INTELIGENCIA DE NEGOCIOS - ISIS 3304

"Proyecto 1 Analítica de textos"

Profesora: Haydemar Núñez

Nicolas Rozo Fajardo - n.rozo
Sara Benavides Mora - s.benavidesm
Laura Calderón Molina - l.calderonm

Índice

Introducción

El presente proyecto tiene como objetivo automatizar el proceso de preparación y análisis de textos mediante la construcción de un pipeline eficiente y replicable, que permita clasificar textos en base a los Objetivos de Desarrollo Sostenible (ODS). Para lograr esto, se desarrolló un modelo basado en aprendizaje automático, el cual clasifica textos en las categorías ODS 3, 4 o 5, relacionadas con la salud, educación e igualdad de género. Este modelo fue integrado en una aplicación web interactiva que permite a los usuarios, como analistas de datos de instituciones como el Fondo de Poblaciones de las Naciones Unidas (UNFPA), acceder fácilmente a la predicción y a la probabilidad de la clasificación. Además, la aplicación incluye funcionalidades de reentrenamiento del modelo para adaptarse a nuevas tendencias y datos. Este desarrollo busca optimizar la toma de decisiones en temas críticos para la sociedad, automatizando procesos que tradicionalmente requieren análisis manual de grandes volúmenes de datos textuales.

Proceso de automatización del proceso de preparación de datos

El proceso de automatización fue diseñado para crear un pipeline replicable que integre las distintas etapas de preparación de los datos, el entrenamiento del modelo, y la persistencia del mismo. Este pipeline permite que el flujo de procesamiento de datos sea estandarizado y eficiente, de modo que cualquier nueva instancia de datos pueda pasar por las mismas etapas de preprocesamiento, vectorización y predicción o reentrenamiento sin necesidad de intervención manual.

1.1. Limpieza de Datos y Preprocesamiento

La primera parte del pipeline se encarga de la preparación de los datos de texto en español. Para este proceso, se empleó la clase Cleaning que automatiza varias operaciones clave:

- **Corrección de codificación:** Dado que los datos pueden contener caracteres mal codificados (como resultado de distintas fuentes de datos), se creó un mecanismo para corregir estos caracteres y asegurar que el texto se procese correctamente.
- **Normalización:** El texto es convertido a minúsculas y se eliminan acentos para asegurar que palabras como "educación" y "educacion" sean tratadas de manera uniforme.
- **Eliminación de caracteres especiales:** Se eliminan signos de puntuación y otros caracteres irrelevantes para el análisis.
- **Eliminación de palabras:** Las stopwords o palabras vacías (como "el", "de", "la") son eliminadas debido a su poco aporte semántico al modelo y su capacidad de sesgar la importancia de otras palabras relevantes. De igual forma, los verbos son lematizados para evitar aumentar la cardinalidad del modelo innecesariamente.7
- **Eliminación de rutas y enlaces:** Finalmente, se eliminan todos aquellos enlaces o rutas correspondientes tanto a páginas web como ubicaciones de archivos debido a su irrelevancia semántica y capacidad de afectar el nivel de predicción del modelo.

Este paso es fundamental porque garantiza que los datos sean consistentes y relevantes antes de ser procesados por el modelo de machine learning. Además, el proceso es automatizado, lo que significa que cada nueva instancia de datos pasará por estas mismas transformaciones de manera uniforme.

1.2. Vectorización de Textos

La segunda parte del pipeline se encarga de la vectorización, es decir, de transformar el texto en una representación numérica que pueda ser utilizada por el modelo. Se empleó un modelo TF-IDF para ponderar las palabras según su frecuencia en el corpus de datos. Esto permite que el modelo de clasificación identifique cuáles palabras son más relevantes para determinar la clase de ODS a la que pertenece cada texto.

- **Impacto y ponderación de palabras:** El vectorizador TF-IDF asigna un peso mayor a las palabras que son más específicas a ciertos textos, y un peso menor a las que aparecen con frecuencia en muchos textos, lo que es esencial para clasificar correctamente.

Este proceso es crucial para la calidad del modelo, ya que una buena vectorización asegura que el modelo tenga una representación precisa del contenido semántico de los textos y, por ende, se pueda llevar a cabo una buena tarea predictiva.

