

---

Desarrollo de una herramienta basada en  
lenguaje de apoyo a la terapia basada en  
reminiscencia  
A language-based tool to support reminiscence  
therapy

---



Trabajo de Fin de Grado  
Curso 2023–2024

Autor  
Marta Vicente Navarro

Director  
Gonzalo Mendez Pozo

Grado en Doble grado en Ingeniería Informática y  
Matemáticas

Facultad de Informática  
Universidad Complutense de Madrid



Desarrollo de una herramienta basada en  
lenguaje de apoyo a la terapia basada en  
reminiscencia

A language-based tool to support  
reminiscence therapy

Trabajo de Fin de Grado en Doble grado en Ingeniería  
Informática y Matemáticas

**Autor**

Marta Vicente Navarro

**Director**

Gonzalo Mendez Pozo

**Convocatoria:** *Junio* 2024

Grado en Doble grado en Ingeniería Informática y  
Matemáticas

Facultad de Informática

Universidad Complutense de Madrid

3 de mayo de 2024



# Dedicatoria

*A Mamá*



# Agradecimientos





# Resumen

## Desarrollo de una herramienta basada en lenguaje de apoyo a la terapia basada en reminiscencia

La enfermedad de Alzheimer es una condición neurodegenerativa que afecta las funciones cognitivas, la memoria, el pensamiento y el comportamiento. Su llegada supone un cambio significativo en la vida de quienes la padecen y en su entorno. Actualmente, se estima que en España 900.000 personas sufren esta y otras formas de demencia, y se proyecta que los casos se duplicarán para el año 2050. Por lo tanto, es de vital importancia desarrollar técnicas que puedan ralentizar el avance de la enfermedad. Aunque no es reversible, existen terapias que pueden mejorar la calidad de vida tanto del paciente como de sus seres queridos.

La terapia de reminiscencia es una modalidad terapéutica que se enfoca en ayudar a las personas a recordar y compartir sus experiencias y recuerdos pasados, especialmente aquellos relacionados con eventos significativos en sus vidas. Aunque es comúnmente empleada con personas mayores, también puede resultar efectiva en otros grupos de edad.

Este proyecto se centra principalmente en el desarrollo de un ChatBot que interactúe con el paciente, recopilando la información necesaria para construir una historia de vida y las imágenes asociadas. Para lograrlo, se ha clonado la API de Gemini y se ha entrenado con datos personalizados, de manera que genere historias de vida específicas y pertinentes.

Este enfoque promete ser una herramienta valiosa para mejorar la calidad de vida de las personas afectadas por Alzheimer y demencias similares.

## Palabras clave

Reminiscencia, Chatbot, historia de vida, alzhéimer



# Abstract

## A language-based tool to support reminiscence therapy

Alzheimer's disease is a neurodegenerative condition that affects cognitive functions, memory, thinking, and behavior. Its onset marks a significant change in the lives of those affected and their surroundings. Currently, it is estimated that 900,000 people in Spain suffer from this and other forms of dementia, and it is projected that cases will double by the year 2050. Therefore, it is crucial to develop techniques that can slow down the progression of the disease. Although it is not reversible, there are therapies that can improve the quality of life for both the patient and their loved ones.

Reminiscence therapy is a therapeutic approach focused on helping individuals remember and share their past experiences and memories, especially those related to significant events in their lives. While commonly used with older individuals, it can also be effective with other age groups.

This project primarily focuses on the development of a ChatBot that interacts with the patient, gathering the necessary information to construct a life story along with associated images. To achieve this, the Gemini API has been cloned and trained with customized data, enabling it to generate specific and relevant life stories.

This approach holds promise as a valuable tool for enhancing the quality of life for individuals affected by Alzheimer's and similar dementias.

## Keywords

reminiscence, chatbot, life story, Alzheimer's



# Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	2
1.1.1. Enfermedad de Alzheimer . . . . .	2
1.1.2. Terapias de reminiscencia . . . . .	3
1.1.3. Historia de vida . . . . .	4
1.2. Objetivos . . . . .	5
1.3. Estructura de la memoria . . . . .	5
<b>Introduction</b>	<b>7</b>
1.4. Motivation . . . . .	8
1.4.1. Alzheimer's disease . . . . .	8
1.4.2. Reminiscence Therapy . . . . .	9
1.4.3. Life History . . . . .	10
1.5. Objectives . . . . .	10
1.6. Document Structure . . . . .	11
<b>2. Estado de la Cuestión</b>	<b>13</b>
2.1. Evolución del Procesamiento del Lenguaje Natural . . . . .	14
2.2. Word embeddings . . . . .	14
2.2.1. TF-IDF . . . . .	15
2.2.2. Bolsa de palabras . . . . .	15
2.2.3. Word2Vec . . . . .	16
2.3. Transformers . . . . .	17
2.3.1. Aprendizaje por transferencia . . . . .	17

2.4.	Modelos de Lenguaje de Gran Tamaño (LLM)	18
2.4.1.	BERT	19
2.4.2.	T5	20
2.4.3.	GPT <i>Generative Pretrained Transformer</i>	20
2.4.4.	Llama	24
2.4.5.	LaMDA	24
2.5.	Alucinaciones	25
2.6.	Otros trabajos relacionados	27
2.6.1.	Proyecto Cantor	27
2.6.2.	Celia	29
<b>3.</b>	<b>Tecnologías utilizadas</b>	<b>31</b>
3.1.	Bibliotecas de Procesamiento del Lenguaje en Python	32
3.1.1.	NLTK	32
3.1.2.	SpaCy	34
3.2.	APIs de procesamiento del lenguaje	36
3.2.1.	Bard	36
3.2.2.	Gemma	37
3.2.3.	GPT API	38
3.2.4.	Rasa	38
3.3.	Gemini	38
3.3.1.	Procesamiento de texto	39
3.3.2.	Procesamiento de imágenes	42
3.4.	Almacenamiento de la información	44
3.4.1.	JSON	44
3.4.2.	RDF	45
3.5.	Desarrollo de la interfaz	46
3.6.	Programación orientada a objetos	47
3.7.	VPN	48
<b>4.</b>	<b>Desarrollo de prototipos</b>	<b>51</b>
4.1.	Planteamiento del problema	51
4.2.	Preguntas predefinidas	52

4.3.	BARD . . . . .	52
4.4.	Prototipo usando GEMMA . . . . .	55
4.4.1.	Google Collaborate . . . . .	55
4.4.2.	Linux . . . . .	56
4.5.	Desarrollo de la interfaz con Telegram . . . . .	57
4.6.	Gemini . . . . .	59
4.7.	Análisis de las respuestas . . . . .	59
4.7.1.	Análisis de las respuestas y identificación de la información omitida . . . . .	59
4.7.2.	Generación de preguntas para obtener la información faltante . . . . .	60
4.8.	Interfaz e interacción con el usuario . . . . .	60
4.8.1.	Integración de Gemini y Telegram . . . . .	61
4.8.2.	Telegram . . . . .	62
4.9.	Extracción de información a partir de imágenes . . . . .	62
<b>5.</b>	<b>Diseño de un chatbot de ayuda a la terapia de reminiscencia</b>	<b>65</b>
5.1.	Herramientas y puesta en marcha . . . . .	65
5.1.1.	VPN . . . . .	65
5.1.2.	Instalación de la API de Gemini . . . . .	66
5.1.3.	Puesta en marcha . . . . .	66
5.2.	Arquitectura del sistema . . . . .	66
5.2.1.	Servidor . . . . .	66
5.2.2.	Procesamiento . . . . .	67
5.2.3.	Almacenamiento y manejo de la información . . . . .	68
5.3.	Interfaz e interacción con el usuario . . . . .	68
<b>6.</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>71</b>
6.1.	Estado actual del procesamiento del lenguaje . . . . .	71
6.2.	Desarrollo del proyecto . . . . .	72
6.3.	Resultado final . . . . .	73
6.4.	Trabajo futuro . . . . .	74
	<b>Conclusions and Future Work</b>	<b>75</b>
6.5.	Actual state of PLN . . . . .	75

6.6. Project development . . . . .	76
6.7. Final result . . . . .	77
6.8. Future work . . . . .	77

<b>Bibliografía</b>	<b>79</b>
---------------------	-----------



# Índice de figuras

2.1. Desarrollo de los LLM por Yang et al. (2023) . . . . .	18
2.2. Diagrama del modelo T5 de Raffel et al. (2020) . . . . .	20
2.3. Arquitectura de decodificadores usada por GPT-2 . . . . .	22
2.4. Ejemplo de alucinación de ChatGPT Wikipedia (Fecha de última modificación) . . . . .	26
2.5. Ejemplo de alucinación en una imagen Rohrbach et al. (2023) . . . . .	26
3.1. Árbol sintáctico generado con nltk . . . . .	33
3.2. Árbol sintáctico generado con nltk . . . . .	35
3.3. Ejemplo de uso de BARD. . . . .	36
3.4. Ejemplo de uso de GEMMA. . . . .	37
3.5. Imagen de entrada a Gemini en el primer ejemplo . . . . .	42
3.6. Imagen de entrada a Gemini en el segundo ejemplo . . . . .	43
3.7. Imagen de entrada a Gemini en el ejemplo . . . . .	44
3.8. Ejemplo de un recurso RDF. . . . .	45
3.9. Ejemplo de un recurso RDF SCHEMA. . . . .	46
3.10. Ejemplo de interfaz generada con PyQT5 . . . . .	47
4.1. Ejemplo de uso de BARD. . . . .	53
4.2. Muestra de la interacción chatbot-usuario en el primer prototipo . . . . .	54
4.3. Ejemplo de uso de BARD. . . . .	58
4.4. Análisis de respuestas . . . . .	60
4.5. Generación de preguntas . . . . .	60
4.6. Ejemplo de extracción de información a partir de imágenes . . . . .	63

5.1. Arquitectura del sistema . . . . .	67
5.2. Ejemplo de bienvenida y primeras interacciones con la versión en ordenador . . . . .	69
5.3. Ejemplo de bienvenida y primeras interacciones con la versión en móvil	70

# Introducción

*“Hay enfermos incurables, pero ninguno incuizable”*  
— Francesc Torralba

El constante aumento en la esperanza de vida a nivel mundial ha dado lugar a una población cada vez más envejecida. Este hecho ha traído consigo un aumento en las enfermedades asociadas con la vejez, como el Alzheimer y otros tipos de demencia. Los esfuerzos de la investigación en estos campos se han centrado en desarrollar tratamientos que puedan ralentizar el avance de esta enfermedad. La memoria juega un papel fundamental en la formación de nuestra identidad a través de nuestras experiencias y vivencias. Su pérdida puede ser devastadora, ya que afecta profundamente a la calidad de vida de la persona y de quienes la rodean.

Entre las estrategias no farmacológicas para el tratamiento del deterioro cognitivo y la memoria, destaca la terapia de reminiscencia. Esta técnica busca estimular los recuerdos pasados del paciente, lo que puede tener beneficios significativos para su bienestar social, mental y emocional.

Por otro lado, la inteligencia artificial está cada día más presente en nuestra vida diaria. Tiene una gran variedad de usos y aplicaciones que abarcan desde asistentes virtuales en nuestros teléfonos inteligentes hasta sistemas de recomendación en plataformas de streaming y comercio electrónico. Además, la inteligencia artificial se utiliza en la medicina para diagnósticos más precisos, en la industria para automatizar procesos de fabricación e incluso en la conducción autónoma de vehículos. Esta omnipresencia de la inteligencia artificial está transformando profundamente diversos sectores, impulsando la eficiencia, la personalización de servicios y generando nuevas oportunidades de desarrollo tecnológico.

El presente proyecto trata de encontrar la forma de aplicar la inteligencia artificial, y en concreto el procesamiento del lenguaje a la terapia de reminiscencia. En concreto, se busca desarrollar una herramienta conversacional de apoyo para la realización de este tipo de terapia.

Este capítulo tiene tres objetivos principales. En primer lugar, explicar cuál es la motivación actual con la que parte este problema. En segundo lugar, dejar claros cuáles son los objetivos que se plantean como punto de partida del proyecto.

Finalmente, con la intención de dar una visión global del proyecto, que permita entenderlo y hacerse una idea de como es la presente memoria, habrá una sección final en la que se explicará la estructura de este documento, haciendo hincapié en qué se puede encontrar en cada capítulo.

## 1.1. Motivación

A pesar de los avances en la investigación médica, todavía no se ha encontrado una cura definitiva para el Alzheimer. La mayoría de los tratamientos se centran en aliviar los síntomas y ralentizar la progresión de la enfermedad. Entre estos tratamientos, la terapia de reminiscencia ha surgido como una opción no farmacológica destacada.

Esta terapia se centra en estimular los recuerdos del pasado del paciente, lo que puede tener beneficios significativos en su bienestar social, mental y emocional. Ayuda a las personas a recordar y compartir experiencias pasadas, lo que puede ser reconfortante y estimulante, especialmente para aquellos que enfrentan el desafío de la pérdida de memoria asociada con el Alzheimer.

Durante el período académico 2023-2024, se desarrolló la aplicación *chatbot yayo* con el objetivo de facilitar a los terapeutas la realización de terapias basadas en reminiscencia, simplificando y agilizando el proceso. Este desarrollo parte desde cero y busca crear un chatbot funcional y útil, capaz de mantener una conversación útil con el paciente para extraer la información necesaria.

Este Trabajo de Fin de Grado tiene como propósito asistir a terapeutas, familiares o amigos de pacientes con demencia en la obtención del material necesario para llevar a cabo terapias de reminiscencia. Se busca mejorar la eficacia de estas terapias y, en consecuencia, la calidad de vida tanto de los pacientes como de sus familiares.

### 1.1.1. Enfermedad de Alzheimer

La enfermedad de Alzheimer (EA) es la principal causa de demencia en adultos mayores. La EA es una enfermedad compleja que viene determinada por múltiples factores. A veces es hereditaria y está caracterizada por la pérdida de neuronas y sinapsis, junto con la presencia de placas seniles y degeneración neurofibrilar. Clínicamente, se manifiesta como una demencia progresiva, comenzando con fallas en la memoria reciente y avanzando hasta la dependencia total del paciente.

Según Donoso (2003) La incidencia de la EA aumenta con la edad, siendo rara antes de los 50 años y afectando a un 1-2% de los sujetos a los 60 años, 3-5% a los 70, 15-20% a los 80 y hasta la mitad de los mayores de 85 años. Es más frecuente en mujeres, posiblemente debido a la mayor longevidad.

La enfermedad se desarrolla en distintas etapas. En la etapa inicial, los síntomas pueden pasar desapercibidos o atribuirse a simples descuidos. La persona afectada puede experimentar dificultades para recordar nombres, eventos recientes o encon-

trar las palabras adecuadas en conversaciones. A pesar de estos desafíos, generalmente conservan la capacidad de realizar tareas cotidianas con cierta independencia. Sin embargo, es posible que comiencen a perder interés en actividades previamente disfrutadas.

A medida que la enfermedad progresa, los síntomas se vuelven más evidentes y problemáticos. La pérdida de memoria se vuelve más pronunciada, con dificultades para reconocer a familiares y amigos cercanos. Además, pueden surgir problemas de orientación en tiempo y espacio, lo que puede resultar en desorientación incluso en entornos familiares. Las habilidades de comunicación también se ven afectadas, con dificultades para seguir conversaciones o expresar pensamientos de manera coherente.

Finalmente, el Alzheimer alcanza su punto más devastador en la última etapa. La pérdida de memoria es profunda y completa, con una incapacidad para recordar incluso eventos recientes o reconocer caras familiares. La persona afectada puede experimentar cambios significativos en la personalidad y el comportamiento, volviéndose agitada, ansiosa o incluso agresiva en ocasiones. La capacidad para realizar actividades básicas de la vida diaria, como vestirse o alimentarse, se ve seriamente comprometida, y la supervisión constante se vuelve esencial.

El tratamiento farmacológico de la enfermedad de Alzheimer (EA) se basa en el uso de medicamentos para mejorar los defectos cognitivos y corregir trastornos conductuales. Los fármacos más valorados son los inhibidores de la acetilcolinesterasa, que intentan compensar la pérdida de neuronas colinérgicas en la corteza cerebral. El tratamiento también abarca fármacos para tratar trastornos conductuales, como antidepresivos, tranquilizantes o inductores del sueño (Chung y Cummings (2000)).

Además del tratamiento farmacológico, la actividad física y mental es fundamental para estimular la actividad cerebral y prevenir la pérdida de memoria. La orientación familiar es esencial para ayudar a los familiares a manejar los trastornos conductuales y aumentar la calidad de vida de los pacientes con EA. Un ejemplo de terapia no farmacológica que ayuda a mantener la actividad mental es la terapia de reminiscencia.

### 1.1.2. Terapias de reminiscencia

La reminiscencia, según la definición de la CEAFA (Confederación Española de Asociaciones de Familiares de personas con Alzheimer y otras demencias), es una técnica que busca evocar recuerdos en las personas, especialmente aquellos relacionados con eventos importantes de su vida.

En las terapias de reminiscencia se emplean diversos materiales, como fotografías, vídeos, noticias de periódicos, audios y objetos significativos, con el objetivo de estimular el recuerdo y activar la memoria a través de las emociones que despiertan estos elementos en el paciente.

Además de estimular los cinco sentidos para provocar el recuerdo, también se utiliza la narración de historias conocidas por el paciente. Por lo tanto, es crucial

conocer las experiencias pasadas del individuo afectado para adaptar los materiales utilizados en las terapias. La participación de personas cercanas al paciente, como familiares y cuidadores, en ejercicios de reminiscencia puede mejorar significativamente su calidad de vida.

Es recomendable construir relatos basados en las historias de vida del paciente para estos ejercicios de memoria. Las terapias de reminiscencia se consideran como parte de un proceso más amplio en el que el individuo intenta recuperar y vincular recuerdos que abarcan gran parte de su vida.

Una revisión sobre los posibles efectos de la terapia de reminiscencia en personas con demencia y sus cuidadores concluyó que disminuía las alteraciones de conducta y los síntomas depresivos, y que mejoraba la cognición (Huang et al. (2015)). En otra revisión se observaron mejorías en el estado de ánimo, la habilidad cognitiva, la conducta social y el bienestar general (Cotelli et al. (2012)), y recientemente se encontró que la terapia de reminiscencia mejoraba la salud física y que aumentaba la participación de los pacientes (Irazoki et al. (2017)).

### 1.1.3. Historia de vida

Las Historias de Vida son registros detallados de los aspectos más relevantes de la vida de un paciente, o de personas significativas para él. Son especialmente importantes en las primeras fases del Alzheimer, cuando la memoria aún es relativamente intacta. Estas historias proporcionan dignidad al paciente y permiten a quienes lo rodean conocer mejor su identidad, a medida que las pérdidas de memoria comienzan a ser significativas.

La creación de estas historias es vital para preservar la identidad del paciente. Pueden incluir detalles sobre la familia, la carrera profesional, los viajes y otros aspectos importantes de la vida del individuo. La elaboración puede realizarse de diversas formas, como escribir un libro, crear collages de fotos, producir una película o usar una “caja de memoria”.

Es fundamental que estas historias reflejen la perspectiva personal del paciente, incluyendo emociones, sentimientos e interpretaciones. No se trata simplemente de relatar hechos cronológicos, sino de capturar la esencia única de la persona.

Un terapeuta suele encargarse de recopilar y estructurar los eventos importantes de la vida del paciente para crear la Historia de Vida. La estructura puede variar según las necesidades del paciente y los objetivos terapéuticos.

La participación de familiares o conocidos puede facilitar la evocación de recuerdos y enriquecer la Historia de Vida. Además, fortalece la comunicación entre el paciente y sus seres queridos, facilitando el proceso terapéutico.

## 1.2. Objetivos

Este trabajo busca desarrollar un chatbot que permita a los terapeutas aplicar terapia de reminiscencia a sus pacientes. En concreto, se centra en la primera etapa de búsqueda de la información con el objetivo de, más adelante generar las historias de vida y aplicar la terapia.

Para cumplir este objetivo general, se abordan los siguientes objetivos específicos:

- Desarrollar un primer chatbot básico capaz de realizar preguntas predefinidas y almacenar las respuestas de forma eficiente.
- Estudiar las técnicas modernas de procesamiento del lenguaje como los LLMs, bibliotecas como NLTK y spaCy. Además, de las diferentes APIs de para el desarrollo de chatbots.
- Mejorar el primer chatbot haciendolo más inteligente y capaz de analizar las respuestas, identificar la información omitida y hacer preguntas específicas para obtener la información faltante.
- Hacer una versión final del chatbot que sea capaz de analizar las repuestas y sepa tirar del hilo. También que sea capaz de generar preguntas adecuadas.
- Generar una interfaz sencilla de usar para que haga del chatbot una herramienta útil para el propósito para el que ha sido desarrollada.
- Ordenar la información obtenida para que sea fácil de analizar, entender y procesar, con la intención de generar historias de vida a partir de ella.

Para llevar el control de versiones utilizaremos el repositorio de github:  
<https://github.com/NILGroup/TFG-2324-ChatbotCANTOR>.

## 1.3. Estructura de la memoria

El presente documento esta formado por 5 capítulos, incluyendo el presente capítulo. Estos se organizan como sigue.

- **Capítulo 1: Introducción.** El capítulo presente, se presenta el proyecto, motivación, objetivos y estructura del mismo.
- **Capítulo 2: Estado de la cuestión.** Es segundo capítulo, nos muestra una aproximación a las historias de vida, la enfermedad del Alzheimer y la terapia de reminiscencia. También muestra otros trabajos relacionados, contexto y precedentes del proyecto.

