Documentación Electrónica
Práctica final: "Clustering"
Curso 2008/2009

Carlos Noreña Martín y David Rozas Domingo

27 de mayo de 2009

Índice

1. Descripción del problema

El objetivo de esta práctica consiste en diseñar e implementar un algoritmo de clustering de documentos XML que contienen noticias en español. Para ello se utilizará la librería Lucene, que permite generar fácilmente un motor de búsqueda basado en texto. Para comprobar la calidad de la solución se utilizará una evaluación externa mediante la Medidad F, que comparará con una solución que se considera ideal.

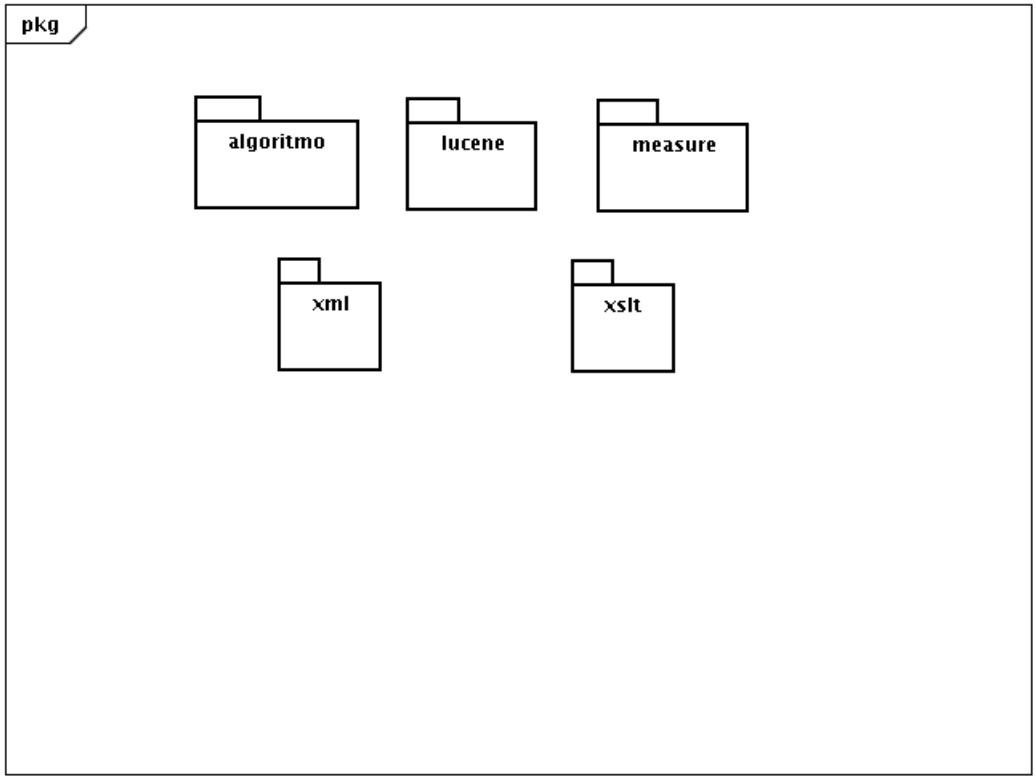
El resultado se transforma en un fichero XML, que para una mejor visualización es transformado en una página HTML a través de una hoja de estilo XSLT.

2. Diseño e implementación

A continuación se explican los detalles y decisiones más importantes que se han tomado durante las fases de diseño e implementación.

2.1. Paquetes

La aplicación ha sido dividida en diferentes paquetes acordes a cada tipo de funcionalidad (la imagen original se encuentra en: `/diagramas_uml/diagrama_clases.png`)



- `es.urjc.es.etsii.gavab.de.clustering`: contiene la clase principal desde la que se arranca la aplicación.
- `es.urjc.es.etsii.gavab.de.clustering.algoritmo`: contiene todas las clases necesarias para implementar el algoritmo de clustering propuesto.
- `es.urjc.es.etsii.gavab.de.clustering.lucene`: contiene las clases relativas al motor de búsqueda.

