
Chatbot para la recuperación de información personal



Trabajo de Fin de Grado
Curso 2022–2023

Autora
Lucía Latorre Magaz

Directores
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Chatbot para la recuperación de información personal

Trabajo de Fin de Grado en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia Artificial

Autora
Lucía Latorre Magaz

Directores
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Dirigida por el Doctor
Gonzalo Méndez Pozo
Pablo Gervás Gómez-Navarro

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

24 de enero de 2023

Agradecimientos

Texto de los agradecimientos

Resumen

Este trabajo se centra en el desarrollo de un Chatbot que recopile y almacene los recuerdos de una persona con demencia. El objetivo es facilitar la tarea de los terapeutas con ese paso previo a la construcción de la historia de vida de un paciente. Se trata de una herramienta de apoyo de cara a las terapias basadas en reminiscencia.

El chatbot se ha programado en base a tres componentes principales. El primero es un analizador de sentimientos que detecta si un texto tiene connotaciones negativas o positivas. En segundo lugar se encuentra el clasificador de recuerdos en etapas de vida que indica si el texto se corresponde con un recuerdo de la infancia, de la juventud, de la etapa adulta o de la vejez. El último componente es un módulo que elige cuál es la mejor siguiente pregunta a formular, la que más relación tenga con la respuesta que haya dado el usuario.

Para que la interacción con el Chatbot sea atractiva, visual y fácil, se ha creado una aplicación web complementaria dirigida tanto a terapeutas como a pacientes. Aunque el foco principal de la aplicación sea el bot conversacional, también se ofrecen otras funcionalidades como la posibilidad de revisar los recuerdos ya almacenados o la de crear nuevas terapias.

El código desarrollado en el proyecto puede encontrarse en <https://github.com/NILGroup/TFG-2122-Chatbot>.

Palabras clave

Chatbot, aplicación web, demencia, recuerdo, terapia, historia de vida

Abstract

This project focuses on the development of a Chatbot that collects and stores the memories of people who suffer dementia. The main goal is to facilitate the therapist's job by helping with the steps prior to shaping a patient's life story. It's a support tool for reminiscence-based therapies.

The Chatbot has been programmed considering three main components. The first one is a sentiment analyser that detects whether the text contains negative or positive connotations. Secondly, we have a memory classifier that indicates whether the text corresponds to a memory from the childhood, youth, adulthood or old age. The last component is a module that chooses the next best question to be asked, the one that is most related to the answer given by the user.

To make the interaction with the Chatbot attractive, visual and easy, a complementary web application has been created for both therapists and patients. Although the main focus of the app is the conversational bot, other functionalities are also offered, such as the possibility of reviewing the already stored memories or creating new therapies.

The code developed throughout the project can be found at <https://github.com/NILGroup/TFG-2122-Chatbot>.

Keywords

Chatbot, web application, dementia, memory, therapy, life stories

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Plan de trabajo	2
1.4. Estructura de la memoria	2
2. Introduction	5
2.1. Motivation	5
2.2. Objectives	5
2.3. Work plan	6
2.4. Document structure	6
2. Estado de la Cuestión	9
2.1. Demencia	9
2.2. Terapia ocupacional basada en reminiscencia	9
2.3. Trabajos previos	10
2.4. Procesamiento del lenguaje natural	11
2.4.1. Modelos para el PLN	11
2.4.2. Componentes del PLN	11
2.4.3. Aplicaciones del procesamiento del lenguaje natural	12
2.4.4. Ventajas del PLN	12
2.4.5. Análisis de sentimiento	13
2.5. Chatbots	13
2.5.1. Tipos	14
2.5.2. Herramientas de desarrollo	14
3. Arquitectura del Chatbot	17
3.1. Repertorio de preguntas	17
3.2. Clasificación de los recuerdos	18
3.2.1. Pruebas del análisis de sentimiento	22
3.2.2. Clasificación en etapas de vida	25
3.3. Procesamiento del texto para encadenar preguntas y respuestas	29
3.3.1. Ejemplo de funcionamiento de la selección de la siguiente pregunta a realizar	29

3.4. Base de datos MongoDB	32
4. Aplicación Web	35
4.1. Prototipo inicial	35
4.2. Base de datos MySQL	39
4.2.1. Tabla de usuarios	39
4.2.2. Tabla de terapias	40
4.3. Implementación final	41
4.3.1. Funcionalidad inicio de sesión y registro	43
4.3.2. Funcionalidad chatbot	44
4.3.3. Funcionalidad usuario terapeuta vs usuario paciente	45
4.3.4. Funcionalidad consulta de pacientes registrados por parte del terapeuta	46
4.3.5. Funcionalidad crear nuevo paciente	46
4.3.6. Funcionalidad recopilación de los recuerdos de un paciente	47
5. Conclusiones y Trabajo Futuro	51
5. Conclusions and Future Work	53
Bibliografía	55

Índice de figuras

3.1. Diagrama de flujo del categorizador	20
3.2. Entrenamiento del modelo	21
3.3. Documento de la colección de preguntas	33
3.4. Documento de la colección de respuestas	34
4.1. Prototipo inicial de la aplicación para el usuario con demencia	36
4.2. Prototipo inicial de la aplicación para el terapeuta: Consultar los pacientes registrados	36
4.3. Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 1	37
4.4. Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 2	37
4.5. Prototipo inicial de la aplicación para el terapeuta: Crear y consultar las terapias creadas	38
4.6. Prototipo inicial de la aplicación para el terapeuta: Consultar y reproducir una terapia concreta	38
4.7. Diagrama de entidad-relación MySQL	40
4.8. Diagrama de la tabla de usuarios	41
4.9. Diagrama de la tabla de terapias	42
4.10. Funcionalidad inicio de sesión	43
4.11. Funcionalidad registro	44
4.12. Funcionalidad chatbot	45
4.13. Menú de los pacientes	46
4.14. Menú de los terapeutas	46
4.15. Funcionalidad consultar pacientes registrados	47
4.16. Tabla de terapias en MySQL	48
4.17. Funcionalidad crear nuevo paciente	49
4.18. Funcionalidad recopilación de recuerdos del paciente	50

Capítulo 1

Introducción

“Conozca todas las teorías, domine todas las técnicas, pero al tocar un alma humana sea simplemente otra alma humana”

— Carl Gustav Jung

Las personas con Alzheimer u otros tipos de demencia pueden beneficiarse del uso de la llamada terapia basada en reminiscencia. Se basa en la construcción de un libro de vida del paciente que recopila recuerdos positivos de su vida. Esta información se puede utilizar posteriormente para ejercitarse la memoria y retrasar el deterioro de la enfermedad. Además, permite aumentar el bienestar de los pacientes.

En el presente proyecto se propone el desarrollo de un chatbot que permita recopilar y estructurar esta información para ayudar a los terapeutas en la elaboración de los libros de vida.

1.1. Motivación

Las terapias basadas en reminiscencia¹ favorecen la evocación de recuerdos mediante la estimulación, comunicación y entrenamiento con el objetivo de preservar todo lo posible la reserva cognitiva y el sentido de identidad. Para incitar al paciente a rememorar ciertos recuerdos es muy importante que el terapeuta tenga en cuenta la historia de vida de esa persona y entender el contexto en el que cada recuerdo se enmarca. Un terapeuta no puede trabajar un recuerdo con el usuario sin tener en cuenta las consecuencias que puedan suponer para esa persona, no solo por el hecho de hacer referencia a él sino también la forma de tratarlo. El paciente puede reaccionar de muchas formas pudiendo llegar a tristezas, enfadarse, etc. agravando así su situación.

Sabiendo la importancia que tiene la historia de vida en este tipo de terapias, los terapeutas se encargan de narrarla, proceso que requiere mucho trabajo y que suelen hacer manualmente. Esto implica recoger los datos de la persona para luego transformarlos en un relato biográfico. Poder acelerar este proceso mediante la recopilación automática de esa información ahorra mucho tiempo y esfuerzo a los terapeutas que pueden dedicarlo al trabajo con el paciente para ralentizar su deterioro.

¹<https://orpea.es/terapia-de-reminiscencia-para-personas-con-demencia/>

1.2. Objetivos

Este proyecto tiene como objetivo desarrollar un chatbot con el que recabar información personal sobre la vida de la persona con demencia, clasificarla y estructurarla siguiendo un esquema que pueda facilitar la tarea de los terapeutas a la hora de construir un libro de vida.

Este objetivo principal se divide en los siguientes objetivos específicos:

- Creación de un chatbot que recoja los recuerdos de los usuarios.
- Creación de una interfaz conversacional de un chatbot.
- Clasificación de los recuerdos en base a unos criterios predefinidos por expertos en terapia ocupacional. Se categorizarán en función de:
 - **Emoción:** Se clasificarán los recuerdos en positivos y negativos. Es importante identificar y evitar los recuerdos negativos en las terapias para no afligir al paciente.
 - **Etapa:** Los recuerdos pertenecerán a una de las siguientes etapas: infancia, juventud, edad adulta o tercera edad según el periodo temporal en que acontecieron.
 - **Categorías:** Cada recuerdo entrará dentro de una o varias categorías que recojan una característica del recuerdo. Ejemplos de categorías: guerra civil, bailes, ocio, familia, aficiones, amigos...
- Almacenamiento de los recuerdos estructurados y clasificados en una base de datos .
- Desarrollo de una aplicación web que permita al usuario interactuar con el chatbot a través de una interfaz gráfica atractiva y sencilla de usar.

1.3. Plan de trabajo

Con el fin de cumplir con los objetivos planteados en la sección anterior, se ha fijado una planificación dividida en cuatro etapas:

- Investigación sobre demencia y terapias ocupacionales basadas en reminiscencia. A nivel más técnico, investigación sobre herramientas existentes para el desarrollo de chatbots.
- Construcción de un prototipo inicial de análisis de textos y elección de tecnologías.
- Implementación del Chatbot y desarrollo de la memoria del TFG.
- Pruebas, revisión de esta memoria y entrega.

1.4. Estructura de la memoria

La presente memoria está compuesta por cinco capítulos, entre ellos esta introducción. Cada uno de estos capítulos se expone a continuación:

- **Capítulo 2:** En el estado de la cuestión se presenta el concepto de demencia y de las terapias ocupacionales basadas en reminiscencia. También se explican los diferentes trabajos previos que se han ido desarrollando en el marco del proyecto CANTOR. Se da a conocer la importancia del procesamiento del lenguaje natural y el contexto de los chatbots y cómo implementarlos.
- **Capítulo 3:** El apartado de la arquitectura del chatbot está centrado en cómo se ha desarrollado el bot conversacional de forma que pueda recopilar los recuerdos del usuario con demencia y almacenarlos de manera estructurada. Para ello, se explican los componentes del chatbot que son los siguientes: el módulo de preguntas, el módulo de clasificación de recuerdos y el módulo de elección de la mejor siguiente pregunta.
- **Capítulo 4:** En el capítulo de la aplicación web se explican los pasos que se han seguido para desarrollar la página web desde el principio con un prototipo inicial hasta el final con una aplicación web funcional y con distintas vistas. Entre medias se describe el uso de una base de datos MySQL.
- **Capítulo 5:** Se trata del capítulo de conclusiones y trabajo futuro en el que se recogen las conclusiones extraídas de los resultados obtenidos en este trabajo. También se detallan las posibles líneas de trabajo que deja abiertas este TFG para su continuación en un futuro.

Chapter 2

Introduction

“Know all the theories, master all the techniques, but as you touch a human soul be just another human soul”
— Carl Gustav Jung

People who suffer Alzheimer's or other types of dementia may benefit from the use of the so-called reminiscence-based therapy. It is based on the composition of a patient's life book that gathers positive memories from his or her life. This information can then be used to exercise their memory and delay the damage that causes the disease. It also helps increasing the patient's well-being.

In this project the main goal is to develop a chatbot to collect and structure this information. It helps the therapists in the elaboration of life books.

2.1. Motivation

Reminiscence-based therapies¹ encourage the evocation of memories through stimulation, communication and training with the aim of preserving as much cognitive reserve and sense of identity as possible. In order to prompt the patient to recall certain memories, it is very important for the therapist to consider the person's life history and to understand the context in which each memory is framed. A therapist cannot work on a memory with the user without taking into account the consequences for that person, not only by referring to it but also by the way it is dealt with. The patient may react in many ways and may become sad, angry, etc., thus aggravating his/her situation.

Knowing the importance of the life history in this type of therapy, therapists are in charge of narrating it, a process that requires a lot of work and which is usually done manually. This involves collecting the person's data and then transforming it into a biographical account. Being able to speed up this process by automatically collecting this information saves therapists a lot of time and effort that can better be spent on working with the patient to slow down their deterioration.

2.2. Objectives

This project aims to develop a chatbot with which to collect personal information about the life of the person with dementia, classify it and structure it according to a scheme that

¹<https://orpea.es/terapia-de-reminiscencia-para-personas-con-demencia/>

can facilitate the task of therapists when building a life book.

This main objective is divided into the following specific goals:

- Creation of a chatbot that collects users' memories.
- Creation of a chatbot conversational interface.
- Classification of memories based on criteria predefined by occupational therapy experts. They will be classified according to:
 - **Emotion** Memories shall be categorised into positive and negative. It is important to identify and avoid negative memories in therapy so as not to distress the patient.
 - **Stage:** The memories will belong to one of the following stages: childhood, youth, adulthood or old age depending on the time period in which they occurred.
 - **Categories:** Each memory will fall into one or more categories that capture a characteristic of the memory. Examples of categories: civil war, dance, leisure, family, hobbies, friends...
- Storage of memories structured and classified in a database.
- Development of a web application that allows the user to interact with the chatbot through an attractive and easy-to-use graphical interface.

2.3. Work plan

In order to meet the objectives set out in the previous section, a plan divided into four stages has been established:

- Research on dementia and reminiscence-based occupational therapies. On a more technical level, research on existing tools for the development of chatbots.
- Construction of an initial prototype for text analysis and choice of technologies.
- Chatbot implementation and development of this document.
- Testing, document's review and turn in the project.

2.4. Document structure

This document is composed of five chapters, including this introduction. Each of these chapters is presented below:

- **Chapter 2:** The state of the art presents the concept of dementia and reminiscence-based occupational therapies. It also explains the different previous projects that have been developed in the context of the CANTOR project. Furthermore, it illustrates the importance of natural language processing and the context of chatbots and how to implement them.

- **Chapter 3:** The section on the architecture of the chatbot focuses on how the conversational bot has been developed in such a way that it can collect memories of the user with dementia and store them in a structured manner. For this purpose, the components of the chatbot are explained in this chapter. They are the following: the question's module, the memories sorting module and the module for choosing the next best question.
- **Chapter 4:** The web application chapter explains the steps that have been followed to develop the website from the beginning with an initial prototype to the end with a functional web application with different views. In between, the use of a MySQL database is described.
- **Chapter 5:** This is the conclusions and future work chapter, which includes the conclusions drawn from the results obtained in this work. It also details the possible lines of work that this TFG leaves open for its continuation in the future.

Capítulo 2

Estado de la Cuestión

En este capítulo se introducen las diferentes temáticas relacionadas con el trabajo desarrollado. Se presenta el concepto de demencia y cómo ralentizar el deterioro de las personas que la padecen mediante terapias de reminiscencia. También se explican los diferentes trabajos previos que se han ido desarrollando en el marco del proyecto CANTOR y que están relacionados con este TFG. Entrando en detalles más técnicos, se explica la importancia del Procesamiento del Lenguaje Natural de cara al desarrollo de tecnologías que comprendan a los humanos y, también se da a conocer el contexto de los Chatbots y cómo implementarlos.

