Chatbot para la recuperación de información personal

Trabajo de Fin de Grado
Curso 2021–2022

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

7 de noviembre de 2022

Dedicatoria

Texto de la dedicatoria...

Agradecimientos

Texto de los agradecimientos

Resumen

Resumen en español del trabajo

Palabras clave

Máximo 10 palabras clave separadas por comas chatbot, reminiscencia, demencia, recuerdo

Abstract

Abstract in English.

Keywords

10 keywords max., separated by commas.

Índice

Índice de figuras

Índice de tablas

Chapter 1

Introduction

Introduction to the subject area.

Capítulo 1

Introducción

"Frase célebre dicha por alguien inteligente"
— Autor

Introducción temporal

Las personas con Alzheimer u otros tipos de demencia pueden beneficiarse del uso de la llamada terapia basada en reminiscencia, que se basa en la construcción de un libro de vida del paciente que recopila recuerdos positivos de su vida que se pueden utilizar posteriormente par ejercitar su memoria y retrasar el deterioro, además de permitir 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

Introducción al tema del TFG.

1.2. Objetivos

Este proyecto tiene como objetivo desarrollar un chatbot mediante el cual recabar información personal sobre la vida del paciente 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.

La clasificación de recuerdos se hará en base a unos criterios predefinidos por expertos en terapia ocupacional del proyecto CANTOR. Se clasificará en función de:

- **Emoción:** Se clasificarán los recuerdos en positivos y negativos, siendo estos últimos para identificar qué recuerdos no deben tratarse en las

terapias por afligir al paciente. Los positivos se puntuarán de 0 a 10 en función de la felicidad que le traen al paciente.

- **Etapa:** Los recuerdos pertenecerán a una de las siguientes etapas: infancia, adolescencia, edad adulta o tercera edad según el periodo temporal en que aconteció.
- **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...

1.3. Plan de trabajo

El plan de trabajo ha consistido de tres etapas:

- Investigación y construcción de prototipo en la que se ha consolidado la idea del TFG, investigado sobre demencia y terapia ocupacional basada en reminiscencia, elegido tecnologías y creado un prototipo de análisis de texto usando spaCy para identificar si un recuerdo es positivo o negativo.
- Programación del Chatbot y desarrollo de la memoria
- Pruebas, revisión de la memoria y entrega

Capítulo 2

Estado de la Cuestión

Introducción de lo que voy a hablar en el estado de la cuestión y por qué

2.1. Demencia

La demencia [1] 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. Trabajar los recuerdos de una persona que sufre demencia ayuda a retrasar los efectos de la misma. Hablaremos para ello de la terapia ocupacional basada en reminiscencia.

2.2. Terapia ocupacional basada en reminiscencia

La terapia ocupacional [2] 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 a realizar o el entorno para que pueda llevarlas a cabo.

(Sacar información del seminario cantor) 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

[1] https://www.alz.org/alzheimer-demencia/que-es-la-demencia?lang=es-MX
[2] https://aptoca.org/terapia-ocupacional/que-es-la-terapia-ocupacional-2/

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.

2.3. Trabajo previo

Herramienta de ayuda guiada para la reminiscencia (rem) : Generación de historias a partir de una base de conocimiento: recomendación de temas a tratar en la terapia + aplicación web que enlaza situaciones y vivencias mediante grafos y luego permite añadir recursos fotográficos asociado a un tema. (Más como un chatbot que va sugiriendo temas a tratar)

Sistema de asistencia para cuidados de enfermos del Alzheimer (asi) : Página que guarda información sobre pacientes y terapeutas asociados. Información relevante + historia de vida formada por instancias de recuerdos

Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning (pre) : chat desplegado con telegram. De las fotos que tiene archivadas va preguntando al usuario cosas relacionadas con la imagen

Generación de resúmenes de video-entrevistas utilizando redes neuronales (res) : transcripción de video-entrevistas a texto

Extracción de información personal a partir de redes sociales para la creación de un libro de vida (rrs)

Alameda Salas, María Cristina (2022) Generación de historias de vida usando técnicas de Deep Learning.

Barquilla Blanco, Cristina y Díez García, Patricia y Mulas López, Santiago Marco y Verdú Rodríguez, Eva (2022) Recuérdame: Aplicación de apoyo para el tratamiento de personas con problema de memoria mediante terapias basadas en reminiscencia.

