

# Retroalimentación por relevancia: nueva perspectiva desde la programación evolutiva.

**Cristina López-Pujalte, Vicente P. Guerrero Bote**

Facultad de Biblioteconomía y Documentación, Universidad de Extremadura.

Alcazaba s/n, 06071 Badajoz.

E-mail: clopez@alcazaba.unex.es, vicente@alcazaba.unex.es

**Félix de Moya Anegón**

Facultad de Biblioteconomía y Documentación, Universidad de Granada.

Colegio Máximo, Campus Cartuja, 18071 Granada.

E-mail: felix@goliat.ugr.es

**Resumen:** La *Retroalimentación por Relevancia* es la más popular de las estrategias de modificación de consultas. Este proceso, introducido en los años 60, es un proceso controlado y automático de definición de consultas, que es sencillo de utilizar y extraordinariamente efectivo. Numerosos investigadores (Salton, Rocchio, Ide, Van Rijsbergen, Robertson, Sparck Jones, Fox, etc.) a lo largo de todos estos años, la han abordado desde distintos puntos de vista y se ha incorporado en famosos prototipos de Sistemas de Recuperación de Información (SRI), como por ejemplo el sistema *SMART*. Sin embargo, en los últimos años, han empezado a aparecer estudios sobre esta técnica de optimización de consultas que la implementan mediante algoritmos evolutivos. Este trabajo, fruto de anteriores investigaciones, desarrolla la retroalimentación por relevancia desde esta nueva perspectiva, presentando el *Algoritmo Genético* óptimo hallado para dicha técnica que mejores resultados obtuvo, superando incluso al más eficiente de los métodos clásicos (método de *Ide dec-hi*).

## 1. Introducción

En las operaciones de Recuperación de Información (RI) la mayoría de los usuarios, los cuales no conocen los detalles de la estructura de la colección y del entorno de recuperación, encuentran difícil formular una pregunta bien diseñada para el propósito de recuperación que tienen. Esto sugiere que la primera operación de recuperación debe verse como un tanteo, como una ejecución de prueba solamente, cuyo objetivo es recuperar algunos elementos útiles de la colección dada. Estos elementos inicialmente recuperados pueden ser examinados entonces para ver su relevancia y puede entonces construirse una nueva y mejorada definición de la consulta con la aspiración de recuperar elementos adicionales útiles en las siguientes búsquedas.

La *retroalimentación por relevancia* es una de las más populares estrategias de modificación automática de consultas. La idea principal consiste en una vez identificados ciertos documentos previamente recuperados como relevantes o irrelevantes por el usuario,

utilizar la información que proporcionan para adaptar la pregunta, de forma que se recuperen más documentos como los primeros y menos como los segundos (Salton y Buckley, 1990).

El efecto de este proceso de alteración de la pregunta es el de "*mover*" la consulta en la dirección de los documentos relevantes y alejarla de los no relevantes, con la esperanza de recuperar así más documentos deseados y menos documentos no deseados en una búsqueda posterior (Figura 1).

Esta técnica, introducida en los años sesenta, ha sido estudiada por numerosos investigadores hasta nuestros días, ya que se trata de un proceso extraordinariamente eficaz, capaz de mejorar en gran medida el comportamiento de un SRI, pero es en los últimos años cuando han aparecido estudios que unen este concepto a lo que conocemos como *Soft Computing*; así, han aparecido trabajos que implementan la retroalimentación por relevancia aplicando lógica difusa o/y programación evolutiva (Yang y Korfhage, 1994; Robertson y Willet, 1996;

Chen et al., 1998a y b; Horng y Yeh, 2000, López-Pujalte et al., 2002a, b y c; Kraft et al., 1997; Martín-Bautista et al., 1999; Córdón et al., 2000 y 2002).

La mayoría de dichos trabajos siguen el modelo vectorial y emplean *algoritmos genéticos* (a partir de ahora AGs) para implementar la técnica de retroalimentación.

