Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

E-Commerce Text Classification

Avance

Chaparro Sicardo Tanibeth, Malváez Flores Axel Daniel

Project #1

2023

Definición del problema

Con el crecimiento exponencial de los datos y de las ventas en línea, lo que proponemos es hacer un clasificador de texto correspondiente a los productos de venta en línea y su descripción. El volumen de información disponible hace que la revisión y clasificación manual de los productos se vuelva extremadamente ineficiente, consumiendo una cantidad significativa de tiempo y recursos. En este caso lo que se desea es poder hacer este tipo de clasificaciones de manera eficiente y correcta.

Objetivos

`Objetivo general`

Se tiene como objetivo desarrollar un clasificador que automatice el proceso de categorización de productos de venta en línea a partir del análisis de su descripción. Se busca determinar si el uso de técnicas avanzadas de Procesamiento de Lenguaje Natural (PLN) logran mejorar el proceso de clasificación (a diferencia de los trabajos que se tienen como "estado del arte" que no realizan un amplio preprocesamiento del texto) al extraer la información relevante de la descripción de los productos. Por otro lado, se desea identificar el modelo de clasificación más eficaz para el problema en cuestión, para esto se realiazará una comparación entre los resultados del modelo usando diferentes métricas de evaluación. Por último se desea mejorar los scores que se tienen en otros trabajos similares a este (arriba del 93%).

`Objetivos especificos`

- Procesamiento del Lenguaje Natural: Aplicación de técnicas de PLN para la extracción de información relevante de las descripciones de los productos. Generación de etiquetas. Extracción de características y vectorización.
- Selección del algoritmo: Elegir el mejor algoritmo de clasificación de textos para nuestra tarea. Los algoritmos con los que se trabajará son Naive Bayes, Support Vector Machines y XGBoost.
- Entrenamiento y evaluación: Utilizar el conjunto de datos para entrenar el modelo seleccionado. Realizar múltiples pruebas con métricas, como **accuracy**, **f1-score** y **confussion matrix**, para evaluar la precisión del modelo y a partir de los resultados realizar los ajustes necesarios al modelo.

Avances y Resultados

`Preparacion y Exploracion de Datos`

Para este proyecto se está usando un conjunto de datos obtenido de la plataforma Kaggle [Sha19] que contiene la descripción de productos pertenecientes a las categorías *Electronics*, *Household*, *Books* y *Clothing & Accesories*. Los datos se encuentran en formato ".csv" y contienen dos columnas, una columna de texto con la descripción de los productos (*Description*) y la segunda con la etiqueta de su categoría (*Category*).

El documento fue cargado en Python como un *DataFrame* de **Pandas**. La exploración inicial mostró que en total se cuenta con 50,425 registros, de los cuales 38.3% pertenecen a *Household*, 23.5% a *Books*, 21.1% a *Clothing & Accesories* y 17.2% a *Electronics*. Las estadísticas nos indican un desbalance en las clases, lo cual será necesario tomar en cuenta al momento de elegir los

modelos de clasificación y las métricas de evaluación.

Limpieza, Stemming y Lematizacion

En el proceso de limpieza se buscaron registros con información faltante. Se encontró un único registro con valor nulo en la columna *Description*, por lo que la observación fue eliminada. Posteriormente se realizó el procesamiento del texto. Este paso es necesario pues es a partir de este que podemos realizar las transformaciones posteriores de nuestros datos.

Empezamos estandarizando las cadenas de texto de la columa *Description*; es decir, se convirtió el texto a minúsculas, se eliminaron los caracteres especiales y las stopwords (palabras irrelevantes para el análisis pues no aportan significado semántico). Después se realizó el conteo de las palabras en cada descripción del producto a partir de la tokenización, el cual es un proceso de segmentación del texto por palabra. Finalmente utilizamos dos técnicas de reducción de palabras, **Stemming** y **Lematización**, las cuales buscan facilitar el análisis del texto.

El procesamiento anterior fue realizado con las siguientes funciones y utilizando las herramientas disponibles en la librería **NLTK**, como **PorterStemmer** y **WordNetLemmatizer**:

1. ***process_text***: Procesa una cadena dada eliminando los caracteres especiales, convirtiendo el texto a minúsculas y eliminando stopwords.
2. ***tokenize_stemming_func***: Tokeniza una cadena dada y realiza stemming.
3. ***tokenize_lemma_func***: Tokeniza una cadena dada y realiza lematización.

Debido al volumen de nuestros datos fue necesario usar la librería **PySpark** para poder pasar las funciones a cada registro. El *DataFrame* de Pandas se convirtió en un *DataFrame* de Spark, lo que permitió hacer uso del procesamiento distribuido para optimizar el tiempo de ejecución de las funciones sobre los datos.

Al final de este proceso, el *DataFrame* de Spark fue convertido a dos *DataFrames* de Pandas: uno conteniendo la columna *Description* tokenizada y lematizada, y el segundo con la columna *Description* tokenizada y con stemming.

Vectorizacion

La **vectorización** es un proceso imprescindible al momento de querer trabajar con lenguaje natural para una tarea de clasificación pues convierte texto en datos numéricos los cuales ahora dichos datos pueden ser utilizados para realizar la tarea antes mencionada.

