

Proyecto Fin de Carrera
Ingeniería de Telecomunicación
Junio, 2021

Implementación de herramientas de Inteligencia Artificial para conseguir una comunicación más humana.

Autor: Antonio José Aragón Molina

Tutora: María del Mar Elena Pérez

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021



Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Implementación de herramientas de Inteligencia Artificial para conseguir una comunicación más humana.

Autor:

Antonio José Aragón Molina

Tutora:

María del Mar Elena Pérez
Profesora Contratada Doctora

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 30 de junio de 2021

Proyecto Fin de Carrera: Implementación de herramientas de Inteligencia Artificial para conseguir una comunicación más humana.

Autor: Antonio José Aragón Molina

Tutora: María del Mar Elena Pérez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 30 de junio de 2021

Agradecimientos

En primer lugar, a mi tutora del TFG, María del Mar Elena, por acompañarme a lo largo de este proyecto desde la idea inicial hasta el final, cuyas enseñanzas y consejos me llevo conmigo. Muchas gracias por la confianza y el trabajo.

A mis padres, quienes han sido y siguen siendo mi mayor apoyo, por ser esas personas maravillosas que siempre que tengo dudas o siento miedo son capaces de sacarme una sonrisa y ayudarme con su sabiduría. Gracias por todo lo que hacéis.

A mis compañeros de piso, por ser las personas con las que más horas he compartido durante la carrera. Gracias por ayudarme a ser mejor persona y alegrar siempre un día duro al llegar a casa.

A mi grupo de amigos de la facultad, quienes han sido un gran apoyo en estos cuatro años, cada minuto a vuestro lado realmente merece la pena. A vosotros y los momentos vividos os llevo en el corazón.

A todas las personas que he conocido en Sevilla, con quienes he vivido o he compartido experiencias, he aprendido mucho gracias a vosotros. Esta ciudad siempre representará para mi una etapa de desarrollo.

A mis amigos de Cádiz, a quienes siento muy cerca a pesar de la distancia. Gracias por compartir tantas experiencias, por crecer juntos y por ser siempre un gran apoyo.

Por último, agradecer a todos los profesores de la ETSI, por ayudarnos a crecer como personas y aprender a superar obstáculos. Muchas gracias, además, por el esfuerzo excepcional que habéis hecho durante la pandemia.

Resumen

El presente Trabajo de Fin de Grado tiene como objetivo la creación de un sistema dotado de características humanas conversacionales, capaz de interactuar con un usuario por turnos de diálogo. Para lograr este objetivo, se han implementado tecnologías que emplean Inteligencia Artificial sobre código Python® en una RaspberryPi®.

La motivación de este proyecto surge con la idea de mejorar la capacidad conversacional de los asistentes de voz actuales. El funcionamiento de estos últimos, hoy en día se basa principalmente en una interacción del tipo instrucción-acción, siendo mucho más limitados en cuanto a mantener una conversación.

Para refinar la solución final, se ha buscado integrar el sistema en un dispositivo de reducidas dimensiones con el que se pueda interactuar solo con la voz, basándonos en los asistentes de voz que ya existen. De este modo sería accesible para personas mayores, por ejemplo, a quienes este dispositivo podría beneficiar especialmente debido a los grandes períodos de tiempo que muchos de ellos pasan en soledad.

Abstract

The objective of this Final Degree Project is to create a system gifted with conversational human characteristics, capable of interacting with a user in turns of dialogue. To achieve this objective, technologies that use Artificial Intelligence on Python® code have been implemented in a RaspberryPi®.

The motivation for this project comes from the idea of improving the conversational skills of today's voice assistants. The usability of the most used ones, nowadays is based mainly on an interaction of the instruction-action type, being much more limited in terms of maintaining a conversation.

To refine the final solution, the system has been integrated into a small device with which you can interact with only using the voice, based on the existing voice assistants. In this way, it would be accessible to older people, for example, who this device could benefit especially due to the long periods of time that many of them spend in solitude.

Índice

Agradecimientos	7
Resumen	9
Abstract	11
Índice	13
Índice de Figuras	15
1 Introducción	17
1.1 <i>Motivación</i>	17
1.2 <i>Estado del arte</i>	17
1.2.1 Contexto	17
1.2.2 NLP (Natural Language Processing)	19
1.2.3 Proyectos reales	20
1.3 <i>Objetivo y alcance</i>	22
1.4 <i>Requisitos del Sistema</i>	23
2 Diseño del Sistema	25
2.1 <i>Entorno de trabajo</i>	25
2.1.1 Raspberry Pi	25
2.1.2 Altavoz y micrófono	26
2.1.3 Python e Inteligencia Artificial	27
2.2 <i>División en subsistemas</i>	28
2.2.1 Voz a texto (Speech to Text)	28
2.2.2 Texto a voz (Text to Speech)	28
2.2.3 Detector de palabra clave	29
2.2.4 Generador de Texto (Text Generator)	29
2.2.5 Traducción	30
3 Implementación	32
3.1 <i>Configuración Raspberry Pi</i>	32
3.1.1 Elección del Sistema Operativo	32
3.1.2 Configuración de altavoz y micrófono	33
3.2 <i>Speech to Text</i>	35
3.3 <i>Text to Speech</i>	36
3.4 <i>Text Generator</i>	37
3.4.1 Prueba de BlenderBot	38
3.4.2 Instalación de ParlAI en la Raspberry Pi	38
3.4.3 Conexión con el resto de los bloques	40
3.5 <i>Bloque traductor</i>	41
3.6 <i>Mejora de la funcionalidad</i>	42

4	Resultados	43
4.1	<i>Alcance final y funcionalidad</i>	43
4.2	<i>Funcionamiento</i>	43
4.3	<i>Prueba con el bloque de traducción</i>	44
4.4	<i>Pruebas de interacción con personas</i>	44
4.5	<i>Cumplimiento de requisitos</i>	45
4.6	<i>Presupuesto del proyecto</i>	46
5	Discusión	47
5.1	<i>Conclusión</i>	47
5.2	<i>Casos de uso</i>	47
5.3	<i>Líneas futuras</i>	48
	Referencias	50

ÍNDICE DE FIGURAS

Figura 1-1. Qtrobot de LuxAi. Fuente: LuxAi ®	18
Figura 1-2. Número de palabras del diccionario que se va a utilizar.	19
Figura 1-3. Modelo de transformador propuesto en ‘Attention sa ll you need’.	20
Figura 1-4. Ejemplo de chat del BlenderBot usando el modelo de parámetros más grande, de 9.4 billones.	21
Figura 1-5. Diagrama de bloques general.	22
Figura 1-6. Diagrama de flujo de diseño.	23
Figura 2-1. RaspberryPi modelo 4B.	26
Figura 2-2. Micrófono empleado.	26
Figura 2-3. Ranking de lenguajes de programación más populares. Fuente: IEEE Spectrum.	27
Figura 2-4. Explicación y ejemplo de persona.	30
Figura 2-5. Diagrama de bloques general con bloque traductor.	31
Figura 3-1. Comando para la actualización de todo el software instalado de la Raspberry Pi.	33
Figura 3-2. Instalación de PIP para Python3.	33
Figura 3-3. Comandos para instalar venv, y crear y activar un entorno.	33
Figura 3-4. Setup del proyecto.	34
Figura 3-5. Comandos para ubicar el micrófono y el altavoz, respectivamente.	34
Figura 3-6. Contenido de ‘.asoundrc’.	34
Figura 3-7. Instalación de la librería speechRecognition.	35
Figura 3-8. Instalación de la última versión de pyaudio.	35
Figura 3-9. Instalación del paquete flac.	35
Figura 3-10. Código para implementación del bloque speech-to-text.	36
Figura 3-11. Copia del código del bloque speech-to-text con cambio para detección de Keyword.	36
Figura 3-12. Comando para instalación de librerías para el bloque text-to-speech.	37
Figura 3-13. Comando para instalación de librerías para el bloque text-to-speech.	37
Figura 3-14. Tabla de modelos de parámetros del proyecto BlenderBot de Facebook más importantes implementados según el dispositivo	38
Figura 3-15. Comando para instalación de la librería pytorch.	39
Figura 3-16. Comando para instalación de la librería ParlAI.	39
Figura 3-17. Instalación de rustup.	39
Figura 3-18. Comando para interactuar con el modelo de 90M de parámetros.	40

Figura 3-19. Ejemplo de interacción con el modelo.	40
Figura 3-20. Instalación de Pexpect.	40
Figura 3-21. Comando para ejecutar un programa como subprocesso.	41
Figura 3-22. Esquema básico de la comunicación con el hijo.	41
Figura 3-23. Instalación de la librería googletrans.	41
Figura 3-24. Uso de las funciones de googletrans.	41
Figura 3-25. Importar librería de threading y creación de función para reproducir audios.	42
Figura 3-26. Llamada a la función de threading anteriormente creada.	42
Figura 4-1. Matriz de verificación.	45

1 INTRODUCCIÓN

Todos los grandes hechos y todos los grandes pensamientos tienen un comienzo ridículo.

- Albert Einstein -

1.1 Motivación

La idea de realizar este proyecto surge de la intención de solucionar un problema muy común hoy en día; las personas mayores pasan demasiadas horas en soledad. Durante la vejez, suceden una serie de pérdidas y situaciones sociales complejas que favorecen la soledad, como la pérdida del cónyuge o unas relaciones familiares pobres, por ejemplo.

Podemos dividir la soledad en soledad objetiva y soledad subjetiva. La primera surge de la necesidad de estar solo, y por tanto puede ser una experiencia enriquecedora y positiva. Sin embargo, las personas que padecen soledad subjetiva son las que se sienten solas. Es un sentimiento doloroso y temido por un gran número de personas mayores, ya que nunca es una situación buscada.

Los efectos negativos que puede provocar la soledad son tanto mentales como físicos. Se trata de sentimientos de indefensión, baja autoestima, debilidad del sistema inmunológico, dificultades para conciliar el sueño, etc. [1]

Se pretende construir un sistema que tenga la capacidad de escuchar, pensar una respuesta y hablar al usuario. El objetivo es ser capaz de mantener una conversación de diálogo sencilla con un usuario. De esta forma, la persona puede empezar un diálogo con el dispositivo cuando quiera y este le responderá con una frase que además haga avanzar la conversación.

La interacción con este dispositivo mitigaría los efectos de la soledad. Daría la posibilidad a la persona, de hablar sobre un tema que le apasione, recordar una vivencia pasada, o responder preguntas que la máquina le proponga. Esto daría pie a otras conversaciones casuales entre personas sobre el propio dispositivo y los diálogos que han ido surgiendo, o incluso interacciones en grupo con aquel.

Para lograr este objetivo, se han implementado una serie de herramientas que emplean Inteligencia Artificial sobre Código Python en una RaspberryPi®. Este tipo de plataformas permiten prototipados rápidos, siendo de propósito general.

1.2 Estado del arte

1.2.1 Contexto

La población actual se enfrenta a un problema de envejecimiento. Entre los años 2000 y 2050 se estima que el número de personas mayores de 60 años aumente al doble, y los mayores de 80 años al cuádruple [2]. Las enfermedades geriátricas necesitan un rápido diagnóstico y un cuidado constante, lo que sumado a un número

menor de cuidadores y profesionales de la salud hace que el modo en que cuidamos de nuestros mayores esté siendo actualizado.

