Prácticas de
Sistemas de Recuperación de la Información

Cristóbal Rodríguez Alba

Sistema Básico

Antes de empezar a explicar cada práctica. Comentaré las estructuras básicas que he utilizado para su previo procesamiento.

Antes que nada he creado un ArrayList<String> para el fichero "conf.data" para crear tanto carpetas como los ficheros que utilizare para evitar tanto rutas relativas como rutas absolutas. A continuación creo un bucle para indicar un índice para cada línea de este archivo.

Declaro variables para controlar el tiempo de ejecución. También creo un vector de File para cada documento de la colección, que recorreré para cada documento.

Para poder ejecutar bien la aplicación, estos documentos son vitales para su funcionamiento y no se deben borrar o modificar:

- coleccionEsUja2017 Contiene la colección de documentos
- library Contiene las librerías que han sido utilizadas
- src Contiene el código de la aplicación
- conf.data Contiene el nombre de las carpetas que crearemos
- palabrasVacias.txt Contiene las palabras vacías que eliminaremos

Las carpetas index, normalizate, stemmer y stopper se crearan con la ejecución de la aplicación.

Práctica 1.1. Procesar colección HTML

He creado un módulo para esta práctica llamado "Tokenization".

- La primera función que tengo es "removeHTML" que devuelve el texto del documento pero sin las etiquetas HTML. Cómo las etiquetas son en lengua inglesa utilizamos la codificación UTF-8. Este texto obtiene todo el texto del documento HTML incluyendo cabecera, pie de página, teléfono, dirección, etc.

- La segunda función que tengo es "cleanAccents" que devuelve el texto introducido como parámetro que es el previamente devuelto en la función anterior, con las siguientes modificaciones:
  - Cambio todo el texto a minúscula.
  - Elimino tildes, diéresis y todo tipo de acento.
  - Elimino todo aquel carácter que no esté en este intervalo:
    ['a'...'z', 'A'...'Z', ' ', '_', '-', '\n','0'..'9']
  - Recorro un bucle por cada palabra hasta llegar a un espacio, un guion o un salto de línea.

  A continuación en el programa principal creo una carpeta llamada "normalizate" donde irán los documentos con las palabras tokenizadas.

- La tercera función que tengo es "tokenization" que devuelve un número entero que cuenta las palabras de cada documento. Tiene dos parámetros, uno es el texto que queremos tratar y el otro es la ruta donde escribiré este texto tokenizado, es decir, tendré tantas líneas en el documento como palabras haya.

Para esta práctica he utilizado la librería StringUtils para eliminar acentos del texto y Jsoup para la eliminación de las etiquetas HTML. Estas librerías las añado en una carpeta llamada “library” en la raíz de la aplicación.

Al final de mi ejecución obtengo los siguientes datos, salvo las etiquetas HTML que como previamente comente se eliminaron todas:

```
---------------------------------------------
-       AFTER TOKENIZATION          -
---------------------------------------------
```

We have: 540605 words.	Número de tokens
We have: 645 per file.	Número medio de tokens por documento
We have: 838 documents.	Número total de documentos
The task has taken 3495 milliseconds.	Tiempo en realizar el proceso

Práctica 1.2. Normalización (stopper)

He creado un módulo para esta práctica llamado “Stopper”.

- La primera función es “loadEmptyWords” que devuelve un vector de String que contendrá las palabras vacías que he descargado desde http://members.unine.ch/jacques.savoy/clef/. Cómo parámetro le paso la ruta donde encuentro el archivo, en mi caso lo leo desde el archivo conf.data.

- La segunda función es “tokenizationAfter” que devuelve el texto ya procesado sin palabras vacías. Como parámetro le pasamos el vector de palabras vacías, la ruta donde guardare dichos textos dentro de la carpeta “Stopper” correspondiendo con cada documento y el texto que ya obtuve de la práctica anterior.

