
Práctica 1

Recuperación de Información

Amin Kasrou Aouam



**UNIVERSIDAD
DE GRANADA**

2020-10-28

Índice

Práctica 1	3
Instalación	3
Usando Nix	3
Sin Nix	3
Ejecución	4
Implementación	4

Práctica 1

En esta práctica, vamos a obtener información de una serie de documentos usando *Apache Tika*:

1. Obtener los metadatos (nombre , tipo, codificación e idioma)
2. Extraer todos los enlaces que aparecen
3. Generar un fichero con las ocurrencias de cada término, ordenados de forma descendente
4. Generar una gráfica con los términos y su ocurrencia

Instalación

Implementamos la práctica usando *Java* como lenguaje de programación, y *Maven* como herramienta de gestión del proyecto. No es necesario utilizar esta herramienta, pero nos ofrece una mayor reproducibilidad del proyecto.

En el caso de que deseemos instalar fácilmente todas las dependencias, podemos instalar el gestor de paquetes Nix (compatible con Linux, MacOS y WSL)

Usando Nix

1. Instalamos Nix:

```
1 sh <(curl -L https://nixos.org/nix/install) --daemon
```

1. Cambiamos la ruta al directorio del proyecto:

```
1 cd P1
```

1. Entramos en el entorno de desarrollo reproducible y aislado:

```
1 nix-shell
```

Nix se encargará de resolver las dependencias, podemos dirigirnos directamente a la sección de ejecución.

Sin Nix

En el caso de que no deseemos usar Nix, deberemos instalar las siguientes dependencias en nuestro sistema:

- OpenJDK (> 8.0)
- Maven
- Gnuplot

Ejecución

En el caso que deseemos utilizar *Maven*, debemos ejecutar los siguientes comandos:

1. Compilar el proyecto

```
1 mvn compile
```

1. Ejecutar el proyecto

```
1 mvn exec:java -Dexec.mainClass="org.RI.P1.AnalyzeDirectory" -Dexec.args="data metadata"
```

Debemos modificar el argumento **metadata** según la salida que deseemos:

- **metadata**: obtenemos la información de los archivos (nombre, codificación, tipo)
- **links**: obtenemos la lista de enlaces de cada archivo
- **frequency**: se guarda la frecuencia de las palabras de cada documento en un archivo (se encuentran en la carpeta output).

Implementación

Tal y como podemos observar en la sección previa de ejecución, hemos optado por añadir un argumento para elegir la acción que deseamos realizar. En caso de que se omita éste, las instrucciones de ejecución aparecen por pantalla.

```
1 private static void usage() {
2     System.out.println("Usage: AnalyzeDirectory <directory> <option>");
3     System.out.println("option metadata: shows the filename/file type/
4         encoding and language of the files");
5     System.out.println("option links: shows all the links contained in
6         each file");
7     System.out.println("option frequency: saves word frequency to a
8         file");
9     System.exit(1);
10 }
11
12 public static void main(String[] args) throws IOException,
13     TikaException, SAXException {
14     if (args.length != 2) {
15         usage();
16     }
17     String directory = args[0];
18     String action = args[1];
19     readFiles(directory);
20     chooseAction(action);
21 }
```

Esta funcionalidad permite al usuario seleccionar la salida que desea, además de disminuir el contenido que aparece por pantalla y el tiempo de ejecución (dado que no computamos acciones innecesarias).

En el caso de la opción *frequency*, guardamos las ocurrencias de los términos en un fichero en el directorio **output** con el formato *filename.dat*. Utilizaremos estos archivos para generar las gráficas.