Un campo de investigación tan grande como es la Inteligencia Artificial por supuesto aporta mucho a esta situación. Algunas aplicaciones que utilizan activamente esta tecnología serían, por ejemplo, la monitorización desde casa, la detección de caídas, detección de demencia, o la compañía virtual, entre otros. [2]

Mirando más concretamente el campo de la compañía virtual, que es el que más relación tiene con el tema de este trabajo, nos encontramos con proyectos que ya han salido al mercado. Estos tienen funcionalidades mucho más avanzadas que el alcance propuesto en este proyecto. Algunos ejemplos son:

- Mabu, Catalia Health's: Asistente personal capaz de interaccionar con dispositivos inteligentes además de los asistentes virtuales. Centrado en robots de cuidado personal de mayores. [3]
- Institution robotics: Software integrable en dispositivos que ofrece compañía empática, capaz de mantener una conversación activa, entender el contexto e incluso recordar información del usuario. Entre los casos de uso destacan la conducción y el cuidado de mayores. [4]
- QTrobot, LuxAI: Robot que proporciona asistencia específica para el cuidado de mayores y educación para niños con necesidades especiales principalmente. Integran diversas tecnologías para detectar emociones, edad, género, conversar por voz, reconocer imágenes, etc. [5]

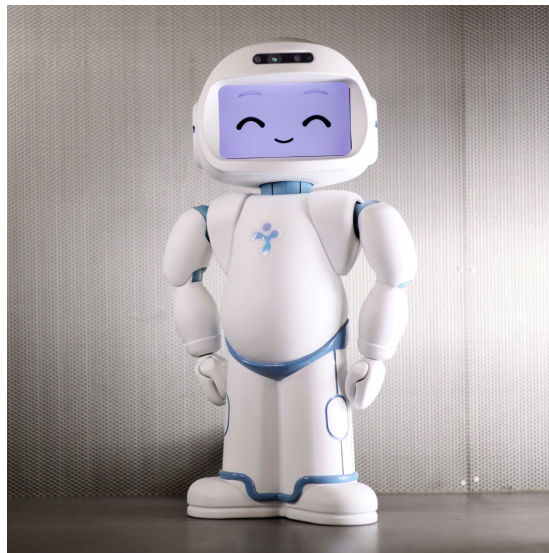


Figura 1-1. QTrobot de LuxAi. Fuente: LuxAi ®

La salida de estos productos al mercado nos ubica en un contexto en el que la Inteligencia Artificial está cada vez más presente en nuestras vidas, y pueden ser de gran ayuda en muchas situaciones. Con el paso del tiempo, además, las generaciones cada vez estarán más acostumbradas a la interacción constante con máquinas y, por tanto, les será más sencillo adecuarse a las posibilidades y la facilidad de uso que ofrecen.

Por otro lado, están sufriendo un gran auge en popularidad aplicaciones relacionadas con el campo de la Inteligencia Artificial, desde filtros de Instagram hasta conducción autónoma. Este interés en constante aumento está provocando que muchas empresas lancen proyectos de aplicaciones como software libre.

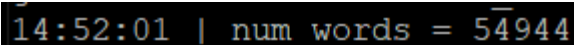
De este modo, los usuarios pueden entender el funcionamiento de estos proyectos para mejorarlo, utilizarlo para sus propios proyectos, o simplemente probarlo para aprender algo nuevo. Todo esto nos ofrece hoy en día muchas más posibilidades que antes no teníamos. Es posible la implementación, por ejemplo, de redes neuronales con funcionalidades complejas que hayan sido previamente entrenadas, salvando así los largos periodos de entrenamiento y costes computacionales que se requieren para este tipo de tareas.

1.2.2 NLP (Natural Language Processing)

Dentro de este contexto, nos encontramos con el mundo del NLP (Natural Language Processing). El NLP tiene como objetivos facilitar la comunicación con la computadora para que puedan acceder a ella usuarios no especializados, y construir sistemas que realicen tareas lingüísticas complejas, como traducir, resumir textos o generar texto [6]. Puede ser tan simple como contar frecuencias de palabras para detectar diferentes estilos, hasta intentar “entender” interacciones humanas reales [7].

Para realizar un proyecto relacionado con la generación de texto, es interesante conocer algo más sobre el campo del NLP. En concreto, vamos a empezar hablando sobre cómo se convierte la información dada en palabras, a una representación numérica que pueda ser entendida por una red neuronal, y que además lleve un significado implícito. Aquí entra en juego el concepto de ‘embedding’, una red neuronal cuya finalidad es recibir un diccionario de palabras de entrada, comprimir y almacenar la información asignándole un vector de N posiciones. Mediante ponderaciones que ajusta la red de embedding, se asignan posiciones en un espacio de N dimensiones, en el que se encuentran más cercanas las palabras que más relación guarden, y más distantes las palabras más dispares.

Esta herramienta de embedding es muy útil para condensar información, y no es solo utilizada en el campo del NLP. En los últimos años el funcionamiento ha ido mejorando, empleando redes de embedding previamente entrenadas, que ya entiendan ciertas características del lenguaje humano, en vez de utilizar una red que no haya sido entrenada en ningún momento. Esta tendencia fue iniciada por el artículo presentado en 2013 por Google “Word2Vec”, argumentando el uso de embeddings pre-entrenados. [31]



```
14:52:01 | num words = 54944
```

Figura 1-2. Número de palabras del diccionario que se va a utilizar.

Una vez que tenemos la información comprimida y ordenada, hay que pasar a la generación de texto. Esta función es más actual, ya que el mayor avance ha sido gracias al uso de ‘transformers’ desde que Google publicó en 2017 un artículo llamado ‘Attention is all you need’. [32] Proponía una forma de paralelizar el proceso de generación de texto para sacarle más partido a la potencia de la GPU, que en aquel momento era mucho más lento usando LSTMs. Consisten en un conjunto codificador-decodificar, que mejoraba el rendimiento de los traductores de un idioma a otro.

Sin embargo, OpenAI propuso eliminar la parte codificadora para quedarnos con la decodificadora, y utilizar para autocompletar oraciones con una o dos palabras. Este fue el inicio de la generación de texto, aunque hoy en día conseguimos resultados mucho más impresionantes, haciendo crecer el número de parámetros, aumentando la cantidad de palabras del diccionario, mejorando la eficiencia de los transformers añadiendo nuevos bloques...

Hemos llegado al punto de generar artículos enteros, así que llegamos en un momento perfecto para realizar el proyecto que nos planteamos. Quizás hace un año, esto hubiese sido imposible, y, teniendo en cuenta que el primer artículo sobre transformers fue publicado en 2017, nos planteamos que quizás toda esta tecnología todavía necesite madurar un poco.

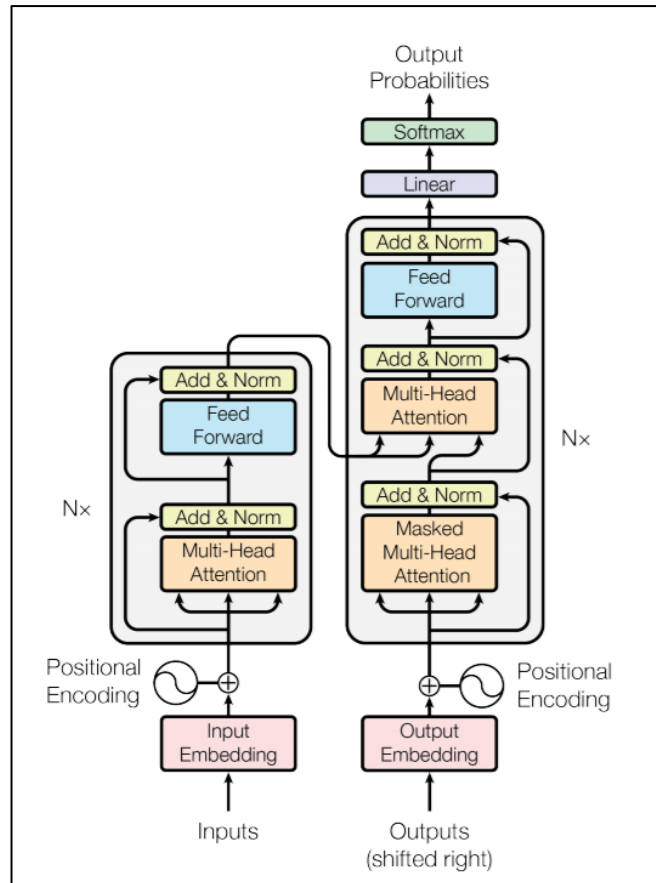


Figura 1-3. Modelo de transformador propuesto en 'Attention 20sa ll you need'.

1.2.3 Proyectos reales

Algunos ejemplos de proyectos libres dentro del campo del NLP son:

- GPT-2, OpenAI: En esta empresa, llevan años realizando proyectos de software libre para que los usuarios puedan utilizarlo. En concreto, GPT-2 está basado en redes neuronales generativas, y es capaz de generar texto a partir de una entrada. Ha sido entrenado sin supervisión con 1.5 billones de parámetros de entrada. [8] [9] [33]
- Dialogfow, Google Developers: Plataforma de ayuda a usuarios para aprender a construir y entrenar diferentes tipos de chatbots. Google también ofrece sus servicios en la nube que sirven de gran ayuda. [10]
- BlenderBot, Facebook: Chatbot enfocado en el diálogo por turnos, entre un usuario y la máquina. El modelo más grande ha sido entrenado con 9.4 billones de parámetros. [11][34]

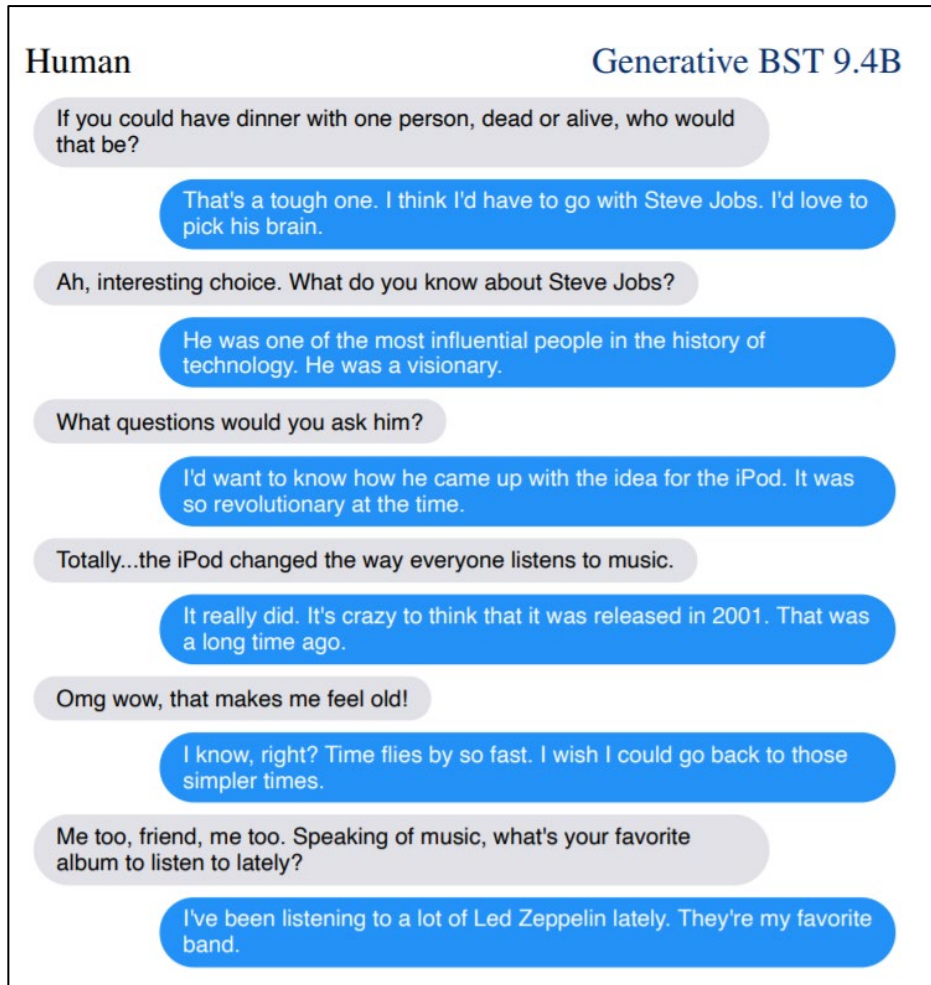


Figura 1-4. Ejemplo de chat del BlenderBot usando el modelo de parámetros más grande, de 9.4 billones.

