

Preprocesado de documentos

Parser de documentos con TIKa

September 30, 2021

1 Objetivo

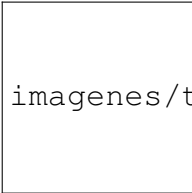
En esta práctica veremos cómo poder extraer información (texto) a partir de un documento dado, independiente del formato del archivo, paso previo para cualquier proceso de recuperación de información o análisis de textos.

La práctica se puede realizar de forma individual o en grupos de dos personas. En este caso, al final de la misma se debe entregar un informe que necesariamente debe incluir una sección denominada *Trabajo en Grupo* en el que se indicará de forma clara la contribución de cada alumno.

1.1 Documentos digitales

El mundo de los documentos digitales y sus distintos formatos es un mundo donde cada uno tiene un lenguaje diferente (pdf, jpg, gif, png, xls, doc, odt, html, xml, rss, mp3, ...). La mayoría de los programas sólo entienden su propio formato o un conjunto pequeño de formatos relacionados. Sin embargo, son muchas las aplicaciones que deben acceder a la información almacenada en un fichero, como por ejemplo los buscadores.

En esta práctica daremos los primeros pasos en una herramienta *Tika*, consultar la URL <https://tika.apache.org/>, que nos va a ser de utilidad a la hora de realizar este trabajo. *Tika* nos permite detectar los distintos formatos de ficheros de forma automática así como bucear dentro de los mismos para extraer el contenido textual así como sus meta-datos.



imagenes/tika.png

Los metadatos nos proporcionan información sobre un determinado fichero, que puede ir desde el tamaño del fichero, localización, autor, versión, etc..

El estándar MIME (Multipurpose Internet Mail Extension) especifica que el formato de un fichero consiste en un identificador de tipo/subtipo y un conjunto opcional de elementos atributo=valor. Por ejemplo

- text/plain; charset=ISO-8859-1
- application/msword;
- application/pdf; version=1.5
- image/jpeg

Ejemplos de metadata son,

```
1 Author: Witten , I. H.; Frank , Eibe .
2 Content-Length: 8138814
3 Content-Type: application/pdf
4 Creation-Date: 2005-09-28T07:50:58Z
5 ISBN: 0120884070
6 Last-Modified: 2005-09-28T07:51:42Z
7 Last-Save-Date: 2005-09-28T07:51:42Z
8 cp:subject: Team DDU
9 created: Wed Sep 28 09:50:58 CEST 2005
10 creator: Witten , I. H.; Frank , Eibe .
11 ....
```

o bien

```
1 Content-Encoding: ISO-8859-1
2 Content-Length: 3821
3 Content-Type: text/plain; charset=ISO-8859-1
4 X-Parsed-By: org.apache.tika.parser.DefaultParser
5 resourceName: kapins.log
```

2 Tika

Apache Tika (<https://tika.apache.org/>) es una herramienta que extrae metadatos y texto de una gran variedad de tipos de ficheros (PDF, PPT, XLS, XML, HTTP, MPEG4, etc.). Tika nos proporciona una interfaz sencilla para el acceso a las distintas componentes, para lo cual hace uso de distintos tipos de analizadores sintácticos (parser), ya sean genéricos como específicos, facilitando el desarrollo de nuestra tarea.

De forma genérica, en Tika podemos detectar tres tipos de componentes:

- Tipo: Detección del tipo de documento
- Parser: Que permite extraer el contenido textual del fichero
- Detección de MIME: Detecta los metadatos
- Detección de idioma: Puede determinar de forma automática si en texto está en castellano, inglés, francés, ...

Una ayuda mas completa sobre Tika la podremos encontrar en la página web de la asignatura (libro Tika in Action de Chris A. Mattmann y Jukka L. Zitting), el tutorial que se encuentra en la página web <https://www.tutorialspoint.com/tika>, así como en la wiki <https://cwiki.apache.org/confluence/display/tika/>

2.1 Aplicaciones que usan Tika

Existen varias aplicaciones que utilizan Tika internamente, pasamos a destacar algunas de ellas:

- Motores de Búsqueda: Utilizan Tika para extraer el contenido y los metadatos de los documentos en la fase de indexación.
- Crawler (Arañas): Hacen uso de Tika para extraer los enlaces de los documentos web.

