PRÁCTICA II: Indexación de Textos

Parte I. Tratamiento de textos

October 3, 2016

1 Objetivo

El objetivo de la práctica es conocer los procesos claves en la creación de un índice para Recuperación de Información. Se debe trabajar en grupos. En particular, nos centraremos en las primeras fases de un proceso de indexación que nos permitirán analizar los documentos a indexar para extraer finalmente los tokens de indexación y construir un "índice" básico.

La práctica se realizará en grupo, si tienes algún problema en la formación del grupo puedes enviarme un correo para poder indicarte algún compañero que también este buscando grupo.

Al final de la práctica 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. Se recomienda seguir la metodología SCRUM para el desarrollo del proyecto.

2 Snowball

Snowball es un pequeño lenguaje de procesamiento de cadenas diseñado para poder reducir una palabra a su raíz (stemming). Así, por ejemplo, palabras como `toreado, toreados, toreándolo,torear, toreara, torearlo` tienen todos la misma raíz `tor`, que será el término que finalmente se indexará. Por tanto, esto nos permite incrementar el número de documentos que se pueden

encontrar para la consulta `torear` pues cualquier documento que contenga una palabra con la raíz `tor` será considerado como relevante, incrementando el recall.

Normalmente el proceso de stemming es heurístico, que aplica un conjunto de reglas a la cadena a stemizar. En esta práctica consideraremos Snowball (`http://snowballstem.org/`) como herramienta para realizar el stemming. Una demo de ejemplo de su uso (en inglés) lo podemos ver en `http://snowballstem.org/demo.html`.

Snowball se puede considerar como un lenguaje que nos permite declarar nuestras propias reglas para realizar el stemming, sin embargo, en esta práctica nos centraremos en como utilizar los algoritmos que ya tiene implementados, que permiten realizar el proceso en lenguajes como el Inglés (implementa el algoritmo de Porter), Ruso, Francés, Español, Italiano, Alemán, etc.

Más detalles de las reglas que emplean los distintos algoritmos, tanto en inglés como en español, los podemos encontrar en

`http://snowballstem.org/algorithms/english/stemmer` y

`http://snowballstem.org/algorithms/spanish/stemmer.html`,

respectivamente. Si estamos interesados en cómo funcionan dichos algoritmos se recomienda leer la documentación asociada.

2.1 Download

Snowball está disponible en varios lenguajes de programación como C, Java o Phyton y lo podremos descargar de `http://snowballstem.org/download.html`. En nuestro caso, nos centraremos en la versión en Java, pues lo debemos integrar con el resto del software que desarrollemos. Aunque en la página de la asignatura podemos encontrar el fichero **libstemmer.jar** para incluirlo en el class path de Java, estos son los pasos que deberías seguir para poder generarlo.

- Descarga el fichero tgz.
- Descomprimirlo
- Ve al directorio libstemmer_java y mira el directorio README.
- Sigue las instrucciones para compilarlo (utilizando javac)

- Crear el fichero jar: Ve al directorio libstemmer_java/java y ejecuta
  jar cvf libstemmer.jar *

2.2 Ejemplo de uso

Mostraremos el primer programa en Java para realizar el stemmer. Por ejemplo, desde Netbeans podemos crear una nuevo proyecto que llamaremos testsnowball, añadiéndole el fichero libstemmer.jar dentro de la carpeta de librerias.

```
1 import org.tartarus.snowball.ext.spanishStemmer;
2 ...
3 spanishStemmer stemmer = new spanishStemmer();
4 stemmer.setCurrent("termino");
5 if (stemmer.stem()){
6     System.out.println(stemmer.getCurrent());
7 }
```

La primera linea nos permite importar la clase spanishStemmer. setCurrent es un método que nos permite indicar que palabra queremos stemizar, en el ejemplo “termino”. Al ejecutar el método stem, se hace la transformación de la palabra, para dar como salida “caden”.

Hay que notar que si intentamos hacer el stemming de un string, como por ejemplo “estamos trabajando con una secuencia de cadenas”, sólo hará el stemming de la última palabra. Por lo tanto será necesario tokenizar dicho string, pudiendo utilizar la clase StringTokenizer. Será útil leer la documentación de la clase StringTokenizer para poder identificar de forma correcta los tokens en un fichero de texto. El siguiente ejemplo nos puede servir como punto de partida.

```
1  String texto="estamos trabajando con una secuencia de cadenas";
2  StringTokenizer tokens = new StringTokenizer(texto);
3  SnowballStemmer stemmer;
4
5  stemmer = (SnowballStemmer) new spanishStemmer();
6  while (tokens.hasMoreTokens()){
7        stemmer.setCurrent(tokens.nextToken());
8        if (stemmer.stem()){
9              System.out.println(stemmer.getCurrent());
10       }
```

```
11 }
```

La salida de este código será: “estam trabaj con una secuenci de caden”

Notar la diferencia en la declaración del objeto stemmer. Para más información se recomienda consultar la documentación y ejemplos que nos dan en Snowball.

2.3 Colecciones de documentos

Para esta práctica, el alumno dispondrá de un fichero de texto `quijote.txt` con la novela “El ingenioso hidalgo don Quijote de la Mancha”, de Miguel de Cervantes.

3 Tareas a realizar

Como entrada, tomaremos un documento de texto y lo procesaremos de forma que seamos capaces de identificar los distintos tokens (una vez eliminados signos de puntuación) que pasarán al proceso de indexación. En esta práctica utilizaremos la herramienta Snowball para hacer el stemming (http://snowballstem.org/)

El alumno deberá realizar un programa que lea el documento y obtenga el stem (token de indexación) asociado a cada término del mismo así como su número de ocurrencias de dicho token, esto es su frecuencia. Una vez obtenida dicha salida, debemos hacer un gráfico donde se presenten en el eje de las X los términos ordenados en orden decreciente de frecuencia y el 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

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

Por tanto, si realizamos sobre el gráfico log-log un ajuste lineal, podremos obtener dichas constantes k y m de forma sencilla.

4 A entregar

4.1 Fecha de entrega

1. Ley de Zipf: 21 de Octubre