UNIVERSIDAD RAFAEL LANDÍVAR

FACULTAD DE INGENIERÍA

INTELIGENCIA ARTIFICIAL

SECCIÓN 1 VESPERTINA

ING.MAX CERNA

PROYECTO: CLASIFICACIÓN USANDO NAIVE BAYES

Julio Anthony Engels Ruiz Coto 1284719

César Adrian Silva Pérez 1184519

Eddie Alejandro Girón Carranza 1307419

GUATEMALA DE LA ASUNCIÓN, ABRIL 23 DE 2024

INTRODUCCIÓN.

El trabajo se sustenta en el dataset Sentiment140, que contiene 1.6 millones de tweets etiquetados automáticamente, y emplea un pipeline de preprocesamiento riguroso para normalizar elementos como URLs, menciones de usuario, hashtags y emojis, transformándolos en tokens semánticamente significativos. Además, se incorporan estrategias avanzadas como la detección de negaciones, preservación de palabras clave emocionales y generación de bigramas, permitiendo al modelo capturar relaciones contextuales críticas. La elección de Naïve Bayes, con suavizado Laplace ("alpha=2.5"), se justifica por su eficiencia computacional y robustez ante grandes volúmenes de datos, logrando un equilibrio entre precisión y velocidad, con métricas de evaluación (F1-score=0.802) que validan su competencia en escenarios reales.

Más allá del aspecto técnico, el proyecto integra un frontend web desarrollado en Django, demostrando cómo modelos de machine learning pueden desplegarse en entornos accesibles para usuarios finales. Esta dualidad precisión algorítmica y usabilidad práctica subraya la relevancia de soluciones escalables en un mundo donde la data textual crece exponencialmente. A través de este enfoque, no solo se contribuye al campo del NLP, sino que también se evidencia cómo algoritmos clásicos, optimizados con preprocesamiento estratégico, pueden rivalizar con arquitecturas más complejas en tareas específicas, ofreciendo un balance entre rendimiento y recursos computacionales.

DEFINICIÓN DEL PROBLEMA Y OBJETIVOS (GENERALES Y ESPECÍFICOS).

General

Desarrollar un sistema de clasificación de tweets positivos o negativos utilizando el algoritmo Naïve Bayes.

Específicos

- Entrenar un modelo de machine learning capaz de clasificar si el texto ingresado contiene sentimientos positivos o negativos.
- Lograr que el modelo posea cierta certeza al momento de clasificar los textos ingresados.
- Desarrollar una interfaz web que permita al usuario ingresar texto, enviarlo al backend y visualizar los resultados de la clasificación de manera clara.

DESCRIPCIÓN DEL DATASET UTILIZADO.

El dataset Sentiment140, es un recurso utilizado en el análisis de sentimientos en redes sociales. Contiene 1.6 millones de tweets recopilados mediante la API de Twitter, etiquetados automáticamente según su polaridad emocional. Su objetivo principal es servir como base para entrenar modelos de Machine Learning capaces de detectar sentimientos (positivos o negativos) en textos cortos e informales, como los que se encuentran en plataformas como Twitter(X).

Contiene 1,600,000 tuits extraídos usando la API de Twitter. Los tuits han sido etiquetados automáticamente con un valor de sentimiento:

- 0 = negativo
- 2 = neutral
- 4 = positivo

Este dataset puede utilizarse para entrenar modelos que detecten sentimientos en textos.
Cada fila del CSV contiene:

- **target**: el sentimiento del tuit (0 = negativo, 2 = neutral, 4 = positivo).
- **ids**: el ID del tuit (por ejemplo: 2087).
- **date**: la fecha en que se publicó el tuit (ejemplo: Sábado 16 de mayo de 2009, 23:58:44 UTC).
- **flag**: la consulta de búsqueda usada (por ejemplo: "lyx"). Si no se usó ninguna, aparece como "NO_QUERY"
- **user**: el nombre de usuario de quien publicó el tuit.
- **text**: el contenido del tuit (ejemplo: "Lyx is cool").