Es necesario previamente ya haber preprocesado nuestro texto para que pueda estar limpio y en un formato estándar. Dado que ya se describió anteriormente cómo se realizo este paso de una manera detallada ahora lo que sigue es crear nuestro corpus. Para esto primero debemos entender los datos y como se describió anteriormente se cuenta con dos *DataFrames* de pandas, el primero contiene el texto tokenizado y lematizado, mientras que el segundo contiene el texto tokenizado y con stemming. Posteriormente el proceso que se realizará es el mismo para ambos *DataFrames*.

1. Comenzamos importanto las librerías necesarias (además de Numpy y Pandas) las cuáles nos serán utiles para el procesamiento del texto, entre estas se encuentran **re** y **gensim**.
2. Posteriormente a nuestro *DataFrame* le aplicamos un pequeño preprocesamiento para obtener nuestros documentos en forma de lista con los tokens (lemmatizados o stemming).

3. El primer modelo **word2vec** [Mik+13] que usaremos será uno entrenado por nosotros con nuestro corpus (el conjunto de todos los documentos), este entrenamiento se realiza con un sistema de redes neuronales usando la función **Word2Vec** de **gensim** [Řeh23]. Para poder entrenar este modelo, ajustamos los siguientes hiperparámetros : *window*=20 (distancia máxima de la palabra objetivo y sus palabras vecinas las cuales afectan al vector de la palabra objetivo), *min_count*=2 (ignora a aquellas palabras que aparezcan con dicha frecuencia en el texto por ser irrelevantes), *workers*=4 (número de threads que se usan al entrenar al modelo), *vector_size*=300 (dimensión en columnas de los vectores resultantes), *sg*=1 (el tipo de algoritmo CBOW o SG) [Yad21]. Haremos una comparativa para ver según el tipo de algoritmo los vectores mejores resultantes (al clasificar).

4. Después de entrenar nuestros modelos (dos por los algoritmos distintos) procedemos a guardar simplemente los vectores de las palabras y eliminamos de memoria el modelo (debido a que ya no se seguirá entrenando con más palabras) y así aumentamos los recursos disponibles.

5. Una vez que ya tenemos estos vectores de cada una de las palabras del corpus procedemos a crear un vector para cada documento el cuál es creado sumando los vectores de cada palabra dentro de dicho documento y dividiendo por el total de palabras. Repitiendo así para cada uno de los documentos dentro del corpus. Esto resultará en tener una matriz de dimensión 50424×300

El segundo vector será obtenido de manera similar al procedimiento anterior, sin embargo en lugar de utilizar un word2vec entrenado con nuestro corpus, usaremos el word2vec de Google, el cuál ya está preentrenado y cuenta con $3,000,000$ de registros de dimensión 300. Nuevamente sumamos cada vector de cada palabra dentro de un documento y dividimos entre el total de palabras.

Finalmente la última representación numérica es la matriz de términos **TF-IDF**, la cuál con la ayuda de la librería *sklearn* creamos nuestra matriz. Esta matriz tiene como columnas a cada una de las palabras distintas en el corpus y como entrada a cada uno de los documentos del mismo. (Una especie de one-hot).

Es así como logramos obtener nuestras representaciones numéricas de nuestro texto y con lo que posteriormente utilizaremos para entrenar a nuestros algoritmos de **Machine Learning**.

Cronograma

TAREAS	FECHA DE INICIO	FECHA DE ENTREGA	% AVANCE
Definir el problema que queremos resolver	21/08/23	25/08/23	100%
Obtener los datos que necesitamos	21/08/23	25/08/23	100%
Propuesta del Proyecto Final	28/08/23	31/08/23	100%
Presentar la Propuesta	05/09/23	05/09/23	100%
Preparar y explorar los datos	11/09/23	17/09/23	100%
Limpieza (stopwords, caractéres especiales)	18/09/23	24/09/23	100%
Stemming y Lemmatizer	25/09/23	30/09/23	100%
Convertir texto en vectores	25/09/23	01/10/23	100%
Comprender los algoritmos (SVM, XGBoost, NN)	02/10/23	22/10/23	0%
Crear algoritmos con datos de entrenamiento y prueba	02/10/23	22/10/23	0%
Fine-tuning de los algoritmos	23/10/23	29/10/23	0%
Comprender los resultados	30/10/23	05/11/23	0%
Escribir reporte final de actividades y resultados y una presentación	06/11/23	17/11/23	0%
Presentación y entrega del proyecto	20/11/24	24/11/23	0%

Bibliography

word2vec [Mik+13] Mikolov, T. et al. *Efficient Estimation of Word Representations in Vector Space*. https://code.google.com/archive/p/word2vec/. Accessed on October 3, 2023. 2013.

gensim_word2vec [Řeh23] Řehůřek, R. *Gensim: models.word2vec – Word2vec embeddings*. https://radimrehurek.com/gensim/models/word2vec.html. Accessed on October 3, 2023. 2023.

shahane2019 [Sha19] Shahane, S. *E-commerce text dataset*. Version 2. 2019. URL: https://doi.org/10.5281/zenodo.3355823.

medium_word2vec [Yad21] Yadav, S. *Word Embedding: Word2Vec with Genism, NLTK, and t-SNE Visualization*. https://medium.com/swlh/word-embedding-word2vec-with-genism-nltk-and-t-sne-visualization-43eae8ab3e2e. Accessed on October 3, 2023. 2021.