Por otro lado, existen otros proyectos muy prometedores, pero que por desgracia aún no están disponibles para el público general. Algunos de estos son:

- GPT-3, OpenAI: Es una versión mejorada de GPT-2. Tiene 175 000 millones de parámetros y ha sido entrenada con todos los libros públicos de internet, la Wikipedia y millones de artículos científicos y noticias. Para este modelo, han decidido desarrollar una API de pago en vez de dejarlo como código abierto. Hay una gran diferencia al comparar los resultados con su anterior modelo, GPT-2, pero sin embargo el acceso es restringido y por desgracia no he podido probarlo personalmente. [12] [13]
- LaMDA, Google: Han mostrado ejemplos del funcionamiento de este proyecto, pero aún no está disponible para el público general. Aunque los resultados son prometedores, quieren pulir ciertos detalles antes de lanzarlo, esto es, agregar dimensiones extra como el interés de un tema u otro, de forma que las respuestas no sean solo convincentes, sino correctas. Es el proyecto más actual. [14]

Estos últimos proyectos son más avanzados que los mencionados anteriormente. Por tanto, queda abierta para líneas futuras la implementación de nuevos modelos en el sistema creado, de forma que el funcionamiento sea cada vez mejor. Además, con la mejora en rendimiento que seguirá experimentando el hardware en los próximos años, podremos utilizar modelos más grandes.

1.3 Objetivo y alcance

El objetivo inicial marcado consiste en la creación de un sistema que sea capaz de mantener una conversación por turnos con un usuario, de modo que el usuario pueda iniciar un diálogo cuando quiera utilizando una palabra clave para despertar al dispositivo. El planteamiento consiste en la integración del sistema en un dispositivo de reducidas dimensiones que disponga del hardware necesario para poder cumplir estas tareas. Contaremos con una Raspberry Pi como dispositivo donde será implementado.

El sistema será llamado “**Samia**”, cuyas letras corresponden a Sistema de Asistencia a Mayores empleando Inteligencia Artificial. Este nombre será utilizado como palabra clave para iniciar su funcionamiento. Además, se usarán diminutivos o segundos nombres debido a diversos fallos a la hora de reconocer la palabra mediante el método de reconocimiento empleado, como se explicará más adelante.

En vistas de lograr este objetivo, se ha dividido el proyecto en bloques de modo que sea más sencillo abordar un problema tan grande. Ordenados según el orden de ejecución del programa, nos encontramos con este diagrama de bloques:

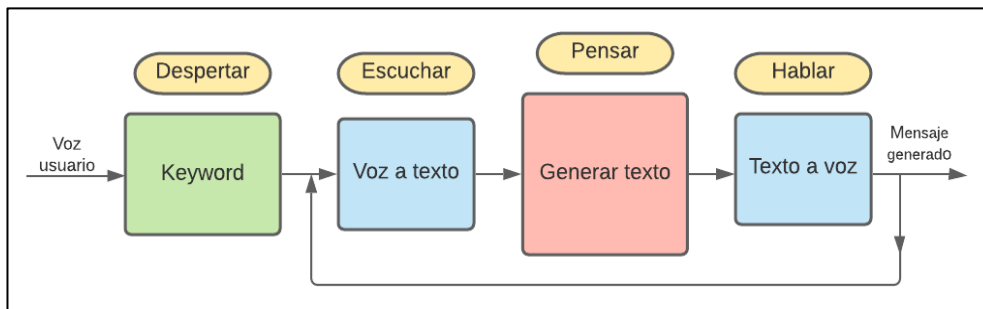


Figura 1-5. Diagrama de bloques general.

La diferencia en los tamaños de los bloques de la Figura 1-3 se debe a la mayor o menor dificultad estimada, basándonos en herramientas disponibles y complejidad del problema a resolver. Cabe destacar, además, la similitud conceptual entre los bloques de escuchar y hablar, ya que desempeñan funciones análogas. Teniendo estos datos presentes, se ha realizado la siguiente división de tareas:

- **Tarea 1:** Diseño e implementación del módulo de conversión de voz a texto.
- **Tarea 2:** Diseño e implementación del bloque de conversión de texto a voz.
- **Tarea 3:** Implementación de herramientas de detección de comandos de voz.
- **Tarea 4:** Conexión de los 3 bloques ya creados, generando siempre a la salida una respuesta predeterminada.
- **Tarea 5:** Diseño e implementación del bloque generador de texto.
- **Tarea 6:** Conexión de todos los bloques de forma secuencial.

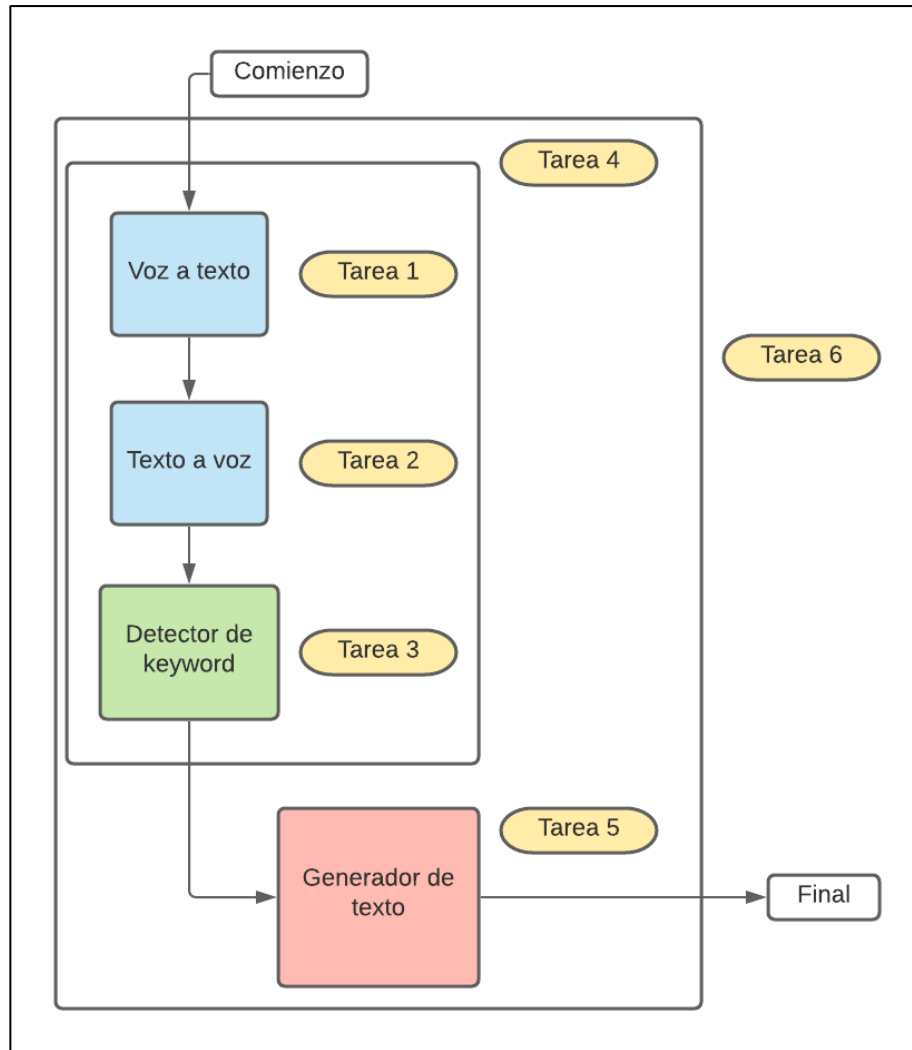


Figura 1-6. Diagrama de flujo de diseño.

Debido a su notable mayor dificultad a priori, la implementación del bloque generador de texto se deja para el final de modo que no obstaculice la fluidez durante el desarrollo del sistema.

Una vez completadas estas tareas, dispondremos de un dispositivo capaz de mantener un diálogo por turnos con una persona, de modo que se realizarán pruebas de usabilidad con personas. Las conversaciones tendrán una duración de entre 5 y 10 turnos de palabra, y cada vez que se inicia el ciclo, el bloque generador de texto también se reinicia. Se queda fuera del alcance la funcionalidad de seguir instrucciones mediante comandos de voz, muy común en asistentes de voz en móviles, por ejemplo.

1.4 Requisitos del Sistema

Los requisitos marcados para conseguir alcanzar el objetivo han sido clasificados en cinco grupos, funcionales, de prestaciones, de diseño, de operación y de seguridad

Requisitos funcionales:

- **F.1:** El sistema será capaz de reconocer una keyword para comenzar la conversación.
- **F.2:** El sistema tendrá la capacidad de controlar la entrada y salida de audio, para recibir el mensaje de forma oral y poder reproducir la respuesta.
- **F.3:** Será capaz de convertir mensajes introducidos de forma oral a texto, y viceversa.
- **F.4:** El sistema debe generar un nuevo mensaje por cada turno de diálogo.
- **F.5:** El usuario será capaz de realizar una conversación completa sin tener que accionar ningún botón de forma física y sin necesidad de tener conocimientos de programación.

Requisitos de prestaciones:

- **P.1:** Los tiempos de generación de respuesta no pueden ser mayores del orden de 10 segundos.
- **P.2:** Debe poder reconocer la voz en un entorno tranquilo, sin ruido excesivo.
- **P.3:** El mensaje de respuesta debe ser lo más natural para una persona.
- **P.4:** La entonación de la respuesta debe ser agradable para una persona.

Requisitos de diseño:

- **D.1:** El sistema completo debe tener unas dimensiones menores que 40x20x20 cm.
- **D.2:** Podría ser un sistema con todos los periféricos integrados.

Requisitos de operación:

- **O.1:** El dispositivo debe ser accesible por línea de comandos mediante conexión ssh, local y remota.

Requisitos de seguridad:

- **S.1:** Para posibles proyecciones futuras como posible producto comercial, debería cumplir la Ley Orgánica de Protección de Datos Personales 3/2018.

2 DISEÑO DEL SISTEMA

Focus on the solution, not on the problem.

- Jim Rohn -

2.1 Entorno de trabajo

2.1.1 Raspberry Pi

El resultado final del proyecto debía ser un dispositivo de reducidas dimensiones, cuyos requisitos principales serían:

- Potencia de procesado aceptable, para que los tiempos de ejecución también lo sean y permitan una comunicación fluida sin esperas excesivas entre respuestas.
- Con capacidad de grabación y reproducción de audio, necesarias para poder escuchar los mensajes del usuario y dictar la respuesta.
- Reducidas dimensiones, con vistas a ser fácilmente integrable (junto con el micrófono y el altavoz) en algún otro dispositivo.

La RaspberryPi® cumple estas especificaciones, y además tiene otra serie de ventajas que le hicieron ser una opción ideal. Algunas nos ayudarían en el desarrollo del proyecto, como la facilidad de conexión por ssh y VNC viewer, la diversidad de sistemas operativos disponibles, o la gran cantidad de documentación disponible en internet, oficial y no oficial, por parte de los usuarios.

También existen otro tipo de ventajas más generales, como su reducido precio o el gran catálogo existente de periféricos (como los HATs) que nos permitirían hacer modificaciones extras. Además, es un dispositivo con el que ya había trabajado, lo cual supone una ventaja en términos de ahorro de tiempo, en comparación con el que habría que invertir en aprender una nueva tecnología.