La imagen 1, muestra las gráficas que indican la cantidad de datos correctos en el dataset y contiene los IDs únicos de los tweets.

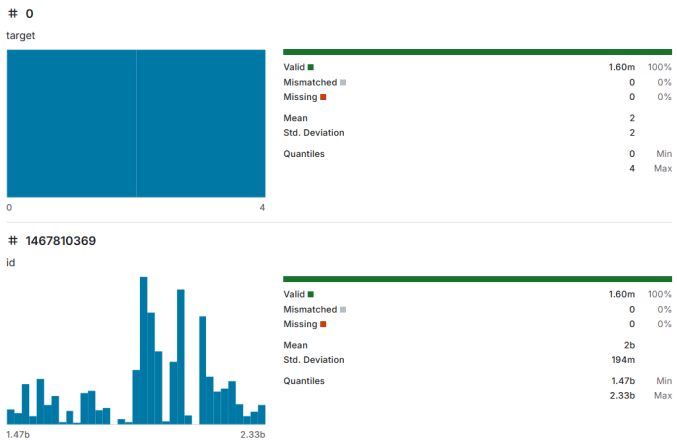

Imagen 1, fuente https://www.kaggle.com/datasets/kazanova/sentiment140

Los datos en la imagen 2, indican las fechas donde se obtuvieron los tweets, los usuarios, y el método de obtención de ellos lo que hace ver que no fue una única consulta.

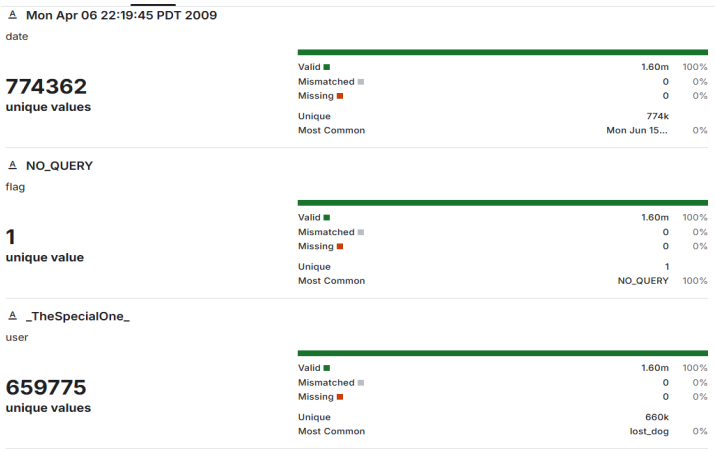

Imagen 2, fuente https://www.kaggle.com/datasets/kazanova/sentiment140

Los datos de la imagen 3, indican que todos los tweets tienen texto (0% missing) y todos están correctamente formateados.

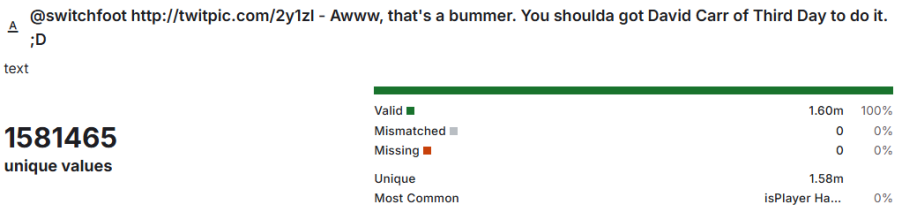

Imagen 3, fuente https://www.kaggle.com/datasets/kazanova/sentiment140

DESCRIPCIÓN DEL PREPROCESAMIENTO APLICADO.

