Ayudantías

Fundamentos de Programación

Semana 12

Diccionarios y conjuntos

Ejercicio 1

Se necesita crear un programa para generar las estadísticas de las palabras en un texto. Por ejemplo, a partir del siguiente texto:

```
Con la ayuda de un grupo de amigos y de valientes aliados Frodo emprende un peligroso viaje con la
misión de destruir el Anillo Único Pero el Señor Oscuro Sauron quien creara el Anillo envía a sus
servidores para perseguir al grupo Si Sauron lograra recuperar el Anillo sería el final de la Tierra
Media Ganadora de cuatro Oscars este inmortal relato sobre el bien y el mal la amistad y el sacrificio
te transportará a un mundo más allá de tu imaginación
...
```

su programa generaría el siguiente diccionario, apoyándose en las cinco funciones que se detallan abajo. Asuma que **cada palabra tendrá 20 caracteres como máximo** y que el **número máximo de palabras por líneas es 30**. También tiene una lista **stopwords** = ['la', 'con', …] con palabras que no agregan mayor significado al texto.

```
{
   'NTP': 83, 'NPC': 22,
   'palabras':
   {
     'Anillo': {'veces': 3, 'NL': (2,3,4)},
     'Sauron': {'veces': 2, 'NL': (2,3)   },
               ...
   }
}
```

Donde,

NTP = Número total de palabras
NPC = Número solo de palabras stopwords
NL = Número de líneas

Ahora implemente:

1) La función **cargarArchivo(nombre),** que leerá el texto desde el archivo nombre y creará una matriz M de NumPy (**con dtype='U20'**) donde cada fila representa una línea, y cada columna, una palabra de dicha línea. Si la línea tiene menos de 30 palabras, las celdas restantes deben ser llenadas con un string vacío. Cada línea del archivo está limitada por '\n'. Cada palabra está separada por un espacio en blanco.
2) La función **ocurrencias(palabra, M),** que devuelve el número de veces que el string palabra aparece en la matriz M.

3) La función **líneas(palabra, M),** que devuelve una tupla con los números de fila donde aparece palabra en M.
4) La función **contarPalabras(M, stopwords)** que devuelve una tupla con el número total de palabras del texto (**incluyendo las palabras stopwords**) y el número **solo** de palabras stopwords. **Cada palabra (regular o stopword) debe ser contada una sola vez** sin importar cuantas veces se repita en el texto.
5) Implemente la función **concordancia(M, stopwords)** que devuelve un diccionario con las estadísticas de las palabras (ver ejemplo arriba). El diccionario interno **'palabras' no debe incluir las palabras stopwords**.

Ejercicio 2

Para el intercambio comercial entre países se tienen archivos con las transacciones de venta. Las transacciones están almacenadas en un archivo diferente por cada categoría (Flores, Frutas, Maderas, etc.) de producto. Por ejemplo, las ventas de claveles, rosas, tulipanes, girasoles, etc. están almacenadas en el archivo "Flores.txt"; las ventas de cacao, banana, etc. están almacenadas en el archivo "Frutas.txt"

Flores.txt

```
Comprador,Vendedor,Producto,UnidadesVendidas,VentasEn$,Fecha
Estados Unidos,Ecuador,rosas,59284,631432.21,2018-01-10
Holanda,Japon,tulipanes,2384,12434.87,2017-11-22
…
Estados Unidos,Ecuador,girasoles,38284,331432.75,2018-02-01
```

Note que un país puede vender el mismo producto al mismo comprador pero en una fecha diferente.

