Grup.clase[i]);
           sumabuenp=0;
           for (int i=0;i<MAXDOCUM;i++)
           { buenp=0;
             printf("\nDocumento a Buscar %d Clase %d en el Arbol %d\n",Grup.vv[i],Grup.clase[i],Grup.arbdocum[i]);
             MMbosquefitness[0].BuscaVer(Grup.arbdocum[i],Grup.clase[i]); // Buscamos el numero de documento del arbol y
                                                                          // verificamos grupo
             sumabuenp+=buenp;                      // Buenp 1 acerto, y 0 no acerto en la agrupacion del arbol
              printf("\nSuma Parcial %d",sumabuenp);
             }
              printf("\nSUMA FINAL %d",sumabuenp);

            buenos=sumabuenp;

          }  // Fin del cambio de mejor fitness
          else
          {
            sumabuenp=buenos;

           }

       MatrizGenetico(generacion,seed,n_genes,pf1);

 }

void GeneraNuevosElite(int tipo,int generacion,int n_genes,Arbol bosque[LONG_POBLACION],double
bosquefitness[LONG_POBLACION])

// Generamos los nuevos individuos y aplicamos el elitismo
{
  unsigned char *ViveNewpopulation[LONG_POBLACION];      // Auxiliares para almacenar los cromosomas
  double Vivebosquefitness[LONG_POBLACION];             // sobrevivientes aplicando el elitismo
  double Vivebosquedistancia[LONG_POBLACION];           // sobrevivientes aplicando el elitismo
  double Vivebosquesimilitud[LONG_POBLACION];           // sobrevivientes aplicando el elitismo

  int nuevo=0;                                  // Empieza en 0 y dira cuantos nuevos sobreviven

   // Verificamos los tiempos de vida de los cromosomas
    Mbosquefitness[0].edad=Mbosquefitness[0].edad +1;   //  Incrementamos la edad del Individuo
```

```
    if (Mbosquefitness[0].edad <=Mbosquefitness[0].tiempovida)  // Comprobamos si el individuo vive
     {                                          // de acuerdo a su tiempo de vida
      Mbosquefitness[0].vive = 1;
      nuevo++;                                  // nuevo indica la cantidad de sobrevivientes
      Vivebosquefitness[nuevo]=Mbosquefitness[0].fitness;         // Almacenamos su fitness
      Vivebosquedistancia[nuevo]=Mbosquefitness[0].distancia;
      Vivebosquesimilitud[nuevo]=Mbosquefitness[0].similitud;
      ViveNewpopulation[nuevo]=Mbosquefitness[0].genoma;          // Y almacenamos el genoma para que sobreviva
     }
     else
      Mbosquefitness[0].vive = 0;                       // Matamos el individuo

    for (int i=0;i<nuevo;i++)                       //  Mantenemos los sobrevivientes para la siguiente generacion
    {
      for (int j=0;j<n_genes;j++)
       population[i][j]= ViveNewpopulation[i+1][j];

      bosquefitness[i]= Vivebosquefitness[i+1];
      bosquedistancia[i]= Vivebosquedistancia[i+1];
      bosquesimilitud[i]= Vivebosquesimilitud[i+1];
    }

   // Generamos nuevos individuos para los individuos que no van a sobrevivir de forma aleatoria

   bool bOk;

   for (int i=nuevo;i<LONG_POBLACION-1;i++)       // Generamos los nuevos cromosomas considerando los sobrevivientes
   {
      do
      {
        generaCromosoma(population[i]);                 //Generamos todos los cromosomas

        bOk=true;
        for (int j=0;j<i && bOk;j++)                    //Comprobamos que no estaba ya generado ese cromosoma
        {
           bOk=false;
           for (int k=0;k<=n_genes && !bOk;k++)
              if (population[i][k]!=population[j][k])
                 bOk=true;
        }   // Fin del For

      } while (!bOk);     // Fin de Generacion del cromosoma

       for (int v=0;v<n_genes;v++)          // Verificamos no exceda del numero de documentos
      { // printf("%2d*",population[i][v]);
        if (population[i][v]>MAXDOCUM)
            population[i][v]=0;
       }

