Your allies for change

Investigación en Multimodalidad en LLM

Enfoque practico de modelos Multimodales Gemini 2 y Azure OpenAI RealTime

# Introducción

* Breve descripción de la importancia de la multimodalidad en los modelos de lenguaje.
* Definición de multimodalidad: capacidad de procesar y generar múltiples tipos de datos (texto, imágenes, audio, video).
* Objetivo del documento: explorar cómo los modelos en la nube, como Gemini y OpenAI, abordan la multimodalidad desde una perspectiva práctica.
* Relevancia de los modelos en la nube para el desarrollo y despliegue de aplicaciones multimodales.

# Contexto general de la multimodalidad en LLM

La multimodalidad se refiere a la capacidad de los modelos de aprendizaje automático para procesar e integrar datos de diferentes dominios o modalidades, tales como texto, imágenes, audio y video. En el contexto de los Large Language Models (LLM), la multimodalidad abre posibilidades para una comprensión y generación de contenido más rica, al combinar la fortaleza del modelado del lenguaje con la capacidad de interpretar y relacionar múltiples tipos de información.

## Evolución de los modelos multimodales

A lo largo de la historia de la inteligencia artificial, los primeros esfuerzos se centraron en desarrollar modelos capaces de procesar un único tipo de información (unimodales). Por ejemplo, se entrenaban redes neuronales para realizar tareas de clasificación de texto o reconocimiento de imágenes de manera independiente. Sin embargo, con los avances en arquitectura de redes neuronales y la disponibilidad de grandes volúmenes de datos, emergió la necesidad de combinar la información de distintas fuentes para lograr un entendimiento más completo y contextual.

### De lo unimodal a lo multimodal

* Los primeros modelos de lenguaje se enfocaban exclusivamente en texto (por ejemplo, Word2Vec, GloVe). Paralelamente, en visión por computadora surgieron modelos capaces de etiquetar imágenes o detectar objetos (p. ej., AlexNet, VGG).
* Con el aumento en la capacidad de cómputo y la aparición de técnicas de aprendizaje profundo, se fueron desarrollando enfoques para unificar redes de texto e imagen, aprovechando la información que cada modalidad aporta.

### Avances recientes en multimodalidad

* **CLIP (Contrastive Language-Image Pretraining)**: Desarrollado por OpenAI, CLIP entrena conjuntamente modelos de lenguaje y de visión para alinear el espacio semántico de texto con el de imágenes, permitiendo relacionar descripciones textuales con su correspondiente contenido visual.
* **DALL·E**: También de OpenAI, DALL·E y su sucesor DALL·E 2 generan imágenes a partir de descripciones de texto, demostrando la capacidad de los modelos para “entender” y “crear” representaciones visuales a partir de una instrucción textual.
* **Flamingo**: Este modelo, creado por DeepMind, está diseñado para manejar diálogos multimodales en contexto, integrando tanto lenguaje natural como componentes visuales para responder de manera coherente y contextualizada.

## Beneficios de la Multimodalidad

La integración de múltiples modalidades en los modelos de lenguaje ha traído consigo numerosas ventajas en términos de capacidad de entendimiento, aplicabilidad y robustez de las soluciones de inteligencia artificial.

### Mejora en la comprensión contextual

Al combinar información visual y textual (o de otras modalidades, como audio), los modelos pueden desarrollar un entendimiento más profundo del contexto. Por ejemplo, al incluir imágenes, un LLM puede discernir aspectos semánticos que el texto por sí solo no aclara, mejorando la precisión en la generación de respuestas y la coherencia en tareas de razonamiento.

### Aplicaciones prácticas en distintos sectores

* Comercial …
* Industrial …
* **Educación**: Se pueden crear herramientas interactivas que, además de proveer explicaciones y guías en texto, incluyan elementos visuales o auditivos para mejorar la experiencia de aprendizaje y adaptarse a diferentes estilos cognitivos.
* **Salud**: Los modelos multimodales ayudan a procesar tanto informes clínicos (texto) como datos de estudios de imagen (radiografías, resonancias) para apoyar el diagnóstico y seguimiento de pacientes.

