Diseño de un motor de recuperación de la información para uso experimental y educativo

[versió catalana]

Carlos G. Figuerola
Facultad de Documentación
Universidad de Salamanca
figue@gugu.usal.es

José Luis Alonso Berrocal
Facultad de Documentación
Universidad de Salamanca
berrocal@gugu.usal.es

Ángel Francisco Zazo Rodríguez
Facultad de Documentación
Universidad de Salamanca
afzazo@gugu.usal.es

Resumen

Se describe el diseño y funcionamiento de un motor de recuperación de información, basado en el modelo vectorial y cuya finalidad es servir de base de experimentación en tareas de investigación, así como de recurso para la docencia. No obstante, el motor resulta completamente operacional, y puede ser utilizado en entornos documentales. Construido sobre una base de datos relacional, facilita la observación y manipulación de estructuras y resultados intermedios; realiza las operaciones fundamentales a partir de sentencias SQL, lo cual permite una fácil modificación de su funcionamiento interno y, en consecuencia, la experimentación.

1 Introducción

La recuperación de la información es una disciplina de creciente interés, habida cuenta del aumento de la disponibilidad de documentos en soporte electrónico y de la necesidad consiguiente de obtener en cada momento aquéllos que responden a una necesidad informativa dada. Su alcance y objetivos han sido definidos repetidas veces desde hace ya tiempo; una descripción de los mismos puede encontrarse en Rijsbergen (1979). Su inclusión, de una manera u otra, en los currículums de los titulados en Ciencias de la Documentación, así como su evidente perfil como campo de investigación, hacen aconsejable disponer de herramientas que permitan, por un lado, facilitar la enseñanza de dicha materia; y, de otro, posibilitar el diseño y la realización de experimentos como parte de diferentes trabajos de investigación.

Estos motivos nos han conducido a la construcción de un pequeño pero robusto motor de búsqueda, que pudiera servir para diversos fines. Los objetivos básicos a la hora de diseñarlo han sido:

a) disponer de un programa que implementara algunos de los modelos teóricos más difundidos en recuperación de la información

b) disponer de un programa que permitiese examinar fácilmente los resultados de ciertas operaciones internas, como el cálculo de pesos de los términos
c) la posibilidad de discriminar campos en los documentos, y de valorarlos de manera diferente
d) la posibilidad de modificar fácilmente la mecánica interna de operación del programa, aplicando diferentes criterios y sistemas de cálculo de pesos y similaridades, a fin de observar el efecto de dichas modificaciones
e) la posibilidad de implementar conocimiento lingüístico, adaptando las posibilidades de recuperación a las características propias de un idioma determinado

Aunque, como se ve, los objetivos prioritarios eran diseñar una herramienta para la docencia y la investigación, y no un motor de búsqueda operacional, el resultado es lo suficientemente robusto como para ser utilizado con éxito en determinados entornos documentales. Así, aunque el código no está optimizado y puede resultar algo lento, puesto que está escrito pensando en su fácil comprensión y modificación, sus tiempos de respuesta son aceptables: no más de 1 segundo para una colección de 15.000 documentos, en un Pentium II (266 mhz) con 64 M de RAM, para una consulta de diez palabras.

Nuestro motor de búsqueda documental, al que hemos llamado Karpanta, adopta el conocido modelo del espacio vectorial (Salton y McGill, 1983). Brevemente, según este modelo, cada documento es representado mediante un vector de *n* elementos, siendo *n* igual al número de términos indizables que existen en la colección documental. Hay, pues, un vector para cada documento, y, en cada vector, un elemento para cada término o palabra susceptible de aparecer en el documento. Cada uno de esos elementos es cubierto u ocupado con un valor numérico. Si la palabra no está presente en el documento, ese valor es igual a 0. En caso contrario, ese valor es calculado teniendo en cuenta diversos factores, dado que una palabra dada puede ser más o menos significativa (tanto en general como, sobre todo, en ese documento en concreto); este valor se conoce con el nombre de peso del término en el documento.

Siempre según el modelo del espacio vectorial, las consultas son representadas también mediante un vector de las mismas características que las de los documentos (variando los valores numéricos de cada elemento en función de las palabras que forman parte de la consulta, claro está). Esto permite calcular fácilmente una función de similaridad dada entre el vector de una consulta y los de cada uno de los documentos. El resultado de dicho cálculo mide la semejanza entre la consulta y cada uno de los documentos, de manera que, aquéllos que, en teoría, se ajustan más a la consulta formulada, producen un índice más alto de similaridad. Naturalmente, se asume que la consulta se formula en lenguaje natural (podría ser, incluso un documento de muestra, para recuperar los que fuesen parecidos a él), y, de lo dicho se deduce que el resultado de la consulta consiste en una lista de documentos ordenada en orden decreciente en función de su similaridad con la consulta. Diversos factores son modificables dentro de este modelo de recuperación. Así, entre otros, el sistema de cálculo de pesos. Los esquemas habituales se basan, de una u otra forma, en las frecuencias de aparición de la palabra cuyo peso se quiere calcular. Así, muchos sistemas utilizan algún mecanismo basado en la multiplicación del IDF del término por su

frecuencia en el documento en cuestión. El propio IDF (Inverse Document Frequency) tiene varias versiones, por lo general basadas en una función inversa del número de documentos en que aparezca el término en cuestión (Harman, 1992a; Salton y Buckley, 1988). Nuestro motor de búsqueda se diseñó pensando en la fácil sustitución de una fórmula de cálculo por otra cualquiera, pero también en la identificación, mediante etiquetado, de partes del documento, de manera que, las palabras integrantes de zonas marcadas, pudieran pesarse de forma distinta en función del etiquetado. Así, por ejemplo, podría desearse adjudicar un mayor peso a las palabras que apareciesen en los títulos de los documentos.

