Trabajo Fin de Máster

Máster Ciencia de Datos 2023-2024

Implementación de un modelo de traducción de lenguaje natural para la consulta de datos de Google Analytics 4

https://github.com/danielurrutxua/NL-to-GA4-Query

Autores:

Alberto Sebastián
Álvaro Salmerón
Daniel Urruchua
Juan Ignacio Alberola

ÍNDICE

Abstract

El objetivo del presente proyecto de fin de máster, elaborado para la edición 2023-2024 del Máster de Ciencia de Datos de Kschool, es desarrollar un traductor de lenguaje natural (en castellano) a un listado de métricas y dimensiones comprensibles por la API de Google Analytics 4 (GA4). La aproximación para conseguir el objetivo consiste en el uso de modelos Transformer Encoder-Decoder.

El principal reto es asegurar que el modelo interprete correctamente una variedad suficiente de métricas, dimensiones y rangos de fechas, permitiendo a usuarios sin conocimientos técnicos realizar consultas sobre las estadísticas de una página web.

La finalidad del proyecto es poder implantarlo inicialmente en un departamento de una agencia digital para que el área de desarrollo de negocio y venta para reducir la dependencia del equipo de analistas de forma que si deben consultar algún dato sobre el uso que los clientes dan de sus activos digitales no dependan de la disponibilidad de los analistas.

Una vez que el modelo sea utilizado y mejorado con las consultas de los primeros usuarios, se prevé comercializar la herramienta mediante una suscripción, generando ingresos adicionales.

Para alcanzar este objetivo, es esencial crear una amplia colección de pares de datos que relacione consultas en lenguaje natural con su correspondiente nomenclatura técnica de métricas, dimensiones y rangos de fechas de Google Analytics 4. Esta tarea es crucial debido a la falta de conjuntos de datos preexistentes específicamente diseñados para esta función.

Tras entrenar el modelo con estos pares de datos y realizar las pruebas pertinentes, el modelo será desplegado en una máquina virtual optimizada para deep learning en Google Cloud. A través de una REST API, el modelo permitirá recibir consultas en lenguaje natural, traducirlas automáticamente a métricas y dimensiones de GA4, ejecutar la consulta mediante la Data API y devolver los resultados en un formato comprensible para el usuario.

Palabras clave: analítica digital, redes neuronales, deep learning, Google Analytics 4, Google Cloud, inteligencia artificial.

1. Introducción

Este informe describe el desarrollo de un proyecto cuyo objetivo es traducir consultas en lenguaje natural a Métricas y Dimensiones específicas para extraer datos de la API de Google Analytics 4 (GA4). GA4 es una herramienta potente que proporciona análisis detallados sobre el comportamiento de los usuarios en sitios web y aplicaciones móviles. Sin embargo, su complejidad y la necesidad de conocimientos técnicos avanzados para realizar consultas precisas representan una barrera significativa para muchos usuarios. Nuestro objetivo es facilitar el acceso a estos datos mediante una interfaz que permita realizar consultas en lenguaje natural, haciéndolo más accesible para usuarios sin habilidades técnicas avanzadas. Esto se logrará utilizando modelos avanzados de procesamiento de lenguaje natural (NLP) que conviertan automáticamente las consultas en lenguaje natural a un formato compatible con la API de GA4.

1.1 Caso de uso y comprensión del negocio

El caso de uso de este proyecto se centra en profesionales de marketing, propietarios de pequeñas empresas y otros usuarios no técnicos que necesitan acceder a los datos de GA4 para tomar decisiones informadas. Actualmente, estos usuarios dependen en gran medida de analistas de datos o de tener conocimientos técnicos para extraer información útil de GA4, lo cual puede ser un proceso lento y costoso. Este proyecto pretende eliminar estas barreras mediante la implementación de una solución que permite a los usuarios formular preguntas en lenguaje natural, como "¿Cuántos usuarios nuevos tuvimos el mes pasado?" o "¿Cuál fue la tasa de rebote la semana pasada?", y obtener respuestas precisas sin la necesidad de conocer la documentación de la API y realizar consultas complejas.

Contexto del Problema:

En muchas organizaciones, la capacidad de acceder y analizar datos es crucial para la toma de decisiones estratégicas y operativas. Sin embargo, la complejidad de las herramientas de análisis como GA4 puede limitar su uso a personas con habilidades técnicas específicas. Esto crea una dependencia significativa de los equipos de analistas de datos, lo cual puede retrasar la obtención de insights valiosos y limitar la agilidad de la empresa en respuesta a nuevas oportunidades o desafíos.

Solución Propuesta:

La solución propuesta implica el desarrollo de un modelo de NLP que traduce consultas en lenguaje natural a Métricas y Dimensiones de GA4. Esta solución permitirá que los usuarios formulen consultas de manera intuitiva y reciban respuestas rápidas y precisas, facilitando así el acceso a datos importantes sin la necesidad de conocimientos técnicos avanzados.

1.2 Objetivos analíticos y plan de proyecto

Objetivos Analíticos:

- **Desarrollo del Modelo NLP**: Crear y entrenar un modelo de procesamiento de lenguaje natural que pueda interpretar consultas en lenguaje natural y traducirlas a Métricas y Dimensiones de la API de GA4.
- **Acceso a Datos de GA4**: Asegurar que el modelo pueda interactuar de manera efectiva con la API de Google Analytics 4 para ejecutar las consultas traducidas y recuperar los datos solicitados.
- **Usabilidad**: Diseñar una interfaz de usuario que permita a los usuarios formular consultas en lenguaje natural de manera sencilla y eficiente.

Plan de Proyecto (Visión General):

El proyecto se divide en varias fases:

- **Recopilación y Análisis de Datos**: Recolectar y preparar los datos necesarios para entrenar el modelo de NLP.
- **Desarrollo del Modelo NLP**: Entrenar el modelo utilizando técnicas avanzadas de deep learning.
- **Desarrollo de la Interfaz de Usuario**: Crear una interfaz intuitiva para interactuar con el modelo de NLP.
- **Pruebas y Validación**: Asegurar que el modelo y la interfaz funcionen correctamente.
- **Despliegue**: Implementar la solución en un entorno de producción.
- **Mantenimiento y Mejoras Continuas**: Asegurar el funcionamiento continuo y la mejora del sistema.

2. Contexto del negocio

Introducción:

Este Trabajo de Fin de Máster (TFM) se desarrolla con un enfoque práctico y profesional, simulando un proyecto empresarial real. El objetivo es abordar una necesidad concreta del mundo empresarial: permitir que usuarios no técnicos, como profesionales de marketing y propietarios de pequeñas empresas, puedan realizar consultas en lenguaje natural a los datos de Google Analytics 4 (GA4) y obtener insights valiosos sin la necesidad de conocimientos avanzados en APIs o SQL. La solución propuesta consiste en la implementación de un modelo de procesamiento de lenguaje natural (NLP) que traduzca consultas en lenguaje natural a Métricas y Dimensiones de la API de GA4.

Antecedentes:

Google Analytics 4 (GA4) es una herramienta avanzada de análisis web utilizada por empresas para obtener insights detallados sobre el comportamiento de los usuarios en sus sitios web y aplicaciones móviles. Sin embargo, la complejidad de GA4 y la necesidad de conocimientos técnicos avanzados para realizar consultas precisas limitan su uso a analistas de datos y personal técnico especializado. Esta barrera dificulta que profesionales de marketing y otros usuarios no técnicos puedan acceder a información crítica de manera eficiente y oportuna.

Objetivos Empresariales:

- **Accesibilidad Mejorada a Datos**: Permitir a los usuarios no técnicos acceder a los datos de GA4 a través de consultas en lenguaje natural, eliminando la necesidad de conocimientos especializados en APIs o análisis de datos.
- **Reducción de Dependencia Técnica**: Disminuir la carga de trabajo de los analistas de datos y el personal técnico, permitiendo que los usuarios no técnicos realicen consultas de datos por sí mismos.
- **Mejora en la Toma de Decisiones**: Facilitar la toma de decisiones basada en datos al proporcionar insights rápidos y precisos, accesibles directamente por los profesionales de marketing y gerentes de negocios.
- **Aumento de la Eficiencia Operativa**: Optimizar los procesos operativos al permitir un acceso más rápido y directo a los datos necesarios para la toma de decisiones.
- **Monetización de la Herramienta**: Después de una implementación inicial exitosa, comercializar la herramienta como un servicio de suscripción para generar ingresos adicionales.

Criterio de Éxito Empresarial:

- **Adopción del Sistema**: Medir el nivel de adopción y uso del sistema por parte de los usuarios no técnicos mediante encuestas y análisis de uso.
- **Reducción del Tiempo de Consulta**: Evaluar la reducción del tiempo promedio necesario para obtener insights desde la formulación de la consulta hasta la obtención de resultados.

- **Satisfacción del Usuario**: Mejorar la satisfacción del usuario, medida a través de encuestas de feedback y análisis de uso de la interfaz.
- **Impacto en la Toma de Decisiones**: Incrementar el número de decisiones informadas tomadas basadas en los insights proporcionados por el sistema.

Inventario de Recursos:

- **Recursos Humanos:**
    - **Desarrolladores de Software**: Para el desarrollo de la interfaz y el backend.
    - **Expertos en NLP**: Para el desarrollo y entrenamiento del modelo de procesamiento de lenguaje natural.
    - **Analistas de Datos**: Para la validación y verificación de la precisión de los datos.
    - **Diseñadores de Interfaz de Usuario**: Para el diseño y usabilidad de la interfaz.
- **Tecnología:**
    - **Herramientas de Desarrollo de Software**: Python, TensorFlow, React.js.
    - **Infraestructura en la Nube**: Google Cloud Platform (GCP) para el despliegue del modelo y la aplicación.
    - **API de GA4**: Para acceder y consultar los datos de Google Analytics.
- **Datos**: Acceso a datos de prueba de GA4 y generación de datasets de pares de consultas en lenguaje natural con Métricas y Dimensiones de la documentación de GA4.

Requisitos, Suposiciones y Restricciones:

- **Requisitos:**
    - **Acceso a la API de GA4**: Permite la extracción y consulta de datos.
    - **Entornos de Desarrollo**: Adecuados para entrenar y desplegar modelos NLP.
    - **Capacitación del Equipo**: Conocimiento en tecnologías de NLP y análisis de datos.
- **Suposiciones:**
    - **Acceso a Internet**: Los usuarios deben tener acceso a Internet para utilizar la aplicación web.
    - **Funcionalidad de la API de GA4**: La API debe proporcionar los datos necesarios sin restricciones mayores.
- **Restricciones:**
    - **Presupuesto**: Limitaciones presupuestarias para infraestructura en la nube y desarrollo.
    - **Tiempo**: Plazos ajustados para el desarrollo y despliegue del proyecto.
    - **Privacidad y Seguridad**: Garantizar la privacidad y seguridad de los datos de los usuarios.