En el programa principal utilizo la librería StringUtils para poder contar cada palabra del nuevo texto que ha sido procesado y una variable general para guardar el total de palabras. Pasado el bucle donde obtengo los textos procesados calculo la media por archivos, máximo y mínimo antes y después de su procesado.

Para la función de calcular las 5 más frecuentes es un poco más compleja, como parámetro le paso la dirección de la ruta en el archivo anteriormente nombrado, esta función devuelve un vector de Pares (que es una clase creada junto al programa principal) de String y Enteros. Dicha función empieza con una cola de prioridad y vamos creando un mapa de String y Enteros con la funcionalidad de que si la palabra recorrida se encuentra en el mapa añade en uno su frecuencia y si no se encuentra que le añade un 1 y añada la palabra. Luego para cada fila del mapa voy recorriendo y añadiendo en la cola de prioridad que cree al principio de la función y ya solamente me queda recorrer en 5 o el tamaño que desee y voy añadiendo al vector desde la cola que será devuelto por la función.

- Before of remove empty words: - Antes de eliminar las palabras vacías

---

We have: 540605 words. Total de palabras

We have: 645 per file. La media por cada documento

Maximum is: 1956 words. Máximo de palabras

Minimum is: 205 words. Mínimo de palabras

Top 5 are: Las 5 palabras más frecuentes

1. de: 49753 times
2. la: 24951 times
3. el: 14650 times
4. en: 11892 times
5. y: 11798 times

- After of remove empty words: - Después de eliminar las palabras vacías

---

We have: 323059 words. Total de palabras

We have: 385 per file. La media por cada documento

Maximum is: 1060 words. Máximo de palabras

Minimum is: 162 words. Mínimo de palabras

Top 5 are: Las 5 palabras más frecuentes

1. jaen: 8422 times
2. diario: 5047 times
3. universidad: 4378 times
4. uja: 4043 times
5. 21: 3507 times

The task has taken 6359 milliseconds. Tiempo en realizar el proceso

Práctica 1.3. Normalización (stemmer)

Para empezar he elegido la librería “Snowball” que puedes descargarla aquí http://snowball.tartarus.org/download.html

He creado un módulo llamado “stemmer” en el que extraerá las raíces de cada palabra.

- Sólo tengo una función llamada “noRoots” que los parámetros que tengo son la ruta donde guardare estos nuevos textos procesados y el texto que voy a procesar con la librería mencionada.

En el programa principal utilizo el texto de la práctica anterior y con la librería de StringUtils contare las palabras y otra variable para el total de palabras como anteriormente hice. Para la salida por pantalla realizare las mismas operaciones que utilizamos en la práctica anterior.

- After of remove roots: - Después de eliminar las raices

We have: 323059 words. Total de palabras

We have: 385 per file. La media por cada documento

Maximum is: 1060 words. Máximo de palabras

Minimum is: 162 words. Mínimo de palabras

Top 5 are: Las 5 palabras más frecuentes

1. jaen: 8422 times

2. diari: 5063 times

3. univers: 5004 times

4. uja: 4932 times

5. 21: 3507 times

The task has taken 8939 milliseconds. Tiempo en realizar el proceso

Como se ve las cantidades no cambian, solo cambian las palabras

Práctica 1.4: Creación de pares palabra-frecuencia y ordenación

La Estructura de datos que utilizaré para esta práctica será de un mapa de String, Enteros.

Para ello he creado el módulo “Index”.

- Solo tengo una función llamada “loadDataStructure” que se basa en la función de crear las 5 palabras más frecuentes pero con la mínima diferencia de que dentro de la función también comprobamos si es mayor o menor para devolverlos por pantalla.

La Estructura de Datos general que utilizare será un mapa de mapas de String, Enteros, es decir, un mapa del índice del documento que contiene un mapa de palabras con frecuencias.

En el programa principal en el bucle donde recorro los documentos guardo el índice del documento y la ruta absoluta de dicho documento para después obtener los nombres de los documentos.