Del mismo modo, es también fácilmente sustituible la ecuación de cálculo de similaridad. Una de las más utilizadas es la conocida como fórmula del coseno (Salton, 1989), pero puede sustituirse con facilidad por otra cualquiera.

El mismo concepto de término indizable es flexible en dos sentidos: en primer lugar, definiendo una lista de palabras vacías. Esta lista puede ser modificada en cualquier momento, y, puesto que Karpanta ofrece la posibilidad de examinar frecuencias de palabras en la colección, es posible utilizar esta información directamente para construir o modificar la lista de palabras vacías (Wilbur y Sirotkin, 1992). En segundo lugar, las palabras indizables necesitan ser normalizadas, lo que en Karpanta se consigue a través de la lematización (Hull, 1996). Karpanta incorpora un *s-stemmer*, pero, dado que se trata de una función autónoma, es fácil sustituirla por otro algoritmo diferente. Obsérvese que la lematización puede llevarse a cabo también mediante un pre-procesado externo de los documentos, o ceñirla sólo a las consultas (Kraaij y Pohlmann, 1996). Diferentes estudios muestran que éste es uno de los elementos en los cuales las peculiaridades lingüísticas juegan un papel importante (Harman, 1991; Popovic y Willet, 1992).

2 Arquitectura

Básicamente, Karpanta se apoya en dos módulos: uno de indización, que construye los vectores de los documentos, y otro de consulta, que calcula la similaridad con una consulta dada. Tanto los documentos como los vectores resultantes, así como productos intermedios y auxiliares, se almacenan en una base de datos relacional. A pesar de que los sistemas de gestión de bases de datos relacionales han padecido en el pasado un cierto descrédito en lo que se refiere a su utilización en entornos documentales (Trigueros Díaz e Higuera Matas, 1997; Moya Anegón, 1995; Codina, 1994), creemos que constituyen un medio idóneo para esta tarea. A las ventajas genéricas ya conocidas de este tipo de sistemas (prevención contra la redundancia y la inconsistencia, facilidad de modificación de estructuras, flexibilidad de manejo, estandarización, etc.), hay que añadir el hecho de que en la actualidad han superado algunos de los inconvenientes que tradicionalmente se les han achacado: posibilidad de campos de tamaño variable (los conocidos campos memo), posibilidad de almacenar datos binarios (imágenes, sonido, referencias a objetos externos), y, desde luego, campos repetibles (aunque siempre ha sido posible tener campos repetibles con una base de datos relacional; de hecho, ésta es una de sus razones de ser (Date, 1983)). Adicionalmente, sistemas de gestión de bases de datos relacionales

(con sus lenguajes estándar de manipulación de datos e interrogación) corren hoy de forma ágil y sin problemas en ordenadores personales.

En nuestro caso, se eligió el SGBD Microsoft Access. Aparte de la fácil disponibilidad del mismo, la transparencia de este sistema, y la posibilidad de acceder y ver (modificar, en su caso) sus tablas, con los productos intermedios de la indización, son importantes para nosotros, teniendo en cuenta los objetivos de aplicación a la docencia e investigación planteados a la hora de diseñar Karpanta. Así, es posible mostrar (a un alumno, por ejemplo) los términos y los pesos de cada uno de ellos, obtener los de mayor o menor peso, y cuestiones similares. En el mismo sentido, es fácil modificar manualmente cualquiera o todos los vectores y observar el resultado. Dado que se utilizan prestaciones estándar, es posible portar el sistema a otro SGDB (por ejemplo, para otro sistema operativo, como Unix) sin problemas.

De otro lado, la mayor parte de las operaciones se llevan a cabo mediante el lenguaje estándar en las bases de datos relacionales, SQL (Date, 1983). Esto implica, desde luego, una portabilidad garantizada entre sistemas, pero, sobre todo, y desde nuestro punto de vista, una facilidad enorme de modificación en cosas como el cálculo de pesos o de similaridad; cualquiera de estas operaciones no requiere más que cambiar una sola línea de código. La utilización de SQL para implementar sistemas basados en el modelo del espacio vectorial fue planteada ya hace algunos años (Grossman [*et al.*], 1995; Grossman [*et al.*], 1996). SQL destaca por su sencillez y precisión conceptual, pero desde un punto de vista práctico, resultaba en el pasado tal vez demasiado costoso en capacidad de proceso y recursos de máquina. En la actualidad, sin embargo, con el aumento creciente en la velocidad de los procesadores, y con el reducido precio de las memorias, SQL resulta perfectamente ágil y operativo en cualquier ordenador personal.

En la misma línea, las sentencias SQL deben ir embebidas en algún lenguaje anfitrión; e, igualmente, es preciso utilizar algún lenguaje de programación para llevar a cabo determinadas operaciones, como la separación o segmentación en palabras de los documentos. En la actualidad, Karpanta hace esto a través de Visual Basic, pero no sería muy diferente si se hiciese a través de otro lenguaje (por ejemplo, Perl, si se porta a un sistema Unix).