Primero, se convierte todo a minúsculas y se reemplazan patrones "ruidosos" como URLs (http://…), menciones de usuario (@nombre) y hashtags (#palabra) por marcadores tipo URL, USER o HASHTAG_palabra, de modo que el modelo conserve la pista de su existencia sin arrastrar caracteres innecesarios. Luego se transforman emojis felices (:D, :-)) en un token HAPPY_EMOJI y emojis tristes (:(, :-() en SAD_EMOJI, y se convierten secuencias de exclamaciones e interrogaciones en STRONG_EMOTION, EMOTION, STRONG_QUESTION o QUESTION. Y se elimina el resto de la puntuación y normalizan los espacios.

Después se extrae un conjunto de características adicionales (¿hay exclamaciones?, ¿hay mayúsculas?, ¿negaciones?) que luego se añaden como tokens especiales (FEATURE_EXCLAMATION, FEATURE_NEGATION, etc.). Cuando se tokeniza, se corta el texto por espacios y filtran las "stopwords" en inglés —como "and", "the", "in"— excepto aquellas palabras clave de sentimiento (por ejemplo: "love", "hate", "amazing", "worst") las cuales se preservan. Finalmente, se generan bigramas sencillos para capturar frases donde un "no" o "never" modifica la carga emocional.

El resultado es una lista de tokens y tokens-función que representa cada reseña de manera rica y consistente: contiene palabras limpias, indicadores de emoticonos(emojis, signos de exclamación/pregunta), y construcciones de contexto (negaciones, bigramas) que alimentan directamente al Naive Bayes multinomial con suavizado Laplace.

EXPLICACIÓN DEL ALGORITMO NAÏVE BAYES Y JUSTIFICACIÓN.

El Multinomial Naive Bayes es un algoritmo de clasificación que trabaja con probabilidades usando como base el teorema de Bayes. Su funcionamiento se basa en que, para cada texto que analiza y cada categoría posible, calcula qué tan probable es que ese texto pertenezca a dicha categoría. Esto por medio de la siguiente fórmula:

$$P(c \mid d) \propto P(c) \cdot P(d \mid c)$$

– P(c) es la probabilidad a priori de la clase, es decir, cuántos documentos de entrenamiento pertenecen a c dividido por el total.

– P(d | c) es la probabilidad de observar el documento "d" dado que pertenece a "c". Bajo la hipótesis "naive" de independencia condicional de los tokens, esta probabilidad se factoriza como el producto de las probabilidades de cada token "t_i" en "d":

$$P(d \mid c) = \prod_i P(t_i \mid c)$$

En la variante multinomial, se cuentan las veces que aparece cada palabra (token) en todos los documentos de cada clase. Para estimar P(t | c) se usa suavizado Laplace (alpha) y se calcula:

$$P(t \mid c) = (\text{count}(t,c) + \alpha) / (\text{totalTokens}(c) + \alpha \cdot |V|)$$

donde count(t,c) es la frecuencia del token t en todos los documentos de clase c, totalTokens(c) es la suma de frecuencias de todos los tokens en c y |V| Es el tamaño del vocabulario global. El suavizado α evita probabilidades cero para palabras no vistas.
Justificación

- Entrenamiento y predicciones eficientes, aún manejando grandes cantidades de datos.
- Funciona bien aún con un uso diverso de vocabulario, debido a que solo necesita la aparición de palabras clave en el texto.
- Poco demandante computacionalmente, lo que permite un desarrollo en sistemas con recursos limitados.
- Toma en cuenta una pequeña probabilidad de aparición de palabras no vistas en el entrenamiento(desconocidas) por medio del suavizado de Laplace.

EXPLICACIÓN DE LA EVALUACIÓN DEL MODELO (MÉTRICAS UTILIZADAS Y RESULTADOS OBTENIDOS).