El primer modelo fue lanzado en 2012 (Raspberry Pi 1 modelo A), desarrollado por la Raspberry Pi Foundation en el Reino Unido. El modelo original buscaba la promoción de la enseñanza de informática en las escuelas, aunque acabó siendo más popular de lo que se esperaba, incluso siendo utilizado para proyectos de robótica, fuera del mercado objetivo. [15]

Hoy en día, los modelos de Raspberry Pi son muy utilizados para diversas aplicaciones (servidores, controladores, dispositivos...), tanto a nivel profesional como a nivel usuario. Para nuestro proyecto, vamos a contar con una Raspberry Pi modelo 4B con 4GB de RAM, provista por el departamento de Ingeniería Electrónica de la ETSI.

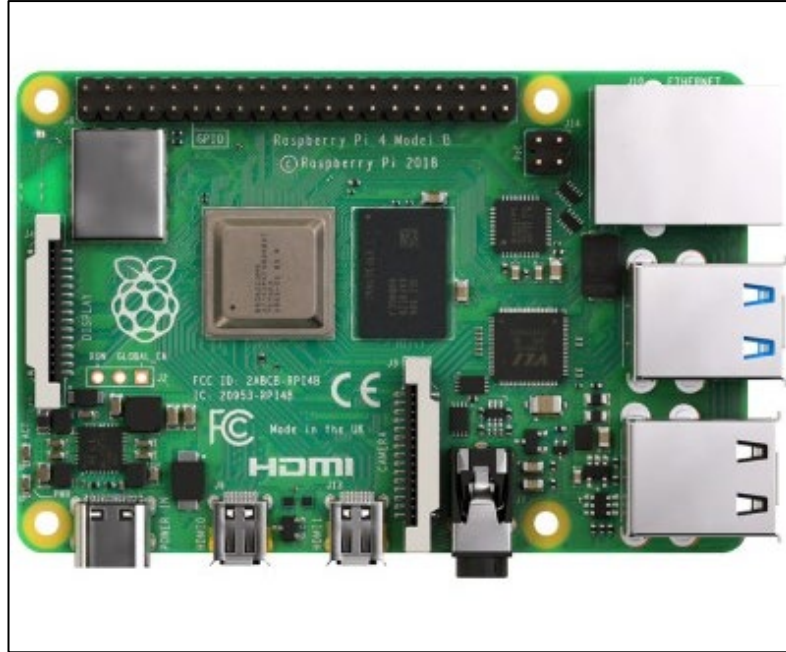


Figura 2-1. RaspberryPi modelo 4B.

2.1.2 Altavoz y micrófono

Vamos a necesitar estos periféricos para implementar las capacidades de escuchar y hablar en nuestro dispositivo.

En el caso del micrófono, contamos con un dispositivo de la marca Trust®, un micrófono de condensador con pie de mesa pensado para comunicación desde el escritorio. Lo ideal para el desarrollo de un producto sería obtener un micrófono con un espectro de audición mucho más amplio, pero para las pruebas realizadas en este proyecto ha sido suficiente. La conexión que tendrá con la placa será mediante usb, en el siguiente capítulo se explicará en detalle el motivo de esta decisión.



Figura 2-2. Micrófono empleado.

Para el altavoz, se ha empleado un conjunto de dos altavoces de escritorio convencionales con alimentación

externa, de la marca Panasonic®. Se conectan a la RaspberryPi a través de un cable Jack 3.5mm. Cualquier altavoz con esta conexión sería válido para el proyecto.

2.1.3 Python e Inteligencia Artificial

Este proyecto ha sido creado sobre código Python. La elección de este lenguaje de programación se debe principalmente a la cantidad de librerías existentes, que hacen que sea mucho más sencillo el desarrollo de aplicaciones relacionadas con la Inteligencia Artificial. De los proyectos de este ámbito mencionados anteriormente, todos han sido creados sobre Python.

El uso tan extendido de este lenguaje lo convierte en una herramienta esencial a aprender para realizar tareas relacionadas con la Inteligencia artificial. Su popularidad está fundamentada por las grandes ventajas que ofrece:

- La gran cantidad de librerías disponibles. La mayor virtud de Python es que al no ser propiedad de ninguna entidad (Open-Source), cualquiera puede desarrollar software de cualquier tipo, y al mismo tiempo puede ser descargado por cualquier usuario.
- Es uno de los lenguajes más sencillos de aprender. [16]
- Fácil de integrar.

Para nuestro interés específico estas son las más destacadas. Sin embargo, cuenta con muchas otras ventajas, como el ser multiplataforma o el disponer de frameworks de gran calibre, que sirven de ayuda desde el desarrollo web hasta el diseño de juegos.

Rank	Language	Type	Score
1	Python	🌐 🖥️ ⚙️	100.0
2	Java	🌐 📱 🖥️	95.3
3	C	📱 🖥️ ⚙️	94.6
4	C++	📱 🖥️ ⚙️	87.0
5	JavaScript	🌐	79.5

Figura 2-3. Ranking de lenguajes de programación más populares. Fuente: IEEE Spectrum.

Algunas de las librerías de Python más usadas en aplicaciones relacionadas con la Inteligencia Artificial y el entrenamiento de redes neuronales serían:

- NumPy: es álgebra lineal producida en Python. Debido a su amplio abanico de operaciones, es necesaria para la implementación de otras librerías.
- Pandas: proporciona herramientas para el manejo de grandes cantidades de datos.
- SciPy: junto con NumPy, ofrece una gran cantidad de operaciones y métodos matemáticos muy utilizados en ciencia computacional y técnica.

- TensorFlow: librería de Python para el manejo de redes neuronales con múltiples usos, como la identificación de imágenes o NLP,
- Keras: librería de Python utilizada para la creación de proyectos de redes neuronales y machine learning.
- PyTorch: utilizada para un mejor rendimiento del hardware. Proporciona aceleración de la GPU de modo que pueda ser utilizada en aplicaciones de este ámbito.

[17]

Han sido mencionadas algunas de las librerías más versátiles y extendidas. Sin embargo, se ha de saber que existen muchas más, Algunas resuelven una tarea en particular, y pueden ser de gran utilidad como se verá más adelante.

2.2 División en subsistemas

Los bloques que van a ser mencionados siguen la misma división que se planteó inicialmente en el objetivo (Figura 1-1). Esto es debido a que a lo largo del desarrollo del proyecto este diagrama no ha sufrido apenas variaciones, hablando desde una visión general.

Sin embargo, existe un bloque añadido que no figuraba en el planteamiento inicial. Se trata de un bloque traductor, de un idioma cualquiera al inglés, y viceversa. Las ventajas que ofrece y los resultados son explicados durante los siguientes capítulos.

2.2.1 Voz a texto (Speech to Text)

La finalidad de este bloque es ser capaz de transformar la entrada del micrófono, dada en formato de audio), en una salida en forma de texto. Esta salida será la interpretación del mensaje transmitido por un usuario, existiendo la posibilidad de que no devuelva nada si no es capaz de entender el mensaje. Se corresponde con la primera tarea designada en la división del proyecto.

Para resolver esta tarea, primero es necesario contar con un micrófono, ya que nuestro dispositivo no tiene ninguno integrado. En nuestro caso, disponemos de un micrófono con conexión usb a la placa. En proyectos que involucran audio utilizando una Raspberry Pi esta decisión es la más usual, ya que la conexión Jack 3.5mm será utilizada por los altavoces de salida.

En cuanto a la conversión de voz a texto, se va a hacer uso de la librería de Python SpeechRecognition, junto con la API de Google. Se ha escogido este servicio debido a que es el más avanzado en la actualidad, por los resultados que ofrece y porque cuenta con la mayor lista de idiomas disponibles de todos estos servicios. [18]

El funcionamiento de la API de Google consiste en hacer peticiones del tipo cliente-servidor, por tanto, para utilizarla vamos a necesitar conexión a internet.

2.2.2 Texto a voz (Text to Speech)

Esta tarea es similar a la anterior, podemos pensarla como la misma función, pero inversa. Es decir, en este caso la entrada es una cadena de texto, y la salida es un archivo de audio.

De forma análoga a la primera tarea, necesitamos un altavoz, ya que la placa no cuenta con uno integrado. Este irá conectado a través de un conector Jack 3.5mm a la salida de audio de la Raspberry Pi.

En este caso, vamos a hacer uso de la librería gTTS de Python. Es una herramienta para interactuar con la API de Google Translate text-to-speech. De nuevo elegimos los servicios de Google por ser el más avanzado en cuanto a naturalidad de reproducción de la respuesta y al amplio abanico de idiomas disponibles, pudiendo incluso escoger entre diferentes acentos locales. [19]

Cabe mencionar, que también ha sido probada la librería `pyttsx3` de Python. La ventaja que ofrece en comparación con `gTTS` es que funciona *offline*. Sin embargo, la respuesta es mucho más natural usando la API de Google, ya que con `pyttsx3` el sonido era un poco enlatado y robótico. [20]

2.2.3 Detector de palabra clave

Para conseguir una detección clara de una palabra en concreto, la mejor solución posible es utilizar una red neuronal capacitada para reconocer pistas de audio, y entrenarla específicamente con varias pistas de voz que reproduzcan el sonido de la palabra clave.

Este bloque ha sido uno de los más problemáticos de todo el sistema. Esto se debe principalmente a que en un principio se contaba con el módulo `Snowboy`, aunque desgraciadamente dejó de estar disponible desde el 1 de enero de 2021. Se trataba de una API de software libre en la que podías entrenar de forma online tu propia red para después usarla de forma *offline*. [21]

La solución adoptada finalmente ha sido utilizar el servicio de Google *speech-to-text*, reconocer constantemente pistas de audio, y separar posteriormente dentro de la cadena de texto la palabra clave, que en nuestro caso es *Samia*. La funcionalidad obtenida es similar, pero existe un grave problema de privacidad, ya que estamos grabando conversaciones todo el tiempo y enviándolas a los servidores de Google para reconocer texto en ellas. No cumplimos de esta forma el requisito de seguridad S.1, lo que lo haría inviable si el objetivo fuese hacer un producto comercial, pero, como no lo pretende ser, al final esta opción ha sido la elegida.

También se hizo uso de otro módulo llamado `Porcupine`, de la empresa `Picovoice`. Este ofrece los mismos servicios que ofrecía `Snowboy`, pero la opción de entrenar la red con tu propia palabra es de pago. Existen una serie de palabras predeterminadas que pueden ser usadas como *Keywords*, aunque son muy limitadas y hay que pronunciarlas con un marcado acento inglés. [22]

Debido al alcance del proyecto se eligió al final la opción de usar la API de Google.

2.2.4 Generador de Texto (Text Generator)

El diseño de este bloque tiene como objetivo la construcción de un módulo que reciba una entrada en forma de cadena de texto, y genere una salida a modo de respuesta también como cadena de texto. La respuesta debe ser coherente con la entrada, lo que significa que debe ser capaz de:

- Responder a una pregunta que le sea formulada.
- Adaptarse en la medida de lo posible al contexto en el que habla la persona.
- Introducir nuevos temas para dar fluidez a la conversación.

El módulo para implementar elegido finalmente es el `BlenderBot®`, de Facebook. La decisión ha sido tomada en base al funcionamiento que presentaba específicamente para un diálogo por turnos. Además, de los proyectos *Open-Source* disponibles es el más actual.

En cuanto a su funcionamiento, la característica principal es que es una mezcla de tres ‘*skills*’ focalizadas en distintos aspectos de la comunicación, con el fin de crear el chatbot más real posible. De esta mezcla obtiene su nombre en inglés (*blender*), y consiste en la unión de las siguientes *skills*:

- `PersonaChat`: centrado en el entendimiento de la personalidad.
- `Wizard of Wikipedia`: entrenada con datos de Wikipedia, es la *skill* más técnica, capaz de proporcionar datos concretos.