García González, Hugo (2022) Extracción de recuerdos de vídeos de entrevistas con personas con problemas de memoria.

2.4. PLN

El procesamiento de lenguaje natural

2.4.1. Chatbots

2.4.2. Análisis de sentimiento

El análisis de sentimiento [3] 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 [4] 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.

Las principales herramientas para el análisis del sentimiento son:

- Lingmotif [5] → 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 Finder [6] → 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.

- LIWC [7] → "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.

[3] https://gaeapeople.com/marketing-estrategia/sentiment-analysis
[4] https://blog.pangeanic.es/funcionamiento-herramientas-analisis-sentimiento-basadas-en-inteligencia-artificial
[5] https://ltl.uma.es
[6] https://mpqa.cs.pitt.edu/opinionfinder/
[7] https://www.liwc.app

Capítulo 3

Arquitectura

3.1. Prototipo primero de análisis de sentimientos de un texto

Se ha probado una forma de clasificar texto utilizando TextCategorizer [1] de la librería spacy de python. Se trata de entrenar un modelo para saber identificar si un recuerdo es positivo o negativo. Se ha utilizado el código del artículo *How to Train Text Classification Model in spaCy* [2] para probar cómo se podría entrenar el modelo para que supiese diferenciar los recuerdos negativos de los positivos y así tener la primera clasificación que pidieron los terapeutas especializados del proyecto CANTOR. Antes de probar con recuerdos se probó con un dataset muy grande de reseñas de moda que se ponía como ejemplo en el artículo del que hablábamos anteriormente. Después, se ha probado usando una batería menor de recuerdos positivos encontrados entre los archivos del TFG de Laura Castillo (rem). También había que incluir recuerdos negativos, como no se ha encontrado ningún dataset ni textos relacionados, se han usado frases negativas aleatorias.

El prototipo funciona siguiendo una serie de pasos. En primer lugar, se añade el componente TextCategorizer (textcat) a un modelo en blanco del idioma español. Un modelo en blanco es un modelo que no tiene ningún componente de spaCy definido (como serían NER que explicaré más adelante), es decir, el texto que se almacena en los documentos de spaCy no es analizado por ninguna cadena de procesos porque la tubería de componentes está vacía. Si hubiésemos cogido un modelo pre-entrenado como "es_core_news_sm", el clasificador de textos se sumaría al trabajo previo de los procesos que analizan el texto como serían ['tok2vec', 'morphologizer', 'parser', áttribute_ruler', 'lemmatizer', ñer']. Por ejemplo, el componente

[1] https://spacy.io/api/textcategorizer
[2] https://www.machinelearningplus.com/nlp/custom-text-classification-spacy/

NER, más reconocido como *Named Entity Recognizer*, reconoce entidades dentro del texto como nombres de personas, fechas o lugares. Al usar el modelo en blanco empezamos con el español de cero, sin analizar. Tras añadir el componente textcat, al crear un documento de spaCy, 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 configurándolo, el prototipo coge el texto que usaremos para entrenar nuestro modelo y lo prepara adecuándolo al formato que entiende el clasificador, siendo éste una lista de tuplas (texto, etiqueta). Además, cogerá un porcentaje de esta batería de recuerdos (% que previamente hemos definido) y lo reservará para la evaluación de las predicciones, es decir, para analizar cómo de bien predice nuestro modelo tras entrenarlo.

Ya tenemos casi todo preparado para comenzar a entrenar el modelo, se dividen los casos de entrenamiento en lotes que se analizarán y evaluarán un número definido de iteraciones para asegurar que el entrenamiento es lo más preciso posible sin pasarnos de vueltas para que sea óptimo. Se trata de un proceso iterativo en el que las predicciones del modelo se comparan con las etiquetas de referencia para estimar el gradiente de la pérdida. El modelo se entrena utilizando una función que lo analiza y actualiza en cada iteración, la función update(). También se comprueban las predicciones del modelo en cada vuelta, se comparan con las etiquetas de referencia para estimar la desviación de la pérdida.

Una vez afinado el modelo mediante el entreno previo, ya podemos ponerlo a prueba. El prototipo sacará la probabilidad de que un texto procesado por spacy sea un recuerdo positivo y la probabilidad de que sea uno negativo.