Para el entrenamiento del modelo se realizaron varias iteraciones cambiando los valores de alpha en laplace y la cantidad de k-folds en los que se dividiría, entre las tantas pruebas que se realizaron se obtuvieron los datos de precisión, recall y f1-score los cuales indican la precisión del modelo entrenado, en la imagen 4 se pueden ver 4 iteraciones las cuales fueron entre la cantidad realizada las mas significativas ya que el resto de las pruebas de combinaciones realizadas daban el mismo puntaje que las mostradas en la imagen.

alpha	K	
2.5	15	Macro promedios: Precisión: 0.783 Recall: 0.783 F1: 0.783
1.0	5	Macro promedios: Precisión: 0.781 Recall: 0.780 F1: 0.780
2.5	15	Macro promedios: Precisión: 0.798 Recall: 0.798 F1: 0.798
2.5	90	Macro promedios: Precisión: 0.802 Recall: 0.802 F1: 0.802

Imagen 4, fuente propia

De los datos analizados la que mejor puntaje brindó fue alpha = 2.5 y k = 90, con un puntaje en las 3 métricas de 0.802, en la imagen 5 podemos apreciar la matriz de confusión de dicho entrenamiento la cual nos indica que de los datos de entrenamiento acerto 635603 datos correctamente pero fallo 164397 entre los datos para su clasificación.

```
Macro promedios:
Precisión: 0.802
Recall: 0.802
F1: 0.802

Matriz de Confusión Global:

Matriz de Confusión:
                neg     pos
          ____________________
     neg  642276  157724
     pos  164397  635603
array([[642276, 157724],
       [164397, 635603]])
```

Imagen 5, fuente propia

DIAGRAMAS:

○ ARQUITECTURA DE LA SOLUCIÓN (MOTOR DE INFERENCIA + PÁGINA WEB).

