Proyecto PLN - Inteligencia Artificial Avanzada

Sebastián Daniel Tamayo Guzmán.

Lenguaje y librerías utilizadas.

Para la realización de este proyecto opté por utilizar Python 3 y utilicé las siguientes librerías: * **csv:** Esta librería facilita la lectura y tratamiento de archivos de tipo .csv . * **re:** Esta librería proporciona herramientas para la creación de expresiones regulares que serán utilizadas en el preprocesamiento con el fin de identificar y eliminar URLs. * **math:** Esta librería proporciona herramientas de cálculo matemático, por ejemplo para el cálculo de logaritmos. * **nltk:** Natural Language Tool Kit es una herramienta que proporciona numerosas herramientas para el procesamiento y análisis de lenguaje natural humano. En mi trabajo utilizo la función *word_tokenize* con el objetivo de extraer las palabras independientes o tokens de las descripciones.

Preprocesamiento del corpus

Se han llevado a cabo las siguientes tareas: * Paso a minúsculas de las palabras. * Eliminación de caracteres no alfabéticos. * Eliminación de palabras reservadas (stopwords). * Eliminación de URLs.

Implementación del código

`vocabulario.py`

Este programa se encarga de leer el fichero de entrenamiento ecom-train.csv. A continuación aplica el preprocesado a las descripciones de los artículos y computa el vocabulario, generando una lista de palabras únicas y ordenadas que serán volcadas en el archivo de salida vocabulario.txt.

`aprendizaje.py`

Este programa también leerá el fichero de entrenamiento, pero su objetivo será recoger información relativa a cada corpus generado por las distintas categorías de los artículos. De esta forma se generarán cuatro archivos correspondientes a las cuatro categorías existentes, Household, Books, Clothing & Accessories y Electronics, que contendrán el vocabulario correspondiente a la categoría en cuestión y el logaritmo neperiano de la probabilidad de aparición de cada palabra. A este valor se le aplica suavizado Laplaciano. Cabe destacar que se agrega la palabra \<UNK> a cada corpus para representar las palabras desconocidas. Un valor dentro del código (kUnk) indica el mínimo de apariciones que ha de tener una palabra para no ser considerada desconocida. Los resultados se guardarán correspondientemente en los ficheros de salida aprendizaje<H, B, C o E>.txt .

`test-helper.py`

En este proyecto se plantea usar el mismo fichero de entrenamiento para realizar una prueba de predicción. Este programa será de utilidad para generar un fichero de testeo, que solo contenga descripciones y un fichero de validación, que solo contenga las categorías a las que pertenecen las descripciones del primer fichero.

`clasificacion.py`

Este programa analizará un fichero de prueba que contenga solo descripciones de artículos y generará un fichero con las predicciones para las categorías de las descripciones de entrada. Para conseguir este objetivo, se aplicará el preprocesado a las descripciones de entrada y será calculado el valor asociado a cada descripción, correspondiente a la suma de los logaritmos neperianos de las probabilidades de las palabras que contenga la descripción.

`compute-error.py`

Este programa tomará como entrada un fichero en el que se encuentran las predicciones realizadas por el programa anterior y un fichero de validación con las clases que realmente corresponden a las descripciones analizadas para la generación de las predicciones a validar. La finalidad del programa será comparar ambos ficheros para calcular el porcentaje de error en la predicción.

Error de la predicción

La predicción realizada por mi conjunto de programas tiene un error del **4,76%**. Cabe destacar que este valor no es muy orientativo, pues la validación y cálculo del error debería realizarse con un conjunto distinto al de entrenamiento.