Esto no nos debe extrañar fundamentalmente por dos razones: por un lado, los espacios documentales derivados de la aplicación del modelo vectorial son espacios reales de grandes dimensiones donde se pueden utilizar AGs para su exploración en busca de mejores soluciones, ya que estos algoritmos han demostrado ser especialmente aptos para explorar espacios de esas características. Por otro lado, los AGs ya habían sido utilizados anteriormente para resolver problemas en los que el entorno proporcionaba una retroalimentación. En concreto eran utilizados para ajustar los parámetros, por ejemplo en simulaciones de yacimientos de petróleo, análisis de mercado, clasificación, etc.

## 2. Algoritmo Genético para retroalimentación por relevancia.

Todas las aplicaciones de AGs sobre retroalimentación existentes en la literatura tienen en común que han utilizado un AG muy básico, prácticamente el algoritmo clásico, con pocas alternativas. La principal diferencia entre ellas radica en la función de adaptación utilizada, y es precisamente ésta la que consigue que el AG alcance el éxito o el más rotundo de los fracasos. Desgraciadamente la mayoría adolecen de la comparación con algún método clásico de retroalimentación por relevancia, por lo que no se puede estimar si se ha producido mejora, y en caso afirmativo cuánta, con el empleo de las técnicas genéticas.

Como sabemos, un AG clásico corresponde al algoritmo que ilustra la Figura 2: durante la iteración (generación)  $t$  el algoritmo genético cuenta con una población de soluciones potenciales  $P(t)$  (los cromosomas o vectores). Cada cromosoma se evalúa por medio de la función de adaptación para medir su idoneidad; entonces se seleccionan los individuos más idóneos para formar parte de la población de reproducción. Algunos de los miembros de esta población intermedia de reproducción sufren alteraciones debidas a la acción de los operadores genéticos (cruce y/o mutación) para

formar nuevas soluciones que constituyen una nueva población  $P(t+1)$ , que a su vez será la población inicial de la siguiente generación, y así sucesivamente, hasta alcanzar la condición de parada.

Al aplicar los AGs a la retroalimentación por relevancia, se puede partir de un conjunto de posibles soluciones, preguntas, que el algoritmo hace evolucionar tratando de conseguir la optimización. Para guiar la evolución de las posibles soluciones se tienen que diseñar funciones de adaptación que permitan valorar la bondad de cada una de éstas, utilizando para ello la información procedente de la retroalimentación del usuario.

Pues bien, para conseguir desarrollar un AG óptimo que implementase la técnica que nos ocupa, fuimos añadiendo al esqueleto del AG básico las alternativas que encontramos en la literatura especializada, y algunas otras, y finalmente el AG que mejores resultados obtuvo presenta las siguientes características (López-Pujalte, 2000; López-Pujalte et al. 2002a):

**2.1. Representación de los cromosomas.** Los vectores correspondientes a los documentos suministrados como retroalimentación sufren un proceso de conversión tal como utiliza Chen (1995) para transformarse en los cromosomas con los que trabajará nuestro AG. Dichos cromosomas utilizan la representación real, y tendrán tantos genes (componentes) como términos con pesos distintos de cero haya en la pregunta y en los documentos de la retroalimentación. Se calcula primero el conjunto de términos contenidos en esos documentos y en la pregunta, y el tamaño de los cromosomas será igual al número de términos de dicho conjunto.

**2.2. Población.** Nuestro AG recibe una población inicial compuesta por los cromosomas correspondientes a los documentos relevantes, a los no relevantes, a estos últimos con sus términos en negativo (lógicamente, en todos los casos nos referimos a los documentos procedentes de la retroalimentación) y por último a la pregunta supuestamente optimizada, es decir, se trata de un algoritmo genético híbrido (Davis, 1991).

**2.3. Selección.** El AG utiliza como mecanismo de selección el *muestreo aleatorio simple* (Goldberg, 1989; Holland, 1992).

También utiliza como complemento del mecanismo de selección la estrategia del *elitismo* (De Jong, 1975), de forma que si después de generar la nueva población no se encontraba en ella el mejor cromosoma de la población anterior por caprichos del azar, se elimina de la nueva población el peor individuo para poner en su lugar a éste.