———Meter problemas probando el neutro

3.2. Módulo primero de encadenamiento entre 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 texto Spacy utilizando el código del TFG de Laura Castilla Castellano (Generación de historias a

partir de una base de conocimiento), en concreto el código que aparece en el archivo "analysis.py" de su código, que contiene 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. Solo los lemas de las palabras importantes del texto serán usados 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. Módulo primero de clasificación de respuestas en etapas de vida y en sentimientos

Para categorizar las respuestas según el sentimiento usaremos directamente el prototipo del apartado de análisis de sentimientos de un texto. Se ha añadido un archivo "analyze_answer" que entrena primero al modelo con textos positivos y negativos y luego evalúa el texto que le llega e identifica si es negativo o positivo. Esto se añade como una categoría que se meterá en la base de datos junto a la respuesta. Este análisis de sentimiento también servirá para identificar temáticas dolorosas para el interrogado que no deberían ser usadas en las terapias.

Otra categoría que se añadirá junto a la respuesta será la de la etapa de la vida a la que corresponde. Primero tenemos que definir esas etapas y para ello me he basado en la clasificación que hace el Ministerio de Salud de Colombia (

https://www.minsalud.gov.co/proteccionsocial/Paginas/cicloVida.aspx) porque no he encontrado ninguna clasificación tan clara para España y me precía una forma sensata de clasificación. Le he hecho algunas modificaciones para adecuarme más a lo que quería sacar en claro de la información que se me presentaba. Las etapas serían las siguientes:

- Infancia de 0 a 11 años
- Adolescencia de 12 a 17 años
- Juventud de 18 a 26 años
- 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 periodos de tiempo en los que se divide la vida de las personas, el objetivo es 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 el prototipo primero de 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, que toda la información esté bien estructurada en periodos de tiempo.

—- Meter problemas con la etapa de vida indeterminada

Se ha probado con neutro pero no funcionaba igual de bien, se necesitan muchos más textos de entrada. Para poder hacer este análisis se han utilizado los recursos de la una pagina web [3] de demostración del uso de spacy. El manual de usuario de spacy [4] no ha resultado ser muy explicativo.

3.4. Bases de datos

Se han usado dos tipos de bases de datos, una relacional en MySQL encargada de almacenar datos de inicio de sesión para la página web del chatbot. La otra base de datos es no relacional levantada en MongoDB que almacena los datos relacionados con los recuerdos de la persona con demencia y de la interacción con el chatbot.

[3] https://www.machinelearningplus.com/nlp/custom-text-classification-spacy/
[4] https://spacy.io/api/textcategorizer

3.4.1. MySQL

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. Se ha construido una base de datos con una única tabla de usuarios en la que para cada registro tendremos el identificador, único y clave primaria; el nombre de usuario; el correo electrónico, único también, y la contraseña, que se guardará cifrada. Cuando se intente iniciar sesión, se comprobará si los datos son correctos.

3.4.2. MongoDB

3.5. Aplicación web

La aplicación web se ha creado para que el chatbot sea más visible y manejable.

3.5.1. Prototipo inicial

Al inicio de este trabajo, había que ver qué dirección se iba a seguir, en esos momentos yo planteaba a mis tutores diferentes ideas que se me iban ocurriendo y ellos me guiaban por la propuesta más adecuada. Una de las ideas que se propusieron era la de este prototipo de aplicación web inicial. Realicé un esbozo sobre lo que creía que iba a necesitar el 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 3.2, 3.3, 3.4, 3.5, 3.6). Por otro lado, están las funcionalidades pensadas para el paciente con demencia (ver Figura 3.1) y que hará uso del chatbot en sí. Es importante aclarar que el terapeuta tiene sus sesiones con el paciente como se hace habitualmente y las funcionalidades que tiene disponibles en la aplicación son para ayudarle a programar esas sesiones. El chatbot es solo accesible desde la cuenta de un paciente porque es una herramienta de apoyo para el terapeuta para facilitar la recolección de recuerdos del paciente y, conseguir la información de otra forma para así ganar tiempo.

Para el usuario con demencia el prototipo es muy sencillo, tiene 2 funcionalidades básicas propias (ver Figura 3.1):

- 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.