- **Análisis de documentos:** En el campo de la inteligencia artificial podemos encontrar herramientas que analizan el documento a nivel semántico y extraen todos los datos de los mismos, como por ejemplo en tareas de clasificación documental
- **Gestión de Recursos Digitales:** Sistemas de información que permiten gestionar datos digitales (imágenes, animaciones, presentaciones, documentos, etc.) permitiendo tareas como organización, búsqueda, conversión, edición, distribución, verificación de integridad, ...
- **Sistemas de Recomendación,** como las utilizadas en el comercio electrónico (Apache Mahout en un entorno que proporciona algoritmos de aprendizaje automático y que se utiliza para el desarrollo de sistemas de Recomendación)

2.2 Descargar Tika

El código de completo de Tika lo podemos descargar de <http://tika.apache.org> y seguir los pasos allí indicados. Con fecha 30 de Septiembre de 2021, la versión de Tika es Tika v2.1.0

Además, en la página web de la asignatura nos podremos descargar el fichero `.jar` con todo lo necesario para poder trabajar (está en el subdirectorío Software). Una forma fácil de entender las potencialidades de Tika es utilizar la GUI, ejecutando `java -jar tika-app-2.1.0.jar -g`, o bien a través de la línea de comandos, ejecutando `java -jar tika-app-2.1.0.jar -help` nos muestra la lista de las opciones disponibles.

```
1 usage: java -jar tika-app.jar [option ...] [file|port ...]
2
3 Options:
4   -?  or --help          Print this usage message
5   -v  or --verbose       Print debug level messages
6   -V  or --version       Print the Apache Tika version number
7
8   -g  or --gui           Start the Apache Tika GUI
9   -s  or --server        Start the Apache Tika server
```

10	-f or --fork extraction	Use Fork Mode for out-of-process
11		
12	-x or --xml	Output XHTML content (default)
13	-h or --html	Output HTML content
14	-t or --text	Output plain text content
15	-T or --text-main content only)	Output plain text content (main
16	-m or --metadata	Output only metadata
17	-j or --json	Output metadata in JSON
18	-y or --xmp	Output metadata in XMP
19	-l or --language	Output only language
20	-d or --detect	Detect document type
21	-eX or --encoding=X	Use output encoding X
22	-pX or --password=X	Use document password X
23	-z or --extract directory	Extract all attachments into current
24	--extract-dir=<dir>	Specify target directory for -z
25	-r or --pretty-print newlines and	For XML and XHTML outputs , adds
26		whitespace , for better readability
27		
28	--create-profile=X	
29	Create NGram profile , where X is a profile name	
30	--list-parsers	
31	List the available document parsers	
32	--list-parser-details	
33	List the available document parsers and their supported mime types	
34	--list-parser-details-apt	
35	List the available document parsers and their supported mime types in apt format.	
36	--list-detectors	
37	List the available document detectors	
38	--list-met-models	
39	List the available metadata models , and their supported keys	
40	--list-supported-types	
41	List all known media types and related information	

```
42
43 Description :
44     Apache Tika will parse the file(s) specified on the
45     command line and output the extracted text content
46     or metadata to standard output.
47
48     Instead of a file name you can also specify the URL
49     of a document to be parsed.
50
51     If no file name or URL is specified (or the special
52     name "-" is used), then the standard input stream
53     is parsed. If no arguments were given and no input
54     data is available, the GUI is started instead.
55
56 - GUI mode
57
58     Use the "--gui" (or "-g") option to start the
59     Apache Tika GUI. You can drag and drop files from
60     a normal file explorer to the GUI window to extract
61     text content and metadata from the files.
62
63 - Server mode
64
65     Use the "--server" (or "-s") option to start the
66     Apache Tika server. The server will listen to the
67     ports you specify as one or more arguments.
```

Por ejemplo, para conocer el lenguaje en que se encuentra escrito un texto podemos ejecutar

```
1 java -jar tika-app-2.1.0.jar --language nombrefichero
```

y nos dará la salida en (english), es (español), fr (francés), etc.

