Algoritmos Genéticos Aplicados a la Categorización Automática de Documentos

Yolis, E.[1], Britos, P. [2, 3], Perichisky, G. [3] & García-Martínez, R. [2]

[1] Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería.
Universidad de Buenos Aires. Paseo Colón 850 4to Piso. Ala Sur. (1063). Capital Federal. ARGENTINA
eugenioy@ubbi.com
[2] Centro de Ingeniería del Software e Ingeniería del Conocimiento (CAPIS)
Instituto Tecnológico de Buenos Aires. Av. Madero 399. (1106) Capital Federal. ARGENTINA
pbritos@itba.edu.ar, rgm@itba.edu.ar
[3] Programa de Doctorado en Ciencias Informáticas
Facultad de Informática. Universidad Nacional de La Plata. ARGENTINA
gperi@mara.fi.uba.ar

Resumen - La categorización automática de documentos ha estado recibiendo creciente atención debido al incremento en la cantidad de información disponible en forma electrónica y a la necesidad cada vez mayor de encontrar la información buscada en un tiempo mínimo. Si bien existen numerosos algoritmos para categorizar documentos, todos ellos evaluan un subconjunto pequeño del espacio de posibles soluciones. Este trabajo presenta un algoritmo genético adaptado al problema de categorización de documentos. El algoritmo propuesto introduce 5 nuevos operadores, diseñados específicamente para la resolución del problema de categorización. Los resultados obtenidos demuestran que el algoritmo genético logra explorar el espacio de búsqueda más amplia y eficientemente que los algoritmos previos tomados como referencia.

Palabras clave: Categorización Automática de Documentos, Algoritmos Genéticos, Computación Evolutiva

Abstract - Automatic document clustering has been receiving increasing attention due to the growing amount of information available in digital formats, and the importance of finding the information required faster every day. Even though several clustering algorithms have been developed, all of them evaluate just a small part of the solution space. This paper presents an adaption of a genetic algorithm to the document clustering problem. This new algoritm introduces 5 new operators, designed specifically for the problem being solved. The obtained results show that the genetic algorithm achieves a wider and more efficient exploration of the solution space than the previously developed algorithms that were taken as reference.

Keywords: Automatic Document Clustering, Genetic Algorithms, Evolutionary Computation

Introducción

La *Categorización de Documentos* (Document Clustering) puede definirse como la tarea de separar documentos en grupos. El criterio de agrupamiento se basa en las similitudes existentes entre ellos [1].

El objetivo de este trabajo es estudiar cómo pueden aplicarse los algoritmos genéticos, que han demostrado ser útiles para ayudar a resolver problemas de optimización, al problema de encontrar en forma automática la mejor categorización de documentos. Se estudia de qué forma las características de los algoritmos genéticos pueden utilizarse para diseñar un algoritmo que mejore el rendimiento de los algoritmos que actualmente se aplican al problema, y se evaluan los resultados de acuerdo a la calidad de las soluciones obtenidas.

La categorización automática de documentos se comenzó a investigar dentro del campo de "Recuperación de Información" (en inglés, "Information Retrieval").

Ésta es la rama de la informática que investiga la búsqueda eficiente de información relativa a un tema en particular en grandes volúmenes de documentación [2].

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" como parte de su contenido.

En el contexto del recupero de información, Van Rijsbergen formula en 1979 la denominada "Hipótesis del Agrupamiento" (en inglés, "Cluster Hypothesis") [3]. Básicamente, la hipótesis del agrupamiento sostiene que "los documentos fuertemente asociados tienden a ser relevantes para la mísma consulta". Esta hipótesis ha sido verificada experimentalmente. [4] [5] [6]

Basándose en esta hipótesis, la categorización automática de documentos tiene en cuenta el contenido de los documentos para agruparlos, ya que documentos similares contendrán palabras (términos) similares.

La categorización automática de documentos se investiga dentro del campo del recupero de información como una herramienta capaz de mejorar la calidad de las soluciones ofrecidas.

Sus aplicaciones más importantes son:

- Mejorar el rendimiento de los motores de búsqueda de información mediante la categorización previa de todos los documentos disponibles [3] [7] [8].
- Facilitar la revisión de resultados por parte del usuario final, agrupando los resultados luego de realizar la búsqueda [4] [9] [10] [11].

Una definición del problema del agrupamiento de documentos (que es aplicable al agrupamiento de cualquier tipo de elementos), se puede enunciar de la siguiente manera (adaptada de [12]):

Dado un conjunto *S*, de *N* documentos, se quiere encontrar la partición $S_1, S_2, ..., S_k$, tal que cada uno de los *N* documentos se encuentre sólamente en un grupo S_i, y que cada documento sea más similar a los documentos del mismo grupo que a los documentos asignados a los otros grupos.

Para poder definir medidas de semejanza entre los objetos a agrupar, éstos se representan mediante vectores $v = (a_1 , a_2 , ... , a_m)$, donde cada componente del vector es el valor de un atributo del objeto.

De esta forma, cada uno de los objetos a agrupar es un punto en un Espacio Euclideano de *m* dimensiones, R^m.

Para el manejo de documentos, los investigadores dedicados a las diversas ramas de la Recuperación de Información adoptaron tempranamente una representación vectorial para ellos [3]. En este modelo, cada documento es considerado un vector *d* en el espacio de términos (el conjunto de palabras que aparecen en por lo menos uno de los documentos de la colección).

La definición de "centroide" de un grupo de elementos representados vectorialmente es utilizada por muchos algoritmos de agrupamiento.

Se define al centroide C_s de un grupo de elementos *S*, que contiene *h* elementos s_i como [12] [13]:

$$C_s = \frac{\sum_{i=1}^{h} s_i}{h} \quad (1)$$

Lo que es simplemente el promedio de los vectores que componen el grupo. Cada componente del vector centroide es el promedio del valor de esa componente para los miembros del grupo.

Para poder evaluar la semilitud existente entre dos documentos, es necesario definir una medida cuantitativa para la misma.

En el modelo de representación vectorial, siendo $||v||$ la longitud (norma) del vector *v*, una de las medidas de semejanza más utilizadas [12] [13] [14] [15] es el Coeficiente del coseno extendido:

$$\cos(d_1, d_2) = \frac{d_1 \bullet d_2}{\|d_1\| * \|d_2\|} \quad (2)$$

Esta medida tiene la propiedad de no depender del tamaño de los documentos, ya que $\mathbf{cos(d_1,d_2) = cos(a.d_1,d_2)}$ para a > 0. Sin embargo, los documentos se normalizan para que tengan longitud unitaria, ya que entonces,

$$\cos(d_1, d_2) = d_1 \bullet d_2 \quad (3)$$

Y la semejanza entre los documentos se puede calcular como el producto vectorial entre ellos.

Trabajo previo sobre categorización

Las formas de clasificación 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 1.