- After of create index: - Después de crear índice

We have: 838 files. Total de documentos

The document shortest is: es_46056.html, with: 163 words. Documento más corto

The document longest is: es_44660.html, with: 1061 words. Documento más largo

We have: 11377 differents words. Diferentes palabras

The task has taken 7833 milliseconds. Tiempo en realizar el proceso

Práctica 1.5: Fichero de pesos sin normalizar. Fichero de pesos normalizado.

Para esta práctica he creado una carpeta llamada “Index” donde contienen dos subcarpetas llamadas IDF y Wnij que cada una contiene un archivo XML donde contienen la estructura de datos obtenida.

He creado un módulo llamado “VSM”. Con las siguientes funciones:

- La primera función llamada “highestFrequency” normaliza los pesos de cada palabra en cada documento por lo que devuelve un mapa de mapas como anteriormente mencioné y como parámetro la ruta que procesaré. Esta función además de guardar el mapa con sus palabras de cada documento también guarda el máximo de veces que se repite una palabra. Al final vuelvo a crear un mapa igual pero ahora solo tengo que dividir todas las frecuencias entre el máximo para poder normalizar cada palabra dentro del documento.

$$\frac{tf_{ij}}{max\{tf_{xj} \forall x \in j\}}$$

Donde, tf_{ij} es la frecuencia de la palabra i en el documento j

- La segunda función crea los IDFs por lo que devuelve un mapa con las palabras y sus frecuencias, sus parámetros son el mapa de mapas creados anteriormente y el tamaño de la colección. Recorremos esos mapas y si no están en el mapa nuevo que devuelve le añadimos uno y si está sumamos uno a su frecuencia, ya para poder normalizarlo solo nos falta recorrer este segundo mapa y realizarle un algoritmo a la división entre el tamaño de la colección y la frecuencia de la palabra.

A continuación, aplicamos los pasos vistos en teoría sobre el modelo espacio vectorial, para así calcular los IDFs, los pesos sin normalizar, las normas y los pesos normalizados.

$$idf_i = \log(\frac{N}{df_i})$$

- La tercera función crea los wij que son los pesos de las palabras sin normalizar y consiste en multiplicar la palabra i del mapa de mapas creado en la primera función y la palabra i en el mapa de la segunda función, necesitaré los parámetros del mapa de mapas y los IDFs.

$$w_{ij} = tf_{ij} \times idf_i$$

- La cuarta función llamada “wnij” consiste en normalizar estos pesos con sus normas, para ello utilizaré la siguiente fórmula:

$$wn_{ij} = \frac{w_{ij}}{\sqrt{\sum_{\forall x \in j} w_{xj}^2}}$$

Donde:

- df_i es la frecuencia documental de la palabra i
- w_{ij} es el peso de la palabra i en el documento j

Práctica 1.6: Preprocesado de la consulta y cálculo de la similitud.

En esta práctica creo una nueva aplicación llamada “Searcher” que será mi buscador on-line.

Las cosas a tener en cuenta es que para que esta aplicación funcione correctamente debemos de descargarla junto a la otra aplicación llamada “IRS”, otra cosa vital para su funcionamiento es que se ejecute primero la aplicación “IRS” para que pueda crear las estructuras de datos necesarias para su correcto funcionamiento.

Tanto la consulta como el número de documentos que deseamos visualizar son pedidos por pantalla y debemos introducirlos mediante la misma.

Para calcular la similitud he creado el módulo llamado “Similarity” con las siguientes funciones:

- La primera función llamada “sim” calcula la similitud entre la consulta y los documentos de la colección mediante la siguiente fórmula:

$$\text{sim}(d_j, q) = \frac{\sum_{\forall i \in q} wn_{ij} * wn_{iq}}{\sqrt{\sum_{\forall i \in j} wn_{ij}^2} * \sqrt{\sum_{\forall i \in q} wn_{iq}^2}}$$

  Que devuelve un mapa con la identificación del documento y la similitud que tiene con la consulta. En mi caso no hace falta dividirlo por la norma porque estos datos ya han sido normalizados previamente por lo que su resultado ya es normalizado.