Riesgos y Contingencias:

- **Riesgos Técnicos:**
  - **Problemas de Integración**: Posibles problemas al integrar el modelo NLP con la API de GA4.
  - **Errores en el Modelo**: Errores y limitaciones en la precisión del modelo NLP.
  - **Escalabilidad**: Desafíos para asegurar que la infraestructura en la nube pueda escalar adecuadamente.
- **Contingencias Técnicas:**
  - **Pruebas Extensivas**: Implementar pruebas exhaustivas antes del despliegue.
  - **Plan de Respaldo**: Contar con un plan de contingencia para la infraestructura.
  - **Soporte Continuo**: Proporcionar soporte técnico continuo.
- **Riesgos Operativos:**
  - **Baja Adopción**: Resistencia al cambio por parte de los usuarios.
  - **Falta de Capacitación**: Usuarios no capacitados adecuadamente para utilizar la herramienta.
- **Contingencias Operativas:**
  - **Sesiones de Formación**: Realizar sesiones de formación y proporcionar documentación detallada.
  - **Soporte al Usuario**: Ofrecer soporte continuo y recoger feedback para mejorar la herramienta.

Costes y Beneficios:

- **Costes:**
  - **Desarrollo de Software**: Salarios del equipo de desarrollo.
  - **Infraestructura en la Nube**: Costes asociados a Google Cloud Platform.
  - **Tiempo de Desarrollo**: Horas dedicadas al desarrollo y pruebas.
- **Beneficios**:
  - **Ahorro de Tiempo**: Reducción de la dependencia de analistas de datos, permitiendo una extracción de datos más rápida.
  - **Mejora en la Toma de Decisiones**: Acceso a datos precisos y oportunos para una mejor toma de decisiones.
  - **Mayor Eficiencia**: Optimización de procesos operativos al permitir un acceso directo a los datos.

Objetivos de Ciencia de Datos y Criterios de Éxito:

- **Objetivos**:
  - **Desarrollo del Modelo NLP**: Crear un modelo de procesamiento de lenguaje natural que traduzca consultas en lenguaje natural a la estructura requerida por la API.
  - **Integración con GA4**: Asegurar que el modelo pueda interactuar eficazmente con la API de GA4 para ejecutar consultas y recuperar datos.

    - **Diseño de Interfaz de Usuario**: Crear una interfaz intuitiva que facilite la interacción con el sistema.
- **Criterios de Éxito:**
    - **Precisión de Traducción**: Nivel de precisión y relevancia de las consultas traducidas por el modelo NLP.
    - **Usabilidad**: Evaluación de la usabilidad y satisfacción del usuario con la interfaz.
    - **Eficiencia en la Recuperación de Datos**: Tiempo y precisión en la recuperación y presentación de datos.

Solución Operativa:

La solución operativa consiste en un sistema que recibe consultas en lenguaje natural, las traduce a Métricas y dimensiones utilizando un modelo NLP, ejecuta las consultas a través de la API de GA4 y presenta los resultados de manera comprensible para el usuario. El sistema debe ser escalable, seguro y fácil de usar, con capacidades de monitoreo y mantenimiento para asegurar su funcionamiento continuo.

Plan de Proyecto:

- **Fase 1: Recopilación y Análisis de Datos**
    - **Objetivo**: Recolectar y preparar los datos necesarios para entrenar el modelo NLP.
    - **Actividades**: Realizar web scraping de la documentación oficial de GA4 para obtener una lista completa de métricas y dimensiones, y generar un dataset de pares de consultas en lenguaje natural y su equivalente.
- **Fase 2: Desarrollo del Modelo NLP**
    - **Objetivo**: Entrenar el modelo utilizando técnicas avanzadas de deep learning, específicamente Transformers.
    - **Actividades**: Preprocesamiento de datos, entrenamiento del modelo, y ajustes iterativos para mejorar la precisión y relevancia de las consultas traducidas.
- **Fase 3: Desarrollo de la Interfaz de Usuario**
    - **Objetivo**: Crear una interfaz intuitiva que permita a los usuarios interactuar con el modelo de NLP.
    - **Actividades**: Diseño y desarrollo de la interfaz utilizando tecnologías web modernas (React.js), integración con el backend para manejar las consultas y visualización de resultados.
- **Fase 4: Pruebas y Validación**
    - **Objetivo**: Asegurar que el modelo y la interfaz funcionen correctamente y satisfagan las necesidades de los usuarios.
    - **Actividades**: Realización de pruebas unitarias y de integración, recopilación de feedback de usuarios, y ajustes basados en los resultados de las pruebas.
- **Fase 5: Despliegue**
    - **Objetivo**: Implementar la solución en un entorno de producción, asegurando escalabilidad y disponibilidad.

    - **Actividades**: Configuración de una máquina virtual en Google Cloud, despliegue del modelo de NLP y de la interfaz de usuario, y configuración de una REST API para manejar las solicitudes de traducción.#
- **Fase 6: Mantenimiento y Mejoras Continuas**
    - **Objetivo**: Asegurar el funcionamiento continuo y la mejora del sistema basado en el feedback de los usuarios y nuevos requerimientos.
    - **Actividades**: Monitoreo del rendimiento del sistema, actualización del modelo y la interfaz según sea necesario.

Evaluación Inicial de Técnicas Analíticas:

Se evaluaron varias técnicas analíticas, incluidas las RNN (Redes Neuronales Recurrentes) y los modelos Transformer. Se optó por los modelos Transformer debido a su capacidad para manejar dependencias de largo alcance en el texto y su eficacia comprobada en tareas de traducción de lenguaje natural. La evaluación incluyó pruebas de precisión y relevancia de las consultas traducidas, así como la capacidad del modelo para escalar y manejar un volumen significativo de datos.

3. Datos

Durante el proceso de desarrollo del proyecto hemos tenido que tener en cuenta que se va a trabajar sobre dos fuentes de datos, la primera de ellas son los propios datos de Google Analytics 4 de una cuenta de prueba sin información confidencial ni personal de los usuarios.

Por otro lado, y a fin de entrenar el modelo se ha de generar un dataset de parejas entre lenguaje natural y sus correspondientes métricas y dimensiones además de rangos de fechas para que el modelo sea capaz de extraer dicha información y mediante la API oficial de GA4 se tenga la capacidad de recuperar los datos.

Se pasa a explicar el primero de los dos puntos.

3.1 Google Analytics 4

Google Analytics 4 (GA4) es la última versión de la plataforma de análisis de Google, diseñada para proporcionar un análisis más completo y detallado del comportamiento de los usuarios en sitios web y aplicaciones móviles. A diferencia de las versiones anteriores de Google Analytics, GA4 se centra en un modelo de datos basado en eventos, lo que permite una mayor flexibilidad y profundidad en el análisis.

Características Clave de GA4:

1. **Modelo Basado en Eventos**: Cada interacción del usuario se registra como un evento, permitiendo un análisis detallado y específico de cada acción realizada por el usuario.

2. **Medición Multiplataforma**: GA4 permite el seguimiento de usuarios a través de diferentes plataformas y dispositivos, proporcionando una vista unificada del comportamiento del usuario.

3. **Capacidades de Análisis Avanzadas**: Incluye características avanzadas de análisis como machine learning para proporcionar insights automáticos y predictivos.

4. **Privacidad y Cumplimiento**: Diseñado para adaptarse a los crecientes requisitos de privacidad y regulaciones como el GDPR.

Esquema de Dimensiones y Métricas de GA4

Para entender y utilizar eficazmente GA4, es crucial conocer las dimensiones y métricas que ofrece.

Las dimensiones y métricas son los componentes básicos que permiten analizar el comportamiento de los usuarios.

Dimensiones y dimensiones

Las dimensiones son atributos descriptivos de los datos, que categorizan y agregan contexto a las métricas. Algunas de las dimensiones más relevantes son:

1. **deviceCategory**: Categoría del dispositivo desde el cual se originó la sesión. Acepta algunos valores comunes como desktop, tablet, mobile.
2. **defaultChannelGrouping**: Agrupación predeterminada de canales que trajo tráfico al sitio web. Acepta algunos valores comunes como Organic Search, Paid Search, Direct, Referral, Social, Email.
3. **sessionMedium**: Medio a través del cual se adquirió el tráfico. Acepta algunos valores comunes como organic, cpc, referral, none.
4. **country**: País desde el cual se origina la sesión. Acepta algunos valores comunes como nombres de países → United States, Canada, Mexico, etc.

Se puede consultar el listado de dimensiones en la documentación oficial.

Las métricas son medidas cuantitativas que indican el rendimiento de diferentes aspectos del sitio web. Algunas de las métricas más importantes son:

1. **sessions**: Número total de sesiones.
2. **totalUsers**: Número total de usuarios únicos.
3. **newUsers**: Número de usuarios que visitan el sitio web por primera vez.
4. **pageviews**: Número total de visitas de la página.
5. **bounceRate**: Porcentaje de sesiones de una sola página.

Se puede consultar el listado de métricas en la documentación oficial.

La adquisición de datos se realizará utilizando la API de GA4. La API permite realizar consultas detalladas para extraer dimensiones y métricas específicas necesarias para el análisis. Adicionalmente, se realizó un web scraping de la documentación oficial de GA4 para obtener un listado completo de las dimensiones y métricas disponibles, que se almacenó en un archivo CSV.

3.1 Creación del dataset de pares

Se ha generado un dataset de 2.000.000 de pares de equivalencias entre frases en castellano y su equivalente válido para consultar a la API.

Para generar el dataset de entrenamiento constituido por pares de frases en castellano con su equivalencia de métricas, dimensiones y fechas se han seguido los siguientes pasos:

3.1.1 Web scraping de la documentación oficial

Para obtener un listado completo de las dimensiones y métricas de GA4, se realizó un web scraping de la documentación oficial de Google Analytics. A continuación, se detalla el proceso seguido:

1. **Identificación de la Fuente**:
    - La documentación oficial de Google Analytics proporciona descripciones detalladas de todas las dimensiones y métricas disponibles en GA4. Esta información se encuentra en formato HTML en el sitio web de Google Developers accesible desde este enlace.

2. **Extracción de Datos**:
    - Se utilizó la librería de scraping BeautifulSoup en Python para extraer la información de la página web.
    - El scraping se enfocó en las tablas y secciones que listan las dimensiones y métricas, capturando sus nombres, descripciones y posibles valores.

3. **Almacenamiento en CSV**:
    - La información extraída se organizó y almacenó en un archivo CSV para facilitar su uso posterior.
    - Cada fila en el CSV corresponde a una dimensión o métrica, con columnas para el nombre, descripción y otros detalles relevantes.

3.1.2 Procesamiento del scraping

El objetivo de esta sección es explicar cómo se ha generado el dataset que relaciona una consulta en lenguaje natural con su equivalente query en la API de Google Analytics 4 (GA4). El repositorio con el código fuente completo se encuentra en el siguiente enlace: NL-to-GA4-Query.

Funcionamiento de la API de GA4

A continuación, se detallan algunos aspectos clave del funcionamiento de la API de GA4:

Métricas

Las métricas son valores numéricos que representan diferentes aspectos del tráfico y comportamiento del usuario en un sitio web o aplicación. Son datos cuantitativos que se pueden medir y analizar.

- **Ejemplos de Métricas:**
    - sessions: Número de sesiones.
    - pageViews: Número de vistas de página.
    - bounceRate: Tasa de rebote.
    - transactionRevenue: Ingresos por transacción.

Dimensiones

Las dimensiones son atributos de los datos que se utilizan para segmentar y analizar las métricas. Son datos cualitativos que describen los diferentes aspectos de las sesiones y los usuarios.

