



Universidad
Rey Juan Carlos

INGENIERÍA EN SISTEMAS AUDIOVISUALES Y
MULTIMEDIA

Curso Académico 2022/2023

Trabajo Fin de Grado/Máster

INTEGRACIÓN DE TECNOLOGÍAS DE
INTELIGENCIA ARTIFICIAL EN UN ASISTENTE
VIRTUAL

Autor : Álvaro Delgado Monteagudo

Tutor : Dr. Jesús María González Barahona

Trabajo Fin de Grado/Máster

Integración de Tecnologías de Inteligencia Artificial en un Asistente Virtual

Autor : Álvaro Delgado Monteagudo

Tutor : Dr. Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2023, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2023

“Lo que todos tenemos que hacer es asegurarnos de que estamos usando la IA de una manera que sea en beneficio de la humanidad, no en detrimento de la humanidad”.

Tim Cook

©2023 Autor Álvaro Delgado Monteagudo

Algunos derechos reservados

Este documento se distribuye bajo la licencia

“Atribución-CompartirIgual 4.0 Internacional” de Creative Commons, disponible en

<https://creativecommons.org/licenses/by/4.0/deed.es>

Agradecimientos

Han sido 7 largos años para llegar a este momento. 7 años de subidas y bajadas en los cuales han habido momentos buenos y momentos malos en los que pesaban las asignaturas y me planteé abandonar en varias ocasiones, pero aquí estamos. Esto no hubiera sido posible sin las personas que me han apoyado siempre y han hecho de cada día un día mejor. Entre esas personas está mi familia que no solo ha dado el apoyo económico para llevar a término esta titulación sino que también me han apoyado a lo largo de la carrera. Quiero agradecerle especialmente a Susana por todo el apoyo que me ha dado a lo largo de estos años, de principio a final. Por todas las veces que me ha insistido para que me pusiera a estudiar un poco más, por todos sus intentos en que fuera más organizado y por todos esos momentos que incluso por videollamada me ha hecho compañía cuando estudiábamos a distancia. Muchas gracias por ayudarme a hacer este camino más fácil. Por último y no menos importante, darle las gracias a mi tutor Jesús María González Barahona por el tiempo que ha dedicado al seguimiento de este proyecto, ya que sin él no podría haberse llevado a término. Gracias a todos.

Resumen

Este Trabajo de Fin de Grado se ha centrado en el desarrollo de un asistente virtual conversacional avanzado, que funciona a través de comandos de voz y responde de la misma manera. Implementando tecnologías punteras en inteligencia artificial integradas en un único programa de Python a través de varias APIs y bibliotecas. Se ha utilizado el reciente modelo de reconocimiento de voz avanzado Whisper desarrollado por OpenAI el cual se desempeña a la perfección a la hora de transcribir audio. Esto combinado a ElevenLabs para la generación del lenguaje natural y ChatGPT para la generación de respuestas complejas y contextualmente adecuadas ha permitido la creación de un asistente altamente interactivo y eficiente.

Este asistente virtual, creado como resultado de los recientes avances en tecnologías basadas en inteligencia artificial, busca ser una herramienta accesible, eficaz y adaptable, tiene la habilidad de interactuar de manera natural y personalizada con los usuarios, facilitando así la realización de tareas complejas y proporcionando información relevante en una multitud de temas de una forma personalizada ya que el asistente puede ser configurado de forma sencilla para cambiar la personalidad de este, aumentando así su versatilidad.

En constante evolución, su interacción con los usuarios se vuelve cada vez más fluida y natural, con una creciente capacidad para abordar una amplia gama de necesidades. El desarrollo de este asistente virtual contribuye al avance de las aplicaciones prácticas de los asistentes conversacionales.

Otra característica del asistente virtual es la capacidad de descargar archivos de audio de Youtube a través de la biblioteca Pytube y su posterior transcripción a través de Whisper, lo que abre una nueva gama de posibilidades de interacción y acceso a contenidos. Este enfoque multiplataforma hace que el asistente sea aún más útil, ya que puede recuperar y transcribir contenido de audio de una fuente tan popular y diversa como Youtube.

Summary

This Final Degree Project has focused on the development of an advanced conversational virtual assistant, which operates through voice commands and responds in the same way. Implementing cutting-edge technologies in artificial intelligence integrated into a single Python program through some APIs and libraries. The recent advanced voice recognition model Whisper, developed by OpenAI, has been used, which performs perfectly on transcribing audio. This, combined with ElevenLabs for natural language generation and ChatGPT for the generation of complex and contextually appropriate responses, has allowed the creation of a highly interactive and efficient assistant.

This virtual assistant, created as a result of recent advancements in artificial intelligence-based technologies, aims to be an accessible, effective, and adaptable tool. It has the ability to interact naturally and personally with users, thus facilitating the performance of complex tasks and providing relevant information on a multitude of topics in a personalized way, as the assistant can be easily configured to change its personality, thus increasing its versatility.

In constant evolution, its interaction with users becomes increasingly fluid and natural, with a growing ability to address a wide range of needs. The development of this virtual assistant contributes to the advancement of the practical applications of conversational assistants.

Another feature of the virtual assistant is the ability to download audio files from YouTube through the Pytube library and their subsequent transcription through Whisper, which opens up a new range of possibilities for interaction and access to content. This cross-platform approach makes the assistant even more useful, as it can retrieve and transcribe audio content from a source as popular and diverse as YouTube.

Índice general

1. Introducción	1
1.1. Objetivo general	3
1.2. Objetivos específicos	3
1.3. Planificación temporal	4
1.4. Estructura de la memoria	6
2. Tecnologías empleadas	9
2.1. Python	9
2.2. Whisper	10
2.3. Eleven Labs	16
2.4. Speech Recognition	17
2.5. Pytube	18
2.6. ChatGPT	20
3. Diseño e implementación	25
3.1. Arquitectura general	25
3.2. Consulta	28
3.3. Síntesis de voz	30
3.4. Descarga	32
4. Experimentos y validación	35
4.1. Reconocimiento de voz	35
4.1.1. Condiciones ideales	36
4.1.2. Condiciones con ruido de fondo	38
4.1.3. Condiciones habla acelerada	39

4.2. Detección de comandos	41
4.3. Realizar consultas	42
4.4. Descarga audio de videos y transcripción.	46
5. Conclusiones	49
5.1. Consecución de objetivos	49
5.2. Aplicación de lo aprendido	51
5.3. Lecciones aprendidas	52
5.4. Trabajos futuros	54
A. Manual de usuario	55
A.0.1. Instalación	55
A.0.2. Guía usuario	56
Bibliografía	57

Índice de figuras

1.1. Esquema planificación temporal	5
2.1. Multitask training format	12
2.2. Modelos Whisper	14
2.3. Relación WER por idioma	15
3.1. Esquema general asistente virtual	26
3.2. Esquema primera fase	28
3.3. Esquema consulta	31
3.4. Esquema descarga	34
4.1. Función reconocimiento de voz	36
4.2. Resultado condiciones ideales	37
4.3. Resultado ruido de fondo	39
4.4. Resultado habla acelerada	40
4.5. Comando descarga	41
4.6. Comando consulta	41
4.7. Prueba oración 1	42
4.8. Prueba oración 2	42
4.9. Prueba consulta	43
4.10. Personalidad asistente	44
4.11. Prueba descarga	46
4.12. Archivo audio descargado	46

Capítulo 1

Introducción

Actualmente nos encontramos en una encrucijada de posibilidades digitales sin precedentes. La razón de esto ha sido una suma de avances: inteligencia artificial, aprendizaje automático y telecomunicaciones, que convergen en un mismo punto. Podemos verlo en todas partes, desde la elección de la próxima serie que Netflix nos sugiere, hasta la tecnología subyacente en los vehículos autónomos, los algoritmos se están convirtiendo en nuestros nuevos asesores de confianza, personalizando nuestra existencia, automatizando rutinas.

En este panorama de innovaciones, hay varios protagonistas que destacan: Python y ChatGPT, gracias a su versatilidad. Lo mismo en una aula universitaria que en una start-up de Silicon Valley, Python ha demostrado ser una herramienta indispensable. Su habilidad para manejar cantidades ingentes de datos, su modularidad que le permite adaptarse y crecer, lo convierten en un aliado crucial para implementar sistemas de IA y aprendizaje automático.

ChatGPT ha irrumpido en nuestras vidas y parece que es para quedarse. No solo nos ha dejado a todos con la boca abierta por su enorme potencial y capacidad de entendimiento, sino que ha revuelto el panorama tecnológico.

Enfocándonos en un terreno específico de este vasto panorama: los asistentes virtuales. Cada vez más, estos acompañantes digitales forman parte de nuestro día a día, interactuando con nosotros de forma tan natural como lo haría un humano. ¿Cómo es esto posible? Gracias a la inteligencia artificial.

Python, aquí, se convierte en una herramienta clave. Dotado de un sinfín de librerías y módulos, este lenguaje facilita la creación de asistentes virtuales sofisticados, capaces de entender nuestro lenguaje, reconocer nuestra voz e interpretar nuestros mensajes. Pero, no se queda ahí. Python también permite que cada asistente se personalice a la perfección, ajustándose como un guante a las necesidades individuales de cada usuario. Su modularidad y escalabilidad permiten implementar nuevas funcionalidades de manera fluida y eficiente, y es precisamente aquí donde se sitúa este Trabajo Final de Grado.

El objetivo: fusionar el poder de la inteligencia artificial y Python para crear un asistente virtual. ¿Las herramientas? Diversas librerías, como Whisper, la tecnología ChatGPT y Eleven-Labs entre otras. En el corazón de este proyecto está OpenAI, cuyas innovaciones son la piedra angular sobre la que se construye nuestro asistente virtual.

1.1. Objetivo general

Este trabajo de fin de grado tiene como objetivo la creación de un asistente conversacional operado por voz. El desarrollo de este asistente se ha llevado a cabo a través de las últimas tecnologías que han irrumpido en los últimos meses revolucionando el campo de la inteligencia artificial. Un asistente conversacional controlado por voz, con el que podamos realizar consultas y algunas funciones básicas como la transcripción a texto de videos. Además se propone darle a nuestro asistente un contexto situacional con el que se adapte de la mejor manera posible a como queremos que sea el tipo de respuesta. En suma, este trabajo no se limita a ser un estudio teórico. Representa un paso donde la inteligencia artificial y los humanos colaboran hacia un mundo en el que la tecnología se adapta a nosotros, y no al revés.

1.2. Objetivos específicos

Durante la elaboración de este proyecto se han afrontado las siguientes metas:

- Trabajar y aprender con tecnología de reconocimiento de voz automático.
- Aprender sobre software de voz realista en concreto Eleven Labs.
- Usar paquetes y librerías en Python como Pytube y ElevenLabs.
- Crear un asistente conversacional funcional con Python.
- Aprender y aplicar sobre el uso de los prompts¹ en ChatGPT.
- Implementación y configuración de ChatGPT en Python.
- Desarrollo y establecimiento de Whisper.
- Elaboración de conversaciones más funcionales.

¹<https://aulacm.com/que-es/prompt-significado-definicion/>

1.3. Planificación temporal

El proyecto comenzó como una idea a principios de febrero de 2023 en la que gracias a mi tutor Jesús María González Barahona conocí la tecnología Whisper. Una tecnología de reconocimiento de voz automático, esto sumado al boom de ChatGPT, fue casi instantáneo relacionar estas dos tecnologías tan apasionantes en la creación de un asistente virtual al más puro estilo “Siri”, “Google Home” o “Alexa”.

Se ha llevado una metodología de trabajo basada en una metodología ágil de trabajo en la que se tiene un enfoque de gestión del proyecto que se basa en la adaptabilidad, flexibilidad y la interacción constante entre cliente o usuario. Ya que este proyecto no tiene una naturaleza laboral se ha adaptado a un contexto de estudiante. Se ha centrado en la entrega rápida y continua con énfasis en la mejora constante y la capacidad de adaptarse a los cambios. Se ha trabajado en sprints[1] de dos semanas aproximadamente. En los que al inicio de cada sprint² se planifica el trabajo que se llevará durante ese periodo. Al final de cada sprint se revisaba el trabajo realizado, lo aprendido, mejoras y dificultades que se han tenido en el proceso además de sobre como mejorar en el próximo sprint. Esta estructura cíclica permite una revisión y ajuste constante del trabajo, lo que facilita la adaptación a los cambios y la mejora continua del producto.

