**PRACTICA 1 :**

**RECUPERACIÓN DE INFORMACIÓN**

Uso de software Tika.



4º Grado en Ingeniería Informática

**Realizado por Miguel Acedo y Pablo Mariano Moreno**

**INTRODUCCIÓN:**

En esta memoria explicaremos cómo hemos afrontado la práctica usando la herramienta tika para implementar una aplicación a través de la api disponible para java, que nos servirá para analizar y obtener información de los documentos situados en un path dado como parámetro de entrada.

En la aplicación se pueden distinguir dos varias partes bien diferenciadas. En primer lugar tendremos un main en la aplicación principal que nos servirá para controlar el flujo y ofrecer al usuario las distintas opciones implementadas en la aplicación. Además desde aquí se inicializan los objetos correspondientes a cada uno de los archivos leídos con toda la información obtenida.

Otra de las partes será las funciones a la que cada uno de estos elementos del menú llamará para realizar cada una de las funciones ofrecidas. Mostrará por pantalla lo correspondiente aludiendo a las funciones de los objetos.

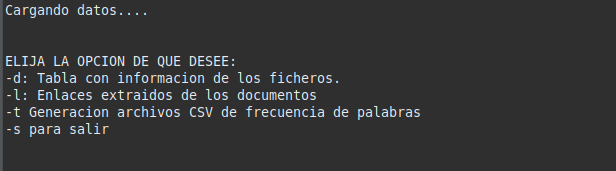
Por último en un fichero aparte hemos implementado una clase que contiene todos los datos obtenidos a través de la api de tika. Cuando este objeto se inicializa utilizará una variable tika global de la clase para obtener como se ha indicado en el guión los elementos necesarios en el programa principal. La mayoría han sido obtenidos gracias a los metadatos obtenidos con el parser de tika.

Algunos elementos obtenidos de los metadatos han necesitado un análisis o postprocesado anterior. En concreto el texto de los documentos. En primera instancia para obtener el lenguaje en el que se han escrito. Y en segundo lugar, a modo de procesado hemos necesitado eliminar signos de puntuación, tildes o mayúsculas.

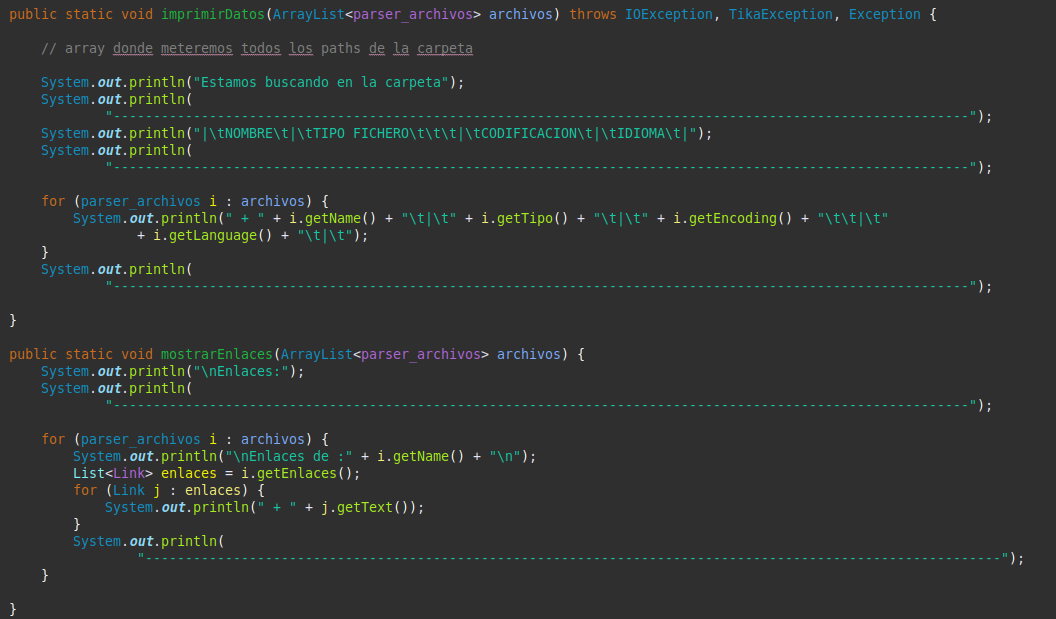
A continuación se explicará de manera más detenida cada uno de los apartados.