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

- **Extrínsecas (supervisadas)** : Las clases a las que pertenecen los objetos están predefinidas, y se conocen ejemplos de cada una, ó algunos de los

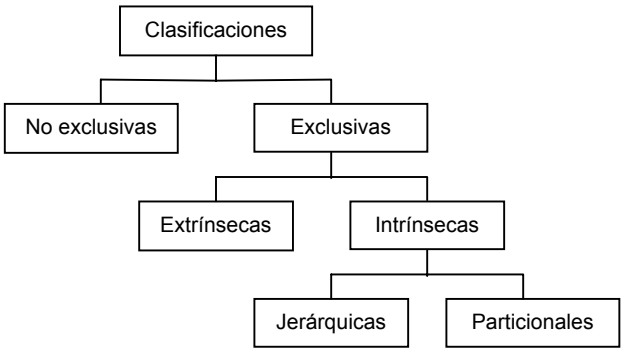

Figura 1 - División de las formas de clasificar objetos

objetos ya están clasificados y son utilizados por el algoritmo para aprender a clasificar a los demás.
- **Intrínsecas (no supervisadas)** : La clasificación se realiza en base a las características propias de los objetos, sin conocimiento previo sobre las clases a las que pertenecen.

- **Jerárquicas** : Los métodos jerárquicos consiguen la categorización 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 categorización de los documentos [17].
- **Particionales (no jerárquicas)** : Los métodos no jerárquicos, tambien llamados particionales, o de optimización, no producen una serie de grupos anidados, sino que llegan a una única categorización que optimiza un criterio predefinido o función objetivo [18].

La categorización automática de documentos se encuentra en la categoría “intrínseca”, ya que los criterios de agrupamiento se basan en la información contenida en los mismos para determinar sus similitudes

Algoritmos jerárquicos

Los algoritmos jerárquicos se caracterizan por generar una estructura de árbol (llamada “dendograma”), en la que cada nivel es un agrupamiento posible de los objetos de la colección [19]. Cada vértice (nodo) del árbol es un grupo de elementos. La raiz del árbol (primer nivel) se compone de un sólo grupo que contiene todos los elementos. Cada hoja del último nivel del árbol es un grupo compuesto por un sólo elemento (hay tantas hojas como objetos tenga la colección). En los niveles intermedios, cada nodo del nivel *n* es dividido para formar sus hijos del nivel *n* + 1.

Los algoritmos de agrupamiento jerárquicos fueron uno de los primeros enfoques para los problemas de categorización de documentos, y todavía se siguen utilizando debido a la forma simple e intuitiva en la que trabajan [20].

De acuerdo a la metodología que aplican para obtener el dendograma, los algoritmos jerárquicos pueden dividirse en *aglomerativos* y *divisivos* [21].

Los métodos aglomerativos parten de las hojas del árbol, ubicando a cada elemento en su propio grupo, y en cada paso buscan los dos grupos más cercanos para juntarlos. Los divisivos, por su parte, hacen el camino inverso. Comenzando en la raiz, en cada paso seleccionan un grupo para dividirlo en dos, buscando que el agrupamiento resultante sea el mejor de acuerdo a un criterio predeterminado.

Entre las características de éstos métodos pueden destacarse las siguientes:
- Su forma de trabajo es simple e intuitiva.

El enfoque utilizado por estos métodos es semejante al que utilizaría una persona para realizar la tarea del agrupamiento, especialmente los aglomerativos (comenzar juntando los documentos más similares entre si, y luego buscar similitudes entre los grupos).
- Su resultado es una serie de agrupamientos anidados

Esto facilita la revisión de los resultados por parte del usuario, que puede recorrer la estructura de árbol para ver agrupamientos con diferentes niveles de detalle [22].
- Son deterministas

Al aplicar dos veces un algoritmo jerárquico a una colección de documentos, las dos veces seguirá el mismo camino hacia la solución. Hay algunos agrupamientos que el algoritmo nunca considerará, sin importar la cantidad de veces que se lo ejecute [13].
- No revisan las decisiones que toman en los pasos anteriores

Una vez que dos documentos se han asignado al mismo grupo (o se han colocado en distintos grupos, en los divisivos), ningún paso posterior los volverá a separar (o juntar), por lo que una mala asignación en los primeros pasos no puede corregirse [14].
- Requieren grandes tiempos de cómputo

La forma de buscar en cada paso los grupos a unir (o dividir, en los divisivos), hacen que las implementaciones conocidas de estos algoritmos

tengan tiempos de ejecución del orden de n^2 ó n^3 [23].

Si bien se acepta que los métodos jerárquicos aglomerativos obtienen resultados de buena calidad [4] [14] [20], los tiempos de cómputo requeridos hacen que no sean una solución factible para la mayoría de las aplicaciones prácticas [20] [23].

Algoritmos particionales

Los métodos de optimización o particionales, a diferencia de los jerárquicos, no van generando distintos niveles de agrupamiento de los objetos, sino que trabajan en un sólo nivel, en el que se refina (optimiza), un agrupamiento [18].

Las técnicas de optimización se están comenzando a utilizar con más frecuencia en aplicaciones de categorización automática de documentos debido a requieren considerablemente menos recursos que los algoritmos jerárquicos [12].

Estos métodos asumen que el valor de *k* (la cantidad de grupos), está definida de antemano [16].

La estructura general de éstos métodos se compone de los siguientes pasos (adaptado de [21]):

1) Seleccionar *k* puntos representantes (cada punto representa un grupo de la solución).
2) Asignar cada elemento al grupo del representante más cercano, de forma de optimizar un determinado criterio.
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.

Existen numerosas variantes de algoritmos de optimización en la literatura [16] [19] [21] [24], que implementan la estructura básica de cuatro pasos descripta anteriormente.

Estos algoritmos son muy similares entre sí, por lo que se describirá únicamente el algoritmo “k-means”, que, además de ser el referente más típico, en la bibliografía es el que más frecuentemente se encuentra aplicado al campo de categorización automática de documentos [13].

Algoritmo K-Means

Este algoritmo, presentado originalmente por [25], utiliza a los centroides de cada grupo como sus puntos representantes.

Partiendo de una selección inicial de *k* centroides (que pueden ser *k* elementos de la colección seleccionados al azar, o los que se obtengan mediante la aplicación de alguna técnica de inicialización), cada uno de los elementos de la colección se asigna al grupo con el centroide más cercano.

A continuación, se calcula el centroide de cada uno de los grupos resultantes. En los primeros pasos se obtienen las mayores diferencias entre los centroides originales y los calculados luego de las reasignaciones.

Los puntos de la colección vuelven a asignarse al grupo del centroide más cercano, y estos pasos se repiten hasta que los *k* centroides no cambian luego de una iteración (ésto es equivalente a decir que el valor de la función utilizada como criterio de optimización no varía).

El algoritmo “k-means” encuentra una categorización que representa un óptimo local del criterio elegido [26].

Entre las características de éstos métodos pueden destacarse las siguientes:

- Pueden ser no deterministas.

Partiendo del mismo agrupamiento inicial, los métodos llegarán siempre a la misma solución. Sin embargo, casi todos los métodos para la selección inicial son no deterministas. El algoritmo evaluará diferentes agrupamientos cada vez que se lo ejecute, y (si los grupos no están claramente separados) podrá llegar a soluciones distintas [13].

- Pueden corregir errores cometidos en pasos anteriores

En cada paso del algoritmo los objetos de la colección se asignan al grupo más apropiado según el criterio de optimización. De esta manera, el algoritmo va refinando el agrupamiento en cada iteración [16].

