README

Para poder ejecutar correctamente nuestro código, considerando que es un fichero python, tenemos que abrirlo en un IDE que soporte ese tipo de lenguaje, preferiblemente Anaconda o Google Colab.

Primeramente, para empezar ejecutar nuestro código se debe ejecutar la serie de "imports" que podemos encontrar en el inicio de nuestro fichero python.

A continuación abrimos el fichero 'tw_hurricane_data.json' en 'lines'; que recoge información sobre diferentes tweets, para ello tenemos que poner el path donde guardamos dicho file en nuestro ordenador.
Hemos imprimido el doc 12 (lines[12]), para comprobar la información de dicho tweet. Convertimos el archivo 'lines' que es json, en un diccionario guardado en 'datos_diccionarios'; hemos hecho en print para comprobar cual es tweet id del doc 12 según: datos_diccionario[12]['id'].

Después de guardar el fichero en 'datos_diccionario', tenemos que ejecutar las funciones: build_terms(), create_index(), rank_documents() y search_tf_idf(), por orden de cómo aparecen en el fichero python.
Después nos encontramos con una línea de código que llama la función create_index() que se tiene que ejecutar a continuación de las funciones anteriores.
Seguidamente, encontramos las cinco queries, que se tienen que ejecutar por el orden en que aparecen.

A continuación abrimos el fichero 'evaluation.csv' en 'search_result'; que recoge información sobre los ids de 3 queries, los ids de los docs relacionados con dichas queries y el label que interpretamos que es el doc score; para ello tenemos que poner el path donde guardamos dicho file en nuestro ordenador.
Hacemos un print para saber cuántos valores tiene la columna label y según está crear una nueva columna de la relevancia binaria de cada doc.
A continuación, hacemos cambios en la tabla 'search_results' descritos en el report.

Seguidamente, creamos una lista 'queries' que contiene las 3 queries del pdf, creamos un dataframe 'work_data' con 3 columnas que guardará la información con la que trabajaremos query_id, tweet_id y predicted relevance. Buscamos los docs más relevantes para cada una de las queries con ayuda de search_tf_idf y rellenamos la tabla 'work_data' con dicha info. Además añadimos otra columna a nuestro dataframe que indique la relevancia binaria de cada doc de forma aleatoria.

Llamamos a la función precision_at_k de la query 1, teniendo en cuenta la relevancia de los docs devueltos al buscar dicha query y la relevancia predecida de los docs al buscar dicha query por nuestro sistema de búsqueda (ambos datos se guardan en 'work_plan'). Hemos incluido un print de 'work_data' para la query 1, que solo refleje los docs relevantes.

Después de la llamada de función de precision, ejecutamos la función recall(), y el siguiente bloque de código, dónde se calculará el recall por k=5.
Después tenemos que ejecutar la función f1_score(), y la casilla de código que encontramos a continuación para calcular el valor del f1_score.

Seguidamente, ejecutamos la función avg_precision_at_k(), y la siguiente línea de código para calcular la avg_precision_at_k por k=10.

Después tenemos la función map_at_k implementada y en la siguiente casilla de código llamamos a esa función pasándole por parámetro work_data y k = 10.

Después podemos ver que está la función rr_at_k y la llamamos justo debajo con los parámetros labels, scores y k = 10. Labels corresponde a una array con los valores de "is_relevant" del work data cuando la query es la número 1. Scores corresponde a una array con los valores "predicted_relevance" del work data cuando la query es la número 1.

Ya para ir terminando, ejecutamos la casilla de código con las funciones dcg_at_k y también ndcg_at_k. Después de esto, encontramos otra casilla de código, que al ejecutarla nos aparecerá el resultado del ndcg de nuestra query con su respectivo valor. Después de esta casilla, vamos a ejecutar la siguiente casilla de código, y en esta casilla, hacemos un for para calcular el average del ndcg de nuestras queries, con el uso de un for. Podemos ver su respectivo resultado.

Finalmente, para generar el plot usamos las librerías Word2Vec, TSNE y plt. A la función Word2Vec le pasamos por parámetro los tweets limpios, workers = 4, min_count = 50, window = 10, sample = 1e-3. A TSNE el número de componentes que es 2 y al scatter plot tsne[:,0] y tsne[:,1].