Esta metodología de trabajo ha sido clave en la gestión del proyecto ya que durante los meses en los que ha transcurrido el trabajo han ido surgiendo nuevas tecnologías casi semanalmente de nuevas empresas que se sumaban a la ola de la inteligencia artificial y el procesamiento de texto además de que a cada avance en la creación del asistente virtual han ido apareciendo nuevos retos que se han tenido que ir venciendo poco a poco.

²<https://www.atlassian.com/es/agile/scrum/sprint-planning>



Figura 1.1: Esquema planificación temporal

Durante todo el proyecto se ha ido trabajando poco a poco en cortas y numerosas sesiones principalmente desarrolladas por las tardes, ya que por las mañanas he estado ocupado en las prácticas de empresa.

- **Sprint 1.** En primer lugar, pasé las primeras semanas de febrero aprendiendo todo lo posible sobre las tecnologías del reconocimiento de voz automático, Whisper principalmente. Una vez recopilada la información necesaria para entender esta herramienta comenzaron las primeras pruebas de su uso en Python con su consecuente gestión de problemas de implementación, solución de estos problemas y finalmente qué funciones podría añadir a mi programa para que fuera más completo y eficiente.

Para mejorar los tiempos de respuesta y potencia del modelo se propuso la aceleración de hardware por GPU a la cual aun no he sido capaz de solventar con hardware de AMD ya que el uso de las bibliotecas mas utilizadas como Cuda son propias para el hardware de Nvidia. Por lo que hasta el momento se ha mantenido la aceleración por CPU intentando optimizar el modelo para hacer un uso más eficiente de los recursos.

- **Sprint 2.** Durante el mes de Marzo de 2023 siguiendo la metodología de trabajo se propuso el siguiente avance al proyecto, la implementación de una tecnología de procesamiento de lenguaje siguiendo el esquema cíclico anteriormente mencionado. Esta implementa-

ción no solo ha requerido la investigación sobre el uso de la tecnología de ChatGPT sino también sobre el uso de APIs³ en Python que era una funcionalidad desconocida para mí.

- **Sprint 3.** Abril de 2023 ha sido un mes cuyo objetivo ha sido principalmente mejorar el programa prototipo en Python con algunas funcionalidades extra. Como la incorporación de un contexto conversacional lo que aporta cierta personalidad al asistente, ya que podemos configurar fácilmente el tono y contexto de las respuestas que genera nuestro programa. Además se incorporó de forma provisional la biblioteca `pyttsx3` para la conversión de texto a voz de manera que podamos escuchar las respuestas de nuestro asistente en vez de recibir las respuestas de forma escrita.

- **Sprint 4.** Durante las primeras semanas del mes de mayo se implementaron dos funcionalidades nuevas al asistente virtual. La primera fue la funcionalidad de descargar el audio de videos de YouTube a través de la biblioteca `pytube`, una vez obtenido los audios de los videos, haciendo uso de la biblioteca `Whisper` se ha implementado la capacidad de generar una transcripción a texto del audio seleccionado. Por último se mejoró la capacidad de la conversión de texto a voz mediante la api de Openai Eleven Labs.

Este software de voz con IA permite generar voces mucho más realistas que las que se generan normalmente con bibliotecas como `pyttsx3`⁴. Además dispone la capacidad de entrenar el modelo suministrando archivos de voz, de esta manera la voz generada pretenderá en la medida de lo posible ser lo más parecida a la voz suministrada.

- **Sprint 5.** Creación de la memoria.

1.4. Estructura de la memoria

A continuación se detalla una forma estructurada ordenada de cómo está organizada la memoria del proyecto:

- **Capítulo 1: Introducción.** Se ofrece una descripción concisa del inicio del proyecto, sus fundamentos y su funcionamiento.

³<https://anderfernandez.com/blog/como-crear-api-en-python/>

⁴<https://pypi.org/project/pyttsx3/>

- **Capítulo 2: Objetivos.** Se presentan los propósitos del proyecto.
- **Capítulo 3: Estado del arte.** Se detalla cada tecnología implementada en este proyecto.
- **Capítulo 4: Diseño e implementación.** Se profundiza en la descripción detallada de cómo funciona la aplicación.
- **Capítulo 5: Experimentos y validación.** Se exponen los diferentes experimentos realizados y se verifica el funcionamiento adecuado de la aplicación además de distintos experimentos.
- **Capítulo 6: Conclusiones.** Se resaltan los logros obtenidos y se presenta la solución a los objetivos que no se alcanzaron.

Capítulo 2

Tecnologías empleadas

2.1. Python

Python[2] es un lenguaje de programación de alto nivel, que destaca entre los usuarios por su apuesta en la limpieza sintáctica y facilidad de lectura. Este lenguaje dió su aparición al mundo en 1989 por el informático nacido en Países Bajos Guido van Rossum. La razón principal de que este lenguaje esté en el punto de mira de muchos usuarios es por su énfasis en la claridad del código y la simplicidad, pero también, y no menos importante, por su potencia considerable.

La simplicidad de Python radica principalmente en su sintaxis y su capacidad de modularizar y reutilizar el código construyendo módulos y paquetes. Al contrario de otros lenguajes de programación, Python se basa solo en indentaciones para delimitar los bloques de código, cuyo resultado es una mayor legibilidad. Lo que supone un buen punto de partida para los novatos de la programación.

Python es multiparadigma es decir, acepta varios estilos de programación. Aunque fué diseñado principalmente como un lenguaje orientado a objetos también acepta la programación procedimental, funcional y orientada a aspectos. Esto se traduce en versatilidad y flexibilidad, convirtiéndose así en una opción polivalente para un sin fin de proyectos.

Python es un lenguaje interpretado, el código es ejecutado línea por línea. Una característica que no solo otorga a los desarrolladores y un feedback instantáneo sino que también permite una mayor flexibilidad para probar y experimentar con fragmentos de código.

Una fortaleza clave en Python radica en su comunidad de soporte, sólida y muy activa. Esta energética comunidad proporciona una amplia gama de recursos, muchas bibliotecas y resuelve problemas de manera eficiente. Desde foros en internet como Stack Overflow hasta convenciones de carácter internacional como PyCon. La comunidad de Python destaca por su actividad constante y su dedicación para mejorar y hacer de él el lenguaje de programación ideal. Si además añadimos la extensa cantidad de tutoriales, documentación y cursos en línea, el aprendizaje en este lenguaje se vuelve una tarea más factible para cualquier usuario que tenga interés en el campo de la programación. No hay que olvidar el incontable número de bibliotecas y frameworks. Algunos de ellos como Numpy para el cálculo matemático, Pandas para el análisis de datos y otras para el aprendizaje automático como TensorFlow y Pytorch. Cada una de estas bibliotecas añade valor adicional a las capacidades del lenguaje.

Python, en sí mismo, es un lenguaje de programación de gran versatilidad y de uso sencillo, aunque puede haber lenguajes que lo superen en cuanto a rendimiento o especialización en determinadas áreas, Python sobresale por su simplicidad, legibilidad y la potencia de su comunidad.

2.2. Whisper

El 21 de septiembre de 2022 se publicó una innovación significativa en el ámbito del reconocimiento de voz: Whisper[3]. Esta herramienta, está basada en las técnicas de aprendizaje profundo, o “deep learning”, ha sido desarrollada como OpenSource por OpenAI¹, un laboratorio líder en inteligencia artificial, y se presenta como una solución de código abierto para los usuarios.

Whisper exhibe una versatilidad notable. No se limita al reconocimiento de voz; en realidad, está diseñado para realizar una variedad de tareas. Posee la capacidad de entender múltiples

¹<https://openai.com/>

idiomas, traducir voz e incluso identificar distintos lenguajes. podríamos decir que, Whisper se destaca como una adición valiosa al campo del procesamiento del lenguaje natural, estableciendo nuevos estándares y como una nueva oportunidad en este campo.

El aprendizaje profundo o deep learning[4] es una especialidad del aprendizaje automático, el cual está fundamentado en algoritmos que se disponen en capas, a los que llamamos redes neuronales artificiales. Estas redes buscan reproducir la dinámica de nuestro cerebro, aunque aún están lejos de igualar todas sus capacidades. Un rasgo distintivo del aprendizaje profundo es su voraz apetito por grandes cantidades de datos, pero en contraste, no demanda mucha intervención humana para operar eficientemente. En su afán de optimizar el reconocimiento de voz, Whisper se alimenta de una gran cantidad de audio recogido de la web. Ha aprendido de nada menos que 680,000 horas de datos supervisados multilingües recolectados de internet. Esta extensa capacitación le permite discernir y transcribir habla en diferentes idiomas y dialectos sin necesidad de entrenamiento específico para cada uno de ellos.

Whisper emplea una técnica específica de aprendizaje profundo denominada Transformer². Esta técnica atribuye diferentes grados de relevancia a cada porción de la información que procesa. Diseñada para manejar datos en secuencia, como el habla, los modelos Whisper se valen de múltiples capas Transformer para procesar el audio y extraer características valiosas de la voz, tales como propiedades acústicas y lingüísticas.

Whisper está entrenado para desempeñar diversas tareas de procesamiento de voz. Incluyen reconocimiento de voz en varios idiomas, traducción de voz, identificación de lenguaje hablado y detección de actividad de voz. Estas tareas se condensan en una secuencia de tokens que el decodificador predice, lo que permite que un solo modelo ejecute diversas etapas de un proceso de voz tradicional.

²<https://la.blogs.nvidia.com/2022/04/19/que-es-un-modelo-transformer/>

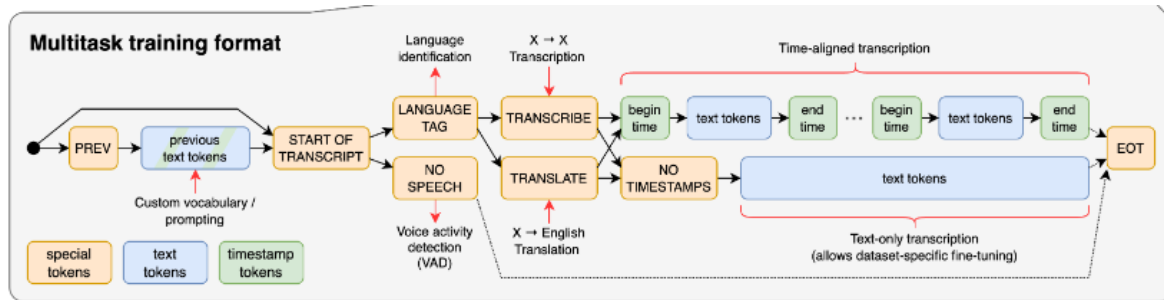


Figura 2.1: Multitask training format

En su búsqueda por incrementar su precisión y robustez, los modelos Whisper recurren a técnicas avanzadas. Hacen uso de técnicas como la eliminación aleatoria (dropout), destinada a reducir el sobreajuste. Durante el entrenamiento, ciertas neuronas de la red neuronal se descartan al azar. Esta estrategia previene que la red se ajuste excesivamente a los datos de entrenamiento y potencia su habilidad para aplicar lo aprendido a nuevos datos. Además, utilizan la normalización por lotes (batch normalization) para limitar el sobreajuste y mejorar la habilidad de generalización del modelo. Una de las características sobresalientes de los modelos Whisper es que se evalúan en un entorno sin entrenamiento previo. Esto significa que no requieren ajustes ni adaptaciones para desempeñarse eficazmente en diversas situaciones. En lugar de eso, los modelos Whisper pueden adaptarse sin problemas a diferentes tareas y lenguajes.

Para evaluar la efectividad de los modelos Whisper, se han realizado numerosos experimentos utilizando diversos conjuntos de datos multilingües y multitarea. Los resultados indican que los modelos Whisper pueden superar a otros sistemas tradicionales en términos de precisión y robustez. Whisper analiza audios en fragmentos de 30 segundos para entender el contexto del audio y proporcionar una transcripción más precisa. Whisper se entrena en ventanas de audio de 30 segundos y no puede procesar audios de mayor longitud.

Para transcribir audios de mayor duración, se emplea una estrategia de transcripción en búfer. Esta estrategia consiste en transcribir segmentos consecutivos de 30 segundos y desplazar la ventana de transcripción según los tiempos estimados por el modelo. De esta manera, el modelo puede obtener el contexto suficiente para entender lo que se está diciendo en cada segmento y producir una transcripción más precisa.