- La segunda función llamada “sortByValue” ordena un mapa dado mediante su valor en orden de mayor a menor. Que necesita como parámetro el mapa que queremos ordenar.

En el programa principal de esta aplicación, creo el vector de String con tantos datos como líneas tiene mi archivo de configuración “conf.data”. Justo después cargo las estructuras de datos necesarias (WNij, IDF, ID) además del fichero de palabras vacías para su siguiente procesado. Realizo la consulta de forma on-line hasta que mi consulta sea la palabra clave “stop” para dejar de realizar consultas. Una vez introducida la consulta realizo su procesado (Tokenización, “stopper”, “stemmer”) y ya sólo nos falta calcular los pesos normalizados con la fórmula de la práctica anterior pero esta vez para la consulta. El programa principal tiene una serie de funciones privadas que son las siguientes:

- La primera función llamada “doQuery” que pide la consulta que trataremos de manera on-line. Que devuelve el texto de dicha consulta.

- La segunda función llamada “load” que carga desde el archivo de configuración “conf.data” los parámetros necesarios para su funcionamiento. Que devuelve el vector de String que tiene el principio de nuestro programa principal.

- La tercera función llamada “modify” trata el texto de la consulta y devuelve ya la consulta procesada.

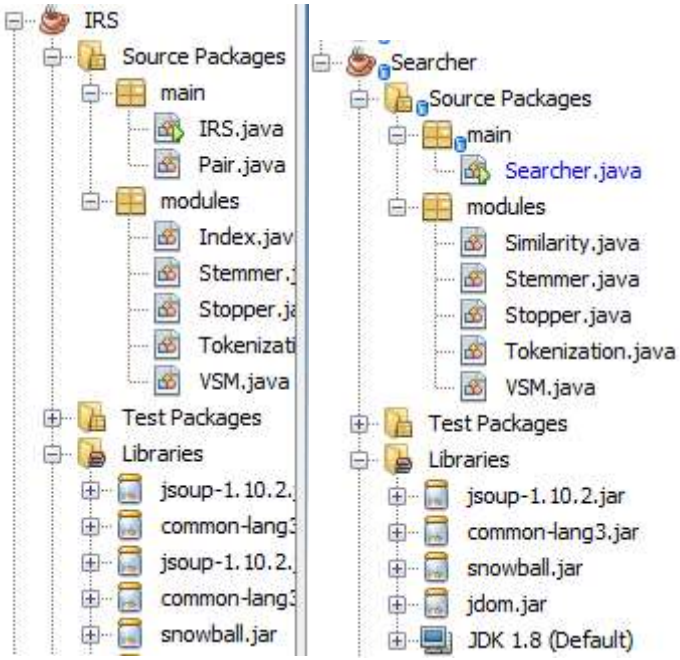
- La cuarta, quinta y sexta función carga las estructuras de datos necesarias y cómo parámetros necesita la ruta de los archivos XML que serán cargados en el mapa que corresponda a cada estructura de datos.