### Desafíos asociados

* **Procesamiento eficiente**: El incremento en la complejidad de los datos multimodales demanda más recursos de cómputo y optimizaciones en la arquitectura de redes neuronales.
* **Integración de datos heterogéneos**: Combinar distintas fuentes de información con formatos, resoluciones y dominios dispares requiere desarrollar técnicas de preprocesamiento y representación adecuadas, a fin de unificar la información sin perder calidad.

# Análisis de Gemini (Google)

## Introducción a Gemini

**Gemini** es un modelo multimodal que destaca por su capacidad de procesar y generar contenidos en diferentes formatos (texto, audio y video) de manera simultánea y en tiempo real. Para habilitar estas interacciones, Google ha desarrollado la **API de Multimodal Live**, la cual permite:

* **Interacciones de voz y video bidireccionales de baja latencia**: Los usuarios pueden mantener conversaciones de voz naturales y fluidas con el modelo, interrumpirlo en cualquier momento e, incluso, combinar la comunicación de texto y video según la necesidad.
* **Conversaciones “human-like”**: Su arquitectura está diseñada para ofrecer respuestas que imitan la dinámica de las interacciones humanas, dando pie a experiencias más inmersivas.

Gracias a estas funcionalidades, Gemini abre la puerta a una variedad de usos prácticos que van desde asistentes virtuales conversacionales hasta plataformas de educación y entretenimiento en tiempo real.

## Características Multimodales de Gemini

La **API de Multimodal Live** incorpora una serie de funciones clave que convierten a Gemini en un modelo sumamente versátil:

* **Multimodalidad**

El modelo puede “ver, escuchar y hablar”, lo que significa que integra de forma nativa texto, audio y video en sus procesos de entrada y salida.

* **Interacción en tiempo real de baja latencia**

Permite entregar respuestas rápidas, factor esencial en aplicaciones donde la inmediatez es prioritaria, como la atención al cliente o la formación en entornos virtuales.

* **Memoria de sesión**

Durante una sola sesión, Gemini retiene la información de interacciones previas, facilitando el seguimiento del contexto y evitando la repetición innecesaria de datos.

* **Compatibilidad con llamadas a funciones, ejecución de código y búsqueda externa**

Ofrece la capacidad de conectarse con APIs de terceros, ejecutar rutinas específicas o consultar fuentes externas, ampliando de forma significativa el espectro de posibilidades para el desarrollo de aplicaciones integradas.

* **Detección automática de actividad de voz (VAD)**

Reconoce de manera precisa el inicio y el final de la voz del usuario, fomentando conversaciones fluidas y permitiendo la interrupción del modelo cuando sea necesario.

## Uso Práctico de Gemini

Actualmente, Google ofrece la posibilidad de explorar **Gemini** y sus capacidades multimodales a través de plataformas como [**aistudio.google.com**](https://aistudio.google.com/).

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Sin embargo, el enfoque de esta investigación no se limita a la prueba directa en la plataforma oficial, sino que se orienta a:

* **Integrar Gemini en aplicaciones existentes**, como **GPTup o AIsQ**, para potenciar la experiencia conversacional. Esta integración permitiría a GPTup aprovechar la multimodalidad de Gemini y ofrecer a sus usuarios una interacción más rica (voz, video y texto), junto con mejores capacidades de contextualización.
* **Implementar y probar el modelo usando código Python** de forma local o en entornos de desarrollo personalizados. Aunque aquí no profundizaremos en la codificación en sí, el objetivo es mostrar cómo Gemini puede combinarse con aplicaciones ya creadas para proporcionar respuestas rápidas y fluidas, ejecutar tareas específicas, consultar fuentes externas y adaptarse a distintas necesidades de negocio o investigación.

## Implementación y Validación del Modelo en Desarrollo

En esta sección se aborda tanto la base teórica como la aplicación práctica de las tecnologías clave que permiten la integración de Gemini en entornos de desarrollo. Concretamente, se describe el uso de la **API de Multimodal Live**, la cual funciona sobre un modelo de comunicación con estado que se apoya en **WebSockets** para habilitar interacciones multimodales en tiempo real.

