Your allies for change

Investigación sobre RAG Multimodal con Gemini

Exploración práctica del Modelo “Gemini 2 Flash Exp”

# Introducción

## Contexto y motivación

En la investigación anterior sobre la multimodalidad en el modelo Gemini, se realizaron pruebas sobre su capacidad de procesamiento de voz a voz, evaluando su capacidad de comprensión del habla, tiempos de inferencia y latencia. Se concluyó que el modelo tiene una alta precisión en el reconocimiento de voz y una velocidad de inferencia competitiva.

Dado que Gemini es un modelo multimodal, una de las siguientes etapas naturales en esta investigación es la integración de un sistema **RAG (Retrieval-Augmented Generation)**. Este enfoque permitiría restringir y contextualizar las respuestas del modelo dentro de un dominio específico, mejorando la precisión y relevancia de sus respuestas.

## Objetivos de la investigación

El objetivo principal de este estudio es explorar la implementación de un sistema RAG sobre el modelo Gemini con capacidades multimodales, evaluando distintas configuraciones de entrada y salida para determinar su viabilidad en escenarios reales. Para ello, se seguirá una metodología incremental que analizará tres niveles de complejidad:

1. **Entrada multimodal (imagen y texto) con salida en texto**: Se probará la capacidad de Gemini para gestionar grandes contextos (hasta 1M de tokens) y se evaluará su rendimiento en la generación de respuestas textuales basadas en datos externos.
2. **Entrada en texto y salida en audio**: Se analizará la calidad del audio generado, tiempos de inferencia y limitaciones en la producción de contenido auditivo.
3. **Implementación de un sistema RAG con entrada y salida en voz**: Se integrará un sistema de recuperación de información para alimentar a Gemini con datos específicos, permitiendo la generación de respuestas coherentes y precisas dentro de un dominio predefinido.

## Estructura del informe

Este documento se organiza en las siguientes secciones:

* **Configuración y entorno de pruebas**: Se describirán las herramientas utilizadas, el acceso a la API de Gemini y los criterios de evaluación.
* **Prueba inicial: Entrada multimodal y salida en texto**: Se analizará la capacidad del modelo para procesar texto e imágenes, su límite de tokens y la precisión en la respuesta.
* **Segunda prueba: Entrada en texto y salida en audio**: Se estudiará la calidad de la respuesta auditiva y su latencia.
* **Implementación de RAG con entrada y salida en voz**: Se presentará el desarrollo del sistema RAG y su impacto en la generación de respuestas.
* **Conclusiones y próximos pasos**: Se sintetizarán los hallazgos y se sugerirán mejoras para futuras iteraciones.

# Configuración y entorno de pruebas

## Requerimientos previos

Para llevar a cabo las pruebas, es necesario instalar las dependencias requeridas mediante el siguiente comando:



Además, se necesita una API Key de acceso a Gemini, que se puede obtener desde [Google AI Studio](https://aistudio.google.com).

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Una vez obtenida, esta clave debe configurarse en las variables de entorno o en el código fuente para la autenticación de las peticiones a la API.

## Metodología de pruebas

La metodología utilizada en esta investigación sigue un enfoque experimental básico, en el que se evaluará el rendimiento del modelo a través de pruebas sucesivas con distintos tipos de entrada y salida. Cada prueba se ejecutará varias veces para detectar patrones de comportamiento y posibles inconsistencias en las respuestas generadas. Los principales criterios de evaluación serán:

* **Precisión de la respuesta**: Análisis cualitativo de las respuestas generadas por el modelo.
* **Tiempo de inferencia**: Medición del tiempo que tarda el modelo en generar una salida.
* **Latencia en distintas condiciones de conexión**: Evaluación del impacto de una red inestable en el rendimiento del modelo.
* **Capacidad de retención de contexto**: Verificación del manejo de grandes cantidades de información en la sesión.
* **No abandonar el contexto**: Se realizaran preguntas y técnicas de HackLLM para ver si se sigue manteniendo en el contexto del RAG.

Estas pruebas permitirán establecer una línea base de rendimiento antes de integrar el sistema RAG en las siguientes fases de la investigación.

# Prueba inicial: Entrada multimodal (imagen y texto) y salida en texto

El siguiente código prueba la capacidad de Gemini para manejar grandes contextos, en este caso cargando un libro completo en formato texto y permitiendo consultas sobre su contenido:

Texto

El contenido generado por IA puede ser incorrecto.

Este simple rueba demuestra que Gemini es capaz de manejar un contexto extenso (hasta 1M tokens), permitiendo realizar preguntas sobre el contenido cargado y obtener respuestas precisas.

Texto

El contenido generado por IA puede ser incorrecto.