- Pueden implementarse en forma eficiente

Las restricciones de recursos son la causa principal por la que se utilizan este tipo de métodos. La mayoría de las variantes pueden implementarse de forma que sus tiempos de ejecución sean del orden de *n* [21].

Si bien el algoritmo K-Means es mucho más eficiente que los algoritmos jerárquicos, es dependiente de la selección inicial de centroides [26]. Sus resultados pueden ser bastante pobres y suelen variar mucho si se aplica varias veces a la misma colección de documentos, ya que si la selección de centroides al azar es mala, la solución encontrada también lo será.

Algoritmo Bisecting K-Means

Varios investigadores han presentado alternativas a los algoritmos tradicionales de categorización automática de documentos [4] [13] [26].

El algoritmo “Bisecting K-Means” es una de ellas [13]. En este trabajo se decidió tomar esta variante como referencia para compararla con la solución propuesta, por varios motivos:

- El trabajo que la presenta es uno de los más recientes, (año 2000), y esta variante se analiza tambien en un trabajo del año 2001 [12].
- Uno de los autores (que participa en ambos trabajos) es una personalidad reconocida dentro del dominio del recupero de información y de la categorización automática, con más de 70 trabajos publicados, y varios desarrollos de software relativos al tema.
- Los trabajos demuestran que la calidad de los resultados supera tanto a los algoritmos jerárquicos aglomerativos como al “K-Means”, con tiempos de cómputo lineales con la cantidad de elementos a agrupar.

El algoritmo “Bisecting K-Means” sigue los siguientes pasos:

1) Colocar todos los documentos en un sólo grupo
2) Elegir el grupo más grande para dividirlo.
3) Dividir el grupo en 2 subgrupos usando el algoritmo “K-Means” (paso de bisección).
4) Repetir el paso de bisección 5 veces, y tomar la división que lleve a una mayor similitud promedio.
5) Repetir los pasos 2, 3 y 4 hasta alcanzar el número de grupos deseado.

Al utilizar el algoritmo “K-Means” para formar sólamente 2 grupos cada vez, y repetir 5 veces cada división, se intenta atenuar la dependencia de este algoritmo con la selección inicial de los representantes.

Algoritmo “Bisecting K-Means con refinamiento”

Esta variante consiste en refinar la solución obtenida con el algoritmo “Bisecting K-Means” utilizando el algoritmo “K-Means” estándar.

El agrupamiento generado por el algoritmo “Bisecting K-Means” se toma como el punto de partida para el algoritmo “K-Means”.

Este refinamiento apunta a corregir posibles asignaciones incorrectas de documentos a los grupos que se podrían haber realizado durante las fases de división del algoritmo. Los resultados muestran que el paso de refinamiento mejora la calidad de las soluciones obtenidas [13].

Aunque el algoritmo Bisecting K-Means ha demostrado obtener mejores soluciones que el algoritmo k-means y los métodos jerárquicos aglomerativos [13], en tiempos de cómputo lineales con la cantidad de documentos a agrupar, la forma en que explora el espacio de búsqueda es claramente sub-óptima.

Al momento de dividir un grupo, el algoritmo “Bisecting K-Means” realiza 5 divisiones diferentes del grupo (aplicando 5 veces el algoritmo k-means con k=2 a los elementos del grupo), y luego elige una de ellas (la que más hace crecer el criterio de optimización), descartando las demás divisiones que creó.

El siguiente análisis muestra por qué esta forma de trabajo no es eficiente:

- Cuando el grupo es fácilmente divisible (hay 2 subgrupos claramente definidos), las 5 corridas del algoritmo K-Means encontrarán la misma división, o divisiones con muy pocas variaciones. En este caso, el algoritmo estaría repitiendo 5 veces el mismo trabajo.
- Cuando no hay dos subgrupos claramente definidos, las 5 corridas del algoritmo K-Means encontrarán divisiones diferentes. En este caso, el problema es que el algoritmo Bisecting K-Means no tiene ningún mecanismo para aprovechar el trabajo realizado al armar las divisiones que son descartadas. Es muy posible que en alguna de ellas haya encontrado un grupo excelente y uno muy malo, que en promedio, hagan que descarte esa división, y no tiene ningún mecanismo que le permita quedarse con la parte buena y descartar sólo lo malo.

Algoritmos Genéticos

Los algoritmos genéticos fueron desarrollados por John Holland, junto a su equipo de investigación, en la universidad de Michigan en la década de 1970 [27].

Los algoritmos genéticos combinan las nociones de supervivencia del más apto con un intercambio estructurado y aleatorio de características entre individuos de una población de posibles soluciones, conformando un algoritmo de búsqueda que puede aplicarse para resolver problemas de optimización en diversos campos [28].

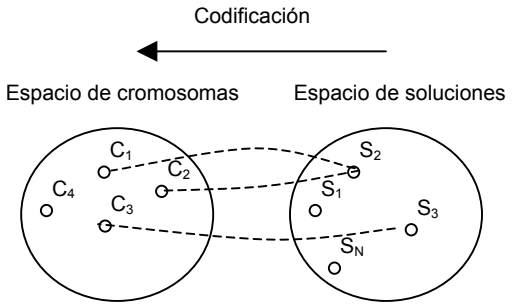

Figura 2 - Cada cromosoma codifica una posible solución al problema

Imitando la mecánica de la evolución biológica en la naturaleza, los algoritmos genéticos operan sobre una población compuesta de posibles soluciones al problema. Cada elemento de la población se denomina “cromosoma”.

Un cromosoma es el representante, dentro del algoritmo genético, de una posible solución al problema. La forma en que los cromosomas codifican a la solución se denomina “Representación”.

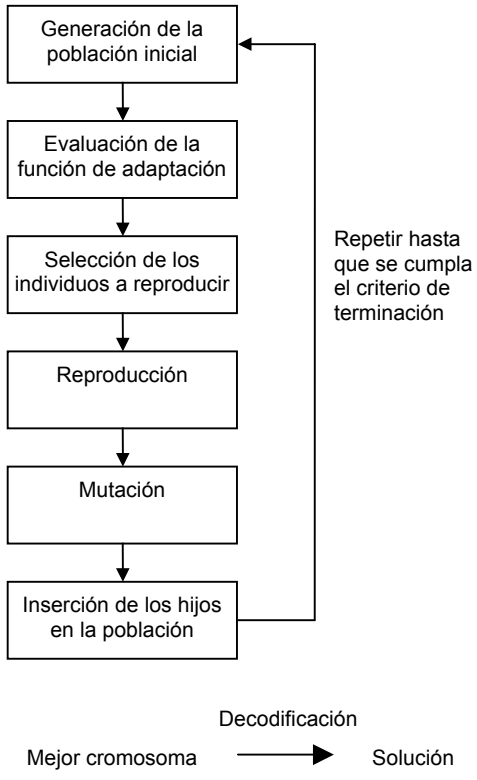

Figura 3 - Esquema de funcionamiento del algoritmo genético

El algoritmo genético va creando nuevas “generaciones” de esta población, 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, adaptados de la genética natural.

La figura 3 representa el esquema de funcionamiento del algoritmo genético.

El proceso comienza seleccionando un número de cromosomas para que conformen 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. La reproducción 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. Hay varios criterios que pueden utilizarse para elegir a los cromosomas que serán reemplazados.