1.3. Persistencia del Modelo

Una vez que el modelo de clasificación ha sido entrenado, se guarda la totalidad del pipeline haciendo uso de un archivo binario con extensión .joblib. Esto permite que tanto los pasos de preparación de datos como el modelo puedan ser reutilizados sin necesidad de tener que ajustarlos cada vez que se usa. Este archivo se almacena en la carpeta assets, lo que facilita su acceso por parte de la API. A su vez, cabe resaltar que la persistencia de los procesos de preparación y del modelo garantiza que se pueda construir una aplicación escalable la cual no tiene la necesidad de recalcular o reconstruir el modelo desde cero cada vez que se recibe una solicitud de predicción.

Construcción del Modelo

El modelo utilizado en este proyecto es un Random Forest Classifier, que es adecuado para manejar grandes volúmenes de datos y realizar predicciones robustas en contextos no lineales. El RandomForest fue elegido por su capacidad para reducir el riesgo de sobreajuste y proporcionar predicciones más estables.

2.1. Entrenamiento del Modelo

Durante el proceso de entrenamiento, los datos de texto vectorizados son utilizados para ajustar el modelo. Este proceso implica dividir los datos en conjuntos de entrenamiento y prueba para validar el rendimiento del modelo.

- **División de datos:** Se utilizó un 80% de los datos para el entrenamiento y un 20% para la prueba. Este enfoque garantiza que el modelo sea capaz de generalizar sus predicciones a datos que no ha visto antes.

- **Métricas de evaluación:** El modelo fue evaluado utilizando métricas de clasificación estándar como precisión, recall y F1-score. Estas métricas proporcionan una visión completa de cómo el modelo está manejando la clasificación de textos en cada una de las clases de ODS.

- **Uso de hiperparámetros:** Para el entrenamiento y reentrenamiento del modelo se usaron los hiper parámetros hallados mediante validación cruzada en la Fase 1 del proyecto. Se decidió no buscar nuevos hiperparámetros ya que se tiene certeza que los encontrados funcionan correctamente y, al mismo tiempo, se reduce el tiempo necesario para el proceso de entrenamiento.

Este proceso asegura que el modelo esté bien ajustado a los datos y sea capaz de realizar predicciones con una alta precisión y confiabilidad.

2.2. Reentrenamiento del Modelo

El proceso de reentrenamiento es fundamental para asegurar que el modelo pueda adaptarse a cambios en los datos y mantener su precisión a lo largo del tiempo. El sistema actual implementa una estrategia en la que se combinan los datos nuevos con los datos originales, pero existen otras alternativas que podrían considerarse. A continuación, se presentan tres enfoques diferentes para el reentrenamiento del modelo, cada uno con sus ventajas y desventajas.

- **Reentrenamiento combinando nuevos datos con los datos originales**
    - En esta opción, el modelo es reentrenado utilizando tanto los datos nuevos como los datos originales. Todos los datos anteriores se concatenan con los nuevos y se utiliza este conjunto combinado para volver a entrenar el modelo.
    - **Ventaja**: Este enfoque asegura que el modelo conserve toda la información histórica. Los patrones que fueron identificados en los datos originales no se pierden, lo que permite que el modelo tenga una visión más completa de los datos a lo largo del tiempo.
    - **Desventaja**: A medida que el conjunto de datos crece, el tiempo y los recursos computacionales necesarios para reentrenar el modelo aumentan significativamente. Además, si los datos originales incluyen información que ya no es relevante, podrían afectar negativamente la capacidad del modelo para adaptarse a nuevas tendencias.
- **Reentrenamiento solo con los nuevos datos**
    - En este enfoque, el modelo es reentrenado utilizando únicamente los nuevos datos proporcionados, sin incluir los datos originales. Esto implica que el modelo se entrena solo con la información más reciente disponible.
    - **Ventaja**: Este método es computacionalmente más eficiente, ya que solo se entrena con los nuevos datos. Además, permite que el modelo se ajuste

rápidamente a cambios recientes en los patrones de datos sin la interferencia de información desactualizada.

    - **Desventaja**: Al no incluir los datos históricos, el modelo podría perder patrones importantes que podrían seguir siendo relevantes. Esto puede llevar a un sobreajuste a los datos recientes, lo que podría afectar negativamente su capacidad de generalización en otros contextos.

- **Reentrenamiento con una combinación de nuevos datos y un porcentaje de los datos originales**

    - Este enfoque implica reentrenar el modelo utilizando los nuevos datos junto con un subconjunto de los datos originales, por ejemplo, el 20% de los datos antiguos. Este subconjunto puede seleccionarse de manera aleatoria o utilizando algún criterio basado en la relevancia o antigüedad de los datos.