- **Ejemplos de Dimensiones:**
  - country: País del usuario.
  - deviceCategory: Categoría del dispositivo (desktop, tablet, mobile).
  - browser: Navegador utilizado por el usuario.
  - source: Fuente de tráfico.

Rangos de Fechas

Los rangos de fechas especifican el periodo de tiempo para el cual se desea extraer los datos. Pueden ser fechas absolutas o relativas.

- **Ejemplos de Rangos de Fechas:**
  - Fechas Absolutas: 2023-01-01 a 2023-01-31
  - Fechas Relativas: last7days (últimos 7 días)

Filtros

Los filtros se utilizan para refinar los resultados de las consultas. Pueden aplicarse tanto a métricas como a dimensiones y pueden ser de varios tipos:

- **Tipos de Filtros:**
  - Igualdad (EQUALS): Filtra por valores exactamente iguales.
    - Ejemplo: country == "Spain"
  - Mayor Que (GREATER_THAN): Filtra por valores mayores que un valor especificado.
    - Ejemplo: sessions > 100
  - Menor Que (LESS_THAN): Filtra por valores menores que un valor especificado.
    - Ejemplo: bounceRate < 0.5
  - Entre (BETWEEN): Filtra por valores dentro de un rango específico.
    - Ejemplo: transactionRevenue BETWEEN 100 AND 500

Solicitudes a la API de GA4

Las solicitudes a la API de GA4 se realizan mediante POST requests en formato JSON. La estructura de una solicitud incluye las métricas, dimensiones, rangos de fechas y filtros que se desean aplicar.

- **Consulta en Lenguaje Natural:** *"Necesito el número de sesiones por país en enero de 2023, filtrado por España."*
- **Consulta Equivalente en la API de GA4 (JSON):**

```json
{
  "metrics": [{"name": "sessions"}],
  "dimensions": [{"name": "country"}],
  "dateRanges": [{"startDate": "2023-01-01", "endDate": "2023-01-31"}],
  "dimensionFilter": {
    "filterType": "EQUALS",
    "fieldName": "country",
    "value": "Spain"
  }
}
```

Afinando el alcance

Después de entender cómo funciona la API de GA4 y cómo se estructuran las solicitudes, es importante definir claramente qué aspectos de la API vamos a utilizar y cuáles quedarán fuera del alcance de este proyecto.

En el siguiente apartado, detallaremos los criterios de selección y las limitaciones establecidas para la generación del dataset.

Para la generación del dataset, nos hemos centrado en un conjunto reducido de métricas y dimensiones que son las más utilizadas en la mayoría de los análisis de datos. Esto incluye métricas como sessions, pageViews, bounceRate, entre otras, y dimensiones como country, deviceCategory, browser, y source. Además, se considerarán tanto rangos de fechas absolutas como relativas, permitiendo flexibilidad en el análisis temporal de los datos.

Los filtros también serán parte integral del alcance, permitiendo refinar los resultados de las consultas. En este proyecto, se utilizarán filtros básicos sin concatenaciones de OR o AND, es decir, simplemente se podrá filtrar por una métrica, una dimensión o ambos. Los filtros de cadenas de texto incluirán los siguientes tipos:

- **Igualdad (EXACT)**: Filtra por valores exactamente iguales.
- **Comienza con (BEGINS_WITH)**: Filtra por valores que comienzan con un valor especificado.
- **Termina con (ENDS_WITH)**: Filtra por valores que terminan con un valor especificado.
- **Contiene (CONTAINS)**: Filtra por valores que contienen un valor especificado.

Para los filtros numéricos, se incluirán los siguientes tipos:

- **Igualdad (EQUALS)**: Filtra por valores exactamente iguales.
- **Mayor Que (GREATER_THAN)**: Filtra por valores mayores que un valor especificado.
- **Menor Que (LESS_THAN)**: Filtra por valores menores que un valor especificado.
- **Mayor o Igual Que (GREATER_THAN_OR_EQUAL)**: Filtra por valores mayores o iguales a un valor especificado.

- **Menor o Igual Que (LESS_THAN_OR_EQUAL)**: Filtra por valores menores o iguales a un valor especificado.

Sin embargo, para mantener el enfoque y la simplicidad del proyecto, se ha decidido excluir ciertos aspectos avanzados. Las segmentaciones complejas basadas en el comportamiento del usuario o en atributos personalizados no se abordarán en esta fase. También se omitirán los filtros que requieren lógica condicional avanzada. Las consultas que requieran datos en tiempo real no serán implementadas inicialmente, y tampoco se considerarán métricas y dimensiones altamente específicas o personalizadas que no son comúnmente utilizadas.

Al limitar el alcance de esta manera, podemos centrarnos en los casos de uso más comunes y relevantes, garantizando la creación de un sistema eficiente y fácil de usar.

Estructura y Funcionamiento de la Solución

La solución desarrollada para generar el dataset que relaciona consultas en lenguaje natural con sus equivalentes en la API de Google Analytics 4 (GA4) sigue una estructura modular y organizada. A continuación, se describe cómo funciona la solución, el input que recibe, cómo se lleva a cabo la generación de las consultas y la exportación de los resultados.

Funcionamiento General

La aplicación principal se encuentra en el archivo `app.py`, donde se inicia el proceso de generación de pares de consultas. La función principal gestiona la interacción con el usuario, solicitando el número de pares a generar y si se desean incluir filtros y rangos de fechas en las consultas. Con esta información, se procede a construir cada consulta en lenguaje natural y su equivalente en la API de GA4.

Entrada del Usuario

La aplicación recibe los siguientes inputs del usuario:

1. **Número de Pares:** El usuario especifica la cantidad de pares de consultas que desea generar. Esto permite controlar la cantidad de datos en el dataset final.
2. **Filtros:** El usuario indica si desea incluir filtros en las consultas mediante una respuesta de sí (y) o no (n). Los filtros pueden ser aplicados tanto a métricas como a dimensiones, y pueden ser de varios tipos, como igualdad, mayor que, menor que, entre otros.
3. **Fechas:** El usuario indica si desea incluir rangos de fechas en las consultas mediante una respuesta de sí (y) o no (n). Los rangos de fechas pueden ser tanto absolutos (por ejemplo, un mes específico) como relativos (por ejemplo, los últimos 7 días).

Generación de Consultas

El core de la aplicación se encarga de generar las consultas en lenguaje natural y sus equivalentes en la API de GA4 de la siguiente manera:

Inicio de la Frase

Para la generación de una query, primero se selecciona una forma aleatoria con la que empezar la frase. Esto se realiza buscando en un diccionario de sinónimos el inicio de la frase, asegurando así una mayor variedad en las consultas generadas. El objetivo es que las frases no solo sean correctas y naturales, sino que también presenten variedad para enriquecer el dataset de entrenamiento.

Por ejemplo, si el inicio de la frase original es "Dame", se pueden utilizar sinónimos y variantes como "Quiero", "Muéstrame", "Proporcióname", "Necesito", "Enséñame", "Solicito", "Desearía ver", "Requiero" o "Podrías darme".

Ejemplos**:**

- Dame el número de sesiones.
- Quiero el número de sesiones.
- Muéstrame el número de sesiones
- Necesito el número de sesiones.

La selección aleatoria de estas variantes se realiza utilizando un diccionario de sinónimos, que se ha compilado a partir de fuentes confiables y se almacena en el archivo *sinonimos_dime.csv*. Este archivo contiene múltiples formas de expresar la misma solicitud, permitiendo a la aplicación generar frases variadas y naturales.

Selección de Métricas

Una vez seleccionada la frase inicial, el siguiente paso en la generación de la consulta es la inclusión de métricas. Las métricas representan valores numéricos que describen diferentes aspectos del tráfico y comportamiento del usuario en un sitio web o aplicación. En esta parte del proceso, hay dos opciones: incluir una sola métrica o incluir dos métricas.

Las métricas se seleccionan aleatoriamente de un archivo CSV llamado metrics_reduced.csv. Este archivo se generó a partir del scraping

Ejemplo del CSV metrics_reduced.csv:

Nombre de la Métrica	Sinónimos
sessions	Sesiones, Visitas, Inicios de sesión, Accesos, Entradas, Conexiones,...
activeUsers	Usuarios activos, Usuarios en línea, Usuarios conectados, Usuarios presentes, Usuarios vigentes, Usuarios actuales
averageSessionDuration	Duración media de la sesión, Tiempo promedio de sesión, Duración habitual de sesión

purchaseRevenue	Ingresos por compras, Ganancias por compras, Ventas por compras, Ingresos totales por compras, Ganancias totales por compras

En cada generación de consulta, se selecciona una métrica (o dos métricas) aleatoriamente del CSV, y se elige un sinónimo para cada métrica para asegurar la variabilidad en las frases generadas.

Ejemplo de Selección de Métrica:

1. Una Métrica: "Muéstrame el número de sesiones."
2. Dos Métricas: "Proporcióname las vistas de página y la tasa de rebote."

Selección de dimensión

Después de seleccionar las métricas, el siguiente paso en la generación de la consulta es la inclusión de dimensiones. En esta parte del proceso, hay dos opciones: incluir una dimensión o no incluir ninguna dimensión.

Las dimensiones se seleccionan aleatoriamente de un archivo CSV llamado `dimensions_reduced.csv`. Este archivo se generó a partir del scraping de la documentación oficial de Google Analytics y contiene las siguientes columnas:

- **Nombre de la Dimensión:** El nombre oficial de la dimensión tal como aparece en la documentación.
- **Sinónimos:** Variantes y sinónimos del nombre de la dimensión, añadidos para aumentar la variabilidad y naturalidad de las consultas generadas.

Ejemplo del CSV `dimensions_reduced.csv`:

Nombre de la Dimensión	Sinónimos
browser	Explorador web, Browser, Visor de internet, Navegador web, Explorador de la web, Navegador
country	Nación, Estado, República, Territorio, País soberano, País
platform	Sistema, Entorno, Framework, Plataforma tecnológica, Ecosistema, Plataforma

En cada generación de consulta, se selecciona una dimensión aleatoriamente del CSV, y se elige un sinónimo para cada dimensión para asegurar la variabilidad en las frases generadas.

Ejemplo de Selección de Dimensión:

1. Muéstrame el número de sesiones por país.
2. Proporcióname las vistas de página por categoría del dispositivo

Selección de filtros

Los filtros son una parte crucial en la generación de consultas, ya que permiten refinar los resultados obtenidos mediante métricas y dimensiones. En este proceso, los filtros pueden ser aplicados tanto a métricas (metricFilter) como a dimensiones (dimensionFilter). Cada consulta puede incluir uno de los siguientes escenarios:

1. Solo un filtro de métrica.
2. Solo un filtro de dimensión.
3. Ambos filtros (métrica y dimensión).
4. Ningún filtro.

Para determinar qué filtros aplicar, se selecciona aleatoriamente una métrica o una dimensión del dataset correspondiente. Luego, se elige aleatoriamente un tipo de coincidencia (matchType), que puede ser de tipo string o numérico.

