Universidad de Granada

PRÁCTICA 1

PARSER DE DOCUMENTOS CON TIKA

Antonio Jiménez Rodríguez

1. Instrucciones de uso

Para facilitar la compilación y ejecución del programa en Java se ha creado un archivo run.sh que se hará cargo de estas acciones. Nos colocaremos en el directorio de la práctica y podremos ejecutar:

```
$ ./run.sh -d docs      # Información de los documentos
$ ./run.sh -l docs      # Enlaces de cada documento
$ ./run.sh -t docs      # Generador de CSV con ocurrencias
```

2. Documentación de código

Para la realización de esta práctica se han creado dos clases:

➔ **Document.java:** La clase *Document* guarda la información sobre el documento que necesitaremos (nombre, tipo, codificación, language y el contenido). La misma clase es la encargada de realizar la extracción de la información haciendo uso de la librería *TIKA*.

Creando un *InputStream* del documento junto con las instancias de la librería *TIKA* necesarias inicializaremos el parser para poder acceder a toda la información. Se usará el *TeeContentHandler* para "unificar" todos los ContentHandler que deseamos usar.

Para el tipo del documento accederemos al "Content-Type" del documento gracias a la clase *Metadata*.

metadata.get("Content-Type")

Así obtendremos en un *String* el tipo junto a la codificación. Para guardar únicamente el tipo en la propiedad de la clase bastaría con hacer un split de la respuesta.

La codificación del documento, además de poder obtenerla con la clase *Metadata* con el "Content-Type" junto con el tipo podremos acceder a ella con:

metadata.get("Content-Encoding")

El cuerpo del documento lo podremos obtener con la instancia de la clase BodyContentHandler con el método *toString().*

Por último, para el idioma basta con crear una instancia del *LanguageIdentifier* pasándole al constructor el cuerpo del documento, que analizará el texto y devolverá el Lenguaje en el que está escrito.

Además de las propiedades de la clase podremos acceder a métodos públicos de la clase *Document* para obtener los links y las ocurrencias del documento.
Los links, al igual que con el cuerpo, hay que llamar al método *getLinks()* de la instancia de la clase *LinkContentHandler*. Este devolverá un *List<Link>* de todos los links del documento.

Para las ocurrencias he utilizado un *HashMap* para almacenarlas. Lo primero que hacemos es limpiar el body de caracteres “especiales” y posteriormente hacemos un *split* del *string* por espacios y por saltos de línea. Una vez obtenemos el vector de string vamos guardando las palabras en el *HashMap,* si no existe la añadimos y si existe incrementamos el valor de la ocurrencia.

➔ **DocumentsParser.java:** La clase *DocumentsParser* será la encargada de la ejecución. Se han añadido las comprobaciones necesarias para los distintos argumentos que se le deben pasar.
Destacar de esta la función estática *DocumentsParser.createCSV*. A esta función le pasaremos el nombre del documento y el HashMap asociado con las ocurrencias. Crearemos el directorio donde guardaremos el CSV y el archivo del cual crearemos el *Writer* donde escribiremos las ocurrencias del documento con el delimitador “;”.

3. Nube de palabras

A partir de uno de los documentos CSV generados con la aplicación podemos generar una nube de palabras de forma sencilla.

Nube de palabras del documento Curriculum_Vitae.csv generada con https://wordart.com

4. Ley de Zipf

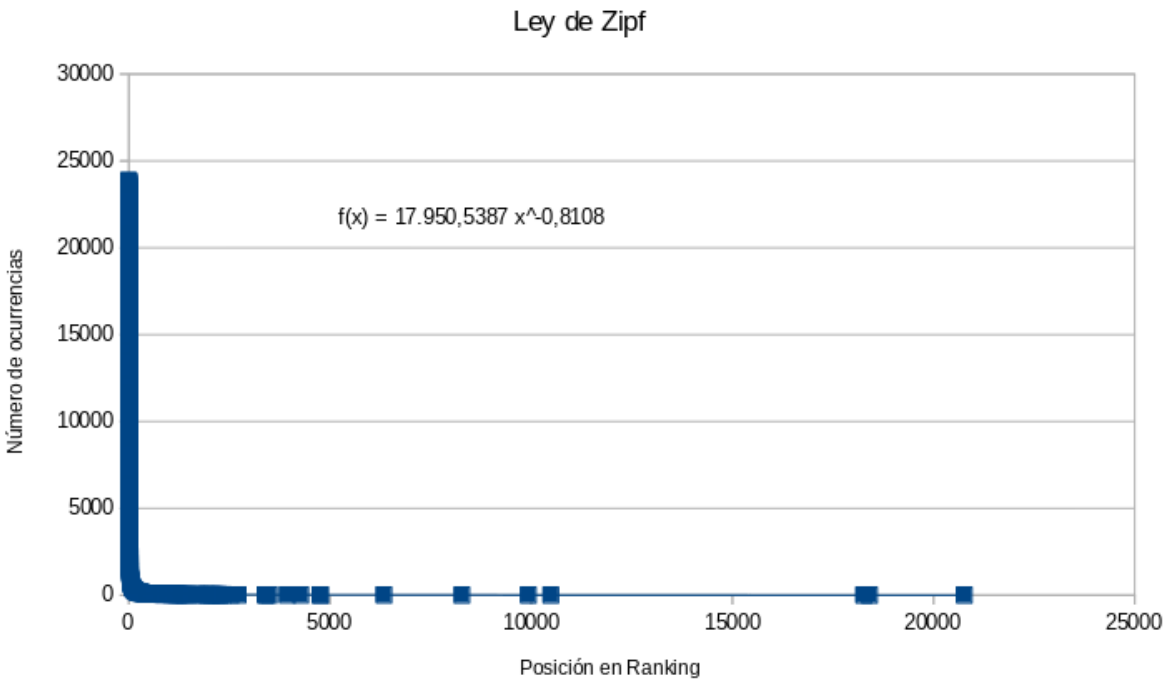

Gráfico realizado con los datos obtenidos del libro *El Quijote.*

De las mediciones del gráfico de dispersión junto con la línea de tendencia podemos recuperar las constantes *k=17959,5387* y *m=-0,8108*.