Karpanta almacena inicialmente los documentos en un campo de tipo memo en una tabla que tiene otro campo adicional para proporcionar una clave a cada documento; obsérvese que nada impide establecer campos adicionales (por ejemplo, fecha del documento). Gracias a la sencillez del SQL, es muy fácil crear filtros adicionales a las consultas para estos nuevos campos. Las palabras consideradas a priori vacías se almacenan en otra tabla, con un único campo destinado a este fin. Durante el proceso de indización, Karpanta construirá un fichero invertido, con cada aparición u ocurrencia de cada término, junto con la clave del documento en que aparece, que almacenará en otra tabla, a partir de la cual, mediante la oportuna sentencia SQL, Karpanta calculará IDFs de cada término, así como pesos de cada término en cada documento.

doc_term : Tabla

termino	frecuencia	peso	clave
ACTIVIDADES	2	8,919612	1
AFINES	1	6,68443	1
AUTONOMA	3	17,30442	1
BIBLIOGRAFIA	1	8,070724	1
BIBLIOTECA	1	2,98932	1
BREVE	1	4,89267	1
CADA	1	3,95985	1
CIENCIAS	3	14,43788	1
CIENTIFICA	4	18,95408	1
CITAN	1	6,972112	1
CLASIFICACIO	1	5,126285	1
COLABORA	1	6,972112	1
COLABORACIC	1	4,851848	1
CONOCER	1	4,93523	1
CONSULTADA	1	8,070724	1
CREACION	1	4,08174	1
DA	3	15,54106	1
DEPARTAMEN	3	18,37444	1
DETALLAN	1	6,972112	1
DOCUMENTAC	6	18,40067	1
DOCUMENTAR	1	7,377577	1
EMPLEADOS	1	6,278965	1

Registro: 1 de 36925

Ilustración 1. Tabla con términos, puntero a documentos, frecuencia en documento y peso

La arquitectura del módulo de indización es, básicamente, la siguiente:

a) Procesado de documentos (a través de un lenguaje de programación convencional, como VB).

1. obtención de palabras de cada documento
2. filtrado y eliminación de palabras vacías
3. normalización de caracteres (mayúsculas, minúsculas, acentos)
4. lematización. En la actualidad, Karpanta aplica un S-Stemmer modificado ligeramente para el castellano
5. almacenamiento en tabla de cada término resultante, junto con la referencia o clave de los documentos en que aparece

b) Cálculo de frecuencias de términos, IDFs y pesos, mediante sentencias SQL, y almacenamiento de resultados en una tabla.

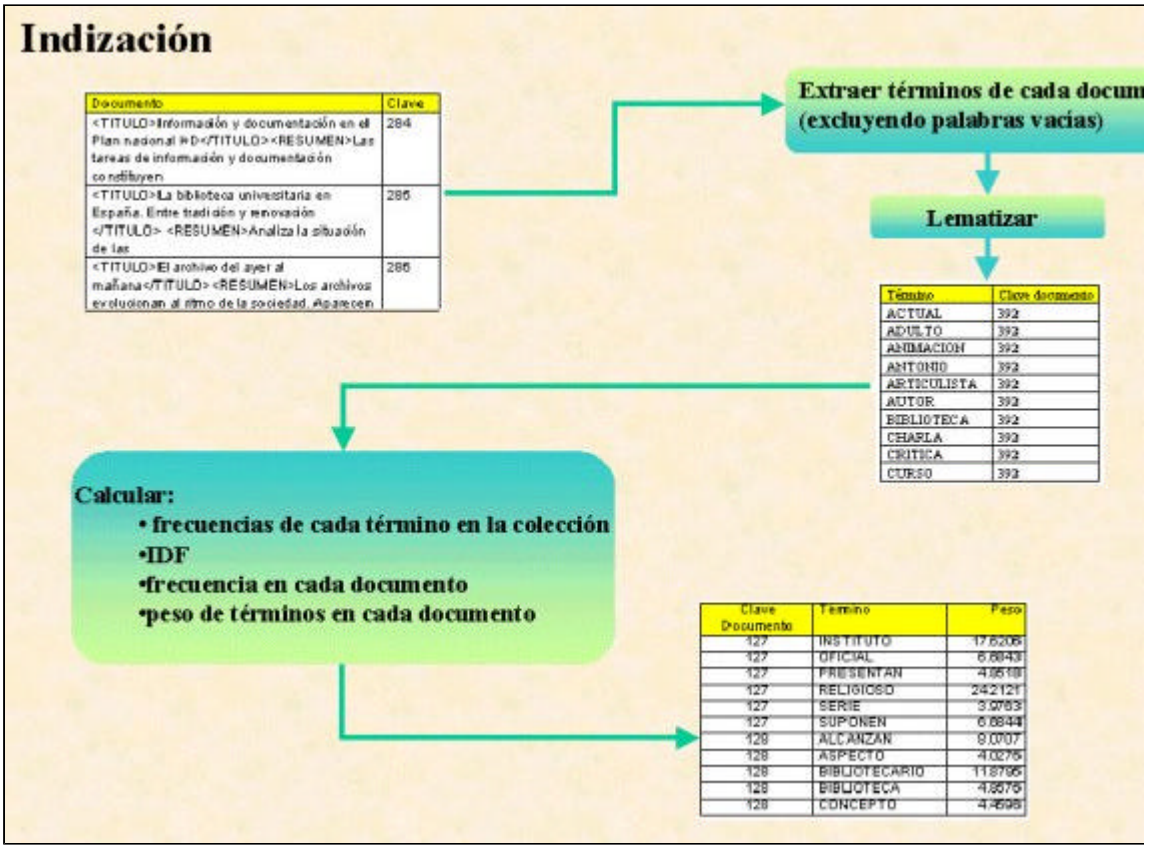

Ilustración 2. Proceso de indización

El módulo de consulta es aún más simple. Dado que una consulta en lenguaje natural ha de ser tratada como un documento cualquiera, requiere las mismas operaciones:

a) Procesado del texto de la consulta (obtención de palabras, eliminación de vacías, normalización de caracteres, lematización

b) Cálculo de pesos de los términos de la consulta, utilizando los datos de IDF almacenados en una tabla en la operación de indizado

c) Cálculo de similaridad entre consulta y cada uno de los documentos, mediante una simple sentencia SQL