2.1. Demencia

La demencia (?) es una condición neurodegenerativa progresiva, caracterizada por un deterioro cognitivo que interfiere con la vida cotidiana afectando a la memoria, al pensamiento, al lenguaje, al juicio y al comportamiento. La demencia no es una enfermedad específica aunque la mayor parte de los casos de demencia son provocados por la enfermedad de Alzheimer. Muchas veces se confunde la demencia con una consecuencia más del envejecimiento, cuando no tiene por qué ser así.

Hay muchos síntomas asociados a la demencia pero, en este trabajo, nos vamos a centrar en la pérdida de memoria. Actualmente no existen tratamientos que curen la enfermedad de Alzheimer, lo que hace necesario otro tipo de intervenciones para retrasar su desarrollo. Uno de los abordajes es el terapéutico, en concreto, trabajar los recuerdos de una persona que sufre demencia ayuda a retrasar los efectos de la misma. Hablaremos por ello de la terapia ocupacional basada en reminiscencia.

2.2. Terapia ocupacional basada en reminiscencia

La terapia ocupacional (?) se centra en que el paciente sea capaz de participar en las actividades de la vida cotidiana. Es decir, se basa en ayudar al individuo a llevar una vida lo más normal posible adaptando las tareas cotidianas o el entorno para que pueda llevarlas a cabo.

La terapia ocupacional basada en reminiscencia se centra en mejorar la calidad de vida de la persona con demencia. Se trata de una técnica basada en la recuperación de recuerdos dentro de un periodo de tiempo en la vida de la persona con el objetivo de construir la historia de vida del sujeto. La historia de vida surge de la sucesión de acontecimientos que componen la totalidad de la vivencia del sujeto. Para la revisión del pasado de la persona

con demencia, es fundamental introducir en la terapia estímulos que ayuden al paciente a recordar como por ejemplo, fotografías, objetos, música, etc. Los resultados de estas terapias están estudiados y concluyen que ayudan a ralentizar el deterioro cognitivo del usuario y mejoran su estado de ánimo.

2.3. Trabajos previos

Este trabajo está enmarcado en el proyecto CANTOR (?) dedicado al desarrollo de herramientas digitales de apoyo para las terapias ocupacionales basadas en reminiscencia dirigidas a personas con demencia. El objetivo del proyecto es realizar un programa usando tecnologías de Inteligencia Artificial capaz de recopilar la historia de vida del paciente y capaz de presentar esa información conseguida para que pueda ser revisada posteriormente.

Dentro de este contexto de trabajo, se han llevado a cabo diferentes proyectos por parte de antiguos alumnos de la Facultad de Informática enfocados a desarrollar distintas partes del proyecto CANTOR. Algunos de los relacionados con este TFG se cuentan a continuación:

1. Generación de historias a partir de una base de conocimiento de ?: Aplicación para usar en entrevistas con personas con demencia que sugiere temas de los que hablar y con los que poder recordar más fácilmente. Permite ir apuntando las ideas más importantes sobre cada tema tratado. La información recopilada se puede visualizar en forma de grafos o diccionarios porque la herramienta se encarga de ir enlazando los recuerdos que se van sacando. Además permite añadir, asociado a un tema, estímulos que ayuden a recordar al usuario, como recursos fotográficos.
2. Recuérdame, aplicación de apoyo para el tratamiento de personas con problema de memoria mediante terapias basadas en reminiscencia de ?: Aplicación web dirigida a los terapeutas encargados de tratar a personas con demencia. Es una herramienta de ayuda a la hora de planificar las sesiones. Se usa también como almacén de recuerdos de los pacientes que luego también pueden consultar la historia de vida que se ha ido construyendo.
3. Generación de historias de vida usando técnicas de Deep Learning de ?: Desarrollo de un sistema capaz de redactar las historias de vida de las personas a partir de sus recuerdos estructurados.
4. Sistema de asistencia para cuidados de enfermos del Alzheimer de ?: Aplicación que guarda la información sobre los terapeutas y sus pacientes. Con ella se recogen los recuerdos del paciente. La información recopilada más relevante se utiliza para generar la historia de vida de la persona en forma de narración.
5. Extracción de información personal a partir de redes sociales para la creación de un libro de vida de ?: Interfaz web donde se podrán consultar datos recopilados sobre personas con demencia en forma de historia de vida. Alimentando a la aplicación se encuentra un programa que se encarga de extraer la información del paciente de sus redes sociales de forma automática.
6. Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning de ?: Se trata de una terapia en forma de chat con un bot de Telegram. El usuario se encarga de enviar fotos que puedan

representar recuerdos suyos a la aplicación y la aplicación extrae de esas fotos preguntas relacionadas que puedan ayudar a la persona con demencia a recordar con más detalle lo relacionado con esa foto.

7. Generación de resúmenes de vídeo-entrevistas utilizando redes neuronales de ?: Aplicación web que utiliza redes neuronales para transcribir entrevistas en vídeo de personas con demencia y generar el resumen en español.
8. Extracción de recuerdos de vídeos de entrevistas con personas con problemas de memoria de ?: Herramienta capaz de crear transcripciones automáticas de entrevistas a personas con demencia en vídeo. Además, extrae la información importante de forma estructurada y la presenta en forma de grafo.

2.4. Procesamiento del lenguaje natural

El Procesamiento del Lenguaje Natural (?) es un campo de la Inteligencia Artificial que estudia las interacciones entre personas y máquinas mediante el uso del lenguaje natural, es decir, investiga cómo pueden comunicarse las computadoras y los humanos de forma eficiente. El PLN es un campo que se ha estado desarrollando durante los últimos 50 años y que, aunque tuvo poco éxito en un principio, en la actualidad, se emplea en muchos ámbitos. Es por intereses económicos y prácticos que los desarrollos en este área de conocimiento se hayan realizado para las lenguas más habladas como el inglés, alemán, español y chino. No obstante, existen muchas herramientas muy potentes que trabajan muchos idiomas como por ejemplo, el traductor de Google. El desarrollo de técnicas de procesamiento de lenguaje natural es vital también para el funcionamiento de los chatbots (tema en el que está centrado este TFG) como podrían ser los tan reconocibles Siri de Apple y Alexa de Amazon. Es importante tener en cuenta que se avanza mucho más en el lenguaje por escrito ya que hay muchos más datos y es más fácil de guardar en formato electrónico.

2.4.1. Modelos para el PLN

Para que las máquinas puedan tratar el lenguaje natural es necesario describirlo en términos matemáticos porque los ordenadores solo entienden de bytes. Existen dos formas de modelar el lenguaje:

1. Modelo gramatical: Esta formado por reglas de reconocimiento de patrones estructurales relacionados con la fonética, la morfología, la semántica, la sintaxis etc. Estas reglas las definen expertos lingüistas y son las que permiten a las máquinas reconocer lo que solicita la persona.
2. Modelo probabilístico: Se aplican técnicas matemáticas para extraer el conocimiento. En vez de usar reglas gramaticales, se recoge una gran cantidad de ejemplos y datos para calcular la frecuencia de aparición de las diferentes unidades lingüísticas (letras, palabras, oraciones) y su probabilidad de aparecer en un contexto determinado. Con estas probabilidades se puede predecir mucha información. Este modelo es lo que se denomina aprendizaje automático.

2.4.2. Componentes del PLN

Existen varios tipos de análisis para extraer información del lenguaje natural y sus usos dependerán del objetivo de la aplicación:

- Análisis morfológico: Se analiza la estructura interna de las palabras para clasificarlas en categorías (sustantivos, verbos, adjetivos, etc.) y extraer los lemas
- Análisis sintáctico: Consiste en estudiar la estructura sintáctica de las oraciones a partir de una gramática de la lengua en cuestión.
- Análisis semántico: Se utiliza para extraer el significado de las oraciones.
- Análisis pragmático: Añade al análisis el contexto de uso del lenguaje para mejorar la interpretación.

2.4.3. Aplicaciones del procesamiento del lenguaje natural

Algunas de las aplicaciones (?) de mayor utilidad del PLN son:

1. **Traducción automática de textos:** A día de hoy el nivel de traducción es bastante razonable. Existe la posibilidad de traducir textos largos a distintos idiomas. Es más, existen algunos navegadores que traducen automáticamente páginas web de un idioma a otro como por ejemplo “Google Chrome”. Las traducciones suelen ser más fiables para ciertas parejas de idiomas. El inglés suele ser uno de los idiomas más desarrollados en este campo por ser de los más usados en el mundo. Por otro lado, aun no está muy conseguida la traducción de documentos complejos o de aquellos con muchos matices, ni siquiera de un idioma principal a otro. Para dichos textos seguirá siendo necesaria la intervención de un humano.
2. **Sistemas conversacionales con PLN:** Son tecnologías de tipo Chatbot como Siri de Apple o el Asistente de Google. Son aplicaciones que entablan pequeñas conversaciones con el usuario y resuelven sus dudas, aunque a veces no con mucho éxito todavía. Se amplia la información en la sección 2.5.
3. **Respuestas automáticas a preguntas:** El primero que se hizo famoso fue el sistema Watson, desarrollado por IBM. Ganó algunos concursos televisivos contra humanos por su capacidad de responder todo tipo de preguntas gracias al amplio conocimiento que almacena en su base de datos.
4. **Análisis de sentimiento (?)**: Se trata de analizar opiniones sobre personas, productos y temas varios. Su uso más destacable es en el ámbito de las redes sociales para analizar cómo interactúan los usuarios. Se amplia la información en la sección 2.4.5.
5. **Resúmenes de textos automáticos**: A día de hoy existe muchísima información en Internet y muchos textos quedan sin leer por falta de tiempo. Los resúmenes automáticos ayudan a determinar si un texto merece ser leído al completo.
6. **Clasificación de documentos por categorías**: Ayuda a dirigir la información contenida en dichos documentos a los usuarios interesados, ahorrando tiempo.

2.4.4. Ventajas del PLN

- El ahorro de tiempo en trabajos que antes se realizaban a mano.
- Ahorro de costes con procesos automáticas en vez de acudir a profesional.
- Agiliza el etiquetado manual de documentos.

- Ayuda a tomar decisiones de negocio. Por ejemplo: el análisis automático de redes sociales permite detectar una posible crisis de reputación con rapidez y atajarla con mayor brevedad.
- Facilita el turismo con las traducciones entre idiomas y mejora la comunicación entre personas de distintas culturas.

2.4.5. Análisis de sentimiento

El análisis de sentimiento (?) es un método para identificar las emociones que se esconden tras un mensaje concreto y forma parte del procesamiento de lenguaje natural (PLN). Consiste en analizar las frases para extraer de ellas las opiniones o sentimientos acerca de un tema o producto.

Con este análisis (?) se pretende determinar quién es el sujeto del sentimiento, sobre qué o quién tiene ese sentimiento y categorizar esos sentimientos como positivos, negativos o neutros. Tras realizar el análisis del sentimiento se puede averiguar qué se esconde detrás de información subjetiva.

Una de las muchas aplicaciones de esta tecnología está enfocada al marketing, de forma que permite a las empresas averiguar qué es lo que quieren sus consumidores mediante el escrutinio de opiniones en redes sociales u otros medios. Estos sistemas tienen limitaciones, ya que no pueden detectar toda la complejidad del lenguaje humano. Se encuentran problemas a la hora de comprender el contexto en el que se encuentra un texto o para entender la ironía o el sarcasmo.

Algunas herramientas conocidas para el análisis del sentimiento son:

- Lingmotif¹ → Se trata de una herramienta de análisis de sentimiento desarrollada por la universidad de Málaga. Permite obtener valores precisos de las opiniones y sentimientos dentro de un texto.
- Opinion Finder² → Sistema desarrollado por investigadores de la Universidad de Pittsburgh, Cornell y Utah. Permite identificar la subjetividad de frases y varios aspectos de la subjetividad dentro de las propias frases mediante el procesamiento de documentos. Funciona en Inglés.
- LIWC³ → “Linguistic Inquiry and Word Count” es un programa capaz de analizar textos para calcular el uso que hacen las personas de distintas categorías de palabras. Permite saber si los emisores transmiten un mensaje con palabras positivas o negativas entre otras muchas opciones.

2.5. Chatbots

Un Chatbot o asistente virtual inteligente (?) es un programa informático capaz de mantener una conversación real con un usuario en lenguaje natural. Dan respuesta a dudas y tareas planteadas por los internautas. Para desarrollar un Chatbot se suele usar Procesamiento del Lenguaje Natural (PLN) y “Machine Learning”. Algunos ejemplos de asistentes virtuales serían los que recomiendan viajes y lugares turísticos, para la compra

¹<https://ltl.uma.es>

²<https://mpqa.cs.pitt.edu/opinionfinder/>

³<https://www.liwc.app>

de billetes de avión, para aprender idiomas y mejorar las habilidades lingüísticas, gestiones en banca, para consultar dudas sobre contratos de telefonía móvil, etc.

Muchas empresas disponen de un Chatbot para atender a sus clientes y resolver las dudas más frecuentes. Así, de paso, ahorran costes y dejan las preguntas más difíciles para los "call centres" o los chats con agentes humanos. Se consigue contratar a menos agentes, ya que muchas de las preguntas que realizan los usuarios son repetitivas y pueden ser respondidas por los bots. Son capaces de interpretar lo que el usuario pide a través del texto que introduce o lo que dice y de mantener una conversación y dar respuestas concretas.

2.5.1. Tipos

Existen distintos tipos de Chatbots según la finalidad que tengan:

- Asistentes que son capaces de mandar notificaciones simples como mensajes de texto o de Whatsapp.
- Los que se han diseñado para escuchar al usuario en conversaciones cortas como Siri de Apple que interpreta lo que pide el usuario y resuelve dudas o tareas como el tiempo va a hacer, quién ha ganado un partido de fútbol o enviar un mensaje. Conocen las respuestas a las preguntas frecuentes.
- Aquellos que resuelven temas concretos relativamente complejos dentro de una misma temática como, por ejemplo, la compra de billetes de tren, la compra de entradas, resolver dudas sobre contratos y ofertas de telefonía móvil, etc.
- Existen algunos más complejos que mantienen conversaciones con los usuarios como si fueran personas con sentimientos, empatía, conocimiento, etc.

Todos estos asistentes virtuales necesitan entender el lenguaje natural para recibir las peticiones por un lado y por el otro ser capaces de generar lenguaje natural para contestar.

2.5.2. Herramientas de desarrollo

Un bot conversacional se desarrolla dando forma a un programa que sea capaz de extraer los datos de las conversaciones, buscar en bases de datos la información necesaria y dar respuestas a los usuarios, entre otras cosas. Para distinguir las peticiones del usuario dentro del texto que introduce o dice, hay que entrenar al agente para que pueda aprender a reconocerlas. Para ello, se proporciona a la herramienta muchos ejemplos de frases que los clientes puedan llegar a decir. Cuando el usuario real introduzca una frase con una petición similar, el agente será capaz de deducir lo que pide. Para algunas de las peticiones, el Chatbot dará una respuesta inmediata y directa porque la solución es clara. En otros casos el agente requerirá más información del usuario y le planteará preguntas para recabar dicha información. Por ejemplo, en caso de querer hacer un pedido, el bot necesitará más información sobre lo que el cliente desea comprar y formulará la pregunta correspondiente para averiguarlo.

