



UNIVERSIDAD DE GRANADA

Recuperación de información

Practica 1 : Tika

Alejandro Ledesma Pascual

Problema

Se nos pedía que hiciésemos un programa que hiciese tres cosas :

- 1º Que al usar la opción “-d” apareciese una tabla en la que apareciesen el nombre, codificación, idioma y tipo de cada uno de los ficheros que quisieramos (3 tipos diferentes).
- 2º Que al usar la opción “-l” apareciesen todos los enlaces de todos los ficheros que tuviesen.
- 3º Que al usar la opción “-t” se creara un archivo “CSV” en el que apareciesen la frecuencia de aparición de cada palabra junto a ella (de manera decreciente) para así luego hacer una nube de palabras de cada fichero con su csv correspondiente ,además de calcular la Ley de Zipf

Resolución

Para este problema he decidido hacer un programa en java que tiene 2 clases y un enumerado

Una de las clases es el main donde se especifica la opción que queremos hacer (usamos un enumerado para saber que opción vamos a usar) , mientras que la otra clase es la que usaremos para guardar los datos y hacer las operaciones de cada fichero.

```
public static void main(String[] args) throws Exception {

    Operation operation = null;
    // TODO code application logic here

    if(args.length == 0) {
        System.err.print( s: "No hay ningún argumento.");
        System.exit( status:1);
    }else if(args.length > 2){
        System.err.println( x: "Error en el número de argumento. MAX 2");
        System.exit( status:1);
    }else {
        if(args[0].equals( anObject: "-d")) {
            operation = Operation.TABLA;
        }else if (args[0].equals( anObject: "-l")){
            operation = Operation.ENLACE;
        }else if(args[0].equals( anObject: "-t")) {
            operation = Operation.CSV;
        }else{
            System.err.println( x: "Error al escribir los argumentos.");
            System.exit( status:1);
        }
    }
}
```

```
import org.apache.tika.parser.AutoDetectParser;
import org.apache.tika.parser.ParseContext;
import org.apache.tika.parser.Parser;
import org.apache.tika.sax.BodyContentHandler;
import org.apache.tika.sax.Link;
import org.apache.tika.sax.LinkContentHandler;
import org.apache.tika.sax.TeeContentHandler;
import org.xml.sax.SAXException;

/**
 *
 * @author Alex
 */
public class Fichero {

    private Tika tika;
    private String nombre;
    private String tipoFichero;
    private String codificacion;
    private String idioma;
    private String contenido;
    private List<Link> links;
    private HashMap<String,Integer> frecuencia;
    private List palabrasOrdenadas;
```

Dentro del constructor se llama a todas las operaciones para rellenar los datos del fichero.

Apartado 1

Para este apartado he hecho 3 funciones para conseguir nombre, idioma y codificación ,mientras para el tipo de fichero he usado una función que ya daba la clase File.

```
private String getCodificacion(File fichero) throws FileNotFoundException, TikaException {
    InputStream input = new FileInputStream( file:fichero);
    AutoDetectReader reader = null;

    try {
        reader = new AutoDetectReader( stream:input);
    }catch (IOException | TikaException e) {
        e.printStackTrace();
    }

    Charset codificacionAux = reader.getCharset();

    return codificacionAux.toString();
}
```

```
private String setIdioma() {

    LanguageDetector identificador = new OptimaizeLangDetector().loadModels();
    LanguageResult aux = identificador.detect( text: contenido);

    return aux.getLanguage();
}
```

```
private void setNombre(File fichero) {
    nombre = "";
    String aux = fichero.getName();

    for(int i = 0;aux.charAt( index:i) != '.';i++) {
        nombre += aux.charAt( index:i);
    }
}
```