Para los filtros de tipo string, se incluyen los siguientes tipos:

- Igualdad (EXACT)
- Comienza con (BEGINS_WITH)
- Termina con (ENDS_WITH)
- Contiene (CONTAINS)

Para los filtros de tipo numérico, se incluyen los siguientes tipos:

- Igualdad (EQUALS)
- Mayor Que (GREATER_THAN)
- Menor Que (LESS_THAN)
- Mayor o Igual Que (GREATER_THAN_OR_EQUAL)
- Menor o Igual Que (LESS_THAN_OR_EQUAL)

Selección de Métrica o Dimensión:

1. Se selecciona aleatoriamente una métrica del archivo `metrics_reduced.csv` o una dimensión del archivo `dimensions_reduced.csv`.
2. Determinación del Tipo de Coincidencia: Se elige aleatoriamente uno de los tipos de coincidencia (matchType) disponibles.
3. Generación del Valor del Filtro:

    - Si el filtro es de tipo string, se selecciona una palabra aleatoria de un archivo CSV llamado `4000-most-common-english-words-csv.csv`.
    - Si el filtro es de tipo numérico, se genera un número entero aleatorio.
4. Construcción de la Frase: Se crea una frase en lenguaje natural que describe el filtro aplicado. En el caso de que haya dos filtros (métrica y dimensión), se elige aleatoriamente cuál filtro va primero para asegurar la variedad en las consultas generadas.

Ejemplos de Filtros

1. Filtro de Métrica:
    - Muéstrame el número de sesiones mayor que 100.
2. Filtro de Dimensión:
    - Quiero ver las sesiones filtradas por país igual a España.
3. Ambos Filtros:
    - Proporcióname las vistas de página menores que 200 y filtradas por navegador que contiene Chrome."

Este proceso asegura que cada consulta generada no solo sea correcta en términos de formato y significado, sino que también presente una diversidad significativa en la forma en que se expresan las solicitudes. Al incluir filtros de manera aleatoria y variada, el dataset resultante es más robusto y útil para entrenar modelos de procesamiento de lenguaje natural.

Selección de fechas

La selección de fechas es un componente fundamental en la generación de consultas. Las fechas se generan aleatoriamente de acuerdo con diferentes tipos de rangos de fechas. A continuación se explican los distintos tipos de rangos de fechas con ejemplos específicos:

1. YEAR: Un solo año completo.
    - Fecha en Formato Fecha: '2023-01-01' a '2023-12-31'
    - Lenguaje Natural: "a lo largo de 2023"
    - Ejemplo: "Muéstrame las sesiones a lo largo de 2023."
2. YEAR_RANGE: Un rango de dos años distintos.
    - Fecha en Formato Fecha: '2019-01-01' a '2021-12-31'
    - Lenguaje Natural: "entre 2019 y 2021"
    - Ejemplo: "Dame las vistas de página entre 2019 y 2021."
3. MONTH: Un solo mes en un año específico.
    - Fecha en Formato Fecha: '2023-03-01' a '2023-03-31'
    - Lenguaje Natural: "en marzo de 2023"
    - Ejemplo: "Proporcióname las conversiones en marzo de 2023."
4. MONTH_RANGE: Un rango de meses en un año específico.
    - Fecha en Formato Fecha: '2023-03-01' a '2023-05-31'
    - Lenguaje Natural: "entre marzo y mayo de 2023"
    - Ejemplo: "Enséñame los ingresos entre marzo y mayo de 2023."
5. DAY: Un solo día específico.

    - Fecha en Formato Fecha: '2023-03-15' a '2023-03-15'
    - Lenguaje Natural: "el 15 de marzo de 2023"
    - Ejemplo: "Necesito el número de usuarios activos el 15 de marzo de 2023."
6. DAY_RANGE: Un rango de días específicos.
    - Fecha en Formato Fecha: '2023-03-15' a '2023-03-20'
    - Lenguaje Natural: "del 15 al 20 de marzo de 2023"
    - Ejemplo: "Solicito las sesiones del 15 al 20 de marzo de 2023."
7. TODAY: El día actual.
    - Fecha en Formato Fecha: '2023-07-01' (supongamos que hoy es 1 de julio de 2023)
    - Lenguaje Natural: "hoy"
    - Ejemplo: "Dame las conversiones de hoy."
8. YESTERDAY: El día anterior.
    - Fecha en Formato Fecha: '2023-06-30' (supongamos que hoy es 1 de julio de 2023)
    - Lenguaje Natural: "ayer"
    - Ejemplo: "Quiero ver los usuarios activos de ayer."
9. DAYS_AGO: Un número específico de días atrás desde hoy.
    - Fecha en Formato Fecha: '2023-06-15' a '2023-07-01' (hace 15 días desde hoy, 1 de julio de 2023)
    - Lenguaje Natural: "en los últimos 15 días"
    - Ejemplo: "Muéstrame las sesiones en los últimos 15 días."
10. WEEKS_AGO: Un número específico de semanas atrás desde hoy.
    - Fecha en Formato Fecha: '2023-05-01' a '2023-07-01' (hace 8 semanas desde hoy, 1 de julio de 2023)
    - Lenguaje Natural: "en las últimas 8 semanas"
    - Ejemplo: "Proporcióname las vistas de página en las últimas 8 semanas."
11. MONTHS_AGO: Un número específico de meses atrás desde hoy.
    - Fecha en Formato Fecha: '2023-04-01' a '2023-07-01' (hace 3 meses desde hoy, 1 de julio de 2023)
    - Lenguaje Natural: "en los últimos 3 meses"
    - Ejemplo: "Enséñame los ingresos en los últimos 3 meses."
12. YEARS_AGO: Un número específico de años atrás desde hoy.
    - Fecha en Formato Fecha: '2020-01-01' a '2023-07-01' (hace 3 años desde hoy, 1 de julio de 2023)
    - Lenguaje Natural: "en los últimos 3 años"
    - Ejemplo: "Quiero ver las conversiones en los últimos 3 años."

Este enfoque asegura que las consultas generadas cubran una amplia gama de posibles escenarios temporales, proporcionando así un dataset robusto y variado para el entrenamiento del modelo de procesamiento de lenguaje natural.

4 Tecnología

Para explicar las diferentes tecnologías usadas en el proyecto debemos separar el mismo en las diferentes fases, pero antes de ello mostramos gráficamente la arquitectura de todo el proyecto.

Google Colab Pro

Ante la dificultad de trabajar en local, el entrenamiento del modelo se ha hecho usando Google Colab, **en su versión PRO**, haciendo uso de las siguientes librerías en un entorno de ejecución con GPU:

- **TensorFlow 2.16.2**: Framework de código abierto para el aprendizaje automático. Se utiliza para construir y entrenar redes neuronales.
- **Keras 3.3.3**: API de alto nivel para construir y entrenar modelos de aprendizaje profundo. Funciona sobre TensorFlow.
- **Keras NLP 0.12.0**: Conjunto de herramientas para el procesamiento del lenguaje natural utilizando Keras.
- **Numpy 1.25.2**: Biblioteca fundamental para el cálculo numérico en Python, que proporciona soporte para matrices y operaciones matemáticas avanzadas.
- **Pandas 2.0.3**: Biblioteca que ofrece estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar.

Máquina Virtual Deep Learning

El modelo entrenado se despliega en un **Servicio de Virtual Machine de Compute Engine en Google Cloud Platform**. Esta máquina virtual tiene instaladas las mismas librerías en sus versiones correspondientes para asegurar la coherencia entre el entorno de desarrollo y producción. Además, se ha instalado:

- **Flask 3.0.3**: Microframework de Python que se utiliza para desarrollar aplicaciones web. En este proyecto, se usa para crear una API que permite interactuar con el modelo entrenado.

Reflex

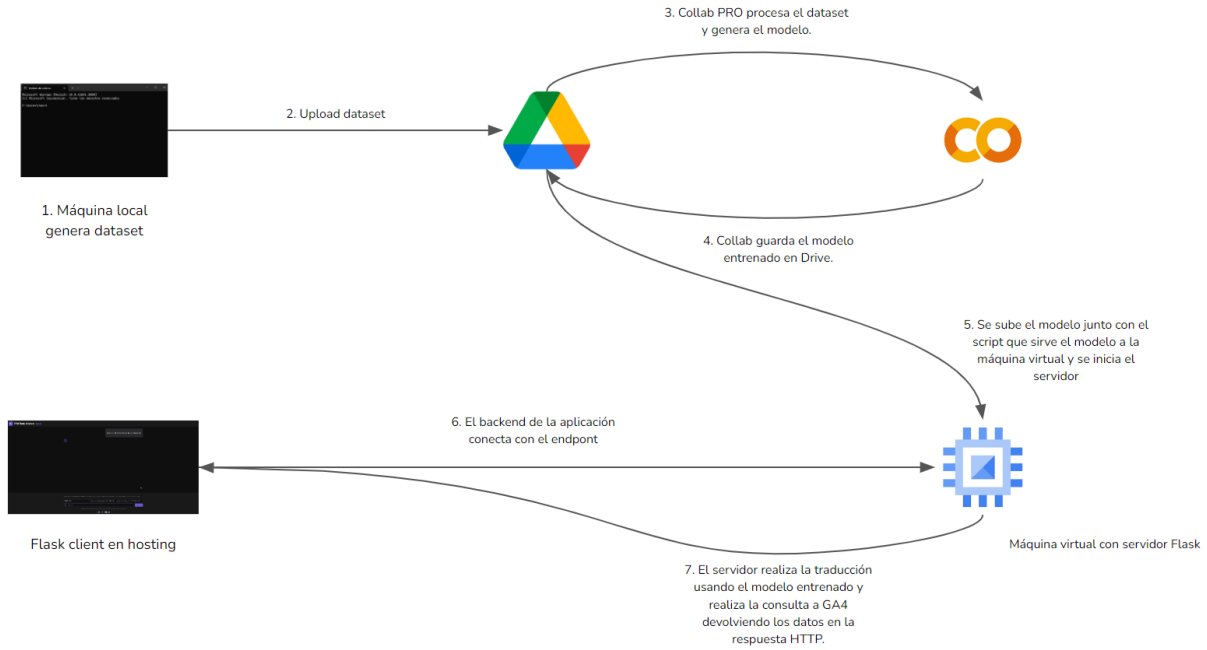
Reflex es una librería para el desarrollo de aplicaciones web full-stack usando únicamente Python. Permite definir tanto el frontend como el backend en una sola base de código, eliminando la necesidad de aprender múltiples lenguajes o frameworks para conectar ambas partes del desarrollo. Reflex compila el frontend a una aplicación React y el backend utiliza FastAPI, manteniendo toda la lógica y gestión de estado en Python.

Uso de Reflex en el Proyecto:

- **Desarrollo de la Interfaz de Usuario**: Reflex facilita la creación de una interfaz de usuario interactiva y dinámica utilizando componentes web definidos en Python.

- **Integración con el Backend**: Reflex permite conectar la interfaz de usuario con la API de Flask, simplificando la comunicación entre el frontend y el backend.
- **Gestión de Estado y Lógica de la Aplicación**: Reflex maneja la gestión de estado y la lógica de la aplicación en el backend, lo que facilita la implementación de funcionalidades complejas.
- **Despliegue Rápido y Sencillo**: Reflex permite desplegar aplicaciones con un solo comando, lo que agiliza el proceso de puesta en producción del proyecto.

Esquema arquitectura de la solución



5 Tratamiento de datos

Con el dataset creado siguiendo las instrucciones de secciones anteriores, antes de comenzar con el entrenamiento del modelo hemos tenido que realizar un preprocesamiento de los datos para que la entrada del modelo se ajuste a las necesidades del mismo.