Normalmente, para asegurar que lo que el chatbot entiende se corresponde con la solicitud del usuario, a cada petición detectada se le asigna un nivel de confianza según la probabilidad de que se haya detectado correctamente o no. Si la probabilidad calculada es baja se le hará otra pregunta al usuario para mejorar dicha confianza como por ejemplo, "no te he entendido, ¿puedes repetir la pregunta?". Se trata de intentar que el usuario

formule la petición de otro modo que sea más comprensible para el Chatbot. El objetivo es afinar lo máximo posible las respuestas que proporciona evitando contestar algo incorrecto. Se debe entrenar mucho al asistente para llegar a un nivel de confianza óptimo (80 %, por ejemplo) y que dicho asistente sea eficaz y útil al cliente.

Algunas de las herramientas (?) que ayudan a desarrollar este tipo de programas inteligentes son:

- Language Understanding (LUIS) de Microsoft: Servicio de procesamiento de lenguaje natural con acceso a una biblioteca de conocimiento que pueden utilizar los bots.
- Google Dialogflow: Es un servicio de Google Cloud con acceso a las técnicas de machine learning de Google. Funciona mediante la comprensión del lenguaje natural.
- Watson Assistant de IBM: Proporciona un servicio en la nube y funcionalidades como respuestas automáticas, procesamiento del lenguaje natural, reconocimiento de peticiones, etc.
- ChattyPeople: De las mejores plataformas. Permite crear bots en Facebook de forma personalizada.
- Botsify: Dotado de una inteligencia artificial muy refinada. Permite conectarse con muchas plataformas a través de plugins.
- Telegram Bots: Pensado para crear chatbots desde la plataforma de mensajería de Telegram. También permite la integración con bots externos.
- Chatfuel: Pensada para cualquier usuario sin necesidad de conocimientos técnicos. Permite automatizar las respuestas a preguntas frecuentes y establecer reglas de mensajes automáticos.
- AIML: Es un lenguaje de marcado, es decir, formado por etiquetas o marcas como HTML. Se basa en la búsqueda de patrones para la creación de bots conversacionales llamados “Alicebots”⁴.
- Python⁵: Creación de chatbots mediante librerías como spaCy⁶, Chatterbot o NLTK.

⁴<https://github.com/ezabou/ElisaBot>

⁵<https://analyticsindiamag.com/top-python-libraries-for-chatbot-development/>

⁶<https://course.spacy.io/es>

Capítulo 3

Arquitectura del Chatbot

Los sistemas de análisis automático de texto tienen como objetivo entender la información no estructurada hablada por los humanos y convertirla en información estructurada como podría ser un resumen del contenido o la clasificación de un documento por temática.

En el caso que nos ocupa, el objetivo del chatbot es conseguir esta información estructurada a partir de la mayor cantidad de recuerdos posibles para ayudar a los terapeutas a construir la historia de vida de un paciente que es mucho más complicada de tratar en bruto. Antes de estructurarla, primero hay que conseguir la información a través de las conversaciones con la persona. De esto se encargará uno de los módulos del bot (detallado en la sección 3.1) que planteará todo tipo de preguntas para recopilar el máximo número de recuerdos. Estas preguntas se extraen de una base de datos de tipo MongoDB y se eligen de forma ordenada.

Las respuestas que da el usuario también se almacenan en Mongo, pero previamente se procesan. El análisis de estas respuestas se va a realizar en varios sentidos, por un lado, se clasificarán los recuerdos en etapas de vida, para estructurar la información lo máximo posible. Por otro lado, se determinará si el texto contiene connotaciones positivas o negativas, de tal forma que en la terapia se puedan evitar los recuerdos negativos que pueden agravar la situación del paciente. Estos dos procesamientos mencionados pertenecen al módulo de clasificación de recuerdos explicado con más detalle en la sección 3.2. Por último, se utilizará el procesamiento del lenguaje natural mediante el análisis morfológico de las palabras para comprender lo que quiere decir el paciente y poder generar la siguiente pregunta más acertada y relacionada con la temática que se está tratando. Este último apartado compondrá el módulo de procesamiento del texto para encadenar preguntas y respuestas descrito en la sección 3.3.

En resumen, para el funcionamiento íntegro del chatbot, deben estar presentes tres módulos: el módulo de las preguntas, el módulo de la clasificación de recuerdos y el módulo de elección de la mejor siguiente pregunta. Se explican a continuación, uno por sección.

3.1. Repertorio de preguntas

El Chatbot que se ha desarrollado no funciona como la mayoría de agentes que se han creado a lo largo de los últimos años. Siri por ejemplo es uno de ellos y funciona de tal manera que es el usuario quien conduce la conversación haciendo preguntas o peticiones. Este programa se diferencia de los demás porque, por el contrario, es el bot quien dirige la conversación planteándole preguntas al usuario.

El sistema funciona realizando una primera pregunta y, a continuación, por cada respuesta que dé el usuario se formula otra pregunta. Esta pregunta estará lo más relacionada posible con la contestación previamente recibida, módulo que se explicará en la sección 3.3.

El repertorio de preguntas se extrae de la base de datos de MongoDB. Esta colección de cuestiones se ha ido formando a lo largo del TFG mediante la reformulación de preguntas sacadas de documentos de terapia ocupacional ofrecidos por expertos terapeutas del proyecto CANTOR. Es decir, no se han cogido directamente de estos documentos con entrevistas, sino que se han ido redactando y adaptando según las necesidades que iban surgiendo para el Chatbot.

3.2. Clasificación de los recuerdos

Tal y como se explica en la introducción de este capítulo, este módulo de clasificación tiene dos componentes, el que analiza el sentimiento del texto y distingue entre recuerdos positivos y negativos y, el que categoriza por etapas de vida. Ambos están fundamentados en base a la misma idea y utilizan el mismo método de clasificación automática basado en etiquetas.

Para poder llevar a cabo la distinción de categorías dentro de un texto se ha utilizado la librería *Spacy* de Python que cuenta con un componente llamado “TextCategorizer”¹. Este componente permite entrenar un modelo capaz de predecir las categorías o etiquetas que se quieran identificar de un texto como en este caso, frases positivas o frases negativas. Se probaron otras tecnologías y librerías de Python para conseguir un resultado parecido. Dos de las librerías consideradas fueron “NLTK” y “Chatterbot”. Pero, finalmente se eligió *Spacy* por su versatilidad, su amplia oferta de funcionalidades y por la facilidad con la que se pueden tokenizar textos. De cara a la evolución del trabajo, prevista en un principio, se trataba de la mejor herramienta para poder ampliar el chatbot gracias a sus componentes destinados al procesamiento del lenguaje natural.

Una vez elegida, se probó primero la herramienta “TextCategorizer” para la clasificación de los recuerdos en positivos y negativos. Por ello, se empezará a explicar haciendo referencia únicamente al análisis de sentimientos. Más adelante en esta sección (apartado 3.2.2) se explicará cómo se adaptó para el módulo de clasificación en etapas.

Spacy proporciona algoritmos de aprendizaje automático que observan unos datos facilitados y construye a partir de ellos un modelo capaz de clasificar gracias a la estadística. Estos algoritmos son los que interesan a la hora de construir el categorizador de textos. Como guía de uso de estas herramientas, se ha utilizado el código del artículo *How to Train Text Classification Model in spaCy*².

Los datos iniciales proporcionados para entrenar al modelo suelen proceder de datasets muy grandes para que el modelo sea lo más preciso posible. En este caso, el objetivo es diferenciar los recuerdos negativos de los positivos y así tener la primera clasificación que pidieron los terapeutas especializados del proyecto CANTOR. Es por ello que lo interesante es utilizar un conjunto grande de recuerdos tanto positivos como negativos. Antes de probar con recuerdos se probó el funcionamiento del “TextCategorizer” con un dataset muy grande de reseñas de moda que se ponía como ejemplo en el artículo del que se hablaba anteriormente. Con ello se comprobó la precisión de clasificación y se pasó a adaptarlo a la situación de este TFG. Al no encontrar ningún dataset de recuerdos por ningún lado, se ha usado una batería relativamente pequeña (comparado con el tamaño habitual de en-

¹<https://spacy.io/api/textcategorizer>

²<https://www.machinelearningplus.com/nlp/custom-text-classification-spacy/>

trenamiento) de recuerdos positivos encontrados entre los archivos del TFG de ?. También había que incluir recuerdos negativos y para ello se han usado frases negativas inventadas, pensadas para entrenar el modelo de la mejor manera posible. Concretamente, se consiguió reunir una cantidad de 32 recuerdos negativos frente a 140 positivos. A continuación se muestran ejemplos de algunos de los textos:

Positivos:

- “El año pasado mis abuelos celebraron 60 años de casados”
- “Cuando era niña, disfrutaba de las bandas sonoras de las películas de Disney”

Negativos:

- “A los 14 años mi padre se murió de cáncer”
- “No me llevaba bien con mi hermano Jorge y todavía hoy no nos llevamos bien.”

En cuanto al desarrollo del código, el proceso de categorización de recuerdos funciona siguiendo una serie de pasos. En la figura 3.1 se encuentra el diagrama de flujo del clasificador con cada uno de los pasos que sigue para entrenar un modelo y poder, así, predecir las categorías.

En primer lugar, se define el modelo. Puede crearse vacío o se puede cargar directamente un modelo pre-entrenado con una serie de componentes. Un componente es una función que se aplica a un texto y que lo devuelve procesado de alguna forma. Por ejemplo, el componente “NER”, más reconocido como *Named Entity Recognizer*, reconoce entidades dentro del texto como nombres de personas, fechas o lugares. Para nuestro caso en concreto, se crea un modelo en blanco y se le añade el componente “TextCategorizer” (textcat). Un modelo en blanco es un modelo que no tiene ningún componente de spaCy definido, es decir, el texto que se quiera procesar no será analizado por ninguna cadena de funciones. El texto procesado que devuelva el modelo se mantiene intacto. Si se hubiese cogido un modelo pre-entrenado como es “es_core_news_sm”, se habría añadido el componente de clasificación de textos por encima del resto de componentes. Es decir, se sumaría al trabajo de los procesos que analizan el texto. En el caso de “es_core_news_sm” está compuesto una serie de funciones (componentes) que se ejecutan en orden. Algunas de ellas son:

- “tok2vec”: transforma los tokens en vectores (array de valores numéricos) para que la máquina pueda entender una secuencia de caracteres.
- “morphologizer”: analiza morfológicamente los tokens
- “lemmatizer”: halla el lema de las palabras
- “ner”: reconoce entidades, explicado anteriormente

En este caso, no se necesitaban las funciones que ofrecían los modelos pre-entrenados y por eso se empieza con uno vacío. Al usar el modelo en blanco se empieza con el español de cero, sin nada que analizar. Tras añadir el componente textcat, cualquier texto a tratar por el modelo, automáticamente, pasará por el proceso de clasificación que le indiquemos. Para que textcat funcione como categorizador de recuerdos, se le añade a nuestro nuevo componente dos etiquetas, NEGATIVO y POSITIVO, que definirán cuánto de negativo es un recuerdo y cuánto de positivo. Para seguir configurando esta función, es necesario cargar los datos iniciales, aquellos recuerdos seleccionados anteriormente que serán los que

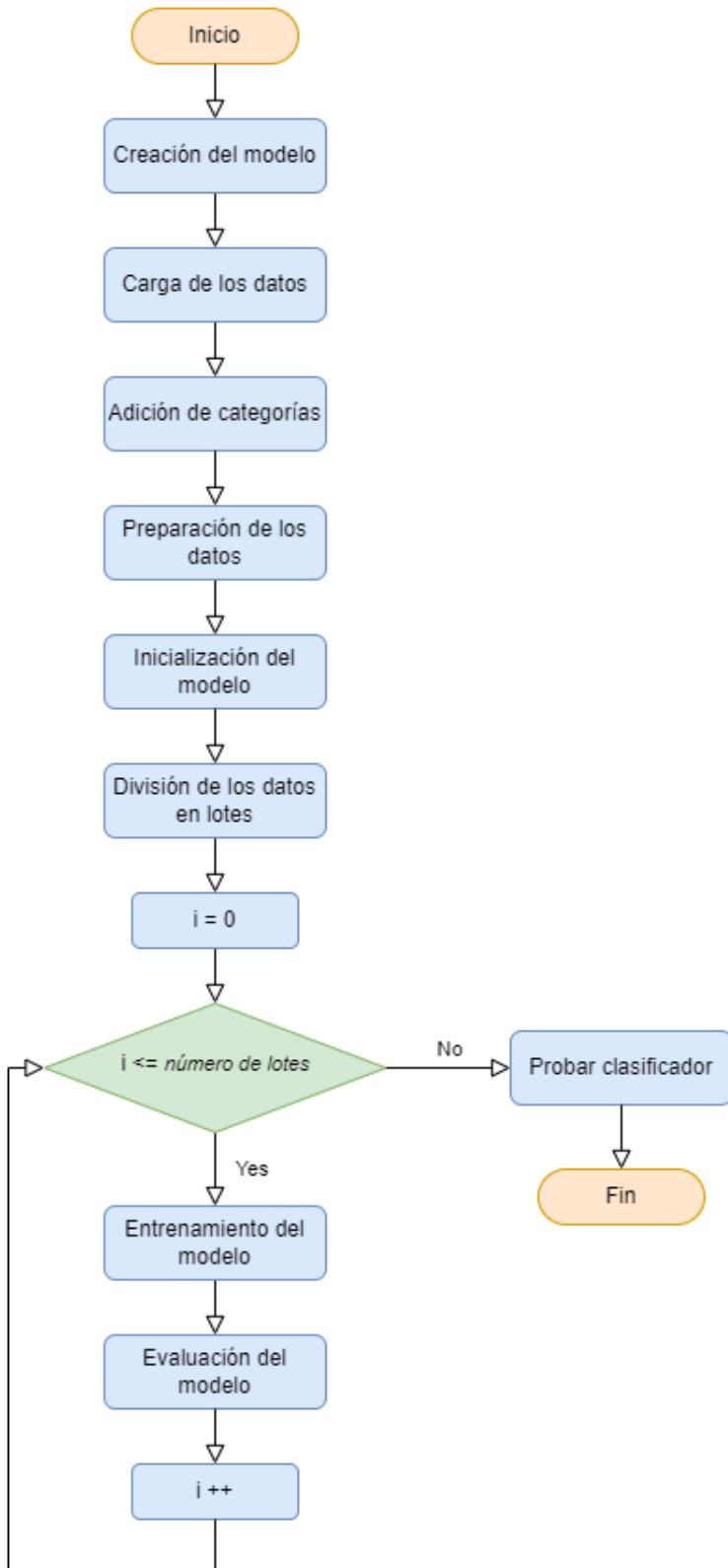


Figura 3.1: Diagrama de flujo del categorizador

entrenen al modelo. Esta información se prepara antes de ser procesada para que el clasificador la entienda. Es decir, los datos se adecuan al formato que entiende el modelo siendo éste una lista de tuplas (texto, categoría). Por ejemplo:

```
(El año pasado mis abuelos celebraron 60 años de casados,
{'cats': {'POSITIVO': True, 'NEGATIVO': False}})
```

El modelo reservará un porcentaje de esta batería de recuerdos (% que previamente se define) para la evaluación de las predicciones, es decir, para analizar cómo de bien predice el modelo tras entrenarlo.