    - **Ventaja**: Al utilizar tanto nuevos datos como una fracción de los datos originales, se logra un equilibrio entre mantener la información histórica y adaptarse a los nuevos patrones. Esto reduce el riesgo de que el modelo pierda patrones importantes mientras permite que se adapte a los cambios recientes.

    - **Desventaja**: Aunque es más eficiente que utilizar todos los datos originales, sigue requiriendo más recursos computacionales que entrenar solo con los nuevos datos. Además, la selección del subconjunto de datos originales puede ser compleja y podría requerir ajustes para optimizar el rendimiento del modelo.

Construcción a través de la API Rest

La API REST fue construida utilizando el framework **FastAPI**, que proporciona una manera eficiente y rápida de manejar solicitudes HTTP. La API está compuesta por cinco endpoints:

3.1. Endpoint de Predicción por Texto

Este endpoint recibe un JSON que contiene una única instancia de texto. El objetivo de este endpoint es procesar el texto mediante el pipeline que limpia, vectoriza y utiliza el modelo entrenado para generar una predicción sobre la clase de ODS (3, 4 o 5) a la que pertenece el texto ingresado, devolviendo un JSON tanto con la predicción realizada como con la probabilidad asociada. Este proceso permite que los usuarios interactúen directamente con el modelo a través de una interfaz sencilla y puedan procesar de forma casi que inmediata el texto de su elección.

3.2. Endpoint de Predicción desde Archivo

Este segundo endpoint está diseñado para manejar archivos CSV o Excel que contengan múltiples instancias de texto. Los usuarios pueden cargar un archivo que contenga una gran cantidad de textos, los cuales son procesados de manera automática. El pipeline se encarga de limpiar y vectorizar los textos, y el modelo de clasificación asigna una clase de

ODS para cada fila del archivo. El resultado es un archivo con la misma cardinalidad de registros el cual incluye la predicción y puntaje asociado a cada instancia. Cabe resaltar que para facilitar su visualización y entendimiento el archivo puede ser descargado.

3.3. Endpoint de Reentrenamiento del Modelo

El tercer endpoint permite el reentrenamiento del modelo de clasificación. Recibe un archivo CSV o Excel que contiene textos adicionales y la clase ODS real (variable objetivo) para cada texto. Este endpoint utiliza estos nuevos datos para reentrenar el modelo, permitiendo que este se ajuste a nuevas tendencias o patrones de lenguaje en los textos. Tras el proceso de reentrenamiento, el modelo actualizado reemplaza la versión anterior y se guardan las nuevas métricas de desempeño (precisión, recall, F1-score), las cuales se devuelven en la respuesta. Esto asegura que el modelo se mantenga preciso y relevante.

3.4. Endpoint para Visualización de Palabras Relevantes

Este endpoint es una funcionalidad extra que proporciona la visualización de las palabras más relevantes asociadas a cada una de las clases ODS (3, 4 y 5). El endpoint recibe el identificador de la clase ODS (3, 4 o 5) y devuelve un conjunto de palabras clave con su importancia o peso en el modelo de clasificación. La respuesta es utilizada en la aplicación web para generar una nube de palabras, lo que ayuda a los analistas a entender qué términos son más influyentes en la clasificación de textos para cada ODS.

3.5. Endpoint de Reporte de Métricas

Este endpoint realiza una solicitud GET para devolver las métricas actuales del modelo, como precisión, recall y F1-score. Los usuarios pueden consultar este endpoint para conocer el rendimiento del modelo en tiempo real, lo que es crucial para evaluar si el modelo necesita ser ajustado o reentrenado con nuevos datos. Estas métricas ofrecen una visión general del comportamiento del modelo y sirven como un indicador de su efectividad.

Desarrollo de la aplicación

4.1. Contexto y justificación

La aplicación que se desarrolló para este proyecto está centrada en facilitar el análisis de textos mediante un modelo de clasificación que asigna cada texto a uno de los Objetivos de Desarrollo Sostenible (ODS) 3, 4, o 5. Esta aplicación está pensada para ser utilizada por analistas de datos del Fondo de Poblaciones de las Naciones Unidas (UNFPA) y otros actores relevantes en la toma de decisiones dentro del contexto de políticas públicas relacionadas con estos ODS.

El problema principal identificado en la Etapa 1 del proyecto es la alta demanda de recursos necesarios para analizar de manera manual las opiniones de los ciudadanos. Los analistas enfrentan el desafío de categorizar y analizar grandes volúmenes de información textual para alinearlas con los ODS específicos. La oportunidad radica en automatizar el proceso, lo que permite mejorar la eficiencia en la toma de decisiones y la asignación de recursos al reducir la carga de trabajo humana y el tiempo requerido para analizar cada texto manualmente.

