Portada

Guardado en base de datos.

Uno de los principales inconvenientes a la hora de implementar un motor de búsqueda, es la gestión de la memoria. Los datos que hay que utilizar son demasiado pesados, por lo que es muy poco recomendable mantenerlos todos en memoria, puesto que ralentizaría todo el sistema, pudiendo llegar incluso a dar fallo por salirse del “heap” (zona de memoria reservada para los datos).

Por ello decidimos guardar los datos utilizados en una base de datos.

Elección de la base de datos.

Hemos barajado dos posibilidades para la base de datos, MySQL y MongoDB.

En este caso MySQL no es una herramienta adecuada, puesto que el volumen de datos que queremos manejar es muy grande y se consulta varias veces la base de datos al realizar búsquedas, por lo que el tiempo necesario para guardar los datos en la BD y en consultarlos es demasiado alto, al tener que insertar y consultar fila a fila en la base de datos.

Por otro lado, MongoDB almacena documentos, por lo que es mucho más rápido a la hora de insertar grandes volúmenes de datos. Y para consultarlos podemos trabajar con los documentos directamente, sin tener que acceder a la BD por cada fila que queramos consultar, con lo que ahorramos mucho tiempo.

Por estas razones, hemos elegido la base de datos MongoDB para el proyecto.

Preparación de la base de datos.

Antes de comenzar a insertar los datos realizamos el diseño que queríamos implementar.

Para el proyecto decidimos utilizar una base de datos con 3 colecciones.

La primera colección “diccionario” está compuesta por múltiples documentos, cada uno de ellos hace referencia a un documento HTML de los utilizados para crear el diccionario.

Los documentos HTML tienen la siguiente estructura:

|  |  |
| --- | --- |
| ID | “nombre del html” |
| Palabra 1 | Valor TF |
| Palabra 2 | Valor TF |
| Palabra 3 | Valor TF |
| Palabra X | Valor TF |

Siendo el ID el nombre del documento original HTML, para poder identificar con qué documento estamos trabajando.

Palabra X es la clave que utilizamos para guardar los valores TF de todas las palabras del documento.

Valor TF es el valor de la palabra del documento.

Con esta estructura creamos el documento en memoria rellenando cada palabra con su valor TF y cuando está completo lo introducimos en la colección, de forma que solo hay 1 acceso a BD por documento HTML.

La segunda colección “idfColl” contiene un único documento llamado IDF.

El documento IDF tiene la siguiente estructura:

|  |  |
| --- | --- |
| ID | IDF |
| Palabra 1 | Valor de IDF |
| Palabra 2 | Valor de IDF |
| Palabra X | Valor de IDF |

El documento tiene como ID el valor IDF puesto que representa estos valores para las palabras. Siendo cada Palabra X una palabra de todas las recogidas en los anteriores documentos HTML, y su correspondiente valor IDF.

La tercera colección “consultas” contiene dos documentos.

El documento “union”, que hace referencia al documento original “union.trel” con la siguiente estructura

|  |  |
| --- | --- |
| ID | union |
| 2010-001 2010-26-075 | 0 |
| 2010-007 2010-54-054 | 1 |
| 2010-007 2010-68-054 | 0 |
| IDConsulta+ + IDDocumento | relevancia |

De esta manera mantenemos en la base de datos el documento union que nos aporta la relevancia de los documentos sobre las consultas.

El documento “topics” hace referencia al original “topics.xml” con la siguiente estructura

|  |  |
| --- | --- |
| ID | topics |
| 2010-001 | What processor obtained the best score in 2009 for the Photoshop benchmark? |
| 2010-002 | What graphics card obtained the best score in 2008 for the Battlefield 2142 benchmark? |
| IDConsulta | Consulta |

De esta forma mantenemos en la base de datos todas las consultas que se van a realizar con su ID.

Funcionamiento

Tras diseñar e implementar la base de datos quedaba utilizarla.

La hemos utilizado para guardar los datos antes mencionados y para consultar al realizar búsquedas.

Guardado de datos: Para guardar los datos hemos utilizado la clase CreadorDiccionario, que se encarga de limpiar los HTML, formatearlos y generar los documentos que se irán guardando en la BD. A su vez va creando y actualizando el documento IDF que se guarda tras leer todos los HTML (de forma que solo se introduce 1 vez en toda la creación). También hemos utilizado la clase LectorMetricas para guardar los ficheros union y topics.

Consulta de datos: el diseño elegido permite realizar una consulta eficiente de los datos, de forma que solo es necesario mantener en memoria los documentos que se quieran utilizar, es por ello que, a la hora de consultar datos, se accede a la BD, se recupera el documento que se quiera consultar (en función del ID) y tras utilizarlo se elimina para que deje de ocupar memoria. Con esto reducimos la cantidad de memoria utilizada al no cargar el diccionario completamente y ganamos en tiempo puesto que tarda menos en recuperar 1 archivo pequeño que todos los archivos HTML almacenados en la BD.

Expansión de consulta

Como mejora para el proyecto hemos incluido la expansión de consulta. Para ello hemos utilizado Lucene Wordnet, que nos proporciona las funcionalidades para expandir las consultas por medio de los sinónimos.

Hemos incluido el diccionario de sinónimos en el proyecto con el nombre “wn\_s.pl”.

Para expandir la consulta hemos añadido los sinónimos que nos ofrecía Wordnet a la consulta original, y la hemos tratado de la misma forma que los HTML que forman el diccionario. De manera que las palabras del diccionario y de la consulta mantienen el mismo formato.

Funcionamiento del programa

El programa tiene un menú de inicio que permite al usuario realizar diferentes acciones.

1. La primera es la creación/ actualización de la base de datos. Esta opción tarda alrededor de 25 minutos, puesto que carga todos los HTML, el idf, el union, y el topics en la base de datos. Todos estos documentos ya se encuentran en el proyecto por lo que no es necesario tocar el código fuente.
2. Ejecutar consultas. Esta opción ejecuta todas las consultas que se encuentren almacenadas en la base de datos.

Primero carga las consultas de la base de datos, la expande y la formatea.

Después, con las consultas originales y las expandidas, calculamos la relevancia de los documentos para cada consulta utilizando la función coseno TF-IDF. De todos los documentos nos quedamos con los 100 más relevantes para la evaluación del motor. Puesto que el cutoff más alto que utilizamos es de 100.

Para la evaluación del motor utilizamos las métricas vistas en la asignatura:

- Recall 5/10: evalúa el silencio de los resultados.

- Precision 5/10: evalúa la precisión de los resultados.

- Fvalue 5/10: evalúa el silencio frente a la precisión de los resultados.

- ReciprocallRank 1/2: evalúa la posición del primer relevante de los recuperados.

- AveragePrecision: evalúa la precisión media

- NDCG:

Tras obtener los resultados se genera un fichero y se guarda.

9. Finalizar programa. Esta opción finaliza la ejecución de programa.