Y para hacer la salida he hecho una función entro del main para que saliese una tabla más o menos ordenada con todos los valores indicados.

```
private static void imprimeTabla(File [] archivos) throws IOException, FileNotFoundException, TikaException, SAXException {
    System.out.println(x:"| Nombre | Idioma | Codificación | Tipo Fichero |");
    System.out.println(x:"-----");

    for(File archivo : archivos) {
        Fichero f = new Fichero(fichero:archivo);
        System.out.println("\n|" + f.getNombre() + " | " + f.getIdioma()
            + " | " + f.getCodificación() + " | "
            + f.getTipoFichero() + " |\n");
    }

    System.out.println(x:"-----");
}
```

SALIDA

```
C:\Users\Alex\OneDrive\Documentos\NetBeansProjects\Practical\dist>java -jar Practica1.jar -d D:\Universidad\Cuarto\RI\Practical\FicherosPractica
| Nombre | Idioma | Codificación | Tipo Fichero |
|-----|
|01 - Amante Oscuro - J | es | UTF-16LE | application/pdf |
|01 SGBDR Oracle y aspectos de red I | es | UTF-16LE | application/pdf |
|02 optimizacion de consulta | es | UTF-16LE | application/pdf |
|04 Enunciado práctica | es | UTF-16LE | application/pdf |
|1-Harry-Potter-and-the-Sorcerers-Stone | en | UTF-16LE | application/pdf |
|J | es | UTF-16LE | application/pdf |
|La Sainte Bible - Français - PDF | fr | UTF-16LE | application/pdf |
|presentación_caso_práctico | es | windows-1252 | application/vnd.oasis.opendocument.presentation |
|Página de inicio _ Departamento de Ciencias de la Computación e Inteligencia Artificial | es | UTF-8 | text/html |
|Tika2223 | es | UTF-16LE | application/pdf |
|-----|
```

Apartado 2

Para esta opción se ha hecho una función que separa el contenido en dos partes, una el contenido como tal del archivo y otra que coge los links.

```
private void setContenido(File archivo) throws IOException, SAXException, TikaException {
    InputStream input = new FileInputStream(file:archivo);
    Parser parse = new AutoDetectParser();
    Metadata metadata = new Metadata();
    BodyContentHandler bodyContent = new BodyContentHandler(writeLimit:-1);
    LinkContentHandler linkContent = new LinkContentHandler();
    TeeContentHandler teeContent = new TeeContentHandler(handlers:linkContent, handlers:bodyContent);
    ParseContext context = new ParseContext();
    parse.parse(in:input, ch:teeContent, mtdt:metadata, pc:context);

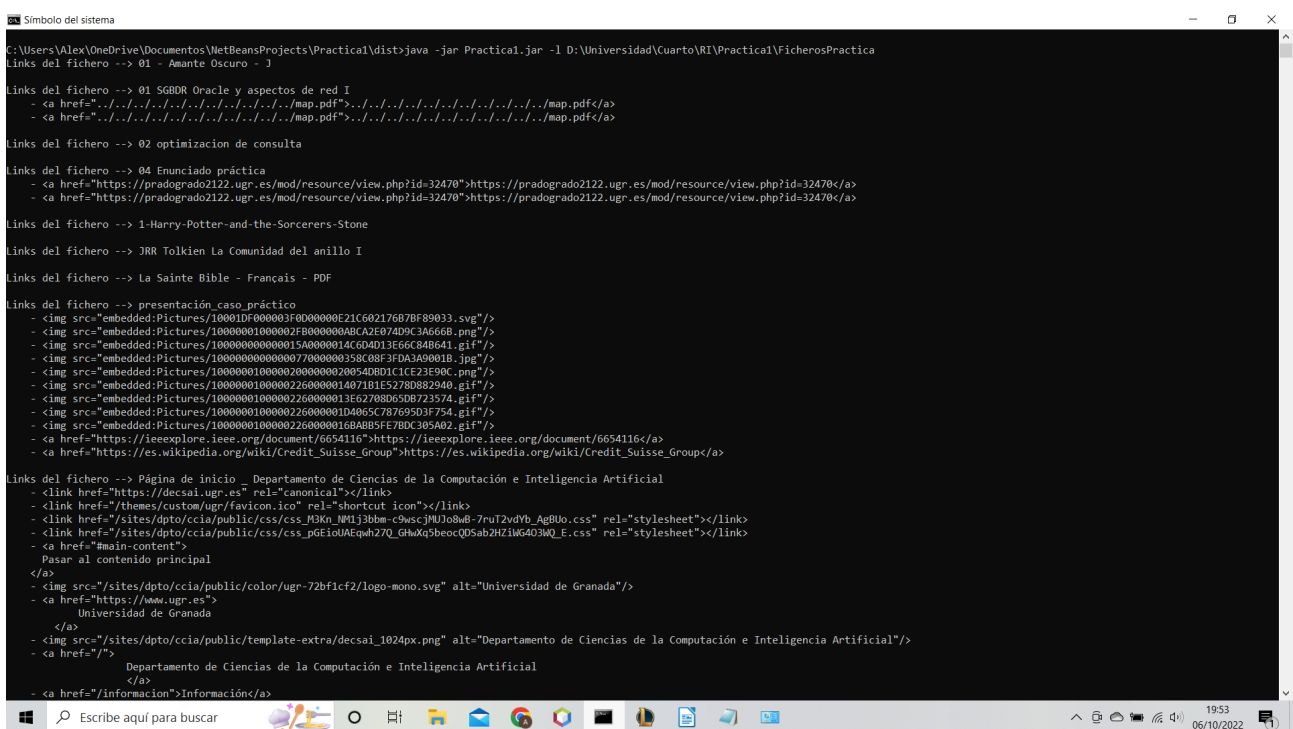
    contenido = bodyContent.toString();
    links = linkContent.getLinks();
}
```