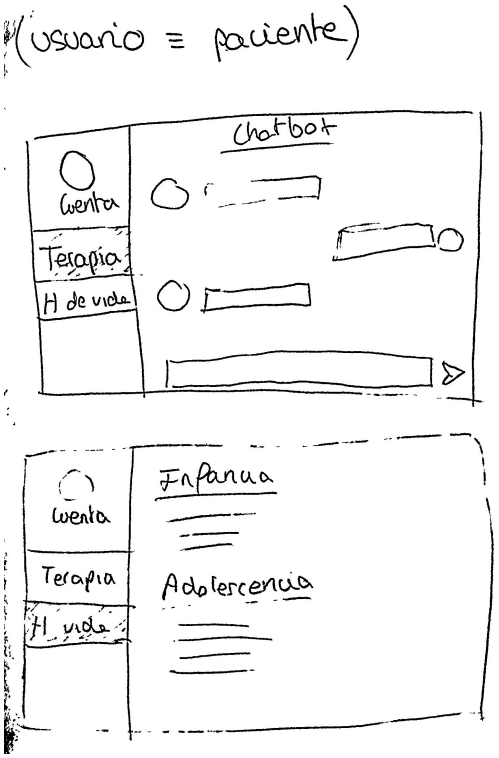

Figura 3.1: Prototipo inicial de la aplicación para el usuario con demencia

- Historia de vida: Los recuerdos recabados se guardan en la aplicación y el usuario 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:

Ver Figura 3.2 →

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

Ver Figura 3.3 →

- Datos personales del paciente: Los datos personales y clínicos del paciente estarán disponibles para que el terapeuta los consulte cuando quiera.
- Terapias del paciente: Se muestran las terapias que se han creado para ese paciente, tanto las que se programaron para fechas pasadas como

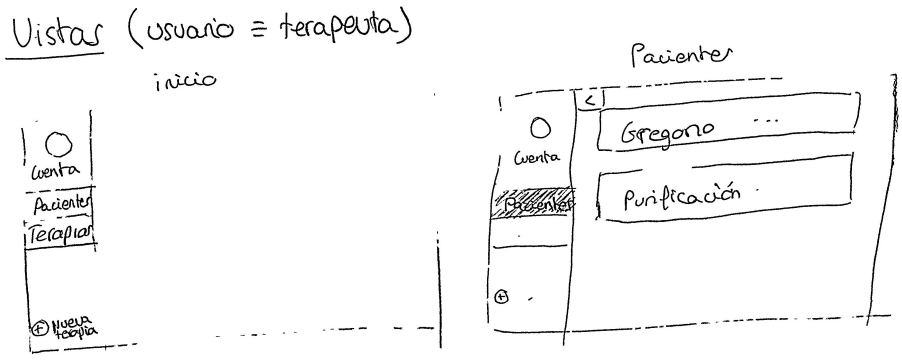

Figura 3.2: Prototipo inicial de la aplicación para el terapeuta: Consultar los pacientes registrados

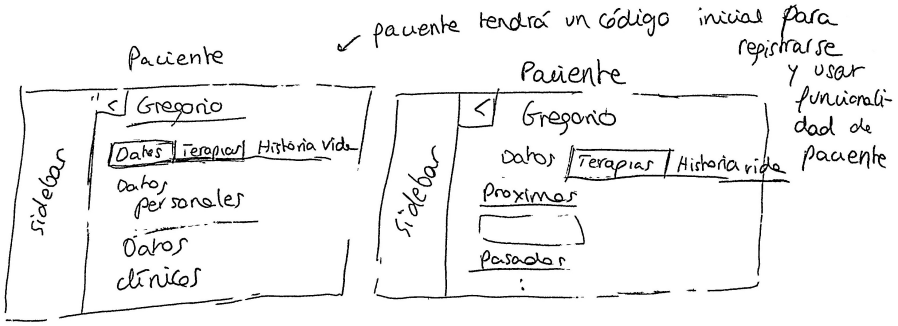

Figura 3.3: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 1

las que ocurrirán próximamente. 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.

Ver Figura 3.4 →

- 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. 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

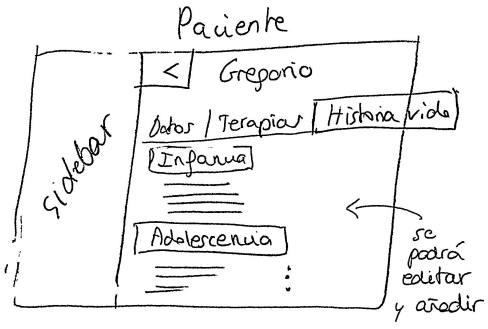

Figura 3.4: Prototipo inicial de la aplicación para el terapeuta: Consultar los datos de un paciente 2

esté mal clasificado en la etapa de la vida en que ocurrió, también se podría mover a su sitio correcto.