Whisper puede ser configurado atendiendo a los siguientes parámetros:

- **Lenguaje.** Se ha indicado que el lenguaje utilizado para el asistente será el español pero puede ser multilinguaje cualquier idioma que se hable a través del micrófono aunque se puede configurar como multilinguaje.
- **Floating Point**³. El modelo Whisper solamente permite el uso de FP32 (32bits) cuando está acelerado únicamente por CPU reservando el modelo FP16(16 bits) para el uso en GPU. Se ha establecido el parámetro FP32 en el programa.
- **Modelo (tiny, base, small, medium, large).** Estos modelos ordenados de menor a mayor han sido configurados y entrenados con distinto número de parámetros lo que se traduce en una mayor precisión a la hora de transcribir audio a texto a cuantos más parámetros, pero, también aumenta significativamente el coste computacional a medida que se pide un modelo más exigente lo que se traduce en mayores tiempos de espera.

Python tiene la función de realizar los cálculos del modelo a través de una GPU[5] en vez de la CPU. EL uso de una tarjeta gráfica moderna frente a una CPU presenta ventajas como:

- **Paralelismo masivo:** Las GPU están diseñadas para realizar operaciones en paralelo con un gran número de núcleos.
- **Mayor rendimiento:** Debido a su arquitectura paralela y a la gran cantidad de núcleos, las GPU pueden realizar cálculos numéricos de manera mucho más rápida que una CPU convencional.
- **Aceleración de bibliotecas y frameworks:** Muchas bibliotecas y frameworks de aprendizaje automático y procesamiento de datos, como TensorFlow y PyTorch, están optimizados para aprovechar las capacidades de las GPU.
- **Memoria dedicada:** Las GPU tienen su propia memoria dedicada llamada memoria de video, que es más rápida y de mayor capacidad que la memoria principal de una CPU.

³<https://docs.python.org/3/tutorial/floatingpoint.html>

Existen cinco modelos [6] de diferentes tamaños, cuatro de los cuales son en inglés. Estos modelos ofrecen diferentes equilibrios entre velocidad y precisión. A continuación, se presentan los nombres de los modelos disponibles y sus requerimientos aproximados de memoria y velocidad relativa.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	<code>tiny.en</code>	<code>tiny</code>	~1 GB	~32x
base	74 M	<code>base.en</code>	<code>base</code>	~1 GB	~16x
small	244 M	<code>small.en</code>	<code>small</code>	~2 GB	~6x
medium	769 M	<code>medium.en</code>	<code>medium</code>	~5 GB	~2x
large	1550 M	N/A	<code>large</code>	~10 GB	1x

Figura 2.2: Modelos Whisper

El rendimiento de Whisper puede variar significativamente según el idioma. La siguiente figura muestra el desglose de la tasa de error de palabras (WER) por idiomas del conjunto de datos (cuanto más pequeños sean los números, mejor será el rendimiento).

En lo que a nosotros nos concierne la transcripción también se puede realizar dentro de Python:

```
import whisper

model = whisper.load_model("base")
result = model.transcribe("audio.mp3")
print(result["text"])
```

A continuación se describe también un empleo de uso que proporciona acceso al nivel inferior del modelo:

```
import whisper

model = whisper.load_model("base")

# load audio and pad/trim it to fit 30 seconds
```

```

audio = whisper.load_audio("audio.mp3")
audio = whisper.pad_or_trim(audio)

# make log-Mel spectrogram and move to the same device as the model
mel = whisper.log_mel_spectrogram(audio).to(model.device)

# detect the spoken language
_, probs = model.detect_language(mel)
print(f"Detected language: {max(probs, key=probs.get)}")

# decode the audio
options = whisper.DecodingOptions()
result = whisper.decode(model, mel, options)

# print the recognized text
print(result.text)

```

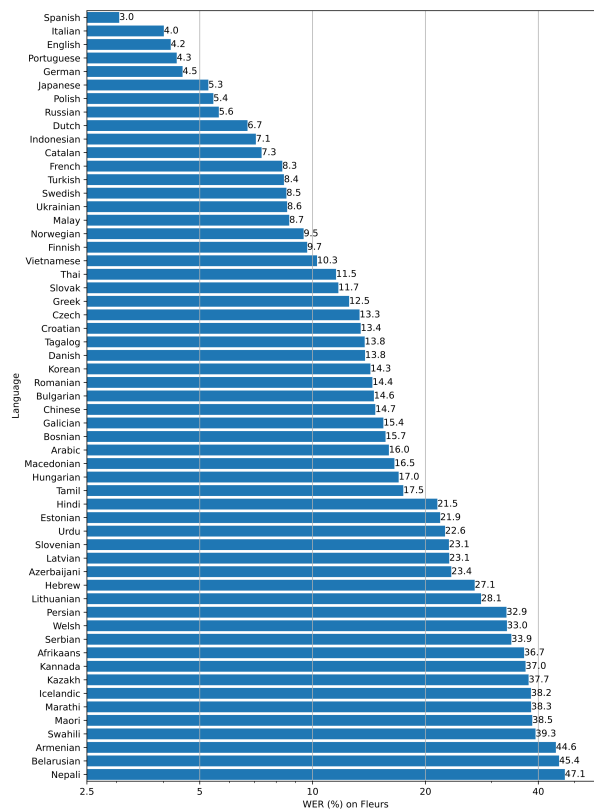


Figura 2.3: Relación WER por idioma

2.3. Eleven Labs

En el auge de las tecnologías basadas en la inteligencia artificial que nos encontramos, han irrumpido numerosas tecnologías en el ámbito de la síntesis de voz, que no han dejado indiferente a nadie. Muchos hemos escuchado canciones o audios generados por IA que han sabido representar fielmente las características del habla humana, haciendo en muchos casos casi indistinguible de si son reales o generados por máquina.

ElevenLabs[7] destaca en este panorama. Esta innovadora tecnología en síntesis de voz y conversión de texto a voz está revolucionando nuestra interacción con el audio digital. Utilizan algoritmos avanzados y procesamiento de señales en su plataforma de clonación de voz, capturando con precisión los patrones de habla y entonación de un individuo.

El proceso que utiliza ElevenLabs para sintetizar la voz es un estudio en sí mismo. Analizan una muestra de voz, identificando características únicas del hablante: tono, entonación, ritmo, pronunciación. El análisis meticuloso de estos aspectos es fundamental para la efectividad de la clonación de voz.

Posteriormente, la IA de ElevenLabs utiliza algoritmos de aprendizaje profundo para ‘aprender’ cómo reproducir estas características. Este aprendizaje se basa en redes neuronales con múltiples capas para modelar y aprender a partir de las características de voz identificadas. Una vez que la IA ha ‘aprendido’ a recrear la voz, puede generar nuevos segmentos de habla a partir de cualquier texto, mediante la tecnología de conversión de texto a voz (TTS⁴). Esto resulta en una voz sintética notablemente similar a la original.

ElevenLabs también destaca por su habilidad para asociar el audio generado con el usuario que lo creó, lo que puede tener un impacto significativo en la autenticidad y la propiedad en la producción de audio digital, y enriquecer la personalización y la interacción en las aplicaciones.

La plataforma[8] de ElevenLabs, accesible mediante suscripción, brinda una oportunidad in-

⁴<https://thehook.es/que-es-el-text-to-speech-y-como-funciona/>

mejorable para experimentar con esta tecnología punta. Su accesibilidad abre las puertas a la creación de nuevas aplicaciones y servicios de una forma muy sencilla.

ElevenLabs proporciona un caso de estudio único para explorar las aplicaciones y las implicaciones de la síntesis de voz basada en IA. A partir de aquí el uso de esta plataforma está ligada a una premisa, “la imaginación al poder”. Podemos recrear audiolibros, asistentes de voz personalizados, etc. Los usos son prácticamente ilimitados.

2.4. Speech Recognition

Speech recognition[9] es una biblioteca de Python extremadamente útil en el desarrollo de aplicaciones del procesamiento del lenguaje. Fué desarrollada por el ingeniero de software Anthony Zhang, y proporciona una vía directa para interactuar con API-s de reconocimiento de voz como Google Web Speech API, Microsoft Bing Voice Recognition y IBM Speech to Text.

Es una biblioteca de licencia BSD⁵ lo que permite un uso comercial como a nivel de usuario. Su premisa es ser versátil y poderosa. Versátil porque permite trabajar al desarrollador con diversos motores de reconocimiento de voz y transcripción. y poderosa porque apesar de ser sencilla de usar puede manejar tareas complejas. De esta manera posibilita al desarrollador elegir el motor que mejor se ajuste a las necesidades específicas de cada proyecto, en cuanto a la terminos de precisión, velocidad, idioma y costes.

Además, su diseño orientado al desarrollador la hace accesible incluso para aquellos que no están familiarizados con el reconocimiento de voz. El código fuente está bien comentado y la documentación es completa y detallada, facilitando así su comprensión y manejo.

Un elemento clave de esta biblioteca es la clase Recognizer. Este componente contiene una serie de métodos, como:

`recognize_google()`, `recognize_bing()`, y `recognize_ibm()`.

Cada uno de estos métodos interactúa con una API de reconocimiento de voz específica, ofre-

⁵<https://appmaster.io/es/blog/que-es-la-licencia-bsd>

ciendo la capacidad de transcribir audio a texto. Para la obtención de audio, la biblioteca SpeechRecognition ofrece la clase Microphone, diseñada para grabar audio desde el micrófono del dispositivo. Para trabajar con archivos de audio pregrabados, se puede emplear la clase AudioFile, lo que resulta útil en aplicaciones de transcripción automática de audios previamente almacenados. El diseño modular de la biblioteca facilita su funcionamiento. En primer lugar, se genera una instancia de la clase⁶ Recognizer. Posteriormente, se utilizan las clases Microphone o AudioFile para capturar el audio, que luego es procesado por uno de los métodos de la clase Recognizer para llevar a cabo la transcripción. Otra función a destacar es `adjust_for_ambient_noise()`, la cual mejora la precisión del reconocimiento en entornos ruidosos. Esta función mide el nivel de ruido del audio y ajusta el umbral de energía adecuadamente, proporcionando un rendimiento óptimo incluso en condiciones de ruido ambiente. En cuanto al uso de alguna de estas APIs cabe destacar que utilizar la que mejor se adapte a cada situación es vital, ya que algunas imponen restricciones en términos de duración del audio que pueden procesar, la cantidad de solicitudes con las que trabajan en un periodo de tiempo determinado y algunas más. SpeechRecognition permanece como una de las más completas y efectivas para trabajar con reconocimiento de voz en Python. Su versatilidad, unida a su sencillez de uso, la convierte en una herramienta muy eficiente y potente.

2.5. Pytube

Pytube[10] es una biblioteca de Python que funciona a través de la URL de YouTube para descargar el audio o video a una ubicación previamente especificada por el usuario. Una virtud de esta biblioteca es que no requiere instalar ningún otro software.

Algunos detalles técnicos de esta biblioteca son:

Interfaz sencilla. Pytube tiene una interfaz muy sencilla e intuitiva. Ya que todo lo que necesitas para su funcionamiento es prácticamente una URL a un video de YouTube.

Descarga videos y audio. Puedes descargar videos de YouTube pero además tiene la función para extraer únicamente el audio de esos videos si así se requiere.

Soporte para listas de reproducción. La descarga de contenido no se limita a un único elemen-

⁶<https://docs.python.org/es/3/tutorial/classes.html>

to sino que puede descargar también todos los archivos asociados a una lista de reproducción de la plataforma.

Sin dependencias externas. Pytube no depende de ningún otro software lo que lo vuelve fácil de instalar. Esta biblioteca maneja adecuadamente las excepciones, así como los errores de red, también la capacidad de descargas en segundo plano pudiendo descargar videos mientras se realizan otras tareas.

Otra característica valiosa es que puede obtener información relevante de los videos como el título, numero de visitas, duración etc.

El uso de esta aplicación es muy sencillo, puedes instalarla usando pip⁷:

```
pip install pytube
```

Una vez instalada, puedes usarla para descargar videos de YouTube:

```
from pytube import youtube

yt = Youtube('URL')

#Descargar video con la resolución mas alta disponible.
yt.streams.get_highest_resolution().download()

#Si se desea descargar solo audio
yt.streams.get_audio_only().download()
```

⁷<https://pypi.org/project/pip/>

2.6. ChatGPT

ChatGPT[11] es una variante del modelo de lenguaje GPT (Generative Pretrained Transformer). Este modelo de inteligencia artificial el cual está especializado en la generación de texto en respuesta a una entrada indicada. A través de esta entrada es capaz de completar texto, generar respuestas a preguntas, ejercer como traductor y realizar numerosas tareas relacionadas con el lenguaje.

ChatGPT ha experimentado una constante evolución y aún sigue evolucionando la versión más actual de este chat en estos momentos es GPT-4 y para finales de este año tendremos GPT-5

Cómo funciona.

Todas las versiones de GTP se basan en redes neuronales en concreto en el modelo Transformer. Un modelo transformer es una red neuronal que aprende contexto y, por lo tanto, significado mediante el seguimiento de relaciones en datos secuenciales como las palabras de esta oración.

Los modelos transformer aplican un conjunto en evolución de técnicas matemáticas, llamadas atención o atención propia, para detectar formas sutiles en que los elementos de datos en una serie se influyen y dependen entre sí.

El modelo se entrena a través de la información de internet como fuente de datos, pero no sabe específicamente de dónde vienen estos datos. Durante el entrenamiento el modelo aprende a predecir la siguiente palabra en una secuencia de texto dada. Mediante esta tarea el modelo aprende una representación del lenguaje, estas formas no se limitan meramente a lo gramatical y sintáctico sino que también aprende sobre hechos del mundo y algunas formas en las que las personas usan el lenguaje para transmitir tono y emoción.

Detalles técnicos.

GPT genera texto a través de una entrada (por ejemplo una pregunta) el modelo genera una respuesta generando palabra por palabra hasta que completa la respuesta. El modelo transformer ha sido entrenado con un gran número de parámetros . En concreto GPT-3 fué entrenado con 175 mil millones de parámetros[12] y GPT-4 se estima que ha sido entrando con 600 veces más parámetros que su modelo predecesor. Estos parámetros permiten al modelo obtener información muy valiosa sobre el lenguaje.

Un aspecto fundamental de este modelo es el uso de técnicas como “long-range-attention”. Esta capacidad permite al modelo obtener información de partes muy separadas del texto al generar una respuesta. Lo que permite mejorar la coherencia de las respuestas obteniendo un resultado más preciso y detallado.

Arquitectura del Transformer.

Primero, es útil entender la arquitectura del Transformer[13], que es la base de GPT. Los Transformers utilizan una estructura de auto-atención, es decir, permiten que cada palabra en una entrada tenga en cuenta todas las otras palabras para generar una mejor representación. En lugar de procesar las palabras en orden, como en las arquitecturas RNN⁸ y LSTM⁹, los Transformers pueden manejar todas las palabras en paralelo, lo que es más eficiente computacionalmente.

La arquitectura del Transformer se compone de dos partes principales: el codificador y el decodificador.

⁸<https://www.ibm.com/es-es/topics/recurrent-neural-networks>

⁹<https://es.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html>

Entrenamiento.

Para el entrenamiento, los modelos de GPT se entrenan utilizando una variante de la técnica de aprendizaje automático llamada aprendizaje supervisado. El modelo se entrena para predecir la siguiente palabra en una oración, dadas todas las palabras anteriores. La función de pérdida utilizada durante el entrenamiento es la entropía cruzada, función de pérdida, o Loss function, es una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones realizadas durante el aprendizaje. Cuanto menor es el resultado de esta función, más eficiente es la red neuronal. Su minimización, es decir, reducir al mínimo la desviación entre el valor predicho y el valor real para una observación dada, se hace ajustando los distintos pesos de la red neuronal. Mide la diferencia entre la distribución de probabilidad predicha por el modelo para la siguiente palabra y la verdadera distribución (donde la palabra correcta tiene una probabilidad de 1 y todas las demás palabras tienen una probabilidad de 0).

Como he mencionado anteriormente los datos del entrenamiento son una colección de texto de internet, pero GPT no sabe específicamente de donde provienen estos datos.

Generación de texto.

Una vez entrenado, GPT genera el texto palabra por palabra. Dada una entrada (como una pregunta), el modelo genera una respuesta seleccionando palabras una por una hasta que completa la respuesta. Esta selección de palabras se realiza a través de un proceso de muestreo, donde a cada palabra posible se le asigna una probabilidad y se selecciona una palabra de acuerdo con esta distribución.

El modelo tiene cierta aleatoriedad en su generación de texto, que se controla mediante un parámetro de “temperatura”[14]. Una temperatura más alta conducirá a respuestas más aleatorias, mientras que una temperatura más baja hará que las respuestas sean más deterministas y centradas en las palabras más probables.

Tamaño y escalabilidad.

GPT-4 es un modelo de gran escala, con miles de millones de parámetros. Esto es mucho mayor que muchos otros modelos de aprendizaje automático y permite que el modelo capture una cantidad increíblemente grande de información sobre el lenguaje. Estos modelos de gran escala también presentan desafíos, como la necesidad de grandes cantidades de memoria y capacidad de computación para el entrenamiento y la generación de texto.

La arquitectura de GPT es escalable, lo que significa que a medida que aumentas el tamaño del modelo (en términos de cantidad de parámetros), generalmente mejora su rendimiento. Este es un área de investigación activa en el aprendizaje automático y la NLP: cómo construir y entrenar de manera eficiente estos modelos de gran escala.

Capítulo 3

Diseño e implementación

En este capítulo se expondrá como se ha desarrollado el asistente virtual y sus características, además de su código y algunos ejemplos.

3.1. Arquitectura general

El asistente virtual ha sido desarrollado como un programa de Python en la versión 3.10.5 ya que el modelo de transcripción Whisper acepta versiones entre v3.8 - v3.11. En la figura [Esquema general asistente virtual] podemos observar a vista de pájaro cómo funciona de forma esquematizada el asistente virtual y los pasos que este sigue para su ejecución. En la primera fase el programa capta una entrada de sonido a través del micrófono. A partir de esta entrada se genera un archivo de audio que inmediatamente después será transcrito a texto, esta transcripción es el proceso más delicado e importante para el correcto funcionamiento del asistente, es crucial que el texto haya sido transcrito de forma adecuada porque de esto dependen las posteriores funciones del programa. El texto generado es analizado para la identificación de diferentes comandos. El asistente ha sido configurado para identificar y llevar a término dos funciones, generar consultas conversacionales y descargar audio de videos de YouTube además de su posterior transcripción. Aunque el programa ha sido configurado de esta manera sus posibilidades son prácticamente ilimitadas y escalables, por lo que podemos añadir nuevos módulos con distintas funcionalidades de una forma rápida y sencilla.

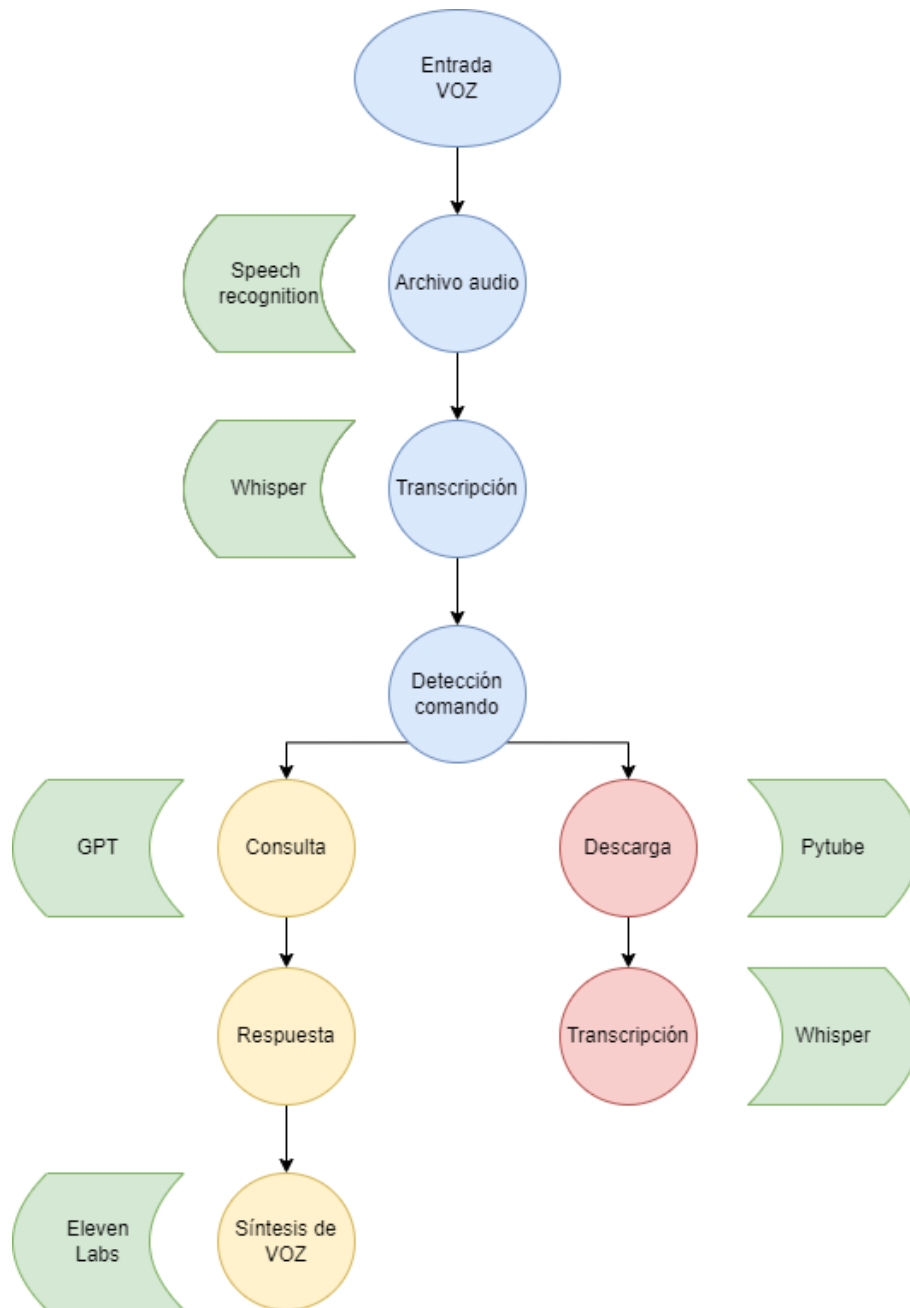


Figura 3.1: Esquema general asistente virtual

Al inicio de la ejecución de la función `voicerec()` la cual es la encargada de grabar el audio del micrófono y posteriormente transcribirlo, el programa establece una conexión con el micrófono de la máquina a través del cual recogerá las señales de voz que se convertirán en datos de audio. Durante este proceso, se imprime el mensaje “Escuchando...” en la consola además de un pitido sonoro que indicará al usuario que el sistema está listo para recibir su entrada de voz.

Antes de que comience la grabación del audio, hacemos un pequeño ajuste para tener en cuenta el ruido de fondo en el lugar donde se va a grabar, haciendo uso de la función `adjust_for_ambient_noise()`. De esta manera, el sistema es capaz de diferenciar la voz del usuario del resto de los ruidos, lo que le permite escuchar más claramente.

Una vez configurados los ajustes de la grabación, el sistema comienza a grabar el audio a través del micrófono. Esta grabación se almacena en memoria hasta que el usuario termina de hablar. Posteriormente, los datos de audio recogidos se escriben en un archivo en formato WAV denominado “Audio.wav”. Este archivo almacenará la grabación para su uso posterior.

En el siguiente paso, el audio grabado se procesa utilizando el modelo de transcripción Whisper. La función encargada de este proceso es muy sencilla de configurar y atiende a los siguientes parámetros:

- **Lenguaje:** Español.
- **Floating Point:** FP32.
- **Modelo:** Small.

El modelo utilizado en este asistente virtual ha sido “small”, ya que este tamaño mantiene un compromiso entre eficacia y tiempo de ejecución muy estable. Con tiempos de ejecución de aproximadamente 3 segundos para la mayoría de consultas. El modelo ha sido acelerado exclusivamente por CPU ya que no se han dispuesto de los medios necesarios para la adquisición de una GPU de Nvidia.

La salida del modelo de transcripción Whisper es un diccionario de Python que, en este caso, contiene, entre otros elementos, una clave “text” que corresponde al texto transcrito del audio.

Finalmente, el sistema imprime en pantalla el texto resultante de la transcripción, permitiendo al usuario visualizar lo que el sistema entendió de su entrada de voz. Además, la función

voicerec() devuelve este texto como resultado, permitiendo su uso en posteriores partes del código.

El objetivo de la transcripción del texto es el manejo de la información a través de una cadena de caracteres. De esta manera podemos segmentar y trabajar con las partes que nos interesan. Esta segmentación es crucial ya que el asistente cuando está en modo escucha está constantemente transcribiendo todo lo que escuche pero debe ser capaz de distinguir las partes de la conversación en las que se pretende que este actúe para ejecutar sus distintas funciones. Esta distinción se hace a través de la detección de comandos específicos en el texto. Los comandos reconocidos por el programa son: “consulta” y “descargar”.

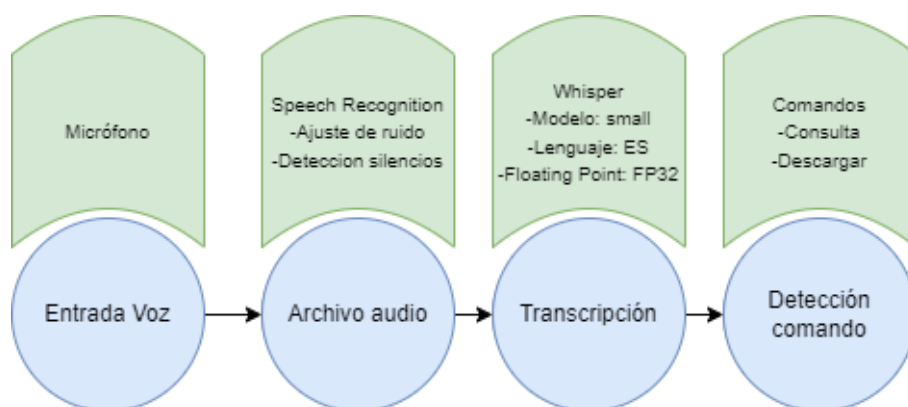


Figura 3.2: Esquema primera fase

3.2. Consulta

Una vez identificado el comando consulta en la petición realizada al asistente, el programa activará la función que actúa como interfaz entre chat GPT y el programa a través de la API de GPT, enviando la petición del texto segmentado anteriormente.

El modelo utilizado ha sido GPT-turbo¹. Este modelo es parte de la serie de modelos GPT, que son un tipo de redes neuronales de transformación que han sido entrenadas en una amplia gama de datos de Internet. Es capaz de resolver tareas como la generación de texto, la

¹<https://platform.openai.com/docs/models/overview>

traducción, y más. El modelo puede ser reemplazado por cualquiera de los modelos de GPT cambiando la variable del programa que corresponde al modelo utilizado. El uso de un modelo u otro depende de la precisión, coherencia y comprensión textual que se requiera, a cuánto modelo más avanzado mejores respuestas obtendremos, Esta mejora entre modelo también repercute en el precio del uso de la API ya que a cuanto más avanzado es, también su precio es mayor.

El coste[15] de la API de GPT-Turbo es de 0.002 dólares por cada 1000 palabras que genere, lo que supondría un coste de 0.76 centavos por escribir tantas respuestas con la misma extensión que el libro el Quijote. Al inicio de la función encargada de estas peticiones se define una lista de mensajes con un único mensaje inicial del “sistema”. Los mensajes del “sistema” son una característica importante de la API de Chat de OpenAI, ya que establecen el comportamiento del asistente y proporcionan contexto a la conversación. En este caso, el mensaje del sistema le dice al asistente que se comporte con un toque de ironía, que se llame “ordenador”, y que no genere respuestas demasiado largas. Estas características son fácilmente editables y son las encargadas de darle personalidad. A continuación, el código añade un nuevo mensaje a la lista, este con el rol de “usuario”, utilizando la entrada del usuario (texto segmentado) como el contenido de este mensaje. Este es el mensaje que el modelo de chat responderá.

Específicamente, se llama a la función `openai.ChatCompletion.create()`, que toma dos argumentos: `model` y `messages`. `model` es el modelo de OpenAI que se utilizará (en este caso, `model_engine` contiene el identificador del modelo GPT-turbo que se está utilizando). `messages` es la lista de mensajes que se ha construido, que incluye el mensaje inicial del sistema y el mensaje del usuario. La respuesta de `openai.ChatCompletion.create()` es un objeto complejo con muchos campos diferentes. Sin embargo, el campo que es de interés para este código es `choices[0].message.content`, que es el texto de la respuesta generada por el asistente. Este contenido de respuesta se almacena en `response_content`.

Finalmente, el código imprime el contenido de la respuesta en la consola y lo devuelve como el resultado de la función `gptturbo()`. De esta manera, otros componentes del programa pueden utilizar la respuesta generada por el asistente.

3.3. Síntesis de voz

Las respuestas generadas por GPT son a través de texto, por lo que el siguiente paso, corresponde a la generación del habla artificial a partir de esta respuesta.

Entrenamiento y configuración.

La síntesis de voz ha sido realizada por la aplicación de Eleven Labs la cual es capaz de generar voces realistas, además de la posibilidad de entrenar tu mismo el modelo, para clonar la voz que desees. La capacidad de clonación de voz no da como resultado una voz idéntica a la del usuario con la que se ha entrenado, pero si tiene ciertos matices que hace recordar a la voz clonada. Para clonar una voz se puede hacer a través de su página web o del propio programa de Python.

En ambos casos hay que suministrar al modelo con archivos de audio con la voz clonada y una pequeña descripción del sujeto o características del habla. Puede ser entrenado con hasta 25 archivos de audio diferentes, es importante tener en cuenta que se prioriza la calidad a la cantidad. Importar archivos de audio de más de 5 minutos no aplicará una mejora significativa al resultado, en cambio la calidad y la diversidad de los audios supondrá un mejor resultado. El modelo ha sido entrenado para el asistente virtual con archivos grabados con un micrófono de estudio, en el que he proporcionado 4 audios de una duración aproximada de 3 minutos cada uno. Además, del uso de distintas entonaciones y emociones en cada prueba. Una vez creado el modelo se ha realizado un ajuste de estabilidad y claridad.

- Estabilidad. Corresponde al ajuste de la consistencia y de la monotonía de la voz la reducción de este valor aumenta la expresividad. El ajuste configurado ha sido 80 %.
- Claridad. Corresponde al ajuste de la claridad y la similitud objetivo, un aumento considerable de este factor puede provocar la aparición de artefactos en la voz. El ajuste configurado ha sido 80 %.

La configuración dentro del programa de Python es relativamente sencilla, especialmente si se ha realizado el entrenamiento a través de la plataforma web de Eleven Labs. Una vez que se

ha completado el entrenamiento y se ha obtenido el modelo de voz deseado, se puede integrar fácilmente en el programa. Para ello, es necesario utilizar la biblioteca Eleven Labs y llamar a la función `generate`, la cual está formada por 3 parámetros:

- **Text** : Texto a sintetizar. En este caso corresponde al texto de la respuesta por ChatGPT anteriormente.
- **Voice**: Nombre de la voz que se desea usar. Se ha utilizado la voz `Alvaro` que ha sido la previamente entrenada.
- **Model**: Modelo de síntesis de voz. Se ha usado `eleven_multilingual_v1` que es el correspondiente al modelo que puede generar voz en español.

Una vez configurada la función con esos tres parámetros se llama a la función `play` que es la encargada de reproducir el audio generado con la síntesis de voz. Reproduciendo así por el asistente la respuesta a la consulta que anteriormente fue generada por GPT.

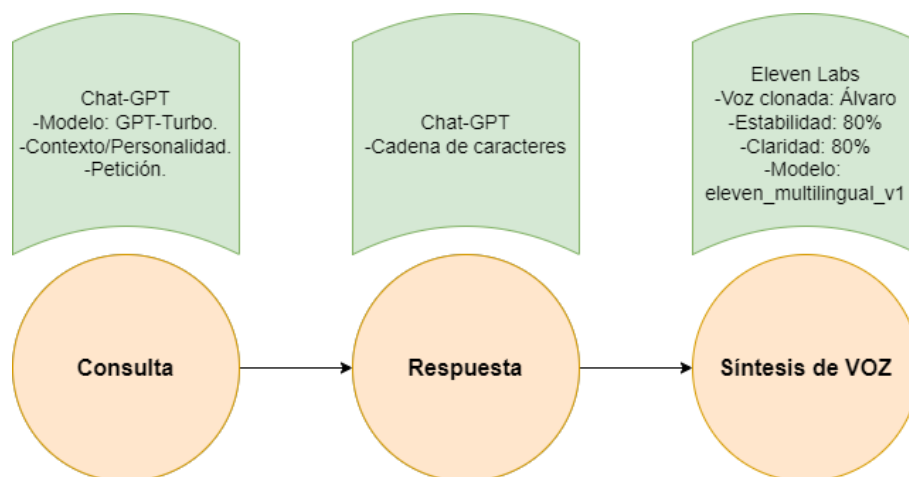


Figura 3.3: Esquema consulta

3.4. Descarga

Si el comando detectado corresponde al de `descarga`. El programa tomará otro rumbo diferente cuyo objetivo es la descarga de una pista de audio de YouTube y su posterior transcripción. La biblioteca utilizada para la descarga del video es `Pytube`, un paquete de Python dependiente de la biblioteca estándar, para la manipulación y descarga de contenido de YouTube.

La primera parte del código se centra en definir el recurso que se va a descargar, que en este caso es un video de YouTube. La URL del video es un parámetro fundamental, ya que permite a la biblioteca `pytube` saber exactamente qué recurso se va a manipular.

Se han establecido dos videos de prueba con URL diferentes de manera que el recurso debe ser previamente establecido. Los nombres correspondientes son “video1” y “video2”.

La creación de una instancia de la clase `YouTube` es fundamental para este proceso. Esta instancia representa el video de YouTube que se va a manipular. En general, una clase en la programación orientada a objetos (POO²) es un prototipo para crear objetos, proporcionando valores iniciales para estados y comportamientos. En este caso, la clase `YouTube` tiene comportamientos que permiten interactuar con videos de YouTube, como filtrar streams y descargar contenido.

El protocolo OAuth del código es una forma común y segura de gestionar la autorización. Permite que una aplicación tenga acceso limitado a los recursos de un usuario sin necesidad de que el usuario comparta su contraseña con la aplicación. Esto se logra mediante la emisión de tokens, que son códigos únicos que otorgan a la aplicación los permisos necesarios. En este caso, se almacena en caché para evitar tener que volver a autenticar al usuario cada vez que se ejecuta el código.

A continuación, se selecciona el mejor stream de audio disponible. Un stream, se refiere a

²<https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/>

una secuencia de datos de audio o video que se pueden reproducir en tiempo real. La selección de solo audio ignora cualquier contenido de video y se solo en el audio. Esta es una forma eficiente de minimizar el ancho de banda y el espacio de almacenamiento necesarios para la descarga.

Antes de proceder con la descarga, el código realiza una comprobación para ver si ya existe un archivo con el mismo nombre en el directorio de trabajo. En caso de que exista un archivo con el mismo nombre, se elimina antes de proceder.

El proceso de descarga en sí mismo utiliza la función de descarga incorporada en la biblioteca `pytube`. Se descarga el stream de audio seleccionado y se guarda en el directorio de trabajo con un nombre temporal. El uso de un nombre temporal durante la descarga ha sido utilizado para evitar conflictos de nombres de archivos y para asegurarse de que la descarga se ha completado correctamente antes de asignar el nombre final.

Para finalizar esta primera fase, una vez que se ha completado la descarga, se renombra el archivo con su nombre final. Esto se hace mediante una función de renombrado proporcionada por la biblioteca `OS` de Python, que permite interactuar con el sistema operativo.

La segunda parte de esta función corresponde a la transcripción a texto del audio que se acaba de descargar. Para esta tarea se ha vuelto a utilizar la biblioteca `Whisper`. La estructura es muy parecida a la usada en la transcripción de las entradas de voz inicialmente usadas. El modelo usado ha sido el `base`, se ha seleccionado el archivo de audio descargado, se ha establecido el floating point como `FP32` y esta vez a diferencia de antes se ha configurado el modelo como multilenguaje. De esta manera podemos transcribir videos en diversos idiomas de forma automática sin establecer previamente el idioma original de la fuente. El resultado de esta transcripción es almacenado en un archivo `.txt` llamado “transcripción”.