- **Capítulo 3: Tecnologías utilizadas** En este tercer capítulo se explica el estudio profundo de las herramientas de procesamiento del lenguaje, interfaces y almacenamiento de la información que se llevo a cabo para decidir cómo implementar el chatbot. Así se explican las diferentes opciones y los motivos que llevaron a elegir cada una de ellas.
- **Capítulo 4: Desarrollo de prototipos** Este capítulo cuenta cómo se ha ido desarrollando el proyecto, explicando cada una de las versiones que se han llevado a cabo. Se explica, desde la primera versión, los problemas que se han ido encontrando, las funcionalidades extra añadidas etc.
- **Capítulo 5: Diseño de un chatbot de ayuda a la terapia de reminiscencia** En el capítulo 5, se presenta la herramienta final con todas sus funcionalidades. El software y las herramientas utilizados, la guía para la instalación y puesta en marcha, la arquitectura del sistema final etc.
- **Capítulo 6: Conclusiones y trabajo futuro** En el capítulo 5, se presenta la interfaz final con todas sus funcionalidades. El software y las herramientas utilizados, la guía para la instalación y puesta en marcha, la arquitectura del sistema final etc.



# Introduction

The steady increase in life expectancy worldwide has resulted in an increasingly aging population. This fact has brought with it an increase in diseases associated with aging, such as Alzheimer's disease and other types of dementia. Research efforts in these fields have focused on developing treatments that can slow the progression of this disease. Memory plays a fundamental role in the formation of our identity through our experiences and experiences. Its loss can be devastating, as it profoundly affects the quality of life of the person and those around them.

Among the non-pharmacological strategies for the treatment of cognitive impairment and memory, reminiscence therapy stands out. This technique seeks to stimulate the patient's past memories, which can have significant benefits for their social, mental and emotional well-being.

On the other hand, artificial intelligence is increasingly present in our daily lives. It has a wide variety of uses and applications ranging from virtual assistants in our smartphones to recommendation systems in streaming platforms and e-commerce. In addition, artificial intelligence is used in medicine for more accurate diagnoses, in industry to automate manufacturing processes and even in autonomous vehicle driving. This omnipresence of artificial intelligence is transforming the way we do business.

The present project tries to find a way to apply artificial intelligence, and specifically language processing to reminiscence therapy. Specifically, it seeks to develop a conversational tool to support the performance of this type of therapy.

This chapter has three main objectives. First, to explain the current motivation for this problem. Secondly, to make clear what are the objectives set as the starting point of the project. Finally, with the intention of giving a global vision of the project, in order to understand it and get an idea of what this report is like, there will be a final section in which the structure of this document will be explained, emphasizing what can be found in each chapter.

## 1.4. Motivation

Despite advances in medical research, a definitive cure for Alzheimer's disease has not yet been found. Most treatments focus on alleviating symptoms and slowing the progression of the disease. Among these treatments, reminiscence therapy has emerged as a prominent non-pharmacological option.

This therapy focuses on stimulating the patient's memories of the past, which can have significant benefits on their social, mental and emotional well-being. It helps people remember and share past experiences, which can be comforting and stimulating, especially for those facing the challenge of memory loss associated with Alzheimer's.

During the 2023-2024 academic term, the *chatbot yayo* application was developed with the goal of making it easier for therapists to conduct reminiscence-based therapies by simplifying and streamlining the process. This development starts from scratch and seeks to create a functional and useful chatbot, capable of maintaining a useful conversation with the patient to extract the necessary information.

This Final Degree Project aims to assist therapists, relatives or friends of patients with dementia in obtaining the necessary material to carry out reminiscence therapies. It seeks to improve the effectiveness of these therapies and, consequently, the quality of life of both patients and their families.

### 1.4.1. Alzheimer's disease

Alzheimer's disease (AD) is the leading cause of dementia in older adults. AD is a complex disease that is determined by multiple factors. It is sometimes hereditary and is characterized by the loss of neurons and synapses, along with the presence of senile plaques and neurofibrillary degeneration. Clinically, it manifests as progressive dementia, beginning with failure of recent memory and progressing to total dependence of the patient.

According to Donoso (2003), the incidence of AD increases with age, being rare before age 50 and affecting 1-2% of subjects at age 60, 3-5% at age 70, 15-20% at age 80 and up to half of those over age 85. It is more frequent in women, possibly due to greater longevity.

The disease develops in different stages. In the early stage, symptoms may go unnoticed or be attributed to simple carelessness. The affected person may experience difficulty remembering names, recent events or finding the right words in conversations. Despite these challenges, they usually retain the ability to perform everyday tasks with some independence. However, they may begin to lose interest in previously enjoyed activities.

As the disease progresses, the symptoms become more evident and troublesome. Memory loss becomes more pronounced, with difficulty recognizing family members and close friends. In addition, problems with orientation in time and space may arise,

which can result in disorientation even in familiar surroundings. Communication skills are also affected, with difficulty following conversations or expressing thoughts coherently.

Finally, Alzheimer's reaches its most devastating point in the last stage. Memory loss is profound and complete, with an inability to recall even recent events or recognize familiar faces. The affected person may experience significant changes in personality and behavior, becoming agitated, anxious or even aggressive at times. The ability to perform basic activities of daily living, such as dressing or feeding oneself, is severely compromised, and constant supervision becomes essential.

Pharmacological treatment of Alzheimer's disease (AD) is based on the use of drugs to improve cognitive defects and correct behavioral disorders. The most valued drugs are acetylcholinesterase inhibitors, which attempt to compensate for the loss of cholinergic neurons in the cerebral cortex. Treatment also includes drugs to treat behavioral disorders, such as antidepressants, tranquilizers or sleep inducers (Chung y Cummings (2000)).

In addition to pharmacological treatment, physical and mental activity is essential to stimulate brain activity and prevent memory loss. Family counseling is essential to help relatives manage behavioral disorders and increase the quality of life of AD patients. An example of nonpharmacological therapy that helps maintain mental activity is reminiscence therapy.

### 1.4.2. Reminiscence Therapy

Reminiscence, as defined by CEAFA (Spanish Confederation of Associations of Relatives of People with Alzheimer's disease and other dementias), is a technique that seeks to evoke memories in people, especially those related to important events in their lives.

In reminiscence therapies, various materials are used, such as photographs, videos, newspaper articles, audios and significant objects, with the aim of stimulating recall and activating memory through the emotions that these elements awaken in the patient.

In addition to stimulating the five senses to elicit recall, storytelling familiar to the patient is also used. Therefore, it is crucial to know the affected individual's past experiences in order to adapt the materials used in the therapies. Involving people close to the patient, such as family members and caregivers, in reminiscence exercises can significantly improve the patient's quality of life.

It is advisable to construct narratives based on the patient's life stories for these memory exercises. Reminiscence therapies are seen as part of a broader process in which the individual attempts to retrieve and link memories spanning much of his or her life.

A review of the possible effects of reminiscence therapy on people with dementia and their caregivers concluded that it decreased behavioral disturbances and

depressive symptoms, and improved cognition (Huang et al. (2015)). Another review noted improvements in mood, cognitive ability, social behavior, and general well-being (Cotelli et al. (2012)), and reminiscence therapy was recently found to improve physical health and to increased patient engagement (Irazoki et al. (2017)).

### 1.4.3. Life History

Life Histories are detailed records of the most relevant aspects of a patient's life, or of significant individuals to them. They are particularly important in the early stages of Alzheimer's disease, when memory is still relatively intact. These histories provide dignity to the patient and allow those around them to better understand their identity as memory losses begin to become significant.

Creating these histories is vital for preserving the patient's identity. They may include details about family, career, travels, and other important aspects of the individual's life. The creation can take various forms, such as writing a book, creating photo collages, producing a film, or using a "memory box."

It is essential that these histories reflect the patient's personal perspective, including emotions, feelings, and interpretations. It is not simply about recounting chronological facts but capturing the unique essence of the person.

A therapist typically gathers and structures the important life events of the patient to create the Life History. The structure may vary based on the patient's needs and therapeutic goals.

Involvement of family members or acquaintances can facilitate memory recall and enrich the Life History. Additionally, it strengthens communication between the patient and their loved ones, facilitating the therapeutic process.

## 1.5. Objectives

This work aims to develop a chatbot that allows therapists to apply reminiscence therapy to their patients. Specifically, it focuses on the initial stage of information gathering to later generate life histories and apply therapy.

To achieve this general objective, the following specific objectives are addressed:

- Develop an initial basic chatbot capable of asking predefined questions and efficiently storing responses.
- Study modern natural language processing techniques such as Large Language Models (LLMs), libraries like NLTK and spaCy, and various APIs for chatbot development.
- Enhance the initial chatbot to be more intelligent, capable of analyzing responses, identifying omitted information, and asking specific questions to obtain missing information.

- Create a final version of the chatbot that can analyze responses and engage in conversation. Ensure it can generate appropriate questions.
- Design a user-friendly interface to make the chatbot a useful tool for its intended purpose.
- Organize the gathered information for easy analysis, understanding, and processing, with the intention of generating life histories from it.

For version control, we will use the GitHub repository:  
<https://github.com/NILGroup/TFG-2324-ChatbotCANTOR>.

## 1.6. Document Structure

This document consists of 5 chapters, including this introductory chapter. They are organized as follows:

- **Chapter 1: Introduction.** This chapter introduces the project, motivation, objectives, and structure.
- **Chapter 2: State of the Art.** The second chapter provides an overview of life histories, Alzheimer’s disease, and reminiscence therapy. It also discusses related works, project context, and precedents.
- **Chapter 3: Technologies Used.** This third chapter explains the in-depth study of language processing tools, interfaces, and data storage that was conducted to decide how to implement the chatbot. It details the different options and the reasons behind each choice.
- **Chapter 4: Prototyping Development.** This chapter describes the project’s development process, explaining each of the versions that were carried out. It discusses the encountered problems, added functionalities, etc.
- **Chapter 5: Design of a Reminiscence Therapy Chatbot.** Chapter 5 presents the final tool with all its functionalities. It covers the software and tools used, installation and startup guide, final system architecture, etc.
- **Chapter 6: Conclusions and Future Work.** This chapter summarizes the project’s outcomes and outlines potential future directions.



## Capítulo 2

### Estado de la Cuestión

Este capítulo muestra la perspectiva teórica de la investigación llevada a cabo como trabajo previo al desarrollo del código. El conocimiento que se presenta en las siguientes páginas es necesario para entender cuál era el contexto del problema y el porqué de las decisiones que se han tomado para la construcción de la solución.

La estructura del capítulo muestra, en orden temporal, las herramientas del procesamiento de lenguaje que se han ido popularizando, desde las alternativas históricas hasta la situación actual. De esta forma, nos permite entender su importancia y conocer otras alternativas de implementación.

Finalmente, este capítulo muestra trabajos relacionados, ya sean puntos de partida para el trabajo actual, trabajos que complementan al presente proyecto, o incluso otros trabajos que cuyo estudio ha servido como herramienta de aprendizaje de cara a preparar el *chatbot*.

En conclusión, se pretende dar una perspectiva global de la situación actual en la que se encuentra el procesamiento del lenguaje en general y el desarrollo del *chatbot* en particular.

## 2.1. Evolución del Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (PLN) ha experimentado una evolución notable, impulsada por avances tecnológicos como la inteligencia artificial (IA). La integración de técnicas de IA en el PLN dió lugar a la rama del procesamiento natural del lenguaje (PLN), que supuso todo un hito en el ámbito del lenguaje escrito.

Inicialmente, los algoritmos de PLN se basaban en reglas, pero con el tiempo se adoptaron modelos de clasificación supervisada. Sin embargo, estos modelos enfrentaban limitaciones al no capturar el contexto completo de las palabras en una frase. Tres avances clave marcaron el camino hacia una PNL más avanzada.

Primero, el surgimiento de los modelos *word embeddings* descritos en la sección 2.2, desarrollados en 2013 que permiten representar palabras en un espacio vectorial considerando su contexto, lo que facilita la comprensión de sinónimos y relaciones entre palabras.

En segundo lugar, la arquitectura de redes neuronales profundas conocida como *transformers*, descritos en la sección 2.3, revolucionó el campo al capturar el contexto completo de un texto mediante una matriz de atención. Además, los modelos basados en *transformers* permiten la transferencia de aprendizaje, lo que facilita la adaptación a diversas aplicaciones.

Finalmente, el surgimiento de modelos generativos de lenguaje multipropósito de gran tamaño, como el archiconocido ChatGPT, ha llevado la PNL a nuevos horizontes. Estos modelos, que se describen en la sección 2.4 con cientos de millones de parámetros, pueden generar texto de calidad comparable a la humana y están generando debates sobre su impacto en diversos ámbitos.

A pesar de estos avances, persisten desafíos en el PLN, como la necesidad de bases de datos específicas y validadas para el entrenamiento de modelos especializados. Los investigadores son llamados a trabajar en la construcción de nuevas bases de datos y en el desarrollo de esquemas de preprocesamiento y ajuste, con el objetivo de impulsar soluciones a problemas específicos a nivel global.

## 2.2. Word embeddings

Los *word embeddings* (Mikolov et al., 2023) son una técnica importante en el procesamiento del lenguaje natural que consiste en representar palabras y documentos como vectores numéricos en un espacio de dimensiones reales. Este enfoque permite que palabras con significados similares tengan representaciones vectoriales similares, lo que facilita que las computadoras comprendan el contenido basado en texto de manera más efectiva.

En los *word embeddings*, las palabras se representan como vectores numéricos en un espacio dimensional reducido, lo que permite capturar información semánti-



ca y sintáctica entre palabras. Estos vectores se utilizan como características para alimentar modelos de aprendizaje automático, lo que permite trabajar con datos de texto y preservar la información semántica y sintáctica. Existen diferentes técnicas para tratar los *word embeddings*.

### 2.2.1. TF-IDF

La frecuencia de término-inversa de frecuencia de documento (TF-IDF) es un algoritmo de aprendizaje automático que se utiliza para la incrustación de palabras en texto. Consta de dos métricas: frecuencia de término (TF) y la inversa de la frecuencia de documento (IDF).

Este algoritmo trabaja en una medida estadística para encontrar la relevancia de las palabras en el texto, que puede estar en forma de un solo documento o varios documentos referidos como corpus.

$\mathbf{tf-idf}_{i,j}$  = Frecuencia del término  $i$  en el documento  $j$   $\times$  Frecuencia inversa de documentos del término  $i$

El puntaje de frecuencia de término (TF) mide la frecuencia de las palabras en un documento particular. En otras palabras, cuenta la ocurrencia de palabras en los documentos.

$$\mathbf{tf}_{i,j} = \frac{\text{Frecuencia del término } i \text{ en el documento } j}{\text{Número total de términos en } j}$$

La inversa de la frecuencia de documento (IDF) mide la rareza de las palabras en el texto. Se le otorga más importancia que el puntaje de frecuencia de término porque, aunque el puntaje de TF otorga más peso a las palabras que ocurren con frecuencia, el puntaje de IDF se centra en las palabras raramente utilizadas en el corpus que pueden contener información significativa.

$$\mathbf{idf}_i = \log \left( \frac{\text{Número total de documentos}}{\text{Número de documentos que contienen el término } i} \right)$$

El algoritmo TF-IDF se utiliza en tareas básicas de procesamiento de lenguaje natural y aprendizaje automático, como la recuperación de información, eliminación de palabras vacías, extracción de palabras clave y análisis de texto. Sin embargo, no captura eficientemente el significado semántico de las palabras en una secuencia.

### 2.2.2. Bolsa de palabras

El Bag of Words (BoW) es una técnica de procesamiento de texto ampliamente utilizada en el procesamiento del lenguaje natural (PLN) y la minería de texto. Esta técnica consiste en representar un documento de texto como un conjunto de palabras, ignorando el orden y la estructura gramatical.

En el modelo BoW, se crea un vocabulario de todas las palabras únicas en un conjunto de documentos de texto. Cada documento se representa como un vector de tamaño igual al tamaño del vocabulario, donde cada posición en el vector indica la frecuencia de una palabra en el documento.

Por ejemplo, si el vocabulario contiene las palabras "gato", "perro" y "juguete", y un documento tiene una frecuencia de dos para "gato" y tres para "perro", su vector de representación BoW sería [2,3,0].

La técnica BoW es útil para la clasificación y agrupación de documentos basados en su contenido textual. Se utiliza en aplicaciones como análisis de sentimientos, clasificación de texto y recomendación de contenidos.

El proceso de creación de la matriz BoW se lleva a cabo en varios pasos:

1. **Creación del vocabulario:** Se genera un conjunto de palabras únicas a partir de todos los documentos de texto.
2. **Vectorización del texto:** Cada documento se convierte en un vector de tamaño igual al vocabulario, donde las palabras presentes en el documento tienen un valor de uno, y las ausentes tienen un valor de cero.
3. **Normalización del peso:** Los vectores se normalizan dividiendo cada valor por la suma total de valores en el vector.

Una vez creada la matriz BoW, se puede utilizar en diversas aplicaciones de inteligencia artificial, como la clasificación de textos, el análisis de sentimientos y la agrupación de documentos.

### 2.2.3. Word2Vec

El método Word2Vec, desarrollado por Google en 2013, revolucionó el campo del procesamiento del lenguaje natural (PLN) al ofrecer una forma innovadora de entrenar incrustaciones de palabras. Este enfoque se basa en una idea distributiva que utiliza skip-grams o una técnica llamada Bolsa Continua de Palabras (CBOW).

En esencia, Word2Vec emplea redes neuronales poco profundas con capas de entrada, salida y proyección. Estas redes tienen como objetivo reconstruir el contexto lingüístico de las palabras, considerando tanto su orden en el texto como su contexto futuro.

El proceso implica iterar sobre un corpus de texto para aprender las asociaciones entre palabras. Se fundamenta en la premisa de que las palabras que aparecen juntas en un texto tienen una similitud semántica entre ellas. Esto permite asignar representaciones vectoriales a las palabras que son cercanas geométricamente en el espacio vectorial.

La similitud del coseno se utiliza como métrica para medir cuán similares son dos palabras o documentos en su significado. Si el ángulo entre los vectores de palabras es pequeño, la similitud del coseno es alta, lo que indica que las palabras tienen

significados similares. Por el contrario, si el ángulo es de 90 grados, la similitud es baja, lo que indica que las palabras son independientes en su contexto.

## 2.3. Transformers

Los modelos pre-entrenados son modelos de aprendizaje profundo o *DeepLearning* que sirven para realizar diversas tareas de Procesamiento de Lenguaje. Son pre-entrenados bajo grandes conjuntos de datos, lo que les permite ajustarse a tareas específicas sin requerir un entrenamiento desde cero. Este pre-entrenamiento es clave para entender por qué son tan valiosos, ya que permiten construir un sistema de generación de lenguaje sin un gran esfuerzo computacional (normalmente los entrenamientos tardan semanas o meses incluso con los mejores computadores).

Dentro de los modelos pre-entrenados, toman especial importancia los *transformers* (Vaswani et al. (2023)). Desde el primer momento, revolucionaron el PLN al introducir mecanismos de atención auto-ajustable, paralelización eficiente, captura de contexto bidireccional y pre-entrenamiento masivo. Todo ello permite construir modelos más poderosos y efectivos.

### 2.3.1. Aprendizaje por transferencia

Los modelos pre-entrenados, frente a los modelos estadísticos u otros tipos de algoritmos de *Deep Learning*, suponen grandes ventajas debido al pre-entrenamiento bajo un gran conjunto de datos y un pequeño ajuste posterior a realizar para adaptarlo a la tarea que se desee resolver, exigiendo mucho menos coste computacional y requiriendo para ello menos tiempo y esfuerzo. Todo esto es posible gracias al aprendizaje por transferencia o *Transfer Learning*.

El *Transfer Learning* es fundamental en el desarrollo y la eficiencia de los modelos pre-entrenados. Aparecen como solución al paradigma del aprendizaje aislado que presentan algoritmos como el aprendizaje supervisado tradicional. Estos modelos pueden aprovechar el conocimiento adquirido previamente en grandes conjuntos de datos para adaptarse a nuevas tareas con un costo computacional y de tiempo significativamente menor que si se entrenaran desde cero. Este enfoque es especialmente valioso en campos como la visión artificial o el PLN.

En el ámbito de la visión artificial, los modelos pre-entrenados pueden ser adaptados para tareas específicas, como la clasificación de imágenes, mediante el aprendizaje por transferencia. Por ejemplo en Tu et al. (2018), el sistema utiliza redes neuronales convolucionales pre-entrenadas para reconocer perros en imágenes, aprovechando el conocimiento previo adquirido por el modelo.

En PLN, los modelos pre-entrenados también son esenciales para resolver diversas tareas, como la detección de noticias falsas (Slovikovskaya (2019)). Estos modelos pueden emplear el aprendizaje por transferencia para adaptarse a nuevas tareas, como la clasificación de texto, utilizando el conocimiento previo obtenido durante el



cluye LLM que son buenos para la comprensión del texto porque permiten que la información fluya en ambas direcciones del texto. En azul, podemos ver el grupo “solo decodificador” que incluye LLM que son buenos en la generación de texto porque la información solo fluye de izquierda a derecha del texto para generar nuevas palabras de manera eficiente y autorregresiva. Finalmente, hay un tipo codificador-decodificador (mostrado en verde) que combina ambos aspectos y se usa para tareas que requieren comprender una entrada y generar una salida, como la traducción.

Dentro de este último grupo encontramos la mayoría de los modelos que fueron considerados para el desarrollo de este trabajo como los distintos modelos de GPT, los de Google (Bard, que evolucionó a Gemini, LaMDA o PaLM).

### 2.4.1. BERT

Los modelos de lenguaje enmascarados, como los Masked Language Models, enmascaran un cierto porcentaje de palabras en una oración y se espera que el modelo prediga esas palabras en función del contexto restante Rothman (2022). Un ejemplo representativo de este enfoque es BERT.

