Práctica obligatoria evaluable
People Name Disambiguation

Objetivo

El objetivo es poner en práctica diferentes conceptos aprendidos durante el curso y relacionados con la Recuperación de Información y el procesamiento de texto con la librería NLTK de Python, aplicándolos a una tarea real concreta.

Normativa de entrega

La entrega de esta práctica es *obligatoria* y puede realizarse ***individualmente o por parejas***. A continuación, se detallan otros datos de interés relacionados con la normativa de entrega:

- La fecha límite de entrega de la práctica será el día **30 *de abril de 2017*.**
- La entrega de la práctica se hará a través de Aula Virtual, empleando el **formulario** habilitado para ello (se habilitará más adelante, se pondrá una noticia en el foro de novedades cuando esté disponible).
- Se deberá subir un único archivo comprimido con todo el código fuente que se haya implementado y se necesite para ejecutar la práctica, así como una memoria explicando qué se ha hecho y por qué.

Enunciado

Resolver la ambigüedad de los nombres de personas en los resultados de una búsqueda web es un problema que ha tomado cada vez más interés dentro de la comunidad de Procesamiento del Lenguaje Natural y Recuperación de Información.

Cuando se hace una búsqueda donde la consulta coincide con un nombre de persona, en los resultados que devuelve el buscador se pueden encontrar páginas de individuos diferentes que comparten el mismo nombre. Un ejemplo se puede ver en la Figura 1.

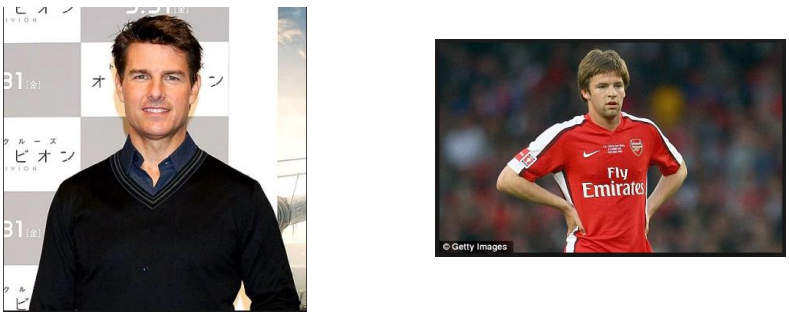

Figura 1. Individuos diferentes que comparten el nombre *Thomas Cruise*.

La tarea de desambiguación de nombres de personas en la web consiste en que dada una consulta que es un nombre de persona, junto con los resultados obtenidos por un motor de búsqueda para la consulta, el objetivo es agrupar las páginas web de acuerdo a las diferentes personas a las que se refiere. El reto es, por tanto, estimar el número de personas diferentes que comparten el nombre de la consulta y agrupar en el mismo grupo o cluster aquellas páginas con información de la misma persona.

Los pasos mínimos a dar serían los siguientes:

1. Búsqueda de información en un motor de búsqueda web a partir de una consulta que es un nombre de persona (normalmente nombre y primer apellido).
2. A partir del ranking devuelto por el buscador con las páginas más relevantes de acuerdo a la consulta, coger los *n* primeros resultados, los cuáles se utilizarán para encontrar los diferentes individuos que comparten el mismo nombre de persona.
3. Descargar el contenido (crawling) de los *n* primeros resultados del ranking.
4. Procesar los documentos descargados y convertirlos en texto. Habitualmente serán páginas web en formato HTML, por lo que mediante alguna librería se procesarán y se convertirán en texto.
5. Procesar el texto para generar una representación común de cada documento, por ejemplo, una representación vectorial donde cada vector representa un documento. Cada componente del vector representa un rasgo del vocabulario del problema (todos aquellos rasgos únicos de todos los documentos que intervienen en el problema) y tiene un peso concreto, que mide la importancia de ese rasgo en el documento.
6. A partir de una matriz de vectores de documentos se utilizaría un algoritmo de clustering que agruparía los documentos en grupos de acuerdo a la similitud de sus contenidos.
   Para el problema de la desambiguación de nombres de personas, cada grupo representa un individuo concreto.

