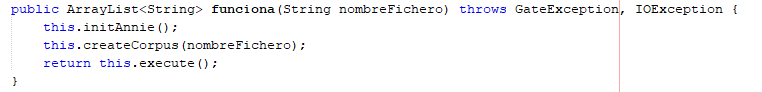
Mejoras del Sistema de Recuperación de Información

Raúl Garrido Iscar

2021

## Integración de la API de GATE

En el directorio del proyecto se añade la carpeta de ANNIE para que el método initAnnie() funcione. La configuración de annie se puede cambiar modificando los ficheros de la carpeta, ya que el gazetteer no etiqueta bien este corpus.



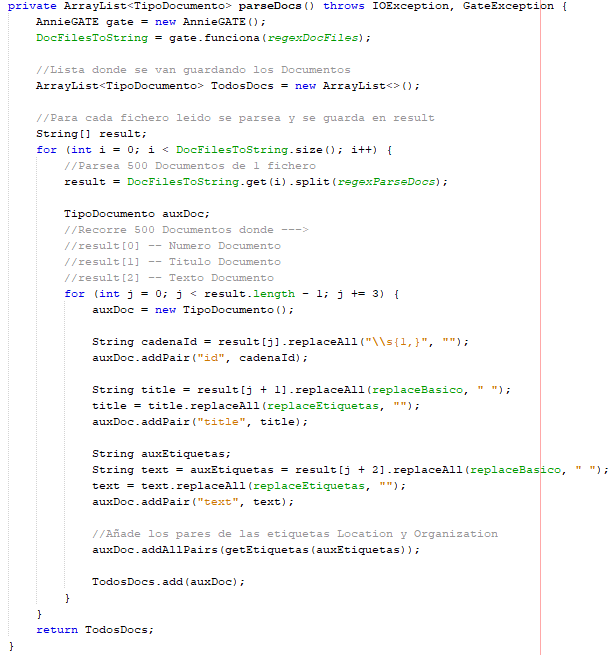
El método initAnnie() carga annie con la configuración, seguidamente en método createCorpus(nombreFichero) crea el corpus a partir de un Arraylist<String> que lee del fichero asociado al parámetro *nombreFichero*.



Tras la creación del corpus, el método execute() del objeto *annieController* es el que trata al corpus creado y le añade las etiquetas.

Finalmente se recorre el corpus y se devuelve un Arraylist<String> con XLM de cada uno de los documentos.

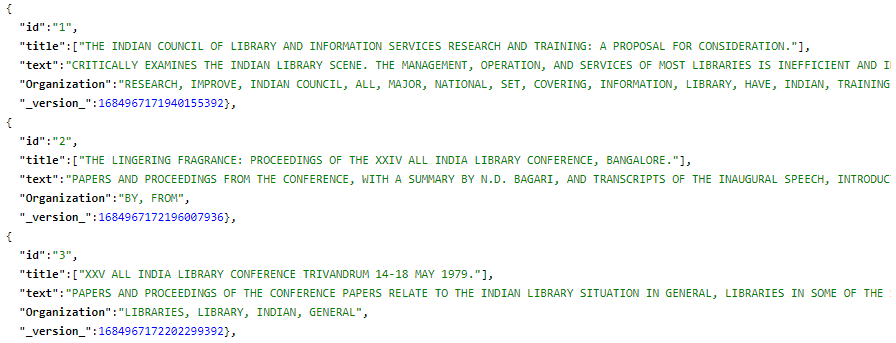
Ejemplo de Parseo de Documentos tras pasar por Annie.



El parseo funciona de tal manera que divide los documentos en 3 partes y cada una de ellas se trata de manera distinta posteriormente.

Finalmente acaba extrayendo de cada String, que sería el XLM de annie, ID del documento, Titulo del documento, texto del documento y las etiquetas encontradas.

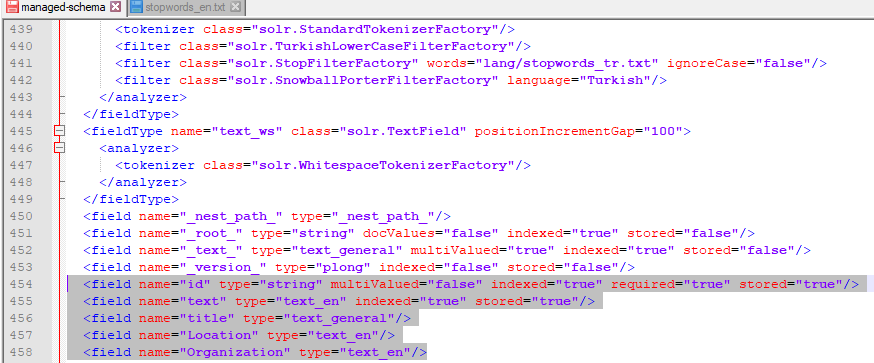
Ejemplo de cómo quedarían indexados en Solr. El texto se muestra completo, pero realmente está tratado para quitar *stopwords* y reducir las palabras a la raíz.



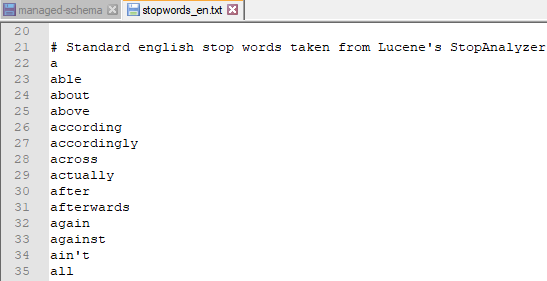
Como se ve, las etiquetas son bastante pobres en información y al realizar las consultas solo sirven algunas etiquetas de *Location*.

