Tercera práctica de laboratorio de Sistemas de Información para la Web

Motivación

Un buscador web tiene como misión encontrar en la Web documentos que sean relevantes para una necesidad de información expresada en forma de consulta. Así, si nuestra necesidad de información fuera *"¿Qué es un buscador web?"* podríamos introducir la consulta `buscador+web` en un buscador y quizás obtendríamos como primer resultado la URL `https://es.wikipedia.org/wiki/Motor_de_b%C3%BAsqueda`. Dicha página, contiene en su inicio el texto siguiente:

> Un motor de búsqueda o **buscador** es un sistema informático que busca archivos almacenados en servidores **web** gracias a su spider (también llamado araña web). Un ejemplo son los buscadores de Internet (algunos buscan únicamente en la web, pero otros lo hacen además en noticias, servicios como Gopher, FTP, etc.) cuando se pide información sobre algún tema. Las búsquedas se hacen con palabras clave o con árboles jerárquicos por temas; el resultado de la búsqueda «Página de resultados del **buscador**» es un listado de direcciones **web** en los que se mencionan temas relacionados con las palabras clave buscadas.

En este caso concreto el resultado obtenido parece bastante relevante para la necesidad de información original; sin embargo, el buscador es un sistema totalmente automático, sin ningún tipo de "inteligencia" para determinar qué objetivo se perseguía con la consulta, ni para evaluar la relevancia real del contenido encontrado.

Lo que sucede en realidad es que el buscador no trata de encontrar documentos relevantes sino documentos **similares** a la consulta de entrada. Similitud que, en los métodos más básicos es puramente léxica.

Así, en negrita se muestran términos que coinciden literalmente con alguno de los términos de la consulta; mientras que se han subrayado aquellos términos que, dependiendo de la implementación del buscador, podrían tener o no coincidencia con los términos de búsqueda.

Es preciso señalar que los buscadores modernos utilizan muchísimos más indicios que la simple aparición de palabras clave pero, aún así, resulta importante comprender cómo funcionan las medidas de similitud más comunes.

Similitud booleana entre documentos o documentos y consultas

Documentos y consultas se representan mediante *bags-of-words*, es decir, vectores donde se tiene en cuenta únicamente los términos que aparecen pero no se considera la frecuencia de aparición. Para dicho modelo existen distintos coeficientes para determinar qué documento es más parecido a una consulta y, en consecuencia, presumiblemente relevante.

Coeficiente de Dice	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$
Coeficiente de Jaccard	$\frac{\lvert X \cap Y \rvert}{\lvert X \cup Y \rvert}$
Coseno	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert \cdot \lvert Y \rvert}$
Coeficiente de solapamiento	$\frac{\lvert X \cap Y \rvert}{\min(\lvert X \rvert, \lvert Y \rvert)}$

En las fórmulas anteriores se entiende que X e Y son vectores que representan documentos (o consultas), el módulo indica el número de términos en un documento, unión de documentos o intersección de documentos.

¿Qué es un término?

A priori puede parecer sencillo determinar de manera automática qué términos aparecen en un documento pero en realidad no es una tarea fácil. Así, en el texto anterior aparecen las siguientes secuencias de caracteres que podrían considerarse (o no) términos diferentes:

```
búsqueda
buscador
busca
buscadores
buscan
búsquedas
búsqueda
buscador»
buscadas.
```

Obsérvese cómo al usar el espacio como separador de términos hay palabras que incorporan símbolos de puntuación y que serían, por tanto, distintas de las mismas palabras sin esos signos. Por otro lado, `buscador` y `buscadores` o `búsqueda` y `búsquedas` se consideran términos distintos según sean plural o singular aunque tengan el mismo "lema". Por último, si en lugar de usar los términos literalmente se redujesen a su raíz (usando un *stemmer*) habría una serie de ellos que pasarían a ser equivalentes entre sí.

A efectos de este ejercicio puede optarse por un enfoque simplista usando cualquier espacio en blanco (espacio, tabulador, salto de línea y retorno de carro) como separador y pasando los términos resultantes a minúsculas. La eliminación de símbolos de puntuación se considerará un bonus en la evaluación.

Quien lo desee puede explorar el uso de bibliotecas para la tokenización del texto y la estematización de los términos resultantes[1]. Para ello se recomienda encarecidamente el uso del toolkit NLTK. El uso de estas funcionalidades será considerado muy positivamente en la evaluación del ejercicio.

Descripción del ejercicio y del entregable

Colección de documentos (y consultas)

Para la realización del ejercicio se trabajará sobre un derivado de la colección Cranfield disponible en el siguiente archivo. Al descomprimirlo se obtienen dos ficheros de texto: `cran-1400.txt` que contiene una serie de documentos (uno por línea) y `cran-queries.txt` que contiene una serie de consultas (también una por línea). A continuación se muestra un ejemplo de cada:

- `I1 experimental investigation of the aerodynamics of a wing in a slipstream brenckman,m j ae scs 25, 1958, 324 experimental investigation of the aerodynamics of a wing in a slipstream an experimental study of a wing in a propeller slipstream was made in order to determine the spanwise distribution of the lift increase due to slipstream at different angles of attack of the wing and at different free stream to slipstream velocity ratios the results were intended in part as an evaluation basis for different theoretical treatments of this problem the comparative span loading curves, together with supporting evidence, showed that a substantial part of the lift increment produced by the slipstream was due to a /destalling/ or boundary-layer-control effect the integrated remaining lift increment, after subtracting this destalling lift, was found to agree well with a potential flow theory an empirical evaluation`