**2.4. Operadores genéticos.** Como operador de cruce nuestro algoritmo emplea el cruce simple y para mutar los genes utiliza la mutación aleatoria (Goldberg, 1989; Holland, 1992; Michalewicz, 1995). También observamos que incluir un proceso de normalización de todos los cromosomas de la población cada vez que se aplicaba sobre ellos los operadores genéticos mejoraba el comportamiento del AG (aunque estos cromosomas provenían ya de documentos normalizados).

**2.5. Parámetros de control.** Todos los parámetros de control fueron fijados por experimentación, por lo que se realizaron numerosas pruebas con el fin de obtener los valores óptimos de dichos parámetros. Hay que señalar que los parámetros de control probabilidad de cruce  $p_c$  y probabilidad de mutación  $p_m$  con los que mejores resultados se han obtenidos son bastante más altos que los que normalmente se utilizan, sobre todo  $p_m$  ( $p_c = 0.8$  y  $p_m = 0.2$ ), debido a que esto ayuda a mantener la diversidad de la población y a evitar la convergencia prematura del algoritmo. Por otro lado, el número de generaciones finalmente empleado fue 20, (se hicieron pruebas con 10, 20, 40, 80, 100 y 200 generaciones) ya que a partir de él la población dejaba de mejorar.

**2.6. Solución.** Al finalizar el AG devuelve como solución tanto el mejor cromosoma hallado durante el proceso, como el centroide de aquellos cromosomas de la población final que tengan un valor de adaptación igual al valor máximo hallado en esa generación. Este segundo método para hallar la solución responde a la idea de no perder información que puede ser valiosa inútilmente, ya que si en la última generación hay más de un cromosoma que presente el valor máximo de adaptación, ¿bajo qué criterio íbamos a escoger uno y rechazar los restantes?. Ambos métodos para obtener la solución se han implementado y evaluado.

**2.7. Función de adaptación.** La función de adaptación, como ya hemos comentado, es de vital importancia en este tipo de algoritmos. De investigaciones que hemos realizado con anterioridad (López-Pujalte et al., 2002a, 2002b y 2002c) se puede concluir que, de las funciones probadas (doce en total, todas las presentes en la literatura y algunas de diseño propio), aquellas que mejores resultados obtienen son las que tienen en cuenta, no sólo qué documentos se recuperan sino el orden en que se recuperan. Es decir, funciones de adaptación que valoran no sólo si la posible solución recupera muchos documentos relevantes y pocos no relevantes, sino que también valoran si los documentos relevantes están al principio de la lista o al final. Hay que señalar que no todas las funciones de adaptación probadas obtuvieron buenos resultados, y que tan sólo dos de ellas superaron los conseguidos por el método de retroalimentación clásico de *Ide dec-hi*, que es de los métodos tradicionales el que mejor comportamiento presenta (Salton y Buckley, 1990).

Por tanto, la función de adaptación que presentamos aquí es una diseñada por nosotros y que, por supuesto, tiene en cuenta el orden de aparición de los documentos recuperados. Consiste en hallar la precisión media en nueve intervalos de exhaustividad (0.1, 0.2, ..., 0.9), pero llevando a cabo un proceso de interpolación para eliminar las ambigüedades.

Primero se calcula la similitud del cromosoma con los documentos suministrados en la retroalimentación utilizando como medida de similitud el coseno; una vez ordenados los documentos recuperados en orden decreciente de similitud, se calcula la precisión media interpolada en los nueve intervalos de exhaustividad que consiste, a grandes líneas, en asignar como precisión de un intervalo la precisión más alta de todos los puntos de exhaustividad que le anteceden por la derecha (Salton y McGill, 1983). Esta precisión media será el valor de adaptación del cromosoma dado.