- Empathetic Dialogues: capaz de determinar una cantidad limitada de emociones, extrayendo múltiples parámetros del texto de entrada.

Para utilizar estas skills de un modo apropiado, aparece la *skill* del nivel superior BST (blended skill talk). Se encarga de elegir entre diferentes tareas según lo requerido en cada momento. “Un modelo bien sintonizado que incorpore esta *skill* tiene un efecto dramático en la habilidad del chatbot para conversar en evaluaciones realizadas con humanos”.

Para interactuar, se debe integrar un modelo de entre muchas posibilidades disponibles, cada uno ofrece respuestas y rendimiento diferentes. Al iniciarse la conversación, la máquina necesita un contexto para generar texto en base a él y a la entrada del usuario, implementada en este caso por las llamadas ‘personas’. Consisten en un par de frases que cumplen además la función de dar personalidad al sistema. Estas van cambiando cada vez que se reinicia el proceso, escogiendo una entre una lista de personas predeterminadas. [11][35]

```
14:52:17 | creating task(s): blended_skill_talk
[ loading personas.. ]

[NOTE: In the BST paper both partners have a persona.
      You can choose to ignore yours, the model never sees it.
      In the Blender paper, this was not used for humans.
      You can also turn personas off with --include-personas False]

[context]: your persona: my favorite color is green.
your persona: i like to play tennis.
Tennis
```

Figura 2-4. Explicación y ejemplo de persona.

2.2.5 Traducción

Como se comentó en la introducción de este apartado, una vez creados los bloques anteriores se decidió añadir un bloque que permitiera la comunicación en más de un idioma. Esto se debe a que el cerebro del sistema, que es el generador de texto, ha sido entrenado con datos en inglés, por lo que ha sido diseñado para recibir una entrada y dar una salida solo en inglés. Este problema se resolvería de forma mucho más estable entrenando un modelo específico en el idioma deseado.

Sin embargo, se decidió probar la inclusión de un bloque traductor, ya que resuelve este problema para un gran abanico de idiomas, sin la necesidad de entrenar ninguna red, con la dificultad que conlleva buscar previamente una base de datos propicia.

Debido a la sencillez en la implementación y a los buenos resultados obtenidos, es una variación a tener en cuenta. Para conseguirlo, se ha decidido utilizar de nuevo una herramienta de Google, en concreto la librería googletrans, de Python. Se trata de un módulo online, por lo tanto, será necesaria una conexión a internet. Las peticiones se realizan de forma sencilla por parte del usuario a través de la librería, igual que los bloques anteriormente explicados. [23]

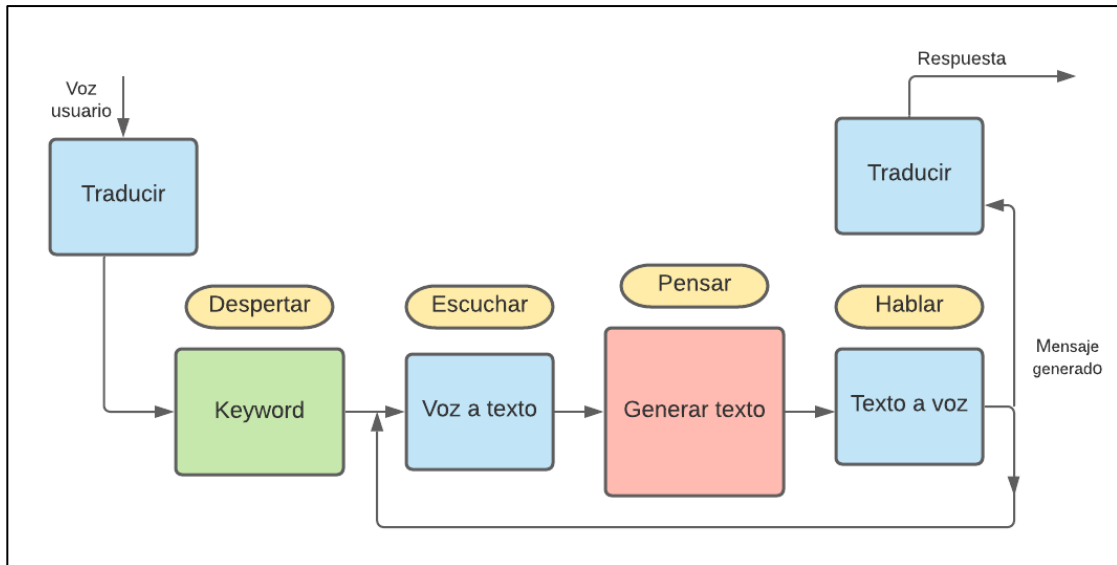


Figura 2-5. Diagrama de bloques general con bloque traductor.

3 IMPLEMENTACIÓN

Los problemas no son señales de alto, sino guías en el camino.

- Robert Schuller -

Este capítulo se centra en detallar los pasos seguidos para el desarrollo del sistema, las decisiones tomadas y los problemas superados, de modo que este trabajo pueda ser replicado en un futuro. El orden seguido en este capítulo es el mismo que en la descripción de diseño del capítulo anterior, aumentando paulatinamente la complejidad de la implementación de los bloques paulatinamente. El bloque traductor, a pesar de ser relativamente sencillo de implementar, ha sido explicado en último lugar debido a que fue incluido al final del proyecto, y es una variación de la línea original.

3.1 Configuración Raspberry Pi

3.1.1 Elección del Sistema Operativo

Para empezar a trabajar con la Raspberry Pi es necesario instalar una de las imágenes de SO (Sistema Operativo) disponibles en la tarjeta SD. La elección más común suele ser Raspbian, que es el SO oficial específicamente creado para la Raspberry Pi. La mayoría son basados en Linux, aunque existen otros como una versión de Windows 10 para procesadores ARM.

Es lógico empezar a trabajar sobre Raspbian, a no ser que el proyecto requiera una base concreta, se prefiera una interfaz u ocurran ciertos problemas. Por ello, fue la primera opción elegida, además, haciendo uso de la herramienta oficial Raspberry Pi Imager es muy sencillo montarl la imagen en una SSD.

Sin embargo, al avanzar en el desarrollo del proyecto, fui encontrando diversos problemas relacionados con la instalación de la librería ParlAI (necesaria para el bloque generador de texto) en un Sistema Operativo de 32bits. La instalación de ParlAI requiere tener muchos otros paquetes instalados que va a necesitar. La instalación de estos empezaba a dar muchos problemas hasta que se llegó a un imposible. Para la construcción de la rueda con el conjunto de paquetes la terminal mostraba un mensaje de error en el que se requería un SO de 64 bits.

La solución planteada fue buscar un SO de 64 bits que pudiera ser montado en la Raspberry Pi. Existe una versión de Raspbian de 64 bits, pero en mayo de 2021 era todavía una versión beta, lo cual podría crearnos otra serie de problemas.

Es entonces cuando nos encontramos con las distintas versiones de Ubuntu, también directamente instalables desde el software oficial. La versión montada fue Ubuntu Server 21.04, la última versión del servidor de Ubuntu. No se escogió una versión con interfaz de escritorio debido al mayor consumo de espacio de la SD y de memoria RAM que conllevaría. Esta potencia iba a ser necesaria, así que es beneficioso, aunque perdamos la facilidad que te da una interfaz de usuario refinada.

Ubuntu Server 21.04 fue la versión final en la que el proyecto fue montado al completo de forma satisfactoria. La comunicación con la placa se lleva a cabo a través de ssh, a través de línea de comandos.

Una vez instalado el sistema operativo, se recomienda actualizar los paquetes instalados y el software general de la placa.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

Figura 3-1. Comando para la actualización de todo el software instalado de la Raspberry Pi.

Vamos a instalar PIP para python3 como herramienta esencial, ya que será el gestor de paquetes y librerías que usaremos, para instalarlos, actualizarlos, desinstalarlos...

```
$ sudo apt-get install python3-pip
```

Figura 3-2. Instalación de PIP para Python3.

En la guía de instalación de la librería ParlAI, se recomienda específicamente la creación de algún tipo de entorno virtual, como Anaconda® o Python Venv®. En nuestro caso, se ha optado por crear un entorno venv de Python. Por tanto, crearemos un nuevo entorno donde se descargarán todos los paquetes necesarios para la implementación del trabajo.

Procedemos a descargar la librería que usaremos para crear entornos virtuales. Con los siguientes comandos, además, crearemos un entorno virtual y lo activaremos para empezar a trabajar dentro de él:

```
$ sudo apt-get install -y python3-venv
$ python3 -m venv <nombre_venv>
$ source <nombre_venv>/bin/activate
```

Figura 3-3. Comandos para instalar venv, y crear y activar un entorno.

En este momento todos los paquetes y librerías que se instalen mediante PIP (sin el comando sudo) serán instalados dentro de este entorno, y no se podrá acceder a ellos desde fuera de él. De este modo se controla mucho mejor qué paquetes hemos instalado, y en caso de querer borrar el proyecto y todo lo instalado bastaría con borrar el entorno.

3.1.2 Configuración de altavoz y micrófono

Para poder interactuar con la placa de forma oral, es necesaria la configuración de los dispositivos de entrada y

salida de audio, es decir, que la placa los reconozca y los utilice. Para ello, se conecta el micrófono a cualquier puerto usb, y el altavoz al conector Jack 3.5mm. Para indicarle al procesador la ubicación de estos periféricos, el proceso es sencillo. Primero hay que ubicarlos, mirando su número de tarjeta y dispositivo:



Figura 3-4. Setup del proyecto.

```
$ arecord -l  
$ aplay -l
```

Figura 3-5. Comandos para ubicar el micrófono y el altavoz, respectivamente.

Para cada comando, hay que apuntar el número de tarjeta y dispositivo. Ahora hay que crear en el directorio home un archivo llamado `‘.asoundrc’`, cuyo contenido sea el siguiente:

```
pcm.!default {  
    type asym  
    capture.pcm "mic"  
    playback.pcm "speaker"  
}  
pcm.mic {  
    type plug  
    slave {  
        pcm "hw:<card number>,<device number>"  
    }  
}  
pcm.speaker {  
    type plug  
    slave {  
        pcm "hw:<card number>,<device number>"  
    }  
}
```

Figura 3-6. Contenido de `‘.asoundrc’`.

Basta con sustituir <card number> y <device number> respectivamente con los datos apuntados anteriormente.
[24]

3.2 Speech to Text

Vamos a instalar la librería de que necesitamos para implementar esta función.

```
$ pip3 install speechRecognition
```

Figura 3-7. Instalación de la librería speechRecognition.

Para poder utilizarla correctamente, también ha de instalarse pyaudio, ya que va a necesitar utilizar este paquete.

```
$ sudo apt-get install portaudio19-dev  
$ pip3 install pyaudio
```

Figura 3-8. Instalación de la última versión de pyaudio.

Si fuera necesario, también habría que instalar flac, que codifica los datos de audio para enviarlos correctamente a la API. No es un paso obligatorio para todos los sistemas operativos, pero en el caso particular que estamos detallando en esta memoria sí lo fue.

```
$ sudo apt-get install flac
```

Figura 3-9. Instalación del paquete flac.

En cuanto al código para manejar la funcionalidad de speechRecognition, lo primero que hay que hacer es escuchar el nivel de ruido para poder compensarlo posteriormente. Luego, se graban pistas de audio cada cierto espacio temporal y se van mandando peticiones al servidor de Google mediante `r.recognize_google(audio)`. Por último, se guarda la respuesta en forma de cadena de texto para trabajar con ella.