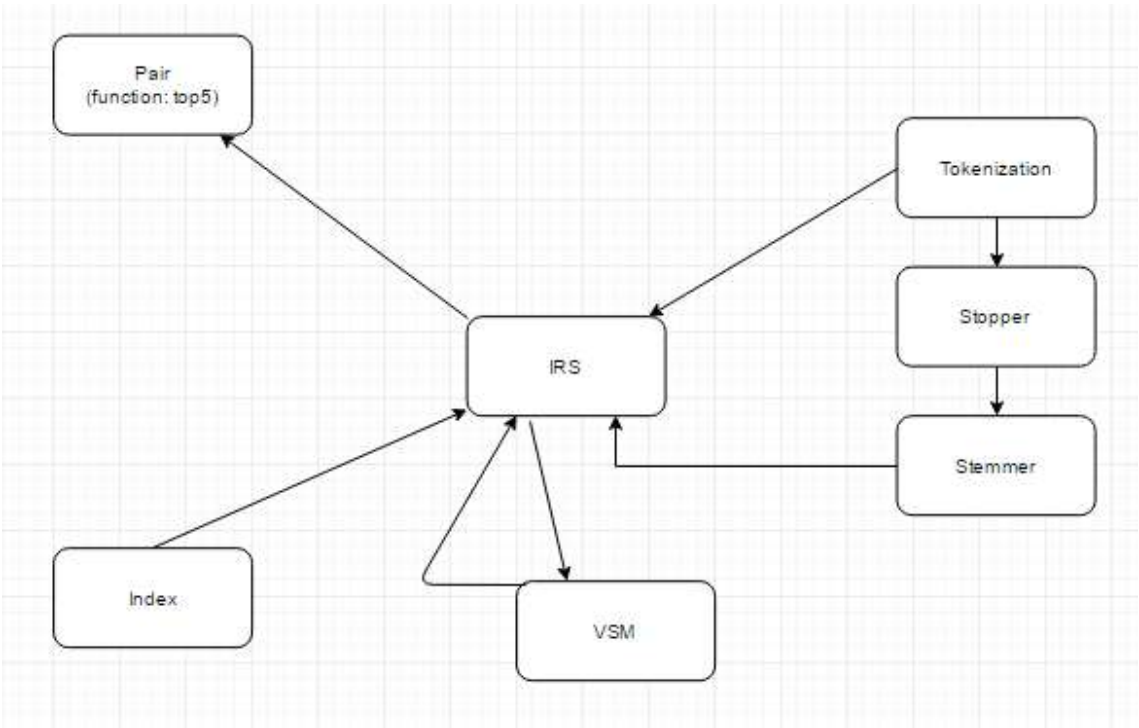
- La séptima función llamada “show” muestra un mensaje de error en el caso de que la consulta solo contenga palabras vacías o palabras que se repitan en todos los documentos, en el caso de que la consulta sea la correcta devolverá el nombre del documento, el título del documento y la primera frase donde aparece una de las palabras clave de la consulta. Cómo parámetros necesita el mapa de similitud, la colección original de documentos, el tamaño de documentos que deseo mostrar por pantalla que se pedirá de forma on-line también y la consulta a tratar.

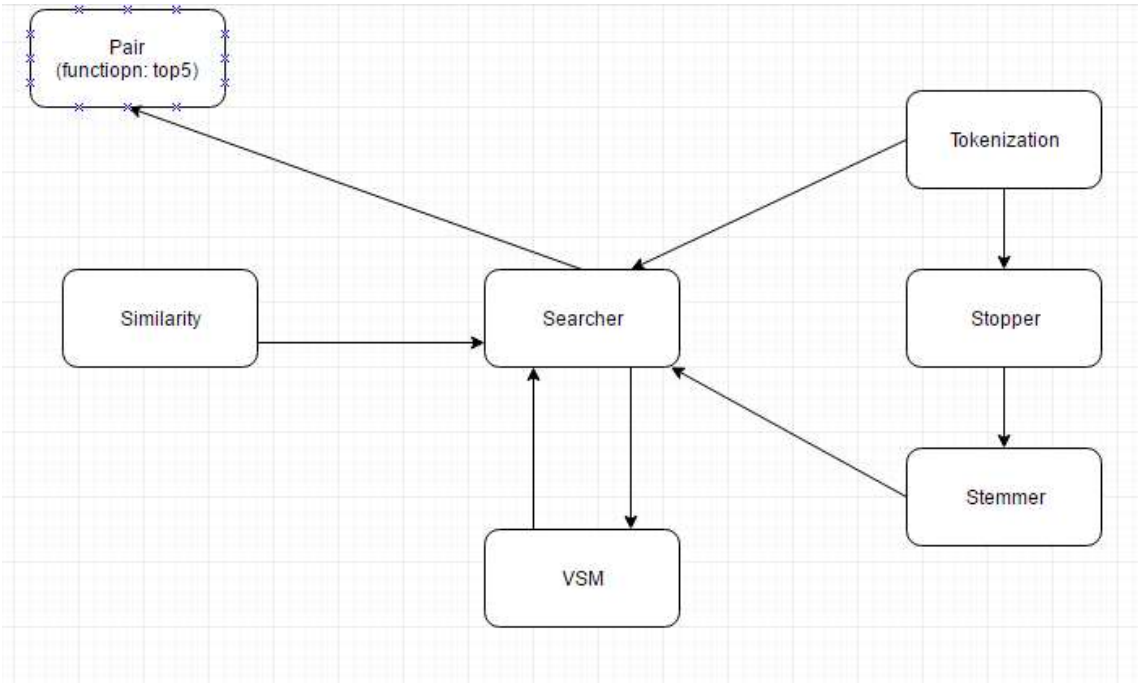
Práctica 1.7: Finalización del sistema básico y entrega.

