“Se Cayó la App”.
Recomendador de Noticias SCA

Contenido

1. Reproducibilidad

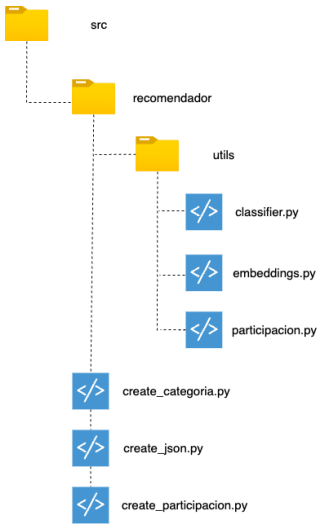
El proyecto se desarrolló en Python 3.10.6 a través de un ambiente de desarrollo virtual. Se recomienda seguir las siguiente instrucciones para reproducir el ejercicio en windows:

- Crear ambiente virtual con Python superior a 3.10.0
- Instalar del archivo requeriments.txt (en la raíz del proyecto) las librerías requeridas.
- Descargar los siguientes archivos de nltk en dado caso de no tenerlos:
    a. nltk.download('stopwords')
    b. nltk.download('punkt')
- Ejecutar desde el ambiente y la raíz del proyecto, los siguientes comandos:
    a. py src/recomendador/create_participacion.py
    b. py src/recomendador/create_categoria.py
- Si bien el repo ya tiene en output el archivo de categorías, los comandos anteriores generan automáticamente esta salida. Por lo que no es indispensable ejecutarlos.

Se aclara que se toma de una carpeta en la raíz del repositorio los datos para generar todo el modelado, **la cual no fue cargada al repositorio**. Esta carpeta se llama “datos” y contiene los 3 archivos compartidos por Bancolombia nombrados exactamente igual.

2. Detalle Repositorio

El repositorio mantiene la estructura recomendada por el equipo de Bancolombia, incluyendo solamente dentro de la carpeta “recomendador” una nueva carpeta que contiene el módulo de Python que almacena las funciones usadas en los otros scripts. Tal como se evidencia en el siguiente diagrama.



Este modulo se basa en 3 scripts principalmente:

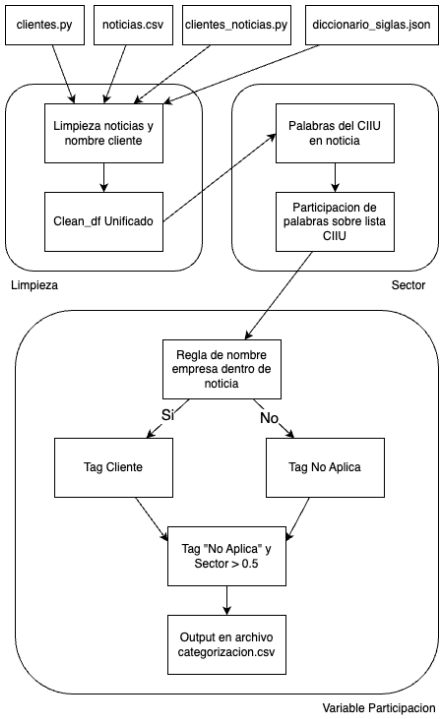
a. Classifier.py contiene las funciones usadas en el proceso de los modelos de clasificación de texto.
b. Embedding.py contiene las funciones que se requieren para procesar y generar el modelo Word2Vec.
c. Participación.py contiene todas las funciones requeridas para construir la variable “participación”

Dentro de cada script se encontrará la respectiva documentación de las funciones.

Los otros scripts dentro de recomendador permiten generar dentro de la carpeta output el archivo categorizacion.csv. Es de aclarar que no es necesario ejecutar el script create_json.py ya que este permite crear un insumo de abreviaciones y siglas que se eliminaran del nombre de la empresa.

3. Variable Participación

El flujo general para construir esta variable se puede entender en el siguiente diagrama, en general se hace todo el procesamiento de la variable ‘nombre’ dentro del archivo clientes.csv, eliminando stopwords y las siglas del archivo json que se mencionó previamente. Al archivo clientes_noticia se le pega la variable text_content_news para posteriormente hacer el mismo proceso que al nombre del cliente, cabe aclarar que no se eliminan letras en mayuscula, ya que atreves de estas se identificaran los nombres de las empresas.