Creamos un parse porque muchos de los ficheros exceden el número de caracteres máximo de un String.

Para la salida usamos una función creada en el main que imprime todos los links por pantalla

```
private static void mostrarLinks(File [] archivos) throws IOException, TikaException, FileNotFoundException, SAXException {  
    for(File archivo : archivos) {  
        Fichero f = new Fichero( fichero:archivo);  
  
        System.out.println("Links del fichero --> " + f.getNombre());  
        for(Link link : f.getLinks()) {  
            System.out.println("    - " + link.toString());  
        }  
        System.out.println();  
    }  
}
```

SALIDA



```
Símbolo del sistema
C:\Users\Alex\OneDrive\Documentos\NetBeansProjects\Practical\dist>java -jar Practical.jar -l D:\Universidad\Cuarto\RI\Practical\FicherosPractica

Links del fichero --> 01 - Amante Oscuro - J
Links del fichero --> 01 5GBDR Oracle y aspectos de red I
- <a href="https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470">https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470</a>
- <a href="https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470">https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470</a>

Links del fichero --> 02 optimizacion de consulta
Links del fichero --> 04 Enunciado práctica
- <a href="https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470">https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470</a>
- <a href="https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470">https://pradogrado2122.ugr.es/mod/resource/view.php?id=32470</a>

Links del fichero --> 1-Harry-Potter-and-the-Sorcerers-Stone
Links del fichero --> JRR Tolkien La Comunidad del anillo I
Links del fichero --> La Sainte Bible - Français - PDF
Links del fichero --> presentación_caso práctico
- 
- 
- 
- 
- 
- 
- 
- 
- 
- <a href="https://ieeexplore.ieee.org/document/6654116">https://ieeexplore.ieee.org/document/6654116</a>
- <a href="https://es.wikipedia.org/wiki/Credit_Suisse_Group">https://es.wikipedia.org/wiki/Credit_Suisse_Group</a>

Links del fichero --> Página de inicio Departamento de Ciencias de la Computación e Inteligencia Artificial
- <link href="https://decsai.ugr.es" rel="canonical"></link>
- <link href="/themes/custom/ugr/favicon.ico" rel="shortcut icon"></link>
- <link href="/sites/dpto/ccia/public/css/css_H3Kn_NM1j3bbm-c9wscjMUJo8wB-7ruT2vdYb_AgBUo.css" rel="stylesheet"></link>
- <link href="/sites/dpto/ccia/public/css/css_pGEloUAEqwh27Q_GhwKq3beocQ05abZHwG403M0_F.css" rel="stylesheet"></link>
- <a href="#main-content">
Pasarse al contenido principal
</a>
- 
- <a href="https://www.ugr.es">
Universidad de Granada
</a>
- 
- <a href="/">
Departamento de Ciencias de la Computación e Inteligencia Artificial
</a>
- <a href="/informacion">Información</a>
```