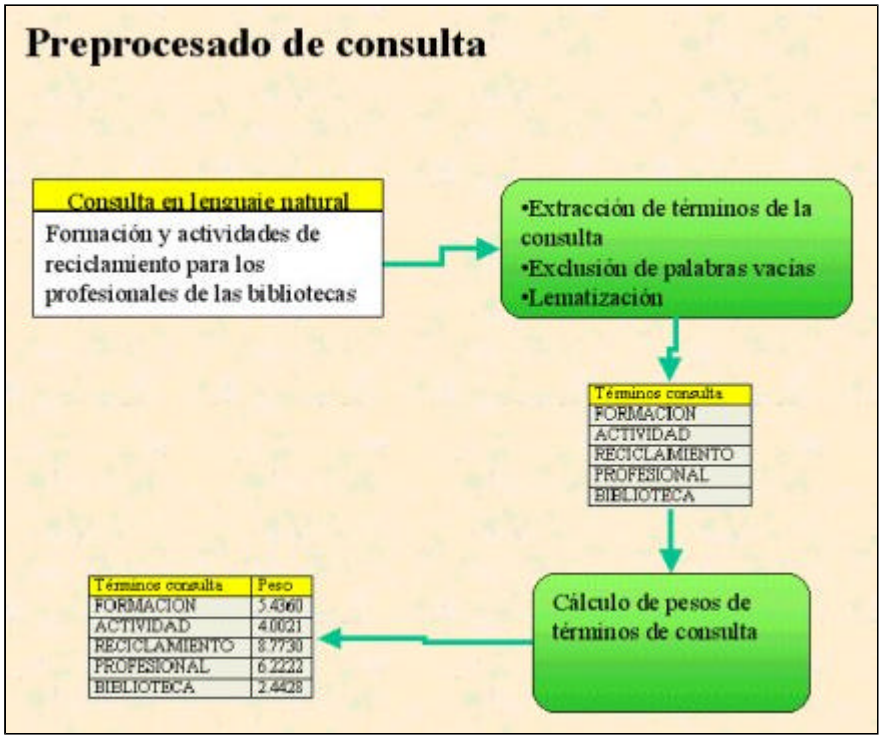

Ilustración 3. Esquema de preprocesamiento de una consulta

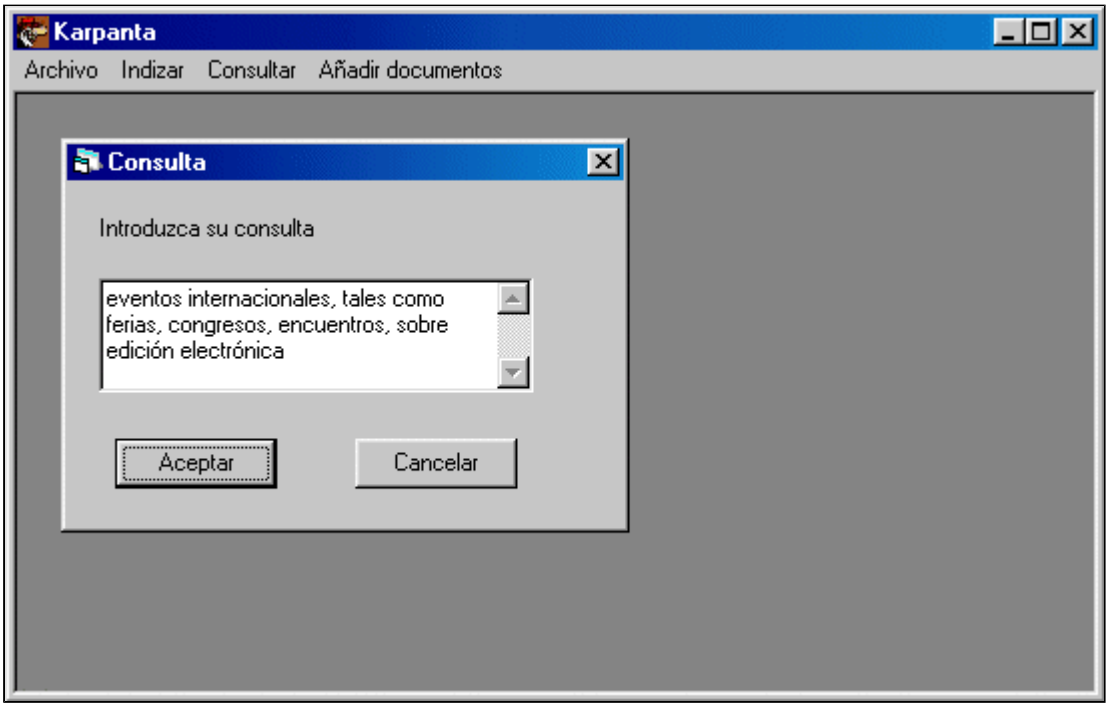

Ilustración 4. *Demo* de Karpanta: consulta en lenguaje natural

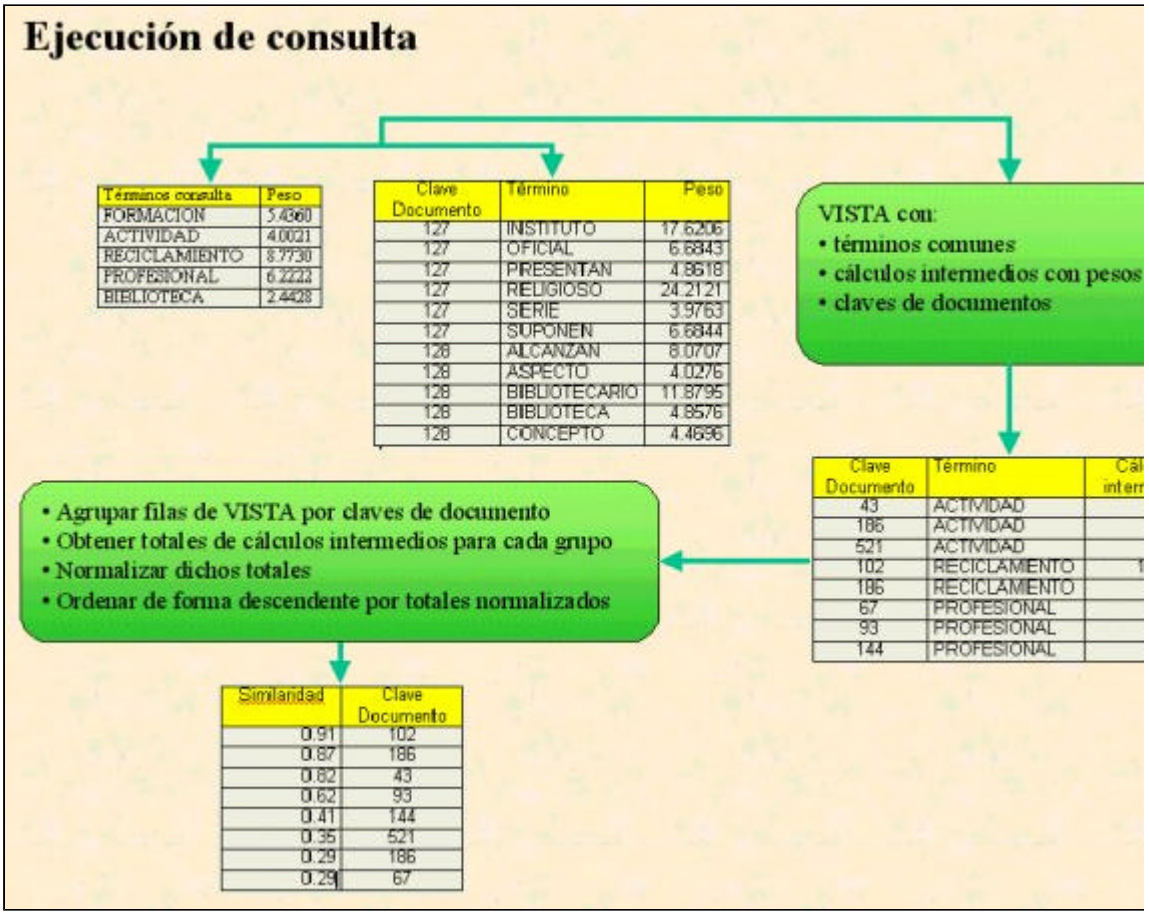

Ilustración 5. Esquema de resolución de una consulta

3 Realimentación de consultas

A partir de esta arquitectura, implementar realimentación de consultas es sencillo. Una buena exposición de las cuestiones planteadas por la expansión de consultas por realimentación puede encontrarse en (Harman, 1992; Buckley [*et al.