Informe del trabajo práctico final de Programación Avanzada 2024

Integrantes:

- **Curcio Ornela**
- **Gonzalez Sergio**
- **De los Angeles Torres Macarena**

Introducción

Para el curso de Programación Avanzada, nuestro equipo se enfrentó al desafío de desarrollar un sistema de recomendación de productos implementado como un servicio accesible a través de una API. Este sistema permite personalizar las publicidades que los usuarios ven en internet, basándose en los datos generados diariamente. A continuación, presentamos un resumen detallado de cómo abordamos el proyecto, los desafíos enfrentados, y las soluciones implementadas.

Creación del Aplicativo

El primer paso en nuestro proyecto fue la creación de datos utilizando un notebook proporcionado y la generación de un pipeline usando Airflow localmente, almacenando los datos procesados en archivos CSV locales. Una vez evaluada la información almacenada en los CSVs, se procedió a la creación de la base de datos para almacenar las recomendaciones finales. Usamos medidas de seguridad básica como el uso de variables de entorno seguras a través de un archivo .env, considerando que el código en GitHub debía ser público para la corrección por los docentes.

Pipeline en Airflow

El pipeline fue configurado dentro de un solo DAG que contiene seis tareas, organizadas en dos paralelos de tres tareas cada uno. Cada tarea en cada caso es similar: la primera tarea lee los archivos CSV de S3 y analiza los advertisers activos en `product` o `ctr`. La siguiente tarea analiza la información previa y genera el top de recomendaciones para los advertisers activos para product o ctr. La última tarea se encarga de escribir la información procesada en la base de datos. El pipeline está configurado para ejecutarse una vez por día a las 00:00, tomando la información del día anterior para simular un entorno más realista. Los logs se almacenan en una base de datos PostgreSQL en AWS RDS y se ejecuta usando un local executor con un paralelismo máximo de dos y dos workers para evitar el sobreuso de nuestra máquina en EC2.

Configuración de la Base de Datos

La base de datos se construyó utilizando PostgreSQL alojado en AWS RDS, debido a medidas de seguridad que impedían tener una base de datos local en una máquina. La base de datos fue diseñada con dos tablas principales: `ctr` y `product`, con una clave primaria que combina la fecha, el producto y el advertiser. Estas tablas almacenan los datos utilizados para generar las recomendaciones.

La API

Se crearon varios endpoints en nuestro 'sistema de recomendaciones' que brinda la recomendación usando la información de dos días previos, considerando el momento del llamado del endpoint. Esta lógica sigue el intento de que nuestra API se aproxime a un mundo real donde al día presente no contamos con datos, los datos del día anterior no existen porque se están procesando en nuestro pipeline y, por consiguiente, los datos de dos días anteriores sí están almacenados en nuestra base de datos.

- **/recommendations/{advertiser_id}/{model}:** Devuelve recomendaciones específicas para un anunciante, basadas en el modelo seleccionado, permitiendo ajustes dinámicos en las estrategias de publicidad.
- **/stats/count/{model}:** Proporciona el número de anunciantes que están utilizando un modelo específico, útil para evaluar la popularidad y efectividad de los modelos implementados.
- **/stats/variability/{model}:** Ofrece datos sobre la variabilidad de las recomendaciones de los anunciantes para un modelo.
- **/stats:** Retorna una lista de los anunciantes con las mayores tasas de click-through (CTR).
- **/history/ctr/{advertiser_id}** y **/history/product/{advertiser_id}:** Proporcionan un historial detallado del rendimiento del CTR y de los productos visualizados para un anunciante específico, respectivamente.
- **/unique_vs_repeated_recommendations/{advertiser_id}:** Compara las recomendaciones únicas versus las repetidas para un anunciante específico.