## Conclusiones

* Este es el sistema clásico de RAG, el cual también es compatible con LangChain, lo que facilita su integración con flujos de trabajo ya existentes y su adaptación a múltiples casos de uso.
* La gran capacidad de contexto permite manejar chunks muy grandes en los embeddings, eliminando la necesidad de dividir documentos en fragmentos pequeños y preservando mejor la coherencia de la información.
* La ventana de contexto de 1M tokens es un punto disruptivo, ya que permite trabajar con documentos completos sin recurrir a estrategias de resumen o truncamiento, lo que mejora la precisión de las respuestas.
* Este avance abre nuevas posibilidades en la recuperación de información, ya que los modelos pueden operar con bases de conocimiento más extensas sin necesidad de recurrir a técnicas tradicionales de reducción de contexto.
* La combinación de Gemini con RAG basado en grandes embeddings permite optimizar el procesamiento de datos y mejorar la relevancia de las respuestas, lo que lo convierte en una alternativa robusta para aplicaciones avanzadas de IA conversacional y análisis de documentos extensos.

# Segunda prueba: Entrada en texto y salida en audio

## Implementación de RAG para recuperación de contexto

Para mejorar la precisión del sistema en la generación de respuestas, se implementó una clase Rag, encargada de gestionar y recuperar fragmentos relevantes de un documento extenso. Su funcionamiento se basa en embeddings de texto y en la similitud coseno para identificar las secciones más relevantes en relación con una consulta. (Esto es para las pruebas, lógicamente aquí tendría que implementarse y hacer pruebas con lo que se tiene de GPTup)

## Explicación detallada de la clase Rag

Texto

El contenido generado por IA puede ser incorrecto.

Se importan las bibliotecas necesarias:

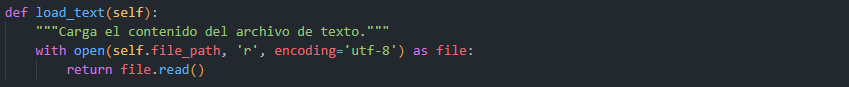
* SentenceTransformer: Para generar embeddings de los fragmentos de texto.
* numpy: Para manejar operaciones matriciales.
* cosine\_similarity de sklearn.metrics.pairwise: Para calcular la similitud entre la consulta y los fragmentos de texto.

### Inicialización de la clase



* file\_path: Ruta del archivo de texto a procesar.
* model\_name: Modelo de embeddings preentrenado, por defecto 'all-MiniLM-L6-v2'.
* max\_length: Número máximo de palabras por fragmento, definido en 500 palabras.

### Métodos principales



Carga el texto completo desde un archivo.

Texto

El contenido generado por IA puede ser incorrecto.

Este método se encarga de segmentar el texto completo en fragmentos manejables de hasta max\_length palabras, lo que permite un procesamiento más eficiente y optimizado para la generación de embeddings. La segmentación se realiza dividiendo el texto en listas de palabras y agrupándolas en bloques consecutivos, asegurando que cada fragmento conserve el contexto necesario para mantener la coherencia semántica. Esta estrategia permite manejar textos largos sin perder información clave y mejora el rendimiento en la recuperación de información relevante en consultas posteriores.

Texto

El contenido generado por IA puede ser incorrecto.

Este método toma cada fragmento de texto previamente dividido y genera su correspondiente representación numérica utilizando el modelo de embeddings preentrenado. Los embeddings son vectores de alta dimensión que capturan el significado semántico del texto, lo que permite comparar fragmentos basándose en su similitud contextual. Esto facilita la búsqueda de los fragmentos más relevantes en función de una consulta posterior. Los embeddings generados se almacenan en una matriz numpy para su procesamiento eficiente en cálculos posteriores

Texto

El contenido generado por IA puede ser incorrecto.

Dado un query, encuentra los top\_k fragmentos más relevantes usando similitud coseno, lo que permite identificar los fragmentos de texto más alineados con la consulta del usuario.

* **Codificación de la consulta**: Se convierte la consulta ingresada en un vector de embedding utilizando el modelo SentenceTransformer.
* **Comparación con los fragmentos indexados**: Se calcula la similitud coseno entre la consulta embebida y cada uno de los fragmentos previamente almacenados en self.embeddings.
* **Selección de los fragmentos más relevantes**: Se identifican los top\_k fragmentos con mayor similitud, es decir, aquellos que tienen la distancia angular más pequeña respecto a la consulta.
* **Retorno de los fragmentos**: Los fragmentos seleccionados son devueltos como una lista de strings, representando la información más relevante respecto a la consulta dada.

### Generación de audio a partir del contexto

Una vez implementada la clase Rag, se integró en el código de procesamiento de audio, al principio se creo el objeto rag y automáticamente lee y parte el fichero en chunks.



Y posteriormente le integramos en el código de la investigación anterior:

Texto

El contenido generado por IA puede ser incorrecto.

Y el mensaje con contexto se lo pasamos al método send que lo envía al modelo LLM

Una captura de pantalla de un celular con texto e imagen

El contenido generado por IA puede ser incorrecto.

# Implementación de RAG con entrada y salida en voz

Integración de un sistema de recuperación de información.

Configuración de entrada por voz.

Procesamiento de la respuesta en voz y análisis de latencia.

Evaluación de la utilidad en un entorno real.

**6. Conclusiones y próximos pasos**

Comparación de los diferentes enfoques probados.

Limitaciones encontradas en cada modalidad.

Posibles mejoras o siguientes iteraciones.

Reflexión sobre el potencial de Gemini 2 Flash Exp en RAG multimodal.