El ciclo de selección, reproducción 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.

Trabajos previos sobre Algoritmos Genéticos aplicados a categorización automática

La aplicación de algoritmos genéticos al problema general de la categorización automática se ha investigado con interés [14] [19] [29] [30] [31] [32] [33] [34]. Sin embargo, son muy pocos los autores que los han aplicado a la categorización automática de documentos [35].

Esto se debe a que, pese a que los algoritmos genéticos han demostrado capacidad de explorar el espacio de búsqueda amplia y eficientemente [28] [33] [36], los resultados de las investigaciones no han dado buenos resultados al aplicarlos a la categorización automática en general, ni a la

categorización automática de documentos en particular.

La causa de estos fracasos radica en la aplicación inadecuada de los algoritmos genéticos al problema. Se ha hecho notar [33] [37] que los algoritmos genéticos no son una solución universal que pueda aplicarse en forma pura a cualquier problema, sino un paradigma que debe adaptarse a los problemas concretos a los que se aplica.

Los fracasos no se deben a que los algoritmos genéticos no puedan aplicarse en forma efectiva en el dominio de la categorización automática de documentos sino a la falta de una correcta adaptación del esquema de representación y los operadores del algoritmo genético a este problema.

Otro punto importante, ignorado por la mayoría de los autores, es la posibilidad de incorporar en los operadores del algoritmo genético conocimiento específico del problema al cual se está aplicando el mismo.

De esta manera, los operadores no trabajan en forma totalmente aleatoria sobre los individuos de la población, sino que tienden a realizar cambios que apunten a buenas soluciones para el problema a resolver.

Sin la aplicación de conocimiento específico del problema que se está resolviendo en los operadores, es casi imposible que el algoritmo genético supere a los algoritmos diseñados *ad hoc* para el problema [28] [37].

Metodología

En esta sección se describe el algoritmo génetico de categorización automática de documentos propuesto.

Este algoritmo presenta un nuevo operador de cruza y cuatro nuevos operadores de mutación diseñados específicamente para el problema de la categorización automática de documentos.

Para el diseño de los operadores se tuvieron en cuenta los siguientes puntos, basados en las cuestiones planteadas previamente:
- Los operadores deben ser eficientes
- Los operadores deben ayudar a explorar el espacio de búsqueda
- Los operadores deben aplicar información del dominio para ayudar al algoritmo a obtener soluciones de buena calidad

Representación

La representación elegida fue la Numeración de grupo. Este método es el más frecuentemente utilizado en la literatura [33].

Para elegir el método de representación se evaluaron los siguientes puntos:
- Que la representación fuera sencilla e intuitiva
- Que se pudiera implementar eficientemente la función de evaluación
- Que se pudieran implementar eficientemente los operadores

Estructura del cromosoma

La estructura del cromosoma utilizado en el algoritmo propuesto puede observarse en la figura 4.

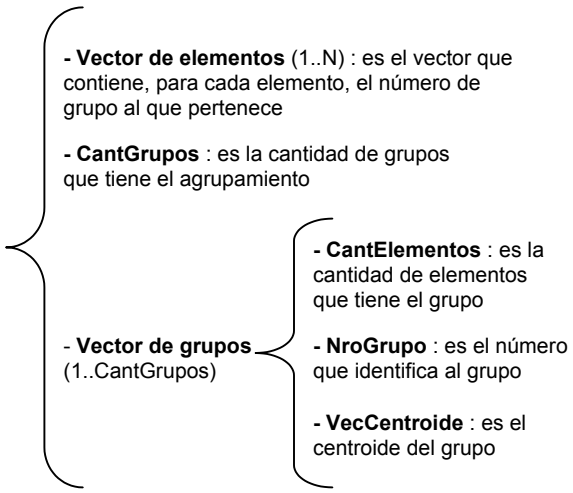

Figura 4 - Estructura del cromosoma

La cantidad de grupos (CantGrupos) y el Vector de grupos no son estrictamente necesarios, ya que todos los datos que contienen pueden obtenerse realizando operaciones a partir de la información del vector de elementos. Sin embargo, como estos datos se acceden muy frecuentemente, se incluyen en el cromosoma por razones de eficiencia.

Generación de la población inicial

Para seleccionar el método de generación de la población inicial se hicieron diversas pruebas y se extrajeron las siguientes observaciones:
- El tamaño de la población requerido para que el algoritmo genético encuentre buenas soluciones es más chico cuanto mejores sean los agrupamientos de la población inicial.
- Generar la población inicial al azar es lo más rápido, pero los agrupamientos generados son de muy baja calidad, y lleva demasiadas generaciones al algoritmo genético llegar a buenas soluciones.
- Utilizando el algoritmo k-means (el más rápido de los algoritmos conocidos que dan soluciones

aceptables) se obtienen buenas soluciones en relativamente pocas generaciones, pero el proceso de generación de la población inicial exige demasiado tiempo (aún teniendo en cuenta que el tamaño de la población inicial puede reducirse), ya que el algoritmo k-means es del orden de (n.k).

Buscando una solución intermedia, se decidió aplicar el algoritmo k-means para generar los agrupamientos de la población inicial, pero en lugar de formar *k* grupos, se generan agrupamientos de log(*k*) grupos. De esta forma, la generación de la población inicial lleva tiempos del orden de (n.log(*k*)).

El algoritmo genético tiene operadores que se ocupan de ir dividiendo a los grupos para llegar a soluciones que tengan *k* grupos, que se detallan más adelante.

Función de adaptación

Para el algoritmo propuesto se eligió la Similitud promedio del agrupamiento como función de adaptación. Esta función se detalla en la sección dedicada a los resultados experimentales.

Selección

Para el algoritmo propuesto se utiliza el método de Torneo, por los siguientes motivos:
- Es independiente de la escala de la función de adaptación
- No requiere ordenar los valores de adaptación de cada cromosoma
- Permite regular la presión de selección

Cruza

Dado que ninguno de los operadores de cruza encontrados en la literatura daba solución a las cuestiones planteadas, se diseñó un nuevo operador de cruza que cumpliera con las características buscadas.

El operador fue diseñado con los siguientes lineamientos:
- La implementación debía ser eficiente
- El operador debía ser sensible al contexto
- El operador debía aplicar información específica del dominio para generar agrupamientos de buena calidad

Cruza Pasa Grupo

El operador diseñado se denomina “Cruza Pasa Grupos”.

El operador genera 2 hijos a partir de los padres. Primero asigna un rol a cada padre para generar el primer hijo, y luego invierte los roles para generar el otro.

La figura 5 muestra el esquema básico de funcionamiento del operador para generar uno de los hijos.

El agrupamiento de uno de los padres es copiado al hijo sin modificaciones, y luego uno de los grupos del segundo padre se pasa al hijo, reacomodando los elementos que hagan falta.

Los pasos que sigue el algoritmo para generar cada uno de los hijos son:
- Crear un cromosoma con el mismo agrupamiento que el primer padre (el padre que pasa el agrupamiento)
- Seleccionar un grupo fuente del segundo padre (el padre que pasa el grupo)
- Seleccionar un grupo destino en el hijo, que va a transformarse en el grupo del padre
- Reacomodar los elementos del hijo según sea necesario para que el grupo destino contenga los elementos que contiene el grupo fuente

Mutación

Durante las pruebas realizadas se observó que los operadores de mutación clásicos retrasaban considerablemente la convergencia del algoritmo genético, ya que lo obligaban a explorar regiones del espacio de búsqueda al azar, donde la mayor parte de las veces no se encontraban mejores soluciones.