El esquema de trabajo anterior sería el lógico a seguir, sin embargo, en esta práctica hay pasos del esquema que no se deben llevar a cabo porque ya se dan hechos:

- Como colección de trabajo se dispone de varios documentos obtenidos del ranking de páginas web relevantes devueltas por un motor de búsqueda web para la consulta “Thomas Baker”. Se trata de una colección pequeña, en total 19 documentos, los cuáles han sido escogidos de un ranking donde previamente ya se habían seleccionado las primeras 100 páginas del ranking devuelto por el buscador. Además, no hay que procesar las páginas en formato HTML (o en cualquier otro formato de origen), porque ese procesamiento ya se ha realizado con Tika (https://tika.apache.org/). Por lo tanto, el formato de los documentos presentes en la colección es txt.

- Se proporciona un código fuente en Python (con la versión de Python 3.5) con un ejemplo completo de procesamiento de las páginas web en formato txt y

agrupamiento mediante un algoritmo de clustering aglomerativo. En este caso se ha utilizado la librería de Machine Learning en Python *Scikit-Learn* (http://scikit-learn.org/stable/). Esta librería presenta diferentes algoritmos para análisis de datos y procesamiento de textos, y es muy común utilizarla junto con NLTK.

En la Figura 2 se presenta una de las funciones principales del código fuente que se proporciona.

```
def cluster_texts(texts, clustersNumber, distance):
    #Load the list of texts into a TextCollection object.
    collection = nltk.TextCollection(texts)
    print("Created a collection of", len(collection), "terms.")

    #get a list of unique terms
    unique_terms = list(set(collection))
    print("Unique terms found: ", len(unique_terms))

    ### And here we actually call the function and create our array of vectors.
    vectors = [numpy.array(TF(f,unique_terms, collection)) for f in texts]
    print("Vectors created.")

    # initialize the clusterer
    #clusterer = GAAClusterer(clustersNumber)
    #clusters = clusterer.cluster(vectors, True)
    clusterer = AgglomerativeClustering(n_clusters=clustersNumber,
                                        linkage="average", affinity=distanceFunction)
    clusters = clusterer.fit_predict(vectors)

    return clusters
```

Figura 2. Agrupamiento de textos con un algoritmo de clustering aglomerativo.

Como se puede ver en la figura, de forma general el código lo que hace es un procesamiento muy básico del texto:

1) Mira cuáles son el conjunto de términos que aparecen en los textos de la colección, para así obtener el vocabulario con el que representar los documentos.
2) Representa cada documento mediante un vector donde cada término del vocabulario se pesa con la medida TF (frecuencia del término en el documento).
3) Agrupa con el algoritmo de clustering, al que le da como parámetros de entrada el número de grupos en los que tiene que agrupar los documentos (en la colección de trabajo el nombre de Thomas Baker es compartido por 4 individuos diferentes), el tipo de enlace (esto es particular del algoritmo de clustering utilizado) y la medida de similitud con la que comparar los vectores de los documentos.

En la Figura 3 se presenta la salida del código proporcionado. Como se puede ver la salida se corresponde con una lista de números (etiquetas), donde cada número representa uno de los clusters generados. Así, cada posición de la lista se corresponde con uno de los documentos de entrada (respetando el mismo orden en el que estaban en la lista de textos de entrada), de manera que aquellas posiciones de la lista con igual número se refieren a documentos que se han agrupado juntos.

```
C:\Anaconda\envs\Python3Env\python.exe C:/Users/Soto/MI_TRABAJO/DOCENCIA
Prepared  19  documents...
They can be accessed using texts[0] - texts[18]
Created a collection of 22301 terms.
Unique terms found:  4874
Vectors created.
test:  [2 2 2 0 2 0 2 2 1 1 2 2 1 2 2 2 1 3 2]
reference:  [1, 2, 3, 1, 1, 1, 4, 1, 1, 1, 3, 1, 4, 4, 1, 2, 3, 1, 2]
rand_score:  -0.1496421600520495


Process finished with exit code 0
```

Figura 3. Resultado de la ejecución del código de partida.