BERT (*Bidirectional Encoder Representations from Transformers*) es un modelo de lenguaje enmascarado basado en la arquitectura Transformer Devlin et al. (2019). Desarrollado por Google en 2018, BERT ha demostrado un rendimiento sobresaliente en una variedad de tareas de procesamiento de lenguaje natural.

BERT se pre-entrena en dos grandes corpus de texto en inglés sin etiquetar: BookCorpus Zhu et al. (2015) y Wikipedia. Estos corpus garantizan una amplia cobertura de datos y una buena calidad para el entrenamiento del modelo.

Aunque inicialmente no estaba diseñado para la generación de texto, BERT ha sido adaptado para esta tarea, logrando mejoras significativas en comparación con modelos como GPT-2 Wang y Cho (2019).

Existen numerosas variantes de BERT, algunas pre-entrenadas en dominios específicos y otras con un ajuste fino para tareas específicas Rajasekharan (2019). Por ejemplo, Beto es la versión en español de BERT, entrenada en un corpus extenso en dicho idioma Cañete et al. (2020).

La innovación clave de BERT es su enfoque bidireccional, que permite al modelo comprender el contexto de una palabra en función de su entorno completo. A diferencia de modelos unidireccionales como GPT-2, BERT considera tanto el contexto anterior como el posterior a una palabra dada.

La arquitectura de BERT consiste en una pila de codificadores de transformadores, con un número variable de capas dependiendo de la versión del modelo. Este enfoque permite a BERT realizar múltiples tareas de procesamiento de lenguaje, incluyendo Modelado de Lenguaje Enmascarado y Predicción de la Siguiente Oración Devlin et al. (2019).

### 2.4.2. T5

T5, o Text-to-Text Transfer Transformer Raffel et al. (2020), es un marco unificado para el aprendizaje por transferencia en procesamiento de lenguaje natural (PLN). Este enfoque revolucionario convierte todos los problemas basados en texto en un formato de texto a texto, donde el modelo recibe texto como entrada y genera nuevo texto como salida. Inspirado en marcos anteriores para tareas de NLP, como la modelización de lenguaje o la extracción de fragmentos, T5 permite aplicar el mismo modelo, objetivo de entrenamiento, procedimiento de entrenamiento y proceso de decodificación a cada tarea considerada.

El objetivo principal de T5 es proporcionar una perspectiva exhaustiva sobre el estado actual del campo de la transferencia de aprendizaje para PLN. En lugar de proponer nuevos métodos, se enfoca en la exploración y comparación empírica de técnicas existentes. Además, para facilitar futuros trabajos en este campo, se liberan conjuntos de datos, modelos pre-entrenados y código fuente.

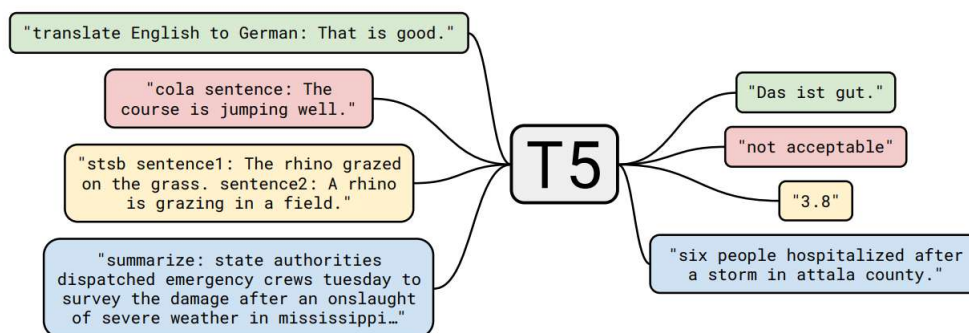


Figura 2.2: Diagrama del modelo T5 de Raffel et al. (2020)

Cómo se puede ver en la figura 2.2 el modelo T5 nos permite realizar diversas tareas como la traducción o la realización de resúmenes.

### 2.4.3. GPT *Generative Pretrained Transformer*

Los modelos de lenguaje casual o *Casual Language Models*, tienen como idea principal predecir una palabra o token enmascarado en una oración. Podemos ver una palabra enmascarada como un hueco en blanco en una frase. En este caso, el modelo solo consideraría el contexto anterior (palabras situadas a su izquierda) y dejaría de tener en cuenta el contexto posterior. El sentido de procesamiento (izquierda a derecha o derecha a izquierda) no es relevante mientras que se realice en un único sentido, teniendo en cuenta para la predicción las palabras pertenecientes a ese contexto y dejando de lado el resto de palabras. Así, la propiedad clave de este tipo de modelos es la naturaleza unidireccional en su predicción de las palabras enmascaradas y se verá reflejado en el esquema de entrenamiento (Rothman, 2022).

### 2.4.3.1. GPT-1

*Generative Pre-trained Transformer 1* (GPT-1) fue el primer modelo de lenguaje grande de OpenAI siguiendo la invención de la arquitectura transformer por parte de Google en 2017. En junio de 2018, OpenAI publicó un artículo titulado “Improving Language Understanding by Generative Pre-Training” Radford et al. (2018), en el cual presentaron ese modelo inicial junto con el concepto general de un *transformer* generativo pre-entrenado.

Hasta ese momento, los modelos neuronales de PLN con mejor rendimiento empleaban principalmente el aprendizaje supervisado a partir de grandes cantidades de datos etiquetados manualmente. Esta dependencia del aprendizaje supervisado limitaba su uso de conjuntos de datos que no estaban bien etiquetados, además de hacer que fuera prohibitivamente costoso y lento entrenar modelos extremadamente grandes; muchos idiomas (como el suajili o el criollo haitiano) son difíciles de traducir e interpretar utilizando tales modelos debido a la falta de texto disponible para la construcción de corpus. En contraste, el enfoque “semi-supervisado” de un GPT involucraba dos etapas: una etapa de pre-entrenamiento generativo no supervisado en la que se utilizaba un objetivo de modelado de lenguaje para establecer parámetros iniciales, y una etapa de ajuste fino discriminatorio supervisado en la que estos parámetros se adaptaban a una tarea objetivo.

El uso de una arquitectura transformer, en lugar de técnicas anteriores que involucraban RNNs con atención aumentada, proporcionó a los modelos GPT una memoria más estructurada de lo que se podría lograr a través de mecanismos recurrentes; esto resultó en un robusto rendimiento de transferencia en diversas tareas.

### 2.4.3.2. GPT-2

GPT-2 es uno de los modelos más representativos de los modelos de lenguaje casual que sigue una arquitectura de tipo Transformer basada en auto-atención (masked self-attention). Presentado por OpenAI en 2019, desde el primer momento fue considerado un gran éxito en el campo del Procesamiento de Lenguaje debido a sus más de 1.5 billones (americanos) de parámetros.

El objetivo de este sistema es construir una distribución de probabilidad en la que para cada palabra posible a generar se le asigna una probabilidad en función del contexto anterior. Este modelo fue pre-entrenado sobre un gran corpus de texto inglés siguiendo el entrenamiento auto-supervisado (self-supervised). El objetivo de este modelo es la predicción de la palabra siguiente dada una secuencia de palabras u oración.

Para pre-entrenar el modelo se creó un conjunto de datos, llamada WebText, formada a partir de la extracción de millones de páginas web obtenidas de enlaces de salida de páginas de Reddit que habían recibido una determinada puntuación mínima (para garantizar la calidad y significancia del enlace). Las páginas de Wikipedia relacionadas con estos enlaces se eliminaron. Es por esto por lo que GPT-2 no está entrenado bajo ningún texto de Wikipedia. El conjunto de datos resultante es un

enorme corpus de 40GB de textos preparado para el entrenamiento de este modelo, GPT-2 Radford et al. (2019).

La estructura de GPT-2 se asemeja mucho a la del modelo Transformer. Originalmente, este modelo constaba de un codificador y un decodificador, diseñado especialmente para tareas específicas como la traducción automática. Sin embargo, GPT-2 abandonó esta arquitectura convencional y optó por utilizar exclusivamente una serie de decodificadores del modelo Transformer. El número de decodificadores utilizados varía según el tamaño de GPT-2: la versión Small emplea solo doce, mientras que la versión Extra Large puede llegar a utilizar hasta cuarenta y ocho decodificadores. Esta configuración basada únicamente en decodificadores, característica distintiva de GPT-2, se ilustra en la figura 2.17.

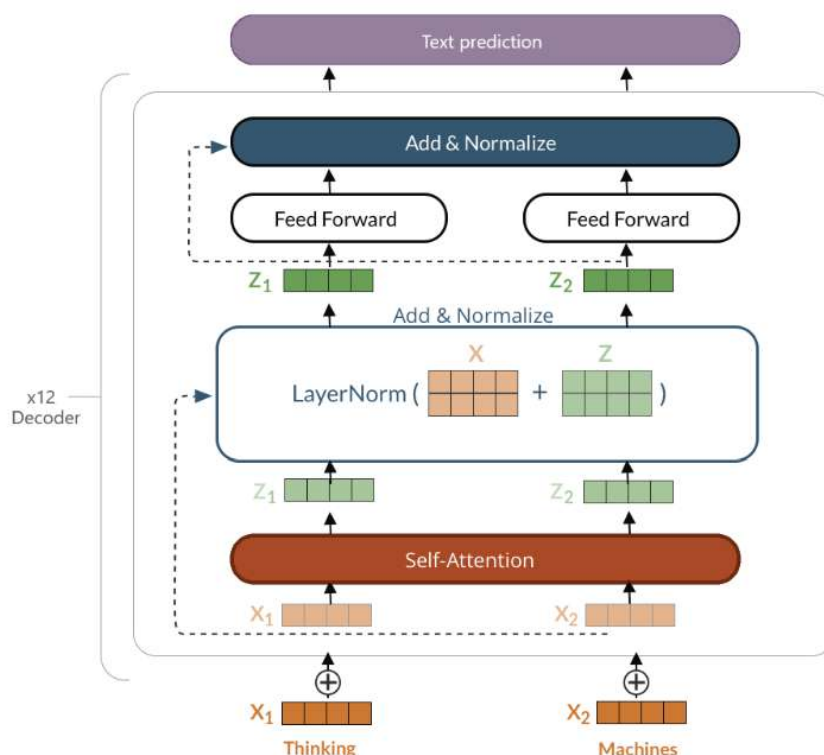


Figura 2.3: Arquitectura de decodificadores usada por GPT-2  
Salas (2020)

GPT-2 es un modelo unidireccional que utiliza auto-regresión para generar texto. Se centra en la secuencia anterior a la última palabra y no considera la secuencia posterior. Genera una nueva palabra basada en la secuencia actual y la añade a la entrada para el siguiente paso. Sin embargo, tiene limitaciones, como sesgos en la generación debido al conjunto de datos y al algoritmo de entrenamiento. Además, solo está disponible en inglés, aunque se puede usar Google Translate para otros idiomas. A pesar de esto, los modelos de lenguaje como GPT-2 son útiles para generar texto fluido y completo, prediciendo palabras de un corpus basándose en el contexto anterior.



### 2.4.3.3. GPT-3

GPT-3 aumentó significativamente la cantidad de parámetros en comparación con GPT-2, alcanzando hasta 175 mil millones de parámetros, lo que permitió una mayor complejidad y capacidad de aprendizaje. Debido a su mayor capacidad y tamaño de conjunto de datos, GPT-3 demostró una mejor capacidad para generalizar y comprender una variedad más amplia de contextos y tareas. Además, requiere menos entrenamiento adicional para tareas específicas en comparación con GPT-2.

Por otro lado, y a pesar de la mayor capacidad, GPT-3 logró reducir la generación de texto tóxico en comparación con GPT-2, aunque aún se requirieron estrategias de mitigación.

GPT-3 produjo textos con una mayor precisión y coherencia, lo que resultó en una calidad general de generación de texto más alta y una capacidad para realizar tareas más complejas.

### 2.4.3.4. GPT-4

GPT-4 según el informe técnico publicado por la propia compañía (Achiam et al., 2023), es el siguiente modelo de gran escala de OpenAI capaz de aceptar entradas de imagen y texto y producir salidas de texto. Aunque menos capaz que los humanos en muchos escenarios del mundo real, GPT-4 exhibe un rendimiento a nivel humano en diversos puntos de referencia profesionales y académicos, incluida la aprobación de un examen simulado de abogacía con una puntuación alrededor del 10 % superior de los examinados. GPT-4 es un modelo basado en *Transformer* 2.3 preentrenado para predecir el siguiente token en un documento. El proceso de alineación posterior al entrenamiento resulta en un mejor rendimiento en medidas de factualidad y adherencia al comportamiento deseado.

Una de las características principales de GPT-4 es su capacidad para procesar entradas de texto y de imagen y producir salidas de texto. Se evaluó en una variedad de exámenes diseñados originalmente para humanos, donde obtuvo resultados destacados, superando en muchos casos a la mayoría de los examinados humanos. Por ejemplo, en un examen simulado de abogacía, GPT-4 obtiene una puntuación que se encuentra en el 10 % superior de los examinados, en contraste con GPT-3.5, que se encuentra en el 1 % inferior.

En una serie de benchmarks tradicionales de NLP, GPT-4 supera tanto a modelos de lenguaje grandes anteriores como a la mayoría de los sistemas de vanguardia (que a menudo tienen entrenamiento específico para el benchmark o ingeniería manual). En el benchmark MMLU, que cubre 57 temas en inglés, GPT-4 no solo supera a los modelos existentes en inglés, sino que también demuestra un rendimiento sólido en otros idiomas.

A pesar de sus capacidades, GPT-4 tiene limitaciones similares a modelos anteriores, como la falta de fiabilidad completa, una ventana de contexto limitada y la incapacidad de aprender de la experiencia. Se debe tener cuidado al usar las salidas de GPT-4, especialmente en contextos donde la fiabilidad es importante. Las

capacidades y limitaciones de GPT-4 plantean desafíos de seguridad significativos y novedosos, y se considera que el estudio cuidadoso de estos desafíos es un área de investigación importante dada la posible repercusión societal.

#### 2.4.4. Llama

A diferencia de la creencia común de que más parámetros conducen a un mejor rendimiento, investigaciones recientes muestran que modelos más pequeños entrenados con más datos pueden superar a los modelos más grandes. Se ha desarrollado una serie de modelos llamados LLaMA, Touvron et al. (2023) que van desde 7B hasta 65B de parámetros, con un rendimiento competitivo en comparación con los mejores LLMs existentes.

Por ejemplo, LLaMA-13B supera a GPT-3 en la mayoría de las pruebas, a pesar de ser 10 veces más pequeño. Se espera que estos modelos democratizen el acceso y el estudio de los LLMs, ya que pueden ejecutarse en una sola GPU. Además, se asegura la compatibilidad con la fuente abierta al utilizar solo datos públicamente disponibles, a diferencia de otros modelos que dependen de datos no disponibles públicamente. El trabajo detalla las modificaciones realizadas en la arquitectura del *transformers* 2.3 y el método de entrenamiento, además de presentar el rendimiento de los modelos en comparación con otros LLMs en una serie de pruebas estándar. También se examinan los sesgos y la toxicidad codificados en los modelos, utilizando benchmarks de la comunidad de inteligencia artificial responsable.

El conjunto de datos de entrenamiento es una mezcla de varias fuentes, que cubren un conjunto diverso de dominios. Mayormente reutiliza fuentes de datos que se han utilizado para entrenar otros LLMs, con la restricción de utilizar solo datos públicamente disponibles y compatibles con la distribución abierta.

La tokenización de los datos se lleva a cabo con el algoritmo de codificación de bytes (BPE), utilizando la implementación de *SentencePiece*<sup>1</sup>. Dividimos todos los números en dígitos individuales y recurrimos a bytes para descomponer caracteres UTF-8 desconocidos. El tamaño total de nuestro conjunto de datos de entrenamiento contiene aproximadamente 1.4T de tokens después de la tokenización. La mayoría de los datos de entrenamiento se utilizan solo una vez durante el entrenamiento, con la excepción de los dominios de Wikipedia y Libros, sobre los cuales se realizan aproximadamente en dos épocas.

#### 2.4.5. LaMDA

LaMDA, *Language Model for Dialogue Applications* (Modelo de Lenguaje para Aplicaciones de Diálogo), es una familia de modelos de lenguaje neuronal conversacional desarrollados por Google.

---

<sup>1</sup>El algoritmo *SentencePiece* Kudo y Richardson (2018) utiliza un enfoque basado en subpalabras, donde construye un vocabulario de subpalabras que se adaptan a la frecuencia de aparición en el corpus de entrenamiento.

La primera generación se anunció en la Google I/O <sup>2</sup> de 2021 y fue presentada como un modelo de lenguaje conversacional. La segunda generación, presentada en el evento del año siguiente, introdujo mejoras y nuevas capacidades, como la generación de conversaciones originales sobre temas no enseñados previamente.

LaMDA llamó la atención cuando el ingeniero de Google, Blake Lemoine, afirmó que el chatbot se había vuelto sensible, lo que generó debates sobre la eficacia de la prueba de Turing para determinar la inteligencia artificial general.

Las habilidades conversacionales de LaMDA tardaron años en desarrollarse. Como muchos modelos de lenguaje recientes, incluidos BERT y GPT-3, se basa en *Transformer* 2.3. Esa arquitectura produce un modelo que se puede entrenar para leer muchas palabras (una oración o párrafo, por ejemplo), preste atención a cómo esas palabras se relacionan entre sí y luego prediga qué palabras cree que vendrán a continuación.

Pero a diferencia de la mayoría de los otros modelos de lenguaje, LaMDA fue entrenado en diálogo. Durante su formación, captó varios de los matices que distinguen la conversación abierta de otras formas de lenguaje. Uno de esos matices es la sensatez.

El modelo de Transformer que utiliza LaMDA es de solo decodificador y está pre-entrenado en un corpus de texto que incluye documentos y diálogos. Se entrenó con datos de ajuste fino y se probó en diferentes modelos con diferentes hiperparámetros, demostrando superar las respuestas humanas en áreas específicas.

## 2.5. Alucinaciones

En el campo de la inteligencia artificial (IA), se utiliza el término “alucinación” o “alucinación artificial” para describir respuestas de IA que no parecen estar justificadas por los datos de entrenamiento Edwards (2023). Este fenómeno, también conocido como confabulación o delirio, se refiere a la generación de respuestas o creencias que carecen de base sólida Ji et al. (2022). Por ejemplo, un chatbot alucinado podría ofrecer información falsa, como afirmar que los ingresos de Tesla fueron de 13.600 millones de dólares, un número aparentemente inventado Lin (2022). Otros ejemplos de alucinaciones serían la generación de respuestas inventadas como se puede ver en la imagen 2.4, donde ChatGPT genera un resumen de un artículo que no existe, o la adición de elementos en el estudio de imágenes, como se ve en la figura 2.5.

Es importante destacar que, aunque se usa el término “alucinación” por analogía con la psicología humana, la diferencia fundamental radica en que las alucinaciones de IA se relacionan con respuestas injustificadas más que con percepciones falsas. Algunos investigadores señalan que este término antropomorfiza de manera poco razonable a los ordenadores.

---

<sup>2</sup>Google I/O es una conferencia anual de desarrolladores realizada por Google en Mountain View, California



Figura 2.4: Ejemplo de alucinación de ChatGPT Wikipedia (Fecha de última modificación)

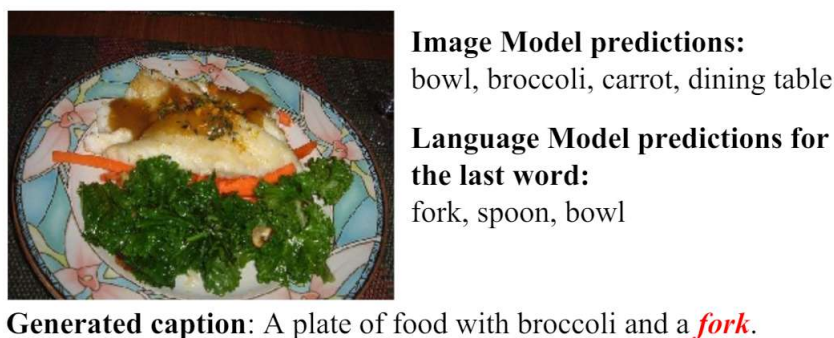


Figura 2.5: Ejemplo de alucinación en una imagen Rohrbach et al. (2023)

El problema de las alucinaciones de IA se volvió prominente alrededor de 2022 con la introducción de modelos grandes de lenguaje como ChatGPT Zhuo et al. (2023). Los usuarios expresaron preocupación por la capacidad de estos bots para insertar falsedades aparentemente plausibles en sus respuestas. En 2023, los analistas reconocieron las alucinaciones frecuentes como un desafío significativo en la tecnología de modelos de lenguaje Leswing (2023).

Se cuestiona la confiabilidad del contenido generado por inteligencia artificial en el ámbito científico Machin-Mastromatteo (2023). Según Spinak (2023), los modelos de lenguaje de IA pueden percibir patrones que son imperceptibles para los humanos, lo que resulta en resultados inesperados o incorrectos, fenómeno conocido como “alucinación” Spinak (2023). Estas alucinaciones pueden llevar a la producción de contenido falso, especialmente fuera de sus dominios específicos o al tratar con temas complejos o ambiguos, lo cual puede ser lingüísticamente plausible pero no científicamente preciso Sage (2023).

## 2.6. Otros trabajos relacionados

### 2.6.1. Proyecto Cantor

El proyecto CANTOR (Composición automática de narrativas personales como apoyo a terapia ocupacional basada en reminiscencia) en el que se enmarca este trabajo, desarrolla herramientas digitales utilizando tecnologías de Inteligencia Artificial para construir automáticamente historias de vida que puedan ser reexaminadas posteriormente como apoyo a las terapias ocupacionales de pacientes con demencias.