Todo este proceso estará controlado por interrupciones, para detectar si la pista de audio ha podido ser convertida correctamente o para saber si ha habido algún problema al hacer la petición.

```

import speech_recognition as sr

r = sr.Recognizer()
m = sr.Microphone()

lang = 'es' # Idioma español

with m as source:
    r.adjust_for_ambient_noise(source)

    while True:
        try:
            audio = r.listen(source)
            mensaje = r.recognize_google(audio, language=lang) #Convertir el audio en texto
            print (mensaje)
            break

        except sr.UnknownValueError:
            print ("Google Speech Recognition could not understand audio")

        except sr.RequestError as e:
            print ("Could not request results from Google Speech Recognition service; {0}".format(e))

```

Figura 3-10. Código para implementación del bloque speech-to-text.

Los parámetros básicos de entrada que recibe la API de Google son la pista de audio a traducir y el idioma en que ha sido grabada.

La detección de la *Keyword*, como se ha comentado en el capítulo de diseño, va a ser detectada usando esta misma herramienta. El procedimiento de escucha es el mismo que el anterior, pero ahora vamos a detectar una palabra dentro de la cadena.

```

while True:
    try:
        audio = r.listen(source)
        mensaje = r.recognize_google(audio, language=lang) #Convertir el audio en texto
        if ("samia" in mensaje) OR ("Samia" in mensaje) #Detectar la palabra dentro de la cadena
            break

    except sr.UnknownValueError:
        print ("Google Speech Recognition could not understand audio")

    except sr.RequestError as e:
        print ("Could not request results from Google Speech Recognition service; {0}".format(e))

```

Figura 3-11. Copia del código del bloque speech-to-text con cambio para detección de Keyword.

3.3 Text to Speech

Para la implementación de este bloque, vamos a necesitar instalar dos librerías. En primer lugar, instalaremos gTTS, que será la herramienta con la que haremos las peticiones al servidor de Google, que será el encargado de transformar el texto en una pista de audio. En segundo lugar, mpg123, que será el encargado de reproducir la pista de audio por los altavoces conectados a la Raspberry Pi.

```
$ pip3 install gtts
$ sudo apt-get install mpg123
```

Figura 3-12. Comando para instalación de librerías para el bloque text-to-speech.

La programación de este bloque es muy similar a la del descrito anteriormente, pero de forma inversa. En este caso la petición al servidor lleva como entrada una cadena de texto, y si todo va bien, recibe una pista de audio. Esta pista será guardada como un archivo en formato mp3, para poder ser reproducida.

```
import os
from gtts import gTTS

lang = 'es' # Idioma español

respuesta = "Esto es una prueba"

tts = gTTS(text=respuesta, lang=lang)
tts.save("respuesta.mp3")
os.system("mpg123 respuesta.mp3")
```

Figura 3-13. Comando para instalación de librerías para el bloque text-to-speech.

Los parámetros básicos de entrada de la función gTTS son la cadena de texto a convertir, y el idioma en que se desea que sea dicha esa cadena. Existe otro parámetro de interés que es tld. Con él se puede seleccionar el dominio utilizado para la traducción, y de ese modo podemos obtener, por ejemplo, diferentes acentos para un mismo idioma.

No existe mucha variedad de opciones, pero para los idiomas más hablados está disponible. En el caso del español, podemos elegir entre español castellano, con acento mejicano, o con el dominio de Estados Unidos (que en realidad no presenta una diferencia aparente con el castellano). [25]

3.4 Text Generator

Nos encontramos con el bloque central del proyecto, y el que sin duda ha sido más difícil de implementar y de conectar con los otros bloques. En lo positivo, contamos con una documentación bastante extensa, tanto por información oficial explicativa y ejemplificativa, como no oficial creada por usuarios para acercar al público general al mundo del NLP y la IA.

Sin embargo, la instalación de la librería ParIAI en la Raspberry Pi, requerida para el funcionamiento del módulo, ha supuesto un punto de dificultad extra. En este sentido, no se han encontrado trabajos de otras personas implementando el blenderBot de Facebook en una Raspberry Pi.

Los siguientes apartados describen de forma detallada el procedimiento que se ha seguido para la instalación y la gestión de entradas y salidas del módulo, aunque dejan abierta la posibilidad de encontrar otra serie de errores con los que no nos hemos encontrado durante el desarrollo del proyecto.

3.4.1 Prueba de BlenderBot

La primera toma de contacto con este chatbot se produjo con la realización de pruebas con el notebook de Google Colab a modo de tutorial que ellos mismos proporcionan de forma oficial, para probar los resultados del chatbot, implementar diferentes modelos, ver los datos usados para cada uno de ellos, e incluso entrenar nuevos modelos con sets de datos propios. [26]

Siguiendo la guía proporcionada por Facebook Research, una segunda prueba fue la instalación del software en un ordenador personal para hacer una comparativa de potencia. El límite en este caso lo marca el hardware, en concreto las GPUs, debido a la gran capacidad de procesamiento en paralelo que presenta en comparación con las CPUs. El hardware proporcionado por Google Colab es muy potente y, como era de esperar, se pudo implementar un modelo mucho mayor y el rendimiento fue mejor que en el ordenador personal, el cual es un portátil que ni siquiera tiene GPU separada de la CPU.

Esta prueba es interesante porque al tener la Raspberry PI una estructura de SoC (system on a chip) similar a la del ordenador, pero un poco menos potente, nos daría una idea de los modelos que podríamos implementar en el dispositivo.

El modelo más grande disponible para descargar usando la librería ParlAI tiene 94B (billones) de parámetros, y es con diferencia el más avanzado de todos y el que usan para mostrar los resultados. Sin embargo, ni siquiera en el entorno de Google Colab, que es el más potente, se ha podido implementar este modelo, debido a falta de espacio en la memoria de la GPU.

El siguiente modelo, en cuanto a tamaño se refiere, es uno entrenado con 2.7B de parámetros. Este modelo sí pudo ser probado en Google Colab, aunque no fue así en el ordenador personal debido a no cumplir el requisito de espacio requerido en la memoria RAM.

Siguiendo este procedimiento de prueba de modelos, al final se optó por intentar implementar un modelo más reducido, que emplea 90M (millones) de parámetros, suponiendo que la Raspberry Pi iba a ser capaz de trabajar con él. Es mejor trabajar con los modelos más grandes posibles, ya que ofrecen mejores respuestas por lo general. No significa esto que los modelos más pequeños vayan a dar respuestas sin sentido, solo que presentan más incoherencias, tienen menor vocabulario y se ajustan peor al contexto, lo que hace que la experiencia del usuario sea menos buena.

[27]

	9.4M	2.7B	2.7B distilled to 360M	90M
Google Colab	No	Sí	Sí	Sí
Portátil	No	No	Sí	Sí

Figura 3-14. Tabla de modelos de parámetros del proyecto BlenderBot de Facebook más importantes implementados según el dispositivo

Como se aprecia en la tabla, existe un modelo condensado del segundo modelo más grande. Este sí pudo ser implementado en el ordenador personal, pero al hacer pruebas posteriores en la Raspberry Pi el resultado no fue tan satisfactorio. De ahí la decisión de utilizar finalmente el modelo de 90M de parámetros.

3.4.2 Instalación de ParlAI en la Raspberry Pi

Esta librería es la que se encarga de gestionar la instalación de los modelos, la implementación de las skills, y

además contiene los archivos que nos permitirán interactuar con aquellos. Para poder instalar esta librería hay dos requisitos esenciales que se deben cumplir. Estos son, trabajar sobre una versión de Python 3.7 o superior, y tener instalado Pytorch 1.6 o versiones más actuales.

En este momento, habría que tener activado el entorno creado anteriormente en Python 3. Para comprobar la versión de Python basta con escribir `python3` en la línea de comandos y ver el número de versión. El siguiente paso es instalar la última versión de Pytorch.

```
$ pip3 install pytorch
```

Figura 3-15. Comando para instalación de la librería pytorch.

Una vez cumplimos estos requisitos, podemos iniciar el proceso de instalación de ParlAI.

```
$ pip 3 install parlai
```

Figura 3-16. Comando para instalación de la librería ParlAI.

Este paso ha sido probado tres veces a lo largo del desarrollo del proyecto, y en todos ellos ha tardado en realizarse entre una hora y una hora y media.

Es interesante mencionar que, si no se realizaba el siguiente paso antes de la instalación de ParlAI, ocurría un fallo que anulaba la construcción de la rueda de paquetes. Consiste en la instalación de rustup, como sugería la salida mostrada en la terminal.

```
$ sudo apt update  
$ sudo apt install snapd  
$ sudo reboot  
$ sudo snap install core  
$ sudo snap install rustup --classic
```

Figura 3-17. Instalación de rustup.

La instalación de rustup no fue necesaria en los otros entornos en los que se ha probado el software de ParlAI. Sin embargo, siguiendo el proceso que se comenta en este documento, sí es necesario para poder instalar la librería correctamente.

Para utilizar de forma interactiva el modelo de 90M de parámetros, el comando a ejecutar es el siguiente:

```
$ parlai interactive -t blended_skill_talk -mf zoo:blender/blender_90M/model
```

Figura 3-18. Comando para interactuar con el modelo de 90M de parámetros.

Como parámetros de entrada al archivo interactive, hemos escogido la BST (blended skill talk) explicada en el capítulo del diseño, y el modelo de 90M de parámetros.

```
Enter [DONE] if you want to end the episode, [EXIT] to quit.
02:57:43 | creating task(s): blended_skill_talk
[ loading personas.. ]

[NOTE: In the BST paper both partners have a persona.
      You can choose to ignore yours, the model never sees it.
      In the Blender paper, this was not used for humans.
      You can also turn personas off with --include-personas False]

[context]: your persona: i am in my second year of medical school.
your persona: my favorite thing to do is watch old movies.
Enter Your Message: Hello my friend!
[TransformerGenerator]: hello , how are you today ? i just got back from volunteering at a local animal s
helter .
Enter Your Message: That's amazing. I've had a bad day
[TransformerGenerator]: i ' m sorry to hear that . do you have any hobbies you like to do ?
Enter Your Message: █
```

Figura 3-19. Ejemplo de interacción con el modelo.

3.4.3 Conexión con el resto de los bloques

Como se aprecia en la figura 3-19, las pruebas de interacción realizadas haciendo uso de la librería ParlAI han consistido hasta ahora en una salida de programa escrita por pantalla. La entrada del usuario se introduce como parámetro de entrada, y el modelo escribe la respuesta generada en cada caso.

Para poder controlar esta entrada y salida del programa, se ha optado por programar un bloque superior que controle al bloque generador de texto de forma externa como un subproceso.

Primero se intentó emplear Subprocess, la librería de Python para gestión de subprocesos. Sin embargo, esta solo permite una interacción simple con el subproceso, lo cual es incompatible para una conversación que deba durar varios turnos. Finalmente se emplea la librería Pexpect, disponible también en PyPi. Esta librería implementa funciones que le permiten a un programa padre controlar a un nodo hijo, leyendo su salida y enviando datos que se reciben por el input del hijo. [28]

```
$ pip3 install pexpect
```

Figura 3-20. Instalación de Pexpect.

La ejecución de un programa como subproceso, se realiza de la siguiente manera.


```
from pexpect import spawn  
child = spawn("<comando>")
```

Figura 3-21. Comando para ejecutar un programa como subprocesso.

Si queremos controlar el hijo desde el padre, vamos a necesitar comunicarnos con él para saber cuándo podemos enviar la respuesta y cuándo él ha enviado la suya. Como nosotros sabemos que el programa siempre empieza su mensaje con “[TransformerGenerator]:”, podemos estar leyendo la salida y esperando a que el programa escriba esa cadena. La entrada que tendremos que proporcionar funciona de la misma forma, ya que siempre se escribe “Enter Your Message”. Existe la función `child.expect(string)` dentro de Pexpect que justo implementa esta funcionalidad, así que será la que usaremos para comunicarnos.

```
child.expect(".*Message.*")  
child.sendline(mensaje)
```

Figura 3-22. Esquema básico de la comunicación con el hijo.