*], 1994). Para una expansión simple, no hay más que añadir al texto original de la consulta el texto de los documentos con los que se desea realimentar, ya sea a través de selección por parte del usuario, ya sea tomando de forma automática los *n* primeros recuperados por la consulta original; tras esto, reejecutar la consulta con los añadidos hechos.

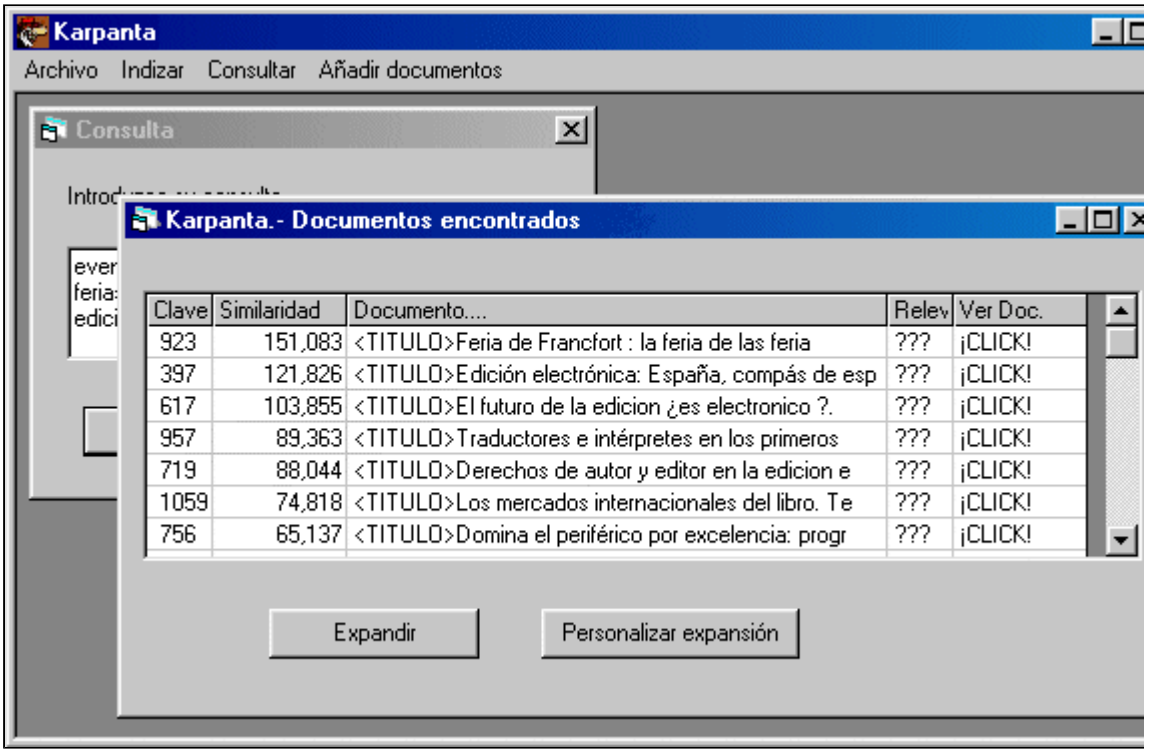

Ilustración 6. Demo de Karpanta: lista de documentos recuperados tras una consulta