### Fundamentos de la API de Multimodal Live y WebSockets

La **API de Multimodal Live** proporciona un canal de comunicación bidireccional y en tiempo real con Gemini, permitiendo a la aplicación cliente enviar y recibir datos de texto, audio o video sin bloqueos ni demoras innecesarias.

* **API con estado**: La sesión mantiene un “contexto vivo” que registra las interacciones previas, de forma que el modelo puede recordar información escuchada o vista anteriormente.
* **Uso de WebSockets**:
  + **Comunicación bidireccional y full-duplex**: WebSocket ofrece un canal continuo entre el cliente y el servidor que permite el envío y recepción de datos de forma simultánea, esencial para conversaciones fluidas en aplicaciones que involucran voz y video.

Diagrama

El contenido generado por IA puede ser incorrecto.

* + **Protocolo estandarizado**: La API WebSocket está normalizada por el W3C, mientras que el protocolo fue definido por la IETF como RFC 6455. Esto brinda compatibilidad y estabilidad a largo plazo para proyectos que quieran integrar la API de Multimodal Live.
  + **Ventajas sobre conexiones HTTP tradicionales**: Dado que muchas redes bloquean conexiones TCP en puertos distintos al 80 o 443, la capacidad de WebSockets de operar sobre estos puertos y multiplexar múltiples servicios en un mismo canal TCP evita la necesidad de abrir puertos adicionales y simplifica la configuración de la infraestructura.

### Implementación en Python

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.Para la implementación en Python, es recomendable seguir las indicaciones proporcionadas en la documentación oficial de Google sobre la **API Multimodal Live**. A continuación, se muestra un ejemplo sencillo de cómo utilizar esta API para la generación de texto a texto en **Python 3.9** o versiones posteriores. [API Multimodal Live](https://ai.google.dev/gemini-api/docs/multimodal-live?hl=es-419)

Vamos a aplicar un ejemplo de cómo usar la API de Multimodal Live para la generación de texto a texto con Python 3.9 o versiones posteriores.

**Instala la biblioteca de la API de Gemini**

Para instalar el paquete google para usar sus modelos.

pip install google-genai

Una vez instalado importamos las librerías utilizadas.

Texto

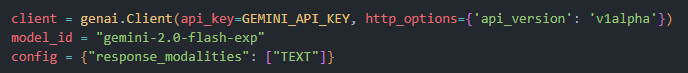
El contenido generado por IA puede ser incorrecto.

**Carga de Variables de Entorno**: Se extrae la clave de API (o cualquier otra configuración sensible) desde un archivo .env, evitando exponer datos críticos directamente en el código.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

**Creación del Cliente**: Se inicializa la clase genai.Client con la clave de API para autenticar las solicitudes al servicio de Gemini y se configura el modelo que se usará, en este caso "gemini-2.0-flash-exp".



**Sesión Asíncrona con WebSockets**:

Se utiliza async with para establecer y mantener la conexión en tiempo real con la API a través de WebSockets.



Durante la sesión, el usuario ingresa texto a través de la consola. Si escribe "exit", se cierra la sesión y el programa termina.

Texto

El contenido generado por IA puede ser incorrecto.

**Interacción con el Modelo**:

**Envío de Mensajes**: El texto introducido por el usuario se envía al modelo con session.send(), indicando con end\_of\_turn=True que se ha finalizado la interacción por ese turno y se espera la respuesta.



**Recepción de Respuestas**: Con session.receive(), se obtiene un flujo potencialmente continuo de datos (ideal si se requiere la respuesta en modo streaming). Si el modelo envía varios fragmentos, se irán imprimiendo en pantalla de forma progresiva.

Texto

El contenido generado por IA puede ser incorrecto.

Todo el código de ejemplo básico.



Este ejemplo, aun siendo básico, demuestra la sencillez y fluidez con las que se puede interactuar con Gemini gracias a las conexiones WebSocket. La baja latencia y la comunicación bidireccional en tiempo real brindan una experiencia óptima de conversación, lo que hace que la integración del modelo en diversas plataformas (web, móviles o incluso gafas inteligentes como las Meta Ray-Ban) sea altamente factible y eficiente.

