PRACTICA 3:

Preprocesado de documentos con Lucene

Alumno: Pablo Valenzuela Álvarez

DNI: 76652136J

Correo: pvalenzuela@correo.ugr.es

Alumno: Francisco Javier García Maldonado

DNI: 76654015-Y

Correo: franelas@correo.ugr.es

1. Análisis comparativo de Tokens

En el primer documento que hemos analizado de los títulos del proyecto Gutenberg es el libro titulado "In the days of queen Mary", el cual contiene una cantidad total de términos, quitando palabras vacías, de 6609.
La frecuencia de todos ellos la podremos encontrar en la siguiente ruta relativa del proyecto:
./resultados/libros/Conteo-Analyzador-"nombreLibro".txt

Aquí mostraremos una captura de algunos de los más frecuentes a modo de ejemplo:

said,287;
will,246;
sir,213;
men,197;
john,153;
ralph,138;
geoffrey,134;
william,130;
great,127;
young,126;
day,124;
night,119;
now,117;
one,116;
susan,114;
lord,104;
yet,104;
king,103;
man,102;
may,101;
upon,98;
cried,92;
philip,90;
hand,88;
gutenberg,87;
project,87;
replied,86;

En esta imagen podemos observar claramente cuáles son los términos más frecuentes y cuales menos, y viendo el título del libro ya podemos ver por dónde van los tiros, claramente los nombres de los protagonistas son los que más se repiten, y luego podemos ver términos como "may" - debe, "King" – rey, "Queen" – reina, etc, que son un indicativo claro de en qué está basada la novela.

Sin embargo, si nos vamos a otro libro, también en inglés, para continuar viendo si se repiten y si la frecuencia de los términos es la misma o si tan siquiera serán los mismos, vemos que, por ejemplo, en el libro "Pride and Prejudice", también del proyecto Gutenberg la frecuencia de términos una vez quitadas las palabras vacías es la siguiente (solamente algunos de los términos ya que, si no, no cabría en el pdf):

mr,786;
elizabeth,597
will,422;
said,401;
darcy,374;
mrs,343;
much,329;
must,318;
bennet,294;
miss,283;
one,270;
jane,267;
bingley,257;
know,239;
though,226;
well,224;
can,221;
never,220;
soon,216;
think,211;
now,209;
may,208;
time,203;
good,201;
might,200;
every,198;
little,189;
lady,183;
sister,180;
without,178;
nothing,177;

En este otro documento, el número de términos, quitando palabras vacías es de 6440, que aproximadamente es el mismo número de términos que tenía el libro de estudio anterior. En este libro, podemos observar a través de la imagen anterior, que vuelven a repetirse términos como los nombres de los protagonistas y además podemos ver también otros términos que son comunes a ambos como, por ejemplo: will y said y además en las mismas posiciones prácticamente. Esto viene a confirmar que algunos de los términos se usan siempre

con la misma frecuencia, al igual que verifica la ley de Zypf.

2. Diseño de Analyzer propio

Hemos diseñado nuestra propia clase Analizador que extiende de la clase "Analyzer". Aquí hemos sobrescrito uno de los métodos de la clase de la que extendemos. Este método recibe como parámetro de entrada un "String" que contiene el texto a Analizar. Una vez que estamos dentro del método y hemos recibido esta variable ahora es momento de procesarla. Lo primero que haremos será quitar todos los signos de puntuación. Luego de esto, lo que haremos será pasar todas las letras mayúsculas a minúsculas. A continuación, con nuestra clase propia llamada "FiltraLetras" hacemos que todas las letras sueltas que encuentre, es decir, palabras de longitud menor o igual a 1 las elimine. Con esto nos quitaremos caracteres sueltos que podamos encontrar, o incluso algunas de las palabras vacías que encontremos en el texto. Por último, lo que hacemos es quitarle las palabras vacías y con esto ya acabamos.

Aquí tenemos una pequeña ilustración de lo que es el código:

```
@Override
protected Analyzer.TokenStreamComponents createComponents(String string) {

    Tokenizer tokenizer = new StandardTokenizer();
    TokenStream filter = new WordDelimiterFilter(tokenizer, 0, CharArraySet.EMPTY_SET);
    filter = new LowerCaseFilter(filter);
    filter = new FiltraLetras(filter);
    filter = new StopFilter(filter, CharArraySet.copy(pVacias));

    //throw new UnsupportedOperationException("Not supported yet."); //To change body of

    return new Analyzer.TokenStreamComponents(tokenizer,filter);
}
```

3. Documentación

https://www.gutenberg.org

https://www.ranks.nl/stopwords

https://lucene.apache.org/core/7_1_0/analyzers-common/org/apache/lucene/analysis/custom/CustomAnalyzer.html

https://citrine.io/2015/02/15/building-a-custom-analyzer-in-lucene/