Un enfoque algo más elaborado, puede requerir un reajuste de pesos de términos, en especial si se tienen en cuenta, entre los documentos recuperados por la consulta original, casos positivos y casos negativos; es decir, documentos recuperados que deben usarse para realimentar (ejemplos positivos), y documentos que, explícitamente, deben usarse como ejemplos negativos (es decir, que no se desean documentos como ésos). La aproximación más frecuente es la utilización del conocido algoritmo de Rocchio (Rocchio, 1971). La implementación de este algoritmo en Karpanta requiere algún sistema para que el usuario, tras examinar los documentos obtenidos en una primera consulta, determine cuáles son los ejemplos positivos y cuáles los negativos. Naturalmente, esta cuestión se resuelve a través del interfaz con el usuario. El recalculado posterior de pesos se efectúa mediante una simple sentencia SQL (Grossmann, 1997). Esto posibilita ajustar manualmente los coeficientes a aplicar (tanto negativos como positivos) sin necesidad de alterar el código del programa.

4 Acciones futuras

Dado el carácter y objetivos planteados al diseñar Karpanta, es evidente que no se trata de un proyecto cerrado. Antes bien, la sencillez y flexibilidad de que se le ha dotado, aún cuando sacrifica levemente la operatividad, permite la adición de nuevas prestaciones. De algún modo podemos decir que se diseñó así precisamente para eso.

Entre las acciones más inmediatas previstas se encuentra la implementación de un algoritmo de lematización específico para el idioma castellano, más eficiente. En la actualidad, Karpanta incorpora un s-stemmer modificado,

después de haber descartado otros algoritmos de uso frecuente con el inglés, tras haber comprobado experimentalmente su limitado alcance (Gómez Díaz, 1998). En la actualidad, un lematizador flexional para el castellano está prácticamente listo para usarse, y se trabaja en la elaboración de un lematizador derivativo; parece fuera de toda duda que la lematización flexiva produce mejoras importantes en la recuperación, desde luego en lo referente a exhaustividad, y parece que también en precisión (Krovetz, 1993; Kraaij y Pohlmann, 1996), aunque en el caso concreto del castellano habrá que probarlo experimentalmente. Por lo que se refiere a la lematización derivacional, se trata de un tema controvertido, y más aún dependiendo de cuál sea la profundidad de la lematización. Se trata de una buena ocasión para observar sus efectos y extraer las conclusiones pertinentes.

De otro lado, se trabaja también en implementar capacidad multilingüe, en un primer estadio, con un esquema de consultas en castellano contra documentos en inglés. Aunque el planteamiento teórico puede ser aplicable a cualquier par de lenguas, es evidente que una implementación requiere conocimiento lingüístico importante y específico por parte del sistema, desde la utilización de léxicos y equivalencias en ambas lenguas, hasta herramientas específicas de desambiguación, que permitan seleccionar de forma automática las acepciones correctas en cada caso.

