Asignatura	Procesamiento del Lenguaje Natural para la Ciencia del Dato
Documento	Practica: Sistema de recomendación de películas
Alumnos	Inmaculada Perea Fernández Juan Raggio Pérez Eduardo Sánchez Karhunen Marina Sarmiento Pelegrina
Fecha	28 de mayo del 2017

Índice

1 Objetivo

La práctica que se presenta a continuación consiste en implementar un sistema de recomendación de películas a partir del modelo analizado en clase.

Se va a desarrollar la opción de introducir en el sistema de recomendación otros parámetros adicionales, además de la sinopsis de la película, para lo que se va a incorporar en el sistema de recomendación el metadato correspondiente al género de la película.

2 Conjunto de datos

Como fuente de dato se ha tomado el dataset disponible en http://www.cs.cmu.edu/~ark/personas/data/MovieSummaries.tar.gz que contiene los siguientes ficheros:

- character.metadata.tsv
- movie.metadata.tsv
- name.clusters.txt
- plot_summaries.txt
- README.txt
- tvtropes.clusters.txt

Para el desarrollo de esta práctica se van a usar los ficheros movie.metadata.tsv y plot_summaries.txt, el primero contiene los metadatos básicos de las películas (Identificador, título, fecha, idioma y géneros) y el segundo contiene la sinopsis de cada película comenzando con el código de película, y a continuación el texto correspondiente a la sinopsis.

3 Implementación

Para el desarrollo de esta práctica se va a usar como lenguaje de programación Python con las siguientes librerías:

- ast: Librería para procesar árboles de la gramática de sintaxis abstracta de Python
- nltk: Librería para el procesamiento del lenguaje natural
- gensim: esta librería se usa para modelados de temas, indexación de documentos y recuperación de documentos con corpus grandes
- numpy: librería para el trabajo con vectores y matrices con funciones matemáticas de alto nivel

A continuación se describen los pasos que se han seguido para la implementación del código.

3.1 Paso 1: Leer ficheros con las sinopsis de las películas

Fase 1: Leer los metadatos

En primer lugar se van a leer los metadatos de las películas del fichero movie.metada.tsv y se van a guardar en un **diccionario** denominado **películas** este diccionario tiene como keys el código identificador de la película y como valores un diccionario por cada película con las siguientes keys:

- código: Código identificador univoco de cada película
- título: Título de la película
- fecha : Fecha de la película
- géneros: Lista de géneros en las que ha sido clasificada la película

Fase 2: Leer las sinopsis de las películas y vinculación a la entrada del diccionario correspondiente

Una vez se ha obtenido el diccionario con la información de las películas especificada en la fase 1, se procede a leer la sinopsis de las películas del fichero plot_summaries.txt.

Fase 3: Crear diccionario que solo tenga las películas para cuyos metadatos hay un resumen y algún genero

Hay que tener en cuenta que no todas las películas tienen sinopsis y géneros por lo que sólo se tendrán en cuenta aquellas que si los tengan ya que va a ser esta la información que se va a usar para comparar las películas. Serán guardadas en un diccionario con la misma estructura que el especificado en la fase1.

Para este apartado se han definido las siguientes funciones:

- ***getListOfGenres*:** tiene como input los géneros que vienen en los metadatos de las películas y devuelve una lista de géneros, se ha usado la función **literalo_eval** de la librería ast
- ***leerPeliculas*:** tiene como input el número de películas que se van a leer de los archivos y como resultado devuelve el diccionario de películas (con sinopsis y géneros) que tiene como keys el código identificador de la película y como valores otro diccionario cuyas keys son:
  - código: Código identificador univoco de cada película
  - titulo: Título de la película
  - resumen: Sinopsis de las películas
  - géneros: Lista de géneros en las que ha sido clasificada la película

3.2 Paso 2: Pre-procesado y limpieza de los resúmenes de las películas

Para este apartado se ha usado la librería **nltk** para pre-procesar los resúmenes de las películas siguiendo los pasos que se describen a continuación:

1. Extraer la lista de todas las palabras que componen el resumen , para lo que se usa la función **RegexpTokenizer**
2. Eliminar los signos de puntuación de los resúmenes , para lo que se usa la función

tokenize

3. Eliminar los nombres propios de los resúmenes ya que no son relevantes para el estudio, para este objetivo se ha definido la función ***obtenerNombresPropios*** en la que se usan las siguientes funciones de la librería nltk:

   - **word_tokenize** devuelve la lista de palabras que forma la frase (tokenización)

   - **pos_tag** devuelve el part of speech (categoría) correspondiente a la palabra introducida

   - **ne_chunk** devuelve la etiqueta correspondiente al part of speech

4. Eliminar las stopwords (artículos, pronombres, preposiciones..), para lo que se usa la función **Stopwords**
5. Eliminar los afijos de las palabras dejando sólo el lexema de estas para lo que se usa la función **SnowballStemmer**

Todos estos procedimientos son llamados en la función **preprocesadoPeliculas** que tiene como input un diccionario de películas y como resultado se obtiene una lista en el que cada elemento de esta en una sublista que guarda una palabra. Además agrega a cada película la key "texto" que tiene como valor el resumen pre-procesado.

3.3 Paso 3: Creación de la colección de textos

En esta parte del código se generan las siguientes funciones:

- ***crearColeccionTextos*** que tiene como input un diccionario de películas y tomando para cada película el apartado texto (resumen pre-procesado) crea a partir de este una lista compuesta por una sublista para cada película en donde cada elemento corresponde a una palabra del texto.
- ***crearColeccionGeneros*** que tiene como input un diccionario de películas y tomando para cada película el apartado géneros crea a partir de este una lista compuesta por una sublista para cada película en donde cada elemento corresponde a uno de los géneros de la película

3.4 Paso 4: Creación del diccionario de palabras y géneros

En este apartado se va a usar la función **corpora** de la librería **gensim** que proporciona un mapeado para cada palabra única con su identificador ya que el diccionario está formado por la concatenación de todas las palabras que aparecen en alguna sinopsis (modo texto) de alguna de las películas. Es decir, si tenemos N palabras, lo que se obtiene es que cada película sea representada mediante un vector en un espacio de N dimensiones.

Este procedimiento se va a aplicar tanto a los textos como a los géneros mediante la función ***crearDiccionario*** que recibe como entrada una colección de elementos (apartado 3.3) devuelve un elemento de tipo gensim.corpora.dictionary.Dictionary

3.5 Paso 5: Creación del corpus de resúmenes pre-procesados

En esta parte del código se va a generar un corpus con la colección de todos los resúmenes previamente pre-procesados y transformados usando el diccionario para lo que se crea la función ***crearCorpus*** que recibe un diccionario (apartado 3.4) y una colección (apartado 3.3) y mediante la función **doc2bow** de la librería gensim nos devuelve una lista de tuplas en el que el primer elemento es el identificador de la palabra y el segundo

elemento es el número de veces que aparece.

Aplicamos esta función tanto a los textos como a los géneros.

3.6 Paso 6: Creación del modelo tf-idf

Una vez se han creado los corpus se genera, a partir de estos, el **Modelo Espacio-Vector Tf-Idf** mediante la función ***crearTfdf*** que recibe un corpus (apartado 3.5) y le aplica a este la función **models.TfidfModel** de la librería gensim y devuelve una lista de tuplas en el que el primer elemento es el identificador de la palabra y el segundo elementocorresponde a la frecuencia de dicha palabra en el corpus.

Aplicamos esta función tanto a los textos como a los géneros.

3.7 Paso 7: Creación del modelo LSA (Latent Semantic Analysis)

Debido a que la dimensión en la que se trabaja es muy elevada, en lo que al corpus de resumes se refiere, se va a usar una técnica denominada **Latent Semantic Analysis (LSA ó LSI**) para reducir la dimensionalidad.

Esta técnica es un modelo puramente matemático basado en operaciones matriciales, para aplicarlo hay que indicar la dimensión a la que queremos reducir el problema.

Fase1: Función crearLSA

Se define la función ***crearLSA*** que tiene como argumentos de entrada un corpus (apartado 3.5), un modelo tf-idf (apartado 3.6), un diccionario (apartado 3.4) y el número de tópicos que se van a usar (se ha metido como parámetro de la función ya que para el caso de los géneros no es necesario disminuir la dimensión por lo que este parámetro tomará un valor para crear el LSA de resúmenes y otro para el de géneros).