Al diseñar un algoritmo genético que pueda competir en términos de eficiencia con los algoritmos específicos del dominio, debe sacrificarse en parte la amplitud de la exploración, restringiéndola a aquellas regiones donde es más probable que haya buenas soluciones.

Esto puede lograrse diseñando operadores de mutación que no hagan cambios totalmente aleatorios, sino que, de la misma forma que el operador de cruza, tiendan a hacer cambios que aumenten la calidad de las soluciones, dejando algunas pocas decisiones libradas al azar.

Estos operadores se aplicarán luego con probabilidades bastante altas comparadas con las de los operadores de mutación estándar, ya que son necesarios no sólo para la exploración del espacio de búsqueda, sino tambien para el aumento de la calidad de la población.

A continuación se describen los 4 nuevos operadores de mutación diseñados para el algoritmo propuesto.

Mutación RefinarKM

Este operador de mutación se aplica con una probabilidad que varía durante la ejecución del algoritmo genético entre los valores 0,1 y 0,25.

Cuando se muta un cromosoma mediante este operador, se aplica al agrupamiento lo que se denomina “Refinamiento ligero”, que es una modificación del algoritmo k-means.

El refinamiento ligero consiste en una sóla iteración del algoritmo k-means. Pero para cada elemento, en lugar de buscar entre los *k* grupos del agrupamiento aquel cuyo centroide sea más cercano al del elemento, la búsqueda se realiza entre 2**log(k)* grupos seleccionados al azar. De esta forma, en lugar de ser del orden de (n.k), la aplicación del operador tiene una complejidad del orden de (n.2.log(k)).

El operador cumple con el doble propósito de mejorar la calidad de la solución, pero al mismo tiempo incluir el azar en las decisiones para ampliar la exploración del espacio de búsqueda.

Mutación Refinar Selectivo

Este operador de mutación se aplica con una probabilidad que varía durante la ejecución del algoritmo genético entre los valores 0 y 0,3.

Al aplicar este operador a un cromosoma, se intenta mejorar la calidad de los mejores grupos del agrupamiento.

Lo que se busca es que el agrupamiento mutado tenga algunos grupos de muy buena calidad que luego, si el cromosoma es seleccionado como padre, pasen al hijo.

El operador identifica los grupos con mayor y menor similitud promedio y luego recorre los mejores grupos en busca de los elementos que están más alejados del centroide del grupo. Esos elementos son removidos del grupo (lo que aumenta su similitud promedio), y son reasignados a alguno de los grupos con menor similitud promedio (lo que disminuye aún más su similitud promedio).

El cromosoma resultante luego de la mutación puede no ser una buena solución, pero va a tener algunos grupos de muy buena calidad que podrán pasar a sus hijos en caso de ser seleccionado para la cruza.

Mutación Split

Este operador de mutación se aplica con probabilidad 1 a todos los cromosomas que tengan menos de *k* grupos, y con una probabilidad que varía durante la ejecución del algoritmo genético entre los valores 0,05 y 0,15 a los cromosomas que tengan *k* grupos.

Como en la generación de la población inicial los agrupamientos se generan con log(k) grupos, este operador es necesario para que los agrupamientos lleguen a tener *k* grupos.

El operador selecciona un grupo y lo divide en 2. Si al cromosoma se le aplicó anteriormente el operador de mutación Join (que se describe más adelante), el grupo formado por este operador es el que se divide. En caso contrario, el grupo a dividir puede ser seleccionado con alguno de los siguientes criterios:

- Mayor tamaño (probabilidad 0,90) : se selecciona el grupo con la mayor cantidad de elementos.
- Menor similitud promedio (probabilidad 0,05) : se selecciona el grupo con la menor similitud promedio.
- Al azar (probabilidad 0,05) : el grupo a dividir es seleccionado al azar.

Una vez que se tiene el grupo a dividir, se lo divide en 2 aplicando a los elementos del grupo la primera iteración (fase inicial) del algoritmo k-means, con k = 2.

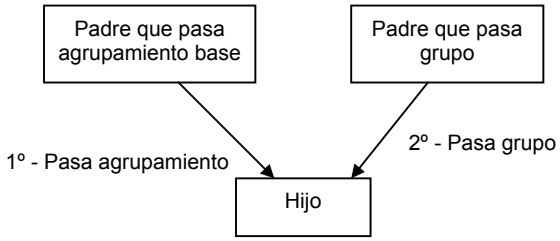

Figura 5 - Esquema básico del funcionamiento del operador Cruza Pasa Grupo

Mutación Join

Este operador de mutación se aplica con probabilidad 1 a todos los cromosomas que tengan más de *k* grupos, y con una probabilidad que varía durante la ejecución del algoritmo genético entre los valores 0,05 y 0,15 a los cromosomas que tengan *k* grupos.

La aplicación de este operador a un cromosoma que tenga *k* grupos hará que en la iteración siguiente se aplique el operador de mutación Split (ya que el agrupamiento tendrá *k-1* grupos).

El operador apunta a juntar en un solo grupo dos grupos de mediana o mala calidad, con la esperanza de que el operador de mutación Split pueda armar grupos de mejor calidad, mejorando la solución.

El operador elige dos grupos del agrupamiento, y junta en un sólo grupo los elementos de ambos.

El primero de los grupos puede ser seleccionado con alguno de estos criterios:

- Menor tamaño (probabilidad 0,7) : se selecciona el grupo con menor cantidad de elementos.

- Menor similitud promedio (probabilidad 0,3) : se selecciona el grupo con menor similitud promedio.

El otro grupo se puede seleccionar con alguno de estos criterios:

- Más similar (probabilidad 0,95) : se selecciona el grupo con el centroide más cercano al del primer grupo elegido.
- Al azar (probabilidad 0,05) : el segundo grupo se selecciona al azar.

Inserción de los hijos en la población

Se utiliza el método de torneo invertido. Cada vez que se realiza la reproducción, se busca mediante éste método dos cromosomas de la población que son reemplazados por los hijos recién generados.

Tamaño de la población

Los tamaños de población utilizados en trabajos previos varían entre valores de 40 a 1000 individuos [14]. Se ha hecho notar [37] que tamaños de población excesivamente grandes retrasan la convergencia del algoritmo genético, impidiéndole competir con otro tipo de métodos.

En las pruebas realizadas se observó que partiendo de una población de individuos de calidad medianamente aceptable, y utilizando los operadores ya descriptos, al agrandar el tamaño de la población se aumentaba la cantidad de generaciones requeridas para obtener buenas soluciones, sin obtener mejoras apreciables en la calidad de las mismas.

En el algoritmo propuesto, se comienza con una población inicial de 12 individuos, que luego disminuye a 10 cuando se han completado el 40% de las generaciones previstas.