- Introducción, explicando de forma general cómo se han realizado las prácticas:
  Al principio de cada práctica desarrollo las ideas generales que he ido desarrollando

- Esquema de las prácticas, con una figura que muestre los módulos desarrollados o conexión de clases:







Para la aplicación “IRS” el programa principal es IRS, la clase Pair sirve para las palabras más frecuentes y tanto los módulos como las librerías se han explicado en cada práctica.
Para la aplicación “Searcher” el programa principal es Searcher y tanto los módulos como las librerías se han explicado en cada práctica.

- Explicación de las estructuras de datos utilizadas y justificación

  En cada práctica donde realizo las estructuras de datos justifico porque elijo esa misma. En la estructura de datos de “IdCollection” el primer campo es el índice en el vector y el segundo la ruta absoluta del fichero

- Desarrollo de las prácticas, con detalles más concretos de implementación y con las decisiones adoptadas en cada una de ellas (no es necesario que incluyáis aquí código fuente)

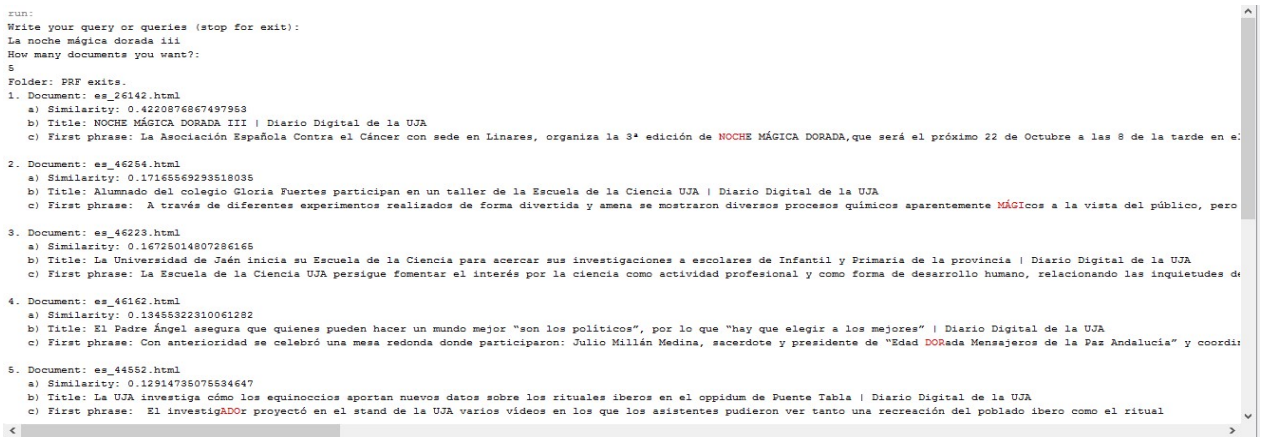
  En cada práctica donde realizo la implementación también está justificada la decisión de ese desarrollo

- Ejecución de la aplicación/aplicaciones
  Cada ejecución solicitada está seguido de la explicación de cada práctica

