**CLASIFICACIÓN DE DOCUMENTOS**

**WORD2VEC**

MEMORIA



DAVID GAVIRA SERRANO ALEJANDRO ORTIZ PAGADOR

CLASIFICACIÓN DE DOCUMENTOS

David Gavira Serrano, Alejandro Ortiz Pagador.

***Resumen* - En este trabajo hemos llevado a la práctica lo aprendido sobre la clasificación de documentos mediante un corpus de entrenamiento y un modelo entrenado.   
Conforme los objetivos del proyecto nos marcaban, hemos utilizado la librería Word2Vec (entre otras), para la vectorización de los corpus tanto del conjunto de datos como del conjunto test.**

**Finalmente, se nos encomendaba realizar adicionalmente un modelo de Naive Bayes con el mismo propósito y comparar los resultados obtenidos.**

**Palabras claves: corpus, vectorización, texto, modelo, entrenamiento, clasificación.**

I. INTRODUCCIÓN

Como proyecto final del curso hemos optado por la clasificación de documentos mediante un programa de Python representado en un Notebook “.*ibynp”*.

En este documento comentaremos como ha sido el desarrollo del código y como hemos obtenido la información necesaria para la realización del ejercicio.

II. DESARROLLO DE CONTENIDOS

Para comenzar el proyecto se llevo a cabo un planteamiento general siguiendo el esquema proporcionado por el profesorado. Se analizaron los diferentes requisitos y se fueron completando de manera progresiva.

*A. Tokenización y adaptación del corpus.*

El primer paso fue la adaptación de los textos que iban a formar parte del corpus de entrenamiento del modelo, aunque posteriormente también necesitamos adaptar y tokenizar el texto de prueba.

Para ello diseñamos un conjunto de restricciones sobre diferentes caracteres o palabras que no daban valor al texto y las retiramos en la función:*“def tokenizar()”.*

Como comentario me gustaría resaltar que este proceso pudo ser un poco complejo debido a la gran cantidad de textos posibles y por lo tanto necesitábamos tener una gran cantidad de caracteres que eliminar para no ser considerados. Aún habiendo realizado un gran estudio de los textos y habiendo añadido una larga lista de caracteres, sería posible que se añadiesen caracteres extraños a los vectores de palabras ya que no nos ha sido posible encontrar ningún archivo con todos los caracteres existentes.

*B. Creación del modelo y entrenamiento.*

El segundo requisito fue crear y entrenar un modelo que posteriormente sirviera para la clasificación de documentos. Durante este apartado rehusamos del documento enunciativo y realizamos unos cuantos intentos por nuestra cuenta analizando diferentes documentos y ejemplos para comprender realmente el proceso.

Pese a nuestro afán de realizar manualmente el requisito de crear y entrenar un modelo, comprendimos que utilizando el conjunto de librerías mencionadas en el enunciado del proyecto suponía mucha mas simplicidad y funcionaba de manera más correcta.

Para concluir este apartado, realizamos las funciones “*add\_text\_to\_corpus(…)*” y *“tran\_word2vec(…)”.*

*C. Vectorización o Embeddings.*

Una vez tuvimos el corpus de entrenamiento y el modelo entrenado pasamos a la vectorización de los textos o como se dice técnicamente, **creación de los embeddings**.

Estos embeddings o vectores se componen de un conjunto de números reales que representan la semántica de la palabra en cuestión en el contexto del corpus.

Hay que añadir que desarrollamos una pequeña aplicación para comprender el uso de dichos elementos, en la cual buscamos el conjunto de palabras mas similares a una palabra contenida en el corpus.

*D. Modelo Naive Bayes.*

Hemos decidido dejar este paso para el final ya que nuestro objetivo era principalmente comprender en profundidad el comportamiento del programa. Una vez terminado y ejecutado comprobando su correcto funcionamiento, comenzamos con el modelo de Naive Bayes.

Para este caso nos vimos obligados a la creación de un método adicional llamado “*embedding\_NB(…)*” donde se realizan pasos análogos al apartado anterior pero formateando los elementos según fuesen necesarios para esta clase de modelo.

*E. MAIN.*

En este apartado se lleva a cabo la ejecución del código completo mediante el conjunto de llamadas a los diferentes métodos encargados de desempeñas cada una de las tareas previamente mencionadas.

Se realizaron algunas instanciaciones necesarias para el correcto funcionamiento del programa y las llamadas a las funciones nativas de las múltiples librerías utilizadas.

III. FUNCIONAMIENTO DE WORD2VEC.

Word2Vec es un algoritmo que tiene como fin el aprender las representaciones vectoriales de un texto dado. De esta manera es posible posteriormente predecir las palabras circundantes a una palabra dada en un contexto determinado.

Inicialmente se construye un vocabulario a partir del corpus de entrenamiento y se le asigna un vector inicial aleatorio. Iterativamente este conjunto de vectores se va modificando para representar las relaciones contextuales entre las palabras del texto.

Tras este proceso obtenemos los conocidos embeddings o vectores que hemos descrito en este mismo documento, que utilizará la librería *sk-learn* para la clasificación de documentos.

IV. CONCLUSIONES.

Para concluir este documento nos gustaría comentar los resultados obtenidos tras la ejecución del programa, donde se ejecutan ambos métodos de clasificación (Word2Vec y Naive Bayes).

Como podrán observar tras la ejecución del código el resultado es similar por lo que entendemos que ambos métodos están funcionando correctamente y que simplemente se diferencian en la lógica interna para el desempeño de la tarea.

Además, cabe comentar que se ha realizado pruebas con textos de un tamaño relativamente pequeño por cuestiones de rendimiento y de facilidad de formateo. Posteriormente los textos de pruebas han sido generado por Chat GPT, de esta manera obteníamos textos con una extensión razonables alrededor de la temática deseada que queríamos comprobar.

V. REFERENCIAS.

[1] Chat GPT. URL: <https://chat.openai.com/>

[2] Recursos Pthon. URL: <https://recursospython.com/guias-y-manuales/convertir-texto-unicode-a-ascii-7-bits/>

[3] Formato IEEE Conference proceedings. URL: <https://revistas.pucp.edu.pe/imagenes/electro/ee_formato_ieee.pdf>

[4] Stop-Word list. URL: <https://github.com/Alir3z4/stop-words/blob/master/spanish.txt>

[5] NLTK URL: <https://www.geeksforgeeks.org/python-stemming-words-with-nltk/>