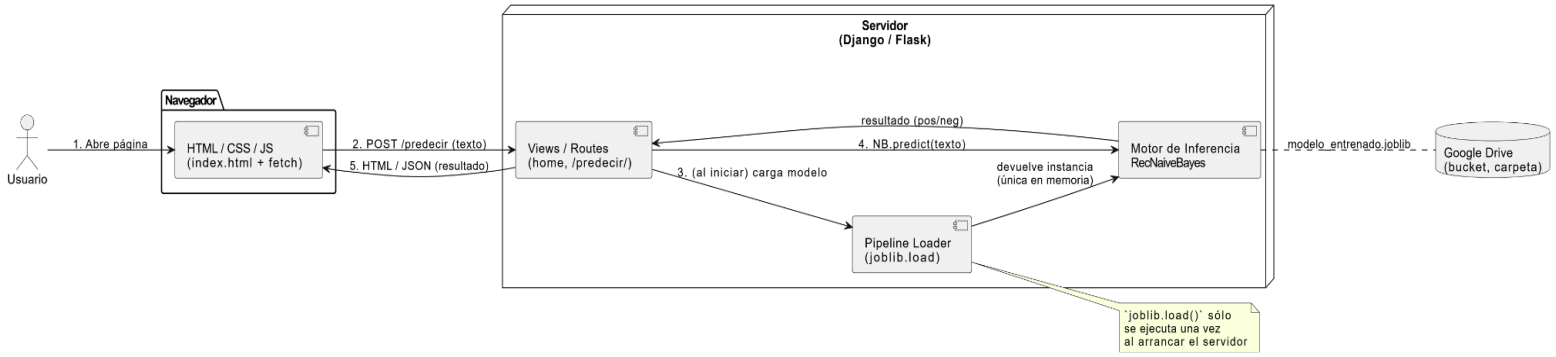

Imagen 6, fuente propia

○ DIAGRAMAS DE CASOS DE USO

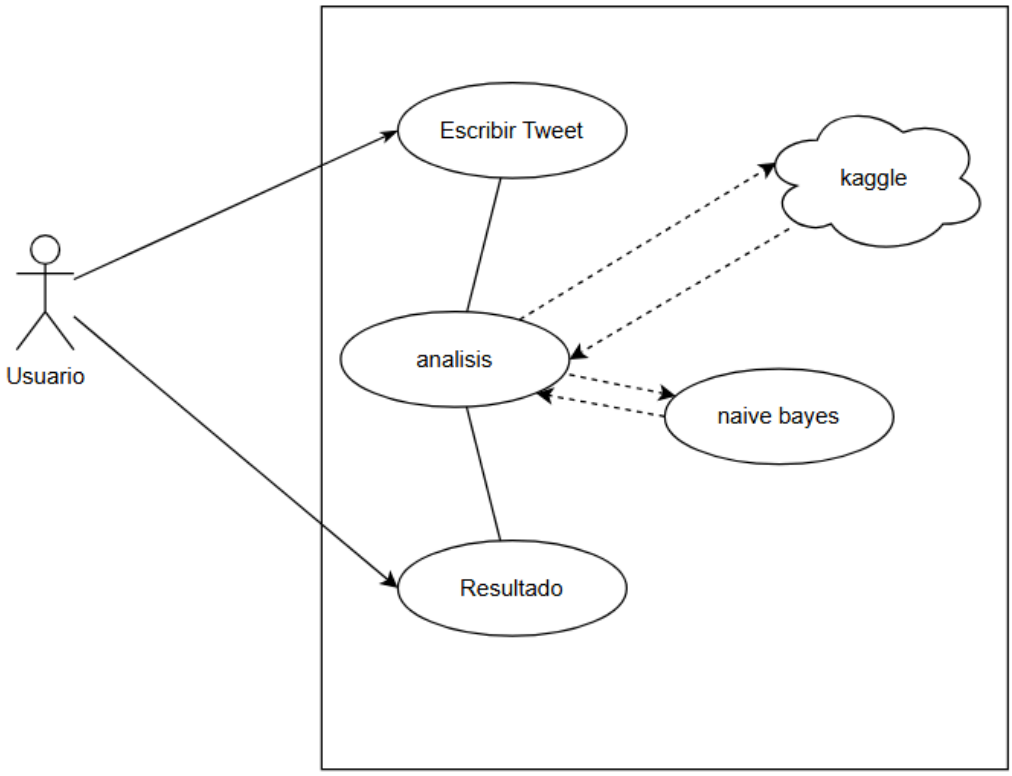

Imagen 7, fuente propia

○ DIAGRAMA DE FLUJO GENERAL