o para consultar el formato de algún fichero del que no estamos seguros (hemos perdido la extensión, nos lo han pasado en un adjunto, etc.)

```
1 java -jar tika-app-2.1.0.jar --detect nombrefichero
```

2.3 Explorando Tika

En este caso, consideramos distintos tipos de ficheros que tengamos en nuestro ordenador (algunos ficheros ejemplo lo podemos encontrar en la web de la asignatura), y veremos que es lo que pasa cuando los exploramos con la interfaz de usuario de Tika, donde podremos ver

- Texto formateado: Extrae el texto formateado como XHTML. Con esto podemos ver cómo Tika entiende la estructura del documento. Idealmente, podremos ver el contenido en orden correcto, con elementos como enlaces y cabeceras bien identificados.
- Texto Plano: Texto extraído, sin tener en cuenta la estructura del documento
- Texto estructurado: fuente del texto en XHTML
- Metadatos: Los metadatos del fichero.

Como hemos visto, también lo podemos ejecutar desde línea de comandos, por ejemplo podemos ejecutar

```
java -jar tika-app-2.1.0.jar < prueba.txt > fichero.xhtml
```

o incluso podemos utilizar una url

```
java -jar tika-app-2.1.0.jar https://decsai.ugr.es
```

Si queremos la salida en texto plano podemos hacer

```
java -jar tika-app-2.1.0.jar -text https://decsai.ugr.es
```

en este caso, utilizará la codificación por defecto del sistema, pero podemos cambiar la codificación de salida utilizando `-encoding`.

Si solo estamos interesados en los metadatos, los obtenemos mediante

```
java -jar tika-app-2.1.0.jar -metadata https://decsai.ugr.es
```

obteniendo

```
1 Content-Encoding: UTF-8
2 Content-Language: es
3 Content-Length: 44092
```

```
4 Content-Type: text/html; charset=UTF-8
5 Generator: Drupal 8 (https://www.drupal.org)
6 HandheldFriendly: true
7 MobileOptimized: width
8 X-TIKA:Parsed-By: org.apache.tika.parser.DefaultParser
9 X-TIKA:Parsed-By: org.apache.tika.parser.html.HtmlParser
10 content-language: es
11 dc:title: Página de inicio | Departamento de Ciencias de la
    Computación e Inteligencia Artificial
12 description: Universidad de Granada – Departamento de Ciencias
    de la Computación e Inteligencia Artificial CCIA-UGR.
13 geo.placename: Granada, España
14 geo.region: ES
15 og:image: https://decsai.ugr.es/themes/custom/ugr/screenshot.png
16 og:image:alt: Logo Universidad de Granada (UGR)
17 og:image:height: 400px
18 og:image:width: 400px
19 og:site_name: Departamento de Ciencias de la Computación e
    Inteligencia Artificial
20 og:title: Página de inicio | Departamento de Ciencias de la
    Computación e Inteligencia Artificial
21 og:type: website
22 og:url: https://decsai.ugr.es
23 viewport: width=device-width, initial-scale=1.0
```

2.4 Incluir Tika en nuestros programas

Tika proporciona un API que implementa la mayoría de los métodos básicos, sin tener que centrarnos en su codificación. Las clases las podemos encontrar en `org.apache.tika.Tika`. Veamos como podemos implementar un programa para extraer el contenido textual de forma simple.

```
1 import java.io.File;
2 import org.apache.tika.Tika;
3
4 public class EjemploSimple {
5     public static void main(String[] args) throws Exception {
6
```



```
7 // Creaamos una instancia de Tika con la configuracion por
  defecto
8 Tika tika = new Tika();
9 // Se parsean los ficheros pasados como argumento y se extrae
  el contenido
10 for (String file : args) {
11     File f = new File(file);
12
13     // Detectamos el MIME tipo del fichero
14     String type = tika.detect(f);
15     System.out.println(file + ":" + type);
16
17     // Extraemos el texto plano en un string
18     String text = tika.parseToString(f);
19     System.out.print(text);
20 }
21 }
22 }
```

para ejecutarlo podemos hacer

```
javac -cp tika-app-2.1.0.jar EjemploSimple.java
java -cp tika-app-2.1.0.jar:. EjemploSimple documento
```

