Clasificación de correos electrónicos

Alumno: Juan Rodríguez Suárez, alu0101477596@ull.edu.es

Asignatura: Inteligencia Artificial Avanzada

Fecha: 26 de abril de 2024

Índice

1. Preprocesamiento

Para preprocesar los datos vamos a realizar las siguientes operaciones sobre todos los emails:

- Pasar a minúsculas.
- Eliminación de signos de puntuación.
- Eliminación de palabras reservadas
- Eliminación de emojis y emoticonos.
- Eliminación de URLs, etiquetas HTML, hashtags

Para ello, se han usado expresiones regulares para suprimir lo mencionado anteriormente y sustituirlo por un espacio en blanco.

Adicionalmente, todos los correos que contengan algún carácter no imprimible se han sustituido por el token <NO-ASCII> directamente. También sucede lo mismo con los correos vacíos o que quedan vacíos después del procesamiento. En este caso sustituimos por el token <EMPTY>.

2. Librerías utilizadas

Se ha utilizado la librería NLTK que contiene una función para tokenizar una frase. Por otro lado, en cuanto a las librerías de Python en sí, se han empleado los siguientes módulos:

- *re* (Expresiones regulares)
- *math* (Para la función logarítmica)
- *time* (Para mostrar los tiempos que tarda cada parte del programa)

3. Implementación de los programas

Todos los programas se han realizado en un entorno Python versión 3.12.3 con los módulos y dependencias ya comentados anteriormente.

3.1. Tokenizador

En el tokenizador, se ha creado una clase *CSVFormatter* que toma como entrada un archivo CSV como el dado para este problema y lo formatea de tal manera que en cada línea quede un correo electrónico preprocesado seguido de un delimitador ';' y la clase a la que pertenece.
Por otro lado, está la clase *Tokenizer* que toma como entrada un fichero de datos formateado como el anterior y devuelve una estructura de datos con los tokens de un email y la clasificación de dicho email. Además, aquí se genera el fichero de *vocabulario.txt*.

3.2. Estimar los modelos

Para estimar los modelos, se parte de la estructura de datos generada por el tokenizador y a continuación se genera otra estructura de datos que para cada palabra almacena la frecuencia de aparición para cada clase.
Además, se ha optado por si en el caso de que una palabra aparezca menos de dos veces, entonces sustituirla por un token <UNK>. Esto nos permite eliminar palabras extrañas además de darle una probabilidad distinta de cero a palabras que no estén en el corpus para el futuro clasificador.
Por último, con la estructura de datos generada, es trivial generar los ficheros de modelos del lenguaje.

3.3. Clasificador

En el clasificador, se ha creado una clase *Predictor* que almacena los modelos en objetos de la clase *Model* que tiene métodos para acceder rápidamente a las probabilidades logarítmicas de cada modelo además de tener toda la información de los ficheros de los modelos.
En la clase *Predictor* se analiza un fichero que contiene emails completos sin formatear en cada línea y para cada uno de ellos aplica el mismo preprocesamiento que para crear los modelos y se hace el sumatorio de los logaritmos de las probabilidades de cada palabra generada más la probabilidad de dicha clase. Finalmente, se escoge la clase con mayor probabilidad obtenida y se muestran los resultados en dos ficheros: uno con información detallada para cada email clasificado y otro resumiendo la clasificación.

4. Estimación probabilidades con diferentes conjuntos de entrenamiento y validación

4.1. Conjunto de entrenamiento es igual al de validación

En este caso, se obtienen 13448 emails clasificados correctamente de los 15000, es decir, un 89.65% de aciertos.

4.2. Conjunto de entrenamiento 2/3 y validación 1/3

En este caso, de los 5000 emails clasificados, 4678 son clasificados correctamente. Eso equivale a un 93.56% de aciertos.