Esta función devuelve como salida el par (lsi,indice) que se describen a continuación:

- lsi: Se genera un modelo LSI usando la función **models.LsiModel** de gensim lo que permite reducir la dimensionalidad mediante la factorización de matrices, y para el que se le indica la dimensión del problema.
- indice: Se calcula la similitud de los resúmenes almacenando la matriz de índice en la memoria. La medida de similitud utilizada es el coseno entre dos vectores. La matriz se almacena internamente como una matriz numpy densa, se ha usado la función **MatrixSimilarity** de la clase similiarities de la librería gensim.

Fase 2: Función crearCodigosPeliculas

Esta función recibe el diccionario de películas y devuelve un array con los códigos de las películas analizadas, va a ser usada para la función que se describe a continuación.

Fase3: Función crearModeloSimilitud

Esta función tiene como objetivo mostrar para cada película aquellas que son similares para lo que se requiere los siguientes parámetros de entrada:

- películas: diccionario de películas
- sinop_tfidf: modelo tf-idf para la sinopsis
- lsi_sinop: modelo lsi para la sinopsis
- indice_sinop:matriz de similitud para la sinopsis
- weight_sinopsis: peso que se le da a la similitud en las sinopsis

- gen_tfidf:modelo tf-idf para los géneros
- lsi_genre: modelo lsi para los géneros
- indice_genre: matriz de similitud para los géneros
- weight_genero: peso que se le da a la similitud en los géneros
- n_similares: número de películas similares para cada película
- salida: archivo de salida

Para calcular la similitud entre películas se han sumado las matrices de similitud de género y similitud de sinopsis multiplicando a cada una de ellas por su respectivo peso.

4 Caso práctico

Para el caso práctico se han leído 50 películas y se han tomado los siguientes parámetros:

- Modelo LSA aplicado al modelo tlidf de los corpus de las sinopsis se han tenido en cuenta un total de 300 elementos con el objetivo de reducir la dimensión.
- Modelo LSA aplicado al modelo tlidf de los corpus de los géneros se ha tenido en cuenta un total de 82 elementos ya que son todas las categorías que puede tomar el metadato genero y en este caso no era necesario reducir la dimensionalidad.
- Modelo de similitud, para construcción de este modelo se ha tomado como peso para la similitud de la sinopsis 0.7, a la similitud de géneros se le ha dado un peso de 0.3 y por último se le ha especificado que devuelva las 4 películas más similares para cada una de las películas.

Como resultado se obtiene el archivo “similitudes.txt” que tiene la siguiente forma:

```
Creando enlaces de similitud entre películas
Peso para sinopsis = 0.7
Peso para género = 0.3
Número de películas similares = 4
Generando salida en fichero  similitudes.txt
============================
[0] [27702708] Película I = Real Steel
   Similitud: 0.236    ==> [32293709] Película J = Buried Secrets
   Similitud: 0.232    ==> [33887035] Película J = Collateral Damage
   Similitud: 0.228    ==> [4519643] Película J = Larki Panjaban
   Similitud: 0.227    ==> [22145673] Película J = Sila Nerangalil Sila Manithargal
============================
[1] [32293709] Película I = Buried Secrets
   Similitud: 0.314    ==> [4519643] Película J = Larki Panjaban
   Similitud: 0.303    ==> [22145673] Película J = Sila Nerangalil Sila Manithargal
   Similitud: 0.291    ==> [24975131] Película J = Turn on to Love
```

Similitud: 0.291 ==> [31533575] Película J = Kamla

==========

[2] [6285672] Película I = Dying to Go Home

Similitud: 0.287 ==> [61323] Película J = The Love Parade

Similitud: 0.282 ==> [12625575] Película J = Turtle Diary

Similitud: 0.217 ==> [12981754] Película J = Heavenly Daze

Similitud: 0.212 ==> [4340003] Película J = Down to Earth

==========

[3] [6701433] Película I = Iyarkai

Similitud: 0.307 ==> [31357914] Película J = Sanju Weds Geetha

Similitud: 0.217 ==> [26297867] Película J = The Amorous Adventures of Moll Flanders

Similitud: 0.211 ==> [12625575] Película J = Turtle Diary

Similitud: 0.21 ==> [6285672] Película J = Dying to Go Home

==========