Ya está todo preparado para comenzar a entrenar el modelo, se dividen los casos de entrenamiento en lotes que se analizan y evalúan un número definido de iteraciones para asegurar que el entrenamiento es lo más preciso posible. Es importante no superar un número de repeticiones para que sea también óptimo. El modelo se entrena utilizando una función de *Spacy* que lo analiza y actualiza en cada repetición con lo nuevo que ha aprendido, la función update³. Funciona de la siguiente forma:

1. Se inicializan los parámetros del modelo de forma aleatoria.
2. Se predice el primer lote de ejemplos con los parámetros que se han fijado.
3. Se revisan estas predicciones comparándolas con las etiquetas correctas.
4. Se decide cómo cambiar los parámetros actuales para obtener mejores predicciones en las próximas veces.
5. Se hace la pequeña corrección de los parámetros.
6. Vuelta al paso 2 pero con el siguiente lote de ejemplos.

Se repite este bucle de entrenamiento hasta que el modelo deje de mejorar. En la figura 3.2 sacada del manual de *Spacy*⁴ se puede visualizar cómo funciona.



- **Datos de entrenamiento:** Ejemplos y sus anotaciones.
- **Texto:** El texto para el cual el modelo debe predecir una etiqueta.
- **Label:** La etiqueta que el modelo debe predecir.
- **Gradiente:** Cómo cambiar los parámetros.

Figura 3.2: Entrenamiento del modelo

Además, en cada vuelta, también se realizan una serie de pruebas para comprobar la precisión de las predicciones, se comparan con las etiquetas de referencia para estimar la

³<https://spacy.io/api/textcategorizer#update>

⁴<https://course.spacy.io/es/chapter4>

desviación de la pérdida, es decir, cuánto de alejado se encuentra del valor real. Estas comprobaciones se realizan para asegurarse que el algoritmo está haciendo bien las cosas.

Una vez afinado el modelo mediante el entrenamiento previo, ya se puede poner a prueba. El clasificador sacará la probabilidad entre 0 y 1 de que un texto procesado por spacy sea un recuerdo positivo y la probabilidad de que sea uno negativo de esta forma:

```
Texto analizado: {''POSITIVO'': probabilidad entre 0 y 1,  
'NEGATIVO': probabilidad entre 0 y 1}
```

Como se trata de dos etiquetas mutuamente excluyentes la probabilidad de una será la diferencia de la otra. Así, la que mayor probabilidad tenga será la elegida como predicción final. Si la probabilidad de que sea un texto positivo es mayor que la de que sea negativo, el recuerdo será finalmente evaluado como positivo y lo mismo al revés.

En resumen, como se ha explicado, el programa entrena primero al modelo con textos positivos y negativos y luego evalúa el texto que le llega del usuario e identifica si es negativo o positivo. Esto se utilizará como una de las categorías que se le asignan a cada respuesta recibida y que se mete en una base de datos Mongo junto a la respuesta que haya dado el usuario al chatbot. Este análisis de sentimiento también servirá para identificar temáticas dolorosas para el interrogado que no deberían ser usadas en las terapias.

Más adelante en el proyecto, se observó que, al analizar el sentimiento de la frase, había recuerdos o simplemente, datos que no encajaban en ninguna de las dos categorías de positivo y negativo. Las emociones neutras son aquellas que no son desagradables ni agradables, es decir, ni negativas ni positivas. Es por esto que se probó a añadir una nueva categoría para los datos neutros como por ejemplo, indicar tu edad, explicar dónde vivías en cierto momento de tu vida, etc. No suponía un gran esfuerzo porque el programa estaba pensado para poder añadir todas las categorías que se considerasen. Sin embargo, a niveles prácticos, se pudo comprobar que no funcionaba igual de bien que la clasificación binaria. El nivel de entrenamiento que necesitaría el modelo para poder distinguir también las frases neutras es mucho mayor, es decir, se necesitarían muchos más casos, ejemplos, para que el modelo aprendiese de ellos. En este caso, solo se añadieron alrededor de 40 recuerdos neutros, de nuevo inventados, por la falta de recursos encontrados en Internet que no proporcionaba ningún dataset parecido al que se buscaba. En la siguiente sección de pruebas se podrá comprobar la diferencia entre meter la categoría neutra y no meterla.

3.2.1. Pruebas del análisis de sentimiento

Para comprobar la precisión y el correcto funcionamiento del módulo de análisis de sentimiento, se hicieron bastantes pruebas que corroboraron que la clasificación en recuerdos negativos y positivos funcionaba bastante bien pero que podría mejorarse metiendo muchos más casos de entrenamiento para afinar el modelo. A continuación se muestran algunos ejemplos de frases que se han analizado con el categorizador de textos y las conclusiones que se han sacado. Entre llaves se muestra la probabilidad sobre 1 de que la frase sea positiva y la probabilidad sobre 1 de que sea negativa. Entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad las dos:

- “Mi abuela se murió en 1998”


```
{'POSITIVO': 0.09926079958677292, 'NEGATIVO': 0.9007392525672913}  
['negativo']
```
- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'POSITIVO': 0.992607593536377, 'NEGATIVO': 0.007392475381493568}  
['positivo']
```

- “Tengo 23 años”

```
{'POSITIVO': 0.9999983310699463, 'NEGATIVO': 1.6568293403906864e-06}  
['positivo']
```

- “Me dolió muchísimo cuando me rompí una pierna”

```
{'POSITIVO': 2.0322097043390386e-05, 'NEGATIVO': 0.9999797344207764}  
['negativo']
```

- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'POSITIVO': 0.9681606888771057, 'NEGATIVO': 0.031839337199926376}  
['positivo']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'POSITIVO': 0.9882893562316895, 'NEGATIVO': 0.011710633523762226}  
['positivo']
```

- “Mi pareja sufrió depresión después del parto”

```
{'POSITIVO': 0.07562904059886932, 'NEGATIVO': 0.9243709444999695}  
['negativo']
```

Por otro lado, también se probó cómo funcionaba el programa añadiendo la clasificación de neutro. Podemos comparar así ambas formas de clasificación y comprobar que, efectivamente, funciona mejor el binario. De nuevo, entre llaves se muestra la probabilidad sobre 1 de que la frase sea positiva, la probabilidad sobre 1 de que sea negativa y la probabilidad sobre 1 de que sea neutra. Entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi abuela se murió en 1998”

```
{'POSITIVO': 0.000133998051751405, 'NEGATIVO': 0.00039583168108947575,  
'NEUTRO': 0.9994701743125916}  
['neutro']
```

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'POSITIVO': 0.0003367967437952757, 'NEGATIVO': 0.9894788861274719,  
'NEUTRO': 0.010184396989643574}  
['negativo']
```

- “Tengo 23 años”

```
{'POSITIVO': 0.016445694491267204, 'NEGATIVO': 0.0031412760727107525,  
'NEUTRO': 0.980413019657135}  
['neutro']
```

- “Me dolió muchísimo cuando me rompí una pierna”

```
{'POSITIVO': 0.0006958534941077232, 'NEGATIVO': 0.9989858269691467,
'NEUTRO': 0.0003183614171575755}
['negativo']
```

- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'POSITIVO': 2.354631942580454e-05, 'NEGATIVO': 0.9972598552703857,
'NEUTRO': 0.0027166458312422037}
['negativo']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'POSITIVO': 0.051713500171899796, 'NEGATIVO': 0.9064642786979675,
'NEUTRO': 0.041822321712970734}
['negativo']
```

- “Mi pareja sufrió depresión después del parto”

```
{'POSITIVO': 2.2260337573243305e-05, 'NEGATIVO': 0.9954761862754822,
'NEUTRO': 0.004501543939113617}
['negativo']
```

En esta sección de pruebas se puede comprobar que algo no debe estar funcionando del todo bien y que el problema de las predicciones puede estar en las dimensiones del dataset que se le proporciona al modelo. Faltos de un dataset grande de recuerdos para el análisis de sentimientos se buscó uno que no fuese de recuerdos como tal pero que sí podría servir para hacer una buena predicción de negativos y positivos. Se encontró un dataset⁵ en español de 50,000 reseñas de IMBD de películas con un sentimiento positivo o negativo adjunto (no se encuentra ninguno que incluya el neutro). Probando el clasificador cargando esta colección de datos al modelo, se observan los siguientes resultados:

- “Mi abuela se murió en 1998”

```
{'POSITIVO': 0.8378736972808838, 'NEGATIVO': 0.16212628781795502}
['positivo']
```

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'POSITIVO': 0.9998306035995483, 'NEGATIVO': 0.00016938232874963433}
['positivo']
```

- “Tengo 23 años”

```
{'POSITIVO': 0.9846466779708862, 'NEGATIVO': 0.015353254973888397}
['positivo']
```

- “Me dolió muchísimo cuando me rompí una pierna”

⁵<https://www.kaggle.com/datasets/luisdiegofv97/imdb-dataset-of-50k-movie-reviews-spanish>

```
{'POSITIVO': 0.08775553107261658, 'NEGATIVO': 0.9122444987297058}  
['negativo']
```

- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'POSITIVO': 0.999742329120636, 'NEGATIVO': 0.0002577087143436074}  
['positivo']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'POSITIVO': 0.9989288449287415, 'NEGATIVO': 0.0010711398208513856}  
['positivo']
```

- “Mi pareja sufrió depresión después del parto”

```
{'POSITIVO': 0.03745762258768082, 'NEGATIVO': 0.9625424146652222}  
['negativo']
```

Podemos observar que acierta todas las frases menos la primera. Se ha probado con más frases relacionadas con la muerte y todas las predice como positivas. En este caso, el entrenamiento parece no haber perfeccionado los resultados, puede ser por la información que proporciona el dataset, que se aleje mucho de lo que pueda surgir de un recuerdo o, por algún fallo en el proceso de entrenamiento.

En conclusión, lo ideal habría sido tener una colección de datos lo suficientemente grande para entrenar bien al modelo y que estuviese relacionado con los recuerdos y memorias de la vida de personas. Se plantea como trabajo futuro. Por la precisión de los resultados obtenidos finalmente, se optó por el primer entrenamiento, que tiene un dataset pequeño de recuerdos pero funciona relativamente bien, como se ha podido observar en estas pruebas. La aplicación, por ende, funciona con el primer clasificador planteado.

3.2.2. Clasificación en etapas de vida

Otra de las categorías que se añadirá junto a la respuesta en la base de datos de recuerdos será la de la etapa de vida a la que corresponde el recuerdo. En un principio, había que definir las etapas y se decidió usar de forma orientativa la clasificación que hace el Ministerio de Salud de Colombia⁶ porque no se encontraron fuentes fiables españolas que diesen una clasificación tan clara y sensata. Aún así se le hicieron algunas adecuaciones para que se ajustase más a lo que se buscaba y lo que se quería sacar en claro. Además, como en el caso del análisis de sentimiento con el neutro, se decidió desde un principio añadir la clasificación “Indeterminado” para aquellos recuerdos que no encajasen en ninguna etapa porque fuesen datos generales. Aquí se veía más claro que había información de los usuarios que era muy general y que por defecto no podía meterse en ninguna otra etapa. Sin embargo, pasaba lo mismo que con el neutro, la clasificación era mucho menos acertada. Más adelante se explicará con más detalle. A continuación se muestra la clasificación inicial que se eligió:

- Infancia de 0 a 11 años
- Adolescencia de 12 a 17 años
- Juventud de 18 a 26 años

⁶<https://www.minsalud.gov.co/proteccionsocial/Paginas/cicloVida.aspx>

- Etapa adulta de 27 a 59 años
- Vejez de 60 años o más
- Indeterminado para aquellos textos en los que no se pueda distinguir la etapa

Una vez elegidos los períodos de tiempo en los que se divide la vida de las personas, el objetivo era clasificar recuerdos, dados en forma de respuesta a una pregunta, según la etapa de la vida a la que pertenecía el recuerdo de la persona interrogada. Para ello, también se ha utilizado la misma tecnología que para el análisis de sentimientos. Además de entrenarse en recuerdos positivos y negativos, ahora el modelo se entrena para distinguir etapas vitales en las que ocurren los recuerdos para así, de cara a la elaboración de un libro de vida, toda la información esté bien estructurada en períodos de tiempo.

Como se explicaba al principio, la clasificación que se eligió inicialmente no resultó dar buenos resultados. El nivel de entrenamiento que necesitaría el modelo para poder distinguir también los recuerdos que no encajan en ninguna de las etapas anteriores es mucho mayor. Además, resultaba muy ambigua la etapa indeterminada porque los recuerdos que se meten en esa clasificación no tienen relación entre ellos para el modelo, no puede aprender de similitudes porque cada recuerdo es muy distinto y muy general. Algo parecido pasaba con la etapa de la adolescencia, no se aprecia bien en los ejemplos la diferencia entre la etapa de la adolescencia y la de la juventud porque no existen ejemplos que hagan una buena distinción entre la una y la otra ya que son muy parecidos los sucesos que podrían ocurrir en una y en la otra. Es por eso que se ha decidido quitar ambas etapas y ponérselo mucho más fácil al modelo para que pueda reconocer bien los recuerdos y no confundirse tan fácilmente. En la sección de pruebas se ve claramente la diferencia entre ambos casos. Finalmente, se ha elegido una clasificación muy clara según los hechos que suelen pasar en la vida de las personas en las distintas etapas y para los que cada etapa tiene ejemplos precisos y nada ambiguos. Se hace la siguiente división:

- Infancia de 0 a 12 años
- Juventud de 13 a 26 años
- Etapa adulta de 27 a 59 años
- Vejez de 60 años o más

3.2.2.1. Pruebas

Se comprueba la precisión y el correcto funcionamiento del módulo de clasificación en etapas de vida, se hicieron bastantes pruebas que indicaron que la clasificación funcionaba bastante bien pero que podría mejorarse metiendo muchos más casos de entrenamiento para afinar el modelo. La clasificación definitiva como se explicaba anteriormente era la que distinguía las etapas de infancia, juventud, etapa adulta y vejez. A continuación se muestran algunos ejemplos de frases que se han analizado con el categorizador de textos y las conclusiones que se han sacado. Entre llaves se muestra la probabilidad sobre 1 de que la frase se corresponda con la fase de la infancia, la probabilidad sobre 1 de que se corresponda con la fase de la juventud, la probabilidad sobre 1 de que se corresponda con la fase de la etapa adulta y la probabilidad sobre 1 de que se corresponda con la fase de la vejez. Entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi mejor recuerdo es el del nacimiento de mi hijo”
{'INFANCIA': 0.003423456335440278, 'JUVENTUD': 0.004028527531772852,
'ETAPA ADULTA': 0.979805052280426, 'VEJEZ': 0.012742995284497738}
['etapa adulta']
- “Me dolió muchísimo cuando me rompí una pierna a los 22 años”
{'INFANCIA': 0.07795873284339905, 'JUVENTUD': 0.7799875736236572,
'ETAPA ADULTA': 0.07531660795211792, 'VEJEZ': 0.0667370930314064}
['juventud']
- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”
{'INFANCIA': 0.9991468191146851, 'JUVENTUD': 0.00011972746870014817,
'ETAPA ADULTA': 0.00013617362128570676, 'VEJEZ': 0.0005973773077130318}
['infancia']
- “Durante la infancia estuvimos viviendo en Moratalaz”
{'INFANCIA': 0.9864945411682129, 'JUVENTUD': 0.0020768800750374794,
'ETAPA ADULTA': 0.005027524195611477, 'VEJEZ': 0.0064010825008153915}
['infancia']
- “Mi pareja sufrió depresión después del parto”
{'INFANCIA': 0.10760761052370071, 'JUVENTUD': 0.003418843261897564,
'ETAPA ADULTA': 0.8073434829711914, 'VEJEZ': 0.08163014054298401}
['etapa adulta']
- “Mi tía siempre se dedicó a la pintura”
{'INFANCIA': 0.15463659167289734, 'JUVENTUD': 0.2145996242761612,
'ETAPA ADULTA': 0.5623230338096619, 'VEJEZ': 0.0684407576918602}
['etapa adulta']
- “Cuando estudiaba en el instituto teníamos tres gatos”
{'INFANCIA': 0.07519558817148209, 'JUVENTUD': 0.6477565169334412,
'ETAPA ADULTA': 0.23660382628440857, 'VEJEZ': 0.040444087237119675}
['juventud']