Texto

El contenido generado por IA puede ser incorrecto.

### Tiene memoria

Un aspecto fundamental de los modelos de lenguaje de última generación es su capacidad para conservar y utilizar el contexto de la conversación a lo largo de múltiples intercambios. A partir del ejemplo mencionado, se observa que el modelo puede recordar y hacer referencia a la información proporcionada en preguntas y respuestas anteriores, lo que se traduce en una interacción más coherente y fluida, y mejora exponencialmente la relación con el usuario.



Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## Implementación de Text to Voice

En esta sección damos un paso más allá al explorar el enfoque para habilitar interacciones en las que el usuario introduce texto y recibe una respuesta hablada generada por el modelo Gemini. Esta arquitectura, conocida como Text to Speech (TTS), permite una comunicación más natural y accesible entre humanos y sistemas de inteligencia artificial. El objetivo de esta implementación es demostrar cómo se puede transformar la entrada textual en audio en tiempo real, utilizando la API experimental de Gemini Multimodal Live, lo cual abre la puerta a aplicaciones innovadoras en asistencia virtual, accesibilidad y interfaces conversacionales.

### Arquitectura del sistema

La solución se basa en el uso del modelo **gemini-2.0-flash-exp** de la API Gemini, que permite la generación de respuestas en formato de audio. La arquitectura del sistema se compone de los siguientes elementos clave:

* **Cliente de la API:** Configurado mediante una clave de API y parámetros específicos (como la versión y el modelo), se establece la conexión asíncrona con la API de Gemini.
* **Configuración del Prompt y Parámetros de Voz:** Se define un mensaje de instrucciones del sistema para establecer el rol del modelo (por ejemplo, "Eres un asistente experto en tecnología que responde en Español") y se especifica la voz deseada (ej. "Charon") a través del parámetro generation\_config.
* **Gestión de Archivos de Audio:** Se utiliza un *context manager* para crear y configurar archivos WAV en los que se almacenan los datos de audio recibidos.
* **Interacción Asíncrona:** Se implementa un bucle asíncrono que envía la entrada del usuario y recibe en fragmentos la respuesta en audio, permitiendo su reproducción al finalizar cada interacción.

### Configuración y Conexión a la API

Se define un objeto de configuración que incluye:

* **Instrucción del Sistema:** Establece el rol del modelo, orientándolo para responder como un asistente experto en tecnología en Español.
* **Parámetros de Generación:** Especifica que la respuesta debe incluir salida en audio y se selecciona la voz (por ejemplo, "Charon").

La configuración se estructura de la siguiente manera

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

### Gestión y Almacenamiento del Audio

Aunque el objetivo de la investigación es que el modelo hable directamente en esta fase de las pruebas va a ser guardado en disco las respuestas del modelo, posteriormente cambaremos este funcionamiento.

Texto

El contenido generado por IA puede ser incorrecto.

### Manejo de sesión y comunicación asíncrona

**Envío de la entrada:** Se solicita al usuario que ingrese una pregunta o comando, que se envía a la API utilizando el método session.send con parámetros por palabra clave.



**Recepción de la respuesta:** La respuesta en audio se recibe en fragmentos a través de un iterable asíncrono. La función async\_enumerate facilita la iteración y se escriben los datos en el archivo WAV configurado.

Texto

El contenido generado por IA puede ser incorrecto.

### Conclusión

La implementación de Text to Voice mediante la API de Gemini demuestra cómo es posible integrar respuestas en audio en aplicaciones interactivas. Los puntos destacados son:

* **Interacción en tiempo Real:** La arquitectura asíncrona permite enviar y recibir datos de forma fluida, logrando una comunicación inmediata.
* **Flexibilidad en la configuración:** La posibilidad de definir el rol del modelo y seleccionar diferentes voces (mediante speech\_config) ofrece una personalización que puede adaptarse a diversos contextos y aplicaciones.
* **Aplicaciones prácticas:** Este enfoque puede extenderse a asistentes virtuales, sistemas de accesibilidad y otras soluciones donde la interacción hablada mejora la experiencia del usuario.
* **Diferentes Voces:** El uso de distintas voces, aunque todas con características similares, añade un plus a la interactividad, permitiendo ajustar la experiencia de usuario según preferencias y necesidades específicas.