3.5 Bloque traductor

Instalamos en primer lugar la librería `googletrans` de Python.

```
$ pip3 install googletrans
```

Figura 3-23. Instalación de la librería `googletrans`.

El uso de esta librería también está basado en una estructura cliente-servidor.

```
from googletrans import Translator  
  
respuestaEspanol = translator.translate(answer, src='en', dest='es')  
respuesta = answerEspanol.text
```

Figura 3-24. Uso de las funciones de `googletrans`.

Los parámetros enviados en la petición son el texto a traducir y los idiomas de entrada y salida, que para este trabajo siempre van a ser el inglés otro idioma en el que se desee comunicarse con el sistema.

3.6 Mejora de la funcionalidad

Con vistas a mejorar la experiencia de usuario, se ha decidido añadir una serie de sonidos para saber en cada momento en qué punto del programa se encuentra (si está esperando recibir respuesta, si está generando la salida ...). Estos sonidos deben reproducirse en paralelo con otras instrucciones que se están ejecutando en el programa, y no pueden entorpecerlas porque harían que los tiempos de respuesta fueran más largos de lo que deberían.

La solución implementada consiste en la utilización de hilos de Python, que son capaces de ejecutar funciones en paralelo con otros procesos de ejecución. La función en nuestro caso solo tiene que ser capaz de reproducir un sonido que le sea transferido como entrada.

```
import threading

def reproducir(audio):
    os.system("mpg123 " + audio)
```

Figura 3-25. Importar librería de threading y creación de función para reproducir audios.

La llamada a esta función será realizada de la siguiente forma, pasando como parámetro de entrada el nombre de la pista de audio en cuestión que será reproducida en cada momento, según el punto del programa en que nos encontremos.

```
x = threading.Thread(target=reproducir, args=(<nombreAudio>,))
x.start()
```

Figura 3-26. Llamada a la función de threading anteriormente creada.

4 RESULTADOS

Quien se enfada por las críticas, reconoce que las tenía merecidas.

- Tácito -

4.1 Alcance final y funcionalidad

Finalmente, se ha conseguido implementar el sistema planteado en un dispositivo de reducidas dimensiones. Es posible la interacción con él mediante la voz, sin necesidad de pulsación de botones o conocimientos técnicos. El diálogo por turnos puede continuar indefinidamente, en cualquier idioma que pueda ser traducido al inglés y viceversa gracias al bloque traductor.

En cuanto a la funcionalidad, hay que seguir esta secuencia de manera continua, muy similar a la comunicación normal que se suele tener con asistentes virtuales como Alexa de Amazon, ya que es muy natural para las personas:

- Para iniciar el diálogo basta con pronunciar en voz alta la Keyword, que ha sido definida como “Samia”. En ese momento empieza a cargar el módulo generador de texto.
- Por los altavoces se reproduce un sonido que indica que se ha detectado la palabra clave y que se está iniciando la aplicación. En ese momento se reproduce una frase de inicio predeterminada, como “hola, ¿qué tal estás?”, seguida del sonido que indicará que está en fase de escucha y que la persona puede hablar.
- La persona dicta en voz alta su mensaje. Cuando haya terminado de hablar, el dispositivo emitirá un sonido de confirmación de recepción de mensaje, y empezará a sonar el audio que indica que está cargando la respuesta.
- La respuesta es reproducida por los altavoces seguida del sonido que indica que se activa el modo de escucha.

Una vez se realice esta secuencia, la persona dictará un nuevo mensaje, de modo que se continuará iterando entre los dos últimos pasos descritos anteriormente de forma indefinida. La conversación terminará cuando el usuario dicte el mensaje clave “Adiós”. En ese momento retrocede a la fase de escucha inicial, en la que se espera que alguien dicte la palabra de inicio de conversación para empezar de nuevo la secuencia.

4.2 Funcionamiento

Al final de la etapa de implementación se decidió utilizar el modelo de 90M de parámetros en la Raspberry Pi, ya que modelos más grandes no ha sido posible probarlos en este dispositivo por limitaciones del hardware. Sin embargo, a pesar de ser el modelo más pequeño de los modelos principales, el rendimiento era bajo. Los tiempos

de generación de respuesta oscilaban en torno a los 20 segundos, lo cual influía muy negativamente en la fluidez de la conversación.

Como solución a este problema, se hizo una prueba con el modelo “tutorial transformer generator”, incluido en el mismo conjunto de modelos que los anteriores, los llamados “zoo models”. Es el modelo que se utiliza en el notebook interactivo de Google Colab que ParlAI deja disponible desde el repositorio de GitHub. Se trata de un modelo de 87M de parámetros, entrenado con conversaciones de Reddit. [29]

El rendimiento mejoró bastante, midiendo ahora tiempos de menos de 10 segundos de generación de respuesta. Esto permite disfrutar más la conversación, llegando a ser molesto esperar mucho tiempo. En cuanto a la precisión en la respuesta, nos ofrece un resultado muy similar al modelo de 90M de parámetros.

Es lógico intentar implementar modelos lo más grandes posibles, debido a la clara mejora en las respuestas que ofrecen los más grandes. Sin embargo, hay otros factores para tener en cuenta, como lo es la capacidad de procesamiento de la GPU, el tiempo de generación de respuesta o como podría ser el espacio ocupado, aunque este último en este caso no lo ha sido. Teniendo por delante todos estos datos y haciendo diferentes pruebas, se ha optado definitivamente por el modelo “tutorial transformer generator”, ya que es el que mejor experiencia de usuario ofrece según las herramientas de las que disponemos.

4.3 Prueba con el bloque de traducción

En principio, el bloque traductor fue pensado como un anexo a la línea de diseño principal, sin la esperanza de que diera resultados aceptables. Con tantos pasos intermedios, lo lógico era pensar que el mensaje iba a ser perturbado por el camino, y que iba a ser imposible comunicarse correctamente.

Por el contrario, este bloque presentó unas respuestas muy fiables en comparación a la traducción que haría una persona capaz de comunicarse en los dos idiomas en los que se ha trabajado, en español y en inglés. Además, apenas añadía retraso a la cadena de bloques inicial, ya que los tiempos de respuesta del servidor son menores del orden de los segundos, prácticamente imperceptibles para una persona.

El problema que podría añadir este bloque es la necesidad de mantenerse conectado a internet para poder utilizar la aplicación. Sin embargo, contamos con otros bloques que también realizan peticiones a servidor, como el bloque speech-to.text o el bloque text-to-speech, y, por tanto, esto no se ha considerado un impedimento.

Se ha decidido finalmente incorporar este bloque de manera fija por las múltiples ventajas que ofrece. Por la naturaleza del trabajo, el idioma por defecto desde el que se escucha y al que se traduce es el español. Este parámetro puede ser editado dentro del código sin necesidad de implementar nuevas funcionalidades.

4.4 Pruebas de interacción con personas

El objetivo del proyecto era construir el sistema completo en un dispositivo, de modo que permitiera la comunicación entre este y una persona. La motivación consistía en utilizar este sistema por medio de un dispositivo de dimensiones reducidas, para mejorar el ánimo de personas mayores que pasan muchas horas en soledad.

Sin embargo, no han sido realizadas pruebas suficientes con personas como para obtener conclusiones. Esto requeriría de un estudio con multitud de personas, a las que se les proporcionaría un dispositivo para que interactuaran con él durante el tiempo que durase el estudio. De forma paralela, un equipo especializado debería recopilar datos para elaborar una hoja de resultados de interés, de los cuales podríamos redactar conclusiones.

Este sería el método para justificar de forma clara la motivación, pero por motivos de extensión, este trabajo no

ha podido alcanzar ese punto. En cuanto al producto, sería conveniente perfilar muchos detalles, como la base de datos con la que ha sido entrenada que podría ser específica para este caso de uso, antes de hacer un estudio de este calibre.

Las pruebas realizadas con personas se han centrado más en la usabilidad del dispositivo, la facilidad para dialogar con él. Ha sido probado por un grupo de 10 personas, de edades entre 18 y 65 años. Estas personas concuerdan en que el tiempo de generación de texto podría ser un poco más rápido para mejorar la fluidez de la conversación, algo que ya habíamos previsto en el capítulo de implementación. Aun así, todas ellas, después de que se les explicara en qué consistían los turnos de palabra, pudieron comunicarse correctamente con el dispositivo y no hubo problemas en ese aspecto.

La generación de texto es más acertada y resultó más natural para los usuarios al dar oraciones de entrada por parte de los mismos no demasiado largas y concretas con el modelo implementado.

4.5 Cumplimiento de requisitos

Se ha realizado una matriz de verificación, para revisar si han sido alcanzados los requisitos planteados en el capítulo 1. No todos estos requisitos eran indispensables, pero el cumplimiento o incumplimiento de algunos de ellos ha definido el alcance final.

Req.	Nombre Req.	Verificación				Descripción prueba	Estado
		I	A	D	T		
F.1	Detección de keyword				X	Detección de ‘Samia’ y ‘Sam’ como diminutivo	Completado, mejorable
F.2	Control de audio				X	Reproducción y grabación de audio	Completado
F.3, P.2, P.4	Capacidad de transcribir y dictar			X		Conversión de voz a texto y de texto a voz	Completado
F.4	Generar texto continuamente		X			Integración de Pexpect y BlenderBot de ParlAI	Completado
F.5, P.3	Prueba con personas			X		Prueba de interacción con personas	Completado
P.1	Medidas de tiempo			X		Pruebas con personas, obteniendo opinión	Pendiente de mejora
O.1	Conexión SSH				X	Conexión con el dispositivo de forma local y remota	Completado
S.1	Privacidad		X			Cumplimiento de la ley de protección de datos	Pendiente

Figura 4-1. Matriz de verificación.

4.6 Presupuesto del proyecto

Por un lado, está el precio por sistema completo según ha sido diseñado, que consta de las siguientes partes:

- RaspberryPi modelo 4B, 4GB RAM: **64€**
- Fuente de alimentación RaspberryPi: **9.03€**
- Micrófono Trust, conexión USB: **18.99€**
- Altavoz de escritorio, conexión Jack 3.5mm: **9.99€**

En total el sistema tiene un precio de **102.01€**.

En cuanto al coste del personal, en este caso solo una persona ha trabajado en el proyecto. Se ha trabajado durante 4 meses, por tanto, si el sueldo mínimo para la titulación es de **1687.02€** al mes, el total sería **6748.08€**. [30]

5 DISCUSIÓN

La mejor manera de predecir el futuro es creándolo.

- Peter Drucker -

5.1 Conclusión

Tras haber alcanzado el objetivo planteado en un principio, el proyecto ha llegado a su fin de manera exitosa. La usabilidad del dispositivo es la esperada, pudiendo mantener conversaciones a cualquier hora del día y estas pueden durar todo lo que se desee. El dispositivo elegido ha dado buenos resultados también, permitiéndonos integrar el sistema completo en él.

Al compararlo con otros dispositivos comerciales que ofrecen la funcionalidad de mantener una conversación, este sistema ofrece resultados muy similares de conversación. Esto se lo debemos en gran parte a la implementación del bloque generador de texto mediante BlenderBot de Facebook Research, ya que la naturalidad y adecuación en la respuesta es la parte más compleja del sistema, y es una de las que más directamente influye en la mejora de la experiencia de usuario. En este sentido, el proyecto cuenta con una gran ventaja a largo plazo, y es que podremos implementar otro modelo generativo si fuese posible en un futuro, de modo que mejore el tiempo de generación de respuesta y la naturalidad de esta.

Al tratarse la capacidad de comunicarse de una funcionalidad novedosa, los usuarios que lo han probado han mostrado interés y asombro, a la vez que un poco de rechazo en un principio, lo cual podría verse como una ventaja o desventaja según el caso. Si se pensara en comercializar el dispositivo, sería conveniente realizar encuestas a un gran número de personas para obtener una opinión más generalizada y poder determinar la aceptación/rechazo del mismo.

