16

**Recuperación de Información**

**Sistema de Recuperación Tradicional**

**Jorge Andrés Galindo 679155 Javier Aranda García 679184**

**ANÁLISIS DE LOS DOCUMENTOS**

Previamente, antes de realizar el proceso de indexación, se realizó un exhaustivo examen de todos los documentos, para conseguir la máxima información posible que ayudara a realizar el programa más completo y más adaptado a la información. El primer campo a analizar fue el nombre y número de etiquetas presentes en cada uno de los ficheros XML. Para ello, se observó que había como máximo 7 etiquetas diferentes entre todos los documentos, y que además, todos poseían estas 7, salvo 4 documentos que no tenían idioma. Las etiquetas encontradas fueron “identifier”, “language”, “creator”, “title”, “description”, “publisher” y “date”.

Sabiendo los campos de los documentos, se pasó a obtener información de cada uno de ellos por separado. En primer lugar, se vio que todos los documentos tenían al menos dos identificadores distintos, y que en todos salvo en 8 documentos uno de ellos estaba repetido, por lo que se debía pensar si guardarlo dos veces. A continuación, se prosiguió con el idioma. La etiqueta “language” indicaba en qué idioma había sido escrito el trabajo que representaba el documento XML. Se observó que como máximo solo tenían una etiqueta de este tipo (salvo algún documento que no tenía ninguna), y que además tenían valores tales como “eng”, “spa”, “por”,”fre”…

En lo que a la etiqueta “creator” se refiere, todos los documentos tenían al menos un campo de este tipo, pero en la gran mayoría de ellos se nombraba a más de una persona responsable de realizar ese trabajo (hasta 6 creadores en un mismo documento). Analizando la etiqueta “title”, como era de esperar, estaba contenida en todos los documentos a indexar, salvo que se vio que en dos casos estaba repetida en el mismo documento. Con la etiqueta “description” pasaba algo parecido, salvo que aquí estaba repetida pero en dos idiomas distintos.

Mirando detenidamente el “publisher”, se observó que estaba en todos los documentos, y que tomaba hasta tres valores diferentes: “Universidad de Zaragoza”, “Universidad de Zaragoza, Universidad de Zaragoza” y “Universidad de Zaragoza, Prensas de la Universidad”. Finalmente, la etiqueta que faltaba por analizar era “date”, que simplemente era un número de cuatro cifras que representaba el año en el que había sido realizado el trabajo, y que además, estaba sólo una vez presente en cada uno de los documentos de la colección.

**INDEXACIÓN**

Tras analizar los documentos detenidamente, se procedió a indexar la información. En primer lugar, se debatió cómo organizarla en distintos índices o campos, llegando a la conclusión de que lo más cómodo era hacer tantos índices como etiquetas distintas podían tener los archivos XML (salvo una excepción). Se pensó si se podía desechar alguno de los campos, como podría ser el “publisher”, ya que tomaba valores muy parecidos, y la colección de documentos era tan sólo de la Universidad de Zaragoza; sin embargo, se optó por dejarlo por si en algún momento se añadían documentos de otras universidades.

El campo “identifier”, como ya se vio en el análisis previo de los documentos, constaba de dos códigos identificativos distintos, y además, en muchos documentos estaba repetido uno de ellos. Se almacenaron los dos códigos en el mismo campo del índice sin repetir aquel que estuviera más de una vez en los documentos. Para la etiqueta “creator”, se decidió no crear un campo propio y guardarla junto con “description” ya que no parecía obvio detectar un nombre propio o apellido en una consulta, así que daría mejores resultados juntarlo todo en un mismo campo que separarlo. Además, es muy frecuente que un usuario escriba su necesidad de información sin mayúsculas, lo que haría aún más difícil detectar los nombres propios. Los creadores de un trabajo van concatenados junto a la descripción uno detrás de otro.

En lo que al “title” se refiere, como se comentó en el análisis previo, tan sólo había uno por documento, salvo en contadas ocasiones que había dos pero el mismo repetido. En este caso, se siguió una estrategia similar que para “identifier”, ignorando el repetido. Además, al igual que “creator”, se juntó también en un mismo campo con “description”, ya que era otro de los campos de los que no se podía extraer patrones con facilidad. La fecha o “date” se almacenó en un campo propio ya que no daba ningún problema en los documentos. Para el “publisher”, como ya se ha comentado, se tomó la decisión de almacenarlo tal cual en un campo del índice, sin modificar nada.