¿Qué se pide con la realización de esta práctica?

Si se analiza el código proporcionado detenidamente se podrá comprobar que el procesamiento que se ha hecho de los textos de las páginas web es muy simple. Por ejemplo, no se están eliminando palabras vacías, no se está haciendo stemming ni lematización, no se reconocen Entidades Nombradas, etc. Esto no es lo habitual en este tipo de problemas. Por lo tanto, lo que debe hacer el alumno es una labor de investigación, de forma que vaya modificando el código, añadiendo nuevo procesamiento con el objetivo de generar la representación de los documentos más adecuada para el problema que se quiere resolver.

Con la métrica empleada para evaluar el resultado de clustering (*Adjusted Rand Index, ARI*[1]*)* se podrá ver si cada modificación que se haga mejora o empeora los resultados obtenidos.

Como se puede ver en las Figuras 4, 5 y 6 las páginas web de la colección son muy variadas, lo que puede dificultar el decidir qué es lo que mejor representa el contenido de las mismas.

[1] https://en.wikipedia.org/wiki/Rand_index

Figura 4. Parte de la página HTML original del fichero 020.txt.

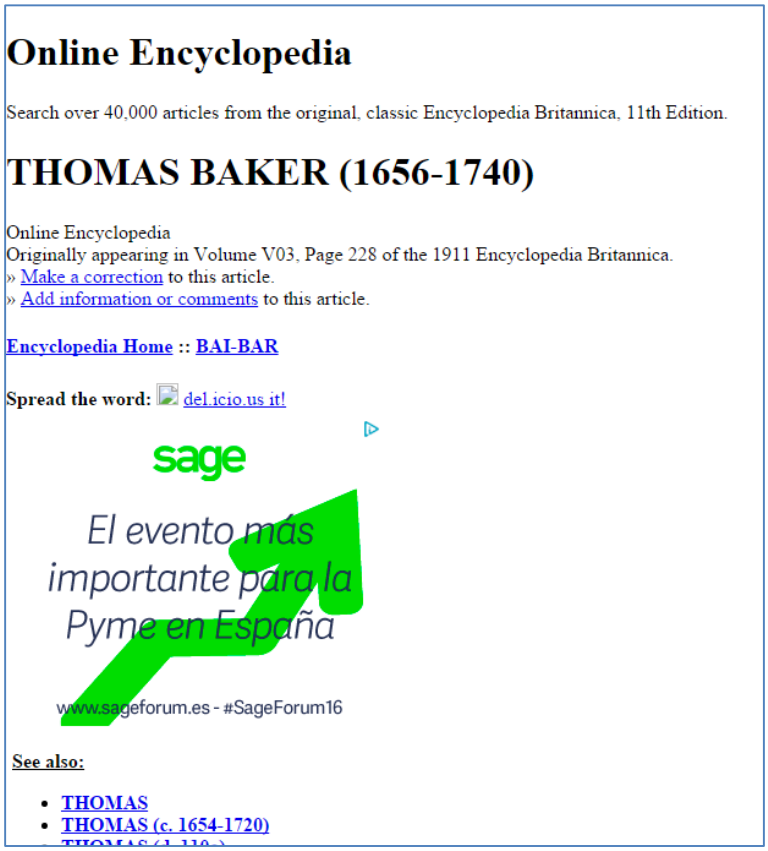

Figura 5. Parte de la página HTML original del fichero 075.txt.