## 2.8. Alternativas probadas.

Vamos a enumerar en esta sección algunas de las alternativas o mejoras al modelo clásico con las que hemos experimentado, pero sin obtener mejores resultados, tales como, siguiendo el modelo que proponen Herrera y colaboradores

(1995):

- Distintos mecanismos de muestreo: muestreo aleatorio simple, muestreo universal estocástico, ambos con o sin elitismo.
- Tipos de cruce: cruce simple, cruce multi-punto, y otro tipo de cruce más diferente, el cruce natural, empleado en el trabajo de Horng y Yeh (2000).
- Mutaciones: mutación aleatoria y mutación aleatoria en el intervalo  $[-1,1]$ , ya que el AG del trabajo de Roberston y Willet (1996) mejoraba de comportamiento conforme se introducían pesos negativos.
- Propuesta de otros operadores: dado que la media de los vectores de la población final ofrecía en algunos casos los mejores resultados, se implementó un operador de media que se aplicaba alternativamente al operador de cruce (aunque con menor probabilidad).
- Modificación en la función de adaptación  $f$ : con el objeto a veces de acentuar las diferencias en algunas evaluaciones que daban resultados muy similares, y otras, de suavizarlas, realizamos escalados de la función de adaptación del tipo  $f^2$ ,  $f^3$ ,  $\sqrt{f}$  y  $10^f$ .
- Técnicas adaptativas: en nuestro trabajo utilizamos también una técnica adaptativa para ajustar los parámetros de control, concretamente la probabilidad de cruce ( $p_c$ ) y la probabilidad de mutación ( $p_m$ ), a lo largo de la ejecución del algoritmo, en particular el método denominado por Yang y Korfhage "método híbrido" en el que  $p_c$  va descendiendo desde 0.9 a 0.6 (0.05 cada tres generaciones) y  $p_m$  va aumentando simultáneamente desde 0.01 hasta 1; si bien  $p_m$  se reduce bruscamente en las tres últimas generaciones para no perder las buenas soluciones (Yang y Korfhage, 1994).
- Hibridación: hemos utilizado AGs Híbridos.

### 3. Experimentación.

Para llevar a cabo nuestros experimentos nos hemos intentado ajustar, en todo lo posible, a los procedimientos seguidos por Salton y Buckley (1990) en su estudio sobre los métodos clásicos de retroalimentación por relevancia más utilizados. En dicho estudio, el método que mejores resultados obtuvo fue el de Ide dec-hi

(Ide, 1971), y es por ello que lo hemos utilizado para comparar con nuestros AGs. Este método consiste en añadir directamente a los pesos de la pregunta original los de todos los documentos relevantes del conjunto de documentos suministrados en la retroalimentación, y restarles los del primer documento no relevante obtenido en la recuperación que pertenezca a dicho conjunto. Formalmente, la reformulación del vector consulta es la siguiente:

$$Q' = Q + \sum_{\substack{\text{todos} \\ \text{relevantes}}} D_i - S$$

donde:

$Q$  es el vector de la pregunta original

$D_i$  es el vector del documento relevante  $i$

$S$  es el vector del primer documento no relevante del ranking.

Nuestra experimentación se realizó sobre la colección *Cranfield*, que consiste en 1.398 documentos sobre distintos aspectos de ingeniería aeronáutica y 225 preguntas para las que se conocen los juicios de relevancia. Uno de los principales motivos para la selección de esta colección fue que ha sido empleada en otras ocasiones para experimentos de retroalimentación (Korfhage, 1997), tanto por métodos tradicionales como con técnicas genéticas, y también lo fue en el trabajo de Salton y Buckley (1990) que nos sirve de modelo, con lo cual nos ofrece la posibilidad de comparar los resultados.

Antes de proceder con las técnicas de retroalimentación lo primero que tuvimos que hacer fue llevar a cabo un proceso de vectorización, utilizado también en otros experimentos (Guerrero y Moya, 2001; Guerrero et al., 2002), tanto de los documentos como de las preguntas de la colección, para lo cual realizamos los pasos siguientes:

- Extracción de todas las palabras de cada uno de los documentos.