Con respecto al grupo de personas que han probado la aplicación, por lo general la primera impresión es más cercana al rechazo cuanto mayor es la edad de la persona. Sin embargo, una vez terminada la prueba de diálogo las buenas sensaciones son un poco más generalizadas, aunque de nuevo, no son datos determinantes.

La mayor desventaja que presenta este proyecto es la mala implementación del bloque detector de keyword, ya que permanece escuchando todo el tiempo, y por tanto no cumple la Ley Orgánica de Protección de Datos Personales 3/2018.

5.2 Casos de uso

Compañía para mayores en soledad: es la idea principal de este Trabajo de Fin de Grado, y la motivación para realizarlo. Basándonos en los productos comerciales comentados en el capítulo introductor, es un caso de uso muy interesante que ayudaría a un gran número de personas. Requeriría un estudio específico del producto final con el grupo de edad que comentamos, además de un posible entrenamiento específico de la red generativa para un mejor desempeño a la hora de generar respuestas.

Despertador interactivo: este caso de uso surge de la inspiración en algunos despertadores interactivos que

requieren de cierta acción por parte del ser humano, como agitar el dispositivo o decir una palabra clave. También existen otros modelos que son capaces de grabar un mensaje determinado y reproducirlo. En nuestro caso, podríamos programar el dispositivo para que funcionase a modo de despertador, dándole al usuario la posibilidad de conversar en caso de que se encuentre en soledad, lo que ayudaría a la persona a despertarse.

Necesitaríamos un tiempo de generación de respuesta especialmente bajo, para no hacer esperar a la persona con el consecuente riesgo de que se queda dormida. Una funcionalidad interesante sería generar una nueva respuesta si la persona no responde al cabo de cierto tiempo, suponiendo que se ha quedado dormida.

Este producto sería interesante para personas que tengan problemas para vencer al sueño al despertarse. No necesitaría ser entrenada con una base de datos específica, pero sí de forma que proponga temas de conversación distintos, algo que ya implementa nuestro dispositivo.

Ayuda para niños con autismo: esta idea se basa en el proyecto de LuxAI comentado en la introducción, ya que uno de sus casos de uso es la ayuda en la educación de niños con autismo. Utilizando un dispositivo de estas características, podría mejorar la interacción entre un tutor humano y un niño, aunque por el contrario quizás también podría favorecer el aislamiento del niño y reducir su comunicación, que es justo lo que se pretende evitar.

Esta línea es la menos documentada de las planteadas, y se propone más como un campo de estudio en el que podría ser interesante el empleo de tecnologías de este tipo.

5.3 Líneas futuras

Para definir los caminos que podrían mejorar este proyecto, hay que determinar primero las carencias más importantes que tenemos. Para hacer una cadena más resistente, debemos reforzar el eslabón más débil.

El principal problema que encontramos al tener una visión general del proyecto se encuentra en la implementación del bloque detector de palabra clave. Aunque la funcionalidad ha sido implementada de manera exitosa, no ha sido del modo más elegante, tal y como se comentó en el capítulo de implementación. Acarrea un grave problema de privacidad que no podemos pasar por alto, por tanto, se deja abierta la búsqueda de otro método para la detección de la palabra clave. Se propone la implementación de una red neuronal entrenada específicamente para este propósito, de creación propia o de otro proyecto, si existiese.

También nos encontramos por lo general con tiempos bajos de generación de respuesta, lo cual afecta de forma directa a la experiencia de usuario, llegando a ser inútil si estos tiempos son excesivos. Este problema nos ha hecho toparnos con un muro, ya que la capacidad del hardware es limitada, y nos gustaría que fuera lo más grande posible. Por lo tanto, se plantea la posibilidad de seguir un proceso de implementación similar al descrito en este proyecto, pero utilizando otro dispositivo más potente, pero de características similares, como alguno de los modelos Jetson de NVIDIA. En esta línea, sería interesante la integración de todo el sistema en un dispositivo único, que no necesite micrófono ni altavoces conectados.

Mirando a largo plazo, la actualización de este sistema implicaría la búsqueda de nuevos proyectos disponibles más modernos, como GPT-3 o Google Lambda. Volviendo a nuestro proyecto implementado, el BlenderBot de Facebook, existen muchos modelos de parámetros disponibles para descarga, y no todos han podido ser probados, aunque sí los principales. Esto deja abierta la posibilidad de encontrar un modelo que se ajuste mejor al funcionamiento buscado, y mejore la experiencia de usuario.

Como funcionalidad añadida, se pensó como idea en la capacidad de recibir comandos y realizar acciones. Este es un comportamiento típico de los asistentes de voz más populares del mercado y no supondría una gran dificultad a la hora de implementarlo. Parecida a esta, existe otra funcionalidad que consiste en condicionar las respuestas según la frase introducida y el contexto de estar hablando con una persona mayor y, en particular al contexto de nuestro proyecto, hablando con una persona mayor. De este modo se piensa que la conversación sería mucho más adecuada al ejemplo de uso, aunque requeriría un mayor estudio del contexto en cuestión y del entrenamiento de la red generativa.

REFERENCIAS

- [1] M. RODRIGUEZ MARTIN, «La soledad en el anciano,» *Gerokomos[online]*, vol. 20, nº 4, p. 8, 2009.
- [2] S. Shourjya, «forbes.com,» Forbes, 31 Oct 2018. [En línea]. Available: <https://www.forbes.com/sites/shourjyasanyal/2018/10/31/how-is-ai-revolutionizing-elderly-care/?sh=32bf9036e07d>. [Último acceso: 26 5 2021].
- [3] Catalia Health Inc., «Catalia Health,» 2020. [En línea]. Available: <http://www.cataliahealth.com/>. [Último acceso: 26 5 2021].
- [4] Institution Robotics, «institution robotics,» 2021. [En línea]. Available: <https://www.intuitionrobotics.com/>. [Último acceso: 26 5 2021].
- [5] LuxAI S.A., «LuxAI,» 2021. [En línea]. Available: <https://luxai.com/>. [Último acceso: 26 5 2021].
- [6] H. V. h. J. P. Q. y. A. M. H. A. Cortez Vásquez, «Procesamiento de lenguaje natural,» *Rev.Investig.sist.inform.*, vol. 6, nº 2, pp. 45-54, 2009.
- [7] E. K. a. E. L. Steven Bird, *Natural Language Processing with Python*, USA: O'REILLY, 2009.
- [8] OpenAI, «OpenAI,» 14 2 2019. [En línea]. Available: <https://openai.com/blog/better-language-models/>. [Último acceso: 27 5 2021].
- [9] OpenAI, «github,» 2 12 2020. [En línea]. Available: <https://github.com/openai/gpt-2>. [Último acceso: 27 5 2021].
- [10] Google, «Google Developers,» 2021. [En línea]. Available: <https://developers.google.com/learn/topics/chatbots>. [Último acceso: 27 5 2021].
- [11] Facebook, «Facebook AI,» 29 4 2020. [En línea]. Available: <https://ai.facebook.com/blog/state-of-the-art-open-source-chatbot/>. [Último acceso: 27 5 2021].
- [12] OpenAI, 11 6 2020. [En línea]. Available: <https://openai.com/blog/openai-api/>. [Último acceso: 31 5 2021].
- [13] OpenAI, «OpenAI API beta,» 11 5 2020. [En línea]. Available: <https://beta.openai.com/docs/introduction>. [Último acceso: 31 5 2021].
- [14] Google, «The Keyword,» 18 5 2021. [En línea]. Available: <https://blog.google/technology/ai/lamda/>. [Último acceso: 31 5 2021].

-
- [15] E. Upton, «Raspberry Pi Blog,» Raspberry Pi, 8 9 2016. [En línea]. Available: <https://www.raspberrypi.org/blog/ten-millionth-raspberry-pi-new-kit/>. [Último acceso: 18 6 2021].
- [16] C. Custer, «DATAQUEST,» 1 10 2020. [En línea]. Available: <https://www.dataquest.io/blog/how-long-does-it-take-to-learn-python/>. [Último acceso: 5 6 2021].
- [17] I. Vyas, «Let's Talk,» 28 12 2020. [En línea]. Available: <https://citrusbug.com/article/best-python-libraries-for-machine-learning>. [Último acceso: 7 6 2021].
- [18] A. Zhang, «PyPi,» 5 12 2017. [En línea]. Available: <https://pypi.org/project/SpeechRecognition/>. [Último acceso: 6 6 2021].
- [19] P. N. Durette, «PyPi,» 4 2 2021. [En línea]. Available: <https://pypi.org/project/gTTS/>. [Último acceso: 6 6 2021].
- [20] N. M. Bhat, «PyPi,» 6 7 2020. [En línea]. Available: <https://pypi.org/project/pyttsx3/>. [Último acceso: 6 6 2021].
- [21] Kitt-AI, «Gihub,» 31 12 2020. [En línea]. Available: <https://github.com/Kitt-AI/snowboy>. [Último acceso: 6 6 2021].
- [22] Picovoice, «Picovoice,» 2021. [En línea]. Available: <https://picovoice.ai/docs/quick-start/console-porcupine/>. [Último acceso: 6 6 2021].
- [23] S. Han, «Pypi,» 14 6 2020. [En línea]. Available: <https://pypi.org/project/googletrans/>. [Último acceso: 6 6 2021].
- [24] Google, «Google Assitant,» 1 3 2020. [En línea]. Available: <https://developers.google.com/assistant/sdk/guides/service/python/embed/audio>. [Último acceso: 6 6 2021].
- [25] Google, «gTTS,» [En línea]. Available: <https://gtts.readthedocs.io/en/latest/module.html#playing-sound-directly>. [Último acceso: 6 6 2021].
- [26] facebookresearch, «GitHub,» 23 4 2021. [En línea]. Available: <https://github.com/facebookresearch/ParlAI>. [Último acceso: 6 6 2021].
- [27] E. D. N. G. D. J. M. W. Y. L. J. X. M. O. K. S. E. M. S. Y.-L. B. J. W. Stephen Roller, «Recipes for building an open-domain chatbot,» *arXiv preprint arXiv:2004.13637*, 2020.
- [28] PyPi, «Pexpect,» 2013. [En línea]. Available: <https://pexpect.readthedocs.io/en/stable>. [Último acceso: 6 6 2021].
- [29] ParlAI, «ParlAI Documentation,» 2020. [En línea]. Available: <https://parl.ai/docs/zoo.html>. [Último acceso: 7 6 2021].
- [30] Agencia Estatal Boletín Oficial del Estado, «núm. 15,» de *BOE*, Ministerio de Empleo y Seguridad Social, 2017, p. 4356 a 4382.
- [31] T. Demeester, T. Rocktäschel, and S. Riedel, “Lifted rule injection for relation embeddings,” *EMNLP 2016 - Conf. Empir. Methods Nat. Lang. Process. Proc.*, pp. 1389–1399, 2016, doi: 10.18653/v1/d16-1146.

- [32] A. Vaswani *et al.*, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-December, no. Nips, pp. 5999–6009, 2017.
- [33] S. Hoppe and M. Toussaint, “Qgraph-bounded Q-learning: Stabilizing Model-Free Off-Policy Deep Reinforcement Learning,” 2020, [Online]. Available: <http://arxiv.org/abs/2007.07582>.
- [34] S. Roller *et al.*, “Recipes for building an open-domain chatbot,” 2020, [Online]. Available: <http://arxiv.org/abs/2004.13637>.
- [35] H. Rashkin, E. M. Smith, M. Li, and Y. L. Boureau, “Towards empathetic open-domain conversation models: A new benchmark and dataset,” *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, pp. 5370–5381, 2020, doi: 10.18653/v1/p19-1534.