Desarrolle lo siguiente:

1. La función **calculaTotales(categoria)** que recibe el nombre de una categoría. La función deberá leer el archivo para esa categoría y retornar un diccionario de totales con la siguiente estructura:

```
totales = {(comprador,vendedor,producto):totalDolares}
```

Por ejemplo, si en la categoría “Flores” Estados Unidos le compró a Ecuador rosas en 12 fechas diferentes, en el diccionario deberá aparecer un solo item con las sumas totales de esas 12 transacciones.

```
{('Estados Unidos','Ecuador','rosas'):32645788}
```

2. La función **consolidado(nomArchivo, categorias)** que recibe una lista de categorías y genera un archivo con nombre **nomArchivo** en el que se listen todos los totales de ventas acumulados por Comprador, Vendedor, Producto. El archivo tendrá la siguiente estructura:

```
Comprador,Vendedor,Categoria,Producto,VentasTotalesDolares
```

Para el resto del ejercicio asuma que tiene una función crearMatriz que recibe el nombre del archivo consolidado y devuelve una tupla con tres elementos: (1) matriz M cuyas filas representan países compradores, columnas representan productos ordenados alfabéticamente dentro de cada categoría y las celdas representan ventas totales en dólares, (2) lista con las etiquetas de las filas y (3) lista con las etiquetas de las columnas. Las categorías no están ordenadas alfabéticamente.

3. La función **comprasCategorias(nomArchivo, dicCat)** que recibe el nombre del archivo consolidado y un diccionario donde las claves son las categorías y los valores son listas con todos los productos ordenados alfabéticamente dentro de cada categoría. La función deberá generar por cada categoría un archivo con el nombre de la categoría y extensión “.txt”. Cada archivo debe contener los **5 países que más han comprado para esa categoría**. Cada archivo tendrá la siguiente estructura:

Ejercicio 3

Dado el archivo rutasManejadas2018.txt con información como la que sigue:

```
id_ruta, id_chofer, fecha
Guayaquil-Cuenca,SMSNADOPN,17-05-2018
Guayaquil-Cuenca,AGBCCAPMP,18-05-2018
Guayaquil-Daule,EVNTAASFL,17-05-2018
Guayaquil-Daule,AAQSPTTGL,18-05-2018
```

Suponga que dispone de una función **calcularFecha(fecha, n)** que recibe una fecha y un entero. La función retorna la fecha correspondiente a los n días anteriores a la fecha del parámetro (sin incluirla).

Implemente lo siguiente:

1. La función **cargarDatos(nomA)** que recibe el nombre del archivo con los datos anteriores. Esta función **retorna una tupla** de dos elementos. El primer elemento es un conjunto con los ids de TODOS los choferes mencionados en el archivo. El segundo elemento es un diccionario con la siguiente estructura: {**id_ruta**: {**fecha**:{ch1,ch2,..., chk}}}. Ejemplo del diccionario:

```
{"Guayaquil-Cuenca":{"17-05-2018":{"SMSNADOPN","AGBCCAPMP",...},
                     "18-05-2018":{"...","...",...},
                     ...},
 "Guayaquil-Daule": {"17-05-2018":{"EVNTAASFL","AAQSPTTGL",...},
                     "18-05-2018":{"...","...",...},
                     ...},
```

2. La función **encontrarChoferes(dicc, fecha, losChoferes, id_ruta, n)** que recibe el diccionario del numeral anterior, una fecha (con formato dd-mm-yyyy), el conjunto con los ids de TODOS los choferes, el nombre de una ruta y un entero n. Esta función retorna un conjunto con los ids de todos los choferes que NO hayan manejado la ruta **id_ruta** en los **n** días anteriores a **fecha** (sin incluir fecha). Por ejemplo, si **n** es 3 y la **fecha** es "02-05-2018", la función devuelve un conjunto con los ids de choferes que NO hayan manejado id_ruta el 29, 30 de abril y el 1 de mayo de 2018.

3. La función **grabarArchivo(fecha, diccionario, losChoferes, n)** que recibe una fecha, el diccionario del numeral 1, un conjunto con los **IDs** de todos los choferes y un número entero **n**. Esta función crea un archivo, cuyo nombre tiene el formato **idRuta_fecha.txt**, para cada ruta con los choferes que NO han manejado la ruta **id_ruta** en los **n** días anteriores a la **fecha** (sin incluir fecha). El formato para este archivo es el siguiente:

```
Para la ruta Guayaquil-Cuenca, los choferes disponibles para la fecha 19-05-2018 que no
hayan manejado 2 dias anteriores son:
VSSUIMCMS
SJMPYSANL
...
```