- Eliminación de las palabras vacías, a partir de una lista de palabras vacías generada con el diccionario de frecuencias de Kuccera y Francis (1967).

- *Reducción a la raíz* del resto de palabras, para lo cual utilizamos el algoritmo de *Porter* que es el más utilizado en lengua inglesa (Porter, 1980; Frakes y Baeza-Yates, 1992).

Realizado el proceso anterior sobre los documentos de la colección *Cranfield* la cifra final de términos es 4307, con lo que

trabajaremos con 1398 vectores documentales de 4307 componentes.

Una vez determinados de este modo los términos que describen a todos los documentos de la colección, para la asignación de pesos empleamos el esquema propuesto en su trabajo por Salton y Buckley (1990) cuya fórmula es:

$$a_{ij} = \frac{\left(0.5 + 0.5 \frac{tf_{ij}}{\max tf}\right) \cdot \log \frac{N}{n_i}}{\sqrt{\left(0.5 + 0.5 \frac{tf_{ij}}{\max tf}\right)^2 \left(\log \frac{N}{n_i}\right)^2}}$$

donde  $a_{ij}$  es el peso asignado al término  $t_j$  en el documento  $D_i$ ,  $tf_{ij}$  es el número de veces que aparece el término  $t_j$  en el documento  $D_i$ ,  $n_i$  es el número de documentos indizados por el término  $t_j$ , y por último,  $N$  es el número total de documentos de la base documental.

Por último, realizamos la normalización de dichos vectores, dividiéndolos por su norma euclídea (Noreault et al. 1981), ya que las mejores medidas de similitud son aquellas que efectúan comparaciones angulares entre vectores.

Un proceso similar al anterior se realiza con las preguntas asociadas a la colección, obteniendo como resultado los vectores normalizados correspondientes a las consultas, que son los que vamos a tratar de optimizar mediante la retroalimentación.

Después, hubo que determinar cuál iba a ser la información que se suministraría al sistema como retroalimentación. El número de documentos fijado para implementar la retroalimentación fue 15, tal y como hacen Salton y Buckley (1990) en sus experimentos. Es decir, para cada pregunta, se examinan los 15 primeros documentos recuperados para ver su relevancia y esta información es la que se facilita al algoritmo encargado de optimizar la consulta mediante esta técnica.

El esquema del programa para implementar la retroalimentación por relevancia mediante los dos métodos (el clásico y el genético) es muy sencillo (López-Pujalte, 2000):

- Cada pregunta de la colección se compara con todos los documentos de la misma, utilizando como medida de similitud el coseno, obteniéndose así una lista con las similitudes de cada pregunta con todos los documentos de la colección.

- Esta lista se ordena por grado de similitud en orden decreciente.

- Los vectores documentales normalizados correspondientes a los 15 primeros documentos de la lista, con sus juicios de relevancia y el vector normalizado de la pregunta se suministran como entrada al algoritmo encargado de realizar la optimización de la consulta, y ésta será la salida de nuestro programa.

## 5. Evaluación

En cuanto al proceso de evaluación seguido, éste consistió en evaluar los resultados de la recuperación mediante las medidas clásicas de *exhaustividad* y *precisión*, calculando la precisión interpolada a intervalos fijos de exhaustividad (Salton y McGill, 1983) y hallando el promedio de precisión en tres puntos de exhaustividad (0.25, 0.5 y 0.75, que representan exhaustividad baja, media y alta respectivamente) tal y como hacen Salton y Buckley (1990) en su estudio sobre retroalimentación para poder comparar los distintos sistemas.

Se utiliza también el método de la *colección residual* en el que todos los documentos previamente vistos por el usuario (sean relevantes o no) se extraen de la colección, y ambas, la consulta inicial y la modificada son evaluadas sobre esta colección residual, ya que la operación de retroalimentación por relevancia debe ser juzgada por su habilidad para recuperar documentos nuevos, no examinados originalmente por el usuario. De hecho, este es el método estándar de evaluación de esta técnica, ya que proporciona una evaluación más realista e imparcial (Salton y Buckley, 1990; Harman, 1992; Baeza-Yates y Ribeiro-Neto, 1999).