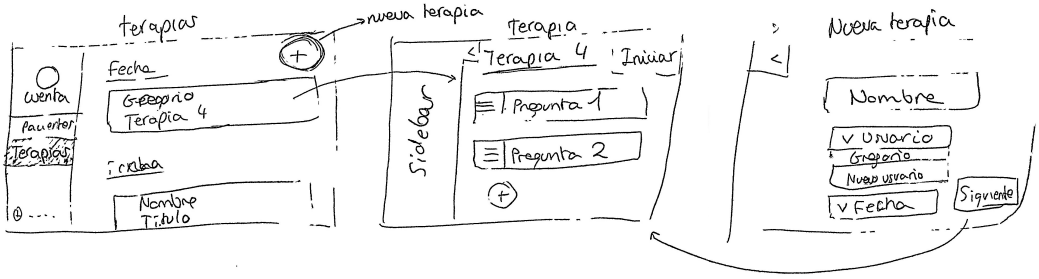

Figura 3.5: Prototipo inicial de la aplicación para el terapeuta: Crear y consultar las terapias creadas

Ver Figura 3.5 →

- Terapias: Se muestra una lista ordenada de las terapias que ha ido creando el terapeuta para los distintos pacientes.
- Consultar terapia: El terapeuta puede revisar una terapia que haya sido ya planificada. 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. 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.

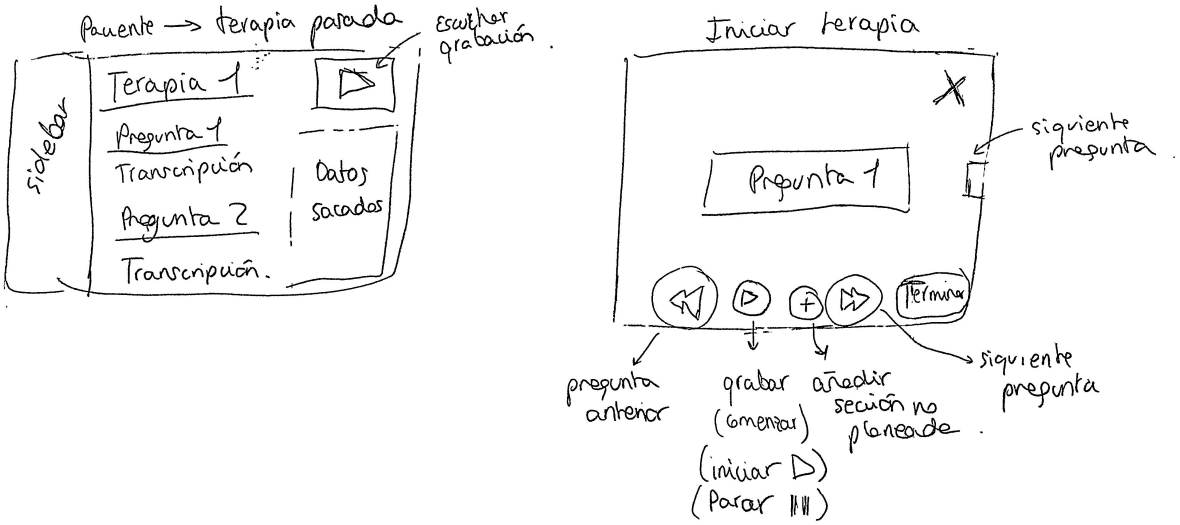

Figura 3.6: Prototipo inicial de la aplicación para el terapeuta: Consultar y reproducir una terapia concreta

Ver Figura 3.6 →

- Terapia finalizada: Se puede consultar una terapia que ya se ha realizado. 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. 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.

Por otro lado, en junio de 2022 se presetó el TFG "Recuérdame: Aplicación de apoyo para el tratamiento de personas con problemas de memoria mediante terapias basadas en reminiscencia", que precisamente se encarga del desarrollo de una aplicación web completa para facilitar la tarea del terapeuta que tiene que ayudar a usuarios con demencia a través de la terapia ocupacional basada en reminiscencia. Es decir, no tenía sentido que yo desarrollase una aplicación web para usuario y terapeuta porque ya se estaba desarrollando una cosa muy parecida en otro trabajo de fin de grado. Es por eso que la recomendación de mis tutores de centrarme en el desarrollo del chatbot fue la acertada. Aun así, sí que se ha desarrollado una pequeña aplicación web para mostrar de forma más visual la funcionalidad del chatbot. El diseño y la implementación de la misma se explicarán en las siguientes secciones.

