# Deep Learning en la práctica: Trabajo final

Aldis Stareczek
Verónica Schiavo

Setiembre de 2023

Objetivo y Alcance

El objetivo principal de este proyecto es aplicar técnicas de procesamiento de lenguaje natural (NLP) y Deep Learning para desarrollar un clasificador de sentimientos que permita analizar y categorizar los comentarios y opiniones.
En nuestro proyecto hemos aplicado dichos conceptos para poder tener un clasificador de comentarios de los clientes de FNC (Fábricas Nacionales de Cervezas) en función de su connotación: positiva, negativa o neutra. Este clasificador se utilizará para mejorar las actividades profesionales relacionadas con la aplicación B2B de FNC y el centro de atención al cliente.

El proyecto se enfoca principalmente en la creación y optimización del modelo de clasificación de sentimientos.
Para la creación del mismo nos hemos basado en el código visto en clase (Análisis de sentimientos - Tweets).

Preprocesamiento de los datos

Contábamos con comentarios de clientes sobre varios aspectos relativos a la compañía de evaluación del servicio brindado. Estos comentarios provienen de dos recolectores de información distintas, con distinto esquema de registro, por lo que hubo que unificarlo en un único listado con: fecha, número de cliente y comentario.

Dado que era esencial para el trabajo, se clasificaron manualmente los distintos comentarios en positivos/negativos/neutros para posteriormente entrenar el modelo. Al registrarse pocos comentarios positivos y neutros, se utilizaron herramientas para generar nuevos comentarios y de esta forma tener proporciones similares de los tres tipos de comentarios.

Modelo aplicado

Los pasos incluidos en nuestro modelo son:

1. Preparación del Ambiente

Para comenzar, configuramos el entorno de trabajo, instalando las bibliotecas necesarias y configurando el entorno de ejecución en Google Colab.

Se utilizaron los siguientes paquetes y bibliotecas:

- Pandas: trabajo con tablas de datos y realizar análisis.
- NumPy: operaciones numéricas y cálculos eficientes.
- NLTK (Natural Language Toolkit): procesamiento de lenguaje natural y análisis de texto.

- WordCloud: visualizador de las palabras más frecuentes en un texto.
- Matplotlib: crear gráficos y visualizaciones.
- Codecs: manejar diferentes codificaciones de caracteres.
- Unidecode: convertir caracteres.
- Scikit-Learn (sklearn): tareas de aprendizaje automático y minería de datos.

2. Preparación de Datos

En esta sección, se explica cómo obtuvimos y preparamos los datos necesarios para entrenar y evaluar el modelo.

Los datos se encuentran subidos en un repositorio de GitHub donde se encontrará además de la tabla con los comentarios, un archivo read me con link al cuaderno del Colab.

Adicionalmente en esta sección luego de llamar al archivo original, filtramos las columnas de interés y separamos aleatoriamente los datos que utilizamos de test y de training.

3. Limpieza de los Datos

Antes de entrenar un modelo, es importante limpiar y preprocesar los datos.

Comenzamos generando una función llamada `corregir_codificacion`, la misma fue creada para corregir comentarios que puedan tener caracteres no legibles o fuera de contexto. Ejemplo: “Siendo cliente por mÃ¡s que sea un almacÃ©n no mayorista a ellos les cambia el precio venden mÃ¡s barata que yo.inclusos algunos comercio pueden venderÃ¡s barato que uno tiene todo al dÃa y hace horas como loco.no es justo”. Luego de pasar esta frase por la función esperamos que la salida sea: “Siendo cliente por más que sea un almacén no mayorista a ellos les cambia el precio venden más barata que yo. Incluso algunos comercio pueden venderás barato que uno tiene todo al día y hace horas como loco. No es justo”. En el caso que la frase no deba ser corregida la salida será la misma sin alteración.

Seguimos por definir la función llamada `eliminar_tildes_y_especiales` que cuenta con la eliminación de tildes y caracteres especiales. Consideramos que esta eliminación nos ayudará a futuro para que todas las palabras repetidas estén lo más similar escritas.

A continuación, generamos la función `clean`, la misma tiene una serie de comandos que seguirá limpiando nuestros comentarios de entrada al modelo.

Se eliminan todas las palabras de 3 o menos caracteres, se pasan todos los caracteres a minúscula y se eliminan las stopwords (set en español).

Finalmente guardamos todas las palabras de cada comentario en vectores con sus correspondientes etiquetas.

4. Visualización de Datos (Wordcloud)

La visualización de datos es útil para comprender mejor el conjunto de datos y poder así entender rápidamente cuales son las palabras claves que definirán el comportamiento de las etiquetas.

5. Clasificador Bayesiano Ingenuo

Implementamos un clasificador de sentimientos utilizando el algoritmo de Clasificador Bayesiano Ingenuo (Naive Bayes). Este es un clasificador probabilístico que se basa en el teorema de Bayes y requiere poca cantidad de datos para entrenar los parámetros. Dado que contábamos con un dataset acotado y no uniforme consideramos que esta metodología era apropiada.

Realizamos distintos pasos que nos permitieron encontrar las palabras claves que al estar contenidas en los comentarios tendrían más probabilidad de ser clasificadas como: positivo, negativo o neutro.

Al final de esta sección utilizamos las funciones de clasificación y entrenamiento. Las mismas, preparan el modelo con el conjunto de datos para entrenar. Se utiliza la función `nltk.NaiveBayesClassifier.train` para aplicar una función de extracción de características a cada elemento en el set de entrenamiento. Esta función de extracción de características convierte cada comentario del conjunto de entrenamiento en un conjunto de características que se utilizarán para el entrenamiento del modelo.

Luego creamos y entrenamos un clasificador de Naive Bayes utilizando el conjunto de entrenamiento que se creó en el paso anterior. Después de entrenar el modelo, el clasificador está listo para realizar predicciones sobre nuevos comentarios y clasificarlos en función de su contenido y características.

6. Resultados

Finalmente, se presentarán los resultados del modelo, incluyendo métricas de rendimiento. Obtuvimos una precisión de 0.87 para los datos del training set y una de 0.81 para los datos de test.

Conclusiones

La precisión obtenida parece un resultado bastante positivo, para las que se plantean algunas consideraciones:

1. Buen Rendimiento en el Conjunto de Entrenamiento: Una precisión del 0.87 en el conjunto de entrenamiento indica que el modelo ha aprendido bastante bien de los datos de entrenamiento. Esto significa que es capaz de clasificar correctamente aproximadamente el 87% de los comentarios que ya ha visto durante el entrenamiento.

2. Una precisión del 0.81 en el conjunto de pruebas es un buen resultado. Indica que el modelo generaliza bien a datos que no ha visto antes. Es capaz de clasificar correctamente aproximadamente el 81% de los comentarios en el conjunto de pruebas, lo que sugiere que es eficaz para predecir sentimientos en nuevos comentarios.

3. La diferencia entre la precisión en el conjunto de entrenamiento y el conjunto de pruebas no es muy grande (0.87 vs. 0.81), lo que sugiere que no hay un sobreajuste significativo.

5. Otras Métricas vistas en el curso: Además de la precisión, podríamos considerar otras métricas de evaluación, como *accuracy*, *presicion*, *exact match, F1 score* para seguir investigando si el modelo ha tenido una buena performance.