Para ello, el programa descrito anteriormente generará también como salida un fichero de máscara que contiene para cada pregunta todos los documentos que no se deben considerar en el proceso de evaluación (que serán los 15 primeros que se han empleado en la modificación de las consultas).

El esquema del programa principal se muestra en la Figura 3.

## 6. Resultados.

Los resultados obtenidos por ambos métodos se muestran en la Tabla 1, donde se puede ver, que nuestro AG obtiene un 127,2 % de mejora

con respecto a la pregunta original (en una sola iteración) frente al 120.8 % de mejora del método de Ide dec-hi, siendo éste, como ya hemos dicho, el que mejor comportamiento presenta.

Y nuestro AG logra superarlo utilizando el método del centroide para obtener la solución, lo que demuestra que es una buena forma de seleccionar la salida del AG en este tipo de problemas.

Por otro lado tenemos que señalar, que el tiempo de CPU empleado por el método clásico, como es lógico, es bastante menor (0.12 seg./preg.) que el empleado por nuestro genético (3.39 seg./preg.); sin embargo pensamos que este último es sobradamente aceptable.

## 7. Conclusiones y Líneas de futuro.

Los AGs son algoritmos especialmente aptos para explorar espacios complejos de grandes dimensiones y, por lo tanto, tienen un importante campo de aplicación dentro de la RI.

Y dentro del estudio que hemos llevado a cabo sobre retroalimentación por relevancia, nuestro AG ha obtenido mejores resultados que el método clásico de Ide dec-hi, que por otro lado, fue el que mejores resultados obtuvo de entre seis métodos de retroalimentación comúnmente utilizados en el trabajo que realizaron Salton y Buckley (1990) sobre el tema.

Sin embargo, hay que señalar, que en investigaciones anteriores (López-Pujalte et al., 2002a) hemos comprobado que no siempre ocurre esto, y que es de vital importancia la elección de la función de adaptación para que ésta guíe bien al algoritmo en su proceso de búsqueda, y podemos concluir que las funciones que mejores resultados obtienen son aquellas que tienen en cuenta, no sólo qué documentos se recuperan sino el orden en que se recuperan.

Por último, y como conclusión final, dada la relativa facilidad de implementación de la técnica de retroalimentación por relevancia (ya sea por métodos genéticos como por métodos más tradicionales) y los excelentes resultados que se consiguen con ella, hoy en día cualquier SRI que se precie debería de incorporar un módulo de retroalimentación.

Como líneas inmediatas de investigación tenemos pensado experimentar con modificaciones del AG que incluyan, sobre

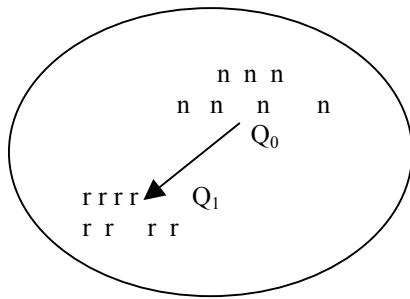
todo, mecanismos de selección y operadores de cruce más apropiados para la representación real, sin dejar, por supuesto, de seguir investigando sobre el diseño de funciones de adaptación, ya que consideramos que ahí radica una clave importantísima del rendimiento de nuestro AG.