- Parámetros de la aplicación/aplicaciones
  Los únicos parámetros que hay que tener en cuenta son los escritos en el archivo de configuración “conf.data”

- Pruebas realizadas
  Para cada prueba mostraba un mensaje por pantalla con lo que necesitaba testear y por ejemplo para las consultas mi frase típica era: La noche mágica

Para la consulta, “La noche mágica dorada iii”:



```
run:
Write your query or queries (stop for exit):
La noche mágica dorada iii
How many documents you want?:
5
Folder: PRF exits.
1. Document: es_26142.html
   a) Similarity: 0.4220876867497953
   b) Title: NOCHE MÁGICA DORADA III | Diario Digital de la UJA
   c) First phrase: La Asociación Española Contra el Cáncer con sede en Linares, organiza la 3ª edición de NOCHE MÁGICA DORADA,que será el próximo 22 de Octubre a las 8 de la tarde en e

2. Document: es_46254.html
   a) Similarity: 0.17165569293518035
   b) Title: Alumnado del colegio Gloria Fuertes participan en un taller de la Escuela de la Ciencia UJA | Diario Digital de la UJA
   c) First phrase:  A través de diferentes experimentos realizados de forma divertida y amena se mostraron diversos procesos químicos aparentemente MÁGIcos a la vista del público, pero

3. Document: es_46223.html
   a) Similarity: 0.16725014807286165
   b) Title: La Universidad de Jaén inicia su Escuela de la Ciencia para acercar sus investigaciones a escolares de Infantil y Primaria de la provincia | Diario Digital de la UJA
   c) First phrase: La Escuela de la Ciencia UJA persigue fomentar el interés por la ciencia como actividad profesional y como forma de desarrollo humano, relacionando las inquietudes d

4. Document: es_46162.html
   a) Similarity: 0.13455322310061282
   b) Title: El Padre Ángel asegura que quienes pueden hacer un mundo mejor “son los políticos”, por lo que “hay que elegir a los mejores” | Diario Digital de la UJA
   c) First phrase: Con anterioridad se celebró una mesa redonda donde participaron: Julio Millán Medina, sacerdote y presidente de “Edad DORada Mensajeros de la Paz Andalucía” y coordi

5. Document: es_44552.html
   a) Similarity: 0.12914735075534647
   b) Title: La UJA investiga cómo los equinoccios aportan nuevos datos sobre los rituales íberos en el oppidum de Puente Tabla | Diario Digital de la UJA
   c) First phrase:  El investigADOr proyectó en el stand de la UJA varios vídeos en los que los asistentes pudieron ver tanto una recreación del poblado ibero como el ritual
```

Para la consulta con palabras vacías, “de la sobre”:

```
Write your query or queries (stop for exit):
de la sobre
How many documents you want?:
5
Folder: PRF exits.
This query isn´t relevant
```