**PROGRAMA PRINCIPAL Y FUNCIONES:**

Inicialmente se nos presentará un menú de selección de las distintas opciones disponibles una vez todos los archivos se hayan cargado correctamente:



Al haber cargado los archivos de manera previa nada mas inicia el programa en sus clases correspondientes, la acción sobre el menú será inmediata. Cada una de estas funcionalidades llamará a la vez a la función correspondiente situada en el mismo fichero. Las funciones simplemente llamaran a los métodos públicos correspondientes de la clase parser mostrando lo que corresponda, a continuación se muestran las capturas correspondiente a esos trozos de código.



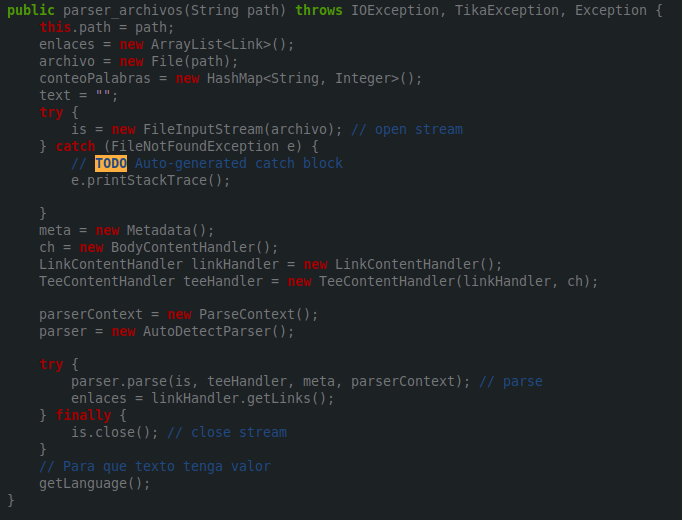


**CLASE PARSER\_ARCHIVOS:**

Esta clase tendrá un por un lado getters y setters clásicos para obtener toda la información requerida por el programa principal de una manera transparente al usuario.

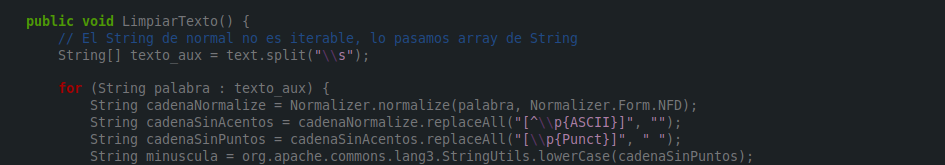
La parte en la que se realiza lo más importante respecto a la implementación con la API de Tika será en el constructor. De la manera que nosotros lo hemos pensado este recibirá el path del archivo correspondiente de manera que se creará un objeto file sobre el mismo. Además se inicializan algunas variables de la clase necesarias para el funcionamiento como pueden ser un Array para los enlaces, un hashmap para guardar las palabras que aparecen en los archivos y el conteo de las mismas.

Posteriormente creamos las variables necesarias para el parser siguiendo el guión de la práctica como se muestra en la siguiente captura:



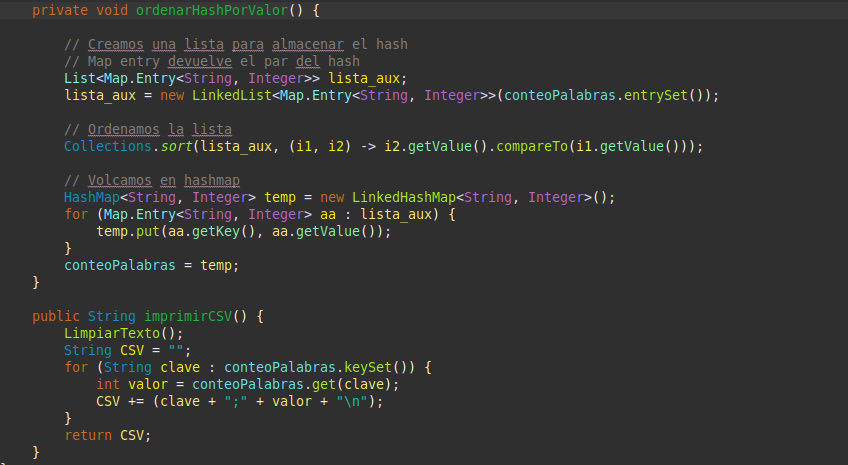
Otro de los aspectos que requiere mención es el proceso de formateo de texto para realizar posteriormente los archivos csv.

En primer lugar dividimos el String que obtenemos del texto en “sub String” para que posteriormente sea más fácil tener las palabras separadas y meterlas en el HashMap. Esto lo ha realizado mediante la función text.split. Este array de String resultantes será recorrido y se le aplicaran varias sentencias en primer lugar normalize y replaceall para quitar acentos, signos de puntuación y por último pasar todas las letras a minúscula.



Tras transformar la palabra en cuestión comprobaremos si ya hay otra palabra igual en el hash que tenemos, si existe aumentamos el valor asociado, en caso contrario la incorporamos con valor a 1.

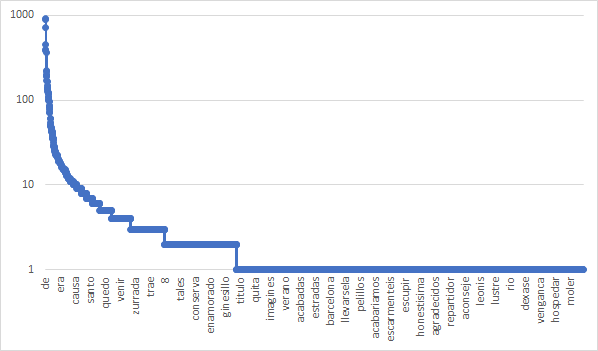
Una vez tenemos el hashmap completo para el archivo completo pasamos a ordenar el mismo. Usamos una función auxiliar que usa una lista para ir comparando con el hashmap inicial y así poder devolverlo ordenado a la clase.



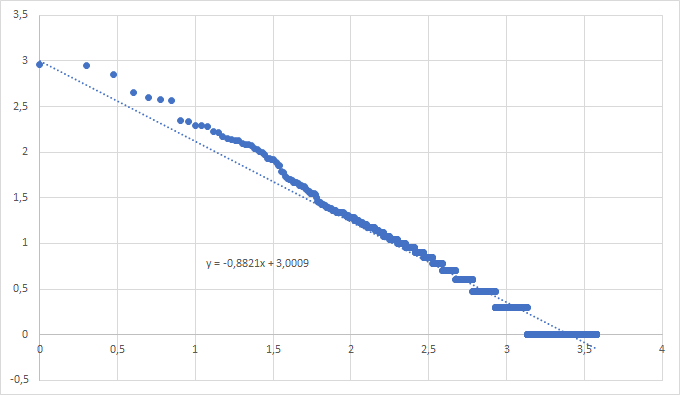
Por último usaremos un función imprimirCSV para imprimir los valores del hashmap en un CSV que procesaremos en otra parte de la práctica.