## Bibliografía

- Baeza-Yates, R.; Ribeiro-Neto, B. (1999). *Modern information retrieval*. Essex, UK: Addison-Wesley.
- Chen, H. (1995). Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science*, 46(3), 194-216.
- Chen, H. ; Chung, Y. ; Ramsey, M. (1998a). A smart itty bitsy spider for the web. *Journal of the American Society for Information Science*, 49(7), 604-618.
- Chen, H.; Shankaranarayanan, G.; She, L.; Iyer, A. (1998b). A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing. *Journal of the American Society for Information Science*, 49(8), 693-705.
- Cordón, O.; Moya, F.; Zarco, M.C. (2000). A GA-P Algorithm to Automatically Formulate Extended Boolean Queries for a Fuzzy Information Retrieval System by means of GA-P Techniques. *Mathware & Soft Computing*, 7:2-3, 309-322.
- Cordón, O.; Moya, F.; Zarco, M.C. (2002). A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. *Soft Computing* (pendiente de publicar).
- Davis, L., (Ed.) (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.
- De Jong, K.A. (1975). An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan.
- Frakes, W.B.; Baeza-Yates, R. (1992). *Information retrieval: data structures & algorithms*. Edited by William B. Frakes, Ricardo Baeza-Yates. Englewood Cliffs, N. J. : Prentice Hall.

- Goldberg, D.E. (1989). Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley.
- Guerrero Bote, V. P. ; Moya Anegón, F. (2001). Reduction of the dimension of a document space using the fuzzified output of a Kohonen network. *Journal of the American Society for Information Science and Technology*, 52, 1234-1241.
- Guerrero Bote, V. P.; Moya Anegón, F.; Herrero Solana, V. (2002). Document Organization using Kohonen's Algorithm. *Information Processing & Management*, 38, 79-89 .
- Harman, D.K.(1992). Relevance feedback and other query modificaton techniques. En: Frakes, W.B.; Baeza-Yates, R. (Eds.) *Information retrieval: data structures and algorithms*. NJ: Prentice Hall, 241-263.
- Herrera, F.; Lozano, M.; Verdegay, J.L. (1995). Algoritmos genéticos: fundamentos, extensiones y aplicaciones. *Arbor*, CLII, 597, 9-40.
- Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems*. 2<sup>nd</sup> ed. MIT Press.
- Hornig, J-T.; Yeh, C-C. (2000). Applying genetic algorithms to query optimization in document retrieval. *Information Processing and Management*, 36, 737-759.
- Ide, E. (1971). New experiments in relevance feedback. In: Salton, G. (Ed.) *The SMART retrieval system*. Englewood Cliffs, N.J.: Prentice-Hall, 337-54.
- Korfhage, R.R. (1997). *Information storage and retrieval*. New York, NY: Wiley Computer Publishing.
- Kraft, D.H.; Petry, F.E.; Buckles, B.P.; Sadasivan, T. (1997). Genetic algorithms for query optimization in information retrieval: relevance feedback. In: Sanchez, E.; Shibata, T.; Zadeh, L.A. (Eds.) *Genetic algorithms and fuzzy logic systems. Soft computing perspectives*, 155-173.
- Kucera, H.; Francis, N. (1967). *Computational analysis of present-day American English*. Providence, RD: Brown University Press.
- López-Pujalte, C. (2000). Algoritmos genéticos aplicados a la retroalimentación por relevancia. Tesis Doctoral, Facultad de Biblioteconomía y Documentación, Universidad de Granada.
- López-Pujalte, C.; Guerrero Bote, V.P ; Moya Anegón, F. (2002a). A test of genetic algorithms in relevance feedback. *Information Processing and Management* (pendiente de publicar).
- López-Pujalte, C.; Guerrero Bote, V.P; Moya Anegón, F. (2002b). Order-based fitness functions for genetic algorithms applied to relevance feedback. *Journal of the American Society for Information Science and Technology* (pendiente de publicar).
- López-Pujalte, C.; Guerrero Bote, V.P; Moya Anegón, F. (2002c). Funciones de adaptación óptimas para implementar la retroalimentación por relevancia en los sistemas de recuperación de información actuales. I Congreso Español de Algoritmos Evolutivos y Bioinspirados (Mérida), 67-72.
- Martín-Bautista, M.J.; Vila, M.A.; Larsen, H.L. (1999). A fuzzy genetic algorithm approach to an adaptive information retrieval agent. *Journal of the American Society for Information Science*, 50(9), 760-771.
- Michalewicz, Z. (1995). *Genetic algorithms + data structures = evolution programs*. Berlin: Springer-Verlag.
- Noreault, T.; McGill, M.; Koll, M.B. (1981). A performance evaluation of similarity measures, document term weighting schemes and representation in a Boolean environment. *Information Retrieval Research*. London: Butterworths. Porter M. F. (1980). An algorithm for suffix stripping. *Program* 14(3), 130-137.
- Porter M. F. (1980). An algorithm for suffix stripping. *Program* 14(3), 130-137.
- Robertson, A.M.; Willett, P. (1996). An upperbound to the performance of ranked-output searching: optimal weighting of query terms using a genetic algorithm. *Journal of Documentation* 52(4), 405-20.
- Salton, G.; Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4), 288-297.
- Salton G.; McGill, M.J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.
- Yang, J.J.; Korfhage, R. (1994). Query modification using genetic algorithms in vector space models. *International Journal of Expert Systems* , 7(2), 165-191.