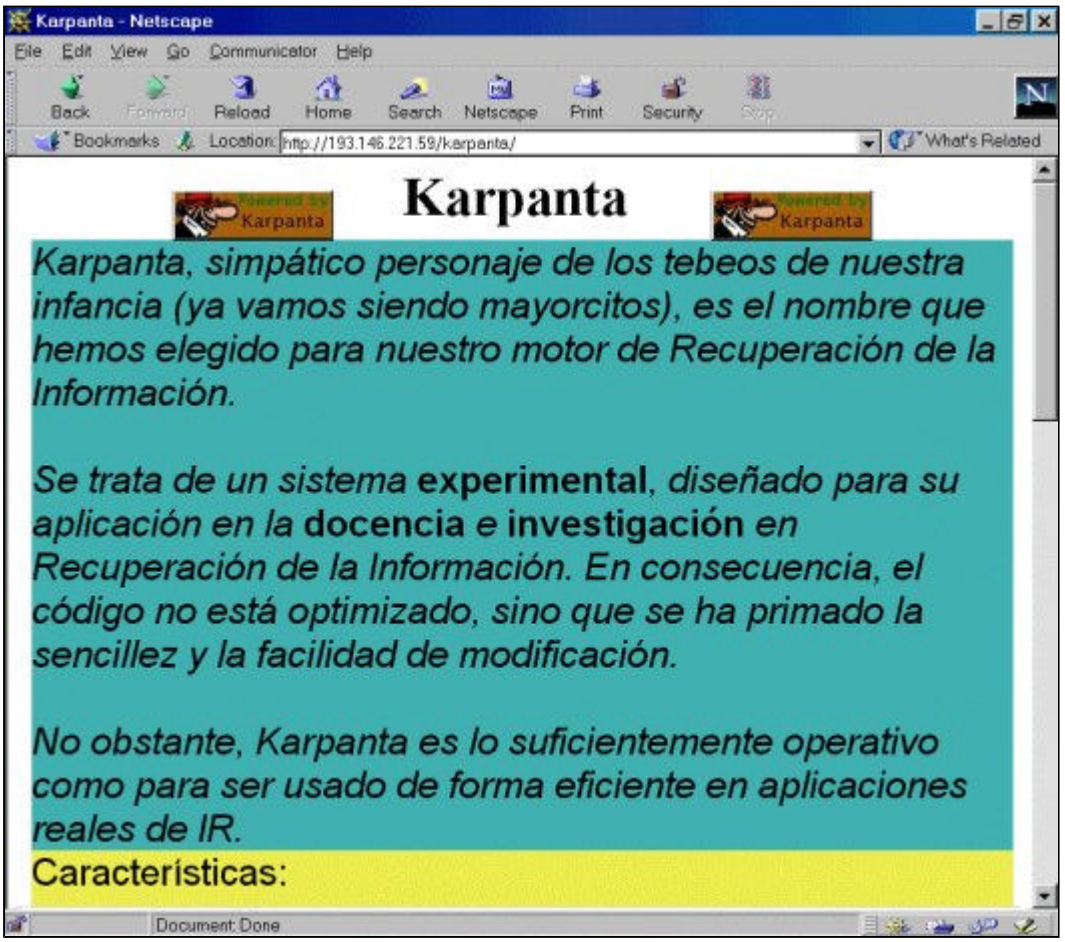

Ilustración 7. Página web de Karpanta en http://milano.usal.es/karpanta donde puede descargarse una *demo*

5 Conclusiones

Se ha diseñado un programa de recuperación de la información que aplica el modelo del espacio vectorial, lo suficientemente abierto y flexible para ser utilizado en labores docentes, así como de investigación. La sencillez de su arquitectura permite tanto la fácil observación de resultados y estructuras intermedias como la modificación y añadido de nuevos módulos y, por consiguiente, la experimentación. De hecho, está siendo utilizado en la docencia de algunas materias relacionadas directamente con la recuperación automatizada de la información, y también en diversos trabajos de investigación. Así, se utiliza para comprobar el efecto en la recuperación de nuevos sistemas de lematización para el castellano, que actualmente diseñan miembros de nuestro departamento; y es la base sobre la cual se implementará un sistema de recuperación multilingüe (castellano-inglés) sobre el cual trabajamos en la actualidad.

A pesar de ser estos los objetivos prioritarios perseguidos en la elaboración del sistema, el programa tiene suficientes prestaciones para resultar útil en entornos operacionales, siendo sencilla su adaptación a distintas necesidades. Un ejemplo es la base de datos especializada en Ciencias de la Documentación Datathéke (http://milano.usal.es/dtt.htm), consultable a través del web, y que, internamente, usa Karpanta como motor de recuperación.

Bibliografía

Buckley, C.; Salton, G.; Allan, J. (1994). «The effect of adding relevance information in a relevance feedback environment». En: *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM, p. 292-300.

Codina, L. (1994). «Bases de datos relacionales: qué son y qué aportan a la gestión documental». *Information World en español*, núm. 29 (1994), p. 18-19.

Date, C. J. (1983). *An introduction to database systems*. Reading (MA): Addison-Wesley.

Gómez Díaz, R. (1998). *La recuperación de la información en español: evaluación del efecto de las peculiaridades lingüísticas*. Trabajo de grado (tesina), Universidad de Salamanca, Departamento de Informática y Automática.

Grossman, D. A.; et al. (1995). «Integrating structured data and text: a relational approach». *JASIS*, vol. 46 (1995).

Grossman, D. A.; et al. (1996). «Improving accuracy and run-time performance for TREC-4». *TREC-4*, NIST Special Publication, núm. 500-236 [en línea] <http://trec.nist.gov/pubs/trec4/papers/gmu.ps.gz> [Consulta: 14 febr. 2000].

Grossman, D. A.; et al. (1997). «Using relevance feedback within the

relational model for TREC-5». *TREC-5*, NIST Special Publication, núm. 500-238 [en línea] <http://trec.nist.gov/pubs/trec5/papers/gmu.trec5.ps.gz> [Consulta: 14 febr. 2000].

Harman, D. (1991). «How effective is suffixing?». *JASIS*, vol. 42, núm. 1 (1991), p. 7-15.

Harman, D. (1992a). «Ranking algorithms». En: *Information retrieval: data structures and algorithms*. Englewood Hills (NJ): Prentice-Hall, p. 363-392.

Harman, D. (1992b). «Relevance feedback and others query modification techniques». En: *Information retrieval: data structures and algorithms*. Englewood Hills (NJ): Prentice-Hall.

Hull, D. (1996). «Stemming-algorithms - a case study for detailed evaluation». *JASIS*, vol. 47, núm. 1 (1996).

Kraaij, W.; Pohlmann, R. (1996). «Viewing Stemming as Recall Enhancement». En: *ACM SIGIR 96*. Konstanz (DE), ACM, p. 40-48.

Krovetz, R. (1993). «Viewing morphology as an inference process». En: *Proceedings of ACM-SIGIR 93*, p. 191-203.

Moya Anegón, F. de (1995). *Los sistemas integrados de gestión bibliotecaria*. Madrid: ANABAD, 1995.

Popovic, M.; Willet, P. (1992). «The effectiveness of stemming for natural-language access to slovene textual data». *JASIS*, vol. 43, (1992) p. 384-390.

Rijsbergen, C. J. van. (1979). *Information retrieval*. London: Butterworths.

Rocchio, J. J. (1971). «Relevance feedback in information retrieval». En: Salton, G. (ed.). *The SMART retrieval system*. Englewood Hills (NJ): Prentice-Hall, p. 313-323.

Salton, G. (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Reading (MA): Addison-Wesley.

Salton, G.; Buckley, C. (1988). «Term-weighting approaches in automatic text retrieval». *Information processing & management*, vol. 24, núm. 5, (1988) p. 513-523.

Salton, G.; McGill, M. (1983). *Introduction to modern information retrieval*. New York: McGraw-Hill.