CANTOR está financiado por el Ministerio de Ciencia e Innovación, en colaboración entre académicos de la Universidad Complutense de Madrid y la Universidad de La Coruña. El objetivo de CANTOR es desarrollar herramientas que faciliten la terapia ocupacional basada en reminiscencia para mejorar la calidad de vida de pacientes con deterioro cognitivo.

En este ámbito se han elaborado varios Trabajos de Fin de Grado en la Facultad de Informática de la UCM. Paso a referir algunos de ellos relacionados con este trabajo.

#### 2.6.1.1. Generación de historias de vida usando técnicas de Deep Learning

En el curso 2021-2022, la compañera María Cristina Alameda Salas Salas (2020), en su trabajo de fin de grado, Generación de historias de vida usando técnicas de Deep Learning, desarrolló un sistema basados en técnicas de Deep Learning que de soporte a la generación de historias de vida. Partiendo de unos datos de entrada en forma de datos estructurados de tipo biográfico, ese trabajo permite la construcción de un sistema de generación de lenguaje natural, transformador de los datos de entrada a un escrito fluido y coherente, que abarque la representación de los datos de partida de manera completa, sin incorrecciones y lo más cercana posible a una redacción humana. Nuestro objetivo ahora sería el desarrollo de un programa que interactuara con el usuario y nos permitiera obtener toda esa información bibliográfica que da lugar a las historias de vida.

#### 2.6.1.2. Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning

En 2021, en la UCM se desarrollo un trabajo de extracción de preguntas a partir de imagenes con técnicas de Deep Learning Boto et al. (2021). Este proyecto ayuda a las personas con problemas de memoria a recordar aspectos de su vida utilizando técnicas de IA como redes convolucionales y recurrentes. Para lograrlo se desarrolló un sistema capaz de extraer preguntas de fotografías que puedan representar recuerdos para las personas con problemas de memoria utilizando un bot que simula una sesión de terapia de reminiscencia. El usuario ha de enviar fotografías al bot y este se encargará de enviarle, una a una, las preguntas generadas por la red neuronal.

En este momento, el usuario deberá recordar todo lo posible sobre la imagen para poder responder a las preguntas y conseguir ejercitar su memoria.

#### **2.6.1.3. Extracción de información personal a partir de redes sociales para la creación de un libro de vida**

Este proyecto, desarrollado por Aguilera Heredero y Molina Muñoz (2021), tiene como objetivo principal ayudar a terapeutas ocupacionales en el tratamiento de pacientes con problemas de memoria, especialmente aquellos relacionados con el deterioro cognitivo asociado con la edad. Se propone la creación de una herramienta para desarrollar un libro de vida, una técnica terapéutica conocida como terapia basada en reminiscencia.

La herramienta combina técnicas de extracción y tratamiento de datos de diferentes redes sociales proporcionadas por el paciente, almacenándolos en una base de datos SQL para obtener la información más relevante. Esta información se utilizará para crear el libro de vida, que se presentará en una interfaz web desarrollada con React. La interfaz permitirá visualizar fácilmente los datos recopilados, utilizando tablas, mapas, líneas de tiempo y galerías de fotos.

#### **2.6.1.4. Generación de historias a partir de una base de conocimiento**

En este proyecto, desarrollado por Magaz (2023), se construyó una aplicación para crear relaciones entre palabras e imágenes, partiendo de unas palabras determinadas. Con las relaciones establecidas la aplicación genera estadísticas a través de las cuales puede evaluarse el progreso del paciente. En cada sesión se trata un tema concreto, pudiéndose elegir el tipo de sesión entre Sesión palabras (se elige una categoría de palabras), Sesión Progreso (se visualiza el avance del paciente a través de estadísticas agrupadas por categorías) o Sesión imágenes (donde se asocia una imagen a un concepto y una categoría).

#### **2.6.1.5. Recuerdame 1.0**

Recuerdame 2.0 Barquilla Blanco et al. (2022) presenta la creación de una aplicación que facilite a los terapeutas la realización de terapias basadas en reminiscencia para tratar a pacientes con alzheimer, haciéndolas más ágiles y rápidas.

La aplicación creada es una aplicación web responsive, con una estructura Modelo Vista Controlador creada mediante lenguajes como HTML, CSS, PHP, JavaScript. La aplicación tiene una usabilidad aceptable, pero tiene detalles que mejorar y algunas funcionalidades que no pudieron ser desarrolladas por la falta de tiempo.

### 2.6.1.6. Recuerdame 2.0

Durante el curso 2022-2023, la aplicación recuerdame 1.0 fue mejorada dando lugar a recuerdame 2.0 Díez Sobrino et al. (2023), para ofrecer una experiencia terapéutica más enriquecedora a pacientes con problemas de memoria. Las mejoras incluyeron la optimización basada en la retroalimentación de usuarios finales, la narración mejorada de Historias de Vida, la generación automática de resúmenes, la integración de terapia con un bot, la mensajería entre terapeutas y cuidadores, y la capacidad de generar vídeos de Historias de Vida. Además, se realizó una evaluación exhaustiva del sistema en instituciones médicas y residencias de ancianos para validar su eficacia.

### 2.6.2. Celia

Celia,<sup>3</sup> es un chatbot impulsado por inteligencia artificial (IA) desarrollado por la compañía Atlantic, con el respaldo de la Xunta de Galicia en España. Este chatbot tiene como objetivo acompañar, entretener y brindar asistencia a las personas mayores y dependientes, y se destaca por su capacidad para detectar indicios y patrones de enfermedades neurodegenerativas, como el Alzheimer, mediante el análisis de la voz del usuario.

A diferencia de otros asistentes de conversación como Alexa o Siri, Celia va más allá al utilizar herramientas biométricas para medir y monitorear parámetros indicativos no solo de enfermedades neurológicas, sino también de condiciones emocionales como la ansiedad y la depresión.

Celia está disponible para su uso a través de tres plataformas: WhatsApp, la versión web y una aplicación oficial disponible actualmente solo para dispositivos Android. Los usuarios pueden interactuar con Celia a través de mensajes de texto o de voz, y la instalación es sencilla, lo que permite un acceso rápido y eficiente a este recurso tecnológico.

Una característica destacada de Celia es su capacidad para tomar la iniciativa en las conversaciones y proponer actividades sin necesidad de instrucciones. Además, ofrece la posibilidad de establecer recordatorios para citas médicas o la toma de medicamentos, brindando un apoyo integral en la gestión de la salud de los usuarios.

---

<sup>3</sup>Así es Celia, la inteligencia artificial para adultos mayores que puede detectar indicios de alzheimer





# Capítulo 3

## Tecnologías utilizadas

El capítulo actual se enfoca en detallar las tecnologías empleadas en la construcción y el despliegue del chatbot diseñado para asistir en terapias de reminiscencia. También se analizarán las herramientas y metodologías utilizadas en el proceso de desarrollo de prototipos, que se presenta en el capítulo 4, así como su integración con los conceptos y conocimientos presentados en el estado de la cuestión.

En cada sección del capítulo, se llevará a cabo un análisis de las diferentes alternativas consideradas para la construcción de cada uno de los módulos que componen el chatbot. Desde la evaluación de diversas API's y bibliotecas de Procesamiento del Lenguaje Natural (PLN) hasta la exploración de las múltiples opciones disponibles para el desarrollo de la interfaz de usuario.

Esta metodología permitirá proporcionar un contexto completo y comprensible que facilitará la comprensión de las decisiones tomadas a lo largo del desarrollo del proyecto, como se detalla en el capítulo 4.

## 3.1. Bibliotecas de Procesamiento del Lenguaje en Python

### 3.1.1. NLTK

La biblioteca NLTK (Natural Language Toolkit) <sup>1</sup> ofrece una amplia gama de herramientas y recursos para tareas de PLN.

En primer lugar, NLTK permite realizar tareas como la tokenización y el etiquetado POS (Part-Of-Speech tagging). Al utilizar las herramientas de tokenización, podemos dividir el texto en unidades más pequeñas, lo que facilita el análisis y la comprensión. El etiquetado POS asigna etiquetas gramaticales a cada palabra en el texto, lo que nos permite identificar la función de cada palabra en la oración.

```
sentence = "Reminiscence Therapy involves the  
discussion of past activities using prompts like  
photos."  
tokens = nltk.word_tokenize(sentence)  
tagged = nltk.pos_tag(tokens)  
tagged[0:len(tagged)]
```

Listing 3.1: Ejemplo de código en Python. Se pueden consultar más ejemplos en Bird et al. (2009)

En este código *nltk* tokeniza la oración introducida y etiqueta cada *token* indicando la categoría sintáctica de cada *token* como sigue:

- NNP: Nombre propio singular
- NN: Nombre, singular o sustantivo singular
- VBZ: Verbo tercera persona del singular presente
- DT: Determinante
- IN: Preposición o oración subordinada
- JJ: Adjetivo
- NNS: Nombre plural
- VBG: Verbo, gerundio o participio
- . : Signo de puntuación

---

<sup>1</sup>Página oficial de NLTK

```
>>>[('Reminiscence', 'NNP'), ('Therapy', 'NNP'), ('
involves', 'VBZ'), ('the', 'DT'), ('discussion', 'NN
'), ('of', 'IN'), ('past', 'JJ'), ('activities', '
NNS'), ('using', 'VBG'), ('prompts', 'NNS'), ('like
', 'IN'), ('photos', 'NNS'), ('.', '.')]

```

Listing 3.2: Tokenización y etiquetado con nltk

Otras de las funcionalidades que nos permite esta biblioteca es el análisis sintáctico o lematización. Por ejemplo, nos permite la obtención de árboles sintácticos, lo que permite visualizar la estructura gramatical de las oraciones, facilita el análisis y la interpretación del texto.

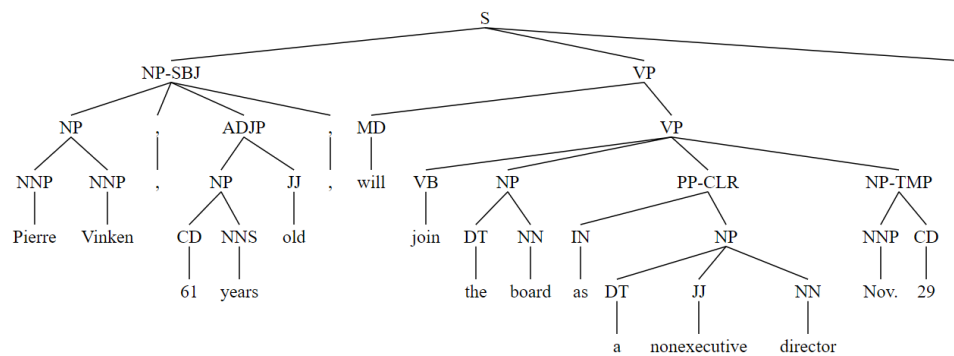


Figura 3.1: Árbol sintáctico generado con nltk

```
entities = nltk.chunk.ne_chunk(tagged)
nltk.download('treebank')
from nltk.corpus import treebank
t = treebank.parsed_sents('wsj_0001.mrg')[0]
t

```

Listing 3.3: Análisis sintáctico y lematización con nltk

Además de estas características fundamentales, NLTK ofrece una serie de otras funcionalidades que amplían aún más su utilidad. Por ejemplo, incluye herramientas para la extracción de entidades nombradas, el análisis de sentimientos, la generación de texto y la traducción automática. Estas capacidades adicionales permiten abordar una amplia variedad de tareas en el procesamiento del lenguaje natural, desde la clasificación de texto hasta la generación de resúmenes automáticos y la traducción de idiomas. En resumen, NLTK es una herramienta invaluable para investigadores, estudiantes y profesionales que trabajan en el campo del PLN, ofreciendo una amplia gama de funcionalidades que facilitan el análisis, la comprensión y la manipulación del lenguaje humano.

### 3.1.2. SpaCy

SpaCy <sup>2</sup> ofrece soporte para más de 25 idiomas y cuenta con 84 pipelines de entrenamiento. Utiliza el aprendizaje multi-tarea con modelos preentrenados como BERT, lo que permite un rendimiento avanzado en tareas de procesamiento del lenguaje natural. Sus componentes incluyen herramientas para el reconocimiento de entidades nombradas, etiquetado de partes del discurso, análisis de dependencias, segmentación de oraciones, clasificación de texto, lematización, análisis morfológico, vinculación de entidades y más.

```
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("Reminiscence Therapy involves the discussion
        of past activities using prompts like photos.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.
                        noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if
                 token.pos_ == "VERB"])
```

Listing 3.4: Ejemplo de tokenización usando spaCy

Este código carga el modelo preentrenado "en\_core\_web\_sm" de spaCy, que incluye herramientas para tokenizar, etiquetar, analizar la sintaxis y reconocer entidades nombradas en textos en inglés. Luego, procesa el texto proporcionado y muestra las frases nominales identificadas utilizando la función *noun\_chunks* y los verbos lematizados utilizando la propiedad *lemma\_*. Este análisis gramatical permite identificar las partes clave del texto, como los sustantivos y las acciones descritas.

```
>>> Noun phrases: ['Reminiscence Therapy', 'the
                    discussion', 'past activities', 'prompts', 'photos'
                    ]
Verbs: ['involve', 'use']
```

Listing 3.5: Resultado de tokenización usando spaCy

Además, es fácilmente ampliable con componentes y atributos personalizados, y es compatible con modelos personalizados en PyTorch, TensorFlow y otros frameworks. SpaCy ofrece visualizadores integrados para la sintaxis y el reconocimiento de entidades nombradas, y facilita el empaquetado, despliegue y gestión de flujos de

---

<sup>2</sup>spaCy

trabajo de modelos. Con su precisión rigurosamente evaluada y su robustez, Spacy es una herramienta poderosa y versátil para el procesamiento del lenguaje natural.

La biblioteca spaCy cuenta con diferentes componentes que interactúan entre sí escuchando la salida unos de otros para mejorar su procesamiento (*listener*). Además, existen una serie de reglas y dependencias entre los componentes. Por ejemplo, el módulo *attribute\_ruler* indica proporciona reglas de etiquetado a *tagger*.

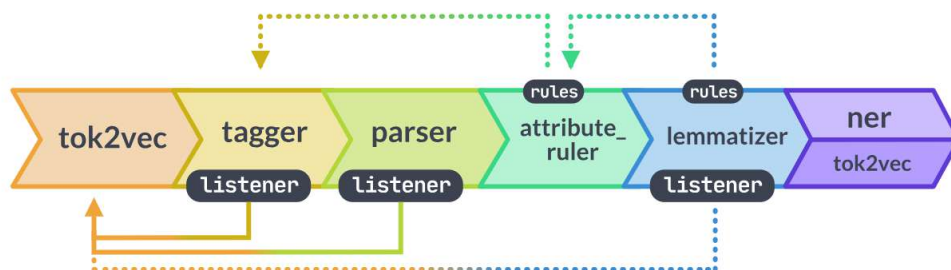


Figura 3.2: Árbol sintáctico generado con nltk

El gráfico representa la estructura de una tubería de procesamiento de lenguaje natural (NLP) en spaCy, mostrando la secuencia de componentes y sus interacciones.

- **tok2vec**: Este componente convierte los tokens en vectores de palabras, que capturan el significado semántico de las palabras en el contexto de la oración.
- **tagger**: El *tagger* asigna etiquetas gramaticales a cada token en el texto, como partes del discurso (POS).
- **parser**: El analizador sintáctico analiza la estructura sintáctica del texto, identificando las relaciones de dependencia entre las palabras.
- **attribute\_ruler**: Este componente aplica reglas para agregar atributos adicionales a los tokens, como excepciones de lema y POS, y manejar espacios en blanco de manera coherente.
- **lemmatizer**: El lematizador determina la forma base de cada palabra (su lema) en función de su contexto y su parte del discurso.
- **ner/tok2vec**: El componente de reconocimiento de entidades (NER) identifica entidades nombradas en el texto, como nombres de personas, lugares o organizaciones. En algunos modelos, este componente comparte la representación de vectores de palabras (**tok2vec**) con otros componentes para mejorar la coherencia y la precisión de las predicciones.

En resumen, este gráfico muestra cómo los componentes de spaCy interactúan entre sí para realizar tareas de procesamiento de lenguaje natural, aprovechando la información compartida y las reglas definidas para mejorar la precisión y la coherencia del análisis lingüístico.

## 3.2. APIs de procesamiento del lenguaje

Las APIs de procesamiento del lenguaje son conjuntos de herramientas y servicios que integran múltiples funcionalidades relacionadas con el PLN en sus aplicaciones y sistemas. Es decir, son herramientas que aúnan y ofrecen funcionalidades como el análisis de sentimientos, el reconocimiento de entidades o la tokenización. Frente a las bibliotecas y modelos presentados anteriormente presentan la ventaja de que se pueden usar sin necesidad de desarrollar desde cero algoritmos o modelos, lo que facilita su uso. Estas características hacen que este tipo de APIs sean comúnmente usadas en variedad de aplicaciones, desde chatbots hasta sistemas de recomendación.

### 3.2.1. Bard

Bard es una API de procesamiento del lenguaje natural desarrollada por Google con el objetivo de ofrecer respuestas conversacionales coherentes y relevantes a través de interacciones de mensajes. Basada en LaMDA, un modelo de lenguaje experimental de Google, Bard compite directamente con ChatGPT en el campo del procesamiento del lenguaje natural, permitiendo realizar consultas y recibir respuestas sin necesidad de navegar por diferentes páginas web.

Google ha priorizado la accesibilidad y la transparencia en el desarrollo de Bard, ofreciendo modelos y recursos de PLN de código abierto que pueden ser utilizados y modificados por la comunidad. Inicialmente lanzado con un modelo reducido de LaMDA 2.4.5, Bard busca ampliar su alcance y obtener comentarios para su mejora continua.

```
prompt = "A partir del texto a continuación, que contiene información  
sobre una persona y damelo en una lista info donde  
info[nombre]:valor_atributo."  
  
texto = "Hola mi nombre es Marta, tengo 22 años y soy de Zaragoza"  
>> **Info[nombre:Marta;edad:22;ciudad:Zaragoza]**  
¿Hay algo más que pueda hacer por ti?
```

Figura 3.3: Ejemplo de uso de BARD.

Cómo se puede ver en el ejemplo, Bard es capaz de analizar la información proporcionada y generar respuestas coherentes y formateadas según las especificaciones dadas. Gracias a su capacidad para comprender el contexto y generar texto de manera precisa, Bard es una herramienta valiosa para tareas que requieren PLN, como la generación de respuestas conversacionales. Todo ello lo convierte en una opción ideal para una amplia gama de aplicaciones, desde chatbots hasta sistemas de asistencia virtual.

Sin embargo, Bard ya no está disponible. En diciembre de 2023, Google fortaleció la capacidad de Bard al incorporar Gemini Pro en inglés, brindando habilidades más

avanzadas de comprensión, razonamiento, resumen y codificación. Más adelante, en febrero de 2024, se anunció la expansión de Gemini Pro a más de 40 idiomas y se oficializó el cambio de nombre de Bard a Gemini, lo que implicó descartar el primer modelo del proyecto desarrollado en Bard debido a la indisponibilidad de Gemini en España en ese momento.

### 3.2.2. Gemma

Gemma es una API de PLN desarrollada por OpenAI. Utiliza modelos de lenguaje basados en la arquitectura GPT (Generative Pre-trained Transformer) para una variedad de tareas de PLN, como generación de texto, análisis de sentimientos, clasificación de texto, y más. Gemma se destaca por su capacidad para generar texto coherente y de alta calidad en una variedad de estilos y tonos, así como por su facilidad de uso y su API intuitiva.

Una de las principales ventajas de Gemma es su rendimiento en tareas de generación de texto, donde ha establecido nuevos estándares de calidad y coherencia en muchos casos. Además, Gemma ofrece modelos pre-entrenados en varios dominios y lenguas, lo que facilita su integración en una variedad de aplicaciones de PLN. Sin embargo, debido a su enfoque en modelos de última generación, Gemma puede requerir recursos computacionales significativos y puede ser más difícil de entender y utilizar para usuarios principiantes en PLN.

En primer lugar, estos modelos se trabajaron en Google Collaborate aumentando el número de GPUs. De esta forma gemma tiene un buen comportamiento y genera respuestas adecuadas y coherentes. En concreto, una respuesta generada por *gemma-7b* sería la siguiente. Sin embargo, las limitaciones propias de Google Collaborate no

```
prompt =gemma_lm.generate("What is the meaning of life?",
max_lenght = 64)
>>> The question is one of the most important questions in the world.
It's the question that has been asked by philosophers, theologians and
scientist for centuries. And it's the question that has been asked by
people who are looking for answer to their own lives.
```

Figura 3.4: Ejemplo de uso de GEMMA.

permitían en la versión gratuita aumentar el número de GPUs de forma frecuente y en consecuencia tuve que estudiar el modelo en otro entorno. Para ejecutarlo de forma local y obtener un buen comportamiento es necesario instalar Linux y descargar el modelo de Hugging face.

Aunque *gemma-2b* en la versión local instalada desde Hugging Face en Linux tiene un buen comportamiento, genera respuestas incoherentes que hacen de este modelo poco útil para nuestro proyecto. La versión *gemma-7b* genera respuestas mucho mejores pero tiene la enorme desventaja de que ocupa una gran cantidad de espacio en memoria.

### 3.2.3. GPT API

GPT API es una API de PLN desarrollada por OpenAI. Utiliza modelos de lenguaje basados en la arquitectura GPT (Generative Pre-trained Transformer) para una variedad de tareas de PLN, como generación de texto, análisis de sentimientos, clasificación de texto, y más. GPT API se destaca por su capacidad para generar texto coherente y de alta calidad en una variedad de estilos y tonos, así como por su facilidad de uso y su API intuitiva.