Para el campo “language”, se decidió entre guardarlo tal cual se leía en su campo, o realizar una modificación y guardar la correspondencia del código (“eng” pasarías a inglés). Finalmente, se optó por guardar la correspondencia con el nombre completo del idioma, ya que para las consultas el parser podía dar problemas con la búsqueda del código, siendo más cómodo buscar con el nombre entero del idioma. Además, aquellos 4 archivos sin idioma, se revisaron y se decidió que si el indexador no encontraba etiqueta de idioma pusiera español “a mano”. Esta decisión se tomó porque aquellos de la colección que no tenían, estaban en español, y la probabilidad de que futuros documentos sin idioma fueran españoles era más alta que de cualquier otro idioma.

Finalmente, el último campo a tratar fue “description”, y el que más problemas dio. Se observó en el análisis previo que existían documentos que lo tenían repetido y en dos idiomas (mayoritariamente en español y en inglés). La solución tomada para otros campos como podía ser almacenarlos concatenados se desechó porque las consultas serían realizadas casi en la totalidad de ocasiones en español. Así pues, se decidió indexar solo el texto en español en caso de que tuviera más de dos etiquetas “description”, que en casi todos los documentos era el primero encontrado (unido a los creadores comentados anteriormente). Así pues, el campo “description” del índice contendría los contenidos de las etiquetas “titlte”, “creator” y “description” de los documentos a indexar.

Además, se observó que en tanto en la etiqueta “title”, como en la etiqueta “description”, aparecían también fechas. Como se tenía previsto realizar consultas por rangos de fechas, se decidió analizar cada una de las palabras de estas dos etiquetas en busca de fechas, considerando fecha un número entero de 4 cifras, y en caso de encontrarla, indexarla en un nuevo campo del índice que se llamaría “fechaTexto”, y que permitiría las ya comentadas búsquedas de fechas por rangos.

Una vez decididos los campos a utilizar y delimitado el filtro a realizar con el contenido de cada documento a guardar, se pasó a decidir qué tipo de campos eran necesarios para almacenar cada uno de los índices. La decisión estaba clara y era de tipo texto en la mayoría de los casos, ya que se iba a realizar consultas de tipo “MultiField” que aceptaba dos array de “String” con campos y texto a buscar. Sin embargo, para los campos “date” y “fechaTexto” que contenían años de fechas, se quería permitir realizar búsquedas de rangos, así que se decidió utilizar el tipo “IntField”.

El campo de tipo texto podía ser de tipo “TextField” o “StringField”. El tipo “StringField” es un campo que se indexa y no se tokeniza, guardado la cadena de texto como un único token. Sin embargo, el campo “TextField” se indexa y tokeniza, por lo que para textos grandes en los que se quiere coincidencia de palabras y no del texto entero sería más adecuado. La decisión que se tomó fue usar “TextField” con un analizador de tipo “SpanishAnalyzer” que, aunque diera peores resultados que otros en campos como el identificador o en textos en inglés, en su totalidad era el que mejor resultado daría para las futuras consultas sobre el indexado.

|  |  |  |
| --- | --- | --- |
| **ETIQUETA** | **CAMPO** | **TIPO** **CAMPO** |
| description | description | TextField |
| language | language | TextField |
| date | date | IntField |
| creator | description | TextField |
| identifier | identifier | TextField |
| publisher | publisher | TextField |
| title | description | TextField |

\* Campo extra llamado “fechaTexto” de tipo “IntField” para fechas en “title” y “description”.

**CONSULTAS**

Para la realización de las consultas, en primer lugar, se realizó una lectura del fichero XML en el que se encontraban las necesidades de información a buscar, guardando tan solo los campos identificador y contenido con la necesidad de información. A continuación, para cada una de las consultas, se decidió realizar dos pasadas. Una primera sobre el texto original sin normalizar, y una posterior sobre la secuencia de tokens normalizada por el parser (eliminando palabras vacías definidas por el analizador más algunas extra que se consideraron que no aportaban información).