La implementación de una herramienta automatizada, basada en aprendizaje supervisado, permite a los analistas de datos reducir considerablemente el tiempo de procesamiento, enfocarse en el análisis de los resultados y tomar decisiones informadas. Además, la posibilidad de reentrenar el modelo con nuevos datos clasificados por expertos asegura que el sistema pueda mejorar con el tiempo, manteniéndose actualizado y relevante frente a nuevas tendencias y cambios en el lenguaje utilizado en los textos.

El impacto que esta aplicación puede tener en la organización es significativo, ya que agiliza la identificación de áreas críticas que requieren atención inmediata en temas como salud, educación e igualdad de género, optimizando el uso de los recursos humanos y tecnológicos.

4.2. Conexión entre la aplicación y el proceso de negocio

La aplicación desarrollada está estrechamente conectada con el proceso de negocio del UNFPA en su objetivo de avanzar en los ODS 3, 4, y 5. Los analistas de datos del UNFPA pueden utilizar esta herramienta para procesar grandes volúmenes de datos textuales y generar insights rápidos y precisos sobre las preocupaciones de los ciudadanos. Esta información procesada permite identificar áreas prioritarias para la implementación de políticas públicas, lo que optimiza tanto el tiempo como el esfuerzo dedicado a la toma de decisiones.

El hecho de que la aplicación pueda reentrenar el modelo con nuevos datos clasificados le permite adaptarse a las variaciones y tendencias en las opiniones públicas, manteniendo así una alta precisión en la clasificación de los textos. Esta capacidad garantiza que las intervenciones del UNFPA sean más precisas y adecuadas a la realidad del contexto en el que operan.

4.3. Desarrollo de la aplicación web

La aplicación web se desarrolló utilizando React para el frontend y FastAPI para el backend. La implementación incluyó el desarrollo de cuatro características principales que permiten la interacción con el modelo de clasificación a través de archivos CSV y entradas de texto manuales. Todas estas características pueden ser accedidas por el usuario mediante la barra de navegación.

- ➔ **Predicción de texto:** El usuario puede ingresar un texto manualmente, y el sistema lo clasifica en una de las categorías ODS 3, 4, o 5. Adicionalmente, se le proporciona la probabilidad asociada a esa predicción, lo cual es importante para entender la confianza del modelo en sus resultados. Este proceso está automatizado y los analistas de datos pueden interactuar directamente con el modelo sin necesidad de conocimientos técnicos avanzados. El frontend implementa una interfaz clara y simple para la interacción del usuario, utilizando formularios para ingresar el texto y recibir la predicción y probabilidad.

- ➔ **Predicción por archivo:** Además del ingreso manual de textos, la aplicación soporta la carga de archivos en formato CSV o XLSX. Esto es útil para analizar grandes volúmenes de datos simultáneamente. El usuario carga un archivo, y el sistema procesa cada línea de texto, clasificándola y generando un archivo de salida

con las predicciones correspondientes. Esto agiliza el análisis de grandes cantidades de información de manera eficiente y facilita las labores que el usuario debe realizar para llevar a cabo estas predicciones.

- ➔ **Reentrenamiento del modelo:** En el contexto de un uso continuo de la herramienta, se ofrece la capacidad de reentrenar el modelo con datos nuevos. Esta funcionalidad es clave para mantener la precisión y relevancia del modelo, y se activa mediante la carga de nuevos datos etiquetados. Este endpoint devuelve métricas como el F1-score, la precisión y el recall, que permiten a los analistas evaluar el rendimiento actualizado del modelo.

- ➔ **Palabras relevantes:** Adicionalmente, la aplicación permite visualizar las palabras más importantes asociadas a cada ODS a través de un gráfico interactivo de nubes de palabras. Esta funcionalidad aporta un valor adicional al negocio, ya que ayuda a identificar términos clave en los textos que pueden influir en la clasificación.

4.4. Justificación de la Implementación

La justificación técnica detrás de la arquitectura de la aplicación se basa en su modularidad y escalabilidad. Se eligió FastAPI como framework backend por su rapidez y capacidad de manejar múltiples peticiones simultáneas de manera eficiente, lo cual es crucial para una aplicación orientada al procesamiento de grandes volúmenes de datos. La separación entre el frontend y el backend asegura que las funcionalidades se puedan modificar o mejorar sin impactar en toda la estructura del sistema.

