Preentrega Proyecto Final de SRI:
Modelo Vectorial

Lauren Guerra Hernández
Paula Rodríguez Pérez
Dennis Fiallo Muñoz

1 Modelo Vectorial

La implementación de este Sistema consta de las siguientes etapas de recuperación de información:

Etapa 1: Verificar si ya se tienen almacenada la información recuperada del set de datos actual, en caso de que no se hayan recuperado los datos de estos documentos con anterioridad se pasa a la Etapa 2 para extraer los datos, sin embargo, si ya se tienen los datos recuperados se pasa a la Etapa 5 para lograr mayor velocidad al mostrar los resultados ya que se cuenta con la información necesaria de los documentos.

Etapa 2: Tomar documentos del set de datos.

Etapa 3: Separar los términos de cada documento y eliminar los signos de puntuación y las stopwords.

Etapa 4: Calcular y almacenar en un diccionario la relación entre término con los documentos en que aparece y de estos su tf, idf, w y frecuencia como se muestra a continuación:
`{término: {documento: {frecuencia: x, tf: x, idf: x, w:x}, ...}, ...}`

Etapa 5: Calcular y almacenar los pesos de las consultas en los documentos del siguiente modo:
`{término: w, ...}`

Etapa 6: Calcular la similitud de la consulta con los documentos usando los datos recopilados anteriormente.

Etapa 7: Devolver ranking de documentos en dependencia de las restricciones que se plantean con anterioridad.

La implementación del código de este sistema va a estar dividida en las siguientes partes:

dataset.py: Contiene la clase `Datasets` la cual se encarga de extraer la información de los sets de datos usando el módulo de python `ir_datasets` del cual se va a usar para esta preentrega el set de datos "cranfield". Esta clase cuenta con el método `get_dataset` el cual extrae y almacena la información referente al set de datos con el nombre que se le pasa al método, además de que posee la función `get_docs_data` la cual devuelve un iterador `de {"id": id, "text": text}`.

stopwords.py: En este aparece la clase `Stopwords` la cual al ser instanciada se le pasa como parámetro el idioma en que se desean estas stopwords y luego se almacenan en un diccionario para lograr un acceso a estas en O(1) al cual se puede acceder con `<Stopwords instance>.stopwords`. Posee la función `not_stopwords_terms` la cual dada una lista de términos retorna un lista con los que no son stopwords.

utils.py: Contiene funciones que podrían ser útiles para la implementación de los modelos.

visual.py y visual.UI: Interfaz visual generada por el módulo `pyQt5`.

visual_actions.py: Se definen las acciones que deben realizar los componentes de la UI al interactuar el usuario con estos. Como método principal tiene click_search el cual recopila todos los datos de entrada del usuario y ejecuta el modelo deseado.

vector_model.py: Contiene la clase `VectorModel` la cual ejecuta todos los procesos del modelo vectorial, teniendo como método principal `run`, siendo este el que ejecuta los procesos descritos anteriormente en las etapas de recuperación de información.

Para correr el código se puede ejecutar el archivo visual.py, o se puede usar directamente el archivo vector_model.py creando una instancia del modelo y ejecutando la función `run` pasándole los parámetros necesarios(esto se encuentra ya planteado en main.py). Además, en “docs” aparece el archivo “cran.qry” el cual contiene casos de prueba de consultas para el set de datos “cranfield”.

Código fuente: https://github.com/dionisio35/sri.git