También se probó cómo funcionaba el modelo en un principio cuando en la clasificación estaban las etapas de adolescencia y la indeterminada. Se puede comparar así ambas formas de clasificación y comprobar que, efectivamente, funciona mejor la que tiene menos categorías y mucho más diferenciadas. De nuevo, entre llaves se muestra la probabilidad sobre 1 de cada una de las fases y, entre corchetes se muestra la categoría a la que pertenece la frase por tener la mayor probabilidad de todas:

- “Mi mejor recuerdo es el del nacimiento de mi hijo”

```
{'INFANCIA': 0.11781484633684158, 'ADOLESCENCIA': 0.29379358887672424,
'JUVENTUD': 0.22842659056186676, 'ETAPA ADULTA': 0.20501598715782166,
'VEJEZ': 0.10608396679162979, 'INDETERMINADO': 0.04886503145098686}
['adolescencia']
```

- “Me dolió muchísimo cuando me rompí una pierna a los 22 años”

```
{'INFANCIA': 0.0013075786409899592, 'ADOLESCENCIA': 0.0039981659501791,
'JUVENTUD': 0.9936805963516235, 'ETAPA ADULTA': 0.0006605012458749115,
'VEJEZ': 0.00015935904229991138, 'INDETERMINADO': 0.00019375460396986455}
['juventud']
```

- “Mi hermano y yo nos pasábamos las tardes haciendo puzzles”

```
{'INFANCIA': 0.9913138747215271, 'ADOLESCENCIA': 0.0026805144734680653,
'JUVENTUD': 0.0005102856084704399, 'ETAPA ADULTA': 0.0003391568607185036,
'VEJEZ': 0.00324622611515224, 'INDETERMINADO': 0.0019098836928606033}
['infancia']
```

- “Durante la infancia estuvimos viviendo en Moratalaz”

```
{'INFANCIA': 0.9902841448783875, 'ADOLESCENCIA': 0.0011972723295912147,
'JUVENTUD': 0.007258791010826826, 'ETAPA ADULTA': 0.00017597108671907336,
'VEJEZ': 0.0006049419171176851, 'INDETERMINADO': 0.00047876848839223385}
['infancia']
```

- “Mi pareja sufrió depresión después del parto”

```
{'INFANCIA': 0.9323758482933044, 'ADOLESCENCIA': 0.011906568892300129,
'JUVENTUD': 0.018807733431458473, 'ETAPA ADULTA': 0.008572818711400032,
'VEJEZ': 0.025070885196328163, 'INDETERMINADO': 0.0032660840079188347}
['infancia']
```

- “Mi tía siempre se dedicó a la pintura”

```
{'INFANCIA': 7.487166294595227e-05, 'ADOLESCENCIA': 0.00675414502620697,
'JUVENTUD': 0.0021927112247794867, 'ETAPA ADULTA': 0.9876710176467896,
'VEJEZ': 0.0032006383407860994, 'INDETERMINADO': 0.00010662782733561471}
['etapa adulta']
```

- “Cuando estudiaba en el instituto teníamos tres gatos”

```
{'INFANCIA': 0.9774642586708069, 'ADOLESCENCIA': 0.005647959187626839,
'JUVENTUD': 0.012261644005775452, 'ETAPA ADULTA': 0.0004069636052008718,
'VEJEZ': 0.0033992205280810595, 'INDETERMINADO': 0.0008198729483410716}
['infancia']
```

3.3. Procesamiento del texto para encadenar preguntas y respuestas

Esta sección consiste en conseguir que las preguntas que se hagan al interrogado tengan sentido con el resto de la conversación previa. Conseguir que sean lo más inteligentes y adecuadas posible.

Lo primero que se ha hecho para encontrar la siguiente pregunta a hacer es coger la lista de todas las posibles preguntas y eliminar aquellas que ya han sido contestadas anteriormente para no repetirlas.

Lo siguiente que se hará es pasar tanto la respuesta más reciente que ha dado el interrogado como todas las posibles preguntas por un proceso de síntesis. Para ello vamos a utilizar el analizador de textos de *Spacy* utilizando algunas de las funciones procedentes del código del TFG de ?, en concreto del archivo “analysis.py” de su código. Se trata de funciones para el procesamiento y síntesis de los textos. Los pasos que Laura sigue para el análisis del texto aparecen en el apartado 5.1 (Analizar textos para obtener sugerencias) de su memoria, páginas 30 y 31. Las funciones que se han reutilizado en este trabajo, con algunas modificaciones, son las siguientes:

- Read: Coge el texto a analizar y elimina signos de puntuación
- Synthesis: Elimina las palabras vacías como las preposiciones del texto
- Lemmatize: Transforma cada palabra del texto sus lemas correspondientes
- Categorize: Separa las palabras en sustantivos, verbos y adjetivos. Además, coge los 30 más comunes de cada tipo
- Get_most_common_words: Coge las palabras más comunes de todos los tipos del texto

Estas funciones se han utilizado para conseguir una representación del texto más precisa y reducida. Únicamente se han utilizado este código para reducir los lemas de las palabras importantes del texto que es lo que se usado para este módulo de encadenamiento de preguntas y respuestas.

Por último, dada la respuesta y todas las posibles preguntas ya reducidas a sus lemas, para encontrar la pregunta más adecuada, se compara una a una los lemas de las posibles preguntas con los lemas de la respuesta. Dentro de la lista de posibles preguntas, la que más lemas comunes tenga con la respuesta, gana como siguiente pregunta a formular.

3.3.1. Ejemplo de funcionamiento de la selección de la siguiente pregunta a realizar

Se comprueba cómo funciona el módulo que se acaba de explicar. Se muestra un ejemplo de este primer acercamiento a un chatbot inteligente que pregunta con algo de sentido. A continuación se muestra la primera pregunta y la respuesta del interrogado:

IA: ¿Quiénes son las personas más importantes de tu vida?

Tú: Mi familia y concretamente mis padres y mi hermano

En el siguiente texto de salida por consola se muestra el análisis que hace el módulo de cálculo de la mejor siguiente pregunta. Irá analizando cada una de las posibles preguntas. La que más coincide con la respuesta del usuario, será la que se plantee para que el usuario pueda contestar de nuevo. La separación entre preguntas se marca con una fila de guiones por consola.

```
Maxima coincidencia de lemas hasta ahora: 0
Pregunta elegida hasta ahora: ¿Cuál es tu sexo?
```

```
Possible pregunta: ¿Hay algún momento que te gustaría volver a vivir?
Lemas de la posible pregunta: {'volver', 'vivir', 'momento', 'gustar'}
Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
Lemas que coinciden: 0
```

La pregunta “¿Cuál es tu sexo?” se escoge al azar de entre todas las posibles que no hayan sido preguntadas anteriormente. A continuación, elige la primera pregunta a estudiar: “¿Hay algún momento que te gustaría volver a vivir?”. Al contrario que la anterior, esta se procesa para comparar la similitud con la respuesta del usuario y comprobar si es mejor pregunta que la ya elegida. En las líneas de la salida aparece al principio el número de coincidencias máxima de lemas hasta el momento. Como ninguna pregunta se ha analizado, el contador se encuentra a cero, a la mínima coincidencia, siempre habrá una mejora. Las siguientes líneas en la salida son la pregunta que hasta el momento va ganando como candidata a ser preguntada, la posible pregunta a analizar ahora, los lemas más relevantes de la posible pregunta entre llaves y los lemas de la respuesta entre llaves.

Se extraen los lemas de la posible pregunta y los lemas de la respuesta del paciente, son los siguientes:

Lemas de la respuesta:	hermano, familia, padre
Lemas de la posible pregunta:	volver, vivir, momento, gustar

Como ningún lema coincide y por ello, no se mejora el contador de máximo número de coincidencias, esta pregunta se descarta y se pasa a estudiar la siguiente.

```
Maxima coincidencia de lemas hasta ahora: 0
Pregunta elegida hasta ahora: ¿Cuál es tu sexo?
```

```
Possible pregunta: ¿Dónde viviste cuando eras pequeño?
Lemas de la posible pregunta: {'pequeño', 'vivistar'}
Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}
Lemas que coinciden: 0
```

```
Maxima coincidencia de lemas hasta ahora: 0
Pregunta elegida hasta ahora: ¿Cuál es tu sexo?
```

Possible pregunta: ¿Has vivido en algún lugar diferente cuando eras pequeño?

Lemas de la posible pregunta: {'lugar', 'pequeño', 'vivir'}

Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}

Lemas que coinciden: 0

Estas dos preguntas tampoco tienen lemas que coincidan y no se eligen como candidatas a ser preguntadas.

Maxima coincidencia de lemas hasta ahora: 0

Pregunta elegida hasta ahora: ¿Cuál es tu sexo?

Possible pregunta: ¿Cómo se llaman tus padres?

Lemas de la posible pregunta: {'padre', 'llamar'}

Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}

Lemas que coinciden: 1

Llegados a la cuarta pregunta, “¿Cómo se llaman tus padres?” va a encontrar una coincidencia de tal forma:

Lemas de la respuesta: {hermano, familia, padre}

Lemas de la posible pregunta: {llamar, padre}

Como podemos observar el lema “padre” coincide en ambas, es decir, en esta posible pregunta encontramos una coincidencia. Como el número de coincidencias es mejor que 0, la pregunta candidata “¿Cuál es tu sexo?” se sustituye por la pregunta “¿Cómo se llaman tus padres?”.

Maxima coincidencia de lemas hasta ahora: 1

Pregunta elegida hasta ahora: ¿Cómo se llaman tus padres?

Possible pregunta: ¿Si tienes hermanos, cómo se llaman?

Lemas de la posible pregunta: {'llamar', 'tener', 'hermano'}

Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}

Lemas que coinciden: 1

Maxima coincidencia de lemas hasta ahora: 1

Pregunta elegida hasta ahora: ¿Cómo se llaman tus padres?

Possible pregunta: ¿Cómo era la casa dónde viviste de pequeño?

Lemas de la posible pregunta: {'pequeño', 'casa', 'vivistar'}

Lemas de la respuesta anterior: {'familia', 'padre', 'hermano'}

Lemas que coinciden: 0

El contador de máximo número de coincidencias ahora está a uno y solo cambiará si encuentra alguna posible pregunta que coincida en dos o más lemas con la respuesta. Es por esto que cuando se han analizado el resto de posibles preguntas, no se ha encontrado ninguna que sea mejor porque no superan el número de coincidencias en lemas. Aunque encuentra la pregunta “¿Si tienes hermanos, cómo se llaman?” que coincide también en un lema, el lema “hermano”, no sustituye la cuestión elegida hasta el momento. En conclusión, “¿Cómo se llaman tus padres?” será la siguiente pregunta que se le plantea al usuario, la cual tiene relación con la que se había hecho anteriormente. La conversación queda de la siguiente forma:

IA: ¿Quiénes son las personas más importantes de tu vida?

Tú: Mi familia y concretamente mis padres y mi hermano

IA: ¿Cómo se llaman tus padres?

Tú: ...

3.4. Base de datos MongoDB

MongoDB es un sistema de base de datos NoSQL, es decir no relacional, formado por colecciones de documentos. Los documentos son los “objetos” que conforman las colecciones y es donde se almacenan los datos. El esquema de datos de la BBDD puede variar dinámicamente o incluso ser inexistente. Es decir, los documentos son flexibles y los campos entre unos y otros pueden variar. Además, estos objetos, se guardan en formato JSON y se encierran entre llaves con elementos de tipo “clave:valor”. La clave se asemeja a lo denominado columna en una base de datos relacional, pero como se ha explicado, en Mongo no son fijas y pueden no encontrarse en todos los documentos de una misma colección. Se ha elegido un modelo no relacional porque la información que se quería almacenar no estaba muy clara en un principio y definitivamente no iba a estar estructurada. Desde un principio se pretendía almacenar todo tipo de datos relacionados con los recuerdos de las personas y no se sabía exactamente que distribución se iba a seguir. Además, el objetivo era almacenar información muy distinta según el tipo de recuerdo y según el camino que siguiese el Chatbot en sus conversaciones. Iban a surgir formatos de objetos muy distintos que no podían amoldarse a una tabla fija. Dentro de las NoSQL se eligió, en concreto, MongoDB por su facilidad de uso, la comodidad del formato JSON, la simplicidad de acceso a los datos y su capacidad de consulta para cualquier campo.

El cometido de MongoDB en este proyecto es almacenar los recuerdos de las personas con demencia que se van recopilando desde el Chatbot. De esta forma, la base de datos que se ha construido tiene dos colecciones principales, una de preguntas que plantea el Chatbot y otra de respuestas de los usuarios que se analizan como recuerdos. Se usaron aparte otras colecciones para hacer pruebas sobre ideas que se descartaron o como complemento a las dos anteriores. La colección de preguntas almacena las posibles cuestiones que planteará el bot al usuario junto a categorías varias relacionadas con la pregunta, como las etapas de vida en las que se puede encajar o, etiquetas que la caractericen (ocio, familia, vacaciones, amigos, etc.). En la figura 3.3 se muestra un ejemplo de documento de la colección de preguntas.