## Implementación de voice to voice

Después de comprobar el funcionamiento del sistema Text to Voice y confirmar que la conversión de texto en audio (TTS) opera correctamente, se ha decidido profundizar en la siguiente etapa: la implementación de un sistema Voice to Voice. Esta solución integra dos tecnologías esenciales: la conversión de voz a texto (STT) para capturar la entrada del usuario y la conversión de texto a voz (TTS) para generar la respuesta hablada. El objetivo es lograr una comunicación completamente natural, donde tanto la entrada como la salida se realicen mediante voz, eliminando la necesidad de interacción escrita y operando en tiempo real. Además, se ha incluido como requisito fundamental que el sistema permita interrumpir al modelo mientras éste está emitiendo la respuesta, facilitando una interacción más dinámica y adaptativa.

### Arquitectura del sistema

### Configuración y Conexión a la API

### Manejo de sesión y comunicación asíncrona

### Conclusión

**2.5. Limitaciones y Consideraciones**

* Costos asociados al uso de Gemini en la nube.
* Restricciones técnicas o éticas.

**Sección 3: Análisis de Modelos Multimodales de OpenAI**

**3.1. Introducción a los Modelos Multimodales de OpenAI**

* Resumen de los modelos más relevantes (por ejemplo, DALL·E, Whisper, GPT-4 con soporte multimodal).
* Comparación con otros modelos unimodales.

**3.2. Características Multimodales de OpenAI**

* Generación de imágenes mediante texto (DALL·E).
* Transcripción y síntesis de voz (Whisper).
* Interacción simultánea con texto e imágenes (GPT-4).

**3.3. Uso Práctico de OpenAI**

* API de OpenAI: cómo acceder a sus modelos multimodales.
* Casos de uso prácticos:
  + Diseño gráfico automatizado.
  + Traducción y transcripción de contenido multimedia.
  + Desarrollo de aplicaciones interactivas.

**3.4. Limitaciones y Consideraciones**

* Costos y disponibilidad de recursos en la nube.
* Restricciones éticas y legales.

**Sección 4: Comparación entre Gemini y OpenAI**

**4.1. Similitudes y Diferencias**

* Comparativa de características técnicas (precisión, velocidad, escalabilidad).
* Enfoques distintivos en la gestión de modalidades.

**4.2. Evaluación de Rendimiento**

* Métricas clave para evaluar modelos multimodales.
* Resultados experimentales (si aplican).

**4.3. Selección del Modelo Adecuado**

* Guía para elegir entre Gemini y OpenAI según el caso de uso.
* Factores a considerar: presupuesto, requisitos técnicos, compatibilidad.

**Sección 5: Implementación Práctica**

**5.1. Ejemplos de Código**

* Demostraciones de cómo interactuar con las APIs de Gemini y OpenAI.
* Fragmentos de código en Python u otros lenguajes.

**5.2. Herramientas y Frameworks Complementarios**

* Bibliotecas útiles para trabajar con datos multimodales (por ejemplo, TensorFlow, PyTorch).
* Integración con plataformas de análisis de datos.

**5.3. Consideraciones Técnicas**

* Optimización del rendimiento en entornos en la nube.
* Seguridad y privacidad de los datos.

**Sección 6: Ética y Responsabilidad en Modelos Multimodales**

**6.1. Impacto Social**

* Implicaciones éticas de la generación automática de contenido multimedia.
* Sesgos en los datos de entrenamiento.

**6.2. Directrices para el Uso Responsable**

* Recomendaciones para minimizar riesgos.
* Marco normativo actual relacionado con IA multimodal.

**Conclusión**

* Resumen de los hallazgos clave sobre Gemini y OpenAI.
* Reflexión sobre el futuro de la multimodalidad en LLM.
* Posibles líneas de investigación futuras.

**Anexos**

* Referencias bibliográficas.
* Documentación técnica adicional.
* Ejemplos detallados de implementaciones.