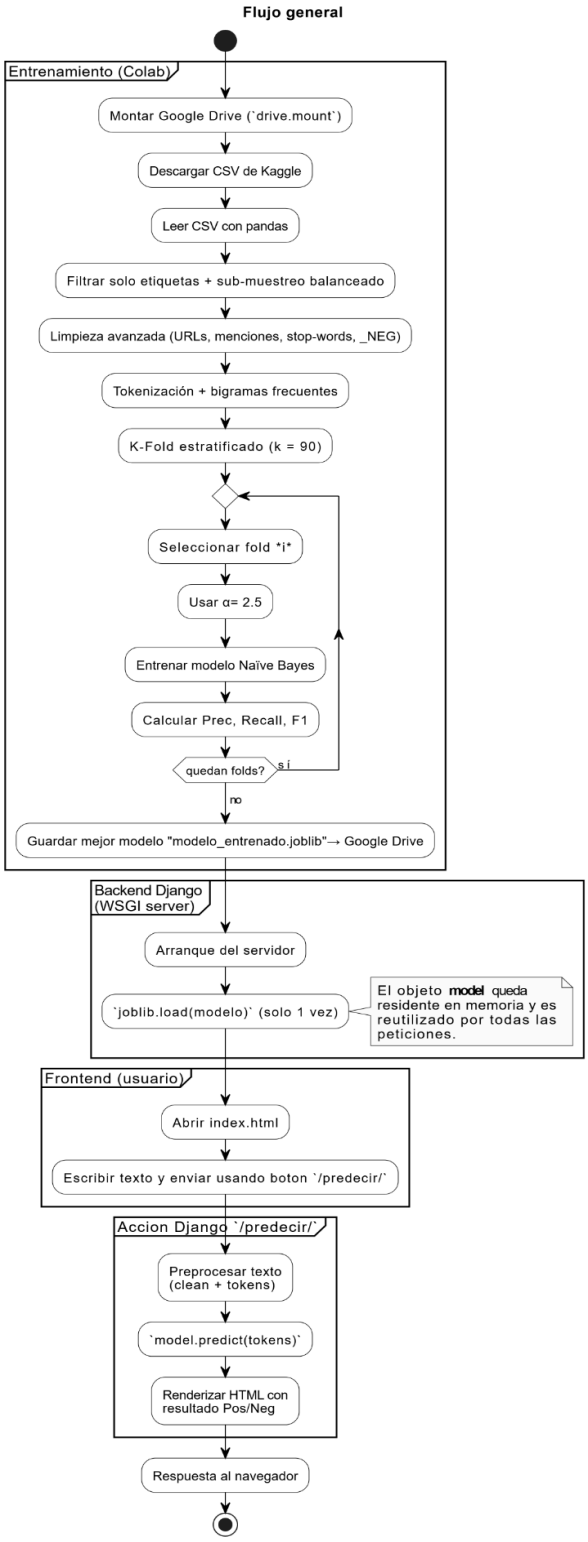

Imagen 8, fuente propia

○ DIAGRAMA DE COMPONENTES

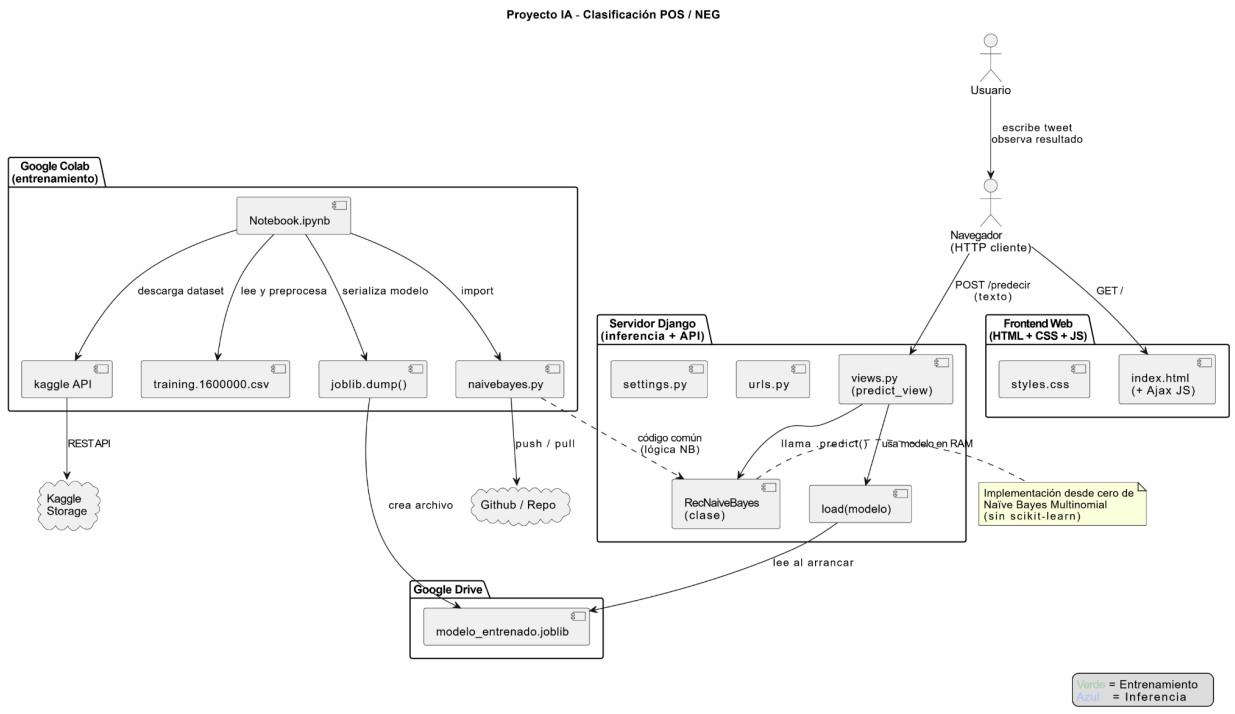

Imagen 9, fuente propia

○ DIAGRAMA DE SECUENCIAS (MODELAR LA INTERACCIÓN