**NUBE DE PALABRAS A PARTIR DE LOS CSV:**

Primer texto(lenguaje espanol):

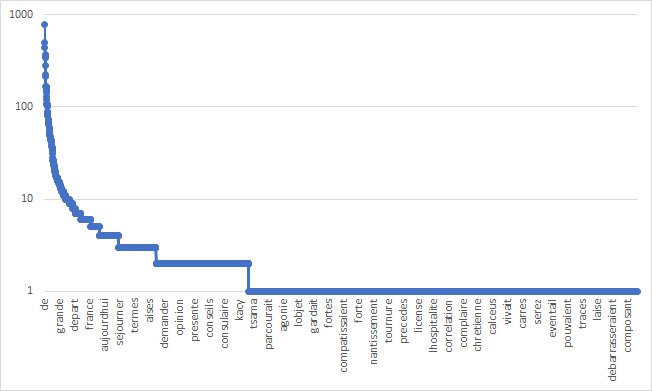
****

**Ajuste lineal**

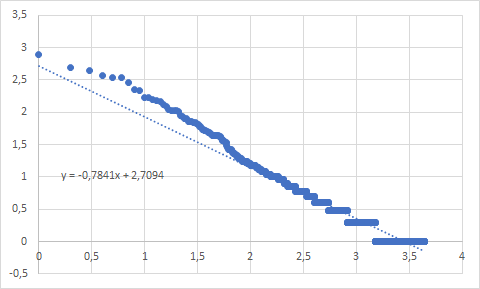
****

Texto 2 (Francés):



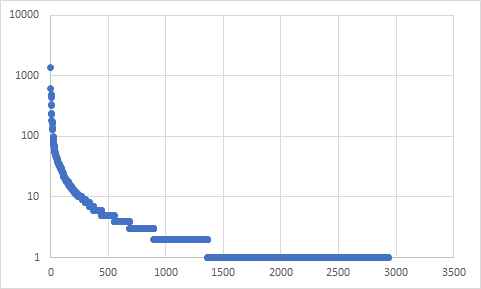


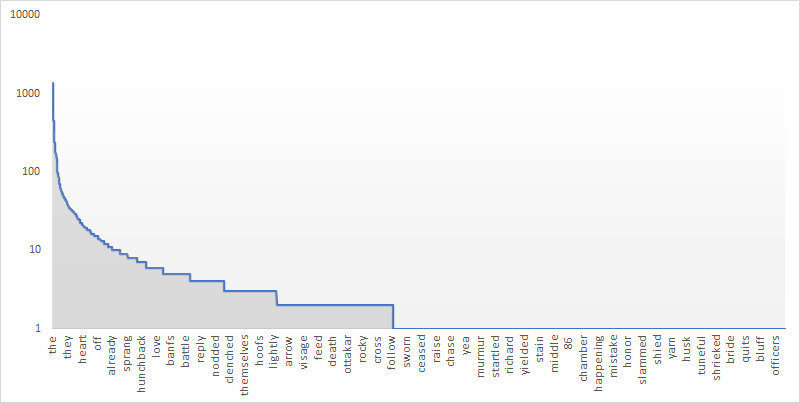
Ajuste lineal :



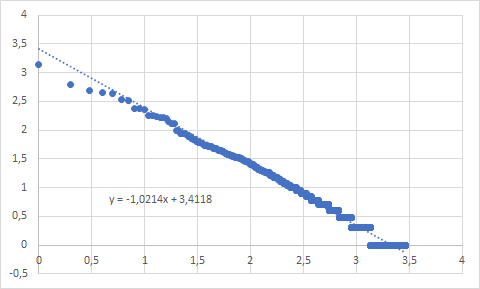
Texto 3 - (Ingles) :



****

****

**Ajuste :**

****

**REPARTO DE TRABAJO:**

De manera general el reparto de trabajo ha sido en nuestra opinión igualitario. Inicialmente Miguel realizó la primera aproximación de las funciones de tika y la implementación de la base del programa principal. Posteriormente tras decidir que sería mejor idea encapsular mediante una clase todos los datos obtenidos Pablo fue el encargado de la creación de la clase y pasar aquellos datos inicialmente obtenidos en el programa principal en los correspondientes métodos.

Una vez ya teníamos estos dos elementos planteados, realizamos el resto de la implementación de manera conjunta presencialmente. Completando los algoritmos sobre el texto y creando las funciones del programa principal.