      EvaluaIndividuo(i,generacion+1,n_genes,bosque); // Evaluamos Cada Individuo Generado de forma aleatoria

      bosquefitness[i]=bosque[i].fitness;          // asigna en bosquefitness el valor del fitness del individuo generado aletoriamente
      bosquedistancia[i]=bosque[i].distancia;      // y los valores de las metricas de distancia y similitud
      bosquesimilitud[i]=bosque[i].similitud;
    }
    // Pasamos el mejor fitness obtenido a la siguiente poblacion en la ultima posicion (LONG_POBLACION-1)
    bosquefitness[LONG_POBLACION-1]=Mbosquefitness[0].fitness;       // asigna en bosquefitness el mejor de los fitness
    bosquedistancia[LONG_POBLACION-1]=Mbosquefitness[0].distancia;   // y los valores de las metricas de distancia y similitud
    bosquesimilitud[LONG_POBLACION-1]=Mbosquefitness[0].similitud;
    for (int s=0;s<n_genes;s++)
     population[LONG_POBLACION-1][s]=Mbosquefitness[0].genoma[s];   // Retornamos el mejor cromosoma de la generacion

}

void PoblacionInicial(int tipo,int generacion,int n_genes)
// Creamos la Poblacion Inicial de individuos (Generacion 0), la Evaluamos y la mostramos en Pantalla
{

   lectura_grupos(tipo);      //lee los vectores de caracteristicas
   bool bOk;
   for (int i=0;i<LONG_POBLACION;i++)
   {
      system("cls");
      do
      {
```

```
        generaCromosoma(population[i]);                  //Generamos todos los cromosomas

        bOk=true;
        for (int j=0;j<i && bOk;j++)                     //Comprobamos que no estaba ya generado ese cromosoma
        {
           bOk=false;
           for (int k=0;k<=n_genes && !bOk;k++)
              if (population[i][k]!=population[j][k])
                 bOk=true;
        }   // Fin del For

      } while (!bOk);     // Fin de Generacion del cromosoma

      for (int v=0;v<n_genes;v++)         // Verificamos no exceda del numero de documentos
      { // printf("%2d*",population[i][v]);
        if (population[i][v]>MAXDOCUM)
            population[i][v]=0;
       }

      EvaluaIndividuo(i,generacion,n_genes,bosque); // Evaluamos Cada Individuo Poblacion Inicial
      free(bosque);

    }  // Fin de Longitud de Poblacion (Numero de Individuos)

    Mostrar(n_genes,generacion,population,bosque,tipo);  // Mostramos todos los Individuos

}

void Genetico(int tipo)
{
   int generacion=0,n_genes;                          // Declaracion de variables: Generacion y numero de genes
   int i,j,datodoc;
   seed=time(NULL);                                   // Asignamos a seed la fecha del sistema
   srand (seed);                                      // fijamos la semilla con la fecha del sistema  semilla
   tdoc=CalculoFicheros(0,tipo);                      // Obtenemos el número de documentos lematizados a procesar
   termlexemas=TERMINOS_MAX;                          // Se pone solo los 40 terminos previamente seleccionados por el metodo zipf
  documentos=MAXDOCUM;                                // Numero de documentos a procesar

   SeleccionaDatos(tipo,tdoc,termlexemas);            // 1 solamente para hacer el Kmeans en el modulo de prepocesa
   n_genes =2*(documentos)-1;                         // Tamaño del cromosoma debe ser igual a (2*numero de documentos) - 1

   PoblacionInicial(tipo,generacion,n_genes);

   do
   {
    generacion++;

     ModiNuevaGeneracion(generacion,n_genes); // Modificamos la Nueva Generacion

    Mostrar(n_genes,generacion,Newpopulation,bosque,tipo);  // Mostramos el Mejor

     if (generacion<N_GENERACIONES-1)
     GeneraNuevosElite(tipo,generacion,n_genes,bosque,bosquefitness); // Creamos la Nueva Poblacion


   }  while (generacion<N_GENERACIONES-1);  // Fin de Generaciones  (N_GENERACIONES-1 porque ya evaluamos la
                                            // Generacion 0)

fprintf(fsalida,"================================================================================
============================== \n");
   fprintf(fsalida,"El Mejor Individuo despues de %d Generaciones evaluadas tiene un Fitness:
%8.8f\n",generacion,MMbosquefitness[0].fitness);
   fprintf(fsalida,"\nEl Mejor Cromosoma es:");
   for (int j=0;j<n_genes;j++)
    fprintf(fsalida,"%2d",MMbosquefitness[0].genoma[j]);    // Mejor Genoma encontrado del AG
     system("cls");
    gotoxy(8,1);
    textcolor(15);
    printf("El Mejor Fitness del AG es: %8.8f cuya representacion grafica es:\n",MMbosquefitness[0].fitness);
   MMbosquefitness[0].Ver(1,40,3);      // Mostramos graficamente el mejor genoma del AG
   gotoxy(10,22);
   keybd_event(44,0,0,0);              // Mete el grafico del arbol al portapapeles, para mirarlo: Inicio-Ejecutar-clipbrd
   system("pause");
   fclose(fsalida);
}
```

MENU PRINCIPAL

MAIN.CPP