Una de las principales ventajas de GPT API es su rendimiento en tareas de generación de texto, donde ha establecido nuevos estándares de calidad y coherencia en muchos casos. Además, GPT API ofrece modelos pre-entrenados en varios dominios y lenguas, lo que facilita su integración en una variedad de aplicaciones de PLN. Sin embargo, debido a su enfoque en modelos de última generación, GPT API puede requerir recursos computacionales significativos y puede ser más difícil de entender y utilizar para usuarios principiantes en PLN.

Sin embargo para obtener el comportamiento que se necesitaba en este trabajo debía ser entrenada, y debido a las limitaciones hardware esto suponía una cantidad de tiempo inviable.

### 3.2.4. Rasa

Entre las opciones que se barajaron para seguir desarrollando el proyecto se encuentra Rasa. Rasa es una plataforma de código abierto diseñada para el desarrollo de chatbots y asistentes virtuales conversacionales. Utilizando técnicas de procesamiento de lenguaje natural y aprendizaje automático, Rasa permite a los desarrolladores crear sistemas de diálogo inteligentes y personalizados. Una de las principales ventajas de Rasa es su flexibilidad y personalización, ya que los desarrolladores tienen control total sobre el comportamiento y la lógica de sus chatbots. Además, Rasa proporciona herramientas robustas para la gestión del diálogo, la comprensión del lenguaje natural y la integración con otros sistemas. Sin embargo, una posible desventaja de Rasa es su curva de aprendizaje, ya que requiere un conocimiento sólido de PLN y aprendizaje automático para aprovechar al máximo su potencial. Además, debido a su naturaleza de código abierto, puede requerir más tiempo y recursos para implementar y mantener en comparación con otras soluciones comerciales. Sin embargo, aunque rasa no es la API más potente en cuanto a generación de texto, tiene numerosas aplicaciones que resultan interesantes.

## 3.3. Gemini

Gemini es una API de procesamiento del lenguaje natural (PLN) desarrollada por Google que permite a los usuarios interactuar con modelos de lenguaje avanzados para generar texto coherente y relevante en respuesta a consultas y solicitudes. Utiliza modelos de lenguaje de última generación entrenados por Google, que son



capaces de comprender y generar texto en varios idiomas y contextos. Los usuarios pueden enviar texto de entrada a través de la API y recibir respuestas generadas por los modelos de Gemini. Ofrece varios modelos para satisfacer diferentes necesidades y casos de uso, entre los que se encuentran:

- **gemini-pro**: Optimizado para entradas de texto.
- **gemini-pro-vision**: Optimizado para entradas multimodales de texto e imágenes.

Gemini puede utilizarse para una variedad de aplicaciones, incluyendo generación de texto a partir de entradas bien sean de texto, o imágenes, conversaciones de varios turnos (chats) o para la obtención de embeddings para modelos del lenguaje.

Para configurarlo, en primer lugar hay que ejecutar el programa `mibot.py` estando en telegram en la conversaa conversación de telegram el comando `/start` para comenzar la conversación y ya.

Sin embargo, y pese a las grandes funcionalidades de todas estas alternativas nos hemos decantado por hacer una interfaz con telegram desarrollando nuestro propio chatbot usando rasa.

Para desarrollar el chatbot de telegram me he decantado por usar la interfaz de telegram para la cuál se necesita la API de Rasa. Las principales ventajas que ofrece esta herramienta es la facilidad del manejo de la interfaz pues telegram es una herramienta muy conocida con la que los terapeutas pueden estar más familiarizados. Además, esto nos permite también usar la versión del chatbotyayo para móvil.

### 3.3.1. Procesamiento de texto

El modelo más apropiado para el procesamiento de texto es *gemini – pro*. La estructura de las respuestas de este modelo es la siguiente.

```
{
  "candidates": [
    {
      "content": {
        "parts": [
          {
            "text": string
          }
        ]
      },
      "finishReason": enum (FinishReason),
      "safetyRatings": [
        {
          "category": enum (HarmCategory),
          "probability": enum (HarmProbability),
          "blocked": boolean
        }
      ]
    }
  ]
}
```

```

    }
  ],
  "citationMetadata": {
    "citations": [
      {
        "startIndex": integer,
        "endIndex": integer,
        "uri": string,
        "title": string,
        "license": string,
        "publicationDate": {
          "year": integer,
          "month": integer,
          "day": integer
        }
      }
    ]
  }
},
"usageMetadata": {
  "promptTokenCount": integer,
  "candidatesTokenCount": integer,
  "totalTokenCount": integer
}
}

```

Listing 3.6: Estructura de una respuesta de Gemini

- **text** El texto generado.
- **finishReason** El motivo por el que el modelo dejó de generar tokens. Si está vacío, el modelo no dejó de generar los tokens. El motivo puede ser cualquiera de los siguientes:
  1. *FINISH\_REASON\_UNSPECIFIED*: no se especifica el motivo de finalización.
  2. *FINISH\_REASON\_STOP*: punto de detención natural del modelo o secuencia de detención proporcionada.
  3. *FINISH\_REASON\_MAX\_TOKENS*: se alcanzó la cantidad máxima de tokens especificada en la solicitud.
  4. *FINISH\_REASON\_SAFETY*: la generación del token se detuvo porque la respuesta se marcó por motivos de seguridad. Ten en cuenta que `Candidate.content` está vacío si los filtros de contenido bloquean el resultado.

5. *FINISH\_REASON\_RECITATION*: la generación del token se detuvo porque la respuesta se marcó para citas no autorizadas.
  6. *FINISH\_REASON\_OTHER*: todos los demás motivos que detuvieron el token
- **category** La categoría de seguridad para la que se configura un umbral. Los valores aceptables son los siguientes: Haz clic para expandir las categorías de seguridad
    1. *HARM\_CATEGORY\_SEXUALLY\_EXPLICIT*
    2. *HARM\_CATEGORY\_HATE\_SPEECH*
    3. *HARM\_CATEGORY\_HARASSMENT*
    4. *HARM\_CATEGORY\_DANGEROUS\_CONTENT*
  - **probability** Los niveles de probabilidad de daños en el contenido.
    1. *HARM\_PROBABILITY\_UNSPECIFIED*
    2. *NEGLIGIBLE*
    3. *LOW*
    4. *MEDIUM*
    5. *HIGH*
  - **blocked** Una marca boolean asociada con un atributo de seguridad que indica si la entrada o salida del modelo se bloqueó. Si **blocked** es **true**, el campo **errors** en la respuesta contiene uno o más códigos de error. Si **blocked** es **false**, la respuesta no incluye el campo **errors**.
  - **startIndex** Un número entero que especifica dónde comienza una cita en el contenido.
  - **endIndex** Un número entero que especifica dónde termina una cita en **content**.
  - **url** Es la URL de una fuente de cita. Los ejemplos de una fuente de URL pueden ser un sitio web de noticias o un repositorio de GitHub.
  - **title** Es el título de una fuente de cita. Los ejemplos de títulos de origen pueden ser los de un artículo de noticias o un libro.
  - **license** Es la licencia asociada con una cita.
  - **publicationDate** La fecha en que se publicó una cita. Sus formatos válidos son **YYYY**, **YYYY-MM** y **YYYY-MM-DD**.
  - **promptTokenCount** Cantidad de tokens en la solicitud.
  - **candidatesTokenCount** Cantidad de tokens en las respuestas.
  - **totalTokenCount** Cantidad de tokens en la solicitud y las respuestas.

### 3.3.2. Procesamiento de imágenes

Los mensajes multimodales son instrucciones que utilizan modelos avanzados de lenguaje y admiten diferentes tipos de entradas, como texto e imágenes. Al combinar múltiples formatos de entrada, estos mensajes permiten una amplia gama de aplicaciones, como la clasificación de imágenes, el reconocimiento de escritura a mano, la traducción y otras tareas creativas.

En esta sección, exploraremos los tipos de instrucciones que pueden generarse al ingresar tanto imágenes como texto en el modelo Gemini. A través de ejemplos interesantes, veremos cómo este modelo puede procesar esta información multimodal y producir respuestas en forma de texto.

En la actualidad, Gemini <sup>3</sup> tiene la capacidad de procesar solicitudes que combinen texto e imagen como entrada, y luego generar respuestas basadas únicamente en texto. El texto puede utilizarse para contextualizar la imagen o para solicitar al modelo que opere y genere una respuesta relacionada con la imagen.

Un ejemplo muy simple que destaca la capacidad del LLM de reconocer la existencia de algo en una imagen o no y de responder al desarrollador de manera booleana, es el que muestra la figura 3.5. Este enfoque puede ser útil en la detección de contenido específico.



Figura 3.5: Imagen de entrada a Gemini en el primer ejemplo

```
>>> ¿En esta imagen aparece un gato?  
Respuesta de Gemini: True
```

Otro ejemplo es el asociado a la figura, 3.6. Aquí vemos el poder de Gemini para narrar historias y formas más creativas de usar la IA generativa.

```
>>> Escribe un haiku sobre esta foto
```

---

<sup>3</sup>Página oficial de Google con más ejemplos

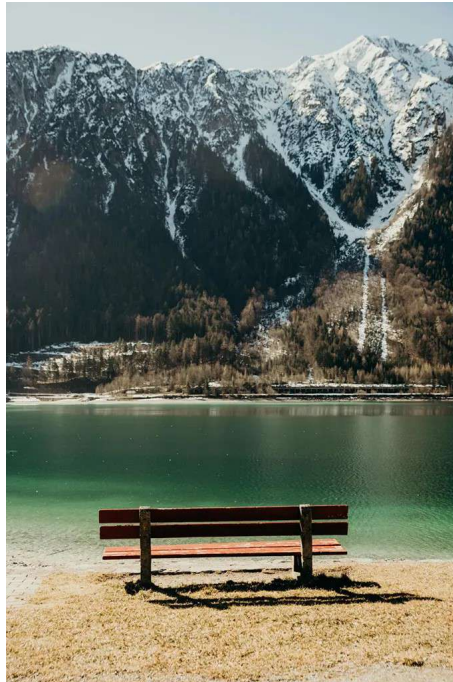


Figura 3.6: Imagen de entrada a Gemini en el segundo ejemplo

Respuesta de Gemini:

Un banco junto al lago,

Una vista de las montañas más allá,

Un momento de paz.

Finalmente, un ejemplo en el que se combinan varias habilidades de Gemini es el que se muestra en la figura 3.7. No solo reconoce las formas, sino que también entiende que, si bien las formas son levemente dibujadas, están destinadas a ser formas distintas vinculadas matemáticamente con atributos específicos (p.ej., 3 lados, 4 lados, 5 lados).

Además, la presencia del signo de interrogación no confunde a Gémini en su interpretación de la progresión lógica de las formas geométricas. En cambio, Gemini identifica que esta es una progresión matemática de 3, 4 a 5 y que, por lo tanto, la última forma tendría 6 lados y propone un hexágono.

>>> ¿Qué sigue? Explica tu razonamiento

Respuesta de Gemini: El triángulo tiene 3 lados, el cuadrado tiene 4 lados y el pentágono tiene 5 lados. El número de lados aumenta en 1 para cada forma. Por lo tanto, la siguiente forma debería tener 6 lados, que es un hexágono.

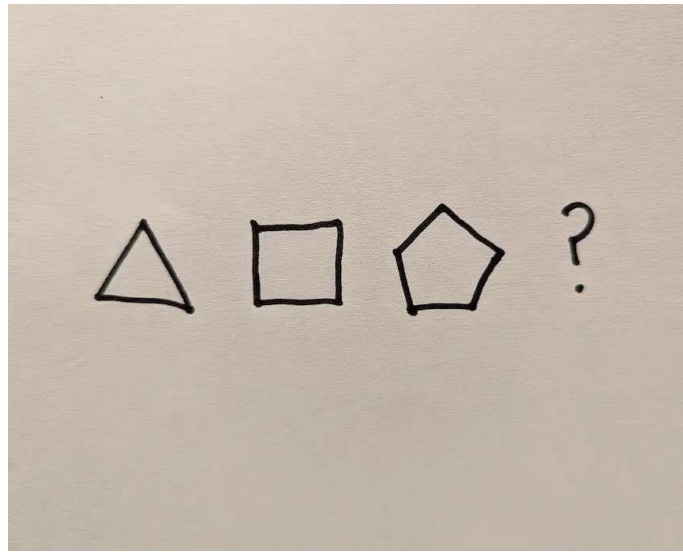


Figura 3.7: Imagen de entrada a Gemini en el ejemplo

## 3.4. Almacenamiento de la información

### 3.4.1. JSON

El JSON, acrónimo de JavaScript Object Notation, es un formato ligero para estructurar datos que se asemeja a los mapas en la programación. Está diseñado para representar datos de manera legible para las máquinas y fácilmente interpretable por los humanos. Se utiliza ampliamente en el intercambio de datos entre aplicaciones web y en el manejo de respuestas de API. El formato JSON consta de pares de clave-valor, donde las claves son únicas y los valores pueden ser cadenas, booleanos, números, objetos JSON o matrices JSON.

En Python, el formato de datos más cercano a JSON es el diccionario. El módulo 'json' de Python permite la conversión entre diccionarios, cadenas JSON y archivos JSON. Para leer un archivo JSON en Python, se utiliza la función `json.load()` para cargar los datos del archivo en un diccionario. Para leer una cadena JSON en Python, se utiliza la función `json.loads()` para analizar la cadena y convertirla en un diccionario.

Para escribir datos en un archivo JSON, se utiliza la función `json.dump()` para escribir un diccionario en un archivo JSON. También se puede utilizar la función `json.dumps()` para convertir un diccionario en una cadena JSON y luego escribir esa cadena en un archivo. Ambas funciones permiten especificar la indentación para formatear el archivo JSON de manera legible. El módulo `json` proporciona una forma conveniente de manipular datos JSON en Python, facilitando el intercambio de datos entre aplicaciones y su almacenamiento en archivos.

### 3.4.2. RDF

Estas tripletas RDF se utilizan para almacenar información de manera estructurada y semántica, lo que permite una representación más rica y significativa de los datos en la web. A través de vocabularios y ontologías, como RDF Schema (RDFS) y Web Ontology Language (OWL), se establecen relaciones y significados precisos entre los términos utilizados en las tripletas RDF.

Las tripletas RDF son ampliamente utilizadas en la web semántica para diversas aplicaciones, como la descripción de recursos y metadatos en la web, la integración de datos de diferentes fuentes, la creación de motores de búsqueda más inteligentes y la construcción de sistemas de recomendación personalizados. Además, RDF proporciona un marco estándar y flexible para representar conocimiento y facilita la interoperabilidad entre diferentes sistemas y aplicaciones en la web.

El Framework de Descripción de Recursos (RDF, por sus siglas en inglés) es un lenguaje de propósito general orientado a la representación de información en la web. Su uso se centra en el desarrollo de la web semántica, y tiene como finalidad describir los recursos de la misma de una manera no orientada a la legibilidad por parte de un humano, sino a la computación de la información contenido por un ordenador.

La web semántica es un proyecto de futuro en el que la información web tiene un significado exactamente definido y puede ser procesado por ordenadores. Por lo tanto, los ordenadores pueden integrar y usar la información disponible en la web. Más información sobre la web semántica puede ser encontrada en da Silva et al. (2007).

RDF está considerado como un lenguaje de metadatos, o "datos sobre datos", ya que con los símbolos de RDF añadimos metainformación a los datos que realmente nos interesan para poder interpretarlos de una manera exacta, es decir, de aquella que el autor de los datos (o, al menos, del autor del marcado RDF sobre estos datos) quería que estos fuesen interpretados.

RDF define los recursos mediante descripciones de los mismos, como puede verse en el listado 3.8. Puede encontrarse más información sobre la manera de la tecnología RDF de describir estos recursos en Champin (2002).

```
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
<cd:artist>Bob Dylan</cd:artist>
<cd:country>USA</cd:country>
<cd:company>Columbia</cd:company>
<cd:price>10.90</cd:price>
<cd:year>1985</cd:year>
</rdf:Description>
```