Por otro lado, la colección de respuestas contiene la información relativa a lo que ha respondido un usuario específico a una pregunta concreta. Es por eso, que el usuario que ha contestado y la pregunta respondida deben ser campos de esta colección. A continuación se explica cada uno de los campos:

```
{
  "_id": {
    "$oid": "628e373b0e1a7168c6412b7d"
  },
  "pregunta": "¿En qué hospital naciste?",
  "etiquetas": [
    "general",
    "infancia",
    "nacimiento",
    "familia"
  ]
}
```

Figura 3.3: Documento de la colección de preguntas

- “user_id”: Para distinguir al usuario se almacena el identificador que le corresponde una vez registrado en la página web. Los detalles de la aplicación web se explicarán en el capítulo siguiente. Sin embargo, para poder entender de dónde se extrae ese identificador, se explicará brevemente en que consiste. Diferentes usuarios pueden acceder al Chatbot desde la aplicación y para poder distinguir la información que se recopila es necesario tener un control de usuarios. Se crea una tabla en MySQL con todos los usuarios que se van registrando ya a cada uno se le asigna un identificador. Este identificador es el que se va a guardar en la colección de respuestas para poder diferenciar las contestaciones que da cada usuario desde la página del Chatbot. En un principio, cuando la aplicación web no se había creado, no era necesaria la distinción de usuarios porque solo se invocaba al chatbot desde un único terminal asociado a una única persona. Es decir, las respuestas se podían almacenar suponiendo que siempre respondía la misma persona. No obstante, esta solución pierde escalabilidad.
- “pregunta”: Se corresponde con la pregunta que ha elegido el Chatbot para lanzar al usuario. Coincidiría con una de las muchas que están almacenadas en la colección de preguntas que se ha explicado anteriormente.
- “respuesta”: La contestación textual que ha dado el usuario a la pregunta formulada por el bot.
- “categorias”: Una serie de características que explican la respuesta. Entre ellas se encuentra la etapa de vida en la que puede encajarse el recuerdo y el resultado del análisis de sentimiento al que se le ha sometido en forma de “positivo” o “negativo” según las connotaciones reconocidas. Además, se almacenan los lemas extraídos mediante el procesamiento léxico que se ha explicado en la sección anterior. Se trata de los lemas de aquellas palabras del texto que no sean vacías y que definen a la contestación recibida. Se consigue un desglose y un análisis bastante exhaustivo de cada respuesta.

En la figura 3.4 se muestra un ejemplo de documento de la colección de respuestas.

```
{  
    "_id": {  
        "$oid": "63a98341a7f516b27aaefab5"  
    },  
    "user_id": 14,  
    "pregunta": "¿Hay algún momento que te gustaría volver a vivir?",  
    "respuesta": "El dia de mi boda, fue un momento mágico porque estaba  
                rodeada de todas las personas a las que quería",  
    "categorias": [  
        "positivo",  
        "juventud",  
        "querer",  
        "boda",  
        "momento",  
        "persona",  
        "mágico",  
        "rodeado"  
    ]  
}
```

Figura 3.4: Documento de la colección de respuestas

Capítulo 4

Aplicación Web

La aplicación web se ha creado con el objetivo de proporcionar una interfaz web atractiva y manejable tanto para los terapeutas como para sus pacientes. Ahora bien, el chatbot es la esencia de este TFG y, por tanto, es la herramienta principal que se desarrolla en la página web. El chatbot se ha pensado para ser utilizado por los pacientes. La aplicación web permite probar el chatbot de forma mucho más manejable y visible que desde un terminal del entorno de desarrollo. Además, se ha querido añadir más funcionalidades para los terapeutas que, podrán acceder a los recuerdos de sus pacientes de forma sencilla y visual. En este capítulo se explicarán los pasos que se han seguido para desarrollar la página web desde el principio con un prototipo inicial a mano alzada hasta el final con una aplicación web funcional y con numerables vistas.

4.1. Prototipo inicial

Al comienzo de este trabajo, había que ver qué dirección se iba a seguir y por ello, se plantearon diferentes propuestas para luego ser guiada por la más adecuada. Una de las ideas iniciales que se propusieron fue la de crear una aplicación web. Se realizó un esbozo sobre lo que se creía que iba a necesitar un chatbot como aplicación para estar completo. Este prototipo a mano alzada está dividido en dos partes. Por un lado, están las funcionalidades pensadas para el uso del terapeuta, que podrá programar las sesiones que quiere hacer personalmente con el paciente (ver Figuras 4.2, 4.3, 4.4, 4.5, 4.6). Por otro lado, las funcionalidades que están pensadas para el paciente con demencia (ver Figura 4.1) y que hará uso del chatbot en sí. Es importante aclarar que no se planteó como sustituto de las sesiones que tienen los terapeutas con sus pacientes y que las funcionalidades que tiene disponibles en la aplicación son para ayudarle a programar esas sesiones. El chatbot debe estar accesible desde la cuenta de un paciente para facilitar la tarea del terapeuta en la recopilación de sus recuerdos.

Para el usuario con demencia el prototipo es muy sencillo, tiene 2 funcionalidades básicas propias (ver Figura 4.1):

- Chatbot: La pestaña de terapia lleva al usuario al chat donde se le hará preguntas sobre su vida para recabar el máximo número de recuerdos. El chatbot es un complemento aparte que ayuda al terapeuta a recoger la información del usuario pero que no sustituye las sesiones habituales paciente-terapeuta.
- Historia de vida: Los recuerdos recabados se guardan en la aplicación y el usua-

$(\text{usuario} \equiv \text{paciente})$

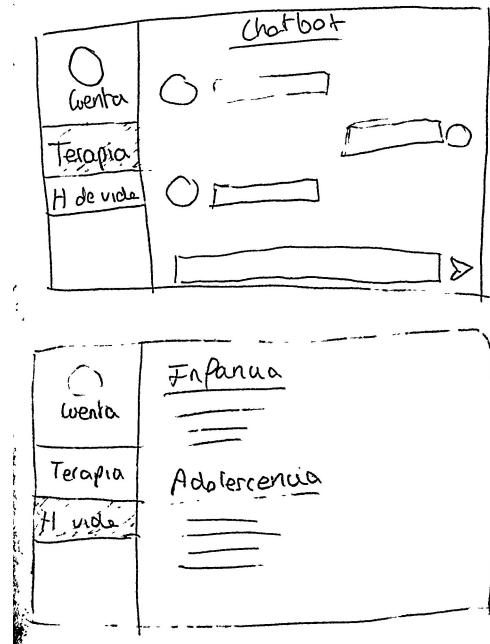


Figura 4.1: Prototipo inicial de la aplicación para el usuario con demencia

rio puede consultarlos divididos por etapas de la vida (infancia, adolescencia, etapa adulta, etc.). Esto sería parte de la terapia ocupacional basada en reminiscencia y un paso previo para construir el libro de vida del usuario. El usuario puede revisitar sus recuerdos a través de la aplicación y esto puede ayudar en el retraso del deterioro cognitivo.

En cuanto a las vistas del terapeuta, tendrá muchas más funcionalidades disponibles:

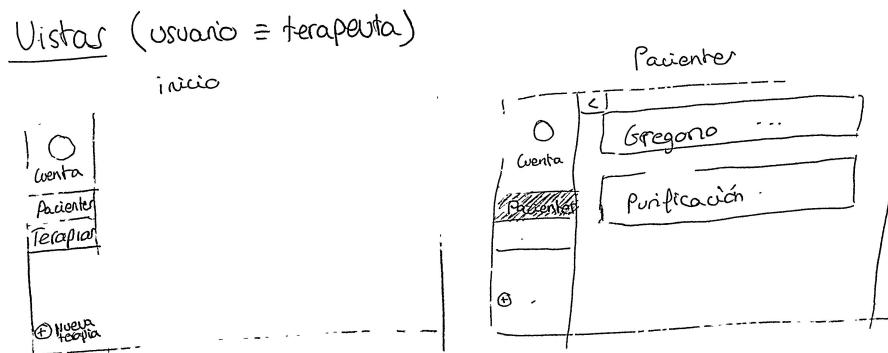


Figura 4.2: Prototipo inicial de la aplicación para el terapeuta: Consultar los pacientes registrados

- **Pacientes:** Se muestra una lista de los usuarios (pacientes) que tiene el terapeuta asignados y que están registrados en la aplicación (ver Figura 4.2).

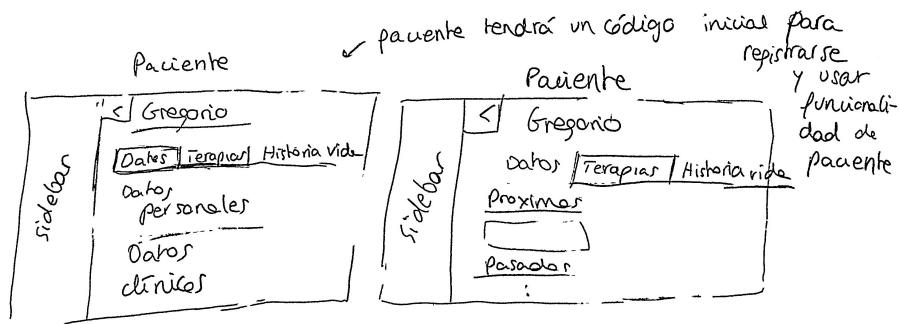


Figura 4.3: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 1

- Datos personales del paciente: Los datos personales y clínicos del paciente estarán disponibles para que el terapeuta los consulte cuando quiera (ver Figura 4.3).
- Terapias del paciente: Se muestran las terapias que se han creado para ese paciente, tanto las que se programaron para fechas pasadas como las que ocurrirán próximamente (ver Figura 4.3). El terapeuta es el que crea estas terapias para sus futuras sesiones con el paciente, las terapias pasadas son las sesiones que se han finalizado y las próximas son las que están pendientes por hacer.

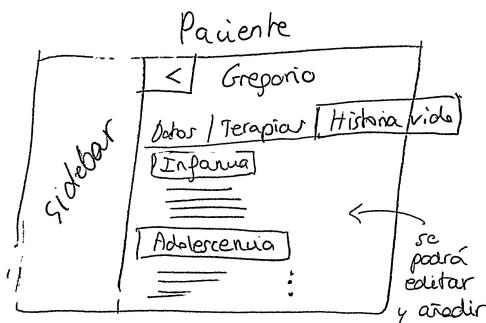


Figura 4.4: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 2

- Historia de vida del paciente: Los recuerdos recabados en las sesiones terapéuticas se guardan en la aplicación y el terapeuta puede consultarlos divididos por etapas de la vida (infancia, adolescencia, etapa adulta, etc.) igual que podía hacer el usuario (ver Figura 4.4). El objetivo de que el terapeuta pueda consultarlos es para revisar que se han grabado los recuerdos correctamente y para comprobar que son precisos y coherentes. En cualquier caso, el terapeuta podrá editar o borrar los recuerdos que crea incompletos o incorrectos y podrá también añadir recuerdos que haya conseguido de alguna otra forma. En caso de que algún recuerdo esté mal clasificado en la etapa de la vida en que ocurrió, también se podría mover a su sitio correcto.

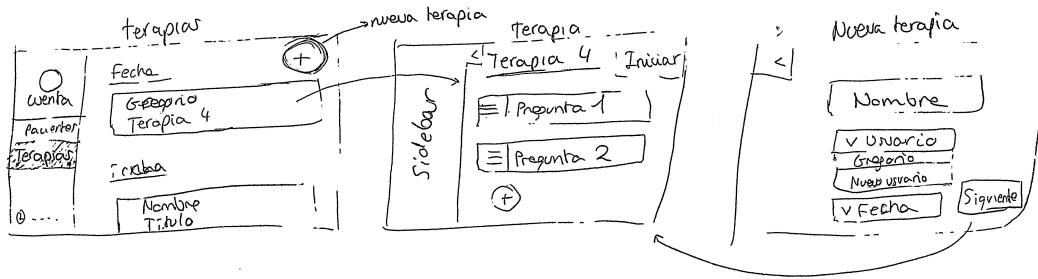


Figura 4.5: Prototipo inicial de la aplicación para el terapeuta: Crear y consultar las terapias creadas

- Terapias: Se muestra una lista ordenada de las terapias que ha ido creando el terapeuta para los distintos pacientes (ver Figura 4.5).
- Consultar terapia: El terapeuta puede revisar una terapia que haya sido ya planificada (ver Figura 4.5). También puede editarla o iniciar la terapia cuando esté con el paciente.
- Nueva terapia: Para crear/programar una nueva terapia es necesario asignarle un nombre, el paciente al que va dirigida y una fecha en la que se llevará a cabo (ver Figura 4.5). El paciente se puede elegir de entre los usuarios que tiene asignados el terapeuta pero, también puede crear un nuevo usuario al que irá dirigida que no es necesario que esté registrado en la aplicación. Después, se crean por cada bloque las preguntas o temas que se quieren sacar en la sesión en el orden que quieran ser utilizados.

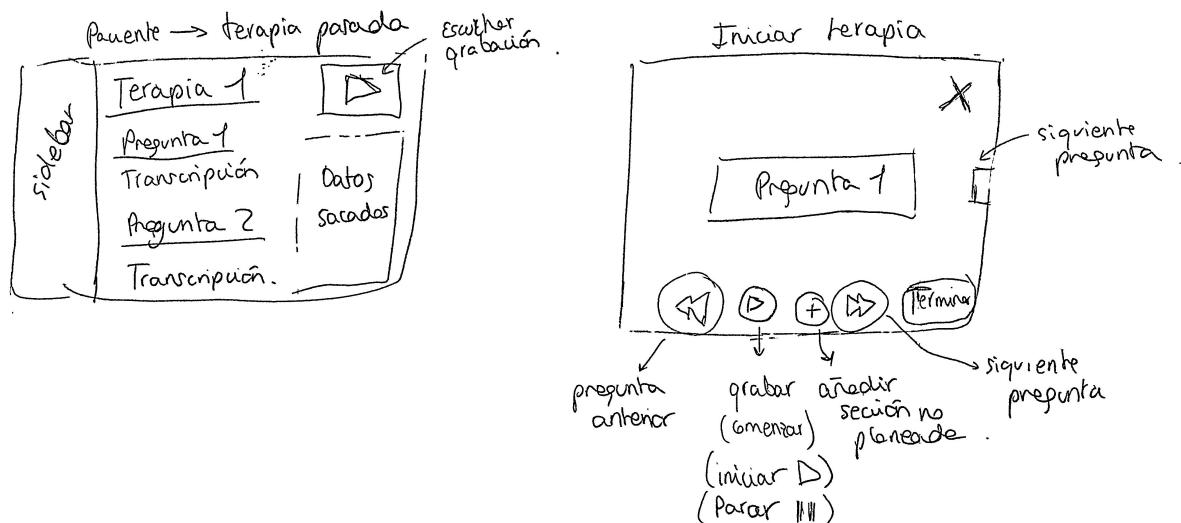


Figura 4.6: Prototipo inicial de la aplicación para el terapeuta: Consultar y reproducir una terapia concreta

- Terapia finalizada: Se puede consultar una terapia que ya se ha realizado (ver Figura 4.6). Se podrá ver toda la información que se ha sacado, apuntes del terapeuta, la grabación de la sesión, la transcripción de la grabación, etc.

- Iniciar terapia: Cuando el terapeuta se reúne con el paciente para la sesión, pulsa a iniciar la terapia y obtendrá en orden los temas y preguntas que le quiere ir sacando al paciente (ver Figura 4.6). Podrá ir pasando de pregunta en pregunta cuando lo crea necesario. Al comenzar, puede poner a grabar la sesión para que luego se haga una transcripción automática de todo lo que ambos dicen. Esa grabación se puede pausar o continuar cuando se quiera. También, se puede añadir una sección nueva que no se contemplase en la planificación pero que se quiera incluir en la terapia. Y finalmente, se puede terminar dar por finalizada la terapia pulsando en terminar.

Este prototipo se descartó porque se centraba mucho en hacer una aplicación web y no tanto en el desarrollo de un chatbot lo suficientemente inteligente para dirigir una sesión de terapia sin necesidad del terapeuta. Con esto, no se desestima la labor del terapeuta que siempre tendría que comprobar que la información recabada por el chatbot es veraz y utilizar esa información para ayudar al usuario con una terapia cara a cara con el usuario. El chatbot es solo una herramienta de apoyo para facilitar el trabajo del terapeuta en conseguir los recuerdos de la vida del usuario.

