Cuarta práctica de laboratorio de Sistemas de Información para la Web

Motivación

En la práctica anterior nos enfrentamos a dos problemas fundamentales en un buscador: (1) cómo representar un texto en base a sus términos constituyentes, y (2) cómo comparar un documento dado con otro texto (p.ej. una consulta) para determinar su similitud y, por extensión, su relevancia.

También vimos que una comparación de una consulta con todos los documentos de una colección no es viable pero antes de ver cómo realizar consultas de forma eficiente debemos resolver otro problema muy habitual en la web: la existencia de documentos duplicados o casi-duplicados.

Imaginemos que al explorar la web acabasen en la colección los documentos A, B y C junto con copias (o cuasi-copias) de los mismos A_2, A_3, A_4, B_2, B_3 y C_2, C_3, C_4. Imaginemos, además, que dada una consulta q los documentos más relevantes son A, B y C en ese orden. ¿Qué aspecto tendría la lista de resultados obtenidos? El siguiente:

$$A, A_2, A_3, A_4, B, B_2, B_3, C, C_2, C_3, C_4, \ldots$$

La lista obtenida contendría información duplicada que es similar a la consulta y, en consecuencia, relevante pero que ha dejado de ser relevante por no ser ya novedosa. En consecuencia, para ser verdaderamente útil para el usuario los documentos similares tendrían que eliminarse.

A tal fin podrían usarse cualquiera de las medidas de similitud ya vistas pero de nuevo nos encontraríamos con un problema de eficiencia. Simplemente no podemos comparar todos los documentos entre sí.

Para resolver ese problema existen las técnicas denominadas *SimHashing*.

Descripción del ejercicio y del entregable

La primera parte de este ejercicio consiste en la lectura atenta y análisis del artículo *"Simple simhashing: Clustering in linear time"* de Ryan Moulton. En dicho artículo, publicado originalmente en Knol, se explica cómo se pueden usar funciones hash para obtener "firmas" de documentos que permitan determinar si son o no similares. En el artículo aparece un parámetro muy importante (`restrictiveness`) que indica cuántos elementos van a formar parte de dicha firma; es parte del objetivo de este ejercicio realizar pruebas con distintos

valores para dicho parámetro y documentar el proceso de decisión para fijar un valor por defecto.

A tal fin se debe implementar en Python la función `SimHash` que recibirá un documento y un valor para `restrictiveness` y retornará su firma.

Nótese que el documento no será una cadena de texto sino que debería ser un vector de términos. Para obtenerlo debería reutilizarse todo el código necesario de la práctica anterior (es decir, pueden usarse tokenizadores, estematizadores, listas de palabras vacías, etc.)

En principio se considerará que los términos corresponderán con "palabras" pero se valorará muy positivamente el uso de n-gramas de términos (véase el apartado *"Turning anything into a set"*). En caso de implementar la versión que usa n-gramas, también es necesario realizar las pruebas pertinentes para fijar un valor por defecto y documentar este proceso.

Para las pruebas del ejercicio puede utilizarse la adaptación de la colección Cranfield del ejercicio anterior o, mejor aún, se puede utilizar el archivo `2008-Feb-02-04.json.gz` disponible en http://goo.gl/aAtVmm. Dicho dataset incluye todos los tuits (en todos los idiomas) publicados entre el 2 y el 4 de febrero de 2008. Puesto que los retuits son elementos cuasi-duplicados dicho dataset contiene multitud de tales ítems.

Con independencia de la colección que se emplee se deberá usar el código desarrollado para encontrar al menos 5 elementos cuasi-duplicados. En consecuencia, además del código desarrollado, se deberá incluir un archivo de texto en el que se incluirán los ejemplos de documentos cuasi-duplicados encontrados de manera automática.

Referencias bibliográficas

- Moulton, Ryan. *"Simple simhashing: Clustering in linear time"*.
- Vassilvitskii, Sergei. *"Duplicate Detection On the Web: Finding Similar Looking Needles In Very Large Haystacks"*
- MinHash Tutorial with Python Code