## Eliminación de Stopwords al indexar el texto

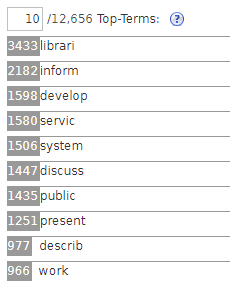
Modificación de los tipos dentro de cada campo que se indexa para que sean tratados como texto en inglés.



Ejemplo con las primeras Stopwords añadidas al fichero

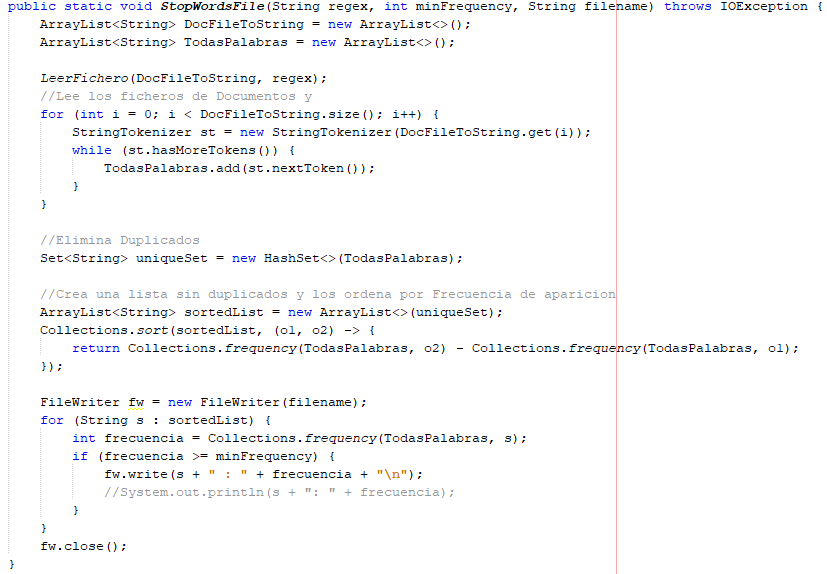


Ejemplo con las 10 palabras más repetidas del índice de Solr, para demostrar que se reducen a la raíz.



## Recopilación de palabras demasiado frecuentes (Ley de Zipft)

Esta mejora la implementé, pero finalmente no la añadí ya que, si quitaba estas palabras de los textos de los documentos, estos perdían demasiada información y hacían las consultas aún menos eficientes.

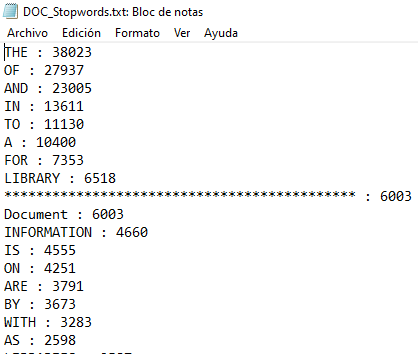


Este método tokeniza las palabras de los documentos del fichero asociado a la String *regex*. Una vez los tokeniza, crea un Set con la lista de palabras tokenizadas. El Set no contiene palabras duplicadas y mantiene internamente la frecuencia de cada palabra.

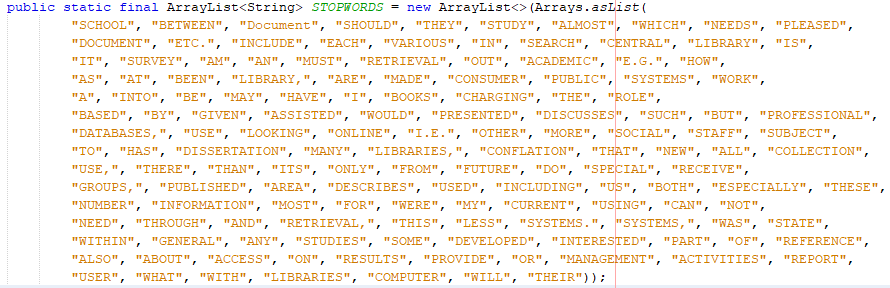
Por otra parte, la interfaz Collections tiene un método *sort* en el que se puede definir el comparador mediante una expresión lambda en la que uso la frecuencia de las palabras comparadas para ordenar. De esta manera queda finalmente una lista de palabras ordenadas por frecuencia que guardo en un fichero con el nombre del parámetro *filename*.

El número de palabras que se guardan viene definido en el parámetro minFrequency para no sobrecargar el fichero creado.

Ejemplo con las palabras más repetidas dentro de los documentos.



Como se puede comprobar muchas de estas palabras coinciden con stopwords, pero las palabras que no son stopwords que aparecen ahí deberían ser quitadas, ya que no aportan utilidad a la hora de distinguir documentos.



Antes de hacer que Solr borrase las stopwords probé a hacerlo yo usando las palabras de ese Arraylist, pero como dije al principio de este punto quitaba mucha información de los textos y lo que quedaba era muy específico y no funciona bien para buscar con las consultas que tenemos.

# Conclusión

El corpus es demasiado pobre en información como para hacer algo que funcione a menos que se modifique a Annie para sacar etiquetas realmente relevantes.