Tal como se ha explicado anteriormente la salida de la generación de pares devuelve un archivo CSV en el cual se disponen de varias columnas.

- **natural_language_query**: Origen en castellano.
- **metrics**: Listado de métricas.
- **dimensions**: Listado de dimensiones.
- **start_date**: Fecha de inicio del periodo en el caso de haber generado fechas.
- **end_date**: Fecha de fin del periodo en el caso de haber generado fechas.
- **dimensionFilter**: Filtrado o condición a aplicar sobre las dimensiones.

Dado que buscamos realizar la traducción de castellano a un lenguaje que nos arroje las métricas, dimensiones y filtros acordes a la API de GA4 se ha pensado en procesar estos datos para que las frases a generar sean del tipo siguiente:

get metric1,metric2,metric2 segmentedBy dimension1 filteredBy dimension1 EQUALS value1

```
import re
import json
print(df.dtypes)
df['metrics'] = df['metrics'].apply(lambda x: re.sub(';', ' ', x))
df['dimensions'] = df['dimensions'].fillna('none')
df['dimensions'] = df['dimensions'].apply(lambda x: re.sub(';', ' ', x))
df['dimensionFilter'] = df['dimensionFilter'].fillna('none')
df['dimensionFilter'] = df['dimensionFilter'].astype('str')


def formatFilter(x):
  if x != 'none':
    res = json.loads(x)
    return 'filteredBy ' + res['filter']['fieldName'] + ' ' + res['filter']['fieldFilter']['match'] + ' ' + str(res['filter']['fieldFilter']['value'])
  else:
    return 'filteredBy none none none'

#df['dimensionFilter'] = df['dimensionFilter'].apply(formatFilter)

df['target'] = 'get ' + df['metrics'] + ' segmentedBy ' + df['dimensions'] + ' ' + df['dimensionFilter']
```

De esta manera tenemos en la columna target del dataframe la frase traducida al formato arriba indicado.

Antes de poder pasar la información al modelo para entrenarlo se ha actuado sobre la frase de entrada y sobre la frase de salida.

Frase de entrada

- Se realiza una normalización NFD para quitar caracteres como tildes o diacríticos.
- Se pasa a minúscula la frase.

- Se sustituye con expresiones regulares todo lo que no sea un carácter alfabético por el carácter vacío.

Frase de salida

Una vez disponemos de la frase en el formato arriba indicado se ha procesado la columna target del dataframe para añadirle al inicio el flag de inicio de secuencia y al final de la misma el flag de finalización de secuencia.

Estos flags son esenciales para la generación de secuencias y la interpretación del texto que se genera.

El flag de **[start]** indicará al modelo el inicio de una secuencia de salida. Este aspecto es fundamental para que el modelo de KerasNLP sepa interpretar correctamente cuándo comenzar a generar una nueva secuencia. Además esta señal ayuda a establecer un contexto inicial para el modelo, permitiéndole generar una secuencia coherente desde el principio.

Por contra, el flag de [end] indica al modelo el final de la secuencia de salida, crucial para que el modelo sepa cuándo detener la generación de tokens. El modelo, al saber cuándo detenerse puede optimizar el uso de recursos de la máquina evitando la generación de tokens innecesarios.

6 Modelización

En este punto tratamos la descripción del modelo o modelos que se van a proceder a implementar para llevar a cabo el objetivo del proyecto.

6.1 Supuestos de modelado

El objetivo de esta sección es identificar y listar las suposiciones necesarias para aplicar la técnica de modelado dado que muchas técnicas de modelado hacen suposiciones específicas sobre los datos, por ejemplo, que todos los atributos tienen distribuciones uniformes o no se permiten valores faltantes.

Al enfrentarnos al problema que queremos resolver suponemos ciertos elementos sobre el dataset que hemos generado.

- Suficiente variabilidad de métricas, dimensiones y filtros para que el modelo sea capaz de entender el significado.
- No se va a captar toda la extensión del lenguaje castellano, por lo que vamos a estar limitados a la hora de poder realizar consultas al modelo.
- Dado que las fechas se pueden consultar de muchas maneras diferentes, para hacerlo compatible con la API se procederá a que el usuario lo seleccione de dos campos de la interfaz como “Fecha Inicio” y “Fecha Fin”

6.2 Diseño de experimento

El objetivo de esta sección es describir cómo se construyen, prueban y evalúan los modelos, comenzando por una descripción del modelado realizado y su relación con los objetivos analíticos del proyecto.

El enfoque principal es desarrollar un modelo que traduzca consultas en lenguaje natural al formato de métricas y dimensiones comprendido por la API de Google Analytics 4 (GA4).

6.2.1 Objetivos Analíticos

El objetivo principal es facilitar el acceso a los datos de Google Analytics 4 a usuarios sin conocimientos técnicos avanzados, permitiendo la consulta de estadísticas a través de un lenguaje natural en castellano. Este objetivo se logra mediante la creación de un modelo Transformer Encoder-Decoder que pueda interpretar y traducir correctamente una variedad suficiente de métricas y dimensiones.

6.2.2 Generación del Dataset

El primer paso en el diseño del experimento fue la creación de un dataset robusto y representativo:

- **Extracción de Datos**: Se realizó web scraping de la documentación oficial de GA4 para obtener una lista completa de métricas y dimensiones.
- **Creación de Pares de Datos**: Se generaron 2,000,000 de pares de consultas en lenguaje natural y sus equivalentes técnicos para GA4. Este proceso implicó la captura de nombres de métricas y dimensiones, así como sus descripciones.

6.2.3 Preprocesamiento de Datos

El preprocesamiento es crucial para asegurar que los datos sean consistentes y útiles para el entrenamiento del modelo:

- **Normalización de Texto**: Conversión de todo el texto a minúsculas y eliminación de caracteres diacríticos mediante normalización NFD.
- **Limpieza de Datos**: Uso de expresiones regulares para eliminar caracteres no alfabéticos.
- **Marcado de Secuencias**: Añadido de marcadores de inicio (`[start]`) y fin (`[end]`) de secuencia a las frases de salida para guiar al modelo durante el entrenamiento.

6.2.4 Construcción del Modelo

Se seleccionó un modelo Transformer Encoder-Decoder debido a su efectividad en tareas de traducción de lenguaje natural:

- **Componentes del Modelo**:
  - **Codificador (Encoder)**: Implementado con `keras_nlp.layers.TransformerEncoder`.
  - **Decodificador (Decoder)**: Implementado con `keras_nlp.layers.TransformerDecoder`.
  - **Embeddings Posicionales**: Implementados con `keras_nlp.layers.TokenAndPositionEmbedding`.

6.2.5 Entrenamiento del Modelo

El proceso de entrenamiento se llevó a cabo en Google Colab Pro, aprovechando los recursos de GPU para manejar grandes volúmenes de datos y reducir los tiempos de cálculo:

- **División de Datos**: Los datos se dividieron en conjuntos de entrenamiento, validación y prueba. La proporción específica no se menciona en el documento, pero típicamente se utilizaría una división de 70% para entrenamiento, 15% para validación y 15% para prueba.
- **Ajuste de Hiperparámetros**: Los hiperparámetros clave, como el número de cabezas de atención y la dimensión de los embeddings, se ajustaron iterativamente para optimizar el rendimiento del modelo.
- **Número de Épocas**: Se determinó un número adecuado de épocas para asegurar que el modelo converja sin sobreentrenarse.

6.2.6 Validación y Evaluación

Durante el entrenamiento, el rendimiento del modelo se evaluó continuamente utilizando el conjunto de validación:

- **Evaluación Continua**: Al final de cada época, se evaluó el modelo en el conjunto de validación para ajustar los hiperparámetros y prevenir el sobreajuste.
- **Evaluación Final**: Una vez finalizado el entrenamiento, el modelo se evaluó utilizando el conjunto de prueba para proporcionar una medida final de su rendimiento. Se alcanzó una precisión del 96% con la configuración óptima de hiperparámetros.

6.2.7 Relación con los Objetivos Analíticos

El proceso de modelado realizado se relaciona directamente con los objetivos analíticos del proyecto. El modelo Transformer Encoder-Decoder desarrollado permite a los usuarios formular consultas en lenguaje natural y obtener resultados precisos sin necesidad de conocimientos técnicos avanzados. Esta capacidad mejora significativamente el acceso a datos analíticos y facilita la toma de decisiones basada en datos en entornos empresariales.

6.2.8 Despliegue del Modelo

Para poner el modelo en producción, se utilizó Google Cloud Platform:

- **Infraestructura**: Implementación del modelo en una máquina virtual con soporte para Flask, permitiendo acceder al modelo a través de una API REST.
- **Monitoreo y Mantenimiento**: Establecimiento de mecanismos para monitorear el rendimiento del modelo y realizar ajustes según sea necesario para asegurar su eficacia continua.

Este enfoque asegura que el modelo no solo sea preciso y eficiente, sino también escalable y fácil de usar en aplicaciones prácticas, cumpliendo así con los objetivos del proyecto de facilitar el acceso a los datos de Google Analytics 4 a través de consultas en lenguaje natural.

6.3 Descripción del modelo

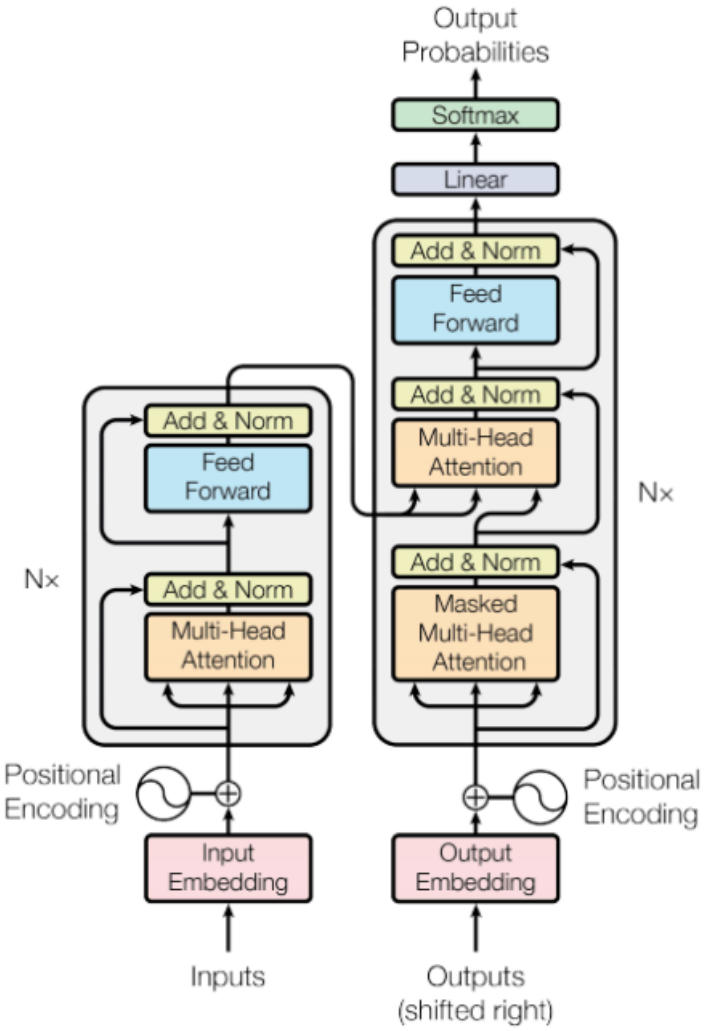
Se ha optado por una solución avanzada de procesamiento de lenguaje natural como son los **Transformers** Encoder-Decoder.