2.5 Extracción del contenido

Como hemos visto, Tika puede extraer el contenido de un documento de forma simple, cuando llamamos al método `parseToString` primero se detecta el MIME del fichero, para después se utiliza el parser correcto para avanzar a través del mismo. Así, si el fichero es un pdf, se utilizará la clase que da soporte al MIME `application/pdf`, esto es, `org.apache.tika.parser.pdf.PDFParser`, convirtiendo el contenido y los metadatos a un formato que entiende Tika.

En este caso, el texto se pasa a un `String`, con un tamaño limitado (100000 caracteres), por lo que si el fichero pdf es grande (por ejemplo un libro) no se analizará todo el texto. En estos casos se recomienda hacer un parser incremental que hace el parser sobre un `Reader`, clase abstracta de Java que permite leer streams de caracteres.

```
Reader texto = tika.parser(file)
```

Otra alternativa, que nos da más control es utilizar la clase Parser,

`org.apache.tika.parser.Parser`¹,

de la que el principal método es `parse`.

```
void parse( InputStream stream, ContentHandler handler,  
Metadata metadata, ParseContext context)  
throws IOException, SAXException, TikaException;  
donde
```

- `InputStream` representa el stream de bytes del que lee el parser. Es leído pero no cerrado por parser. Por tanto, el proceso típico es

```
1  InputStream stream = ...;           // open the stream  
2  try {  
3      parser.parse(stream, ...); // parse the stream  
4  } finally {  
5      stream.close();           // close the stream  
6  }
```

- `ContentHandler` el stream de datos es escrito en el handler en formato estructurado, XHTML, permitiendo que Tika, y las aplicaciones donde se utilice, consideren conceptos como cabeceras, enlaces, etc. La estructura de la salida es:

```
1  <html xmlns="http://www.w3.org/1999/xhtml">  
2      <head>  
3          <title>... </title>  
4      </head>  
5      <body>  
6          ...  
7      </body>  
8  </html>
```

- `Metadata`, es un parámetro tanto de entrada como de salida, y es utilizado para representar los metadatos del documento. Así podemos considerar metadatos de entrada de la aplicación como `RESOURCE_NAME_KEY`

¹Más información la podemos encontrar en <https://tika.apache.org/2.1.0/parser.html>

(nombre del fichero); CONTENT_TYPE (tipo); TITLE (título),... y también podemos incluir metadatos propios que podemos esperar para este tipo de fichero.

Por ejemplo, podríamos considerar

```
1 Metadata met = new Metadata();
2
3 parser.parse(is, ch, met, ...);
4 String docType = met.get("Content-Type");
```

- ParseContext, es utilizado para modificar el comportamiento de ContentHandler indicando información del contexto concreto sobre el vamos a trabajar, dando un control más preciso para el parser permitiendo que la aplicación cliente pueda realizar tareas más específicos.

3 Ejemplos

3.1 Utilizando la clase de alto nivel Tika

Podemos utilizar el patrón fachada Tika, que nos proporciona un interfaz simple para todo el sistema y que delega las operaciones a los objetos apropiados como se indica en el siguiente listado

```
1 import java.io.InputStream;
2
3 import org.apache.tika.Tika;
4 import org.apache.tika.metadata.Metadata;
5 ...
6
7 File file = new File("...../index.xml");
8
9 InputStream is = new FileInputStream(file);
10 Tika tika = new Tika();
11 Metadata metadata = new Metadata();
12
13 String contenido = tika.parseToString(is);
14 String tipoFichero = tika.detect(is);
```

```
15 tika.parse(is, metadata);
16 //Accedemos a los metadatos
17 for (String name : metadata.names()) {
18     String valor = metadata.get(name);
19     if (valor != null) {
20         System.out.println("metadata: " + name + " " + valor);
21     }
22 }
23 // o directamente
24 System.out.print("tipo " + metadata.get(Metadata.CONTENT_TYPE));
25 }
```