4.2. Base de datos MySQL

Se han usado dos tipos de bases de datos en el proyecto. La primera es no relacional levantada en MongoDB, explicada en el capítulo anterior, que almacena los datos relacionados con los recuerdos de la persona con demencia y de la interacción con el chatbot. La segunda base de datos es relacional y se ha creado en MySQL. Esta segunda se encarga de almacenar los datos de los usuarios para poder ser identificados e iniciar sesión. Y, por otro lado, almacena los datos de las terapias para relacionar pacientes con terapeutas. A continuación se procede a explicar la segunda base de datos, la MySQL, cuyo uso es exclusivo de la página web.

Se trata de una base de datos ideal para conjuntos de datos que tienen una estructura fija, como en este caso, los datos de inicio de sesión de los usuarios y los datos de las terapias que asocian a los terapeutas con sus pacientes. Es por esto que se ha construido una base de datos con dos tablas fijas y relacionadas, una que identifica a los usuarios y otra que guarda la información sobre las terapias. En la diagrama de entidad-relación de la figura 4.7 se puede ver la relación entre ambas tablas. Se trata de una relación recursiva ya que la tabla de usuarios se relaciona consigo misma de tal forma que la tabla desempeña un rol de terapeuta en uno de los lados de la relación y un rol de paciente en el otro lado. La tabla de terapias solo cumple la función de intermediaria y en ella se tratan dos tipos de usuarios, terapeutas y pacientes.

4.2.1. Tabla de usuarios

Es la tabla que alberga toda la información de los usuarios, tanto terapeutas como usuarios que se distinguirán mediante el campo “type”. Son los dos tipos de usuarios que podrán iniciar sesión en la aplicación. Por cada registro tendremos los campos que se muestran en la figura 4.8 y que se explican a continuación:

- *id*: El entero que representa el identificador de la fila, único y clave primaria para poder diferenciar cada registro inequívocamente
- *name*: El nombre de usuario, una cadena de caracteres, que se utilizará para referirse al navegante en la aplicación web

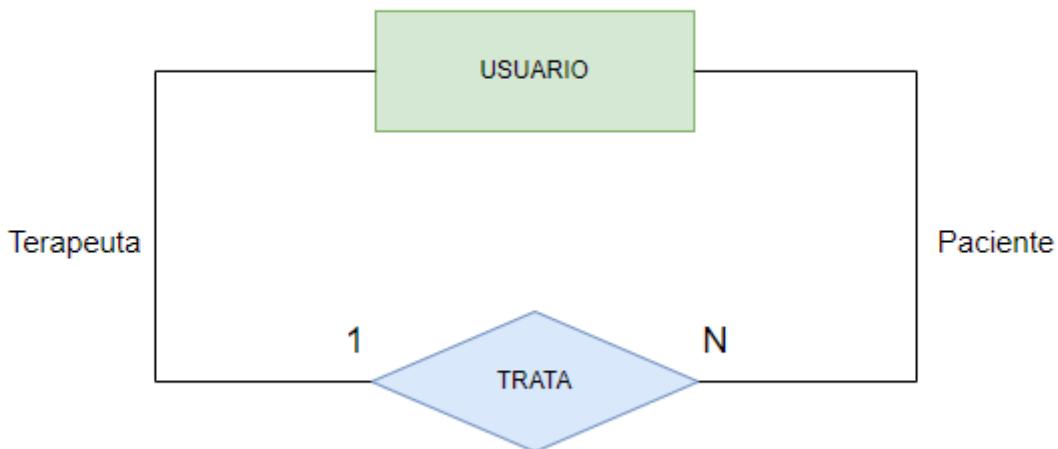


Figura 4.7: Diagrama de entidad-relación MySQL

- *email*: El correo electrónico, una cadena de caracteres, único también para que no puedan existir dos usuarios con el mismo email, lo que supondría un error
- *password*: La contraseña, cadena de caracteres que se guardará cifrada. Cuando se intente iniciar sesión, se comprobará si los datos son correctos y las contraseñas coinciden.
- *type*: Cadena de caracteres que indica si el tipo de usuario es terapeuta o paciente. Es necesario distinguirles porque tendrán acceso a funcionalidades distintas en la aplicación web.

4.2.2. Tabla de terapias

Tabla en la que se guardan las relaciones entre terapeutas y sus pacientes. Por cada registro tenemos los campos que se muestran en la figura 4.9. Las diferentes columnas de la tabla se explican a continuación:

- *id*: El entero que representa el identificador de la fila, único y clave primaria, se utiliza igual que en el de la tabla de usuarios.
- *therapist*: El identificador del terapeuta al que le corresponde la terapia encargado de tratar al paciente que toque.
- *patient*: El identificador del paciente señalado para la terapia
- *code*: El entero que representa un código, único, para que el paciente pueda registrarse como tal en la aplicación y con su terapeuta asignado. Son números elegidos aleatoriamente del 100 al 999. Una vez se registra el paciente, no se vuelve a necesitar este código para nada.
- *user*: Cadena de caracteres utilizada para identificar por nombre a un paciente que aún no se ha registrado pero sí tiene código asignado.

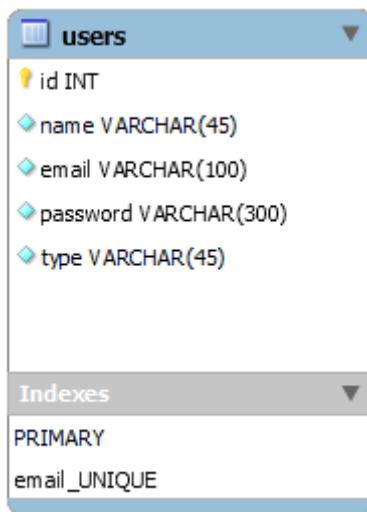


Figura 4.8: Diagrama de la tabla de usuarios

4.3. Implementación final

La aplicación final del chatbot, con la que el usuario con demencia interactuaría, se ha desarrollado usando el *framework* de creación de aplicaciones “Flask”. Se trata de un framework para la creación rápida de aplicaciones web desarrolladas con Python. Se estuvo planteando usar el framework “Django” pero finalmente se optó por Flask porque Django es para aplicaciones mucho más grandes y complejas de lo que se proponía crear para este trabajo. Flask es mucho más intuitivo y fácil de usar, además, con solo un par de líneas se puede levantar una aplicación web sin necesidad de librerías ni herramientas accesorias. Por otro lado, el diseño de la página web se ha desarrollado con hojas de estilos al uso, sin hacer uso de ningún framework para el frontend. Sin embargo, no se ha programado línea a línea sino que se ha replicado el estilo del sitio de administración¹ que proporciona “Django” a todos los desarrolladores de aplicaciones web que utilicen su framework. Como se había trabajado anteriormente con “Django”, se ha querido utilizar su diseño visual por ser conocido y porque el objetivo de la aplicación era mostrar la usabilidad del chatbot, no que fuera visualmente bonito. No se ha trabajado con estilos más elaborados y técnicos para invertir ese tiempo en el desarrollo con Python.

Para empezar con Flask, se utilizó la guía² proporcionada por la página oficial de este framework y un tutorial³ específico para la creación de aplicaciones como esta. Para empezar con Flask lo principal es tener instalado Python e instalar las dependencias necesarias del framework. Esto se puede hacer desde el propio entorno de desarrollo elegido, en mi caso “Visual Studio Code”, usando el instalador de paquetes “pip” de Python. El paquete principal para que la aplicación funcione es el de Flask, en este caso Flask-WTF para que también incluya la validación de formularios, como sería el de inicio de sesión en el que se profundizará más adelante.

Una vez instalados los paquetes, el despliegue de la aplicación es muy sencillo y, tan solo se necesitan unas pocas líneas de código para que funcione. Lo primero sería crear un

¹https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Admin_site

²<https://flask.palletsprojects.com/en/2.2.x/quickstart/>

³<https://j2logo.com/leccion-1-la-primer-a-aplicacion-flask/>

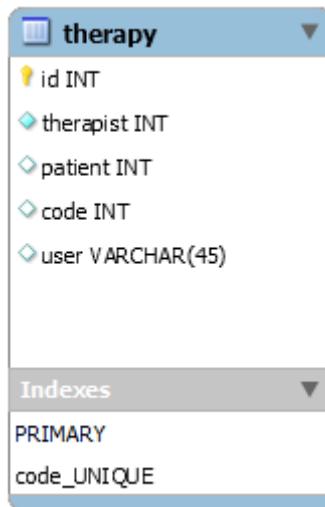


Figura 4.9: Diagrama de la tabla de terapias

fichero python, en nuestro caso llamado “run.py”, en el que irán los métodos asociados a las URLs que formarán la aplicación, el primer método en nuestro caso, sería el que nos lleva a la página principal, el “index”, que es básico para que funcione la aplicación. El fichero run.py más básico para que se despliegue la aplicación quedaría de esta forma:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return render_template("index.html")
```

Para que tenga sentido este fichero sería necesario crear aparte un archivo HTML llamado “index.html” con la lógica de la página principal, que puede ser lo más sencilla que se quiera. Este código funcionará de tal forma que cuando se acceda a la url de la aplicación (identificada también por terminar con este símbolo “/”) se va a cargar la página del index.html. Pero, antes, para arrancar la aplicación se necesita ejecutar varios comandos en el terminal:

```
> $env:FLASK_APP = "run"
> python -m flask run
```

Flask incluye un servidor interno propio al que se podrá acceder desde *localhost*. Para que este servidor sepa qué aplicación debe lanzar, se usa el primer comando que apunta al fichero run del directorio en el que se encuentra la aplicación. El segundo comando es el que definitivamente lanza la aplicación. Podemos comprobar que la aplicación funciona entrando a un navegador y accediendo a la URL “<http://127.0.0.1:5000/>” que cargará el fichero principal index.html.

Una vez tenemos funcionando la aplicación ya solo queda ampliar sus funcionalidades que se explicarán a continuación.

4.3.1. Funcionalidad inicio de sesión y registro

Se añade esta funcionalidad para que cada recuerdo se pueda asociar a un usuario y así dar la posibilidad de guardar incontables historias de vida de diferentes usuarios. De la mano del inicio de sesión, se encuentra la funcionalidad del registro de usuarios que se contará también en esta sección de la memoria. De nuevo, para ambas funcionalidades, se ha utilizado como guía el tutorial de Flask⁴ de la página “J2logo”. En este caso, la página web nos guía en el uso de formularios en Flask, que se requieren tanto para el inicio de sesión como para el registro de usuarios.

Para iniciar sesión en la aplicación (ver Figura 4.10), tanto terapeutas como pacientes deben introducir los campos email y contraseña que se hayan usado en el registro. Si existe en la base de datos un usuario cuyo email coincide con el introducido y, además, la contraseña es correcta se dará acceso al usuario al resto de funcionalidades y se le redirigirá de nuevo a la página principal pero ya identificado. En el caso de que el email no exista o la contraseña sea incorrecta se mostrará el siguiente error: “Error al iniciar sesión”, y se pedirán los datos de nuevo.

Figura 4.10: Funcionalidad inicio de sesión

La página de registro (ver Figura 4.11) es parecida a la de inicio de sesión pero añadiendo varios campos más a llenar. El primero es el del nombre, que se utilizará para que la aplicación se dirija al usuario por ese nombre. El email y la contraseña se guardarán en la base de datos de MySQL y se utilizarán para comprobar si el inicio de sesión es correcto. La contraseña se guarda encriptada mediante un hash gracias a la librería “werkzeug.security” de Python, que proporciona funciones para generar el *hash* y para comprobar si una cadena de caracteres dada coincide con la contraseña encriptada. Después, hay que llenar el campo “Tipo” que indica a la aplicación si el usuario que se registra lo hará como terapeuta o como paciente. En el caso de registrarse un paciente, su terapeuta asociado y ya registrado, le debe haber proporcionado un código que introducirá en el campo “Código”. Esto es necesario para que el usuario se reconozca como paciente del terapeuta correspondiente porque sino el terapeuta no puede acceder a la información del usuario. Una vez registrado el paciente, se crea un nuevo registro que le aparece al terapeuta en el listado de pacientes (ver Figura 4.15) y a partir de ese momento, podrá consultar toda su información. En caso de introducir un código erróneo se muestra el siguiente error: “Error: No existe terapia para ese código”. A nivel backend, el código introducido debe haberse registrado anteriormente

⁴<https://j2logo.com/tutorial-flask-leccion-3-formularios-wtforms/>

en la tabla de terapias y asociado al terapeuta que lo creó. En caso de registrarse un terapeuta, el valor del código no es relevante. Si el email proporcionado ya está registrado por otro usuario, se muestra el siguiente error: “Error: El email ya está siendo utilizado por otro usuario”, y se pedirán los datos de nuevo.

Nombre

Email

Contraseña

Tipo

Código

Registrar

Figura 4.11: Funcionalidad registro

4.3.2. Funcionalidad chatbot

El chatbot (ver Figura 4.12) es la funcionalidad principal de la aplicación y es en lo que a nivel de backend se ha invertido más tiempo. Sin la parte visual del frontend, el chatbot también funciona desde la línea de comandos de Python. Sin embargo, se ha decidido hacer la aplicación web para facilitar la interacción con el usuario. Esta funcionalidad permitirá al usuario interactuar con el bot que hay detrás a través de un chat. Se distingue de otras inteligencias artificiales de tipo chatbot porque es el bot quien dirige la conversación y el que pregunta o habla con el usuario y no al revés. Por ejemplo, el usuario es el que hace peticiones a Alexa que interpreta lo que ha dicho el usuario y le ofrece un servicio. Sin embargo, el chatbot “Ricorda” es el que guía la conversación.

En primer lugar, el bot saluda al usuario, le da la bienvenida y explica que comenzará a hacer preguntas. La primera pregunta se escoge al azar de entre los documentos almacenados en la colección de preguntas. Además, se filtran aquellos que contengan preguntas que ya hayan sido preguntadas al usuario en cuestión. Dada la primera pregunta, se espera a que el usuario responda, solo podrá mandar un único mensaje por cada pregunta porque está configurado de tal forma que responde tras procesar solo un mensaje. La respuesta del usuario se trata y almacena como se explica en el capítulo de la arquitectura del chatbot, es decir, se clasifica según la etapa de vida a la que corresponde (infancia, juventud, etapa adulta y vejez), se clasifica en positivo o negativo con el analizador de sentimientos del texto, se almacena en la colección de respuestas de la base de datos de mongo junto al id del usuario, la pregunta y las categorías anteriores y, finalmente, se le somete a un

análisis léxico para conseguir la mejor siguiente pregunta, la que mayor relación encuentre con la respuesta que ha dado el usuario. Después del análisis de la respuesta del usuario, se lanza la siguiente pregunta aquella que más se corresponda con la respuesta del usuario. La persona volvería a responder y el bot a preguntar hasta que se canse de contestar a las preguntas. Todas las respuestas se almacenan y van así formando la historia de vida del paciente, recuerdo a recuerdo. Estos recuerdos se podrán consultar en la propia página web clasificados en etapas de vida, funcionalidad que se explica más adelante.



Figura 4.12: Funcionalidad chatbot

El diseño de esta vista esta inspirado en el Chatbot web de la página sobre inteligencia artificial “Buff ML”⁵

4.3.3. Funcionalidad usuario terapeuta vs usuario paciente

Dentro de la aplicación existen dos roles para los usuarios, el terapeuta podrá acceder al listado de sus pacientes y a los recuerdos ya recogidos por el chatbot. Por otro lado, el paciente podrá acceder al chatbot con el que interactuará para sacar el mayor número de recuerdos y también consultar toda la información guardada sobre él y sus respuestas.

Los usuarios pacientes podrán acceder a un menú concreto pensado para que puedan acceder a las funcionalidades que les interesan. El menú de los pacientes se puede ver en la Figura 4.13. El menú para los usuarios terapeutas es distinto y está pensado para que puedan acceder directamente a las funcionalidades que les corresponden. Se puede ver en la Figura 4.14. Estos menús siempre están disponibles en la esquina superior derecha de la página.

⁵<https://buffml.com/web-based-chatbot-using-flask-api/>

PEPE GONZALEZ | MIS RECUERDOS | CONVERSAR | LOGOUT

Figura 4.13: Menú de los pacientes

LUCIA | PACIENTES | CONVERSAR | LOGOUT

Figura 4.14: Menú de los terapeutas

4.3.4. Funcionalidad consulta de pacientes registrados por parte del terapeuta

Un terapeuta debe tener la posibilidad de ver una lista de pacientes que estén a su cargo y a los que trata en terapias. Esta funcionalidad está disponible en el apartado “pacientes” del menú (ver Figura 4.15). En esta página tendrá acceso a una tabla con todos sus pacientes que tiene dos columnas, la primera indica el nombre con el que están registrados los pacientes y, la segunda muestra el email asociado a cada uno. Si lo que quiere es obtener la información de uno de sus pacientes, es decir, los recuerdos que ha ido introduciendo el usuario y recogiendo el chatbot divididos por etapas vitales, se puede ver pulsando en la fila correspondiente, concretamente, en el enlace que hay en el nombre de esa persona. Será así redirigido a la página de historia de vida de un paciente. Encima de la tabla de pacientes, en la esquina superior derecha de la página también hay un botón que dice “Nuevo paciente” con el que se accede a la página de creación de un nuevo paciente. Se trata de una funcionalidad para registrar a un nuevo usuario asociado al terapeuta para que pueda tener acceso a la aplicación y por ende, al chatbot.

4.3.5. Funcionalidad crear nuevo paciente

Se trata de una funcionalidad exclusiva del terapeuta. Desde la página del listado de pacientes se accede a la página de creación de un nuevo paciente clicando el botón de la esquina superior derecha “NUEVO PACIENTE”. Una vez pulsado se muestra algo parecido a lo reflejado en la Figura 4.17, por un lado, en la parte de arriba se encuentra un formulario para registrar a un paciente. Hay dos datos claves, el nombre, que debe rellenarlo el terapeuta, y el código, un número aleatorio, por seguridad, del 100 al 999 (a priori no se necesita un rango mayor) que viene dado por el programa y que no ha sido utilizado antes para otro usuario. El código es necesario para que el paciente pueda registrarse en la página de registro de usuarios y es tarea del terapeuta comunicárselo para que ambos usuarios estén conectados gracias a la tabla de terapias. El nombre se introduce para que el terapeuta pueda reconocer el código fácilmente gracias a estar etiquetado. Esto es importante para la segunda parte de la página, la que se encuentra debajo del formulario. El terapeuta va a tener siempre acceso a los códigos que ya ha registrado. Es decir, se muestra una tabla con dos columnas, la del usuario y la del código. La del usuario se corresponde con el campo “nombre” del formulario y la del código muestra el número que debe introducir el paciente para poder registrarse. Cuando el paciente se registra con su código, la fila correspondiente se elimina de la tabla de usuarios sin registrar y, así, el terapeuta puede tener un control de los usuarios dados de alta en la aplicación, a los que puede ir haciendo un seguimiento. También puede controlar los usuarios que todavía faltan por registrar, a los que a lo mejor es necesario recordar el código.

A nivel base de datos, cuando se valida este formulario del código, se añade un registro

The screenshot shows a web application interface for 'Ricorda'. At the top, there is a dark blue header bar with the word 'Ricorda' in white. To the right of the header are links for 'LUCIA | PACIENTES | CONVERSAR | LOGOUT'. Below the header, the main content area has a title 'Pacientes' on the left and a 'NUEVO PACIENTE' button on the right. A table lists four patients with their names and emails:

NOMBRE	EMAIL
Carmen Lopez	carmen.lopez@gmail.com
Ines Castaños	inescastanos_91@gmail.com
Pepe Gonzalez	pepe@gmail.com
Julia	julia.martinez@gmail.com

Figura 4.15: Funcionalidad consultar pacientes registrados

en la tabla de terapias de la base de datos MySQL con los siguientes cinco campos:

- Id de la terapia, clave primaria
- Id del terapeuta al que estará asociado el paciente
- Id del paciente, inicializado a “null” porque no existirá ese usuario en la tabla de usuarios hasta que no se registre
- Código
- Nombre con el que se asocia el código.

Cuando el paciente se registre en la aplicación, en el campo de id del paciente se introducirá el id del nuevo usuario. Además, tanto el campo de código como el de nombre se pondrán a “null” porque ya no son necesarios. El número del código se puede reutilizar. En la Figura 4.16 se puede ver que el único usuario no registrado para el terapeuta con id “1” es el de Mario. El resto de terapias tienen los ids de ambos usuarios rellenos. El registro con id “9” de la tabla de terapias es el que sale en la tabla de la parte inferior de la página de la Figura 4.17 porque es el único sin registrar.

4.3.6. Funcionalidad recopilación de los recuerdos de un paciente

Los recuerdos que se van almacenando para cada usuario en la base de datos se van a poder consultar en esta vista (ver Figura 4.18). La historia de vida de una persona se va

id	therapist	patient	code	user
1	1	9	NULL	NULL
3	1	11	NULL	NULL
7	1	4	NULL	NULL
9	1	NULL	577	Mario
10	1	14	NULL	NULL

Figura 4.16: Tabla de terapias en MySQL

construyendo desde su infancia hasta los últimos años de vida. Esta progresión es lo que pretende reflejar esta funcionalidad. Empezando con la infancia y terminando por la vejez, se muestran los recuerdos del usuario clasificados en cuatro etapas: infancia, juventud, etapa adulta y vejez. Esta categorización se obtiene de la base de datos de Mongo, en concreto de la tabla de respuestas que da el usuario al chatbot. Antes de ser almacenados en la base de datos, los recuerdos se analizan con el clasificador de etapas y sentimientos (explicado en el capítulo de arquitectura del chatbot) y, luego, se guardan junto a las categorías. En esta vista, para cada etapa se enumeran los recuerdos (respuestas al chatbot) correspondientes junto a la pregunta que se le hizo al usuario. Si alguna de las cuatro etapas no tiene recuerdos asociados, no se muestra el apartado.

**Usuarios sin registrar:**

USUARIO	CÓDIGO
Mario	577

Figura 4.17: Funcionalidad crear nuevo paciente

The screenshot shows a web application titled "Ricorda". At the top right, there is a navigation bar with links: "LUCIA | PACIENTES | CONVERSAR | LOGOUT". The main content area is divided into four sections: "INFANCIA", "JUVENTUD", "ETAPA ADULTA", and "VEJEZ". Each section contains a list of questions and their corresponding answers.

INFANCIA

- ¿En qué fecha naciste?
En 1943
- ¿Dónde naciste?
Naci en Madrid en el hospital Gregorio Marañón
- ¿Recuerdas mas momentos felices?
Cuando nació mi hermano pequeño Juan fui muy feliz porque era un bebé al que podía cuidar y con el que jugar

JUVENTUD

- ¿Hay algún momento que te gustaría volver a vivir?
El dia de mi boda, fue un momento mágico porque estaba rodeada de todas las personas a las que quería

ETAPA ADULTA

- ¿Cuál ha sido el momento más feliz de tu vida?
Cuando nació mi primer hijo Andrés
- ¿Ibas a algun sitio a veranear?
Sí, solíamos ir a la playa en julio a Valencia y en agosto mis hermanos y yo íbamos al pueblo

VEJEZ

- ¿Quiénes son las personas más importantes de tu vida?
Mis hijos y mis nietos

Figura 4.18: Funcionalidad recopilación de recuerdos del paciente

Capítulo 5

Conclusiones y Trabajo Futuro

El objetivo de este TFG era construir un Chatbot lo suficientemente inteligente como para conseguir recopilar información sobre la vida de una persona con demencia y almacenarla de forma estructurada. Aunque todavía se puede realizar mucho trabajo a partir de lo ya desarrollado, se ha conseguido construir una aplicación funcional que sirve como herramienta de apoyo, relativamente autónoma, en la dura tarea de redactar una historia de vida.

En un principio se había pensado desarrollar un bot conversacional con el que poder mantener una conversación sofisticada y coherente. Con el paso del tiempo, al encontrarse con numerosas trabas y viendo la complejidad que suponía simular una entrevista del terapeuta, se fue distorsionando un poco la idea inicial. Se ha terminado creado una herramienta bastante útil porque cumple con la función de recoger la información que proporciona el usuario aunque no tan inteligente como se proponía en un principio, sobre todo a la hora de darle humanidad. Escasea la capacidad de ayudar a la persona con demencia a recordar mediante estímulos y a través de una conversación fluida.

En cualquier caso, se ha conseguido realizar un análisis bastante preciso de las respuestas que va dando el usuario a través del chat. Se logra estructurar los recuerdos en las etapas de la vida de la persona a las que corresponden de forma bastante exacta al igual que ocurre con el análisis de sentimiento de los textos. Aunque podrían hacer una distinción más concreta, sobre todo en el caso de recuerdos que no entran en ninguna de las categorías proporcionadas, se deja como posible trabajo futuro. Esto es porque para las tecnologías que se han utilizado, el volumen de datos era bastante pequeño, hubiese funcionado mejor con un dataset mucho más grande de recuerdos. Por otro lado, la conversación se hace más fluida gracias al módulo de encadenamiento de respuestas y preguntas que analiza la contestación del usuario para encontrar la mejor siguiente pregunta a plantear. Con un mayor análisis del contexto de la frase hubiese mejorado mucho la elección de preguntas coherentes y fluidas.

A nivel aplicación web, es bastante manejable e interactiva, ofreciendo una interfaz sencilla de utilizar y dirigida tanto a pacientes como a terapeutas. Ofrece un control sobre los usuarios y sobre la información que se va recopilando. No se han añadido más funcionalidades porque el principal foco debía ser el Chatbot y, además, ya existían otros TFGs que se centraban en un desarrollo web más específico.

En lo relativo al trabajo futuro, se proponen las siguientes mejoras:

- Un análisis de sentimiento y clasificación en etapas de vida más exacto y concreto. Una mejora de este análisis utilizando un dataset de recuerdos mucho más grande

para el entrenamiento del modelo.

- Una conversación más fluida con el usuario. Aparte de preguntar al usuario sobre su pasado, implementar un módulo conversador que simplemente reaccione a las respuestas, siempre acorde con lo que ha dicho la persona. También incluir un programa para completar la información que falte por conocer del usuario según una plantilla a llenar.
- Añadir alguna funcionalidad extra a la aplicación web como definir terapias concretas dirigidas a una temática específica, permitir al terapeuta añadir preguntas a la batería disponible en la base de datos, habilitar la opción de reconocimiento de voz para que al paciente con demencia le resulte mucho más fácil la interacción con el Chatbot, etc.
- Una mejor elección de las cuestiones que se le planteen al usuario. Aparte de comparar la respuesta dada por el usuario para elegir la siguiente pregunta, también comparar con la cuestión que se hizo a la que el usuario contestó. Por ejemplo, si se ha preguntado por algo relacionado con la infancia, que se siga preguntando por cosas relacionadas con la infancia. También se pueden implementar ontologías para que al analizar las respuestas se puedan relacionar palabras o conceptos con posibles preguntas a formular.
- Ampliar la batería de preguntas y de temas a tratar mediante la adaptación de entrevistas tipo que realizan los terapeutas ocupacionales.
- Crear la posibilidad de que las preguntas se puedan generar solas, sin estar previamente redactadas, según la temática que se está tratando y los datos que va proporcionando el paciente.
- Mayor categorización de los recuerdos con etiquetas como familia, amigos, ocio, comida, vacaciones, aficiones, etc.r

Chapter 5

Conclusions and Future Work

The aim of this TFG was to build a Chatbot intelligent enough to collect information about the life of a person with dementia and store it in a structured way. Although there is still a lot of work to be done on what has already been developed, a functional application has been built that serves as a relatively autonomous support tool in the hard task of writing a life story.

Initially, the idea was to develop a conversational bot with which a sophisticated and coherent conversation could be held. As time went by, as we encountered numerous obstacles and saw the complexity of simulating a therapist's interview, the initial idea was distorted a little. In the end, a useful tool has been created because it fulfils the function of collecting the information provided by the user, although it is not as intelligent as initially proposed, especially when it comes to giving it the human touch. It lacks the capacity to help the person with dementia to remember through stimuli and a fluid conversation.

In any case, a pretty precise analysis of the answers given by the user through the chat has been achieved. The tool manages to structure memories into the stages of the person's life to which they correspond in a fairly exact way, as happens with the sentiment analysis of the texts. Although a more concrete distinction could be made, especially in the case of memories that do not fall into any of the categories provided, it is left as a possible future work. This is because for the technologies used, the volume of data was quite small, and would have worked better with a much larger dataset of memories. On another note, the conversation is made more fluid by the answer and question chaining module that analyses the user's answer to find the best next question to ask. Further analysis of the context of the sentence would have greatly improved the choice of coherent and fluid questions.

At the web application level, it is quite manageable and interactive, offering a user-friendly interface aimed at both patients and therapists. It offers control over the users and over the information that is collected. More functionalities have not been added because the focus should be the Chatbot and, in addition, there were already other works (TFGs) that focused on a more specific web development.

In terms of future work, the following improvements are proposed:

- More accurate and concrete sentiment analysis and life-stage classification. An improvement of this analysis by using a much larger memory dataset to train the model.
- A more fluid conversation with the user: apart from asking about their past, implement a conversational module that simply reacts to the answers, always according to what the person has said. Also include a program to complete the missing infor-

mation about the user according to a template to be filled in.

- Add some extra functionality to the web application such as defining specific therapies aimed at a specific topic. Other functionalities could be allowing the therapist to add questions to the battery available in the database, enabling a voice recognition option to make it much easier for the dementia patient to interact with the Chatbot, etc.
- A better choice of the questions posed to the user: apart from comparing the answer given by the user to choose the next question, also compare with the question that was asked to which the user answered. For example, if a question has been asked about something related to childhood, keep asking about things related to childhood. Ontologies can also be implemented so that when analysing the answers, words or concepts can be related to new possible questions to be asked.
- Expand the battery of questions and topics to be addressed by adapting the standard interviews conducted by occupational therapists.
- Create the possibility that questions can be generated on their own, without being previously drafted, according to the topic being dealt with and the data provided by the patient.
- Greater categorisation of memories with labels such as family, friends, leisure, food, holidays, hobbies, etc.

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el cerebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra