Grado en Ingeniería Informática

SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

Sistema Básico de Recuperación

Alumno: Sergio Perea de la Casa.

DNI: 77433569-K.

Correo: spc00033@red.ujaen.es

Profesora: Pilar López Úbeda.

Horario: Viernes (10:30 - 12:30).

Descripción del problema

El informe va dirigido al desarrollo incremental de un Sistema de Recuperación de Información mediante el lenguaje de programación Python 3.8.

En él se realizarán los siguientes pasos para la construcción progresiva de ello:

1. Procesamiento de la colección de documentos textuales.
2. Normalización (stopper).
3. Normalización (stemmer).
4. Creación de pares palabra-frecuencia y ordenación.
5. Fichero de pesos sin normalizar y normalizados.
6. Preprocesamiento de la consulta y cálculo de la similitud.
7. Análisis final del sistema básico de recuperación de información.

Estos pasos han sido divididos en diferentes archivos de python, los cuales se ha creado una clase específica para cada uno de los puntos. Por lo que la explicación de su desarrollo se realizará posteriormente en el informe conforme se avance en dichos puntos.

Parámetros de configuración

Para que el sistema funcione correctamente es importante tener en cuenta que tanto los archivos ejecutables python deben de estar en la misma carpeta todos y además debe de estar en dicha carpeta el archivo ***conf.ini*** donde se almacenará toda información complementaria para la posible realización de la práctica desde cualquier ordenador.

Tras dicho requisito, se explica a continuación el significado de cada uno de los parámetros que aparecen en el ***conf.ini*** para su posible ejecución añadiendo las localizaciones de los directorios y archivos correspondientes:

Primero de todo, se va a diferenciar entre la configuración **offline** y **online**. Esto se indica con un primer parámetro con dichos nombres teniendo así una mayor organización:

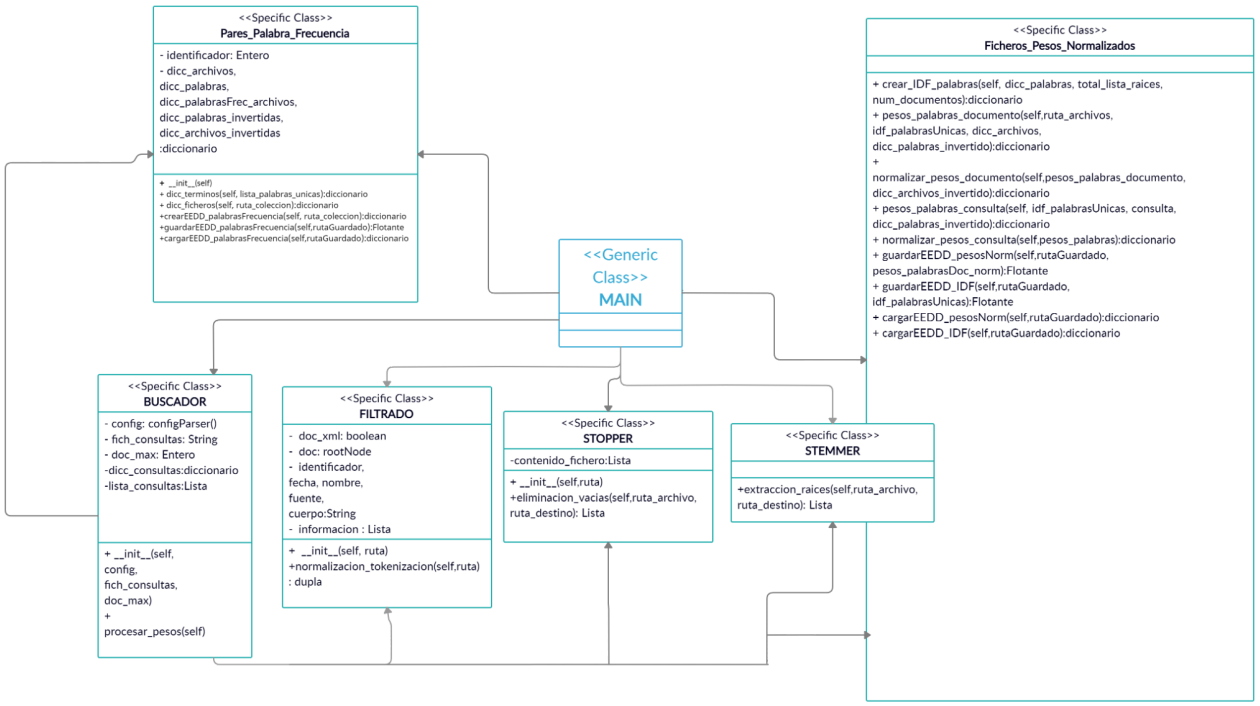
Para la configuración **OFFLINE**:

- **ruta_coleccion_inicio.** Directorio donde se encuentra la colección de documentos sin tratar inicialmente.
- **ruta_coleccion_normalizada.** Directorio donde se almacenará y se recogerá la colección de documentos filtrados con su tokenización (punto 1).
- **ruta_lista_stopword.** Directorio más el nombre del archivo que tratará las palabras vacías (punto 2).
- **ruta_coleccion_stopper.** Directorio donde se almacenará y se recogerá la colección de documentos sin palabras vacías (punto 2).
- **ruta_coleccion_stemmer.** Directorio donde se almacenará y se recogerá la colección de documentos a los que se le ha aplicado el stemmer (punto 3).
- **ruta_almacen_palabras_frecuencia.** Directorio donde se almacenará y se recogerá el diccionario de palabras-frecuencia pedido en la creación de pares palabra-frecuencia (punto 4).
- **ruta_diccionarios_invertidos.** Directorio donde se almacenarán y se recogerán todos los diccionarios invertidos como son el diccionario invertido de palabras y documentos (necesario para el punto 6).
- **ruta_archivo_historial.** Directorio donde se creará un documento de texto que contendrá la información que se ha ido pidiendo sobre el progreso de las prácticas (para observar resultados desde la ejecución de un ordenador diferente).
- **ruta_coleccion_ficherosNormalizados.** Directorio donde se almacenará y se recogerá los diccionarios de IDF de las palabras y la de los pesos normalizados (punto 5).

Para la configuración **OFFLINE**:

- **ruta_fich_consultas.** Directorio más el nombre del fichero de consultas, para su futuro procesamiento (punto 6).
- **ruta_fich_consultas_modificadas.** Directorio donde se creará y se irá modificando un fichero que contiene las consultas con la normalización, stopper y el stemmer hechos (punto 6).
- **ruta_ficheros_consultas_resultados.** Directorio donde se almacenarán los resultados por consulta, siendo cada fichero de resultado creado en el mismo directorio (punto 6).
- **num_doc_relevantes.** Número máximo de documentos relevantes por consulta.

Esquema de los módulos desarrollados



Como se puede observar en el esquema, tenemos un programa MAIN el cual se encarga de la ejecución completa del Sistema Básico de Recuperación. Dicho MAIN usa todos los demás módulos/clases por lo que existe una relación hacia ellos, y dichas relaciones en su mayoría indican la parte offline del código exceptuando la relación con la clase BUSCADOR. Por ello, la clase BUSCADOR vuelve a estar relacionada con los métodos necesarios para realizar la parte online del código, indicados con sus correspondientes flechas.

Por último, es importante destacar que en la ejecución del código del MAIN hay una variable booleana al inicio, la cual indica si se quiere realizar la parte offline o no del código. Por defecto, dicha variable se encuentra en False por lo que no se ejecutará la parte offline.

Estructuras de datos utilizadas

Antes de entrar en detalle de implementación, se explicará qué estructuras de datos se han ido utilizando a los largo de los módulos/clases del programa. También es importante destacar que, tras la implementación de código en Python, las estructuras de datos usadas son referentes a dicho lenguaje de programación.

EEDD en la clase Filtrado

- **Lista**: Estructura usada para recoger toda la información procedente del documento que se va a procesar gracias a la ruta pasada como parámetro. Esto es, donde se almacenará todo el contenido indicado del fichero a procesar.

EEDD en la clase Stopper

- **Lista**: Estructura usada para recoger toda la información procedente del documento que se va a procesar gracias a la ruta pasada como parámetro.