3.1.1 Análisis utilizando el parser

Podemos extraer el texto y los metadatos de forma genérica considerando el siguiente código

```
1 File file = new File(".....l/index.xml");
2
3 InputStream is = new FileInputStream(file);
4 Metadata metadata = new Metadata();
5 ContentHandler ch = new BodyContentHandler();
6 ParseContext parseContext = new ParseContext();
7
8
9 AutoDetectParser parser = new AutoDetectParser();
10
11 parser.parse(is, ch, metadata, parseContext);
12
13 //Tenemos el texto
14 System.out.println("ch " + ch.toString());
```

3.1.2 Trabajando con XML

Los distintos formatos que puede entender Tika los podemos encontrar en <https://tika.apache.org/1.6/formats.html>, así por ejemplo si queremos extraer el texto de un fichero XML, podríamos considerar el XMLParser.

```
1 Parser pxml = new XMLParser();
```

```
2 ToXMLContentHandler chxml = new ToXMLContentHandler();
3
4 pxml.parse(is, chxml, metadata, parseContext);
5
6 System.out.println("texto " + chxml.toString());
```

3.1.3 Parser sobre un PDF

Muchos de los documentos que podemos encontrar se encuentran almacenados en formato PDF, por lo que merece un tratamiento especial. Hemos visto que a través del patrón de diseño (facade) Tika, considerando por ejemplo el método `parseToString` o el método `parse` nos permite extraer la información del pdf, pero siempre podemos recurrir al parser específico para PDF si queremos un mayor control del proceso, este es el `PDFParser`.

`PDFParser` permite procesar PDF encriptados cuando disponemos de la clave como parte de los metadatos de entrada. Un ejemplo de su uso es el siguiente:

```
1      FileInputStream inputstream = new FileInputStream(new File
2          ("Example.pdf"));
3
4      BodyContentHandler handler = new BodyContentHandler();
5      Metadata metadata = new Metadata();
6      ParseContext pcontext = new ParseContext();
7
8      PDFParser pdfparser = new PDFParser();
9
10     pdfparser.parse(inputstream, handler, metadata, pcontext);
11
12     // getting the content of the document
13     System.out.println("Contents of the PDF : " + handler.
14         toString());
15
16     // getting metadata of the document
17     System.out.println("Metadata of the PDF:");
18     String[] metadataNames = metadata.names();
19
20     for(String name : metadataNames) {
```

```
20         System.out.println(name+ " : " + metadata.get(name));
21     }
```

3.1.4 Un ejemplo más completo

En este caso, podemos ver cómo se puede parsear una página web, sacándole distintos tipos de contenido. Para ello se puede utilizar un TeeContentHandler que nos permite agrupar distintos ContentHandler en uno sólo

```
1  import java.io.InputStream;
2  import java.net.URL;
3  import org.apache.tika.metadata.Metadata;
4  import org.apache.tika.parser.ParseContext;
5  import org.apache.tika.parser.html.HtmlParser;
6  import org.apache.tika.sax.BodyContentHandler;
7  import org.apache.tika.sax.LinkContentHandler;
8  import org.apache.tika.sax.TeeContentHandler;
9  import org.apache.tika.sax.ToHTMLContentHandler;
10 import org.xml.sax.ContentHandler;
11
12 public class ParsearHTML {
13
14     public static void main (String args[]) throws Exception {
15         URL url = new URL("http://www.ideal.es");
16         InputStream input = url.openStream();
17
18         LinkContentHandler linkHandler = new LinkContentHandler
19             ();
20         ContentHandler textHandler = new BodyContentHandler();
21         ToHTMLContentHandler toHTMLHandler = new
22             ToHTMLContentHandler();
23
24         TeeContentHandler teeHandler = new TeeContentHandler(
25             linkHandler, textHandler, toHTMLHandler);
26
27         Metadata metadata = new Metadata();
28         ParseContext parseContext = new ParseContext();
29         HtmlParser parser = new HtmlParser();
30         parser.parse(input, teeHandler, metadata, parseContext);
31     }
32 }
```

```
28         System.out.println("title:\n" + metadata.get("title"));
29         System.out.println("links:\n" + linkHandler.getLinks());
30         System.out.println("text:\n" + textHandler.toString());
31         System.out.println("html:\n" + toHTMLHandler.toString());
32         ;
33     }
```