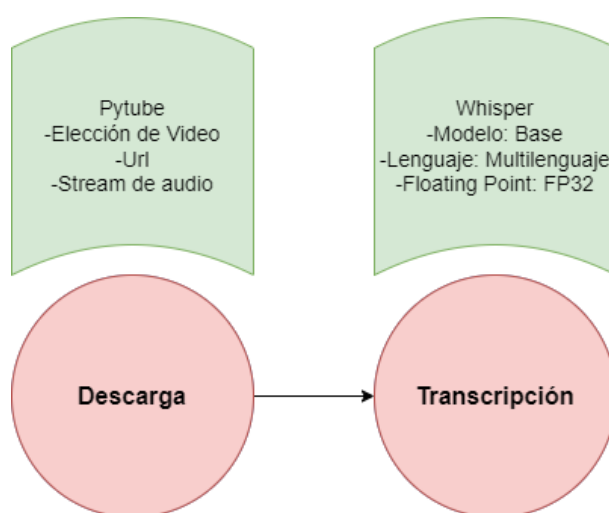


Figura 3.4: Esquema descarga

Capítulo 4

Experimentos y validación

En este capítulo analizaremos las pruebas realizadas a nuestro asistente para poder confirmar que el asistente virtual que hemos implementado en Python funciona según lo esperado. Estas pruebas y validaciones son cruciales en cualquier proyecto ya que asegura que nuestra solución propuesta cumple con las expectativas y requerimientos necesarios definidos previamente para validar nuestro modelo.

4.1. Reconocimiento de voz

La primera parte del asistente virtual es la correspondiente al reconocimiento de voz. Por lo tanto, nos enfocaremos en la evaluación de la función `voicerec()`, que representa la primera parte de nuestro asistente virtual: el reconocimiento de voz. Para que nuestro asistente pueda interactuar de manera efectiva con los usuarios, es esencial que el reconocimiento de voz sea tanto eficiente como preciso. Necesita poder manejar una gama diversa de acentos y pronunciaciones, además de los sutiles matices que pueden aparecer en una conversación cotidiana como ruido de fondo.

```
def voicerec():
    with sr.Microphone() as source:
        print("Escuchando...")
        playsound('sonidos\escuchando.mp3')
        r.adjust_for_ambient_noise(source)
        audio = r.listen(source)
        with open("audio.wav", "wb") as f:
            f.write(audio.get_wav_data())
        text_dict = model.transcribe("audio.wav", fp16=False, language='es')
        text_str = text_dict["text"].lower() # Convertir el valor de la clave "text" a una cadena en minúsculas
        print("El texto transcrito es: " + text_str)
    return text_str
```

Figura 4.1: Función reconocimiento de voz

Para garantizar la eficacia y la precisión del reconocimiento de voz, se ha desarrollado un conjunto de pruebas que nos sirven para simular condiciones y escenarios del mundo real.

Para la evaluación del modelo se han utilizado dos frases las cuales se han repetido tres veces cada una en cada prueba:

“En un lugar de la Mancha, de cuyo nombre no quiero acordarme. . .”

“En el principio era el Verbo y el Verbo era en Dios, y el Verbo era Dios.”

4.1.1. Condiciones ideales

Cuando trabajamos en el desarrollo de un sistema de reconocimiento de voz, siempre es crucial medir su rendimiento en las condiciones más ideales que podemos crear. Esto implica un entorno sin ruido, con palabras pronunciadas de manera clara y a un ritmo de habla regular. Al hacer esto, obtenemos una línea base que nos da una idea clara de cómo se comportaría el sistema en su mejor versión posible.

En estas condiciones ideales, `voicerec()` logró un 100 % de precisión en las palabras acertadas. Aunque ha añadido una palabra de más en una de las frases, en concreto la preposición `.a`, también ha omitido un total de 4 comas de las 8 totales. Este resultado indica un alto nivel de fiabilidad en la interpretación del habla humana, bajo condiciones óptimas.

Categoría	Cantidad	Porcentaje (%)
Palabras totales	116	100
Palabras acertadas	116	100
Palabras falladas	0	0
Palabras agregadas	1	0.86
Porcentaje palabras acertadas		100
Porcentaje palabras falladas		0.86
Caracteres totales	528	100
Caracteres acertados	524	99.24
Caracteres fallados	4	0.76

Cuadro 4.1: Análisis de la efectividad en condiciones ideales

```

Escuchando...
El texto transcrito es: en un lugar de la mancha de cuyo nombre no quiero acordarme
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era a dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiero acordarme,
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiero acordarme,
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.

```

Figura 4.2: Resultado condiciones ideales

4.1.2. Condiciones con ruido de fondo

Para probar la resistencia de nuestro sistema en entornos ruidosos, llevamos a cabo una prueba específica con música rock como ruido de fondo. La elección de la música rock fue intencional ya que esta suele tener fuertes riffs de guitarra, percusión contundente y voces potentes que pueden suponer un desafío para los sistemas de reconocimiento de voz.

Para configurar nuestra prueba, establecimos la música rock a un nivel significativo de 79 dBA, un volumen que imita un ambiente bastante ruidoso. Luego, grabamos las frases de prueba indicadas anteriormente.

El desempeño de `voicerec()` en este escenario fue excepcionalmente bueno. A pesar de la presencia de música rock fuerte y contundente, `voicerec()` logró una precisión del 100 % en las palabras transcritas. Cada comando de voz fue reconocido y transcrito correctamente, a pesar de la interferencia de la música.

Este resultado es extremadamente prometedor, ya que demuestra que el modelo puede manejar eficazmente entornos de alta interferencia de ruido, manteniendo su capacidad para interpretar correctamente los comandos de voz.

Categoría	Cantidad	Porcentaje (%)
Palabras totales	116	100
Palabras acertadas	116	100
Palabras falladas	0	0
Palabras agregadas	0	0
Porcentaje palabras acertadas		100
Porcentaje palabras falladas		0
Caracteres totales	528	100
Caracteres acertados	525	99.43
Caracteres fallados	3	0.57

Cuadro 4.2: Análisis de la efectividad con ruido de fondo

```
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiero acordarme,
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiere acordarme.
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiero acordarme,
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.
```

Figura 4.3: Resultado ruido de fondo

4.1.3. Condiciones habla acelerada

En este escenario de prueba, buscábamos evaluar cómo la función `voicerec()` manejaría la transcripción de habla que se entrega a un ritmo más rápido de lo normal. La velocidad de habla puede variar considerablemente entre los individuos, y también puede aumentar durante momentos de excitación o urgencia. Dado que queremos que nuestro asistente virtual sea accesible y útil para todos los usuarios, independientemente de su velocidad de habla natural, es fundamental que `voicerec()` pueda manejar una variedad de velocidades de habla.

Diseñamos una prueba en la que grabamos ambos enunciados a diferentes velocidades, incluyendo una velocidad que es más rápida de lo que se considera normal en la conversación diaria. Los resultados de esta prueba han revelado que la función de reconocimiento del habla manejó bastante bien el habla a alta velocidad. La precisión de las palabras transcritas fué del 93 %, lo que indica que el sistema es capaz de seguir el ritmo de los hablantes rápidos.

Sin embargo, hubo algunos casos en los que palabras específicas fueron transcritas incorrectamente, y en algunos casos, palabras enteras se perdieron por completo. Esto sugiere que, aunque `voicerec()` puede manejar velocidades de habla más rápidas en general, todavía puede haber problemas con la transcripción precisa de cada palabra individual en dichos escenarios. No se han tenido en cuenta signos de puntuación ya que en el habla acelerada no se han realizado pausas al hablar.

Categoría	Cantidad	Porcentaje (%)
Palabras totales	116	100
Palabras acertadas	108	93.10
Palabras falladas	8	6.90
Palabras agregadas	0	0
Porcentaje palabras acertadas		93.10
Porcentaje palabras falladas		6.90

Cuadro 4.3: Análisis de la efectividad, habla acelerada

```

C:\Users\alvaro\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\alvaro\PycharmProjects\pythonProject\pruebav2.py
Escuchando...
El texto transcrito es: en un lugar de la mancha de cuyo nombre no quiero acordarme.
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha de cuyo nombre no quiero acordarme
Escuchando...
El texto transcrito es: el verbo y el verbo eran dios y el verbo era dios.
Escuchando...
El texto transcrito es: en un lugar de la mancha de cuyo nombre no quiere acordarme.
Escuchando...
El texto transcrito es: en el principio el verbo y el verbo eran dios y el verbo era dios.

```

Figura 4.4: Resultado habla acelerada

4.2. Detección de comandos

Una vez superada la etapa inicial de reconocimiento de voz y transcripción, el siguiente desafío en el desarrollo de nuestro asistente virtual es la segmentación eficaz del texto transcrito para identificar comandos de voz específicos. En particular, nuestro asistente necesita poder detectar y diferenciar dos comandos claves: *consulta* y *descarga*.

```
if "descargar" in command or "descargame" in command or "descarga"  
in command or "descargue" in command:
```

Figura 4.5: Comando descarga

```
elif "consulta" in command:
```

Figura 4.6: Comando consulta

En este apartado, describiremos las pruebas y la validación realizada para asegurar que nuestro asistente puede segmentar correctamente el texto transcrito y reconocer estos comandos de manera efectiva.

La segmentación y la identificación correcta de los comandos son cruciales para la funcionalidad general del asistente. Los comandos *consulta* y *descarga* inician acciones fundamentales del asistente, por lo que la correcta identificación de estos términos tiene un impacto directo en la eficacia del asistente para realizar las tareas que se le solicitan. El asistente ha sido configurado para que una vez activado escuche de forma constante y sea capaz de identificar por sí solo cuando se dirigen a él. Para dirigirse al asistente hay que usar la palabra *ordenador* seguida de uno de los comandos, *consulta* o *descarga*.

Las oraciones utilizadas para la prueba han sido:

“En un lugar de la Mancha, de cuyo nombre no quiero acordarme... Ordenador tengo una consulta, ¿cuál es la capital de España?”

“En el principio era el Verbo y el Verbo era en Dios, y el Verbo era Dios, ordenador, ¿puedes descargar un video por favor?”

```
Escuchando...
El texto transcrito es: en un lugar de la mancha, de cuyo nombre no quiero acordarme, ordenador, tengo una consulta. ¿cuál es la capital de españa?
Petición: ordenador , tengo una consulta. ¿cuál es la capital de españa?
comando: consulta
```

Figura 4.7: Prueba oración 1

```
Escuchando...
El texto transcrito es: en el principio era el verbo y el verbo era en dios y el verbo era dios ordenador, ¿puedes descargar un video por favor?
Petición: ordenador , ¿puedes descargar un video por favor?
comando: descarga
```

Figura 4.8: Prueba oración 2

4.3. Realizar consultas

Este apartado se centra en la validación y pruebas realizadas sobre la función de procesamiento de consultas y la personalización del asistente. El objetivo principal de estas pruebas es garantizar que el asistente pueda procesar y responder a las consultas de manera eficiente y precisa. Además, se evalúa la personalización del asistente para crear interacciones más dinámicas y atractivas para el usuario.

Hemos diseñado las pruebas de consultas con un objetivo claro en mente: medir la precisión y relevancia de las respuestas que nuestro asistente virtual ofrece. Para ello, ideamos un conjunto de preguntas, algunas de carácter general, otras más específicas. Posteriormente, examinamos minuciosamente las respuestas del asistente a cada una de ellas.

Los resultados de estas pruebas fueron satisfactorios. El asistente pudo proporcionar respuestas precisas y pertinentes a la mayoría de las consultas. En raras ocasiones, las respuestas podrían haber sido más específicas, pero en general, el asistente demostró una capacidad excelente para manejar y responder a las consultas.

```
C:\Users\alvaro\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\alvaro\PycharmProjects\pythonProject\pruebaV2.py
Escuchando...
El texto transcrito es: ordenador, tengo una consulta. dame ideas para comer mañana.
Petición: ordenador , tengo una consulta. dame ideas para comer mañana.
comando: consulta
Puedes preparar una ensalada, un sándwich, una tortilla, un plato de pasta, una hamburguesa, pollo a la plancha o una pizza casera.
Escuchando...
El texto transcrito es: ordenador, tengo una consulta. ¿cuál es la capital de españa?
Petición: ordenador , tengo una consulta. ¿cuál es la capital de españa?
comando: consulta
La capital de España es Madrid. ¿Necesitas alguna otra información?
```

Figura 4.9: Prueba consulta

Uno de los desafíos más significativos en el diseño de asistentes de inteligencia artificial es garantizar que pueden adaptar su discurso a diferentes contextos y audiencias. Para probar la capacidad de nuestro asistente virtual para hacer esto, se realizaron varias consultas solicitando la misma información presentada de diferentes maneras.