EEDD en la clase Stemmer

- **Lista**: Estructura usada para recoger toda la información procedente del documento que se va a procesar gracias a la ruta pasada como parámetro. Esto es, donde se almacenará todo el contenido indicado del fichero a procesar.

EEDD en la clase Pares_Palabra_Frecuencia

- **Diccionario**: Estructura usada para el almacenamiento de las palabras únicas de la colección, los archivos de la colección, los identificadores tanto de las palabras como de los archivos (diccionarios inversos) y para almacenar los pares palabra-frecuencia.

EEDD en la clase Ficheros_Pesos_Normalizados

- **Lista**: Estructura usada para la recogida de información de los documentos, en su mayoría, y para el uso en estilo de par palabra-apariciones con lo que a posteriori se pasará a la siguiente estructura de datos.
- **Diccionario:** Estructura usada para almacenar la información correspondiente a los IDF de las palabras y los pesos de las palabras normalizadas tanto para la parte offline (documentos) como para la online (consultas).

EEDD en la clase BUSCADOR

- **Lista**: Estructura usada para la obtención de los resultados del preprocesamiento de las consultas (tokenización, stopper, stemmer).
- **Diccionario:** Estructura usada para almacenar las similitudes de los documentos respecto a una consulta, entre otros.

EEDD en la clase principal MAIN

- **Lista**: Estructura usada para la obtención de los resultados de las ejecuciones pedidos en los diferentes guiones de prácticas.
- **Diccionario:** Estructura usada como variable que recoge los diccionarios devueltos por los diferentes métodos de los objetos de diferentes clases.

Desarrollo, implementación y decisiones

En este módulo se explicará la metodología de desarrollo que se ha ido aplicando en cada uno de los módulos/clases del Sistema.

Desarrollo del programa principal, MAIN

Es el módulo principal donde se abrirá el archivo de configuración y se gestionará todas las rutas explicadas anteriormente. Las diferentes llamadas a las ejecuciones de los diferentes módulos para el desarrollo offline del sistema de recuperación básico se basan en un primer booleano que indica si se realizará o no dichas ejecuciones. En dicha parte offline, se detalla punto por punto hasta el punto 5 inclusive (explicados en la descripción). En cada uno de estos puntos, separados por unos comentarios, se va ejecutando el filtrado, stopper, stemmer, creación de pares palabra-frecuencia y la normalización de los pesos de las palabras. Una vez terminada la parte offline, se ejecuta la parte online donde se crea un objeto de la clase BUSCADOR para poder llamar a los métodos que desarrollen dicha parte.

Desarrollo de la clase FILTRADO

Lo primero a explicar es el constructor parametrizado. En él se diferencia si dicho documento extraído de la ruta pasada como parámetro tiene un formato xml o txt. En caso de tener un formato xml se aplica, mediante la librería minidom de xml.dom, la diferenciación de contenido entre el identificador del documento, fecha, título, fuente y descripción. Por último, todo ello se recoge en una lista llamada "*informacion*". En caso de que el documento sea txt, todo se resume a recoger en dicha lista *información* el contenido que aparece en dicho texto marcando cada elemento de la lista las líneas.

Por otro lado, está el método de ***normalizacion_tokenizacion****.* En él, se consigue procesar dependiendo del formato del documento (xml o txt) y así en un mismo método poder procesar tanto la parte offline de los documentos de la colección como la online de las consultas. A destacar, en este método se debe saber que las tildes en las palabras no han sido eliminadas además de hacer todo lo correspondiente para la limpieza de dichas palabras (minúsculas, no eliminar signos indicados en el guión correspondiente y eliminar todo aquello diferente al alfabeto y a los dígitos). Por último, devuelve una tupla que contiene información necesaria para la documentación a añadir de dicha parte en el main.

Desarrollo de la clase STOPPER

En dicha clase, tenemos el constructor parametrizado donde se procesa y almacena en una lista llamada *contenido_fichero* todo el contenido del documento a partir de la ruta pasada como parámetro.

Por otro lado, tenemos el método ***eliminacion_vacias*** el cual se encarga de, a partir de un fichero que contiene las palabras vacías (cuya ruta se pasa como parámetro), eliminar todas las palabras que no nos servirán para la recuperación de documentos a partir de unas consultas.

Desarrollo de la clase STEMMER

En esta clase, únicamente aparece el método ***extraccion_raices*** el cual se encarga, gracias a la librería **PorterStemmer**, de cambiar las palabras de un documento (cuya ruta del documento se pasa como parámetro) por la raíz correspondiente de dicha palabra. Estas nuevas palabras serán almacenadas en un fichero dentro de la ruta destino pasada como parámetro.

Desarrollo de la clase PARES_PALABRA_FRECUENCIA

Lo primero que tiene es el constructor por defecto, el cual debe de crear los diccionarios necesarios para el futuro desarrollo de estos diccionarios en los diferentes métodos de la clase.

Los siguientes métodos que aparecen son los correspondientes a la creación de los diccionarios de palabras de la colección (***dicc_terminos***) y de archivos de la colección (***dicc_ficheros***). Estos métodos completan los diccionarios correspondientes que fueron instanciados en el constructor. También, a su vez, crean los diccionarios invertidos tanto de las palabras como de los archivos.

El método ***crearEEDD_palabrasFrecuencia*** es el destacado de esta clase. Se encarga de añadir a un diccionario la frecuencia de una palabra en un documento en concreto, haciendo esto para todos los documentos de la colección.

Por último, aparecen los métodos encargados de **guardar** y **cargar** las estructuras de datos (diccionarios) que se habían creado en el constructor por defecto.

Desarrollo de la clase FICHEROS_PESOS_NORMALIZADOS

Para explicar el desarrollo de los métodos de esta clase es importante destacar que se han generado métodos diferentes para la normalización de los pesos de las palabras respecto a los documentos (offline) y la normalización de los pesos de las palabras respecto a las consultas (online). Hay que destacar que tanto uno como otro tiene su método de cálculo de pesos sin normalizar y su posterior método de normalización.

Por otro lado, aparecen los métodos encargados de **guardar** y **cargar** las estructuras que almacenan los pesos normalizados tanto de las palabras respecto a los documentos como los respectivos a las consultas.

Desarrollo de la clase BUSCADOR

Primero de todo, dicha clase tiene un constructor parametrizado donde se pasa el configurador, la ruta-archivo donde se encuentran las consultas y un número que indica el máximo de documentos relevantes a mostrar por consulta. En él, se hace todo el procesamiento de tokenización, stopper y stemmer del fichero de consultas.

Por otro lado está el método ***procesar_pesos*** encargado de calcular primero los pesos normalizados de las palabras de las consultas y a posteriori calcular la similitud de la consulta con los documentos de la colección. Este último paso genera un diccionario donde se almacena, por consulta, el nombre del documento (con el que tiene una similitud > 0) y su similitud. Tras la creación de N archivos correspondientes a las N consultas donde aparecerá la anterior información, también se muestra por pantalla los X documentos relevantes (ordenados de mayor a menor relevancia) de las consultas.

Ejecución de la aplicación

En la ejecución de la aplicación iré indicando, guión a guión, toda la información que se ha ido pidiendo que se completara. En caso de querer comprobar que dichos resultados son verídicos puede ejecutar el programa y en el fichero que se crea llamado ***documentacion*** aparecerán dichos resultados.

Ejecución de la práctica 1.1 (TOKENIZACIÓN)

- El programa ha tardado 8.047642469406128 segundos en ejecutarse.
- Total de archivos procesados -> 1000.
- Total de tokens -> 276271 :: Tokens/archivo -> 276.271.
- Número MÍNIMO de palabras una vez normalizado y tokenizado -> 141
- Número MÁXIMO de palabras una vez normalizado y tokenizado -> 921
- Número MEDIO de palabras por documento -> 276.271
- Las 5 palabras más frecuentes -> [(('de', 23020), 1.0972354623450906), (('la', 11741), 0.559628217349857), (('en', 9332), 0.4448045757864633), (('y', 9133), 0.4353193517635844), (('el', 7499), 0.35743565300285984)]

Ejecución de la práctica 1.2 (STOPPER)