3.5.2. Diseño final

3.5.3. Implementación

Para la aplicación del chatbot, con la que el usuario con demencia interactuaría, se ha decidido usar el *framework* de creación de aplicaciones "Flask". Se trata de un framework para la creación rápida de aplicaciones 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.

Para empezar con Flask, se utilizó la guía [5] proporcionada por la página oficial de este framework y un tutorial [6] 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. En resumen, desde el terminal se instalaron los siguientes paquetes, necesarios para el correcto funcionamiento de la aplicación:

```
> pip install Flask-WTF
> pip install email-validator
```

[5] https://flask.palletsprojects.com/en/2.2.x/quickstart/
[6] https://j2logo.com/leccion-1-la-primera-aplicacion-flask/

```
> pip install flask-login
> pip install flask-sqlalchemy
> pip install cryptography
```

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. El resto son paquetes que se han ido necesitando a lo largo del desarrollo de la aplicación para incluir ciertas funcionalidades.

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 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.

3.5.4. Funcionalidad inicio de sesión

Se añade esta nueva funcionalidad para que cada recuerdo se puedan asociar a un usuario y así dar la posibilidad de guardar incontables historias de vida. 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, me he guiado por el tutorial de Flask [7] de la página "J2logo" y hay algunas líneas de código que he cogido directamente. En este caso, esta 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.

Los formularios que se necesiten para una aplicación de Flask es conveniente tenerlos separados en un archivo Pyhton aparte, en este caso en "forms.py". En este archivo se definirán las clases SignUpForm y LoginForm que heredan de la clase FlaskForm. Dentro de estas clases se definirán los atributos, variables que se corresponden con los campos a validar del formulario. La validación de los campos se hará automáticamente teniendo en cuenta el tipo de dato que se defina para cada atributo y también los parámetros opcionales que se definan para ese atributo. Por ejemplo, para el campo del email que se usa en ambos formularios se define el siguiente atributo:

```
email = StringField('Email', validators=[DataRequired(), Email()])
```

Este atributo *email* se ha definido como una cadena de caracteres y como parámetros opcionales tiene "DataRequired(), Email()". Es decir, cuando se haga la validación del formulario, la aplicación comprobará que el campo no está vacío y que en efecto, se trata de un correo electrónico basándose en su formato.

Aparte de haber definido el formulario, se necesita una url asociada a cada página, tanto la de inicio de sesión como la de registro. Como había comentado antes, las URLs que formen parte de la aplicación deben estar definidas en el fichero "run.py". Serán respectivamente las URLs "/login" y "/signup". Hay dos funciones, una para gestionar el inicio de sesión y otra para el registro. Ambas funciones usan los métodos *GET* y *POST* porque necesitan enviar datos a la página web y recibirlos también. En estas funciones, además, será donde se invoque a las clases que se crearon anteriormente de los formularios, creando una instancia de las mismas.

——- Aquí se ha dejado de contar con tanto detalle puesto que no se si se debería hablar tan en profundidad, incluso sobre el código

El inicio de sesión de usuario consiste en introducir los campos email y contraseña que se hayan usado en el registro. Si coinciden y la contraseña es correcta se se dará acceso al resto de funcionalidades y se redirigirá al usuario de nuevo a la página principal pero ya identificado.En el caso de que el email

[7] https://j2logo.com/tutorial-flask-leccion-3-formularios-wtforms/

no exista o la contraseña sea incorrecta se mostrará el siguiente error: "Error al iniciar sesión", y se pedirán los datos de nuevo.

La página de registro es parecida a la de inicio de sesión pero añadiendo un campo más, el nombre, que se utilizará para que la aplicación se dirija al usuario. 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.

3.5.5. Funcionalidad chatbot

El chatbot 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 Pyhton. Sin embargo, se ha decidido hacer la aplicación web para facilitar la interacción con el usuario. Esta funcionalidad consiste en un chat convencional en el que el usuario interactúa con la máquina que hay detrás, aunque se distingue de otras inteligencias artificiales de tipo chatbot porque es el bot de detrás quién llevará la conversación y el que pregunta o habla con el usuario y no al revés.