Figura 3.8: Ejemplo de un recurso RDF.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base= "http://www.animals.fake/animals#">
<rdf:Description rdf:ID="animal">
<rdf:type
rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID="horse">
<rdf:type
rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
<rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>
</rdf:RDF>
```

Figura 3.9: Ejemplo de un recurso RDF SCHEMA.

RDF no define clases de datos específicas para las aplicaciones. En vez de esto, el estándar RDF dispone de RDF Schema. Estos documentos definen nuevas clases, y relaciones entre ellas (herencia, agregación) de una manera muy similar a aquella con la que se acostumbra en la programación orientada a objetos. En la Figura 3.9 podemos ver un ejemplo de un RDF Schema.

## 3.5. Desarrollo de la interfaz

PyQt, wxPython y Kivy son opciones populares para la implementación de interfaces gráficas, cada una con sus propias ventajas y desventajas.

PyQt es conocido por su completo conjunto de widgets, lo que te permite crear interfaces gráficas complejas y altamente personalizadas. Sin embargo, puede tener una curva de aprendizaje más pronunciada debido a su complejidad y sintaxis más compleja.

Por otro lado, wxPython ofrece una sintaxis más simple y fácil de entender, lo que puede ser beneficioso si estás empezando o prefieres un enfoque más directo. Aunque tiene menos widgets y funcionalidades avanzadas que PyQt, sigue siendo una opción sólida con una comunidad activa que proporciona soporte.

Kivy destaca por su diseño adaptable, diseñado para crear aplicaciones con interfaces gráficas que funcionan en una amplia gama de dispositivos. Utiliza un lenguaje de marcado declarativo que permite definir la interfaz de usuario de manera intuitiva y separada del código Python. Sin embargo, puede tener menos documentación y recursos disponibles en comparación con PyQt y wxPython.



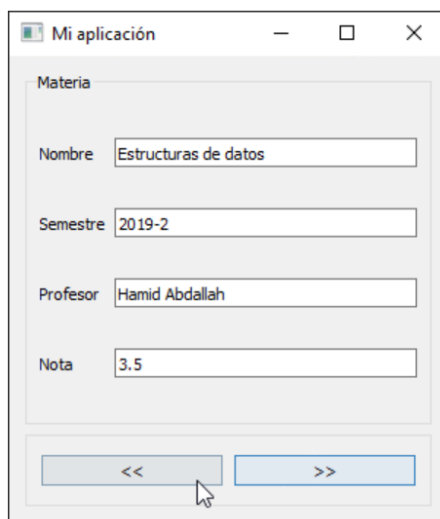


Figura 3.10: Ejemplo de interfaz generada con PyQt5

## 3.6. Programación orientada a objetos

La Programación Orientada a Objetos (POO) ha ganado una popularidad significativa en la comunidad de programación debido a su capacidad para desarrollar aplicaciones más robustas, flexibles y fáciles de mantener. Este paradigma se basa en la organización de programas como una colección de objetos interconectados, cada uno con su propio conjunto de datos y funcionalidades. En este artículo, exploraremos los conceptos clave de la POO, cómo implementarla en diversos lenguajes y cómo aprovechar sus ventajas para construir aplicaciones sólidas y flexibles.

La POO es un paradigma de programación que se basa en la idea de clases y objetos. Se utiliza para estructurar programas de software en piezas simples y reutilizables de código, donde una clase actúa como una plantilla para crear múltiples instancias de objetos. Este enfoque invita a considerar las entidades dentro del contexto del problema a resolver, como libros, bibliotecarios y usuarios, y representarlas como objetos con propiedades y comportamientos.

La POO se inspira en la forma en que percibimos y entendemos el mundo que nos rodea. Cada entidad se convierte en un objeto con sus propios atributos y métodos, y la interacción entre estos objetos es fundamental. La encapsulación, abstracción, herencia y polimorfismo son los principios fundamentales de la POO que permiten crear aplicaciones más organizadas, reutilizables y mantenibles.

La POO permite la reutilización del código, evita la duplicación, protege la información a través de la encapsulación y facilita el trabajo en equipo al minimizar la posibilidad de duplicar funciones. Además, proporciona una estructura más clara y modular para el desarrollo de software, lo que facilita el mantenimiento y la escalabilidad a medida que los requisitos evolucionan.

La Programación Orientada a Objetos es esencial en el diseño de aplicaciones y programas informáticos modernos. Ofrece numerosas ventajas, como la reutilización

del código, la modularidad y la facilidad de mantenimiento, lo que la convierte en una opción ideal para resolver desafíos de programación complejos. Sin embargo, requiere una planificación cuidadosa y un análisis detallado de los requisitos para aprovechar al máximo sus beneficios.

## 3.7. VPN

Claro, aquí tienes un texto más detallado y técnico sobre qué son las VPN y sus ventajas:

Una VPN (Red Privada Virtual) es una tecnología que establece una conexión segura y cifrada entre un dispositivo y una red privada remota a través de internet. Esto se logra mediante la creación de un túnel de comunicación virtual que encripta los datos transmitidos entre el dispositivo del usuario y el servidor VPN. Las VPN son utilizadas ampliamente por individuos y organizaciones para garantizar la privacidad, la seguridad y la accesibilidad de las comunicaciones en línea.

Las características y ventajas técnicas de las VPN incluyen:

Una VPN protege la privacidad al ocultar la dirección IP del usuario y encriptar los datos transmitidos. Esto previene la interceptación de datos por parte de terceros malintencionados, como hackers o agencias de vigilancia. La encriptación de extremo a extremo asegura que solo el dispositivo de origen y el servidor VPN puedan descifrar la información transmitida.

Las VPN utilizan diversos protocolos de seguridad, como IPsec (Protocolo de seguridad de Internet), SSL/TLS (Capa de sockets seguros/Protocolo de capa de transporte seguro), y OpenVPN, para establecer conexiones seguras y confiables. Estos protocolos garantizan la integridad de los datos y la autenticación de las partes involucradas en la comunicación.

Las VPN permiten a los empleados acceder de forma segura a recursos corporativos desde ubicaciones externas, utilizando conexiones cifradas. Esto es esencial para el trabajo remoto y las operaciones empresariales distribuidas, donde se necesita un acceso seguro a la red interna.

Las VPN son utilizadas por empresas para conectar múltiples ubicaciones geográficas en una red privada unificada a través de internet. Esto elimina la necesidad de redes privadas dedicadas y reduce los costos de infraestructura.

Al permitir que los usuarios simulen una ubicación diferente, las VPN facilitan el acceso a contenido y servicios en línea que podrían estar restringidos geográficamente. Esto es útil para usuarios individuales que desean acceder a sitios web o servicios que no están disponibles en su región.

Al utilizar VPN, las organizaciones pueden implementar estrategias de balanceo de carga y tolerancia a fallos para optimizar el rendimiento y la disponibilidad de sus servicios en línea.

Las VPN pueden integrarse con sistemas de filtrado de contenido y seguridad

---

perimetral para proteger la red contra amenazas externas, como malware, phishing y ataques DDoS (denegación de servicio distribuido).



# Capítulo 4

## Desarrollo de prototipos

El capítulo actual tiene como objetivo presentar en orden temporal los distintos prototipos que se han desarrollado hasta llegar a la versión final que se presenta en el capítulo 5.

La estructura del capítulo muestra en orden lineal en el tiempo como se ha ido construyendo la arquitectura del sistema. De esta forma se puede ver como el modelo va pasando por distintos prototipos de forma incremental.

En este capítulo, también se pretende mostrar y hacer hincapié en los diferentes desafíos, contratiempos y retos a los que ha habido que hacer frente durante el desarrollo del prototipo, por qué se han originado y cómo se han solucionado.

Con todo ello, se pretende tener una visión global de lo que ha sido el trabajo de desarrollo del sistema. Partiendo de la situación que se muestra en el capítulo 2, con el estudio teórico llevado a cabo en el capítulo 3 y dando lugar al sistema que se detallará en el capítulo 5.

El código asociado a los distintos prototipos se puede consultar en el repositorio de GitHub, y en concreto, en la carpeta de prototipos.

### 4.1. Planteamiento del problema

Cómo se comentó en el capítulo 1, el objetivo de este proyecto es construir una herramienta que facilite a los terapeutas la labor de realizar terapias de reminiscencia a personas con demencia. Para ello, se trata de obtener un chatbot capaz de realizar preguntas adecuadas, extraer información útil de ellas y generar nuevas preguntas para rellenar la información faltante.

Con este objetivo en mente, se plantea un diseño iterativo que parta de un chatbot capaz de realizar preguntas predefinidas y sobre él ir iterando hasta conseguir un prototipo completo.

## 4.2. Preguntas predefinidas

La primera versión del prototipo tuvo como único objetivo ser una herramienta capaz de realizar una serie de preguntas predefinidas, y almacenar la información obtenida.

Para la selección de las mismas se optó por emplear la plantilla “Life History Templates” UK (2023). Estas preguntas desempeñan un papel crucial en el flujo de interacción del sistema, ya que son el esqueleto principal de la conversación con el usuario.

La plantilla UK (2023), concebida con el propósito de estructurar la recopilación de información sobre la historia de vida de un individuo, ofrece una serie de campos predefinidos que abarcan una amplia gama de aspectos relevantes. Dichos campos han sido seleccionados, con el objetivo de garantizar una exploración exhaustiva y completa de la vida del paciente tratando de asegurar que se aborden aspectos clave de la vida del paciente, tales como eventos significativos, relaciones personales, intereses y preferencias, entre otros. Esto permite obtener una visión completa y detallada del individuo, lo cual es fundamental para proporcionar una atención personalizada y efectiva.

Una vez que se han identificado los campos pertinentes, se procede a la creación de un documento denominado “preguntas.txt”, el cual sirve como guía durante la interacción con el paciente. En este documento se registran las preguntas específicas que se deben realizar en cada campo, así como la información deseada que se espera obtener a través de las respuestas del paciente. Esto facilita el seguimiento y la organización de la información recopilada, asegurando que se cubran todos los aspectos relevantes de la historia de vida del individuo.

Además, la plantilla “Life History Templates” incluye la recomendación de utilizar fotografías como herramienta para estimular la memoria y facilitar la obtención de información por parte del paciente. Esta estrategia, basada en la estimulación visual, ha demostrado ser efectiva para desencadenar recuerdos y promover la narración de experiencias pasadas.

Como parte de la implementación de nuestro prototipo, planeamos integrar esta funcionalidad en la sección dedicada a las imágenes (sección 4.9). Esto permitirá enriquecer la interacción con el paciente y mejorar la calidad de la información recopilada, contribuyendo así a una evaluación más completa y precisa de su historia de vida.

## 4.3. BARD

El segundo hito en el desarrollo del chatbot fue mejorar el primer chatbot haciéndolo más inteligente y capaz de analizar las respuestas, identificar la información omitida y hacer preguntas específicas para obtener la información faltante.

Para el almacenamiento de la información se decidió usar *JSON* debido a su

estructura legible y su amplia compatibilidad, suponen una elección tecnológica ideal para almacenar los datos de este sistema. Su capacidad para representar datos complejos de manera eficiente y su interoperabilidad garantizan una experiencia de desarrollo ágil y una gestión eficaz de la información de los usuarios.

Por otro lado, se desarrollo una pequeña clase de forma provisional. Esta clase es la clase *Paciente* que tenía como objetivo estructurar y reunificar la información.

A continuación, se busco poder hacer un análisis de las respuestas exhaustivo que permitiera extraer información útil de las respuestas dadas por el usuario.

Tras el estudio realizado en la sección 3.2 se tomo la decisión de utilizar la API de BARD. Las principal característica que llevo a tomar esta decisión es que se trata de una API ya entrenada capaz de crear texto de alta calidad similar al humano. Está hecha a medida para desarrolladores y creadores de contenidos, permitiendo la automatización de la generación de contenidos para diversas necesidades, como en este caso la generación de un chatbot. Todo esto, supone una gran ventaja frente a otras herramientas como las distintas bibliotecas que suponen desarrollar y generar nuestros propios algoritmos de generación del lenguaje, o otras APIs que requieren un entrenamiento que da lugar a horas y horas de computo para obtener resultados, a lo sumo, tan buenos como los que ofrecía BARD con su simple instalación.

Para instalar BARD se utilizo el entorno de *Google Collaborate*. En primer, lugar fue necesario instalar los paquetes *bardapi* y *pyTelegramBotApi*. Una vez intaladas las bibliotecas y para usar la API es necesario registrarse y obtener un token que nos sirve como identificador de la API.

Una vez con todo instalado, se utilizó la API Bard para extraer información a partir de respuestas a preguntas específicas. Como se ve en la figura 4.1, BARD

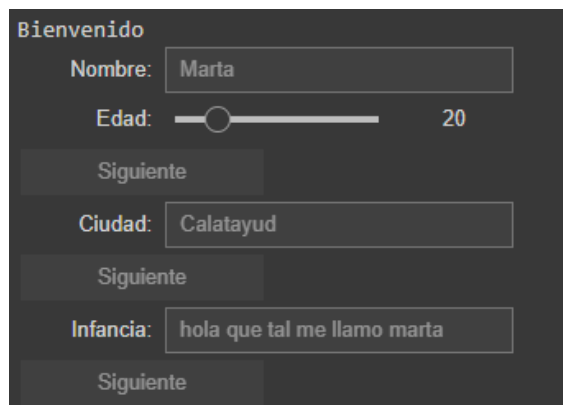
```
prompt = "A partir del texto a continuación, que contiene información
sobre una persona y damelo en una lista info donde
info[nombre]:valor_atributo. Limitate a dar la información pedida. "

texto = "Hola mi nombre es Marta, tengo 22 años y soy de Zaragoza."
>> **Info[nombre:Marta;edad:22;ciudad:Zaragoza]**
¿Hay algo más que pueda hacer por ti?
```

Figura 4.1: Ejemplo de uso de BARD.

supone una herramienta útil para la extracción de información a partir de respuestas y la estructuración de la misma. Sin embargo, aunque indiques que se limite a darte la información pedida, en un amplio número de casos añade texto adicional como en este caso “¿Hay algo más que pueda hacer por ti.”. Para corregir esto se desarrolló una serie de funciones que parsearan el texto de forma manual, a partir de la respuesta dada por BARD. En concreto, la función *extraer\_info* que extrae información de un texto en un formato específico, que es el pedido a BARD, y la devuelve como un diccionario de atributo-valor. Por otro lado, la función *parsear\_info* procesa la información proporcionada por el usuario y actualiza el objeto paciente.

La interacción chatbot usuario de este prototipo se llevaba a cabo mediante una pequeña interfaz desarrollada con widgets de IPython. Esta versión del chatbot solicita al usuario información como nombre, edad, ciudad, etc., y al final muestra un mensaje de despedida con los datos recopilados. Cómo se puede ver en la figura



The image shows a dark-themed chatbot interface. At the top, it says "Bienvenido". Below that, there are four input sections, each with a label and a "Siguiete" button. The first section is "Nombre:" with the text "Marta" entered. The second section is "Edad:" with a slider set to "20". The third section is "Ciudad:" with the text "Calatayud" entered. The fourth section is "Infancia:" with the text "hola que tal me llamo marta" entered.

Figura 4.2: Muestra de la interacción chatbot-usuario en el primer prototipo

4.6, la interacción con el usuario se hace mediante cajas en las que se escribe la información solicitada. Una vez rellanado se envía la información al chatbot para su análisis pulsando en el botón “Siguiete” que despliega un nuevo cuadro con la siguiente información solicitada.

Todo este desarrollo se llevo a cabo durante en los meses previos a diciembre de 2023. Fue ese mes cuando Google fortaleció la capacidad de Bard brindando habilidades más avanzadas de comprensión, razonamiento, resumen y codificación. De esta forma, paso a ser Gemini. Este cambio se oficializo en Febrero de 2024 y supuso la imposibilidad de usar BARD debido a las restricciones de uso según la localización de Gemini. Esta restricción limita el uso de la API de Gemini a una serie de regiones entre las que no se encuentra España ni ningún país de la Unión Europea.

En consecuencia, este primer prototipo paso a un segundo plano y se centro el trabajo de los siguientes meses en encontrar qué API usar para el desarrollo del trabajo. Se barajaron distintas posibilidades.

Por un lado se estudio la API de GPT por sus buenas prestaciones y características, ya presentadas en el capítulo 3. Sin embargo, la API es de pago y para usarla en las versiones de bajo precio o gratuito es recomendable entrenarlo lo que suponía un coste en computo no asumible.

Por otro lado , se decidió estudiar distintos modelos de hugging face <sup>1</sup> entre los que destaco gemma.

---

<sup>1</sup><https://huggingface.co/>



## 4.4. Prototipo usando GEMMA

Al comenzar a desarrollar este prototipo, se trató de seguir una estructura que permitiera reutilizar todo el código posible del prototipo anterior. Para ello, se trató de desarrollar una versión de nuevo en *GoogleCollaborate*. Posteriormente, se estudió la alternativa de la implementación de un modelo local. En el desarrollo de este prototipo se utilizaron diferentes bibliotecas de *HuggingFace*.

`AutoTokenizer` y `AutoModelForCausalLM` son clases proporcionadas por la biblioteca *HuggingFaceTransformers* que se utilizan para cargar y trabajar con modelos de lenguaje preentrenados.

1. `AutoTokenizer`: Esta clase se utiliza para cargar un tokenizador adecuado para el modelo de lenguaje preentrenado que se desea utilizar. El tokenizador se encarga de dividir el texto en unidades más pequeñas, como palabras o subpalabras, y convertirlas en vectores numéricos que el modelo de lenguaje puede entender. En nuestro contexto

```
AutoTokenizer.from_pretrained('google/gemma-7b')
```

se utiliza para cargar el tokenizador asociado al modelo de lenguaje *gemma-7b* de Google.

2. `AutoModelForCausalLM`: Esta clase se utiliza para cargar el modelo de lenguaje preentrenado que se desea utilizar. El modelo de lenguaje es responsable de generar texto basado en las entradas proporcionadas. En nuestro contexto,

```
AutoModelForCausalLM.from_pretrained('google/gemma-7b')
```

se utiliza para cargar el modelo de lenguaje "gemma-7b" de Google, que es capaz de generar texto de manera condicional, es decir, generar texto secuencialmente teniendo en cuenta el contexto anterior.

En resumen, utilizaremos `AutoTokenizer` para cargar el tokenizador asociado al modelo de lenguaje preentrenado, y `AutoModelForCausalLM` para cargar el propio modelo. Estas clases son esenciales para preparar el modelo para su uso en la generación de texto o en otras tareas de procesamiento de lenguaje natural.

### 4.4.1. Google Collaborate

En primer lugar, para poder descargarnos el modelo se tiene que utilizar la biblioteca "os" para establecer las credenciales de autenticación de Hugging Face. Esto es necesario para acceder a modelos alojados en *HuggingFace* y para realizar operaciones como clonar repositorios de modelos.

A continuación es necesaria la carga de la clase “AutoTokenizer” y “AutoModelForCausalLM” de la biblioteca “Transformers” de Hugging Face para cargar un tokenizador y un modelo de lenguaje preentrenado. En este caso, el tokenizador y el modelo elegidos se cargan desde la dirección “*google/gemma – 7b*”.

Para poder realizar todas estas instalaciones se ha de configurar el entorno de ejecución de *GoogleCollaborate* de forma que se utilicen cuatro GPUs. Esto requiere seleccionar la versión “T4 GPU” del acelerador de hardware. Ejecutar el programa sin seleccionar esta opción impedimenta la carga y descarga de los modelos y el programa y en consecuencia su uso.

Una vez realizada toda esta instalación, se puede utilizar el modelo de lenguaje cargado para generar texto a partir de una entrada específica. Sin embargo, en su versión gratuita *Collaborate* tiene limitado el uso de varios GPUs con lo que esta versión apenas pudo ser usada antes de que el entorno de ejecución limitara el uso de más de una GPU y en consecuencia, hubiera que descartar la alternativa de seguir desarrollando con *gemma – 7* en *Collaborate*.

En resumen, este código carga un modelo de lenguaje preentrenado, genera texto utilizando el modelo y realiza operaciones relacionadas con la gestión de modelos alojados en *HuggingFace*.

#### 4.4.2. Linux

Las versiones locales de *gemma – 2b* y *gemma – 7b* en local han de ser instaladas en Linux debido a sus características. Requiere la instalación de las bibliotecas comentadas anteriormente así como de *HuggingFace* para poder acceder a los modelos.

En primer lugar, se optó por el modelo *gemma – 2b* pero debido a las limitaciones del mismo y a la alta exigencia de las tareas que se requieren en este proyecto, se tuvo que usar el modelo de 7 millones de parámetros. Con él, se llegó a implementar una versión similar a la descrita en la sección 4.3 pero mediante el uso de *gemma – 7b* de *HuggingFace*.

Se amplió este modelo, tratando de alcanzar el siguiente reto: la generación de preguntas. Para ello, se desarrolló un módulo de generación de preguntas.

El módulo en concreto aborda la tarea de generar preguntas a partir de un texto dado utilizando herramientas de procesamiento de lenguaje natural. En concreto, se centra en la biblioteca *spaCy*, una poderosa herramienta para el procesamiento avanzado del lenguaje natural que ya ha sido presentada en la sección 3.1.2 .

La generación de preguntas comienza con la tokenización y el análisis sintáctico del texto proporcionado. Para este propósito, se utiliza el modelo preentrenado de *spaCy* en español, denominado “*es\_core\_news\_sm*”. Este modelo es capaz de analizar la estructura gramatical del texto y etiquetar cada token con información como el tipo de palabra (sustantivo, verbo, etc.) y la categoría gramatical.

Una vez que el texto ha sido analizado, el módulo procede a identificar los sustan-

tivos dentro del texto. Esto se logra mediante el análisis de las etiquetas gramaticales asignadas por *spaCy* a cada token. Cada sustantivo identificado se considera como un tema potencial para formular preguntas.

Con los sustantivos identificados, el módulo genera preguntas sobre la experiencia relacionada con cada uno de ellos. Por ejemplo, si el texto menciona la palabra es “viaje”, el módulo formulará la pregunta “¿Qué tal tu experiencia de viaje?”.

Para el desarrollo de la interfaz de usuario, se consideraron dos enfoques diferentes: uno utilizando el framework Flask para crear un servidor web y otro utilizando la biblioteca Tkinter para construir una interfaz gráfica de usuario (GUI) de escritorio.

Con Flask, se optó por crear un servidor web que presenta una página web con un formulario. Flask maneja las rutas y funciones asociadas, permitiendo que los usuarios ingresen información en el formulario y envíen los datos al servidor. Esto resulta en una interfaz basada en la web que puede ser accesible a través de un navegador.

Por otro lado, con Tkinter, se creó una interfaz gráfica de usuario (GUI) de escritorio. Tkinter permite crear ventanas, etiquetas y botones para construir la interfaz. En este caso, se diseñó una ventana con una etiqueta de pregunta y dos botones de opción (“Sí” y “No”). Esta interfaz se ejecuta localmente en la máquina del usuario y es independiente del navegador web.

Este prototipo que supone un incremento respecto al anterior también tuvo que ser descartado sin llegar a desarrollar algunas funcionalidades y módulos hasta el final, como por ejemplo la interfaz o la generación de preguntas. Este parón se debió a un problema hardware. Al hacer instalarme Linux, no tenía apenas espacio en el disco duro por lo que la partición no fue demasiado grande. El gran tamaño de los modelos, en concreto del modelo *gemma – 7b* ocupó gran parte del espacio de la partición. Pocas semanas después del comienzo del desarrollo de este prototipo tuvo que descartarse debido a la necesidad de ejecutarlo en Linux, acompañada de las limitaciones de espacio en mi ordenador que hicieron que no pudiera seguir desarrollándolo en ese sistema operativo.

## 4.5. Desarrollo de la interfaz con Telegram

Paralelamente a todo el trabajo descrito anteriormente, y durante el estudio de las diferentes alternativas de interfaces, APIs y modelos se estudio la alternativa de desarrollar la interfaz con Telegram.

Esta opción fue encontrada al buscar alternativa a BARD o Gemma, y encontrarse diversos chatbots implementados con RASA y que adaptaban facilmente su interfaz a Telegram. Aunque la capacidad de RASA de generación de texto no cumplía los requisitos requeridos para ser considerada en este proyecto.

Sin embargo, se encontró especialmente interesante debido a su fácil integración con Telegram para desarrollar chatbots. Como se puede ver en la figura 4.3 la interfaz generada de esta forma presenta numerosas ventajas.

En primer lugar, Telegram es una plataforma de mensajería ampliamente utilizada en todo el mundo, lo que garantiza un amplio alcance y accesibilidad para los usuarios. Esto significa que los usuarios pueden acceder al chatbot fácilmente desde su escritorio, pero también desde sus dispositivos móviles, lo que facilita su adopción y uso.

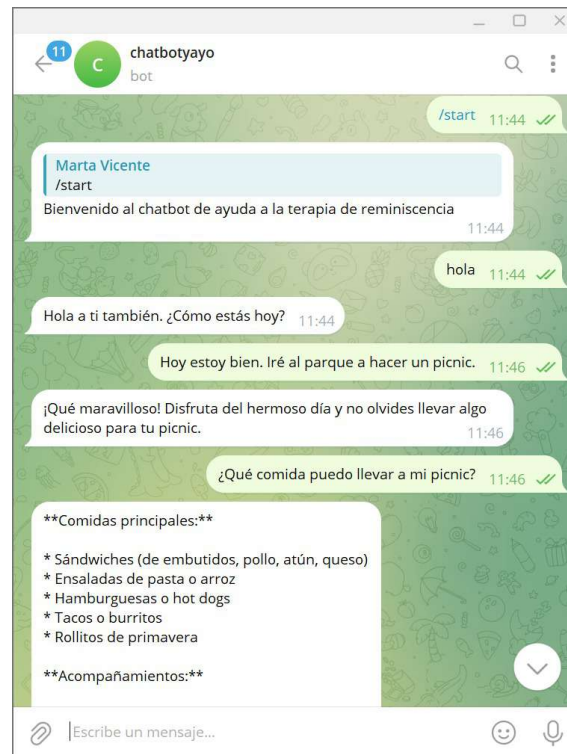


Figura 4.3: Ejemplo de uso de BARD.

Esta forma de generar la interfaz implica el desarrollo de manejadores que controlen el funcionamiento según el tipo de mensaje.

- Manejadores para comandos, por ejemplo, el comando start.
- Manejadores para mensajes de texto.
- Manejadores para imágenes.

Se utiliza la API de Telegram a través del módulo “telebot” para interactuar con el servicio de mensajería y gestionar los mensajes entrantes y salientes. Además, se inicia un bucle de escucha para que el bot esté constantemente activo y pueda responder a los mensajes de los usuarios en tiempo real.

## 4.6. Gemini

A lo largo de los meses se estudiaron múltiples modelos y alternativas a BARD que no resultaban lo suficientemente eficaces. Por contrario, otras sí resultaron eficaces y pese a dar lugar a pequeños prototipos no pudieron seguir desarrollándose debido a los diversos problemas que se han contado a lo largo de este capítulo. Todo este estudio, implementación de modelos y desarrollo de prototipos fallidos conlleva horas de tiempo y esfuerzo. Todo esto deriva en meses centrados en análisis de alternativas en lugar del propio desarrollo del código en sí, por lo que finalmente hubo que decantarse por la API que resultaba más eficaz y fiable y que daba mejores resultados: la API de Gemin. Como se ha comentado antes, esta alternativa se descarto en un primer momento por no estar disponible en España. Sin embargo este problema se pudo solventar fácilmente mediante el uso de VPN.

## 4.7. Análisis de las respuestas

Como ya se ha mencionado anteriormente, el segundo hito a conseguir es la construcción de una herramienta que sea capaz de analizar las respuestas, identificar la información omitida y hacer preguntas específicas para obtener la información faltante. Estas tres tareas se llevan a cabo usando la API de gemini y en concreto, el modelo *gemini – pro*.

### 4.7.1. Análisis de las respuestas y identificación de la información omitida

El proceso de análisis de respuestas implica evaluar las respuestas recibidas a las preguntas planteadas durante la conversación. En nuestro caso se lleva a cabo mediante el uso del modelo y una serie de funciones que reciben como entrada la respuesta del usuario y devuelven un *json* con la información asociada. Para ello, las preguntas predefinidas se almacenan en un fichero junto con una serie de campos que representan la información mínima que se quiere obtener de cada uno de ellos como se puede ver en la figura ??.

¿Tienes hijos? : [Nombre hijos,edad hijos,recuerdos con hijos]

El modelo, gracias a un prompt y a una llamada adecuada, se encarga de extraer la información y asociarla a cada campo, así como de indicar si hay información faltante. Pese a que se pide que la salida se devuelva como un *json* esta información ha de ser parseada manualmente debido a las alucinaciones y otros fenómenos que pueden dar lugar a errores. Otro suceso que ha de ser tenido en cuenta, es la posibilidad de no generación de respuesta por parte de gemini. Es por eso, que hay que tener presentes en todo momento los distintos campos explicados en la sección 3.3, para tener un control de qué esta ocurriendo en todo momento.

Es por esto, que se han desarrollado una serie de funciones que se encargan de transformar la respuesta en un objeto *json* como se muestra en la figura 4.4. Este prototipo también añade información adicional que haya podido obtener de la respuesta y que no se pueda asociar a los campos predefinidos. Sin embargo, estos campos son útiles para llevar un control de qué información no ha podido ser obtenida y en tal caso, generar nuevas preguntas que nos permitan obtenerla.