La primera pasada consistía en buscar patrones para realizar las búsquedas por rangos en los campos “date” y “fechaTexto”. Estos patrones se dividieron en intervalos, años posteriores, años anteriores y año exacto. La segunda, esta vez sobre los tokens normalizados, servía para buscar también patrones, pero en este caso sobre el resto de campos. Se buscaban coincidencias de cada token sobre un idioma, un “publisher” o un identificador. En el caso de que algún token compliera el patrón, se almacenaba para realizar una búsqueda específica en su campo.

En cuanto a la búsqueda en general, se decidió realizarla entera sobre el campo “description”, que englobaba las ya nombradas etiquetas “creator”, “title” y “description”. Esta decisión se tomó ya que la mayor parte de la información estaría contenida en dicho campo, complementando con las búsquedas en otros campos si se encontraban los ya nombrados patrones. Una vez obtenida la consulta con los distintos campos, se procedió a unirla. La consulta se formó con una query “multifield” con todos los campos de tipo texto (“description” y los encontrados por patones) más una o varios queries de rangos para las fechas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **POSTERIORES** |  | **INTERVALO** |  | **ANTERIORES** |  | **EXACTO** |
| a partir del año A (incluido y excluido) |  | entre A y B |  | anteriores a A |  | en A |
| a partir de A (incluido y excluido) |  | desde A hasta B |  | anterior a A |  | en el año A |
| posterior a A |  | desde A a B |  | últimos A años |  | de A |
| posteriores a A |  | de A a B |  |  |  |  |
|  |  | de A hasta B |  |  |  |  |

**ALGORITMO RANKING**

Adicionalmente, se ha utilizado el modelo probabilístico BM25 ya que, tras realizar algunas pruebas, se comprobó que se obtenían mejores resultados para las necesidades de información dadas que el modelo vectorial. Se estudió la posibilidad de combinar ambos modelos, usando el probabilístico en los campos de tipo texto, y el vectorial en los numéricos, pero tras realizar algunas pruebas no se vio viable. El modelo se indica a través de un método al objeto encargado de las búsquedas “IndexSearcher”.

**RESULTADOS**

Una vez realizados todos los pasos anteriores, se procedió a probar el buscador con las necesidades de información que se nos proporcionaron anteriormente. La primera necesidad de información consistió en la búsqueda de información geológica de España. Los 5 primeros documentos obtenidos proporcionan exactamente la información que se pidió. Por lo tanto, la consulta se ha realizado de forma correcta obteniendo resultados satisfactorios.