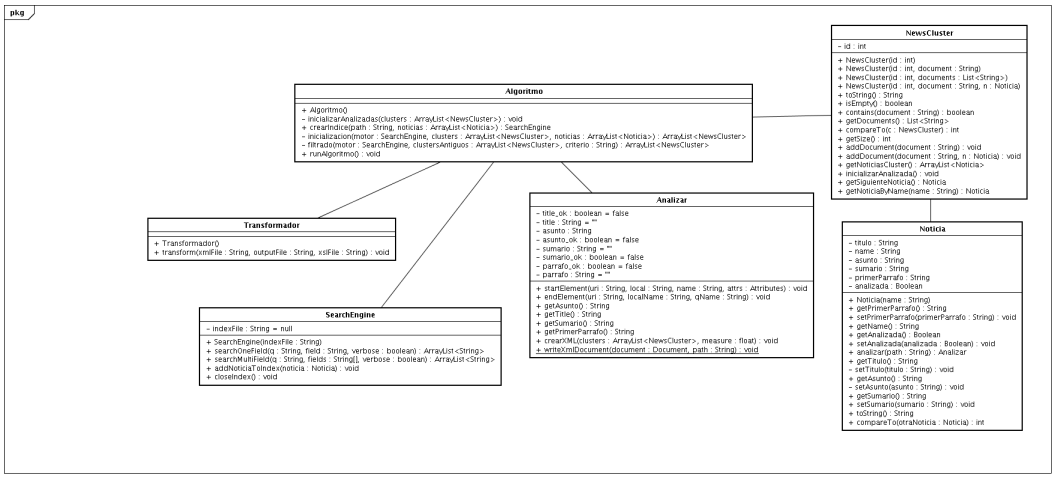
- `es.urjc.es.etsii.gavab.de.clustering.measure`: contiene las clases necesarias para llevar a cabo el proceso de medición de la calidad de la solución.
- `es.urjc.es.etsii.gavab.de.clustering.xml`: contiene las clases necesarias para el análisis y la creación de documentos XML.
- `es.urjc.es.etsii.gavab.de.clustering.xslt`: contiene las clases necesarias para realizar la tranformación del documento XML a HTML.

2.2. Descripción del algoritmo

El algoritmo consta de las siguientes fases:

- Inicialización: en la cuál se crea un cluster con todas las noticias ordenadas por nombre de fichero.
- Filtrado: La lista de clusters es filtrada acorde a ciertos criterios (primer párrafo, título, asunto, etc.). Para cada uno de ellos:
  - Tomamos la primera noticia sin analizar perteneciente a ese cluster, y realizamos una búsqueda.
  - Con todas las noticias devueltas que pertenezcan a dicho cluster, se creará un nuevo cluster.
- Este proceso se repite mientras que haya noticias sin analizar en el cluster.
- Nuestro algoritmo tiene una naturaleza divisiva no binaria, por tanto la primera fase de filtrado es mucho más determinante para el resultado final que las posteriores. La discusión acerca de la elección final de los criterios a aplicar se puede encontrar en apartados posteriores.

Las clases más importantes se muestran en el siguiente diagrama (la imagen original se encuentra en: `/diagramas_uml/diagrama_clases.png`)



Por motivos de eficiencia, se genera un único índice (filtrando los resultados que no pertenezcan al cluster en análisis) en el que se indexan los siguientes campos:

Campo	Almacenado	Analizado
Nombre	Sí	No
Título	No	Sí
Asunto	No	Sí
Sumario	No	Sí
Primer párrafo	No	Sí

Observaciones:

- En los campos analizados se utiliza el diccionario SpanishStopList para eliminar las palabras que forman parte de dicho diccionario.
- En los campos “Sumario” y “Primer párrafo” solamente se tienen en cuenta las entidades.
- Para documentar la aplicación, se ha generado Javadoc de los métodos y clases más importantes. Se encuentra en la carpeta `doc` del proyecto.

2.3. Parseado

El parseado de todas las noticias se ha realizado utilizando el API SAX de Java por motivos de eficiencia:

- Su enfoque orientado a eventos es útil, puesto que estamos procesando documentos en los que sólo se requiere una parte de la información.

- Consume menos memoria, al no tener que generar un árbol completo.
- La búsqueda de información (en el parseo) es más eficiente.

2.4. Clustering y criterios de búsqueda

Los criterios de búsqueda han sido seleccionados de una forma empírica. Se probaron diferentes campos, por separado y combinándolos, varias fases de filtrado, en diferentes órdenes, etc. Finalmente la mejor configuración obtenida (cuya medida-F es de alrededor **0.9**) se obtuvo con la siguiente configuración:

Nº Fase	Campos-consulta	Campos-análisis
1	Entidades de sumario y primer párrafo	Entidades de sumario
2	Entidades de primer párrafo	Entidades de primer párrafo

Debido a que nuestro algoritmo posee además una fase de inicialización en la que se genera un primer cluster con todas las noticias, el orden en que dichas noticias eran agregadas al cluster (dependiente del SO) influía en el resultado final. Por tanto decidimos ordenar previamente las noticias por su nombre de fichero.

El resultado de la Medida-F puede seguir variando ligeramente, como consecuencia de la precisión con la que se trata el tipo float en el SO en cuestión.

3. Transformación de los resultados

Tal y como se especificaba en los requisitos, el proceso de transformación de los resultados es el siguiente:

- Generamos un documento XML con el API DOM a partir del objeto que almacena la lista de clusters.
- Dicho documento XML es transformado mediante una hoja de estilo XSLT en un documento HTML que para cada cluster muestra los títulos de las noticias que pertenecen a él, con un enlace al fichero XML con la noticia original.
- Además dicho HTML contiene una referencia a una hoja de estilo CSS para obtener una presentación más visual.

Referencias

[hs09] http://www.escet.urjc.es/ smontalvo/de/material.html. Apuntes de clase. Mayo 2009.

[htt09a] http://lucene.apache.org/java/docs/. PÃ¡gina principal de lucene. Mayo 2009.

[htt09b] http://www.w3.org/TR/xslt. Xslt (pÃ¡gina del w3c). Mayo 2009.