Inteligencia de Negocios

Juan Nicolás García Esquibel

201717860

Entrega 2 del proyecto

1. Introducción

La etapa actual consiste en la implementación y justificación de una aplicación, en este caso, se decidió por optar una solución web, cuyo objetivo es entrenar y actualizar de manera incremental un modelo de detección de noticias falsas, por lo cual se ofrece un API y una interfaz para interactuar con el modelo.

2. Diseño de la aplicación

Al pensar en los posibles casos de uso reales, una aplicación web aplica más en tales contextos, porque si se va a publicar un artículo es mucho más probable que se esté escribiendo en un computador que en un dispositivo móvil y aún si fuera el segundo mencionado, la solución también estaría al alcance.

Backend

Al ser un framework de Python, FastAPI permite integrar de forma sencilla técnicas de machine learning, además de que se destaca por su excelente rendimiento y capacidad asíncrona, lo que garantiza respuestas rápidas y escalables. Su documentación automática mediante Swagger facilita enormemente el testing y la interacción con la API.

Frontend

En cuanto al frontend, React fue seleccionado por su versatilidad y robustez, sumado a mi experiencia personal con él, lo que garantiza un desarrollo ágil y una excelente experiencia de usuario. React facilita la creación de interfaces interactivas y dinámicas, provee maneras de integración con APIs REST y cuenta con una arquitectura modular que mejora el mantenimiento y la escalabilidad de la aplicación. Además, su amplia comunidad y la disponibilidad de numerosas librerías complementarias hacen de esta una opción muy acertada para construir un frontend moderno y efectivo.

Y, por último, esta alternativa pone a disposición del desarrollador maneras de hacer que el diseño se ajuste a diferentes dispositivos como se dijo en el inicio.

3. Descripción del Back-End

En cuanto a lo que concierne al pipeline se conserva gran parte de la implementación de la Etapa 1, ya que se sigue utilizando el proceso de reemplazar marcadores de números, eliminar caracteres no ASCII, convertir a minúsculas, se remueve la puntuación, omisión de stopwords en español y lematización el texto con spaCy para reducir la variabilidad morfológica. Sin embargo, en esta versión se opta por el HashingVectorizer en lugar del TF-IDF, ya que su carácter stateless permite transformar el texto en vectores numéricos de dimensión fija de manera directa, lo que hace posible el entrenamiento incremental. Finalmente, se utiliza un SGDClassifier, que admite actualizaciones incrementales a través de partial_fit y que el modelo se actualice de forma continua conforme se incorporan nuevos datos.

Al llegar a este punto, surge la pregunta de por qué SGDClassifier y no los algoritmos implementados en la primera etapa y la razon es la decisión del entrenamiento incremental, sobre el cual se darán detalles más adelante, ya que los otros requieren un reentrenamiento desde cero en cada llamado del endpoint.

4. Descripción del Front-End

Su estructura está dividida en 3 secciones, lo cual corresponde con el número de endpoints:

- **Predicción**
  Permite al usuario ingresar un texto, o varios, y obtener la predicción del modelo con sus respectivas probabilidades. Como alternativa, se deja la opción de subir un archivo JSON.

- **Reentrenamiento**
  Permite subir un archivo CSV para actualizar el modelo. Luego del proceso, se muestran las métricas: precision, recall y F1, después del reentrenamiento.

- **Reinicio del modelo**
  Un botón que llama al endpoint /reset para eliminar el modelo actual y comenzar de nuevo. A pesar de que no se pide en el enunciado, me pareció útil durante las pruebas y podría tambien serlo para el evaluador.

5. Usuario final e importancia de la aplicación

Esta aplicación resulta útil, por ejemplo, para administradores de redes sociales en empresas que desean filtrar y verificar la información compartida en sus canales. Al clasificar rápidamente las noticias como potencialmente falsas o reales y asignarles una probabilidad, la herramienta permite que los usuarios que revisan diariamente grandes volúmenes de contenido puedan determinar de forma ágil si aprueban la publicación, solicitan una verificación adicional o rechazan la noticia.

Adicionalmente, "Fake News Detector" agiliza la toma de decisiones en cuanto a la confiabilidad del contenido, reduciendo el riesgo de difundir información errónea que pueda dañar la credibilidad de la organización. Con ello, se ayuda a mantener altos estándares de calidad, de tal manera que la reputación de la empresa es protegida y evidencia una optimización de la gestión del contenido en entornos donde la veracidad es fundamental.