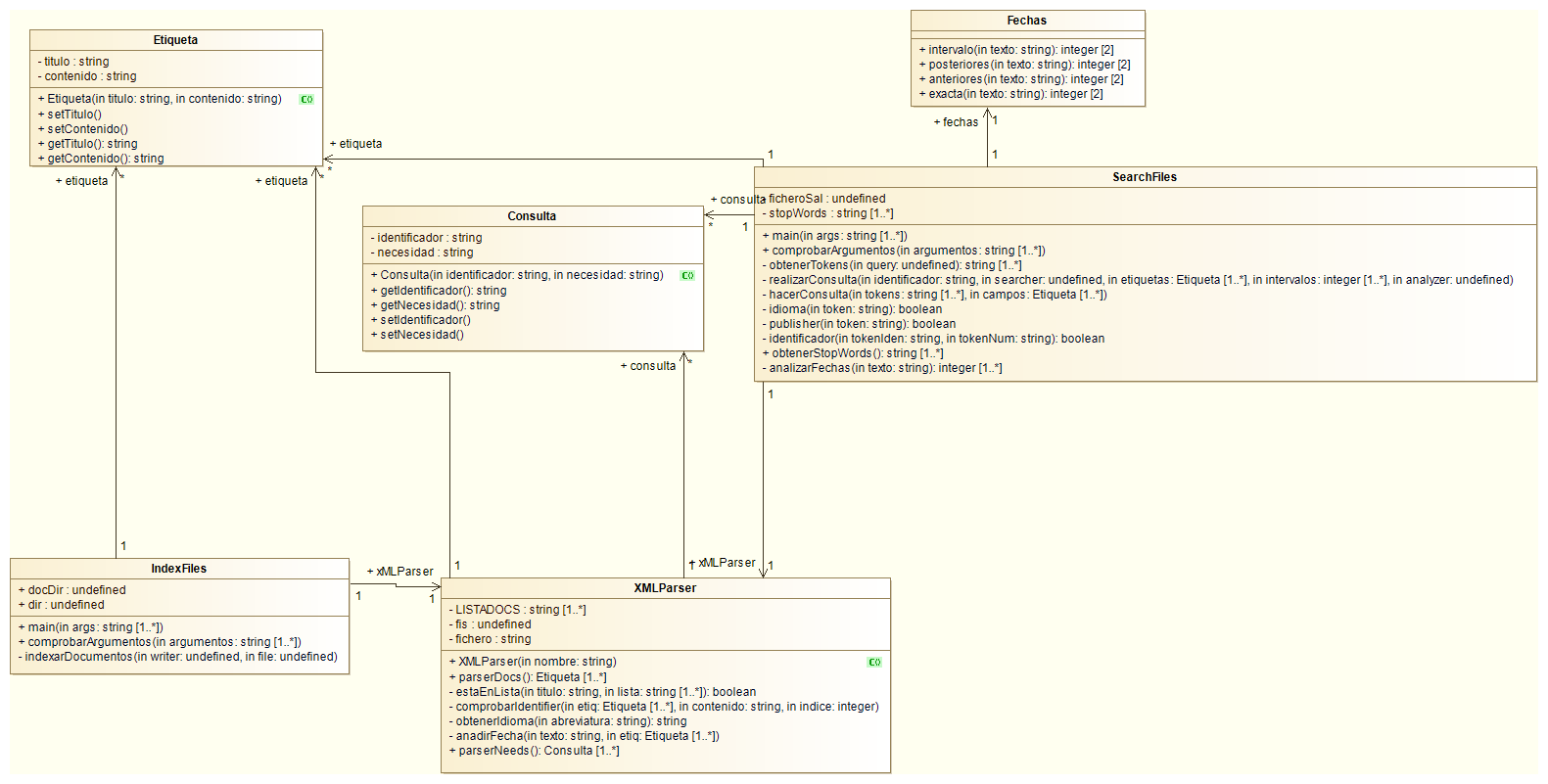
La segunda necesidad de la información consistió en la búsqueda de información sobre enfermedades infecciosas que hayan tenido lugar entre 1980 y 1990. En esta ocasión, uno de los 5 primeros documentos obtenidos no trata sobre el tema solicitado (7108); sin embargo, los años que se piden en la consulta aparecen en el documento, razón por la cual ha aparecido en una posición tan alta en el ranking. En cuanto a los otros 4 documentos, todos informan sobre el tema solicitado y uno de ellos encaja con el rango de fechas solicitado.

La tercera necesidad busca información sobre algoritmos geométricos. En este caso los primeros 5 documentos encajan con lo que se pidió en la consulta a la perfección, por lo tanto no hay mucho más que señalar. Al no especificar ninguna fecha, el ranking se ha basado tan sólo en el resultado del método probabilístico sobre el campo “description”, que englobaba “creator”, “description” y “title” del documento XML original.

La cuarta necesidad busca trabajos relacionados con robótica a partir de 2005 y realizados por un Martínez. En esta ocasión, todos los ficheros encajan con las fechas solicitadas y varios de ellos han sido realizados por un Martínez. Sin embargo, ninguno de los 5 primeros documentos encaja a la perfección con el tema que se ha solicitado, obteniendo malos resultados.

La quinta necesidad busca trabajos relacionados con inteligencia artificial realizados en los últimos 5 años. En este caso todos los documentos encajan con el tema solicitado (inteligencia artificial) y sólo dos de ellos incumplen el rango de fechas demandado, ya que ambos documentos fueron redactados en 2010. Además, la aparición de estos dos documentos que incumplen las fechas es debida con total seguridad a que no existen más trabajos sobre el tema en el rango especificado.

**DIAGRAMA DE CLASES**



En lo que al diagrama de clases se refiere, están presentes las dos clases principales encargadas de la búsqueda e indexación (“SearchFiles” e “IndexFiles”). La clase “XMLParser” es la encargada de parsear los ficheros XML tanto de los documentos a indexar como de las necesidades de información. Además, se ha creado dos clases extras, una para encapsular objetos de tipo etiqueta usados para extraer la información de los archivos XML, y la otra para almacenar las consultas leídas (identificador y contenido). Finalmente, para no cargar excesivamente la clase “SearchFiles”, se han definido en una clase a parte los métodos necesarios para reconocer los patrones de fechas.