## Tablas y Figuras



donde:

$n$  = documentos identificados como no relevantes

$r$  = documentos identificados como relevantes

$Q_0$  = pregunta original

$Q_1$  = pregunta modificada

Figura 1: proceso de retroalimentación por relevancia (adaptado de Salton y McGill, 1983).

## Algoritmo Genético

Inicio

$t = 0$ ;

Inicializar  $P(t)$ ;

Evaluar  $P(t)$ ;

Mientras no sea la condición de parada, hacer

Inicio

$t = t + 1$ ;

Seleccionar  $P(t)$  de  $P(t-1)$ ;

Alterar (cruzar y mutar)  $P(t)$ ;

Evaluar  $P(t)$ ;

Fin

Fin

Figura 2: Estructura en pseudocódigo de un Algoritmo Genético (adaptado de Michalewicz, 1995).

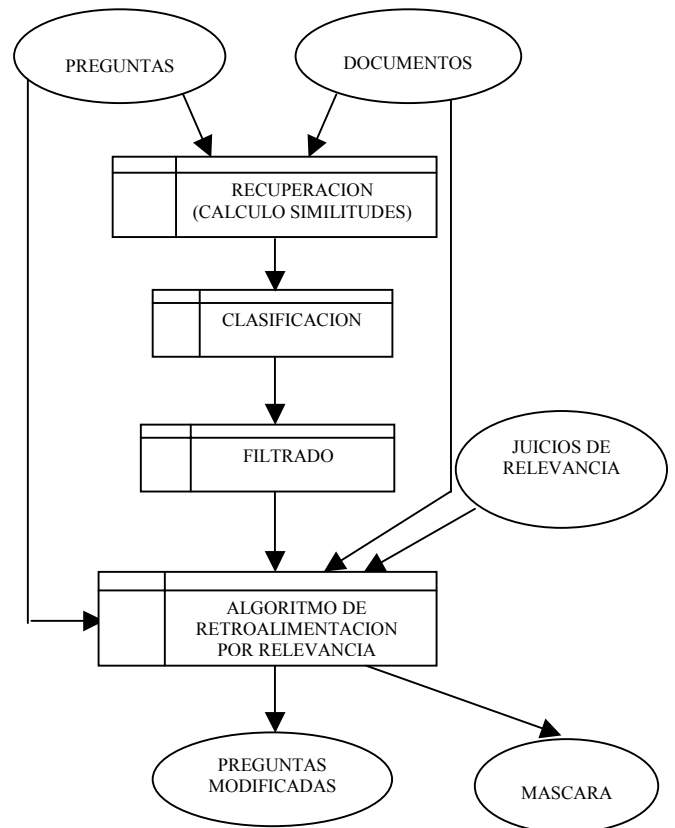


Figura 3: Diagrama de flujo de datos correspondiente al diseño del experimento.

METODO	EL MEJOR		CENTROIDE	
	MEDIA	MEJORA	MEDIA	MEJORA
Sin retroalimentación	0.098			
Ide dec-hi	0.218	120.8 %		
AG	0.218	120.8 %	0.224	127.2 %

Tabla 1: Resultados de los diferentes métodos de retroalimentación por relevancia para la colección Cranfield