- Listado de Stopword seleccionado -> C:\CODIGO\SRI\PRACTICAS_SRI\SISTEMA_BASICO\spanishSmart.txt
- Número TOTAL de palabras una vez limpiada de palabras vacias -> 153060
- Número MÍNIMO de palabras una vez limpiada de palabras vacias -> 75
- Número MÁXIMO de palabras una vez limpiada de palabras vacias -> 455
- Número MEDIO de palabras por documento -> 153.06
- Las 5 palabras más frecuentes -> [(('pacientes', 2088), 0.10595757637267837), (('renal', 1034), 0.05247132852938191), (('estudio', 979), 0.049680300416116915), (('resultados', 965), 0.048969856896376736), (('años', 952), 0.048310159342332286)]

Ejecución de la práctica 1.3 (STEMMER)

- El STEMMER empleado es PorterStemmer de la librería nltk.stem
- Número TOTAL de palabras ÚNICAS de la colección -> 17375
- Número MÍNIMO de palabras ÚNICAS una vez extraidas sus raices de los documentos en la colección-> 55
- Número MÁXIMO de palabras ÚNICAS una vez extraidas sus raices de los documentos en la colección-> 262
- Número MEDIO de palabras ÚNICAS de los documentos en la colección -> 105.289
- Las 5 palabras más frecuentes -> [(('pacient', 2356), 0.13559712230215828), (('estudio', 1235), 0.07107913669064748), (('año', 1172), 0.06745323741007195), (('renal', 1150), 0.06618705035971223), (('resultado', 1008), 0.05801438848920863)]

Ejecución de la práctica 1.4 (PARES PALABRA-FRECUENCIA)

- Tiempo en segundos en calcular y generar la estructura de diccionario (la seleccionada) -> 4.495960712432861
- El espacio en disco para guardar la estructura es de 638638 bytes.
- Las caracteristicas de mi ordenador son:
  - Procesador: Inter(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz.
  - Memoria RAM: 7,82 GB utilizable.

Ejecución de la práctica 1.5 (PESOS NORMALIZADOS)

- Tiempo en segundos en calcular y generar la estructura de diccionario para los pesos normalizados (la seleccionada) -> 4.41736102104187
- Tiempo en segundos en calcular y generar la estructura de diccionario para los IDF (la seleccionada) -> 31.36957049369812
- El espacio en disco para guardar la estructura de pesos normalizados es de 1375760 bytes.
- El espacio en disco para guardar la estructura de los IDF es de 208322 bytes.
- Las caracteristicas de mi ordenador son:
  - Procesador: Inter(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz.
  - Memoria RAM: 7,82 GB utilizable.

Ejecución de la práctica 1.6 y 1.7

- **Consulta 1**: TIEMPO EN SEGUNDOS EN CALCULAR LA SIMILITUD PARA LOS DOCUMENTOS = 0.0

```
Consulta 1: La diabetes en personas mayores.

0.2777473036722275 S0212-97282014000100033.txt
0.1984815914048067 S0212-97282015000300027.txt
0.17861308355055605 S1699-695X2009000100004.txt
0.15791090303852673 S1699-695X2015000300004.txt
0.14337147772966935 S0212-97282014000100036.txt
0.13314516227146783 S1699-695X2012000100002.txt
0.13102907685214182 S0211-69952013000200011.txt
0.1253845194403435 S0212-97282016000300023.txt
0.12285188191406562 S0212-97282015000100003.txt
0.11073871760727218 S1139-76322017000100002.txt
```

- **Consulta 2**: TIEMPO EN SEGUNDOS EN CALCULAR LA SIMILITUD PARA LOS DOCUMENTOS = 0.0

```
Consulta 2: Los jóvenes universitarios.

0.22279398380140863 S1699-695X2014000100003.txt
0.22149909778236493 S0212-97282013000100023.txt
0.21521930903985428 S1139-76322011000200005.txt
0.17181408484012609 S0212-97282014000100033.txt
0.1656856260428348 S1699-695X2010000300002.txt
0.13575779633247395 S1699-695X2011000300003.txt
0.13156708568408573 S1699-695X2009000200005.txt
0.12736132282836815 S0212-97282015000100026.txt
0.12321098798317015 S0212-97282013000100009.txt
0.11378275062379532 S0212-97282013000300027.txt
```