[1] Téngase en cuenta que la reducción a la raíz de un término requiere determinar en qué idioma está escrito un texto dado. En este ejercicio supondremos que los textos están escritos en inglés. Véase, p.ej. la biblioteca *langdetect* para aquellos casos en los que puedan coexistir distintos idiomas.

```
of the destalling effects was made for the specific configuration of the experiment
```

- `I 001 what similarity laws must be obeyed when constructing aeroelastic models of heated high speed aircraft`

Objetivo del ejercicio

Se deben implementar en Python todas las medidas de similitud entre vectores de documentos que aparecen en la motivación de este enunciado. Cada medida será una función que recibirá dos argumentos (los documentos a comparar[2]) y retornará un valor real positivo que será su similitud.

Deberán implementarse, además, funciones que permitan cargar tanto los documentos de la colección como las consultas en sendos diccionarios.

En un script se debe demostrar el uso de dichas funciones obteniendo el documento más similar (para cada coeficiente) a una consulta cualquiera[3].

Queda a la elección del estudiante determinar cómo se obtienen los términos de documentos y consultas, si se usan o no tokenizadores, estematizadores o lematizadores, o si se ignoran las palabras vacías[4].

Entregable

Se subirá al campus virtual un archivo comprimido que contendrá el código fuente correctamente documentado y deberá pasar las pruebas unitarias que se pueden descargar en el campus virtual. En caso de que se considere necesario realizar alguna prueba unitaria adicional, también habrá que añadirla al archivo comprimido. Además se incluirá un breve documento donde se describirán las distintas decisiones tomadas por el estudiante a la hora de implementar el ejercicio.

---

[2] Idealmente ya deberían ser vectores de términos y no simples cadenas de texto.
[3] Nótese que no resulta práctico comparar una consulta dada con todos los documentos de una colección. En una práctica posterior se verá cómo realizar las búsquedas de forma eficiente.
[4] Una posible lista para inglés: http://snowball.tartarus.org/algorithms/english/stop.txt

Pseudocódigo

Este es el pseudocódigo de la función principal

```
def main():
  texts = load_lines(“cran-1400.txt”)
  queries = load_lines(“cran-queries.txt”)
  for q in queries:
    best_text = find_best_text(q, texts, coef_dice)
    # show text

    best_text = find_best_text(q, texts, coef_jaccard)
    # show text

    best_text = find_best_text(q, texts, coef_cosine)
    # show text

    best_text = find_best_text(q, texts, coef_overlapping)
    # show text
```

Para convertir un string en un *bags-of-words* se puede usar el siguiente algoritmo. Dada la complejidad real de la tarea se puede alguna librería de Procesamiento de Lenguaje Natural (o NLP), como NLTK para realizar algunos de los pasos[5].

```
def string_to_bag_of_words(text):
  bag = {}

  # Obtenemos las palabras a mano o mediante NLTK
  words = text.split() # Versión sencilla.

  for word in words:
    # Paso 1: Pasar a minúsculas
    # Paso 2 (opcional): Eliminar símbolos de puntuación
    # Paso 3 (opcional): Eliminar palabras vacías (o stop-words) a
    #   mano o mediante NLTK
    # Paso 4 (opcional): Lematizar mediante NLTK
    # Paso final: Almacenar la palabra después de cada transformación
    #   con el número de apariciones de la misma
    bag[word] += 1

  return bag
```

[5] NLTK book - Language Processing and Python

Es preciso tener en cuenta que, aunque aquí se muestra una función, el código fuente entregado tiene que cumplir la interfaz propuesta en las pruebas unitarias, por lo que tendrá que implementarse en la clase `BagOfWords` con todos los métodos necesarios. Para más información sobre los métodos especiales de Python (los que empiezan y terminan con __) consultar la la página de Python data model.

```
class BagOfWords(object):
  def __init__ (self, text=None, values=None):
    """Constructor

    Si recibe un string mediante el argumento text lo convierte a un
diccionario. Si recibe un diccionario simplemente lo copia para su
uso interno.
    """
    pass

  def __str__(self):
    """Devuelve un string con la representación del objeto

    El objeto BagOfWords("A b a") está representado por el string
    "{'a': 2, 'b': 1}"
    """
    pass

  def __len__(self):
    """Devuelve el tamaño del del diccionario"""
    pass

  def __iter__(self):
    """Crea un iterador que devuelve la clave y el valor de cada
    elemento del diccionario

    El diccionario {'a': 1, 'b': 2} devuelve:
    - ('a', 1) en la primera llamada
    - ('b', 2) en la primera llamada
    """
    pass

  def intersection(self, other):
    """Intersecta 2 bag-of-words

    La intersección de "a b c a" con "a b d" es:
    {'a': 1, 'b': 1}
    """
```

```
    pass

  def union(self, other):
    """Une 2 bag-of-words

    La unión de "a b c a" con "a b d" es:
    {'a': 3, 'b': 2, 'c': 1, 'd': 1}
    """
    pass
```