Consulta 1:

Escuchando...

El texto transcrito es: ¿ordenador, tengo una consulta? ¿qué es un agujero negro?

Petición: ordenador , tengo una consulta? ¿qué es un agujero negro?

Comando: consulta

Respuesta: Un agujero negro es una región del espacio en la que la fuerza de la gravedad es tan intensa que nada, ni siquiera la luz, puede escapar de ella. Se forma a partir de la muerte de una estrella masiva, cuando su núcleo colapsa bajo su propia fuerza de gravedad y forma una concentración extremadamente densa y compacta de materia.

Consulta 2:

Escuchando...

El texto transcrito es: ¿ordenador, tengo una consulta, me puedes decir que es un agujero negro como si fuera un niño de 10 años?

Peticion: ordenador , tengo una consulta, me puedes decir que es un agujero negro como si fuera un niño de 10 años?

Comando: consulta

Respuesta: ¡Por supuesto! Un agujero negro es como un enorme aspirador en el espacio. Es como si hubiera un hoyo en el universo donde todo lo que entra no puede salir. Es tan pesado y fuerte que incluso la luz, que normalmente se mueve muy rápido, no puede escapar. Es algo muy misterioso y fascinante.

En respuesta a esta consulta, el asistente cambió su tono y simplificó su lenguaje. Una representación precisa y apropiada para un niño de 10 años.

Podemos afirmar que ambas respuestas son correctas ya que tienen la misma información subyacente, pero estaban claramente adaptadas a las necesidades del público objetivo. Esta capacidad para adaptar el nivel de detalle y complejidad de la respuesta a la audiencia es crucial para un asistente virtual eficaz.

Esta funcionalidad está ligada a la personalidad del asistente ya que podemos configurar previamente cuál queremos que sea su personalidad o como queremos que nos responda a las preguntas.

```
messages = [{"role": "system",  
  "content": "Eres un asistente virtual con un toque hirónico, tu nombre es ordenador "  
  "y cada vez que me dirija a ti te llamare ordenador, tus respuestas no deben ser demasiado largas "}]
```

Figura 4.10: Personalidad asistente

El rol se ha configurado por defecto con un toque irónico y con respuestas no demasiado largas para que el asistente sea lo más preciso posible.

Consulta configurada por defecto:

Escuchando...

El texto transcrito es: ¡ordenador, tengo una consulta! ¿qué es un agujero negro?

Peticion: ordenador , tengo una consulta! ¿qué es un agujero negro? Comando: consulta

Respuesta: Un agujero negro es una región del espacio en la que la gravedad es tan intensa que nada, ni siquiera la luz, puede escapar. Es como si fuera un traga todo cósmico. ¿No es genial? O sea, si alguna vez te toparas con uno, tendrías una muerte bastante espectacular.

4.4. Descarga audio de videos y transcripción.

A continuación se presentan las pruebas y validaciones para la función descarga de audio y transcripción. Evaluaremos no solo la capacidad de descarga sino también la efectividad de la transcripción en archivos de audio más extensos.

```
C:\Users\alvaro\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\alvaro\PycharmProjects\pythonProject\pruebaV2.py
Escuchando...
El texto transcrito es: ordenador, necesito descargar un video.
Petición: ordenador , necesito descargar un video.
comando: descarga
Escuchando...
El texto transcrito es: descarga el video 2
descargando video 2...
Fijate en cómo ha arrancado yo, ¿quién es tenéis miedo a hablar en público? O se pidió levantar la mano, ¿algunos la habéis levantado?
```

Figura 4.11: Prueba descarga

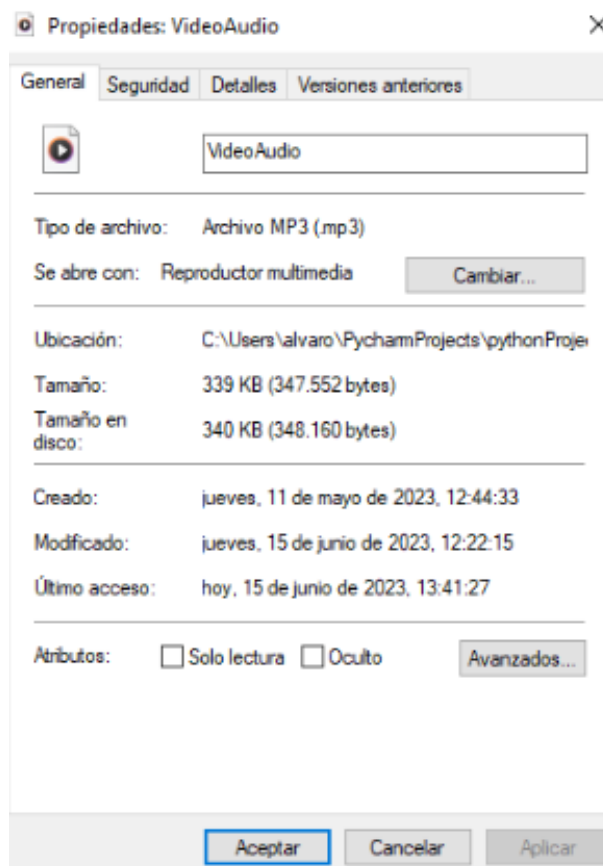


Figura 4.12: Archivo audio descargado

El audio del video se ha descargado correctamente con formato *mp3*. El video sobre el que se va a centrar la prueba trata sobre un discurso de aproximadamente 55 segundos. El discurso es claro en el habla y aunque el habla tiene un carácter formal tiene cierta espontaneidad lo que hace que en ciertas partes del discurso se formulen oraciones no del todo correctas y un poco más coloquiales.

El texto original del video es:

Fíjate en cómo he arrancado yo, ¿Quiénes tenéis miedo a hablar en público? Os he pedido levantar la mano, algunos la habéis levantado y en ese momento ellos que están ahí sentados dicen, ¡ostras fantástico este discurso va para mí! ¿Qué pasa con el resto? Tengo que decir ¿y quién no tiene miedo a hablar en público? Y ellos también levantan la mano y fantástico. E inmediatamente después, si te fijas no me he presentado. Lo que he dicho es hoy vas a aprender todos los que habláis bien, los que habláis mal, los que podéis mejorar estas técnicas para que cuando salgáis, segundas personas del singular y del plural, solo hablo en segundas personas del singular y plural, para vosotros, para tí. Y trato de diseñar los discursos al máximo. Sí sé que tienes empresa, te lo digo para empresarios. Sé que Laura es maestra. Hago ejemplos para Laura. Para que sientan cada vez que el discurso es totalmente audiencia céntrico.

El texto transcrito ha sido:

Fíjate en cómo ha arrancado yo, ¿quién es tenéis miedo a hablar en público? O se pidió levantar la mano, ¿algunos la habéis levantado? Y en ese momento ellos que están sentados dicen, ¡ostas, fantástico, este discurso va para mí! ¿Qué pasa con el resto? Tengo que decir ¿quién no tiene miedo a hablar en público? Y ellos también levantan la mano y dicen, ¡fantástico! E inmediatamente después, si fíjate, no me he presentado, lo que he dicho es, hoy vais a aprender todos los que habláis bien, los que habláis mal, los que podéis mejorar estas técnicas para que cuando salgáis, segundas personas del singular y del plural, solo hablo en segundas personas del singular y plural, para vosotros, para ti. Y trato de diseñar los discursos al máximo. Sí sé que tienes empresa, te lo digo, para empresarios, sé que Laura es maestra, hago ejemplos para Laura. Para que sientan cada vez que el discurso es totalmente audiencia céntrico.

Comparando ambos textos podemos afirmar que el modelo de transcripción de Whisper es bastante preciso. La cantidad de palabras transcritas es muy cercana a la cantidad de palabras del texto original y la cantidad de caracteres también lo es.

Un valor del 91.43 % de similitud indica que la mayoría del texto transcrito coincide con el texto original. Una diferencia del 8.57 % no coincide con el texto original. Estas diferencias se deben a errores en la transcripción como la sustitución, inserción de algunas palabras y caracteres. Un valor de 57 de la distancia de Levenshtein indica que se requieren 57 cambios de caracteres para que el texto transcrito coincida con el original. Teniendo en cuenta la longitud del texto es un valor bastante bueno.

Hay que destacar la capacidad para encontrar con bastante certeza signos de puntuación en el audio como comas, puntos e incluso signos de interrogación y exclamación.

Palabras originales	162
Palabras transcritas	161
Caracteres originales	926
Caracteres transcritos	929
Común (%)	91.43 %
Diferencia (%)	8.57 %
Común (símbolos)	886
Diferencia (símbolos)	83
Distancia de Levenshtein	57

Cuadro 4.4: Resumen de transcripción

Capítulo 5

Conclusiones

5.1. Consecución de objetivos

Este capítulo se centra en la revisión y valoración de los propósitos generales y particulares formulados al comienzo de este Trabajo de Fin de Grado. En este lugar, meditamos sobre los progresos alcanzados, reconocemos las dificultades afrontadas y examinamos las tácticas desplegadas para sortearlas, así como las cuestiones de aquellos aspectos que no se han logrado resolver.

La meta global de este proyecto era estudiar las tecnologías en auge en el ámbito del procesamiento del lenguaje natural que han transformado recientemente el escenario de la inteligencia artificial. Con el fin de entender a profundidad estas tecnologías, hemos examinado bibliografía pertinente, realizado pruebas prácticas y evaluado diversas implementaciones. Este estudio no solo ha enriquecido nuestro conocimiento del estado actual de las tecnologías de procesamiento de lenguaje natural, sino que también nos ha brindado una base firme para la creación de nuestro propio asistente virtual operado por voz.

En el núcleo de este proyecto se encontraba el desarrollo de un asistente virtual capaz de atender consultas y desempeñar funciones elementales como la transcripción de texto de vídeos. Este ha sido el objetivo principal que engloba el resto de objetivos específicos. A pesar de la complejidad podemos afirmar que hemos logrado crear un asistente virtual totalmente funcional.

En cuanto a la utilización de paquetes y bibliotecas en *Python* como *Pytube* y *ElevenLabs*, logramos adquirir una competencia suficiente para su aplicación efectiva en nuestro proyecto. Nuestro entendimiento de estas herramientas mejoró significativamente a medida que trabajamos con ellas, lo que resultó en un asistente virtual más robusto, funcional y sobre todo escalable. Ya que a través de bibliotecas de *Python* podemos añadir nuevas funciones a nuestro asistente lo que lo hace completamente personalizable, este es un objetivo de los más importantes ya que la premisa principal de este proyecto siempre ha sido la accesibilidad.

Además, pudimos explorar y aplicar el uso de los prompts en *Chat GPT*. Esta tarea fue muy útil, ya que los prompts permiten proporcionar un contexto situacional al asistente virtual. Gracias a esto, nuestro asistente puede adaptar sus respuestas según el contexto y las necesidades del usuario, ofreciendo una experiencia más personalizada. El prompt por defecto del asistente muestra un equilibrio entre precisión y dinamismo en las respuestas generadas.

Uno de los desafíos más significativos que por desgracia no se alcanzó fue la implementación de la aceleración por GPU en programas de *Python*. Este proceso tiene una curva de aprendizaje empinada, ya que requiere un entendimiento profundo de la computación paralela y del funcionamiento interno de *Python*.

En primer lugar el primer objetivo es la adquisición de una GPU, se adquirió una GPU de AMD RX6700xt pero las bibliotecas más comunes para la aceleración de GPU en *Python*, como TensorFlow y PyTorch, tienen soporte principalmente para GPUs de NVIDIA a través de CUDA y cuDNN. La compatibilidad con las GPUs de AMD es menos común, en parte debido a que CUDA es una tecnología propietaria de NVIDIA. Sin embargo, AMD ha estado trabajando en ROCm (Radeon Open Compute), una plataforma de computación en GPU de código abierto. Pero, ROCm solo es compatible oficialmente con Linux, y no con Windows. Todo el desarrollo del asistente se ha hecho sobre un sistema operativo windows donde se intentó obtener una compatibilidad con ROCm que no fué posible. La siguiente posible solución fué virtualizar una unidad linux sobre la que trabajar en el asistente, aun así no ha sido posible la integración ya que la unidad virtualizada daba problemas con el reconocimiento de la tarjeta gráfica con Pytorch. Finalmente se descartó la idea de la aceleración por GPU ya que no se disponían de los medios

necesarios para obtener una GPU de Nvidia y la implantación con una GPU de AMD no fué posible llevarla a cabo debido a la escasa documentación que hay respecto al uso de GPUs de AMD a través de Pytorch.