- **Consulta 3**: TIEMPO EN SEGUNDOS EN CALCULAR LA SIMILITUD PARA LOS DOCUMENTOS = 0.0024962425231933594

```
Consulta 3: La insuficiencia renal.

0.36350184963501864 S0211-69952015000400007.txt
0.22648953740154418 S0211-69952010000100011.txt
0.1891449426385646 S0211-69952012000200011.txt
0.18895490177972613 S0211-69952012000500007.txt
0.15998097327305524 S0211-69952012000100009.txt
0.13526677807695525 S0211-69952013000500009.txt
0.13427819184411693 S0211-69952014000100014.txt
0.1333003041799779 S0211-69952009000600013.txt
0.12589250077856706 S0211-69952014000600004.txt
0.11719166703101014 S0211-69952016000700609.txt
```

- **Consulta 4**: TIEMPO EN SEGUNDOS EN CALCULAR LA SIMILITUD PARA LOS DOCUMENTOS = 0.0019941329956054688

```
Consulta 4: El asma un importante factor para predecir virus.

0.3562530255471308 S1139-76322010000500010.txt
0.25063638020202067 S1699-695X2013000300004.txt
0.23829540405273325 S1139-76322009000100007.txt
0.20348404700884046 S1139-76322013000200003.txt
0.16185157368775202 S1139-76322014000400013.txt
0.11694621943019194 S1139-76322009000200005.txt
0.10217204449634613 S0212-97282015000200009.txt
0.09088840260412642 S1139-76322009000200002.txt
0.08564632059645774 S1139-76322010000300005.txt
0.08453602816057922 S1139-76322009000400002.txt
```

- **Consulta 5**: TIEMPO EN SEGUNDOS EN CALCULAR LA SIMILITUD PARA LOS DOCUMENTOS = 0.002035379409790039

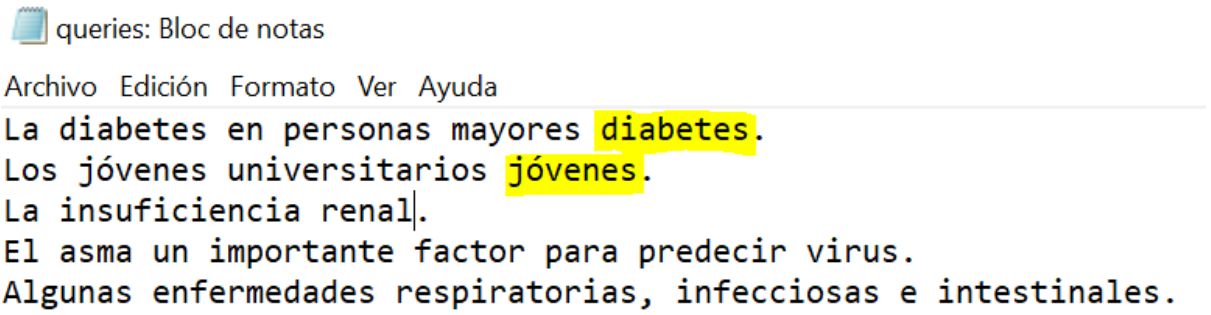
```
Consulta 5: Algunas enfermedades respiratorias, infecciosas e intestinales.

0.22101495951215466 S1699-695X2011000100005.txt
0.173249690269031 S1139-76322014000400014.txt
0.13959977557598274 S1699-695X2009000300004.txt
0.06259375490680466 S1139-76322009000700008.txt
0.060288859540058616 S1135-76062002000200007.txt
0.05213918942864572 S1139-76322012000400007.txt
0.051250533048023225 S1139-76322016000300005.txt
0.048793888520940035 S0211-69952015000100002.txt
0.04704399127170654 S1139-76322016000300012.txt
0.046721910306181134 S1139-76322014000500002.txt
```