Para la consulta de cualquier texto recogido:

```
Write your query or queries (stop for exit):
Concretamente, un centenar de estudiantes de 1º y 2º de Infantil del Colegio Tucci participaron este pasado martes en el taller “Parece magia, pero es… ¡Química!”, dirigido por los pro
How many documents you want?:
5
Folder: PRF exits.
1. Document: es_46223.html
   a) Similarity: 7.986568216363927
   b) Title: La Universidad de Jaén inicia su Escuela de la Ciencia para acercar sus investigaciones a escolares de Infantil y Primaria de la provincia | Diario Digital de la UJA
   c) First phrase: Campus Las Lagunillas s/n | 23071 – Jaén Tlf: +34 953 21 21 21 | Fax: +34 953 21 22 39 | info@ujaen

2. Document: es_46254.html
   a) Similarity: 7.012308016511749
   b) Title: Alumnado del colegio Gloria Fuertes participan en un taller de la Escuela de la Ciencia UJA | Diario Digital de la UJA
   c) First phrase: Campus Las Lagunillas s/n | 23071 – Jaén Tlf: +34 953 21 21 21 | Fax: +34 953 21 22 39 | info@ujaen

3. Document: es_44065.html
   a) Similarity: 1.3675251471069247
   b) Title: La UJA realizará juegos y experimentos para mostrar la cara divertida de la ciencia en la Noche Europea de los Investigadores | Diario Digital de la UJA
   c) First phrase: Campus Las Lagunillas s/n | 23071 – Jaén Tlf: +34 953 21 21 21 | Fax: +34 953 21 22 39 | info@ujaen

4. Document: es_44531.html
   a) Similarity: 1.2352676189404186
   b) Title: Una treintena de estudiantes del IES María Bellido de Bailén participan en los talleres sobre Ingenierías de la EPS de Linares | Diario Digital de la UJA
   c) First phrase: Campus Las Lagunillas s/n | 23071 – Jaén Tlf: +34 953 21 21 21 | Fax: +34 953 21 22 39 | info@ujaen

5. Document: es_46581.html
   a) Similarity: 1.1405626771886042
   b) Title: La UJA celebrará en septiembre ‘La Noche Europea de los Investigadores’ con alrededor de una treintena de actividades interactivas para acercar la ciencia a la ciudadanía |
   c) First phrase: Campus Las Lagunillas s/n | 23071 – Jaén Tlf: +34 953 21 21 21 | Fax: +34 953 21 22 39 | info@ujaen
```

- Conclusión y valoración personal
  Es una de las prácticas que de verdad me ha gustado en los tres años que llevo de carrera y es un tema que poco a poco me ha interesado investigar e incluso trabajar sobre esto. Aconsejo 100% la realización de estas prácticas
  Para poder ver mi código lo comparto a través de mi github personal:
  https://github.com/criso123

Mejoras del sistema

Voy a realizar la mejora 5: Snippets que consiste en mostrar por pantalla el título, frase donde aparece la consulta y el enlace al documento relevante, para ello utilizo un nuevo módulo llamado "snippet" que es del tipo Jframe, recojo desde la función show de "searcher" título, nombre del documento, ruta absoluta y frase mediante dos mapas y cada vez que realizamos una consulta se dispara esta aplicación. Esta aplicación está dirigida solamente a los 5 primeros documentos sin contar el número de documentos que queremos en la consulta por terminal desde el sistema básico.

A partir de aquí realice la primera mejora: Aplicar Pseudorealimentación por relevancia (PRF), que consiste en obtener los N documentos relevantes y de cada documento relevante las M(5) palabras más frecuentes, uno de los requisitos que valoré fue que para poder introducir esas palabras frecuentes a la nueva consulta debía de cumplir que su peso sea distinto de 0, es decir, que la palabra sea relevante. Después de mantener la consulta la volvemos a lanzar para que tenga más precisión a la hora de recuperar los documentos.

Para ello cree un módulo llamado PRF que contiene la función del sistema básica para encontrar las 5 palabras más frecuentes pero con un ligero cambio de en vez de la colección dada solo las busca de un documento concreto.

También en el código principal creo una carpeta llamada "PRF" para los documentos recibidos de las consultas para poder tratar su texto más adelante. También creo un mapa para poder quitar los repetidos e incluirlos posteriormente en la consulta final. Ejecuto dos veces la función show que ha sido cambiada, la primera de inicio y la segunda para obtener mayor precisión.

- La función show ahora devuelve un vector de String con las palabras más frecuentes. Aparte he añadido varios parámetros más como son: la similitud entre palabra y documentos, la colección de documentos, el tamaño de la colección, la ruta absoluta de la colección, la consulta, la ruta donde guardare la carpeta nueva "PRF", el vector de palabras vacías y la estructura de datos con los pesos de cada palabra.

También he creado otra mejora que aparece en el código de la función show y es la mejora 6: Resaltar términos de búsqueda, en vez de negrita yo he querido resaltarlo con un color rojo en el terminal y mediante "#" en la aplicación disparada desde el módulo "Snipper", para ello sólo he tenido que copiar antes del texto de la palabra encontrada de la consulta un código de escape que colorea ese texto.