Criterio de terminación

En las pruebas realizadas se observó que la calidad de la mejor solución aumentaba con cierto ritmo hasta un punto, para luego crecer muy lentamente hasta alcanzar la convergencia. Esto es, una vez que se llegaba a una buena solución, el algoritmo requería una cantidad importante de generaciones hasta que todos los cromosomas alcanzaban niveles cercanos de la función de adaptación.

Para evitar esa espera, se decidió terminar la ejecución del algoritmo genético luego de una cantidad fija de iteraciones.

Experimentalmente se encontró el valor GEN_MAX = (N / 20) + 25. Esto es, dada la cantidad de documentos a categorizar, se calcula la cantidad de iteraciones que debe ejecutar el algoritmo genético.

Luego de llegar al número de generaciones máximo, se selecciona como solución al cromosoma con el mayor valor de la función de adaptación.

Algoritmo Genetico con Refinamiento

Al igual que en el caso del algoritmo “Bisecting K-Means con refinamiento”, esta variante consiste en refinar la solución obtenida utilizando el algoritmo “K-Means” estándar.

Resultados

Para la realización de los experimentos, se utilizaron datos de la Colección de Prueba para Categorización de Documentos “Reuters 21578” [38].

Ésta colección 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 [12] [13] [39] [40].

Muchos de los cálculos que ayudan a medir la calidad de un agrupamiento requieren que cada documento tenga solamente una categoría asignada, por lo que para los experimentos de este trabajo se decidió utilizar dos subconjuntos extraidos de esta colección previamente utilizados por otros investigadores [12]. El procedimiento seguido para extraer los subconjuntos fue tomar solamente los documentos que tenían sólamente un valor en el campo “Tema”, luego dividir los posibles valores del campo en dos conjuntos y asignar a cada documento al conjunto correspondiente. Esto dió lugar a dos conjuntos de datos (re0 y re1), de 1504 y 1657 documentos respectivamente.

Variables a observar

Una vez que se obtienen los agrupamientos de documentos que arroja cada algoritmo como salida, debe medirse cuantitativamente la calidad de cada uno para poder compararlos.

Similitud promedio

Esta es una medida “interna”, ya que puede evaluarse sin conocimentos externos (no hace falta conocer la categorización “real” de los documentos).

La similitud promedio de un grupo es la que surge de promediar la similitud existente entre todos los documentos del grupo tomados de a pares. Por ejemplo, para el grupo j, que tiene n_j elementos, utilizando el coeficiente del coseno extendido como medida de similitud,

$$sim\_prom_j = \frac{1}{n_j{}^2}\left(\sum_{\substack{i=1\\j=1}}^{n_j}\frac{d_i \bullet d_j}{\|d_i\| * \|d_j\|}\right) \quad (4)$$

La similitud promedio de todo el agrupamiento se obtiene sumando la similitud promedio de cada grupo multiplicada por la cantidad de elementos del grupo, y dividiendo el total por la cantidad total de elementos de la muestra,

$$sim\_prom = \frac{\sum_{j=i}^{k} n_j * sim\_prom_j}{N} \quad (5)$$

Esta medida da cuenta de cuánto tienen en común los elementos de cada grupo. Cuanto más homogeneo sea cada grupo, mayor valor tendrá este parámetro. Valores más grandes de similitud promedio indican una mejor calidad del agrupamiento.

Entropía

La entropía es una medida "externa" (para calcularla es necesario conocer a qué categoría "real" pertenece cada documento).

El concepto de entropía tiene su origen en el campo de la física (más especificamente, en el area de la termodinámica), y es utilizada como medida del grado de desorden de un sistema [41]. En 1948, Shannon [42] incorpora el término a la teoría de la información como una función que mide la cantidad de información generada por un proceso.

Para medir la calidad de un agrupamiento utilizando la entropía, primero se calcula entropía de cada grupo. Para cada categoría "real" *i*, se calcula la probabilidad p_{ij} de que un miembro del grupo *j* sea de la categoría *i*,

$$p_{ij} = n_{ij} \Big/ n_j \quad (6)$$

Donde n_{ij} es la cantidad de documentos de la categoría *i* que se encuentran en el grupo *j* y n_j es la cantidad de documentos del grupo *j*.

La entropía de cada grupo se calcula usando la fórmula

$$H_j = -\sum_i p_{ij} * \log(p_{ij}) \quad (7)$$

Donde *i* recorre todas las categorías "reales" de los documentos.

La entropía total del agrupamiento se calcula luego ponderando la entropía de cada grupo de acuerdo a su tamaño,

$$H = \frac{\sum_{j=1}^{k} n_j * H_j}{N} \quad (8)$$

La entropía tendrá un valor máximo cuando los documentos de cada categoría estén distribuidos uniformemente entre los grupos, y un valor igual a 0 cuando cada categoría tenga sus documentos en un sólo grupo. Valores más chicos de entropía indican una mejor calidad del agrupamiento.

Cantidad de operaciones

Esta medida no se relaciona con la calidad del agrupamiento obtenido, sino con la eficiencia del algoritmo.

La cantidad de operaciones da una medida de la complejidad computacional del algoritmo [43], ya que mide el esfuerzo computacional que llevó al algoritmo obtener el agrupamiento, en una escala que es independiente de los recursos de hardware sobre los cuales se realizaron las pruebas (velocidad del procesador, memoria, velocidad del disco, etc).

Las tres operaciones básicas que se repiten en los algoritmos evaluados son:

- Multiplicación de 2 vectores (para calcular la similitud entre los documentos)
- Cálculo de la norma de un vector (para calcular la similitud promedio de un grupo se calcula la norma del centroide)
- Sumas de vectores (cuando se cambia un elemento de un grupo a otro deben actualizarse los centroides).

Para simplificar las mediciones, se supuso que las 3 operaciones eran equivalentes (multiplicar dos vectores lleva el mismo esfuerzo computacional que sumar dos vectores o calcular la norma de un vector). Por la naturaleza de las operaciones (todas iteran sobre los elementos de un vector realizando operaciones algebraicas), es una aproximación razonable. De todas maneras, alrededor del 90% de las operaciones que realizan los algoritmos son multiplicaciones de vectores, por lo que el error es despreciable.

Lógicamente, un menor número de operaciones realizadas (para resultados similares) indica una mayor eficiencia del algoritmo.

Forma de realización de los experimentos

La forma de realizar los experimentos consistió en tomar 20 muestras de datos con una cantidad de documentos fija (500 documentos), y realizar corridas de los algoritmos variando la cantidad de grupos a formar.

Para seguir el comportamiento de cada una de las variables independientes al ir variando la cantidad de grupos, se confeccionaron gráficos con los promedios generales para cada algoritmo.

El eje "x" representa la cantidad de grupos en que se fueron dividiendo las muestras de datos con los algoritmos.

El eje "y" representa la variable independiente que se mide en cada gráfico. El título del gráfico lleva el nombre de la variable independiente.

Adicionalmente, se realizó un análisis estadístico (mediante el test de Wilcoxon para la comparación de medias de muestras apareadas) sobre los resultados experimentales, para medir el nivel de confianza de las conclusiones extraídas.

La figura 6 muestra la similitud promedio de los agrupamientos encontrados en función de la cantidad de grupos armados.

La curva de resultados del algoritmo "Genético" se encuentra ligeramente por encima de la curva del algoritmo "Bisecting K-Means con refinamiento", mientras que la curva para el algoritmo "Genético con refinamiento" supera claramente a la del algoritmo "Bisecting K-Means con refinamiento".