Un transformer es un tipo de RN que basa su su arquitectura en el mecanismo de Atención, concretamente Auto-Atención que permite al modelo **enfocarse en diversas partes de la secuencia de entrada para comprender el contexto global**.

Este tipo de modelos fue introducido por Niki Parmar, Noam Shazeer y Ashis Vaswani en 2017 en el paper “Attention Is All You Need”.

Esta arquitectura pretende **resolver tareas de secuencia a secuencia** al tiempo que maneja dependencias de largo alcance basándose por completo en el mecanismo de self-attention para calcular representaciones de su entrada y salida sin usar RNN ni convolución.

De las tres familias de modelos de transformer para abordar el problema de negocio **vamos a optar por un modelo encoder-decoder orientados a tareas como la traducción** y dado que en nuestro caso se va a traducir desde el lenguaje natural en castellano a un listado de métricas y dimensiones con rangos de fechas consideramos que es la opción más adecuada.



En base a la arquitectura anteriormente explicada **se han identificado 3 elementos básicos** sobre los que plasmar la programación del modelo.

- El **codificador** (keras_nlp.layers.TransformerEncoder).
- El **descodificador** (keras_nlp.layers.TransformerDecoder).
- El **positional embedding** (keras_nlp.layers.TokenAndPositionEmbedding)

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import Model
from tensorflow.keras.layers import Input, Dense, Dropout
from keras_nlp.layers import TokenAndPositionEmbedding, TransformerEncoder
from keras_nlp.layers import TransformerDecoder

np.random.seed(42)
tf.random.set_seed(42)

num_heads = 8
embed_dim = 256

encoder_input = Input(shape=(None,), dtype='int64', name='encoder_input')
x = TokenAndPositionEmbedding(nl_vocab_size, sequence_len, embed_dim)(encoder_input)
encoder_output = TransformerEncoder(embed_dim, num_heads)(x)
encoded_seq_input = Input(shape=(None, embed_dim))

decoder_input = Input(shape=(None,), dtype='int64', name='decoder_input')
x = TokenAndPositionEmbedding(api_vocab_size, sequence_len, embed_dim, mask_zero=True)(decoder_input)
x = TransformerDecoder(embed_dim, num_heads)(x, encoded_seq_input)
x = Dropout(0.4)(x)

decoder_output = Dense(api_vocab_size, activation='softmax')(x)
decoder = Model([decoder_input, encoded_seq_input], decoder_output)
decoder_output = decoder([decoder_input, encoder_output])

model = Model([encoder_input, decoder_input], decoder_output)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary(line_length=120)
```

6.3.1 Necesidades previas

Para configurar adecuadamente el modelo Transformer Encoder-Decoder, es esencial conocer ciertos parámetros que influyen en la preparación y procesamiento de los datos.

Estos parámetros incluyen la longitud máxima de las cadenas de entrada y salida, la longitud máxima de las secuencias, y el tamaño de los vocabularios. A continuación, se detallan estos aspectos y sus implicaciones.

Para establecer correctamente las capas del modelo y optimizar su rendimiento, es fundamental determinar la **longitud máxima de las cadenas de entrada y salida, así como la longitud máxima de las secuencias que el modelo puede manejar**.**Conocer el tamaño de los vocabularios** también es fundamental de cara a definir las capas Embedding. Estas capas (TokenAndPositionEmbedding) **convierten las palabras a tokens** y de tokens a vectores de dimensiones fijas para que el modelo pueda procesarlas.

El número máximo de secuencia de entrada y de salida se usa para determinar cuál es la secuencia máxima en todo el conjunto de frases, se usa principalmente para:

1. **Positional Encoding**

   Estos modelos no tienen una estructura secuencial, por lo que utilizan *positional encoding* para incorporar información sobre la posición de cada token en la

secuencia. Estas codificaciones posicionales se calculan para una longitud máxima de secuencia específica. Esta longitud máxima permite definir las codificaciones posicionales adecuadas para todas las posibles posiciones dentro de la secuencia.

2. Padding y masking

Para procesar lotes de datos de manera eficiente, todas las secuencias en un lote deben ser de la misma longitud. Esto se logra rellenando (padding) las secuencias más cortas hasta la longitud de la secuencia más larga en el lote. Conocer la longitud máxima de secuencia permite establecer de manera uniforme el padding necesario y asegurar que el enmascarado (masking) se aplique correctamente para que el modelo ignore los tokens de relleno.

Para crear este tipo de capas también es necesario **conocer el tamaño del vocabulario** del lenguaje de salida para la última capa del modelo. Una capa densa con activación softmax que necesita conocer el tamaño del vocabulario para definir el número de unidades en esta capa. Esto permite que el modelo asigne una probabilidad a cada posible token de salida.

6.3.2 El codificador (Encoder)

El codificador **es el encargado de procesar la secuencia de entrada y transformarla en una representación interna** que puede ser utilizada por el decodificador para generar la secuencia de salida. Este elemento se compone de diferentes capas encapsuladas de manera eficiente para maximizar el rendimiento del modelo.

¿Cómo funciona?

El codificador se compone de diferentes capas que quedan encapsuladas en la clase de una forma eficiente:

- **Capa Multihead Attention:** Permite al modelo enfocar su atención en diferentes partes de la secuencia de entrada de forma paralela. Calcula las representaciones atendidas de la secuencia de entrada.
- **Capa de Normalización:** Ayuda a la estabilización y aceleración del entrenamiento del modelo. Normaliza las entradas y aplica dropout.
- **Capa Feed-Forward:** Red neuronal totalmente conectada aplicada de forma independiente a cada posición de la secuencia.
- **Conexiones residuales:** Ayuda a la mitigación del desvanecimiento del gradiente para construir redes más profundas. Agrega la entrada original a la salida de cada subcapa para facilitar el entrenamiento.

6.3.3 El decodificador (Decoder)

El decodificador es el **encargado de generar la salida a partir de la salida del codificador y la información procedente de los embeddings de salida**. Tal como sucede con el codificador, este elemento también se compone de diferentes capas con algunas

diferencias orientadas a poder gestionar tanto la entrada ya codificada como la salida generada hasta el momento.

¿Cómo funciona?

Al igual que el codificador, en la clase **TransformerDecoder** se encapsula todo el funcionamiento de un Transformer Decoder. El decodificador en este tipo de modelos se usa para generar secuencias de salida, como en este caso, traducciones.

La arquitectura del Transformer Decoder incluye las siguientes componentes principales:

- **Capa Attention:** Permite que el decodificador preste atención a diferentes partes de la secuencia de entrada. Cada posición de la secuencia de salida generada considera todas las posiciones de la secuencia de entrada para decidir cuál es relevante en cada paso de la generación.