Pruebas realizadas

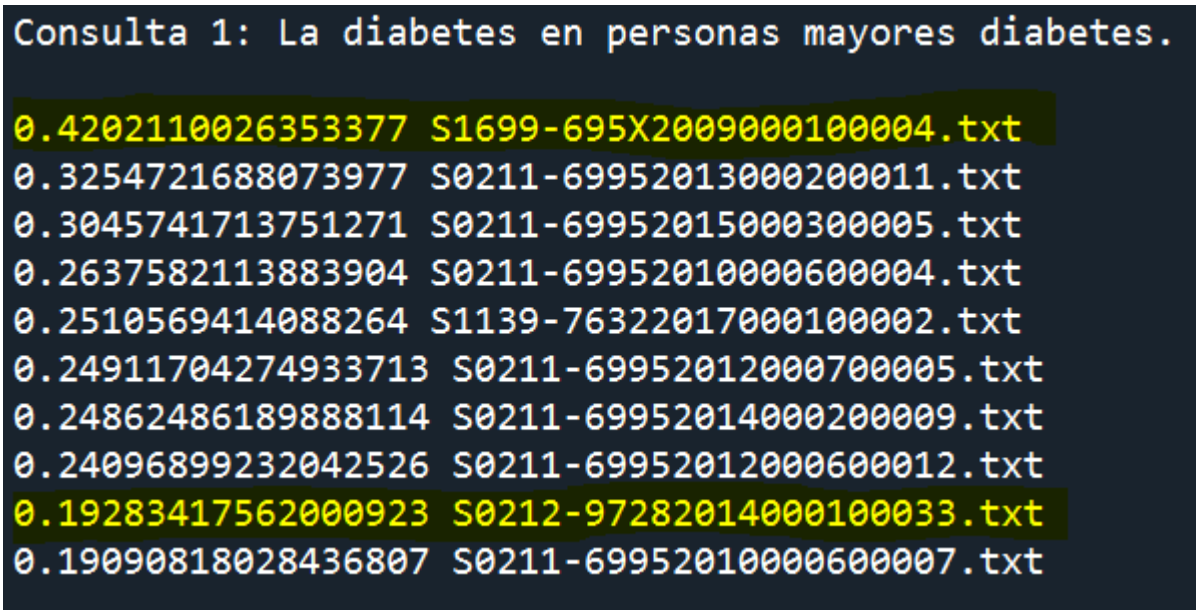
Tras la ejecución de la aplicación, y ver los resultados de relevancia de los documentos respecto a las consultas realizadas, he decidido comprobar el uso de la repetición de palabras en las consultas para ver su mejoría o no en los resultados.

Por ejemplo, se ha realizado una modificación en las consultas 1 y 2 añadiendo las siguientes palabras repetidas:



De esta forma se obtienen los siguientes resultados como documentos de relevancia y su similitud con la consulta:

Primero valoraremos la primera consulta con la repetición de una palabra, aparentemente, clave como es **diabetes**.



Para la primera consulta se observa un gran cambio aparente entre los documentos relevantes pero, ¿realmente sería más relevante en comparación al que era antes el más relevante sin que aparecieran palabras repetidas?

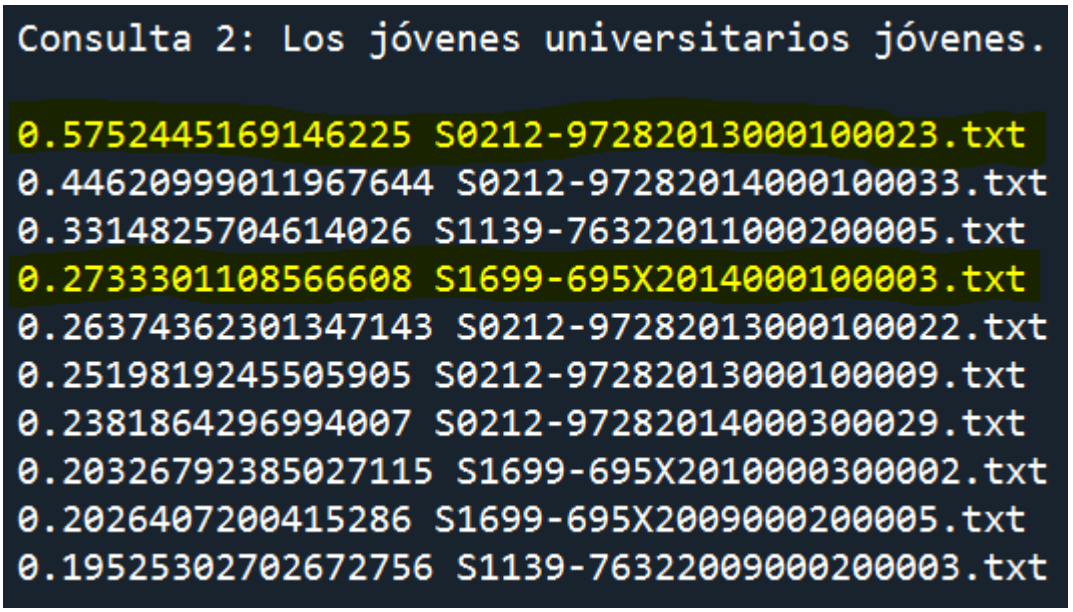
Para responder a la pregunta se va a hacer un estudio de ambos documentos sobre qué ha sido más importante para comprender el contexto de la consulta:

- Por un lado tenemos el archivo S1699-695X2009000100004 sí habla en relación de la diabetes y las personas mayores.
- Por otro lado tenemos el archivo S0212-97282014000100033, el cual habla del riesgo de que las personas mayores tengan menor memoria a la hora de declarar ante un acto de atraco.

Realmente, estas pruebas dan a pensar en que la retroalimentación de las consultas puede ser un factor determinante a la hora de la reordenación de documentos a mostrar al usuario tras realizar las consultas.

Hay que tener cuidado con esto, ya que el usuario no tiene porqué repetir la palabra que crea importante y quizás la repetición de dichas palabras provoque lo contrario y que pierda similitud con un documento el cual hable sobre el tema correspondiente de la consulta.

Ahora, se va a valorar el mismo proceso pero con la segunda consulta, repitiendo una palabra en la consulta, aparentemente, menos relevante en comparación a **universitario**.



Si hacemos el mismo estudio, se puede observar que una vez más el documento de mayor relevancia ha cambiado respecto al anterior (el cual ahora se sitúa en la 4 posición), pero ¿realmente sería más relevante en comparación al que era antes el más relevante sin que aparecieran palabras repetidas?

- Por un lado tenemos el archivo S0212-97282013000100023, el cual habla sobre las víctimas de maltrato infantil enfocadas en los jóvenes. Sin embargo, no habla acerca de los universitarios tras la repetición de la palabra **jóvenes**.
- Por otro lado tenemos el archivo S1699-695X2014000100003, el cual nos habla bastante sobre los universitarios acerca de los síntomas de ansiedad y depresión que pueden tener.

Como se puede observar, ha sucedido lo indicado. Es decir, tras repetir la palabra **jóvenes** se deja de lado el concepto de centrarse en los jóvenes universitarios.

Conclusión y valoración personal

Como he explicado anteriormente, una de las conclusiones a tener en cuenta tras realizar las pruebas del sistema de recuperación básico es añadir técnicas de retroalimentación. Esta técnica creo que debe de tener en cuenta un cierto criterio para evitar que la repetición de palabras sea perjudicial y conseguir que únicamente, si así es posible, sea beneficiosa para reordenar de forma que los documentos devueltos en las primeras posiciones mantengan un mayor contexto que los demás respecto a la consulta.

Por ejemplo, para explicar lo anterior, una idea podría ser que una vez se ha conseguido dar el peso a una palabra la cual ha aparecido ya tanto en la consulta como en el documento, esta vez se valore por la menor varianza de apariciones entre los términos encontrados en la consulta; es decir, si como en el caso analizado anteriormente la palabra **persona** y **mayor** aparecían con mucha frecuencia pero luego no aparecía **diabetes** por ningún lado, dicha varianza debe ser alta y por lo tanto no tener dicho documento tan en cuenta respecto a otros los cuales mantengan un equilibrio entre los 3 términos.

Como valoración personal, creo que el sistema de recuperación básico creado es eficiente en tiempo con una optimización de espacio de las estructuras creadas razonable. Por otro lado, creo que a nivel de eficacia es mejorable por casos como lo explicado anteriormente. Por lo tanto, hay parte que mejorar en términos de eficacia para la mejora de dicho sistema de recuperación básico.