```
>>>¿Tienes hijos?
>>> Sí tengo hijos pero no estoy seguro de dónde están ahora.
    Mi hija Lucía es médico y el mayor es Juan.
>>> {'Nombre hijos': ['Lucía','Juan'], 'edad hijos': 'No
    Encontrado', 'recuerdos con hijos' : 'No Encontrado'}
```

Figura 4.4: Análisis de respuestas

#### 4.7.2. Generación de preguntas para obtener la información faltante

La generación automatizada de preguntas es un componente clave del sistema diseñado. Este proceso implica la creación de nuevas preguntas dirigidas a recopilar información que no ha podido ser obtenida en las preguntas predefinidas. El proceso de generación de preguntas comienza tras la identificación de un campo al que no se ha podido asociar ningún tipo de valor o sobre el que no se ha obtenido información. En ese momento y gracias a gemini se generan una serie de preguntas asociadas a los campos que no han sido rellenados tales como las que se muestran en la figura 4.5. Estas preguntas se van lanzando al usuario hasta rellenar los campos faltantes o hasta agotar el límite de preguntas asociadas a un mismo tema que se va a hacer al usuario.

```
>>>'¿Cuáles son sus edades?'
```

```
>>> '¿Cómo describe la dinámica de su familia?'
```

```
>>> '¿Qué recuerdos tiene con sus hijos?'
```

Figura 4.5: Generación de preguntas

### 4.8. Interfaz e interacción con el usuario

El siguiente hito en el desarrollo del chatbot es desarrollar la interfaz que permita al usuario interactuar de una forma que resulte sencilla y práctica.

Como se estudió en la sección existen numerosas alternativas utilizando diferentes bibliotecas. Estos módulos permiten crear interfaces usables y simples. Sin embargo,

durante el estudio previo realizado apareció en escena la API de Rasa (4.5) y su integración con Telegram.

Esta opción ofrece una interfaz intuitiva y fácil de usar, lo que proporciona una experiencia de usuario más agradable y cómoda. Los usuarios pueden interactuar con el chatbot de manera similar a como lo harían con otros contactos o grupos en la aplicación, lo que reduce la curva de aprendizaje y aumenta la aceptación del usuario.

Telegram también ofrece una variedad de funcionalidades integradas que pueden ser útiles para un chatbot de terapia de reminiscencia, como el envío de fotos de forma sencilla. Esto permite que la experiencia del usuario sea más rica y variada, siendo mucho más fiel a la plantilla de generación de historias de vida mencionada en la sección 4.2. Esto puede mejorar la efectividad del chatbot en la prestación de servicios de apoyo emocional.

### 4.8.1. Integración de Gemini y Telegram

La decisión de desarrollar la interfaz del prototipo integrando Telegram y GEMINI para tener una interfaz en Telegram tuvo como consecuencia la necesidad de llevar a cabo una refactorización.

#### 4.8.1.1. Clase Pregunta

La clase “Pregunta” se crea para almacenar de manera integral toda la información asociada con una pregunta en un programa, especialmente en el contexto de la interacción con Telegram mediante textos. En versiones anteriores del desarrollo, cada vez que se requería información adicional, se realizaban preguntas y el flujo no avanzaba hasta completar estas preguntas extra o obtener la información necesaria. Al integrar esta funcionalidad con Telegram, se necesita devolver desde la función principal la información para que la interfaz pueda mostrar la pregunta, lo cual puede interrumpir el flujo original del código. Esta integración conlleva a una refactorización que dio lugar a la creación de la clase “Pregunta”.

La clase “Pregunta” encapsula una pregunta con su enunciado, respuesta, campos asociados y preguntas adicionales relacionadas. Su objetivo principal es proporcionar una estructura organizada para manejar la información de una pregunta específica y sus detalles. Permite actualizar la respuesta principal, indicar el estado de los campos (respondidos o no respondidos), agregar preguntas adicionales relacionadas y gestionar las respuestas a estas preguntas secundarias.

Además, la clase “Pregunta” ofrece métodos para actualizar la respuesta, marcar campos como respondidos, agregar preguntas adicionales, gestionar las respuestas a estas preguntas secundarias y obtener una representación textual de la instancia de la clase.

### 4.8.2. Telegram

Una vez llevada a cabo la refactorización del código se llevo a cabo la integración del prototipo con *Telegram*. Para ello es necesario importar la biblioteca *telebot* que es la que permite interactuar con Telegram.

Para conectar Telegram con el chatbot desarrollado a través de la API de Gemini, fue necesario obtener un token de Telegram. Es importante mencionar que este paso forma parte de la configuración inicial y no requiere intervención por parte del usuario final durante la puesta en marcha.

Este token se obtuvo a través del usuario de telegram *@BotFather* y con el comando `\newbot` y `\token`. Una vez con el token se creo el *chatbotyayo* con nombre de usuario *@mavice07\_bot*. Para comenzar a interactuar con *chatbotyayo* basta con buscar ese usuario en Telegram y enviar el comando `\start`.

Una vez obtenido el token y finalizadas todas las configuraciones se implementa la lógica del sistema. Se definen tres manejadores de mensajes:

- El manejador *cmd\_start()* se activa únicamente cuando un usuario envía el comando `\start` al bot. Este manejador inicia la lógica del bot al cargar las preguntas utilizando la función *cargar\_preguntas()* del módulo *tfg* y envía un mensaje de bienvenida al usuario.
- El manejador *bot\_mensajes\_text()* se activa cuando un usuario envía un mensaje de texto al bot. Si el mensaje comienza con "\", se envía un mensaje de error indicando que el comando no está disponible. Si es un mensaje de texto estándar, se llama a la función *siguientePregunta()* del módulo *tfg* y se analiza el mensaje como respuesta al último mensaje enviado por el bot. De esta forma se procesa la respuesta del usuario y se genera una respuesta apropiada.
- El manejador *photo()* se activa cuando un usuario envía una foto al bot. El bot descarga la foto, la guarda en el sistema de archivos y llama a la función *analizador\_imagenes()* del módulo *imagenes* para analizar la imagen y generar una respuesta apropiada.

Finalmente, se inicia el bucle de escucha del bot *bot.infinity\_polling()* para que esté constantemente esperando y respondiendo a los mensajes de los usuarios.

## 4.9. Extracción de información a partir de imagenes

La api de Gemini tiene muchas funcionalidades que permiten hacer del chatbot una herramienta completa. En concreto, gracias a la interfaz elegida y al modelo *gemini-pro-vision* se ha implementado una funcionalidad extra respecto a las que estaban planteadas originalmente. Esta funcionalidad es la extracción de información a partir de imagenes. En cualquier momento, el usuario puede adjuntar una foto. El



programa se encarga de analizar la fotografía y a partir de ella hacer preguntas que permitan extraer más información que la obtenida únicamente con las preguntas.

Mediante el reconocimiento y clasificación de objetos, personas y lugares en imágenes, la IA puede identificar elementos significativos que pueden evocar recuerdos en los pacientes. Además, la capacidad de la IA para detectar emociones a través de expresiones faciales puede ser útil para comprender el estado emocional de los pacientes durante las sesiones terapéuticas.

El análisis de imágenes también puede proporcionar contexto y significado a las experiencias pasadas. Al examinar el contenido visual de las fotografías, la IA puede ayudar a contextualizar recuerdos específicos, facilitando así la conversación y la narración de historias durante las sesiones de terapia. La selección cuidadosa de imágenes personalizadas para cada paciente puede estimular recuerdos y promover interacciones significativas.



Figura 4.6: Ejemplo de extracción de información a partir de imágenes



# Capítulo 5

## Diseño de un chatbot de ayuda a la terapia de reminiscencia

Este capítulo tiene como objetivo presentar la solución final desarrollada a partir de la problemática presentada en el capítulo 1.2. Para ello, se describirá en profundidad cada uno de los componentes básicos de la arquitectura del sistema, y se presentará esta en sí misma. Por otro lado, se explicará tanto el proceso de puesta en marcha como las herramientas necesarias para ese mismo objetivo.

Con el objetivo de permitir la opción de estudiar los componentes de forma práctica, es decir, usando el prototipo, este capítulo comenzará por la explicación de la puesta en marcha. A continuación, se dará una idea global de la arquitectura del sistema y finalmente se irá explorando en cada sección, cada uno de los módulos que la componen.

### 5.1. Herramientas y puesta en marcha

Para la puesta en marcha el primer paso es obtener la *API Key* para lo que es necesario el uso de una VPN.

#### 5.1.1. VPN

Debido a las restricciones geográficas actuales de la API de Gemini para poder usarla es necesario el uso de una VPN. En concreto y para el desarrollo de este proyecto la conexión a la VPN se ha realizado mediante la herramienta *hide.me VPN*. Esta herramienta crea un túnel seguro utilizando poderosos protocolos VPN, oculta nuestra IP real con una suya y cifra todo el tráfico de internet que pasa por este túnel para que podamos navegar libremente. Además, *hide.me* está certificada como una VPN cero registros. Esto significa que no se almacena información de ningún tipo. El uso de la VPN es necesario tanto para obtener la *API Key* como para el uso de la misma. Los países en los que se encuentra disponible Gemini se

pueden consultar en la web de la api de gemini.

### 5.1.2. Instalación de la API de Gemini

Para comenzar a utilizar la API de Gemini con Python, es necesario seguir estos pasos para instalar el SDK y configurar tu clave de API.

En primer lugar, instalamos el SDK (Software Development Kit). La API de Gemini está contenida en el paquete `google-generativeai` en PyPI, por lo que el primer paso sera instalar esa dependencia.

```
!pip install -U google-generativeai
```

Para utilizar la API de Gemini, se necesita una clave de API obtenida del Google AI Studio. Una vez que tengas tu clave, puedes configurarla para que el SDK la utilice:

```
import google.generativeai as genai
from google.colab import userdata

GOOGLE_API_KEY = userdata.get('GOOGLE_API_KEY')
genai.configure(api_key=GOOGLE_API_KEY)
```

Siguiendo con la arquitectura del sistema bastaría con modificar en el archivo `config.py` el valor de la `API KEY`.

### 5.1.3. Puesta en marcha

Una vez hechas todas las configuraciones necesarias explicadas en las secciones anteriores, para poder usar el chatbot sería necesario poner en ejecución el módulo `mibot.py` y enviar el comando `\start` al usuario `@mavice07_bot`. Es importante recordar, que para que el análisis de la información y el flujo de la conversación se desarrollen correctamente, hay que estar conectado a una VPN de una de las localizaciones en las que se encuentra disponible la API de gemini, y que como ya se ha comentado con anterioridad, se pueden consultar en la web de Gemini.

## 5.2. Arquitectura del sistema

Con todo lo implementado, el sistema se puede representar como se ve en la figura 5.1.

### 5.2.1. Servidor

El módulo servidor se encarga de manejar el flujo de los mensajes del chatbot. Recibe los mensajes entrantes desde la plataforma de Telegram y coordinar su procesamiento posterior. Este módulo actúa como el punto de entrada centralizado del

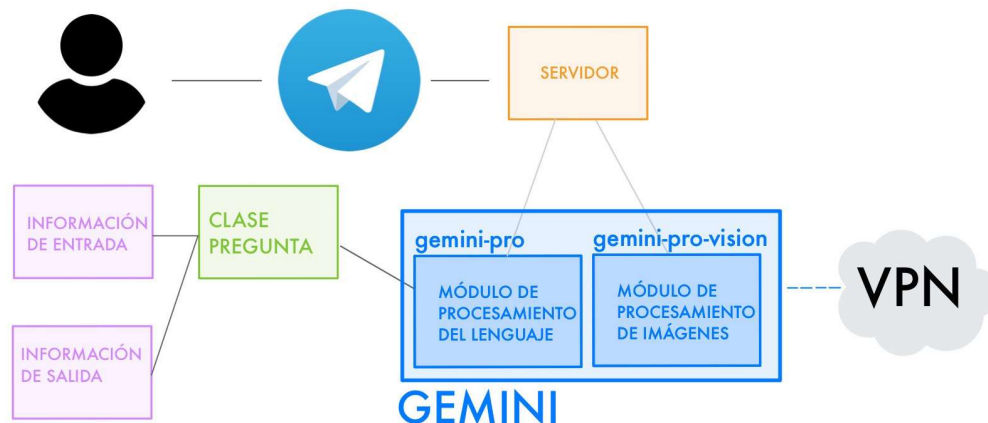


Figura 5.1: Arquitectura del sistema

sistema, y ha de mantenerse en ejecución durante todo el tiempo en el que se quiera usar el chatbot.

La lógica del módulo para manejar los mensajes sigue lo explicado en la sección 4.8.2, utilizando distintos manejadores según el tipo de mensaje. Esta capacidad de coordinación del módulo servidor asegura una distribución eficiente de los mensajes entrantes optimizando así la velocidad de respuesta y la efectividad de las interacciones con los usuarios.

El módulo servidor sirve como el núcleo activo del chatbot, recibiendo mensajes desde Telegram y dirigiéndolos de manera inteligente hacia los componentes adecuados del sistema para su procesamiento y respuesta. Su función es fundamental para mantener el flujo de comunicación entre los usuarios y el chatbot, facilitando una experiencia de usuario fluida y receptiva.

### 5.2.2. Procesamiento

El módulo del procesamiento de la información es el que usa Gemini, y que por tanto requiere estar conectado a una VPN. En función del tipo de mensaje de entrada, imagen o texto, el servidor enviará la información al módulo de procesamiento del lenguaje o al módulo del procesamiento de imágenes.

#### 5.2.2.1. Módulo de procesado de la información

Este módulo reúne la mayoría de la funcionalidad del chatbot. En concreto se encarga de las siguientes tareas:

- Análisis de las respuestas
- Identificación de la información faltante
- Flujo de las preguntas
- Generación de los json con la información
- Generación de preguntas adicionales
- Generación de la historia de vida final

Todas estas funcionalidades ya se han explicado con más detalle en el capítulo 4. La funcionalidad adicional que aparece es generación de la historia de vida final. Esta funcionalidad no es un objetivo en sí del sistema, si no que se crea, a raíz de toda la información obtenida y utilizando el Gemini para dar un feedback final al usuario, a modo de recopilación de la información.

#### 5.2.2.2. Módulo de procesado de imágenes

Este módulo se encarga de cargar las imagenes que se han enviado a Telegram. Por las características del modelo y la arquitectura del sistema, cuando la imagen se envía a través de Telegram el manejador descarga la imagen y la almacena con un determinado nombre, que pasa como atributo al módulo de procesado de imágenes. Una vez hecha la llamada al procesado de imágenes el modelo carga la imagen, la analiza y genera una respuesta a partir de la propia imagen y de un prompt adecuado que en concreto, lo que le pide es que la describa de forma que haga al usuario alguna pregunta. De esta forma, se busca que las imagenes sirvan para obtener información adicional que con solo texto quizás el usuario no hubiera recordado.

#### 5.2.3. Almacenamiento y manejo de la información

El manejo de la información se lleva a cabo a través de ficheros. Cuando el servidor recibe el comando `\start` carga y crea las preguntas del fichero *preguntas.txt*. Una vez llevada a cabo toda la conversación con el usuario, la información obtenida y la información generada se guardan en el fichero de salida *información.txt*

### 5.3. Interfaz e interacción con el usuario

A pesar de que se podría usar a través de su versión por consola, el chatbot esta pensando para ser usado a través de Telegram. Gracias a esto se puede usar desde tantos dispositivos como se puede usar Telegram, entre los que destacaríamos los dispositivos móviles y los ordenadores.

Para poner en marcha la versión por ordenador, es necesario la aplicación de Telegram, ya sea la versión web o la versión local. De igual forma la versión en móvil

u tablet requiere la aplicación de Telegram. Una vez con la aplicación correctamente instalada y abierta, hay que buscar al usuario `@mavice07_bot` y enviado el comando `/start`, recibiremos el mensaje de bienvenida y comenzará la conversación. Ejemplo de esto en la versión de ordenador y móvil se pueden ver respectivamente en las figuras 5.2 y 5.3.

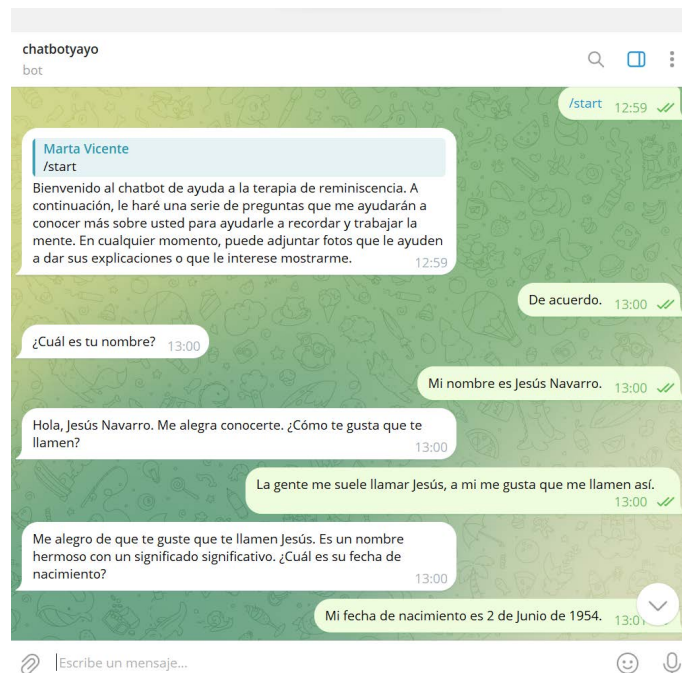


Figura 5.2: Ejemplo de bienvenida y primeras interacciones con la versión en ordenador

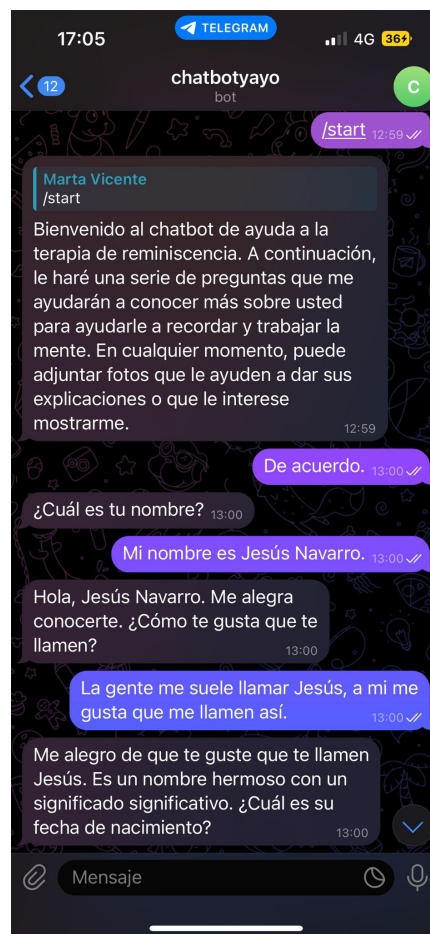


Figura 5.3: Ejemplo de bienvenida y primeras interacciones con la versión en móvil



## Conclusiones y Trabajo Futuro

En el presente capítulo se describen las conclusiones a las que se han llegado tras meses de trabajo en un proyecto de construcción de una herramienta conversacional de ayuda a la terapia de reminiscencia. Pese al desarrollo exponencial que esta sufriendo la IA en el campo del procesamiento del lenguaje, la provisionalidad de los modelos y los constantes cambios en las legislaciones de los territorios han llevado a diversos problemas y en consecuencia a un estudio mucho más amplio de la situación actual y de las diferentes alternativas.

En este capítulo, no solo se pretende hablar de las conclusiones acerca de la herramienta desarrollada, si no a todas las conclusiones asociadas al estudio teórico previo.

Por otro lado, se describirán los posibles trabajos futuros que pueden considerarse como fruto de la investigación.

### 6.1. Estado actual del procesamiento del lenguaje

En los últimos años, el avance del procesamiento del lenguaje ha sido notable, con numerosas ventajas, aunque también con riesgos asociados al desarrollo de software.

Por un lado, los modelos de inteligencia artificial evolucionan rápidamente, generando frecuentes actualizaciones etiquetadas como "última versión estable". Esta denominación se refiere a la versión más reciente del modelo que ha sido probada y validada para su uso generalizado, ofreciendo calidad y fiabilidad ?.

Por otro lado, la IAs se enfrentan al problema de la regulación. La presidenta de la Comisión Europea, Ursula von der Leyen, ha subrayado que la IA está transformando nuestras vidas y ha enfatizado la importancia de un enfoque sensato y generalizado para beneficiar a la economía y la sociedad Comisión Europea (2023). La recién aprobada Ley de IA de la UE, considerada como el primer marco global en esta materia, busca promover la innovación responsable al regular los riesgos identificados y garantizar la seguridad y los derechos fundamentales de las personas y las

empresas.

Este enfoque se basa en evaluar el riesgo de los sistemas de IA: aquellos de bajo riesgo, como los sistemas de recomendación, tendrán libertad y ninguna obligación, mientras que los de alto riesgo, como infraestructuras críticas, sistemas de salud y aplicaciones policiales, deberán cumplir requisitos estrictos de mitigación, calidad de datos y supervisión humana. Además, se prohibirán los sistemas de IA que representen una amenaza clara para los derechos fundamentales, como la manipulación del comportamiento humano. Se introducirán normas específicas para garantizar la transparencia en los modelos de IA de uso general, junto con multas para las empresas que no cumplan con las regulaciones.

En diciembre de 2023, se elaboró un borrador sobre la regulación de la IA por parte del Consejo de la Unión Europea y el Parlamento Europeo. Finalmente, el 2 de febrero de 2024, se aprobó la primera ley del mundo sobre inteligencia artificial: la Ley de IA. Después de una reunión en Bruselas, los embajadores de los 27 Estados miembros dieron su visto bueno político a esta normativa, tras la presentación en enero de la versión final del texto y la creación de la Oficina Europea de Inteligencia Artificial. Aunque algunos países mostraron oposición hasta el último momento, el camino continuó su curso y se prevé que entre 2024 y 2030 todos los países adopten la Ley de IA [ElDerecho.com](https://elderecho.com) (11-04-2024).

Los constantes cambios en los modelos y las modificaciones en las legislaciones nacionales suponen un desafío para los desarrolladores de software. Durante el desarrollo de este proyecto, nos enfrentamos a diversas situaciones derivadas de estos sucesos. Por un lado, tuvimos que buscar una alternativa a la API de Bard, que se transformó en Gemini entre diciembre de 2023 y febrero de 2024, y por otro lado, hacer frente a las limitaciones impuestas por las diferentes leyes, que se solvento mediante el uso de VPN.

De hecho, la situación cambia tan rápidamente, que en la recta final del proyecto hemos observado varios cambios en los términos y condiciones de uso de la API de Gemini a nivel mundial. De hecho, la próxima actualización de los términos de uso será el 22 de mayo de 2024.

En conclusión, en el campo del desarrollo del lenguaje y la extracción de información a partir de imágenes, la IA está experimentando numerosos avances en los últimos años. Sin embargo, es fundamental tener en cuenta que el uso de los distintos modelos está sujeto a cambios tanto en las versiones y sus características, como en las leyes y términos y condiciones de uso.

## 6.2. Desarrollo del proyecto

Para comenzar a desarrollar este trabajo, el primer paso es tener claros cuales son los objetivos y requisitos del mismo. Analizar la problemática en relación con otros trabajos ya existentes para desarrollar una herramienta útil.

Después de establecer los objetivos y requisitos de nuestro proyecto de forma

general, el siguiente paso es plantearlos de manera más técnica y detallada. Para ello, diseñamos una arquitectura basada en las diversas tareas que el sistema debía llevar a cabo. Sin embargo, antes de implementar el sistema, surgieron algunas cuestiones relacionadas con ciertos componentes de la arquitectura.

Principalmente, debíamos resolver la selección adecuada del modelo de lenguaje a utilizar, dada la amplia variedad de modelos disponibles. Asimismo, necesitábamos asegurar el acceso a un conjunto de datos adecuado y relevante para los objetivos de transformación de datos biográficos, que permitiera ajustar el modelo de manera efectiva. Para el desarrollo del núcleo principal del sistema, el módulo encargado del procesamiento de la información entrante, se desarrollaron numerosos prototipos de los cuáles los más importantes se explican en el capítulo 4.

El principal reto de este proyecto ha sido sin duda enfrentarse a todos esos cambios de versiones, actualizaciones y cambios en las legislaciones que alteraban el funcionamiento del modelo. Gran parte del tiempo dedicado a este trabajo ha estado enfocado en la investigación de las diferentes herramientas del procesamiento del lenguaje.

Una vez seleccionado el modelo definitivo, Gemini, y llevado a cabo todo lo necesario para su puesta en marcha comienza la etapa central del desarrollo siguiendo el plan de diseño incremental explicado 1.2.

Finalmente, se procedió a la aplicación del modelo para obtener resultados y analizar su comportamiento.

## 6.3. Resultado final

Como resultado del desarrollo de este proyecto, se ha creado una herramienta conversacional para apoyar la terapia de reminiscencia. La principal función de este bot es buscar y almacenar información relevante para generar historias de vida utilizando otras herramientas. La interfaz de usuario intuitiva y fácil de usar hace que esta herramienta sea accesible no solo para terapeutas, sino también para familiares y pacientes que estén familiarizados con dispositivos móviles y computadoras.

Aunque inicialmente diseñada como una herramienta terapéutica para recopilar información sobre la vida del paciente, esta herramienta tiene múltiples usos. Por ejemplo, las versiones portátiles, como la aplicación móvil o para tabletas, permiten su uso como entretenimiento durante viajes largos que pueden resultar desafiantes para los pacientes. Además, la facilidad de uso también permite que los pacientes la utilicen en momentos de soledad, manteniendo conversaciones interesantes sobre su vida que estimulan la mente y les proporcionan compañía virtual.

## 6.4. Trabajo futuro

Como punto final de este trabajo se exponen aquellas posibles líneas abiertas de desarrollo e investigación, que han surgido tras el análisis del proyecto, y que podrían ser estudiados con la finalidad de mejorar el proyecto.

En primer lugar, aunque este trabajo genere una pequeña historia de vida como resultado final de la conversación sería interesante conectar este trabajo con el proyecto explicado en la sección 2.6.1.1 “Generación de historias de vida usando técnicas de Deep Learning”. Para ello sería interesante transformar toda la información que se obtiene mediante la conversación con este chatbot en un conjunto de entrada con el formato adecuado para la aplicación desarrollada en ese trabajo. De esta forma, se generarían historias de vida más fieles a la vida real de la persona, y se completaría el ciclo completo de trabajo previo a la terapia de reminiscencia.

Por otro lado, otra modificación que podría resultar interesante sería la adaptación del modelo a múltiples lenguajes de forma que se amplié el público objetivo de la aplicación. La API de Gemini está disponible en diferentes lenguajes, pero el manejo de todas las preguntas predefinidas tendría que ser traducido para la interacción con el usuario.

El desarrollo de la API de Gemini, y el acceso de nuevas versiones traerá consigo nuevas posibilidades. Entre ellas, destaca la integración de vídeos para enviar al chatbot y que se analicen con el objetivo de obtener más información bibliográfica. También puede ser interesante considerar la alternativa del reconocimiento de voz de forma que además de la interfaz de Telegram, se pudiera desarrollar una interfaz de voz donde el paciente pudiera hablar y escuchar como si se tratara de una llamada telefónica.

# Conclusions and Future Work

This chapter describes the conclusions reached after months of work on a project to build a conversational tool to help reminiscence therapy. Despite the exponential development that AI is undergoing in the field of language processing, the provisional nature of the models and the constant changes in the legislation of the territories have led to various problems and consequently to a much broader study of the current situation and the different alternatives.

In this chapter, we do not only intend to talk about the conclusions about the developed tool, but also about all the conclusions associated with the previous theoretical study.

On the other hand, possible future works that can be considered as a result of the research will be described.

## 6.5. Actual state of PLN

In recent years, the progress in language processing has been notable, with numerous advantages, but also with risks associated with software development.

On the one hand, artificial intelligence models evolve rapidly, generating frequent updates labeled as "latest stable version." This designation refers to the most recent version of the model that has been tested and validated for widespread use, offering quality and reliability ?.

On the other hand, AIs face the problem of regulation. The President of the European Commission, Ursula von der Leyen, has underlined that AI is transforming our lives and has emphasized the importance of a sensible and widespread approach to benefit the economy and society Comisión Europea (2023). The recently approved EU AI Law, considered the first global framework on this matter, seeks to promote responsible innovation by regulating identified risks and guaranteeing the safety and fundamental rights of people and companies.

This approach is based on assessing the risk of AI systems: those with low risk, such as recommendation systems, will have freedom and no obligation, while those with high risk, such as critical infrastructures, health systems and police ap-

plications, will have to meet strict mitigation, data quality, and human oversight requirements. In addition, AI systems that pose a clear threat to fundamental rights, such as the manipulation of human behavior, will be banned. Specific rules will be introduced to ensure transparency in general-purpose AI models, along with fines for companies that do not comply with regulations.

In December 2023, a draft on the regulation of AI by the Council of the European Union and the European Parliament was prepared. Finally, on February 2, 2024, first law on artificial intelligence was passed, the AI Act. After a meeting in Brussels, the ambassadors of the 27 Member States gave their political approval to this regulation, following the presentation in January of the final version of the text and the creation of the European Artificial Intelligence Office. Although some countries showed opposition until the last moment, the path continued its course and it is expected that between 2024 and 2030 all countries will adopt the law ElDerecho.com (11-04-2024).

Constant changes in models and modifications in national legislation pose a challenge for software developers. During the development of this project, we faced various situations derived from these events. On the one hand, we had to look for an alternative to the Bard API, which was transformed into Gemini between December 2023 and February 2024, and on the other hand, face the limitations imposed by the different laws, which were solved through the VPN use.

In fact, the situation is changing so quickly that in the final stretch of the project we have observed several changes in the terms and conditions of use of the Gemini API worldwide. In fact, the next update of the terms of use will be May 22, 2024.

In conclusion, in the field of language development and information extraction from images, AI is experiencing numerous advances in recent years. However, it is essential to keep in mind that the use of the different models is subject to changes both in the versions and their characteristics, as well as in the laws and terms and conditions of use.

## 6.6. Project development

To begin developing this work, the first step is to be clear about its objectives and requirements. Analyze the problem in relation to other existing work to develop a useful tool.

After establishing the objectives and requirements of our project in a general way, the next step is to present them in a more technical and detailed way. To do this, we designed an architecture based on the various tasks that the system had to carry out. However, before implementing the system, some issues related to certain components of the architecture arose.

Mainly, we had to resolve the appropriate selection of the language model to use, given the wide variety of models available. Additionally, we needed to ensure access to an adequate data set relevant to the biodata transformation objectives,

allowing the model to be adjusted effectively. For the development of the main core of the system, the module responsible for processing incoming information, numerous prototypes were developed, the most important of which are explained in the chapter 4.

The main challenge of this project has undoubtedly been facing all those version changes, updates and changes in legislation that altered the operation of the model. Much of the time dedicated to this work has been focused on research into different language processing tools.

Once the final model, Gemini, has been selected and everything necessary for its implementation has been carried out, the central stage of development begins following the explained incremental design plan ??.

Finally, the model was applied to obtain results and analyze its behavior.

## 6.7. Final result

As a result of the development of this project, a conversational tool has been created to support reminiscence therapy. The main function of this bot is to search and store relevant information to generate life stories using other tools. The intuitive and easy-to-use user interface makes this tool accessible not only to therapists, but also to family members and patients who are familiar with mobile devices and computers.

Although initially designed as a therapeutic tool to gather information about the patient's life, this tool has multiple uses. For example, portable versions, such as the mobile or tablet app, allow for use as entertainment during long trips that can be challenging for patients. In addition, the ease of use also allows patients to use it in moments of solitude, having interesting conversations about their life that stimulate the mind and provide them with virtual companionship.

## 6.8. Future work

As a final point of this work, those possible open lines of development and research are exposed, which have emerged after the analysis of the project, and which could be studied with the aim of improving the project.

Firstly, although this work generates a small life story as the final result of the conversation, it would be interesting to connect this work with the project explained in the section 2.6.1.1 "Generation of life stories using Deep Learning techniques". To do this, it would be interesting to transform all the information obtained through the conversation with this chatbot into an input set with the appropriate format for the application developed in that work. In this way, life stories that are more faithful to the person's real life would be generated, and the full cycle of work prior to reminiscence therapy would be completed.

On the other hand, another modification that could be interesting would be the adaptation of the model to multiples languages, in order to expand the public target. The Gemini API is available in different languages, but the handling of all predefined questions would have to be translated for user interaction.

The development of the Gemini API, and the access of new versions will bring new possibilities. Among them, the integration of videos to send to the chatbot stands out, which are analyzed with the aim of obtaining more bibliographic information. It may also be interesting to consider the alternative of voice recognition so that in addition to the Telegram interface, a voice interface could be developed where the patient could speak and listen as if it were a phone call.



# Bibliografía

- ACHIAM, J., ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L. y ALMEIDA, D. Gpt-4 technical report. *arXiv*, Disponible en <https://arxiv.org/abs/2303.08774>.
- AGUILERA HEREDERO, P. y MOLINA MUÑOZ, C. *Extracción de información personal a partir de redes sociales para la creación de un libro de vida*. Proyecto Fin de Carrera, Facultad de Informática, Universidad Complutense de Madrid, 2021.
- BARQUILLA BLANCO, C., DÍEZ GARCÍA, P., MARCO MULAS LÓPEZ, S. y VERDÚ RODRÍGUEZ, E. Recuérdame 1.0: Aplicación de apoyo para el tratamiento de personas con problemas de memoria mediante terapias basadas en reminiscencia. 2022. Trabajo de Fin de Grado, Universidad Complutense de Madrid.
- BIRD, S., KLEIN, E. y LOPER, E. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009. ISBN 9780596516499.
- BOTO, A., PORTILLO TORRES, A., SANZ MAYO, D. y PORTILLO TORRES, R. *Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning*. Proyecto Fin de Carrera, Facultad de Informática, Universidad Complutense de Madrid, 2021.
- CAÑETE, J., CHAPERON, G., FUENTES, R., HO, J.-H., KANG, H. y PÉREZ, J. Spanish pretrained bert model and evaluation data. En *PML4DC at ICLR 2020*. 2020.
- CHAMPIN, P.-A. Rdf tutorial. *arxiv*, 2002.
- CHUNG, J. A. y CUMMINGS, J. L. Neurobehavioral and neuropsychiatric symptoms in alzheimer's disease: characteristics and treatment. *Neurologic clinics*, vol. 18, páginas 829–846, 2000.
- COMISIÓN EUROPEA. La comisión se congratula del acuerdo político sobre la ley de inteligencia artificial. Comunicado de prensa, 2023.
- COTELLI, M., MANENTI, R. y ZANETTI, O. Reminiscence therapy in dementia: a review. *Maturitas*, vol. 72, páginas 203–205, 2012.

- DEVLIN, J., CHANG, M.-W., LEE, K. y TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. En *NAACL*. 2019.
- DONOSO, A. La enfermedad de alzheimer. *Revista Chilena de Neuro-psiquiatría*, vol. 41(supl.2), 2003. ISSN 0717-9227. ARTÍCULO ORIGINAL.
- DÍEZ SOBRINO, C., GUERRERO SOSA, E., PRIETO CAMPO, A., MARTÍNEZ GONZÁLEZ, P. y ARIAS RODRÍGUEZ-PEÑA, S. Recuérdame 2.0: Mejorando una aplicación de apoyo para el tratamiento de personas con problemas de memoria mediante terapias basadas en reminiscencia. 2023. Trabajo de Fin de Grado, Universidad Complutense de Madrid.
- EDWARDS, B. Why chatgpt and bing chat are so good at making things up. *Ars Technica*, 2023.
- ELDERECHO.COM. Ley de ia: plazos y fomento de la inversión en europa. 11-04-2024.
- HUANG, H.-C., CHEN, Y.-T., CHEN, P.-Y., HU, H.-L. S., LIU, F., KUO, Y.-L. y ET AL. Reminiscence therapy improves cognitive functions and reduces depressive symptoms in elderly people with dementia: a meta-analysis of randomized controlled trials. *Journal of the American Medical Directors Association*, vol. 16, páginas 1087–1094, 2015.
- IRAZOKI, E., GARCÍA-CASAL, J. A., SÁNCHEZ-MECA, J. y FRANCO-MARTÍN, M. Eficacia de la terapia de reminiscencia grupal en personas con demencia. revisión sistemática y metaanálisis. *Revista de neurología*, 2017.
- JI, Z., LEE, N., FRIESKE, R., YU, T., SU, D., XU, Y., ISHII, E. y BANG, Y. Survey of hallucination in natural language generation. *ACM Computing Surveys*, vol. 55(12), páginas 1–38, 2022.
- KUDO, T. y RICHARDSON, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
- LESWING, K. Microsoft's bing a.i. made several factual errors in last week's launch demo. *CNBC*, 2023.
- LIN, C. How to easily trick openai's genius new chatgpt. *Fast Company*, 2022.
- MACHIN-MASTROMATTEO, J. D. Implicaciones y políticas editoriales de la inteligencia artificial. *arxiv*, 2023.
- MAGAZ, L. L. Grado en ingeniería informática. 2023. Curso 2022-2023.
- MIKOLOV, T., SUTSKEVER, I., K. CHEN, G. C., y DEAN, J. Distributed representations of words and phrases and their compositionality. *arXiv*, Disponible en <https://doi.org/10.48550/arXiv.1310.4546>.

- RADFORD, A., NARASIMHAN, K., SALIMANS, T. y SUTSKEVER, I. Improving language understanding by generative pre-training. OpenAI, 2018. Archived (PDF) from the original on 26 January 2021. Retrieved 23 January 2021.
- RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D. y SUTSKEVER, I. Language models are unsupervised multitask learners. 2019.
- RAFFEL, C., SHAZEER, N., ROBERTS, A., LEE, K., NARANG, S., MATENA, M., ZHOU, Y., LI, W. y LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21 (2020) 1-67, vol. 2020, Disponible en <https://jmlr.org/papers/volume21/20-074/20-074.pdf>.
- RAJASEKHARAN, A. A review of bert based models. 2019.
- ROHRBACH, A., HENDRICKS, L. A., BURNS, K., DARRELL, T. y SAENKO, K. Object hallucination in image captioning. *UC Berkeley, Boston University*, 2023.
- ROTHMAN, D. *Transformers for Natural Language Processing*. Packt Publishing, 2nd edición, 2022.
- SAGE. Chatgpt and generative ai: Use of large language models and generative ai tools in writing your submission. Online, 2023.
- SALAS, C. A. Generacion de historias de vida usando técnicas de deep learning. Disponible en <https://docta.ucm.es/entities/publication/28458f98-f632-4b80-a883-a4c3169f9dbd>.
- DA SILVA, P. P., SUGIURA, N., SIMPERL, E., WALLACE, E. y HAN, H. 6th international semantic web conference, 2nd asian semantic web conference, ISWC 2007 + ASWC 2007, busan, korea, november 11-15, 2007, proceedings. 2007.
- SLOVIKOVSKAYA, V. Transfer learning from transformers to fake news challenge stance detection (fnc-1) task. *arXiv preprint arXiv:1910.14353*, 2019.
- SPINAK, E. ¿es que la inteligencia artificial tiene alucinaciones? *SciELO en Perspectiva*, 2023.
- TOUVRON, H., LAVRIL, T., IZACARD, G., MARTINET, X., LACHAUX, M.-A., LACROIX, T., ROZIÈRE, B., GOYAL, N., HAMBRO, E., AZHAR, F., RODRIGUEZ, A., JOULIN, A., GRAVE, E. y LAMPLE, G. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- TU, X., LAI, K. y YANUSHKEVICH, S. Transfer learning on convolutional neural networks for dog identification. En *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, páginas 357–360. IEEE, 2018.
- UK, D. *Life story template*. Packt Publishing, 2023.

- VASWANI, A. SHAZEER, N. PARMAR, N. USZKOREIT, JONES, J., L. GOMEZ, A. N., K. y POLOSUKHIN. Advances in neural information processing systems. *arxiv*, vol. 30, 2023.
- WANG, A. y CHO, K. Bert has a mouth, and it must speak: Bert as a markov random field language model. En *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, páginas 30–36. Association for Computational Linguistics, Minneapolis, Minnesota, 2019.
- WIKIPEDIA. Alucinación (inteligencia artificial) - wikipedia. Fecha de última modificación.
- YANG, J., JIN, H., TANG, R., HAN, X., FENG, Q., JIANG, H., YIN, B. y HU, X. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *arXiv*, 2023.
- ZHU, Y., KIROS, R., ZEMEL, R., SALAKHUTDINOV, R., URTASUN, R., TORRALBA, A. y FIDLER, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. En *The IEEE International Conference on Computer Vision (ICCV)*. 2015.
- ZHUO, T. Y., HUANG, Y., CHEN, C. y XING, Z. Exploring ai ethics of chatgpt: A diagnostic analysis. 2023.