Esta apreciación es consistente con los resultados que arroja el test de Wilcoxon para esta variable. Tomando en cuenta los resultados de las 20 muestras, el test permite afirmar con un margen de confianza del 95% que el promedio de valores de similitud promedio para el algoritmo "Bisecting K-Means con refinamiento" es inferior al de los otros 2 algoritmos.

La figura 7 muestra la entropía de los agrupamientos encontrados en función de la cantidad de grupos armados.

Las curvas de resultados de los algoritmos "Genético" y "Bisecting K-Means con refinamiento" son bastante similares, y tienen diferencias en uno y otro sentido que no permiten afirmar que una sea mejor que la otra, mientras que la curva para el algoritmo "Genético con refinamiento" está claramente por debajo de ambas.

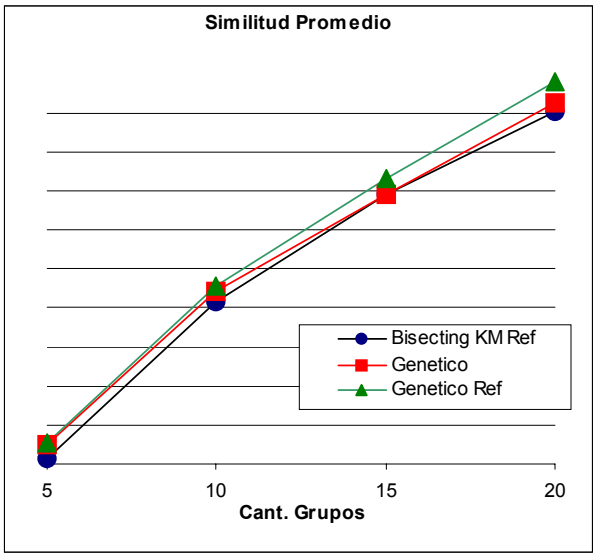

Figura 6 - Similitud promedio de los agrupamientos encontrados en función de la cantidad de grupos.

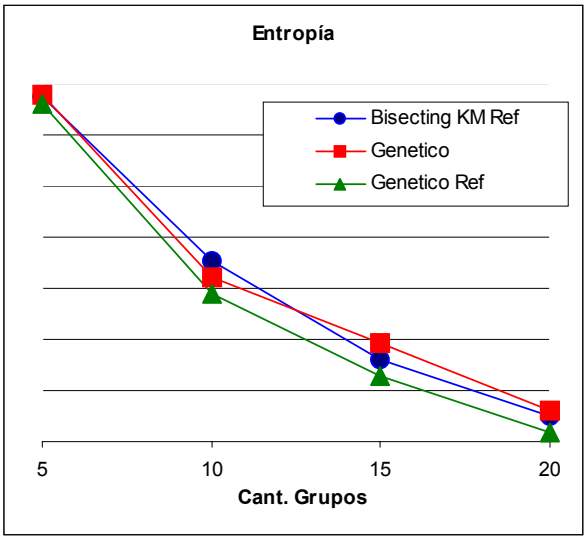

Figura 7 - Entropía de los agrupamientos encontrados en función de la cantidad de grupos.

Esta apreciación es consistente con los resultados que arroja el test de Wilcoxon para esta variable. Tomando en cuenta los resultados de las 20 muestras, el test no puede distinguir si los valores de entropía son diferentes para los algoritmos "Genético" y "Bisecting K-Means con refinamiento", mientras que permite afirmar con un grado de confianza de 95% que los valores para el algoritmo "Genético con refinamiento" son menores que para el algoritmo "Bisecting K-Means con refinamiento".

La figura 8 muestra la cantidad de operaciones realizada por los algoritmos en función de la cantidad de grupos armados.

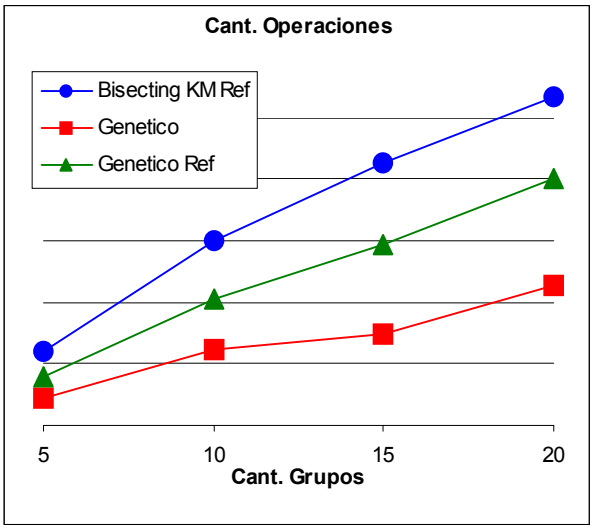

Figura 8 -Cantidad de operaciones realizadas por los algoritmos en función de la cantidad de grupos.

Se ve claramente que los resultados para el algoritmo “Genético” están muy por debajo que los del algoritmo “Bisecting K-Means con refinamiento”, mientras que para el algoritmo “Genético con refinamiento” se encuentran en una posición intermedia, tambien claramente por debajo del algoritmo “Bisecting K-Means con refinamiento”.

Esta apreciación es consistente con los resultados que arroja el test de Wilcoxon para esta variable. Tomando en cuenta los resultados de las 20 muestras, el test permite afirmar con un margen de confianza del 95% que el promedio de operaciones realizadas por el algoritmo “Bisecting K-Means con refinamiento” es superior en más de un 40% a las realizadas por el algoritmo “Genético” y en más de un 15% a las realizadas por el algoritmo “Bisecting K-Means con refinamiento”.

Conclusiones

Este trabajo propone una adaptación de un algoritmo genético al problema de la categorización automática de documentos. Se proponen 5 nuevos operadores (1 de cruza y 4 de mutación) que aplican información del dominio para mejorar las soluciones obtenidas por el algoritmo genético durante el proceso de evolución.

La investigación y los resultados obtenidos confirman que los algoritmos genéticos son una poderosa herramienta para la resolución de problemas en los cuales el espacio de soluciones es amplio y la función de optimización es compleja.

Se ha encontrado tambien que, tal como lo han afirmado otros autores [33] [37], los algoritmos genéticos no son un método de solución universal de problemas, sino un paradigma que debe adaptarse correctamente al problema a resolver. El algoritmo propuesto logra resultados efectivos porque en el diseño del mismo se han adaptado los conceptos que aplican los algoritmos genéticos y se han creado nuevos operadores específicos para el problema a resolver.

Los resultados obtenidos muestran que el algoritmo genético propuesto encuentra soluciones de mejor calidad que el algoritmo “Bisecting K-Means con refinamiento”, requiriendo una menor cantidad de operaciones para lograrlo.

La explicación a este hecho debe buscarse en la forma que tienen de explorar el espacio de posibles agrupamientos tanto el algoritmo “Bisecting K-Means” como el algoritmo genético propuesto.

Se ha detallado previamente por qué la forma de explorar el espacio de soluciones del algoritmo “Bisecting K-Means” es subóptima.