Lista: - Otras tecnologías para bases de datos, para el código - Fotos del funcionamiento de la aplicación - Meter funcionalidad de sacar la informacion recabada tanto para que la vea el usuario como para que la vea el terapeuta + apartado para el terapeuta y que pueda añadir preguntas específicas para cada usuario. - Contar por qué se ha usado mongo y hacer diagramas de base de datos - Flujo de la aplicación (de que pasa en cada caso) - Prototipo final de la app

Capítulo 4

Pruebas

4.1. Módulo primero de encadenamiento entre preguntas y respuestas

Veamos algunos ejemplos de cómo funciona el 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 la siguiente código se muestra el análisis que hace el módulo de cálculo de la mejor siguiente pregunta:

```
¿Hay algún momento que te gustaría volver a vivir?
0
¿Cuál es tu sexo?
[[momento, gustaría, volver, vivir]]
{'familia', 'padre', 'hermano'}
{'volver', 'vivir', 'momento', 'gustar'}
-----------------------------------------------
¿Dónde viviste cuando eras pequeño?
0
¿Cuál es tu sexo?
[[viviste, pequeño]]
{'familia', 'padre', 'hermano'}
{'pequeño', 'vivistar'}
-----------------------------------------------
¿Has vivido en algún lugar diferente cuando eras pequeño?
0
```

```
¿Cuál es tu sexo?
[[has, vivido, lugar, pequeño]]
{'familia', 'padre', 'hermano'}
{'lugar', 'pequeño', 'vivir'}
-----------------------------------------------
¿Cómo se llaman tus padres?
0
¿Cuál es tu sexo?
[[llaman, padres]]
{'familia', 'padre', 'hermano'}
{'padre', 'llamar'}
-----------------------------------------------
¿Si tienes hermanos, Cómo se llaman?
1
¿Cómo se llaman tus padres?
[[tienes, hermanos, llaman]]
{'familia', 'padre', 'hermano'}
{'llamar', 'tener', 'hermano'}
-----------------------------------------------
¿Cómo era la casa dónde viviste de pequeño?
1
¿Cómo se llaman tus padres?
[[casa, viviste, pequeño]]
{'familia', 'padre', 'hermano'}
{'pequeño', 'casa', 'vivistar'}
```

En el código podemos observar el análisis de varias posibles preguntas. La separación entre preguntas se marca con una fila de guiones. En cada apartado nos encontramos con la posible pregunta, después el número de lemas que han coincidido en preguntas anteriores, la pregunta que hasta el momento va ganando como candidata a ser preguntada, las palabras más relevantes de la posible pregunta entre corchetes, los lemas de la respuesta entre llaves y los lemas de la posible pregunta entre llaves. Las tres primeras preguntas no serán elegidas como candidatas a ser preguntadas porque no coincide ningún lema de la respuesta con los lemas de la posible pregunta, es decir, no mejora el número de coincidencias. Cuando llega 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 ha sido el momento más feliz de tu vida?" se sustituye por la pregunta "¿Cómo se llaman tus padres?". Es por esto que cuando analizamos el resto de posibles preguntas, no encontramos ninguna que sea mejor porque no superan el número de coincidencias en lemas. En conclusión, esta será la siguiente pregunta que se le plantee al usuario, la cual tiene relación con la que se había hecho anteriormente.

4.2. Módulo primero de clasificación de respuestas en etapas de vida y en sentimientos

Capítulo 5

Conclusiones y Trabajo Futuro

Conclusiones del trabajo y líneas de trabajo futuro.

Chapter 5

Conclusions and Future Work

Conclusions and future lines of work.

Bibliografía

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

Miguel de Cervantes Saavedra

Extracción de información personal a partir de redes sociales para la creación de un libro de vida. Versión electrónica, Disponible en `https://eprints.ucm.es/id/eprint/68328/`.

Extracción de preguntas a partir de imágenes para personas con problemas de memoria mediante técnicas de Deep Learning. Versión electrónica, Disponible en `https://eprints.ucm.es/id/eprint/66857/`.

Generación de resúmenes de video-enbibtex.exe redes neuronales. Versión electrónica, Disponible en `https://eprints.ucm.es/id/eprint/68333/`.

Herramienta de ayuda guiada para la reminiscencia. Versión electrónica, Disponible en `https://eprints.ucm.es/id/eprint/68332/`.

Sistema de asistencia para cuidados de enfermos del Alzheimer. Versión electrónica, Disponible en `https://eprints.ucm.es/id/eprint/67069/`.