Trigueros Díaz, J. L.; Higuera Matas, R. (1997). «Bases de datos relacionales versus bases de datos documentales». *Boletín de la Asociación Andaluza de Bibliotecarios*, vol. 13, núm. 49, (1997) p. 43-57.

Wilbur, J.; Sirotkin, K. (1992). «The automatic identification of stop words». *Journal of information science*, vol. 18, (1992) p. 45-55.

Glosario

Se incluye a continuación una breve explicación sobre algunos conceptos utilizados en este trabajo, a fin de facilitar la comprensión por parte de lectores no demasiado especializados en estos temas:

Algoritmo de Rocchio: cuando se expande de forma automática una consulta realizada por un usuario, ésta puede ser realimentada utilizando aquellos documentos recuperados por la consulta inicial que el usuario señala como relevantes. En estos casos, es preciso recalcular los pesos o importancia de los términos de la nueva consulta, o consulta expandida. El llamado algoritmo de Rocchio es un sistema de cálculo de dichos pesos, ampliamente utilizado. Desde un punto de vista práctico, su principal ventaja es que permite ajustar la importancia que se desea dar a los términos de los documentos relevantes de la consulta original, y también (en sentido negativo, obviamente) a los de los documentos que no se consideran relevantes.

IDF (*Inverse Document Frequency*): expresión que en ocasiones se ha traducido como Frecuencia Inversa de Documentos o también como Frecuencia Inversa en el Documento, aunque en realidad no es ninguna de estas dos cosas. Se trata de una función que trata de ponderar el valor informativo de un término en general, basándose en su frecuencia de uso o aparición. Se basa en la idea de que ese valor informativo es inversamente proporcional a su frecuencia de uso (de ahí lo de *Inverse*). De otro lado, para evitar distorsiones en la apreciación de esa frecuencia, pues un término podría aparecer muchas veces pero en pocos documentos, dicha frecuencia se mide contando el número de documentos en los cuales aparece ese término, y no las veces en bruto que aparece en toda la colección de documentos.

Lematización: se trata de la operación de hallar el lema del cual se deriva una palabra concreta. En este contexto, su finalidad es agrupar palabras que tienen un contenido semántico muy próximo; por ejemplo: *biblioteca, bibliotecas, o bibliotecario.* Esta agrupación permite comparar o encontrar directamente todas las palabras que tienen el mismo lema, independientemente de que el usuario busque sólo por una de ellas; además, corrige los cálculos de frecuencias y demás, contabilizando lemas y no palabras particulares.

Lematización derivacional: tiene en cuenta la generación de derivados a partir de la adición de diferentes sufijos o prefiijos; por ejemplo, *biblioteca* y *bibliotecario*. Aunque, obviamente, agrupa un mayor número de palabras bajo un mismo lema, tiene dos inconvenientes serios: el primero, que no es nada fácil de hacer de forma automática, al menos para lenguas morfológicamente complejas, como son las que proceden del latín, que es nuestro caso. El segundo inconveniente es que, con frecuencia, la distancia semántica entre palabras derivadas puede resultar demasiado grande; por ejemplo, la palabra *camilla*, claro derivado de *cama*, pero que, sin embargo, en una de sus acepciones más corrientes se refiere no a una cama, sino a una mesa.

Lematización flexional: es la lematización que agrupa bajo un mismo lema solamente las palabras que varían debido a accidentes morfológicos; para

los sustantivos, el cambio de número de singular a plural y viceversa (por ejemplo, agruparíamos *biblioteca* y *bibliotecas*); para los adjetivos, alteraciones en género y número; y para los verbos –lo más complicado, debido a la gran cantidad de irregulares-, además de género y número, hay que tener en cuenta las alteraciones de persona, tiempo y modo.

Modelo vectorial: se trata de una representación abstracta de documentos y consultas, que permite definir las características de ambos, así como calcular el grado de semejanza o similaridad entre unos y otros. Definido a partir de los trabajos de G. Salton en los primeros años 70, ha sido y es ampliamente utilizado en sistemas de recuperación. Básicamente, representa cada documento en una colección mediante un vector de *n* elementos, donde *n* es el número de términos indizables susceptibles de ocurrir o aparecer en cualquier elemento de la colección. Mediante diferentes sistemas de cálculo, a cada término o cada elemento del vector de cada uno de los documentos se le asigna un valor numérico o peso, que pretende significar la importancia o valor informativo de ese término en ese documento. Una consulta dada, formulada en lenguaje natural, puede representarse también mediante el mismo sistema, es decir, mediante otro vector de los mismos *n* elementos, cada uno de los cuales contiene el peso de cada uno de los términos de dicha consulta. En consecuencia, es fácil calcular cualquiera de las funciones utilizadas habitualmente para establecer la similaridad entre dos vectores, aplicándola entre el vector de la consulta y los de cada uno de los documentos. El resultado es un valor numérico que pretende indicar el grado de ajuste o semejanza entre la consulta y cada uno de los documentos; de forma que, aquellos documentos que arrojen una cifra más alta, serán los que más se ajusten a la consulta formulada.

Stemmer: es un programa (o su correspondiente algoritmo) que realiza la operación llamada *stemming*, una suerte de lematización, pero consistente en obtener la raíz a la cual se «pega» un determinado sufijo. Como puede adivinarse, se utiliza profusamente con documentos y consultas en inglés y, con esta lengua, parece dar buenos resultados. Un **s-stemmer** es uno de estos programas pero que, simplemente, elimina la s final de las palabras, con la intención de quedarse con la forma singular; suelen incorporar una lista de excepciones, con plurales irregulares; en castellano, y por lo que se refiere a sustantivos y adjetivos, aunque algo más complicado que quitar una letra, es relativamente fácil construir un programa de este tipo que reduzca tales palabras a su forma singular.