El algoritmo “Genético”, al crear nuevos grupos (o modificar los existentes) con cualquiera de sus operadores, no descarta de inmediato ninguno de ellos, aunque sean malas soluciones. Los malos agrupamientos se van eliminando más tarde mediante el operador de selección, pero mientras tanto, esos agrupamientos pueden ser seleccionados para su cruza con otros, momento en el cual tienen la oportunidad de pasar alguna de sus características positivas a uno de sus hijos.

Es el operador de cruza el que permite al algoritmo “Genético” obtener su solución en forma más eficiente, ya que puede aprovechar las buenas características de cada una de las variaciones que se fue realizando, por lo que prácticamente ningún trabajo es desperdiciado.

Otro punto clave es que el algoritmo “Bisecting K-Means” no puede llegar a ninguna solución que le obligue a pasar por un punto en el que disminuya el criterio de optimización (siempre elige la división que más hace crecer el criterio de optimización). Esto hace que haya regiones del espacio de búsqueda a las que le pueda resultar difícil llegar.

Por otra parte, el algoritmo “Genético” es capaz de generar soluciones de menor calidad como parte del proceso, y extraer características positivas de ellas. De esta manera, el algoritmo genético nunca se encuentra restringido a una región del espacio de búsqueda. Los elementos de azar que intervienen en la cruza y la mutación pueden llegar a

generar agrupamientos en cualquier punto del espacio de búsqueda.

Bibliografía

1. Kaufmann, Leonard y Rousseeuw, Peter J. (1990). *Finding Groups in data: An introduction to Cluster Analysis*, John Wiley & Sons, Inc., NY.
2. ISO/IEC 2382-1:1993 *Information technology -- Vocabulary* --. Part 1: Fundamental terms
3. Van Rijsbergen, C. J. (1979). *Information Retrieval*, Butterworths, London, 2da edición.
4. Cutting, D. R., Karger, D. R., Pedersen, J. O. 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.
5. Schütze, Hinrich y Silverstein Craig (1997) *Projections for Efficient Document Clustering*, in Proceedings of ACM/SIGIR'97, pp.74-81.
6. 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.
7. Dunlop, M.D. y Van Rijsbergen, C. J. (1991). *Hypermedia and free text retrieval*. RIA091 Conference, Barcelona.
8. Faloutsos, Christos y Oard, Douglas W. (1995). *A survey of Information Retrieval and Filtering Methods*. Technical Report CS-TR3514, Dept. of Computer Science, Univ. of Maryland.
9. Croft, W. B. (1978). *Organizing and searching large files of documents*, Ph.D. Thesis, University of Cambridge.
10. 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.
11. 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.
12. Zhao, Y. y Karypis, G., (2001). *Criterion Functions for Document Clustering*. Technical Report #01-40, Department of Computer Science, University of Minnesota.
13. Steinbach, M., Karypis, G., y Kumar, V. (2000). *A comparison of Document Clustering Techniques*. Technical Report #00-034. University of Minnesota. In KDD Workshop on Text Mining.
14. Cole, Rowena M. (1998). *Clustering with Genetic Algorithms*. Thesis for the degree of Master of Science, Department of Computer Science, University of Western Australia.
15. Strehl, A., Ghosh, J. y Mooney, R. (2000). *Impact of Similarity Measures on Web-page Clustering*. AAAI-2000: Workshop of Artificial Inteligence for Web Search.
16. Qin He, (1996). *A review of clustering algorithms as applied in IR*, UIUCLIS-1996/6+IGR, University of Illinois at Urbana-Champaign.
17. Willet, P. (1998). *Recent trends in hierarchical document clustering: a critical review*. Information Processing and Management. 24:577-97.
18. Everitt, Brian S. (1993). *Cluster analysis.* Halsted Press, 3ra edición.
19. Jain, A. K., Murty, M.N., y Flinn, P.J. (1999). *Data Clustering: A review*. ACM Computing Surveys, Vol. 31, Nro 3, Septiembre 1999.
20. 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.
21. 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.
22. Maarek, Yoelle S., Fagin, Ronald, Ben-Shaul, Israel Z. y Pelleg, Dan. (2000). *Ephemeral Document Clustering for Web Applications*. IBM Research Report RJ 10186.
23. Zamir, Oren y Etzioni, Oren. (1998). *Web Document Clustering: A feasibility demonstration*. Proceedings of ACM/SIGIR'98.
24. Rasmussen, E. (1992). *Clustering Algorithms*, W. B. Frakes and R. Baeza-Yates, editors, Information Retrieval, Páginas 419-442. Prentice Hall, Eaglewood Cliffs, N. J.
25. McQueen, J. (1967). *Some methods for classification and analysis of multivariate observations*, 5-th Berkeley Symposium on mathematics, Statistics and Probability, 1, S. 281-298.
26. 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
27. Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

28. Goldberg, David E. (1989). *Genetic Algorithms - in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc.
29. Jones, D. R. y Beltramo, M. A. (1991) *Solving partitioning problems with genetic algorithms*. Proceedings of the fourth International Conference on Genetic Algorithms, pages 442-449.
30. Bezdeck, J. C., Boggavaparu, S., Hall, L. O. y Bensaid, A. (1994). *Genetic algorithm guided clustering*, in Proc. of the First IEEE Conference on Evolutionary Computation, 3440.
31. Estivill-Castro, V. y Murray, A. (1997). *Spatial Clustering for Data Mining with Genetic Algorithms.* FIT, Technical Report, 97-10.
32. Hall, L.O., Ozyurt, B. y Bezdek, J.C. (1999). *Clustering with a genetically optimized approach*. IEEE Trans. On Evolutionaty Computation, 3, 2, 103-112.
33. Falkenauer, Emanuel. (1999). *Evolutionary Algorithms: Applying Genetic Algorithms to Real-World Problems*. Springer, New York, Pag 65--88.
34. Painho, M. y Bação, F. (2000). *Using Genetic Algorithms in Clustering Problems*. Proceedings of the 5th International Conference on GeoComputation, University of Greenwich, United Kingdom.
35. Jones, G., Robertson, A.M., Santimetvirul, C. y Willet, P. (1995). *Non-hierarchical document clustering using a genetic algorithm*. Information Research, an electronic journal, Vol 1, No 1, April, 1995.
36. Koza, John R. (1997). *Genetic Programming*. Cambridge : M.I.T. Press.
37. Estivill-Castro, V. (2000). *Hybrid Genetic Algorithms Are Better for Spatial Clustering.* Pacific Rim International Conference on Artificial Intelligence, pages 424-434.
38. Lewis, D. (1997). *Reuters-21578 text categorization test collection*, http://www.daviddlewis.com/resources/testcollections/reuters21578/ ó http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.
39. Joachims, T. (1998). *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Proceedings of ECML-98, 10th European Conference on Machine Learning, Springer Verlag, Heidelberg, DE.
40. Yang, Y. y Liu, Xin, (1999). *A re-examination of text categorization methods*. 22nd Annual International SIGIR.
41. Carter, T. (2000). *An introduction to information theory and entropy*. Complex Systems Summer School.
42. Shannon, C.E. (1948) *A mathematical theory of communication*, Bell System Technical Journal, vol. 27, pp. 379-423 and 623-656, July and October.
43. Wilf, H.S. (1986). *Algorithms and Complexity*, Prentice Hall.