Enunciado práctica InformationRetrieval

El objetivo de esta práctica es implementar un sistema de recuperación de información (*information retrieval* o IR). Se trata de construir, en primer lugar, un índice invertido (*inverted index*) para recuperar rápidamente documentos que coincidan con las preguntas de los usuarios. Luego, se pretende usar *term-frequency inverse-document-frequency weighting* y la similitud de coseno (*cosine similarity*) para encontrar los documentos más relevantes de la colección (para unas ciertas consultas). Se evaluará la precisión del sistema implementado para diferentes consultas y si el cálculo de los valores del tf-idf y de las similitudes de coseno es exacto.

El sistema de IR pretende encontrar los documentos relevantes dentro de una colección de 60 historias cortas de Rider Haggard. Los datos de entrenamiento se encuentran en la carpeta `datos/RiderHaggard/raw`. Las consultas están en `queries.txt` y las soluciones en `solutions.txt`.

Para mejorar el sistem de IR actual, se deben implementar:

- **Inverted Index**: un diccionario que asigna a cada palabra la lista de documentos en los que aparece.
- **Boolean Retrieval**: un sistema de recuperación booleano, en el que se debe retornar una lista con todos los documentos que contienen las palabras de una consulta; el método se evaluará solo si se usa el algoritmo para la intersección de listas visto en la clase de teoría.
- **TF-IDF**: calcular y almacenar el valor tf-idf para todos los pares de término/documento con frecuencias no nulas:

$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \times \log_{10} (N/dft)$$

- **Cosine Similarity**: implementar la similitud de coseno para mejorar el sistema de recuperación existente, que actualmente recupera documentos basados en el coeficiente Jaccard entre la consulta y cada documento. Tenga en cuenta que al calcular $w_{t,q}$ (es decir, el peso de la palabra w en la consulta) no debe incluir el término idf. Eso es,

$$w_{t,q} = 1 + \log_{10} tf_{t,q}$$

  La solución de referencia utiliza el esquema **ltc.lnn** para calcular los cosenos.

En concreto, se deben implementar y/o mejorar los siguientes métodos:

- `index(self)`: aquí es donde se construirá el índice invertido. Los documentos ya habrán sido leídos en self.docs (una lista de listas) y el vocabulario de palabras en self.vocab (una lista de palabras)
- `get_posting(self, word)`: este método devuelve una lista de enteros (ID de documento) que identifica los documentos en los que se encuentra la palabra.
- `boolean_retrieve(self, query)`: este método realiza una recuperación booleana, devolviendo una lista de IDs correspondientes a los documentos en los que aparecen todas las palabras de la consulta.
- `compute_tfidf(self)`: este método calcula y almacena los valores tf-idf para palabras y documentos.
- `get_tfidf(self, word, document)`: este método retorna el valor tf-idf para una palabra particular y una ID de documento.
- `rank_retrieve(self, query)`: este método se usa para crear una cola de prioridad con los documentos mejor clasificados para una consulta determinada. Actualmente ordena los documentos según su similitud Jaccard con respecto a la consulta. Se pide utilizar la similitud coseno entre los documentos y la consulta para determinar el orden.

Es mejor trabajar los métodos en este orden porque algunos se basan en los anteriores, y también se vuelven un poco más complejos a medida que se avanza en esta lista.

La primera ejecución del programa creará una carpeta `stemmed` en `./data/RiderHaggard`. Ejecuciones sucesivas serían mucho más rápidas. Sin embargo, si algo sucede durante esta primera ejecución y no se procesan todos los documentos, es posible que se quede con un conjunto incompleto de documentos en `./data/RiderHaggard/stemmed/`. Si este es el caso, simplemente elimine la carpeta `stemmed` y vuelva a ejecutar.