En la sección de “Limpieza” se realiza todo lo mencionado anteriormente, en la siguiente sección llamada “Sector” se toma todo el detalle de los código CIIU para crear una lista de palabras que hablan sobre el sector del cliente, posterior a esto se calcula la proporción de palabras de esta lista que aparecen dentro de la noticia, independientemente de cuantas veces aparezca. Esto genera una variable que denominamos “prop_sector” que cuantifica de 0 a 1 si se habla del sector de un cliente en la noticia o no.

Con lo anterior en mente, en la última sección “Variable Participación” se evalúan las siguientes reglas:

- Si se menciona el cliente dentro de la noticia, directamente se asigna el tag “Cliente” sino se coloca “No Aplica”.
- Para todas aquellas noticias con no aplica pero con valores superiores al 0.5 dentro de “prop_sector” se asigna el tag “Sector”.

4. Variable Categoría

Para esta ultima parte, todo el análisis se basó en el archivo “noticias.csv” y en los nombres de las categorías requeridas, sin incluir las categorías “Otras” y “Descartable”.

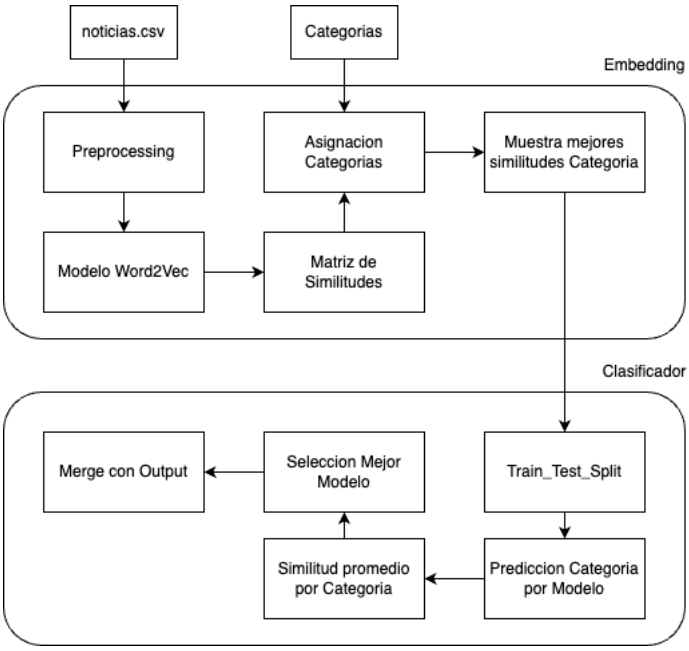
A modo general se tienen dos secciones principales, la primera el Embedding y la segunda relacionada a los modelos de Clasificación.

Para la primera sección se realizó la tokenizacion del texto de la noticia, eliminando stopwords del texto, se usó un modelo Word2Vec para generar representaciones vectoriales de longitud 100 y que

permitió posteriormente usar distancia del coseno para evaluar la similitud de las categorías solicitadas con los vectores promedio de las palabras en cada noticia.

Para asignar una categoría a la noticia, se calculó la distancia para las 6 categorías y se tomó aquella que mayor similitud tuviera.

Al final de este proceso se toman las 100 noticias más similares de cada categoría las cuales entraran al modelo de clasificación.



En la última parte del proceso “Clasificador” se toman como correctas las 100 noticias de las 6 categorías, un total de 600 noticias con tag de categoría que se usaron dentro de varios modelos de clasificación para generar el modelo que permitiera clasificar las demás noticias.

Sobre los modelos evaluados se usó Regresión logística en dos escenarios, el primero sobre el texto de la noticita como tal y el segundo sobre los vectores generados por el embedding. Para este caso se tuvo un acuracy de 99% sobre la muestra, tenemos en cuenta que una muestra de 100 noticias puede ser una muestra baja para poder evaluar el modelo.

Pero no se usó solo el acuracy para medir el desempeño del modelo sino que una vez se generaron las predicciones de todos los modelos usados, se evaluó la máxima similitud promedio para todas las categorías sobre los vectores del embedding creado inicialmente, dando como resultado que la regresión logística logro un promedio de similitud sobre las categorías del 89%

Al final del proceso, se toma el output generado del script de participación para agregar la variable de categoría que será capaz de según la evaluación expuesta previamente seleccionar el modelo con mejores resultados.