Apartado 3

Para crear los csv primero había que hacer un conteo de palabras con sus respectivas frecuencias y ordenarlas de manera descendente. Para ello he creado dos funciones, una que contiene un algoritmo que recorre todo el contenido (una vez eliminados todos los caracteres que no están relacionados con las letras y puestas en minúscula) y otra que crea la lista ordenada de las palabras.

```
private void ordenar() {  
    palabrasOrdenadas = new LinkedList(c:frecuencia.entrySet());  
    Collections.sort(list:palabrasOrdenadas, (Object o1, Object o2) -> ((Comparable) ((Map.Entry) (o1)).getValue()).compareTo(c: ((Map.Entry) (o2))  
}
```

```

private void calcularFrecuencia() {
    String aux = contenido.toLowerCase();
    aux = aux.replaceAll( regex: "[^a-zA-ZáéíóúÁÉÍÓÚüñÑàìòùÀÈÌÒÙ ]", replacement: "");

    String palabra = "";
    for(int i = 0;i < aux.length();i++) {
        if(aux.charAt( index:i) != ' ') {
            palabra += aux.charAt( index:i);
        }else if(aux.charAt( index:i) == ' ') {
            if(frecuencia.containsKey( key:palabra)) {
                frecuencia.put( key:palabra,frecuencia.get( key:palabra) + 1);
            }else{
                frecuencia.put( key:palabra, value:1);
            }
            palabra = "";
        }
    }

    //Integer remove = frecuencia.remove(" ");

    ordenar();
}

