DESARROLLO DE ALGORITMOS ORIENTADOS A OBJETOS

Evaluación

UCO ONLINE

Desarrollo de algoritmos orientados a objetos

Objetivos

Desarrollar un programa en Python que, a parte de hacer uso de las estructuras de datos explicadas, incluidas las del módulo `collections`, esté diseñado con orientación a objetos y preparado como un paquete de Python.

Temporización

360 minutos

Enunciado

Descripción

Nos proponemos crear un módulo que sirva para extraer datos característicos de textos. Posteriormente utilizaremos este módulo para aprender las características diferenciadas de textos en 5 idiomas y crear un algoritmo capaz de averiguar en cual de esos 5 idiomas está escrito un texto.

Objetivo

Implementar un modulo que tenga una clase `Texto`. Los objetos de esa clase se inicializarán con una cadena de texto y tendrán métodos para calcular:

- Número de palabras. Los separadores serán principalmente espacios pero pueden ser otros signos de puntuación o un salto de línea.
- Número de palabras con: a) todas las letras en minúscula, b) con solo la primera letra en mayúscula y c) con todas mayúsculas.
- Número de palabras de cada tamaño. Por ejemplo, el método `palabras_por_tamaño` podría devolver un diccionario con `{1 : 14, 2 : 38, 3 : 50, … }` indicando así que hay 14 palabras con una sola letra, 38 con dos letras, etc.
- Frecuencia de aparición de palabras de cada tamaño. Lo mismo que el anterior pero dividido por el total de palabras que aparecen en el texto. Por ejemplo: `{1 : 0.07, 2 : 0.18, ...}`

- Número de palabras por frase. Las frases tienen que están siempre separadas por un '.'
- La frecuencia de aparición de cada carácter en el texto. Por ejemplo, el método `frecuencia_caracteres` podría devolver un diccionario con `{'a' : 0.023, 'b' : 0.012, … }`.

Usando el módulo anterior implementar un programa que trate de averiguar el idioma en el que está escrito un texto, partiendo de la base del fichero `plantilla_principal.py` que adjuntamos a esta tarea.

Descripción de los requisitos del módulo `texto`

Deberá poderse crear un objeto con un texto así:

```
t = Texto("""Este es un texto muy laaaaaarrrrgggooooooooo que puede tener varias líneas""")
```

Debe usarse la clase `Counter` del módulo `collections` y todas aquellas que se considere oportuno.

Para calcular las frecuencias, debe implementarse una subclase de `Counter` que incluya los métodos:

- `total(self)` que devuelva el número total de elementos contados (palabras o caracteres en nuestro problema).
- `dividido(self, divisor)` que devuelva un diccionario que contenga todos los elementos.

Para devolver las frecuencias recomendamos usar la clase `defaultdict` del módulo `collections`. De forma que, si algún elemento no aparece, su frecuencia sea 0.

Indicaciones para el desarrollo del programa que identifica el idioma

Con los métodos implementados en la clase Texto es posible calcular muchas características de un texto. Algunas de ellas son muy diferentes entre idiomas. La idea es calcular estas características para los textos de cada uno de los 5 idiomas que os proveemos en la plantilla.

Para averiguar el idioma del texto de prueba, calcularemos las mismas características para el texto de prueba y compararemos su valor con las de cada uno de los 5 idiomas. El más similar será aquél que el algoritmo reconocerá como idioma del texto.

Para agregar la diferencia de varias características, podéis usar la distancia de Manhattan (suma del valor absoluto de las diferencias):

$$\sum_{j=1}^{N} |v_j - u_j|$$

donde v_j es el valor de la característica j para uno de los textos y u_j el valor de esa característica en el otro texto.

Invocación del programa

Para poder poner textos largos sin usar ficheros (que se verán en próximas lecciones), introduciremos los textos a probar como se muestra en el ejemplo de la plantilla con el `texto_prueba`. Os animamos a probar con textos en los 5 idiomas. Se pueden encontrar fácilmente en webs como la Wikipedia. Por tanto, un ejemplo de llamada puede ser algo así:

Ejemplos

python3 plantilla_principal.py

Salida → Francés

Extra (para la máxima puntuación)

Los objetos `Texto` con los que estamos trabajando podemos considerarlos inmutables (el texto se asigna en su creación y no se modifica luego). Entonces, las características que hemos calculado sobre ellos no van a cambiar y, por tanto, recalcularas cada vez que se necesite es un coste computacional innecesario. Podríamos tener en cuenta eso al programar los métodos pero os proponemos un ejercicio con decoradores para hacerlo de forma general y elegante.

Ejercicio: Implementar un decorador `@recuerda` que, cuando se asocie a un método, la primera vez que sea llamado utilice el método para calcular lo que hay que devolver y lo almacene en una variable para la próxima vez. Las demás veces que sea llamado devolverá lo almacenado en la variable sin llamar al método. Usarlo en algunos métodos de la clase `Texto`.

Nota: varias de las cosas que os pedimos implementar en este ejercicio ya están implementadas en el propio lenguaje Python o alguna librería. Por ejemplo, ya hay algunos decoradores con una función similar a la de `@recuerda`. Sin embargo, consideramos que es interesante enfrentarse al reto de implementarlas para tener una mejor comprensión de como funcionan internamente y, por ejemplo, ser capaces de adaptarlas en el futuro para otras funcionalidades.