- **Capa Multihead Attention:** Mejora la capacidad del modelo para enfocarse en diferentes partes de la secuencia de entrada desde múltiples perspectivas. Utiliza varias "cabezas" de atención, cada una con sus propias matrices de pesos, lo que permite al modelo captar diferentes tipos de relaciones en los datos.
- **Capa Feed-Forward:** Procesa la información de manera no lineal después de las capas de atención. Consiste en una red neuronal feed-forward que se aplica de manera independiente a cada posición de la secuencia, transformando la representación de la secuencia de entrada en una forma más útil para la tarea de predicción.
- **Conexiones residuales y normalización:** Estabiliza y acelera el entrenamiento del modelo. Las conexiones residuales permiten que la información pase directamente a través de la red, evitando la degradación de la señal en redes profundas. La normalización de capas asegura que los valores numéricos se mantengan dentro de un rango saludable durante el entrenamiento.

6.3.4 El Positional Embedding

El positional embedding se usa como técnica para asignar un vector de embedding único a cada posición en una secuencia. Estos vectores son parámetros que se pueden entrenar, parecidos a los embeddings de palabras. En resumen, además de los embeddings de palabras que únicamente representan el contenido de los tokens, se quiere que el modelo también aprenda representaciones de las posiciones de los tokens en la secuencia.

En el código se puede observar el funcionamiento paso a paso:

- **Paso 1:** Primero se define el rango de posiciones que cobre la longitud máxima esperada de las secuencias de entrada. Es decir, la longitud máxima de las cadenas de entrada.
- **Paso 2:** Se crea una capa de embeddings que asigna un vector a cada posición en dicho rango.

- **Paso 3:** Se suman los embeddings de palabras y los embeddings posicionales. Esta suma actúa como entrada tanto al Encoder como al Decoder con las secuencias de entrada y salida esperada.

7 Evaluación del modelo y resultados

En este punto establecemos el plan para entrenar, probar y evaluar nuestro modelo. Uno de los componentes principales es determinar cómo dividir el conjunto de datos disponible en conjuntos de datos de entrenamiento, validación y prueba.

En el momento de la creación del modelo podemos indicar varios hiper parámetros tanto para su entrenamiento como para su evaluación.

- **Optimizador**: A elegir entre SGD, RMSprop, Adam, AdamW, Adadelta, Adagrad, Adamax, Adafactor, Nadam, Ftrl, Lion o Loss Scale Optimizer.

- **Función de pérdida**: A elegir entre diferentes clases como BinaryCrossentropy, BinaryFocalCrossEntropy, CategoricalCrossentropy, CategoricalFocalCrossentropy, SparseCategoricalCorssentropy, Poisson, KLDivergence o CTC. También a elegir entre funciones como binary_crossentropy, categorical_crossentropy, poisson o kl_divergence.

- **Número de cabezas de atención**: Debemos buscar un valor adecuado ya que con muchas cabezas el modelo se vuelve complejo de entrenar y llevar a problemas de sobreajuste por no mencionar el coste computacional y el uso de memoria (limitada en nuestras máquinas personales).

- **Número de épocas**: Número de ciclos completos a través de todo el conjunto de datos de entrenamiento.

- **Dimensión de los embeddings**: Generalmente con valores < 100 para tareas simples o datos limitados aunque puede no captar bien las relaciones entre tokens. Con valores entre 100-300 para la mayoría de aplicaciones de procesamiento de lenguaje natural. Con valores superiores de 300 usado en aplicaciones avanzadas y en modelos pre-entrenados como BERT o GPT.

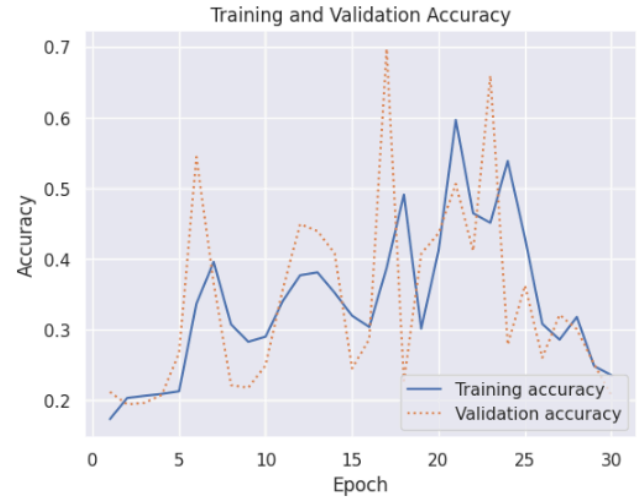
- **Conjunto de datos de validación**: Número decimal entre 0 y 1. Fracción de los datos de entrenamiento a utilizar como datos de validación. El modelo separará esta fracción de los datos de entrenamiento, no se entrenará en ella y evaluará la pérdida y cualquier métrica del modelo en estos datos al final de cada época. Los datos de validación se seleccionan a partir de las últimas muestras de los datos x e y proporcionados, antes de mezclar

7.1 Proceso de evaluación

Los principales hiper parámetros que se han ido ajustando en el modelo han sido el batch size, el número de cabezas de atención y la dimensión de los embeddings. Destacamos que dependiendo del conjunto de datos el haber realizado este proceso de validación y evaluación ha sido complicado por el tiempo de ejecución en Google Collab, por lo que hemos tenido que usar datasets más pequeños.

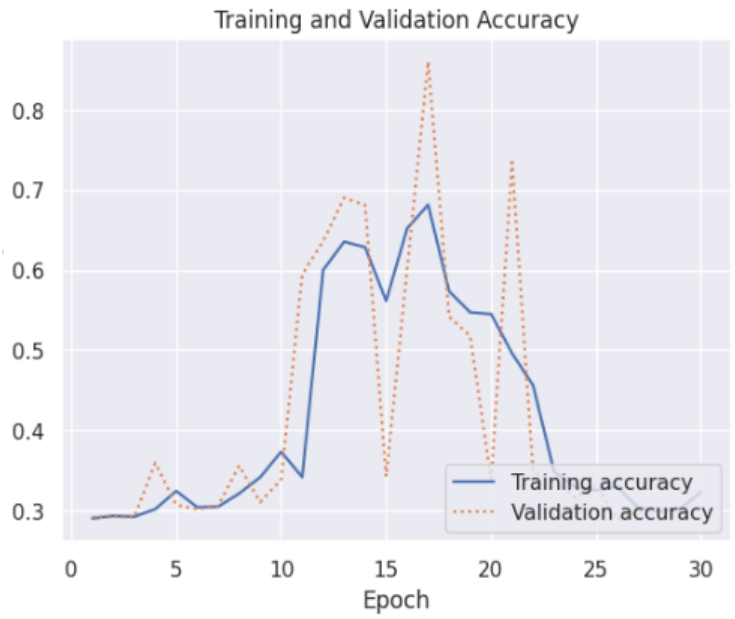
Dejamos a continuación algunos ejemplos que hemos podido extraer debido al excesivo tiempo de ejecución.

[]



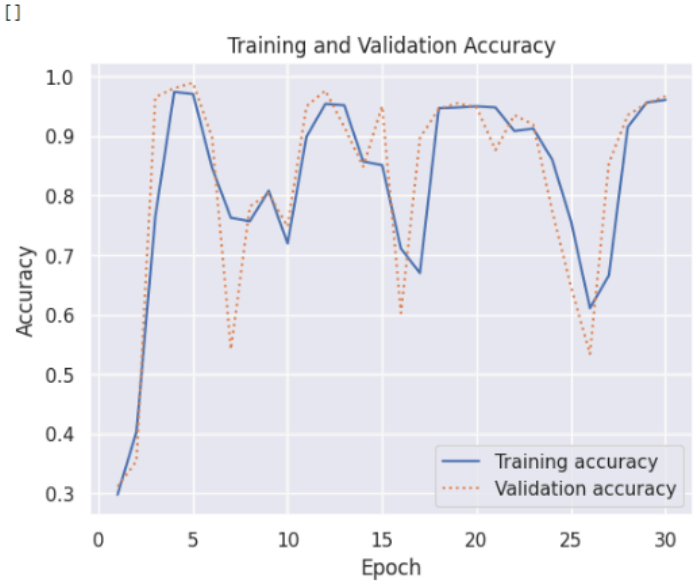
8 cabezas de atención
Embed dim = 256
Batch size = 32
Max accuracy 60%

16 cabezas de atención
Embed dim = 256
Batch size = 32
Max accuracy 85%



8 cabezas de atención
Embed dim = 256
Batch size = 128
Max accuracy 68%

Finalmente nos hemos quedado con el dataset y con los ajustes de esta iteración donde hemos conseguido una accuracy del 96% con 8 cabezas de atención y un *embed dim* de 256.



Este dataset no incluye filtros ni fechas ya que incluyendo estos parámetros el dataset debía ser aún más grande para obtener accuracy aceptable para el problema a resolver.

8 Despliegue

De cara a poner en producción el modelo predictivo que se ha desarrollado se ha optado por **Google Cloud Platform**.

Antes de poner en producción el modelo para poder consultarlo en tiempo real ha habido que guardarlo con la funcionalidad save que pone a nuestra disposición Keras. De esta manera al guardarlo se dispone de un archivo comprimido que podemos utilizar en cualquier entorno compatible a nivel tecnológico con las correspondientes librerías instaladas.

Para la POC que nos ocupa se ha hecho uso de una máquina virtual en el servicio de Compute Engine de Google Cloud Platform, máquina virtual creada desde el servicio Compute Engine.

Google Cloud Platform nos permite crear máquinas virtuales con la mayoría de librerías necesarias e incluso contratar una o varias GPUs.

8.1 Creando el endpoint en Google Cloud Platform

Indicamos paso a paso el proceso seguido para desplegar el modelo:

1. Creación de **máquina virtual en Google Cloud Platform** en Compute Engine.
2. Acceso a la **máquina virtual a través de protocolo SSH** para instalar librerías necesarias para explotar el modelo:
    a. Flask: Para crear el servidor que va a servir el modelo a través de un endpoint.
    b. Keras y KerasNLP: Librerías de procesamiento natural de Keras.
3. Abrir el **puerto 5000** de la máquina virtual para que acepte peticiones HTTP externas.
4. **Subir el modelo** anteriormente guardado a la raíz del servidor.
5. Crear el script que leerá el modelo e **inicialización del servidor con Flask**.

Este script es accesible desde el repositorio.

En el servidor, tal como se muestra en la imagen siguiente se dispone del archivo **deep.py** que es el encargado de levantar el servidor y el fichero **modelo.keras** que es el modelo anteriormente guardado desde Google Collab.