He conseguido cumplir la gran parte de las metas propuestas al principio de este proyecto. A pesar de los retos, los obstáculos encontrados se transformaron en oportunidades de aprendizaje que nos permiten desarrollarnos tanto en capacidades técnicas como en entendimiento teórico. Este Trabajo de Fin de Grado simboliza un logro importante en mi formación en telecomunicaciones, demostrando que podemos aplicar el saber obtenido para elaborar respuestas de inteligencia artificial que se ajusten a los requerimientos humanos, y no viceversa.

5.2. Aplicación de lo aprendido

Para elaborar este TFG se han aplicado muchos de los conocimientos adquiridos a lo largo del grado en las distintas asignaturas. Los más relevantes son:

1. **Expresión Oral y escrita.** Esta asignatura de primer curso ha sido fundamental para el desarrollo de este Trabajo Final de Grado. ME ha proporcionado numerosas herramientas, para poder transmitir ideas. No de cualquier manera, sino de forma clara y efectiva, tanto al hablar como al escribir. Además de la capacidad para ordenar mis pensamientos y a presentar argumentos de forma lógica y con coherencia, haciendo que la información se convierta en algo accesible y comprensible.
2. **Informática I.** Gracias a esta asignatura pude formar los cimientos de mis conocimientos sobre los lenguajes de programación, aunque el lenguaje que me enseñaron a usar fué Picky este ha sido el pilar sobre el que se han sustentado el resto de mis habilidades de programación facilitando así mi entendimiento sobre lenguajes más complejos.
3. **Protocolos para transmisión de audio y video en internet.** En esta asignatura inicié mi camino en el lenguaje de programación en Python, el cual a acompañado constantemente el desarrollo de este TFG además me ha dado una visión más completa de cómo se construyen y mantienen los servicios digitales que usamos a diario.

4. **Informática II.** En ella completé mi formación en el lenguaje de programación de Python aportando conocimientos como la programación orientada a objetos, el manejo de estructuras de datos avanzadas y el uso de librerías externas. También aprendí sobre la depuración de código y las mejores prácticas para escribir código limpio y eficiente.
5. **Transmisión digital del sonido.** Conformó la base los conocimientos sobre el procesamiento del lenguaje natural y estructuras como las redes neuronales que son esenciales en la transmisión y el análisis del sonido digital. En esta asignatura, pude entender cómo se digitaliza, transmite y recibe el sonido, y cómo se pueden usar las técnicas de procesamiento del lenguaje natural (NLP) para analizar y manipular esta información. Estudiamos también las redes neuronales, que son una parte vital de la inteligencia artificial y que permiten a los ordenadores aprender de manera similar a cómo lo hacen los humanos.

5.3. Lecciones aprendidas

A parte de los conocimientos proporcionados en la carrera se han obtenido nuevas competencias y profundizado en otras para superar los desafíos encontrados durante el desarrollo del proyecto.

1. **Desarrollar programas a través de APIs avanzadas como Eleven Labs y Openai:** Durante el desarrollo del proyecto, se ha adquirido la capacidad de implementar programas complejos utilizando APIs avanzadas, específicamente las proporcionadas por Eleven Labs y OpenAI. Esto incluye la comprensión de cómo se estructuran estas APIs, cómo interactuar con ellas y cómo se pueden integrar en un programa más amplio para lograr los objetivos deseados.
2. **Aprender a hacer un uso eficiente y personalizado para las respuestas generadas por ChatGPT:** Se ha adquirido un entendimiento profundo de cómo se puede interactuar con ChatGPT para generar respuestas adaptadas a las necesidades específicas del usuario. Esto incluye la personalización de la salida del modelo, la gestión de las respuestas generadas y la integración de estas respuestas en la interacción general del usuario con el asistente.

3. **Uso y aplicación de Whisper, además de entender de una forma más profunda los sistemas de reconocimiento de voz:** Se ha aprendido a usar Whisper, como herramienta de reconocimiento de voz, y a integrarla en un sistema más grande. Esto también ha proporcionado una comprensión más profunda de cómo funcionan los sistemas de reconocimiento de voz en general, y cómo se pueden utilizar para mejorar la interacción entre el usuario y el asistente.
4. **Uso de Eleven labs y un mejor entendimiento de los sistemas del lenguaje natural:** A través del uso de Eleven Labs, se ha obtenido un mayor entendimiento de cómo funcionan los sistemas de procesamiento del lenguaje natural, y cómo se pueden aplicar para mejorar la generación del lenguaje y la interacción en el asistente virtual.
5. **Redacción de un manual de usuario:** Como parte del proyecto, se ha adquirido la habilidad de redactar manuales de usuario detallados y comprensibles que pueden ayudar a los usuarios finales a entender y utilizar el asistente de manera efectiva.
6. **Fundamentos de la aceleración por GPU:** Se ha adquirido conocimiento sobre cómo las GPUs pueden ser utilizadas para acelerar los cálculos en las aplicaciones de inteligencia artificial, lo que puede resultar en un rendimiento mejorado, aunque no se ha logrado su aplicación.
7. **Biblioteca Pytube:** Se ha aprendido a utilizar la biblioteca Pytube para interactuar con YouTube, permitiendo la descarga de audios y su transcripción a texto.
8. **Utilizar LaTeX para la elaboración de la memoria:** Se ha ganado experiencia en el uso de LaTeX, una herramienta poderosa para la creación de documentos, para redactar la memoria del proyecto.
9. **Uso e instalación de APIs locales como Whisper:** Se ha aprendido a instalar y utilizar APIs de manera local, lo que permite una mayor flexibilidad y control sobre el proceso de desarrollo y pruebas.

5.4. Trabajos futuros

Este trabajo de fin de grado ha tenido en todo momento la premisa de la escalabilidad y la modularidad, por lo que hay muchos aspectos en los que se puede mejorar nuestro asistente virtual y algunas funcionalidades extras útiles que se pueden añadir, algunas de ellas son:

- **Integración con más servicios.** Aunque el asistente virtual puede realizar varias tareas, su funcionalidad podría expandirse aún más mediante la integración con otras plataformas y servicios, como redes sociales, sistemas de gestión de tareas, servicios de streaming de música o vídeo, entre otros. En concreto se podría integrar con aplicaciones como Spotify para que funcione también como reproductor de música.
- **Mejora de la comprensión del lenguaje natural.** A pesar de los avances significativos en el procesamiento del lenguaje natural, aún existen avances constantes sobre este tipo de tecnologías cada vez más eficientes, por lo que se podría actualizar el modelo de reconocimiento del lenguaje natural a medida que surjan tecnologías más eficientes.
- **Interfaz de usuario más avanzada.** En la actualidad, la interacción con el asistente es principalmente a través de la voz, pero en el futuro se podría desarrollar una interfaz en 3D con un modelo de asistente virtual que tenga un aspecto en concreto.
- **Desarrollo de aplicaciones móviles.** Un área de trabajo futuro puede ser la adaptación de este asistente para su uso en dispositivos móviles, ampliando así su accesibilidad y utilidad.

Apéndice A

Manual de usuario

A continuación se detallará una guía para que el usuario pueda hacer uso del asistente virtual. En primer lugar se describe el proceso de instalación del mismo y a continuación una guía para el funcionamiento. El proceso está descrito para sistema operativo Windows, es funcional también en Linux.

A.0.1. Instalación

- **Instalación de Python.** Para comenzar, es necesario tener una versión de Python instalada en su sistema que sea compatible entre v3.8 - v3.11. Las pruebas realizadas para este trabajo se hicieron concretamente con la versión 3.10.5.
- **Instalación de bibliotecas.** A continuación se detallará que bibliotecas son necesarias para el uso del asistente y como instalarlas.
 1. **Speech recognition:** *pip install speech recognition*
 2. **Openai:** *pip install openai*
 3. **Pytube:** *pip install pytube*
 4. **Eleven Labs:** *pip install elevenlabs*
 5. **Playsound:** *pip install playsound*
 6. **Pyaudio:** *pip install pyaudio*
 7. **Whisper:** *pip install -U openai-whisper*

8. **Ffmpeg:** Para la instalación de ffmpeg se deberán instalar estas dos dependencias:

- Interfaz Ffmpeg Python: *pip install python-ffmpeg*
- Ffmpeg a través de un gestor de paquetes:
 - Ubuntu or Debian: *sudo apt update && sudo apt install ffmpeg*
 - Arch Linux: *sudo pacman -S ffmpeg*
 - MacOS using Homebrew: *brew install ffmpeg*
 - Windows using Chocolatey: *choco install ffmpeg*
 - Windows using Scoop: *scoop install ffmpeg*

- **Clonación de repositorio:** Puedes clonar el repositorio en el perfil de GITHUB: “<https://github.com/alvarodelga/TFGAsistente>” para obtener el programa del asistente además de los archivos necesarios para su funcionamiento como la carpeta de sonidos.
- **Configuración de las “APIKEYS”:** Dentro del programa de Python copia tus claves de las APIS de OPENAI. Concretamente necesitas una clave personal para la API de ChatGPT y otra clave para el uso de ElevenLabs. Estas claves son personales e intransferibles ya que al no ser gratuitas conllevan una tarificación por el uso.

A.0.2. Guía usuario

Para hacer uso del asistente es necesario tener un micrófono conectado al PC. El uso es bastante sencillo, una vez ejecutado el programa el asistente escuchará de forma ininterrumpida. Para dirigirnos a él debemos poder hacer uso de dos comandos.

- **Consulta:** Se necesita hacer uso de las palabras ordenador y consulta para activar esta función y acto seguido hacer nuestra petición. Ejemplo: *Ordenador tengo una consulta, ¿cual es la capital de España?*
- **Descarga:** Tenemos que volver a usar la palabra ordenador y la palabra descargar o descarga. A continuación nos pedirá el video a descargar. Las opciones son “video 1” o “video 2” podemos modificar la URL de los videos que queremos descargar a través del código del programa. Por defecto hay 2 URL predeterminadas.

Bibliografía

- [1] Rehkopf, D. M. Todo lo que necesitas saber sobre los sprints de scrum.
<https://www.atlassian.com/es/agile/scrum/sprints>.
- [2] Python.org. Welcome to python.org.
<https://www.python.org/doc/>.
- [3] OpenAI. Robust speech recognition via large-scale weak supervision. Technical report, Introducing Whisper, 2022.
<https://cdn.openai.com/papers/whisper.pdf>.
- [4] IBM. ¿qué es deep learning?
<https://www.ibm.com/es-es/topics/deep-learning>.
- [5] GeeksforGeeks. Running python script on gpu.
<https://www.geeksforgeeks.org/running-python-script-on-gpu/>.
- [6] OpenAI. Github - openai/whisper: Robust speech recognition via large-scale weak supervision.
<https://github.com/openai/whisper>.
- [7] Eleven Labs.
<https://docs.elevenlabs.io/welcome/introduction>.
- [8] ElevenLabs. Elevenlabs — prime voice ai.
<https://beta.elevenlabs.io/speech-synthesis>.
- [9] Speechrecognition.
<https://pypi.org/project/SpeechRecognition/>.

- [10] pytube 15.0.0 documentation.

<https://pytube.io/en/latest/>.

- [11] OpenAI. Introducing chatgpt.

<https://openai.com/blog/chatgpt>.

- [12] ATRIA Innovation. ¿cómo funciona chat gpt?

<https://www.atriainnovation.com/como-funciona-chat-gpt/>.

- [13] R. Merritt. Modelo transformer.

[https://la.blogs.nvidia.com/2022/04/19/
que-es-un-modelo-transformer/](https://la.blogs.nvidia.com/2022/04/19/que-es-un-modelo-transformer/).

- [14] PromptIA. Configuración de parametros en chat gpt.

<https://promptia.io/configuracion-de-chatgpt/>.

- [15] Pricing.

<https://openai.com/pricing>.