```
/* ==================== Menu Inicial del Sistema ============================================== */
//
// Jose Luis Castillo Sequera
//
#include <string.h>
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <process.h>
#include <dos.h>
#include "config.h"

void Genetico(int tipo);
void Preprocesa();
void Resultados();

enum{preproceso=0,sistemagen,resulta,salir};

enum{GenReuters=0,GenDatos,Salida};


void EscogerTipo(char *cadenita)
{
 char titulo[126];
 int linea=6;
 system("cls");
 textcolor(14);
 gotoxy(15,linea-3);
 char *cadena="ESCOGER TIPO DE ARCHIVO A PROCESAR ";
 strcpy(titulo, cadena);
 strcat(titulo,cadenita);

 printf(titulo);
 gotoxy(26,linea-1);
 textcolor(10);
 printf("Escoja su opcion:");
 gotoxy(26,linea+1);
 textcolor(15);
 printf("0. Base Documental Reuters.");
 gotoxy(26,linea+3);
 printf("1. Otra Base Documental.");
 gotoxy(26,linea+5);
 printf("2. Salir.");
 gotoxy(26,linea+7);
 textcolor(10);
 printf("-> ");
}


void Menu(void)
{
  // Colores Textcolor: 9:Azul / 10 : Verde / 11: Celeste / 12: Rojo / 13: Violeta / 14: Amarillo / 15: Blanco
 int linea=6;
 system("cls");

 textcolor(14);
 gotoxy(20,linea-3);
 printf("SISTEMA GENETICO DE AGRUPACION DE DOCUMENTOS");

 gotoxy(26,linea-1);
 textcolor(10);
 printf("Escoja su opcion:");
 gotoxy(26,linea+1);
 textcolor(15);
 printf("0. Preprocesamiento Documental.");
 gotoxy(26,linea+3);
 printf("1. Procesar Sistema Genetico");
 gotoxy(26,linea+5);
 printf("2. Resultados Comparativos.");
 gotoxy(26,linea+7);
 printf("3. Salir.");
 gotoxy(26,linea+11);
 textcolor(10);
```

```
 printf("-> ");
}

int main()

{
 int decide=1,tipo;

 char *cadenita;
 int opcion;
 Menu();
 textcolor(15);
 scanf("%d",&opcion);
 while(opcion!=salir)
{

  switch(opcion)
    {
   case preproceso: printf("Crea BD Documental ");
                             Preprocesa();
                             break;
    case sistemagen: printf("Sistema Genético   ");
                            cadenita="PARA EL SISTEMA GENETICO";
                            EscogerTipo(cadenita);
                            textcolor(15);
                            scanf("%d",&tipo);
                            while(tipo!=Salida)
                            {                                   // Tipo = 0; Fichero Reuters, Tipo = 1: Fichero Texto
                                 switch(tipo)
                                      {
                                        case  GenReuters : printf("Genetico aplicado a  BD Reuters ");
                                        {
                                            Genetico(tipo);
                                            break;
                                         }
                                        case  GenDatos : printf("Genetico aplicado a  BD de Editoriales");
                                         {
                                             Genetico(tipo);
                                              break;
                                         }
                                         default: gotoxy(32,20);
                                         {
                                           printf("Opcion IncorrectaXXX.");
                                          getch();
                                          EscogerTipo(cadenita);
                                          break;
                                          }
                                        }     // Fin de Switch
                                       EscogerTipo(cadenita);
                                       scanf("%d",&tipo);
                                  }         // Fin del Mientras
                                  break;

              case resulta:  printf("Genera los Resultados Comparativos");
                                  Resultados();
                                  break;
              default:
                                  gotoxy(32,20);
                                  printf("Opcion Incorrecta.");
                                  getch();
                                  Menu();
                                 break;
       }  //  Fin del Switch
   Menu();

   scanf("%d",&opcion);

  }      // Fin del Mientras

 gotoxy(32,27);
 printf("Fin de %s",__FILE__);
 system("cls");
 return 0;
};
```