ENTRE EL USUARIO, EL FRONTEND Y EL BACKEND)

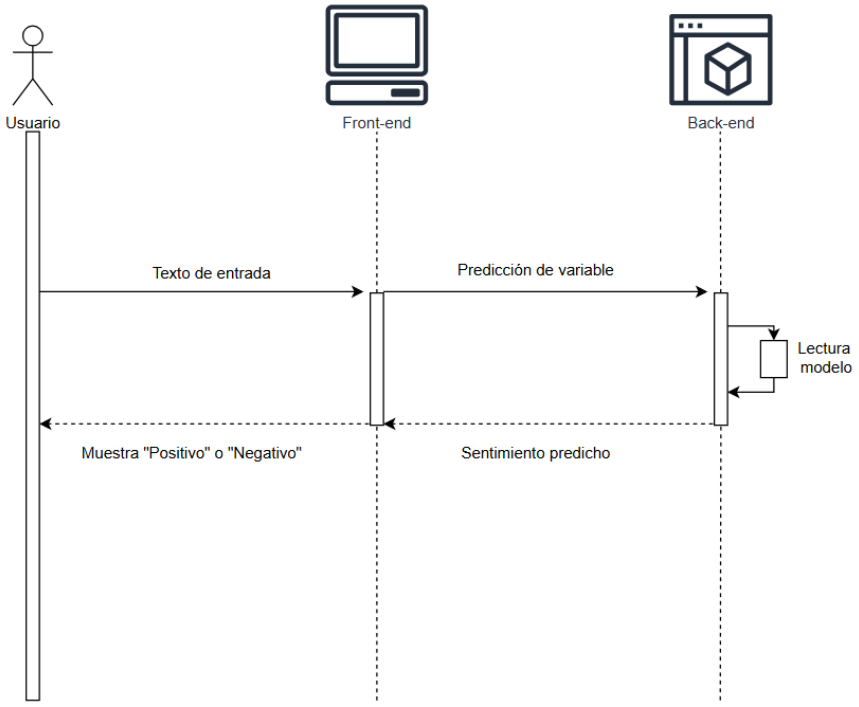

Imagen 10, fuente propia

EVIDENCIAS DE FUNCIONAMIENTO (CAPTURAS DE PANTALLA DE LA WEB APP Y EL MODELO EN ACCIÓN).