Almacenamiento en S3 bucket

Los archivos CSV iniciales, generados por el notebook, se almacenaron en un bucket de S3, y eran consumidos por el DAG de Airflow utilizando boto3. Como uno de los retoques finales para que el aplicativo simule un entorno profesional, se decidió que la información de cada tarea de nuestro pipeline se almacenara como CSV. Esto se consideró porque en un sistema escalable, las tareas pueden ejecutarse en diferentes computadoras, y enviar la información a través de contexto JSON no es eficaz ni a prueba de fallas. Además, el bucket puede configurarse para autolimpiarse cada cierto tiempo, lo que reduce los costos de almacenamiento.

Despliegue de Servicios en AWS

Para el despliegue de nuestros servicios utilizamos AWS EC2 para el servidor de Airflow con una máquina de tipo small, AWS RDS para la base de datos PostgreSQL, S3 para el almacenamiento de los CSVs, y AWS ECS para desplegar la imagen de nuestra API. Estas herramientas no solo proporcionan la robustez necesaria para nuestro sistema sino también la flexibilidad para escalar según la demanda.

Desafíos y Soluciones

Durante el desarrollo del proyecto, nos enfrentamos a varios desafíos técnicos. Uno de los problemas más significativos fue la incapacidad de todos los miembros del equipo

para conectarse simultáneamente a nuestro repositorio en GitHub. Esto generó problemas de versiones en los archivos que estábamos desarrollando. Este inconveniente fue resuelto compartiendo las últimas versiones de los desarrollos vía WhatsApp.

Otro desafío técnico que surgió fue durante la configuración de Docker para nuestra API. Tuvimos problemas al vincular los puertos correctamente, lo que inicialmente impidió que la API estuviera accesible desde el exterior. Este problema fue solucionado compartiendo experiencias con compañeros y entendiendo como sortearon este inconveniente.

Metodología de Trabajo

El desarrollo del proyecto se realizó inicialmente en un entorno local, lo que nos permitió realizar pruebas rápidas y eficientes antes de proceder a su despliegue en AWS. Esta estrategia nos ayudó a identificar y resolver problemas en un entorno controlado, minimizando así el impacto de los errores durante el despliegue en la nube.

En cuanto a la distribución de tareas, el equipo se dividió en dos roles principales: la DevOps Engineer y los Backend Developers. La DevOps fue responsable de configurar y mantener la infraestructura en AWS, asegurando que los servicios como EC2, RDS y ECS estuvieran adecuadamente configurados y optimizados para el rendimiento. Esta tarea incluyó la configuración de la base de datos, la implementación de las instancias de EC2 para Airflow, y la gestión de los contenedores en ECS.

Por otro lado, los Backend Developers se centraron en el desarrollo de la API utilizando FastAPI. Su trabajo fue desarrollar e implementar los endpoints de la API, y asegurar que las interacciones con la base de datos fueran eficientes y seguras. También estuvieron involucrados en la integración de la API con el sistema de recomendaciones y en la implementación de las pruebas.

Esta división de roles permitió que nuestro equipo trabajara de manera más eficaz y organizada, con cada miembro especializándose en áreas clave del proyecto, lo que resultó en un proceso de desarrollo más estructurado y una implementación exitosa.

Consideraciones sobre el código y posibles futuras mejoras:

Nuestro código en GitHub contiene mas archivos que la imagen desplegada en AWS en evidencia del trabajo escalonado que se fue realizando.

El principio de encapsulamiento y DRY fuero violados en pro de que pudiéramos entregar un trabajo que funcionara. Intento ser una meta encapsular el código tratando de arma un html estático para interactuar con la api, un file con de conexión a base de datos con las querys y un file de mensajes de error tratando de que todo funcione como un aplicativo integrado pero no pudo realizarse por falta de tiempo y expertise de algunos participantes del equipo.

Algunos endpoints devuelven respuestas armadas mas estéticamente construidas como diccionarios y no json puros, esto es porque creemos que era mas importante devolver una respuesta entendible para nuestros profesores.

Spanglish, el código sigue la meta de estar escrito en ingles así como la documentación de la API, aunque existen archivos escritos en español como el README y algunos

comentarios. La meta luego de la corrección del trabajo es hacer la migración al idioma ingles en su totalidad para nuestros portfolios.