6. Riesgos del uso

Aunque la herramienta resulta muy útil, el modelo no es perfecto. Incluso después de varios reentrenamientos, puede arrojar falsos positivos o negativos, es decir, en ocasiones puede marcar como falsa una noticia genuina o, al contrario, dar por verdadera una que es engañosa. Por eso, es esencial que los usuarios no se fíen del todo de sus resultados y se apoyen de una revisión manual. Esta verificación por parte de una persona, sumada a reentrenamientos frecuentes con datos nuevos, contribuye a reducir los errores, aunque nunca llega a eliminarlos por completo.

Por otro lado, es fundamental tener en cuenta la privacidad y la seguridad de la información, ya que los textos analizados pueden contener datos sensibles. Por ello, es crucial cifrar la comunicación mediante protocolos y gestionar los datos utilizados en el entrenamiento de manera responsable. Asimismo, conforme el modelo se vuelve más complejo y crece en capacidad, probablemente requiera mayores recursos

computacionales para su entrenamiento y despliegue. Esto implica contar con planes de contingencia, sistemas de respaldo y entornos alternativos que aseguren la continuidad del servicio.

7. Recursos y despliegue para la aplicación

En primer lugar, para el entrenamiento y ejecución del modelo se requiere un servidor que ejecute Python, FastAPI y las librerías de machine learning. En función del volumen de datos y la frecuencia de reentrenamiento, un servidor virtual con 2 a 4 núcleos de CPU y entre 4 y 8 GB de RAM podría ser suficiente; sin embargo, si se espera un alto flujo de noticias en tiempo real, será necesario disponer de más recursos o incluso implementar un despliegue escalable en la nube, como podría ser AWS, Azure o GCP.

El modelo se persiste en un archivo binario mediante joblib, alojado en el mismo servidor de la API, por lo que es fundamental contar con un sistema de respaldo para evitar la pérdida del modelo entrenado o de sus actualizaciones. La aplicación web, desarrollada en React, puede alojarse en el mismo servidor o en un hosting especializado, asegurándose de configurar adecuadamente CORS en FastAPI para permitir la comunicación sin restricciones. Además, la organización puede optar por un contenedor Docker que incluya tanto la API como la aplicación web, facilitando así la portabilidad y escalabilidad.

La integración con los procesos de la organización se haría incluyendo la herramienta dentro del flujo editorial: cuando un redactor o editor revisa un artículo, este se procesa con el “Fake News Detector” antes de publicarlo. En el caso de redes sociales corporativas, la aplicación se puede utilizar como paso previo para programar o autorizar publicaciones, ayudando a identificar si la fuente es potencialmente falsa y, de este modo, contribuyendo a mantener la calidad y credibilidad de la información difundida.

8. Comentario Final

En conclusión, si bien Fake News Detector constituye un avance notable en la automatización del proceso editorial y en la detección de noticias falsas, es importante reconocer que los sesgos son inherentes a cualquier modelo entrenado. Aunque no se alcanzó a profundizar en este aspecto durante el video, es fundamental que el entrenamiento y reentrenamiento se realicen con datos extraídos de una amplia diversidad de fuentes, regiones e ideologías, para mitigar el impacto de dichos sesgos y garantizar que el sistema sea lo más equitativo y representativo posible.

Bibliografía

Universidad de los Andes. Desinformación, la reina en medio de la pandemia. Recuperado de https://www.uniandes.edu.co/es/noticias/periodismo-y-comunicaciones/desinformacion-la-reina-en-medio-de-la-pandemia

Niño, L. (2024, 28 de diciembre). Ni el presidente se salva, estas son las veces que Petro ha publicado noticias falsas en sus redes sociales. Infobae. Recuperado de https://www.infobae.com/colombia/2024/12/28/ni-el-presidente-de-salva-estas-son-las-veces-que-petro-ha-publicado-noticias-falsas-en-sus-redes-sociales/

Hernández Bonilla, J. M. (2024, 19 de enero). El aumento de las noticias falsas en los medios colombianos mina su credibilidad y destruye la confianza de las audiencias. El País. Recuperado de https://elpais.com/america-colombia/2024-01-19/el-aumento-de-las-noticias-falsas-en-los-medios-colombianos-mina-su-credibilidad-y-destruye-la-confianza-de-las-audiencias.html