```

Para crear los csv correspondientes se crea una función en el main que contiene un writer con que se le especifica el nombre del fichero y va creando los ficheros y añadiendo las palabras y su frecuencia.

```

private static void crearCSV(File [] archivos) throws IOException, FileNotFoundException, TikaException, SAXException {

    File csv = new File( pathname: "D:\\Universidad\\Cuarto\\RI\\Practical\\FicherosCSV");
    String ruta = csv.getAbsolutePath();
    ruta = ruta + "\\ ";

    for(File archivo : archivos) {
        Fichero f = new Fichero( fichero:archivo);

        PrintWriter writer = new PrintWriter(ruta + f.getNombre() + ".csv", csn:"UTF-8");

        writer.println( x:"Text;Size");
        Iterator it = new ReverseListIterator( list:f.getPalabrasOrdenadas());

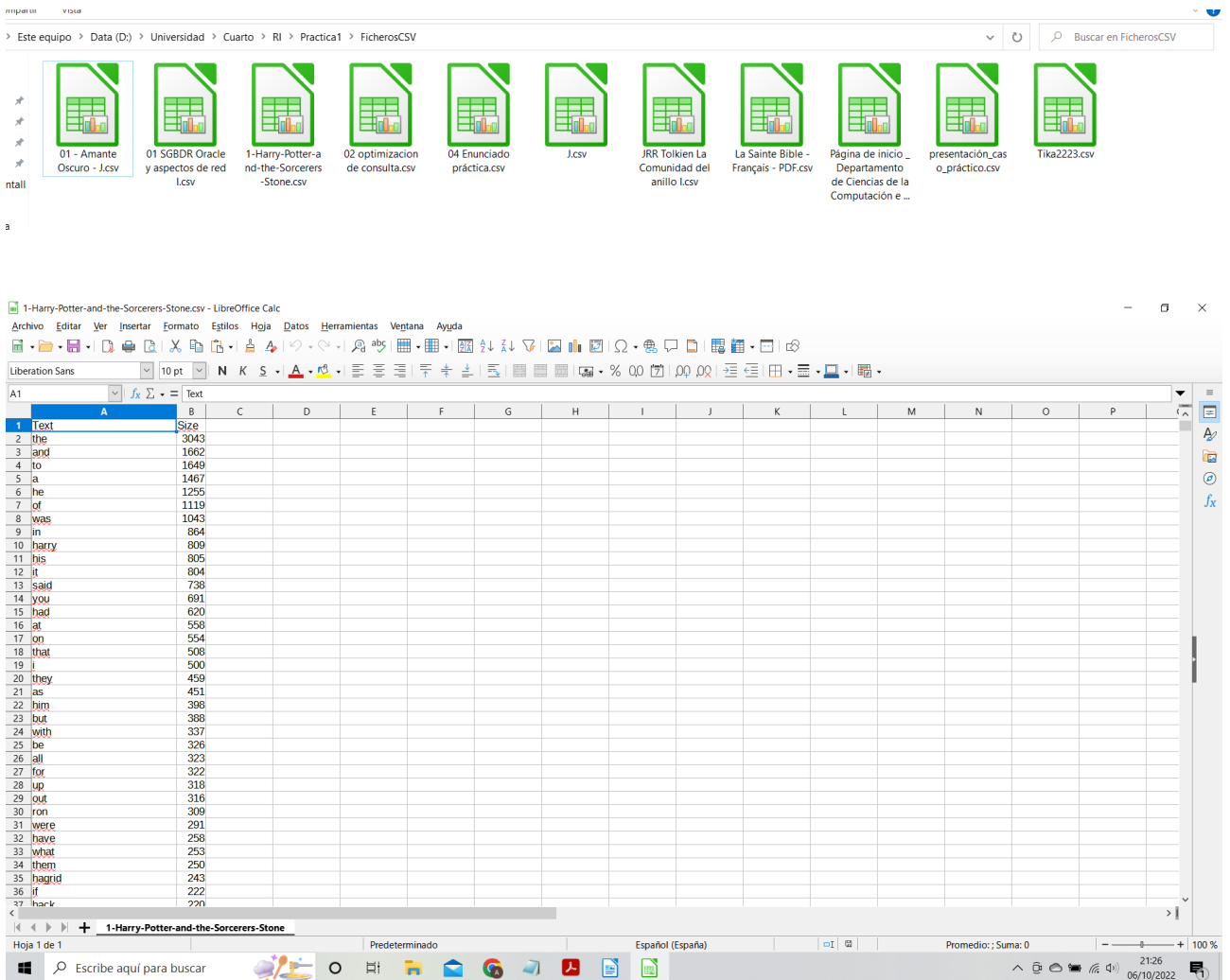
        while(it.hasNext()) {
            String frecuencia = it.next().toString();
            frecuencia = frecuencia.replace( oldChar: '=', newChar: ';');
            if(frecuencia.charAt( index:0) != ';') {
                writer.println( x:frecuencia);
            }
        }
        writer.close();
    }

    System.out.println( x:"\nCSV creados con éxito.");
}

```

SALIDA

```
C:\Users\Alex\OneDrive\Documents\NetBeansProjects\Practical1\dist>java -jar Practical1.jar -t D:\Universidad\Cuarto\RI\Practical1\FicherosPractica1\
CSV creados con éxito.
```



Por ejemplo aquí se pueden ver las palabras con sus frecuencias del libro de Harry Potter en inglés y como se puede ver concuerda con la nube de palabras que mientras más aparece más grande es la palabra.