Aplicación web y modelo funcionando para clasificar texto positivo

Imagen 11, fuente propia

Aplicación web y modelo funcionando para clasificar texto negativo

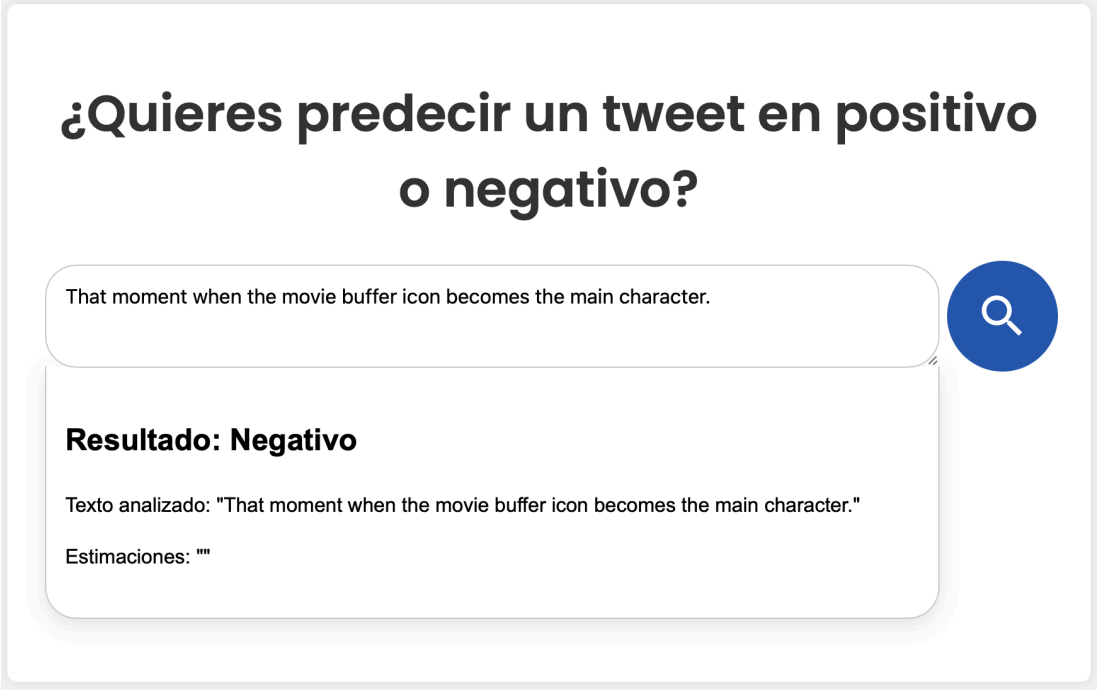

Imagen 12, fuente propia

CONCLUSIONES Y APRENDIZAJES.

- El algoritmo Naïve Bayes demostró ser efectivo para clasificación de sentimientos, alcanzando un F1-score de 0.802, lo que valida su uso en problemas de texto con grandes volúmenes de datos.

- La normalización de URLs, emojis, hashtags y signos de puntuación, junto con la preservación de palabras clave de sentimiento, fue fundamental para mejorar la calidad de los datos de entrada y el rendimiento del modelo.

- Evaluar con precisión, recall y F1-score permitió una comprensión integral del modelo, evitando conclusiones erróneas basadas solo en una métrica.

- El desarrollo de un frontend y backend con Django demostró cómo llevar modelos de machine learning a entornos reales, lo que facilitó la interacción del usuario con el modelo.

- La construcción manual de la matriz de confusión y el cálculo de métricas, sin recurrir a frameworks especializados, introdujo desafíos en la estandarización y validación cruzada de resultados. Esto deja claro por qué necesitamos métodos sólidos, así se asegura que todo sea reproducible y se evitan sesgos, sobre todo cuando se tienen limitaciones técnicas.

- Las métricas finales (Precisión=0.802, Recall=0.802, F1=0.802) indican un equilibrio robusto entre la capacidad de identificar casos relevantes y la exactitud en las predicciones.

- Incluir bigramas y características como “FEATURE_NEGATION” ayudó a capturar relaciones entre palabras (ej: "no bueno"), mejorando la interpretación del sentimiento.

- Utilizar los recursos gratuitos de Google Colab permitió entrenar y validar el modelo sin invertir en infraestructura propia, garantizando reproducibilidad gracias a notebooks versionados en la nube. Además, al montar Drive centralizamos datasets y binarios, evitando copias locales y reduciendo riesgo de extravío. El entorno efímero obliga a automatizar descargas y carga de modelos, fortaleciendo la trazabilidad del proceso. Todo el flujo queda documentado paso a paso, dificultando el plagio porque refleja claramente cada decisión de procesamiento y evaluación. Así, se optimiza costo, transparencia y originalidad en un mismo ecosistema.