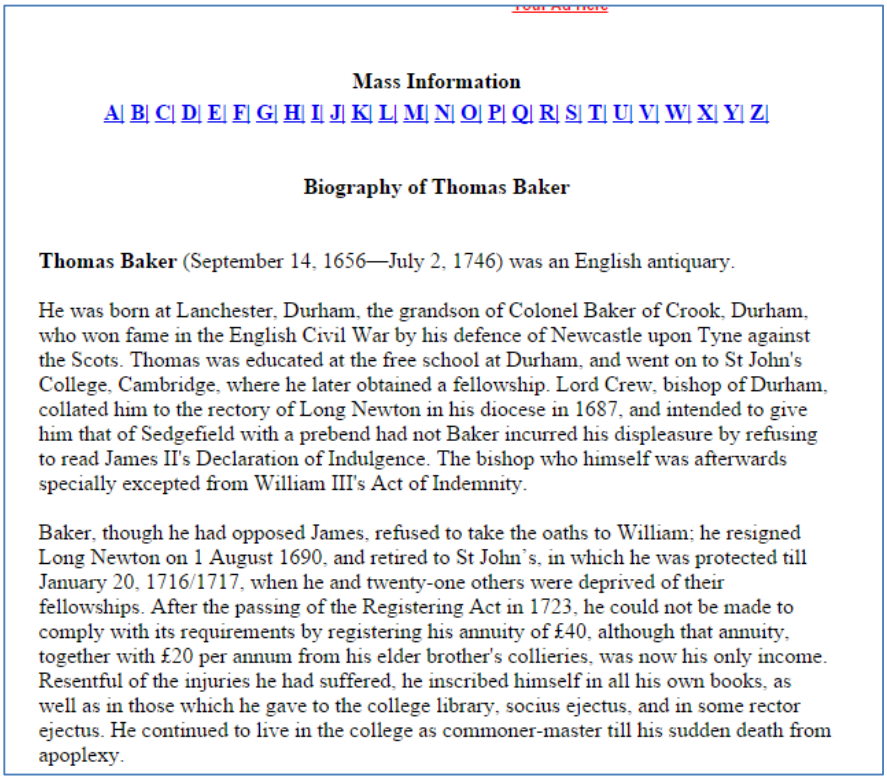

Figura 6. Parte de la página HTML original del fichero 050.txt.

En los trabajos de investigación dedicados a la tarea de desambiguación de nombres de personas en la Web se han utilizado diferentes enfoques, utilizando datos de entrenamiento o no, utilizando diferentes algoritmos, etc. En esta práctica no se utilizarán datos de entrenamiento y el algoritmo de clustering puede ser el propuesto en el código de partida sin necesidad de cambiar a otro. Con respecto a cómo representan las páginas web los diferentes trabajos de investigación sobre esta tarea, también hay mucha variedad: algunos sólo utilizan Entidades Nombradas, o bien las utilizan de forma diferenciada, o utilizan n-gramas, o incluso enriquecen la representación con técnicas de *entity linking* buscando información de los individuos en recursos externos como Wikipedia u otros, etc.

El objetivo de la práctica no es generar una representación de los documentos muy compleja, sino hacer cambios sobre el código proporcionado con cierto criterio, con el fin de mejorar los resultados de agrupamiento. Para ello se tendrá que investigar un poco acerca de esta tarea de desambiguación de nombres de personas en la Web para ver qué se suele hacer, investigar un poco el contenido de las páginas web para tratar de ver qué puede ser más importante a la hora de representar las páginas y también investigar cómo utilizar la librería NLTK para hacer las modificaciones necesarias en la representación de los documentos.

Es importante tener en cuenta que a veces un cambio puede no resultar mejor, pero acompañado de otro cambio adicional sí puede producir mejoras.

Por último, además de modificar y extender el código que se proporciona, se deben explicar qué cambios se han introducido y por qué. Es muy importante que se vayan describiendo los diferentes cambios realizados, junto con un análisis de si funciona mejor o peor y las razones por las que se cree que ocurre. Hay que razonar cada paso que se dé.