El uso de React en el frontend garantiza una experiencia de usuario fluida y dinámica. La posibilidad de manejar múltiples rutas y componentes permite que el usuario navegue fácilmente entre las distintas funcionalidades de la aplicación. Además, el uso de bibliotecas como React-Bootstrap asegura que la interfaz sea intuitiva y estéticamente agradable.

Finalmente, la API y el pipeline de procesamiento de datos permiten una integración fluida con los sistemas existentes en el UNFPA y cualquier otra entidad que desee usar la herramienta. Esto facilita la escalabilidad del sistema y su implementación en diferentes entornos organizacionales.

Conclusiones

El desarrollo de este proyecto permitió crear una herramienta robusta para la clasificación de textos en base a los Objetivos de Desarrollo Sostenible (ODS), automatizando procesos clave como la limpieza de datos, la vectorización y la predicción mediante un modelo basado en RandomForestClassifier. La implementación de la API y la aplicación web ofrece una solución práctica para analistas de datos que requieren procesar grandes volúmenes de información textual, permitiéndoles generar insights útiles de forma más rápida y eficiente. Además, la funcionalidad de reentrenamiento garantiza que el modelo evolucione y se adapte a nuevos datos, manteniendo su relevancia con el paso del tiempo.

El uso de tecnologías como FastAPI para el backend y React para el frontend permitió construir un sistema escalable y modular, lo que facilita futuras mejoras y personalizaciones.

Asimismo, la integración de endpoints variados ofrece versatilidad en el uso de la herramienta, ya sea para la predicción de un solo texto o el análisis masivo de datos mediante archivos.

En resumen, este proyecto ofrece una solución innovadora y eficiente para la clasificación de textos en el contexto de los ODS, con un enfoque en la automatización y mejora continua a través del reentrenamiento. A futuro, el sistema podría expandirse para incluir más categorías de ODS o adaptar los modelos a otros tipos de texto.

Trabajo en equipo

A continuación, se presenta toda la información asociada a la distribución del trabajo realizada por el equipo, la contribución individual, el rol dentro de esta etapa y el puntaje asignado a cada integrante.

Nombre	Rol	Puntaje Otorgado
Nicolas Rozo Fajardo	Líder de Proyecto e Ingeniero de Datos	33.33
Laura Calderón Molina	Ingeniero de Software responsable del diseño	33.33
Sara Benavides Mora	Ingeniero de Software responsable desarrollar la aplicación	33.33

Posteriormente, se presenta la lista de tareas asignadas, los responsables de su realización y el tiempo invertido en completar la tarea. Cabe resaltar que la totalidad de las tareas asignadas fueron completadas.

Tarea	Responsables	Tiempo Invertido
Implementación de endpoints para la API	Nícolas Rozo Fajardo	2 horas
Desarrollo de la funcionalidad de reentrenamiento	Nicolas Rozo Fajardo	2 horas
Diseño e implementación de la UI para la aplicación web	Nicolas Rozo Fajardo, Laura Calderón Molina, Sara Benavides Mora	4 horas
Validación de las predicciones y métricas del modelo	Nicolas Rozo Fajardo, Laura Calderón Molina, Sara Benavides Mora	2 hora
Pruebas y debugging de la	Nicolas Rozo Fajardo, Laura	3 horas

API y frontend	Calderón Molina, Sara Benavides Mora	
Redacción y revisión del informe	Laura Calderón Molina	3 horas
Video	Sara Benavides Mora	4 horas

De la misma forma, se presentan los retos enfrentados durante la solución del proyecto y las decisiones tomadas para poder superarlos.

Reto	Solución
Complejidad en el reentrenamiento del modelo con nuevos datos y datos antiguos	Se implementó una estrategia que combina datos nuevos con un porcentaje del set de datos originales para mantener el balance entre precisión y actualización del modelo.
Manejo de archivos grandes para la predicción masiva	Optamos por dividir el procesamiento en batches y utilizar una estructura asíncrona en FastAPI para mejorar la eficiencia.
Coordinación entre el backend y el frontend	Se trabajó en la sincronización para asegurar que las interfaces entre ambas partes funcionaran sin problemas.

A modo de conclusión, se puede afirmar que el trabajo llevado durante esta etapa fue satisfactorio ya que, además de obtener excelentes resultados en lo que respecta a la construcción de la aplicación, se cumplió con todas las tareas propuestas y no se presentaron grandes inconvenientes a la hora de desarrollar las distintas tareas como grupo.

Bibliografía

[1] *¿Qué es el random forest?* (s.f.). IBM. https://www.ibm.com/mx-es/topics/random-forest