```
(base) juanignacioalberola@keras-vm:~$ ls -al
total 90024
drwxr-xr-x 7 juanignacioalberola juanignacioalberola     4096 Jul  8 15:30 .
drwxr-xr-x 5 root                root                    4096 Jun  9 15:05 ..
-rw------- 1 juanignacioalberola juanignacioalberola     1117 Jun 24 09:24 .bash_history
-rw-r--r-- 1 juanignacioalberola juanignacioalberola      220 Mar 27  2022 .bash_logout
-rw-r--r-- 1 juanignacioalberola juanignacioalberola     3957 Jun  6 16:17 .bashrc
drwxr-xr-x 3 juanignacioalberola juanignacioalberola     4096 Jun  6 16:18 .cache
drwxr-xr-x 3 juanignacioalberola juanignacioalberola     4096 Jun  6 16:17 .config
drwxr-xr-x 2 juanignacioalberola juanignacioalberola     4096 Jun  6 16:17 .docker
drwxr-xr-x 2 juanignacioalberola juanignacioalberola     4096 Jun  6 16:18 .keras
-rw-r--r-- 1 juanignacioalberola juanignacioalberola      807 Mar 27  2022 .profile
-rw------- 1 juanignacioalberola juanignacioalberola      130 Jun 24 09:15 .python_history
drwx------ 2 juanignacioalberola juanignacioalberola     4096 Jul  8 15:28 .ssh
-rw-r--r-- 1 juanignacioalberola juanignacioalberola     4158 Jun 24 09:20 deep.py
-rw-r--r-- 1 juanignacioalberola juanignacioalberola 66844260 Jun 24 09:10 modelo.keras
-rw-r--r-- 1 juanignacioalberola juanignacioalberola 25278100 Jun 24 09:18 output_subset.csv
(base) juanignacioalberola@keras-vm:~$
```

El archivo deep.py tiene todo el preprocesamiento antes de entrenar el modelo en Collab, pero en lugar de entrenarlo llamamos a la función **load_model** para poder cargarlo y usar la función **translate_text** con el cálculo de los parámetros tal como se hizo antes del entrenamiento.

En el mismo archivo se **crea un endpoint, que en este caso se ha llamado translate que recibe peticiones HTTP del tipo POST y una vez recibidas llama al modelo con el texto de la solicitud para devolverlo al cliente**.

8.2 Consumiendo el modelo

Las peticiones al endpoint para consumir el modelo se pueden realizar desde cualquier lenguaje de programación, aunque en nuestro caso se ha optado por crear la interfaz de usuario con REFLEX que usa código Python para generar interfaces web.

Desde la aplicación que presenta la interfaz al usuario se ha hecho una llamada al endpoint del servidor pasando los parámetros necesarios para realizar la traducción, como la cadena a traducir, las fechas de inicio y fin para recuperar los datos y la propiedad a consultar.

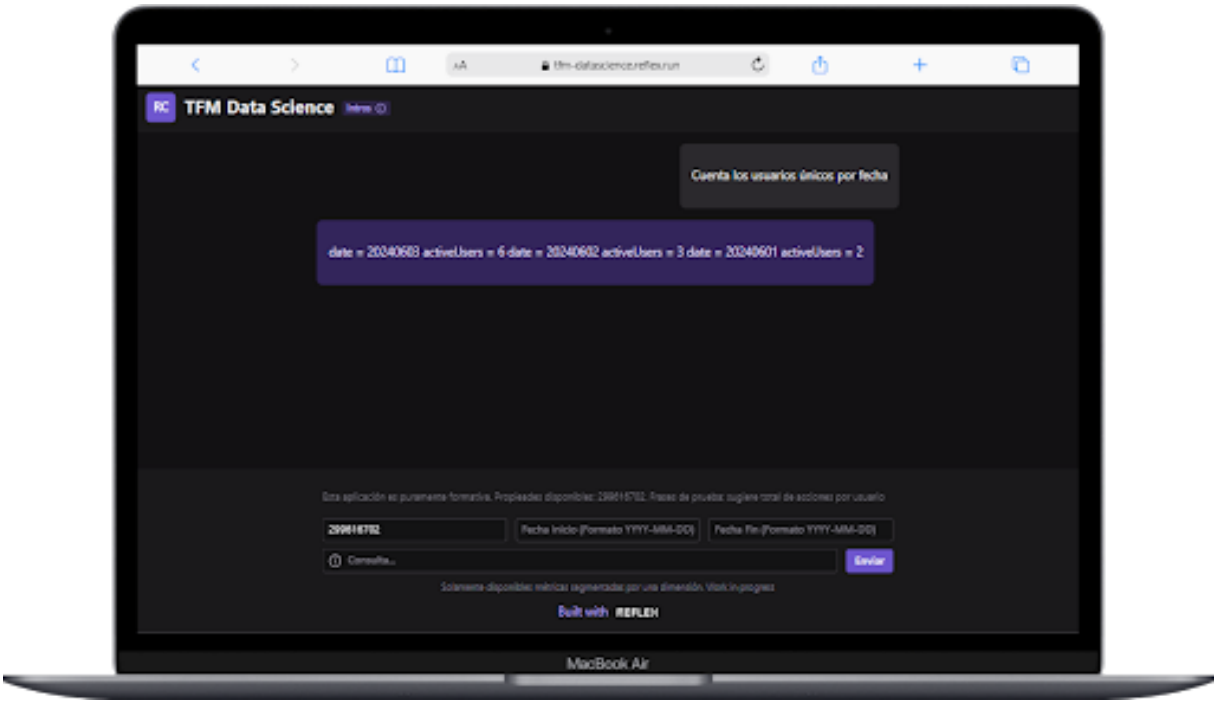
```
endpoint_url = "http://35.247.85.124:5000/translate"
data = {
    "text": question,
    "propiedad": "properties/" + propiedad,
    "inicio": inicio,
    "fin": fin
}
```

No se ha hecho un control de errores ya que la interfaz solamente se ha creado como ejemplo de las posibilidades de presentar un entorno para que los comerciales realicen consultas.

El servidor nos devuelve los resultados de realizar la consulta a la API de GA4 con los parámetros enviados en la petición.

9. Puesta en valor

La puesta en valor inicial en la empresa se hará mediante una interfaz web accesible por parte de los miembros de la empresa a los que se les otorgue permisos. La interfaz es accesible desde https://tfm-datascience.reflex.run/ pero la funcionalidad de traducción no estará disponible si el servidor con el endpoint no está encendido. **Generalmente lo tenemos apagado porque el consumo de la máquina virtual genera una facturación real por la que Google nos cobra por lo que si se usa la interfaz con el servidor apagado devolverá error.**



9.1 Integración de resultados analíticos en procesos empresariales

El objetivo principal de este proyecto es facilitar el acceso a los datos de Google Analytics 4 (GA4) para usuarios no técnicos mediante la traducción de consultas en lenguaje natural a consultas de la API. Para poner en valor los resultados analíticos obtenidos con esta herramienta, se proponen las siguientes estrategias de integración en los procesos empresariales:

1. Acceso Directo a Datos para Toma de Decisiones:

La herramienta permitirá a los profesionales de marketing, gerentes y otros usuarios no técnicos acceder directamente a los datos de GA4. Esto facilitará la toma de decisiones basada en datos en tiempo real, sin depender de analistas de datos.

2. Automatización de Informes:

La capacidad de traducir consultas en lenguaje natural a consultas de la API permitirá la automatización de la generación de informes. Los usuarios podrán

obtener informes personalizados simplemente formulando preguntas en lenguaje natural, lo que reducirá el tiempo y esfuerzo necesarios para la preparación de reportes.

3. Optimización de Campañas de Marketing:

Los equipos de marketing podrán utilizar la herramienta para analizar el rendimiento de sus campañas en GA4. Al poder realizar consultas específicas sobre métricas y dimensiones, podrán identificar rápidamente qué estrategias están funcionando y cuáles necesitan ajustes.

4. Mejora en la Experiencia del Cliente:

Al tener acceso a datos detallados sobre el comportamiento de los usuarios, las empresas podrán personalizar y mejorar la experiencia del cliente en sus plataformas digitales. Las consultas en lenguaje natural facilitarán la obtención de insights sobre patrones de uso, preferencias y problemas que enfrentan los usuarios.

5. Reducción de la Dependencia de Equipos Técnicos:

La herramienta reducirá la dependencia de los equipos de análisis de datos y tecnología de la información, permitiendo que los equipos de negocio sean más autosuficientes. Esto liberará recursos técnicos para que puedan enfocarse en proyectos más complejos y estratégicos.

9.2 Estrategias para la Adopción de la Herramienta

Para asegurar la adopción y uso efectivo de la herramienta en la empresa, se proponen las siguientes estrategias:

1. Capacitación y Formación:

Se organizarán sesiones de capacitación para los usuarios finales, mostrando cómo formular consultas en lenguaje natural y cómo interpretar los resultados obtenidos. Además, se proporcionará documentación detallada y tutoriales en línea.

2. Soporte Continuo:

Se establecerá un equipo de soporte dedicado para asistir a los usuarios en la resolución de problemas y responder a sus preguntas. Este equipo también recopilará feedback para futuras mejoras de la herramienta.

3. Promoción Interna:

Se llevarán a cabo campañas de comunicación interna para promover el uso de la herramienta, destacando los beneficios y casos de éxito dentro de la empresa. Se incentivará a los usuarios a compartir sus experiencias y mejores prácticas.

4. Iteración y Mejora Continua:

Con base en el feedback de los usuarios, se realizarán mejoras continuas en la herramienta. Esto incluirá la actualización del modelo NLP, la adición de nuevas funcionalidades y la mejora de la interfaz de usuario.

9.3 Posibles mejoras

1. Determinación del periodo de tiempo:

En este TFM, para simplificar el proceso de consulta y reducir la complejidad semántica, hemos incorporado casillas en la interfaz donde el usuario debe fijar la fecha inicial y final de la consulta.

Esta decisión se toma con el propósito de reducir la variabilidad del input, permitiendo que el modelo se entrene mejor con un conjunto de datos más controlado y aumentando la precisión.

Aunque en un entorno ideal sería conveniente que el modelo entendiera consultas temporales en lenguaje natural, la restricción a un rango de fechas específico permite una mayor precisión y eficiencia en el contexto de un TFM con tiempo y recursos limitados. En un entorno empresarial, se podría expandir el modelo para que comprenda una mayor variedad de expresiones temporales, mejorando aún más la usabilidad y flexibilidad de la herramienta.

2. Gestión de respuestas con alta separación de coeficientes:

Durante el desarrollo del TFM, cuando el coeficiente de separación de la respuesta sea mayor a un umbral establecido, el sistema mostrará un mensaje de error pidiendo al usuario que reformule su pregunta.

Esta medida evita presentar respuestas potencialmente incorrectas que podrían llevar a decisiones de negocio erróneas o generar desconfianza en el modelo.

En un entorno empresarial con recursos para seguimiento y mejora continua, estas consultas "erróneas" se remitirían a un equipo técnico. Este equipo analizaría la consulta y proporcionaría la respuesta correcta, utilizando este proceso para mejorar y entrenar continuamente el modelo. Este enfoque no solo incrementa la precisión del sistema con el tiempo, sino que también garantiza que los usuarios reciban respuestas exactas, fortaleciendo la confianza en la herramienta y promoviendo su uso.

9.4 Monetización y Comercialización

Tras una implementación exitosa en la empresa, se explorarán oportunidades para comercializar la herramienta a otras empresas y organizaciones. Las estrategias para la monetización incluirán:

1. Modelo de Suscripción:

Se ofrecerá la herramienta como un servicio de suscripción, con diferentes planes según las necesidades y el tamaño de las empresas. Los planes podrán incluir acceso a funcionalidades avanzadas, soporte prioritario y personalización.

2. Partnerships y Alianzas:

Se buscarán alianzas con agencias de marketing digital y consultoras de datos que puedan revender la herramienta a sus clientes. Estas asociaciones ayudarán a ampliar el alcance y la base de usuarios.

3. Versión Freemium:

Se considerará el lanzamiento de una versión freemium de la herramienta, donde los usuarios puedan acceder a funcionalidades básicas de forma gratuita, con la opción de pagar por características premium.

4. Marketing y Publicidad:

Se desarrollará una estrategia de marketing digital para promover la herramienta en redes sociales, blogs especializados y plataformas de software empresarial. Esto incluirá la creación de contenido educativo y demostraciones en vivo.

9.5 Métricas de Éxito

Para evaluar el éxito de la herramienta, se establecerán las siguientes métricas:

1. Tasa de Adopción:

Porcentaje de usuarios finales que utilizan la herramienta regularmente.

2. Satisfacción del Usuario:

Evaluaciones de satisfacción a través de encuestas y feedback directo.

3. Impacto en la Toma de Decisiones:

Número de decisiones informadas basadas en datos obtenidos con la herramienta.

4. Retorno de la Inversión (ROI):

Comparación entre los costos de desarrollo y mantenimiento de la herramienta y los beneficios obtenidos.

5. Ingresos por Suscripciones:

Generación de ingresos a partir de la comercialización de la herramienta.

10. Problemas detectados

Limitaciones computacionales
Ha resultado complicado poder trabajar con los datasets generados en entornos locales, por ello hemos tenido que contratar Google Collab PRO para disponer de más capacidad GPU, más memoria y más unidades de computación. Asimismo para el despliegue del modelo hemos tenido que usar servicios de pago como Compute Engine de GCP.

Variedad del lenguaje
El proyecto está realizado para que se traduzca desde castellano, idioma que tiene una amplia variedad en el lenguaje para realizar las consultas. Se ha hecho complicado el, a través de sinónimos, el poder representar una pequeña parte de todas las formas en las cuales realizar las solicitudes. Esto nos ha hecho pensar que quizá no sea la mejor tecnología para resolver este tipo de problemas y que a futuro puede interesar usar las actuales LLMs.

Validación del modelo
Trabajar con este tipo de modelos en las máquinas personales se hace complicado al necesitar alta capacidad de procesamiento cuando los datasets son elevados. Debido a estas dificultades no hemos podido realizar un ajuste más amplio de hiper parámetros para el dataset con 1.000.000 de filas que se ha usado para el entrenamiento sino que hemos tenido que usar un dataset más pequeño para ilustrar en el TFM el proceso de optimización.

11. Contribuciones

Durante el desarrollo del proyecto cada miembro del equipo ha participado en todas las áreas del mismo en mayor o menor medida.

- Álvaro Salmerón:
  - Elección de tecnología para interfaz.
  - Scraping de documentación de GA4 para extracción de métricas y dimensiones.
  - Generación de sinónimos.
  - Ejemplos de llamadas a la API de GA4 con Python.
  - Elaboración del presente documento.

- Daniel Urrutxua:
  - Script generación datasets.
  - Mantenimiento repositorio en Github.
  - Elaboración del presente documento.
  - Necesidades del negocio.

- Alberto Sebastián:
  - Necesidades del negocio.
  - Puesta en valor de la herramienta.
  - Generación de sinónimos para alimentar el generador de datasets.
  - Documento PTT para presentación del proyecto.
  - Elaboración del presente documento.

- Juan Ignacio Alberola:
  - Necesidades del negocio.
  - Creación y validación del modelo.
  - Montaje de máquina virtual en Google Cloud Platform.
  - Desarrollo del endpoint en Compute Engine.
  - Programación de las llamadas a la API desde la interfaz pública.
  - Documento PPT para presentación del proyecto.
  - Elaboración del presente documento.

12. Bibliografía

- Neural Machine Translation with Transformers.
- Neural Machine Translation with LSTM.
- Architectural Patterns for Text-to-SQL: Leveraging LLMs for Enhanced BigQuery Interactions
- Generate Any SQL Query Using AI-Powered Text2SQL Bot
- Text-to-SQL Benchmarks and the Current State-of-the-Art
- A survey on deep learning approaches for text-to-SQL
- Natural Language Query to SQL Conversion Using Machine Learning Approach
- Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning
- Translating Natural Language Queries to SQL Using the T5 Model
- [GA4] Analytics dimensions and metrics
- Google Analytics 4 Data API Overview
- A Guide to Google Analytics 4 API with Python
- ChatGPT