3.1.5 Identificación del lenguaje

Con Tika podemos identificar de forma simple el lenguaje en que esta escrito un texto, con esto podremos identificar el stemmer a utilizar

```
1  import org.apache.tika.language.LanguageIdentifier;
2  import org.apache.tika.language.detect.LanguageDetector;
3  import org.apache.tika.langdetect.OptimaizeLangDetector;
4  import org.apache.tika.language.detect.LanguageResult;
5  ...
6
7  public static String identifyLanguage(String text) throws
    IOException {
8      LanguageDetector identifier = new
        OptimaizeLangDetector().loadModels();
9      LanguageResult idioma = identifier.detect("This is
        spanish");
10     System.out.println("XXXXXX"+idioma.getLanguage());
11     return idioma.getLanguage();
12 }
```

3.1.6 Mas Ejemplos

Mas ejemplos de uso los podemos encontrar en <https://tika.apache.org/2.1.0/examples.html>. Ejemplos adicionales, incluidos los que aparecen en el libro Tika in Action los podemos entrar en <https://svn.apache.org/repos/asf/tika/trunk/tika-example>

4 A entregar

Se pide realizar un programa que sea capaz de extraer toda la información de los documentos que cuelgan de un directorio, que será pasado como entrada el programa. Para ello nos crearemos un directorio en que almacenaremos como mínimo 10 ficheros distintos, y al menos 3 formatos² y debemos generar las siguientes salidas en función del parámetro de entrada que le pasemos:

- d Realizar de forma automática una tabla que contenga el nombre del fichero, tipo, codificación e idioma.
- l Todos los enlaces que se pueden extraer de cada documento
- t Para cada documento, generar un fichero con formato CSV que contenga la ocurrencia de cada uno de los términos en el mismo. Estos deben aparecer en orden decreciente de frecuencia, con el siguiente formato: termino y frecuencia (número de ocurrencias del término en el documento) separados por ';', como en el siguiente ejemplo obtenido a partir del texto de la práctica

```
Text;Size
De;124
Tika;90
El;58
En;55
Que;50
Apache;45
La;43
Un;40
Los;31
Del;27
Org;27
...
```

Una vez que tengamos el fichero, podremos utilizarlo para crear una nube palabras (podemos utilizar, entre otros <https://wordart.com/create>

²Podemos utilizar los ficheros creados a la hora de elaborar la práctica anterior

con la opción de importar CSV) obteniendo por ejemplo la salida de la figura 1



Figure 1: Nube de palabras a partir del texto de la práctica generadas por Word Art

- En este apartado nos centraremos en aquellos documentos de mayor tamaño, como pueden ser un libro que os podéis descargar del proyecto Gutenberg (<https://www.gutenberg.org/>), considerar al menos tres idiomas diferentes. Una vez obtenida la salida del apartado anterior (se recomienda al menos pasar las palabras a minúsculas), debemos hacer un gráfico donde

se presenten en el eje de las X los términos ordenados en orden decreciente de frecuencia y en el eje de las Y la frecuencia de los mismos. De igual forma se presentará el gráfico log-log.

Mediante este gráfico podremos comprobar si el documento sigue la ley de Zipf (salvo que el documento sea de tamaño considerable, esto será poco probable). Una versión más actualizada de la Ley de Zipf, viene dada por la ecuación de Booth y Federowicz, esto es,

$$F = \frac{k}{R^m}$$

donde F representa la frecuencia, R la posición en el ranking (ordenación) y k y m son constantes. Para obtener dichas constantes podemos hacerlo a partir del gráfico log-log teniendo en cuenta que

$$\ln(F) = \ln\left(\frac{k}{R^m}\right) = \ln(k) - m \ln(R)$$

Por tanto, si realizamos sobre el gráfico log-log un ajuste lineal (podemos utilizar cualquier herramienta, pero quizás la más simple sea una hoja de cálculo tipo office), podremos obtener dichas constantes k y m de forma sencilla.

4.1 Fecha de entrega

- 10 de Octubre