

GRADO EN INGENIERÍA INFORMÁTICA DE
GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

DISEÑO DE UN PACIENTE VIRTUAL CON FURHAT Y CHATGPT



Estudiante: Gómez, Nates, Alejandro

Director/Directora: Gojenola, Gallettebeitia, Koldobika

Codirector/Codirectora: Atutxa, Salazar, Aitziber

Curso: <2023-2024>

Fecha: <Bilbao, 25, 06, 2024>

Resumen

El uso de la inteligencia artificial (IA) para mejorar o no requerir el trabajo humano está proliferando estos últimos años. A la hora de realizar exámenes de Medicina, en los que el médico debe diagnosticar la enfermedad de un paciente, se requiere la actuación de una persona como paciente para realizarse. Ahora gracias a la IA y Furhat (robot social capaz de mantener conversaciones), ha surgido el proyecto de investigación Diseño de Paciente Virtual con Furhat y ChatGPT. Este documento tratará de explicar el planteamiento, ejecución y resultados del mismo.

El objetivo de la investigación es el diseño de un sistema de simulación de un paciente virtual que obtenga las mejores respuestas posibles utilizando IA generativa (IA capaz de crear ideas y conversaciones nuevas). Este desarrollo se utilizaría en la facultad de Medicina de la UPV/EHU en la realización de exámenes. Los alumnos actuando como personal médico hacen preguntas al paciente virtual y así tratan de diagnosticar la enfermedad que padece.

En este proyecto, se utilizará una base de datos creada en otros proyectos [7] y [21]. Esta misma está conformada por conversaciones entre médicos y pacientes, utilizada en el siguiente estudio[11]. Además de estas conversaciones tenemos los historiales médicos que se generaron a partir de las conversaciones con IA Generativa.

En cuanto a las tareas a realizar en este proyecto se encuentran: obtener un modelo que extraiga correctamente la respuesta a una pregunta sobre un historial médico, obtener un modelo que responda a las preguntas de un médico tal y como un paciente real lo haría, y el desarrollo e implementación de un programa capaz de unir Furhat y ChatGPT.

El resultado del proyecto es la obtención de un paciente virtual capaz de responder a un estudiante de medicina, como lo pudiera hacer un paciente real, a partir de su historial médico.

Palabras clave: paciente virtual, historial médico, Inteligencia Artificial, IA generativa, Furhat, ChatGPT, prompt, modelos LLM.

Índice general

1. Introducción	9
1.1. Contexto	9
1.2. Origen del proyecto	10
1.3. Objetivos	10
1.3.1. Hipótesis del estudio	10
1.3.2. Motivación del proyecto	10
1.3.3. Objetivo principal	10
1.3.4. Objetivos secundarios	10
1.4. Beneficios que aporta el trabajo	11
1.4.1. Impacto esperado y valor añadido	11
1.4.2. Posibilidad de generalización de los resultados	11
2. Marco teórico y Estado del arte	13
2.1. Terminología básica	13
2.1.1. Inteligencia Artificial (IA)	13
2.1.1.1. Aprendizaje supervisado y no supervisado	13
2.1.1.2. IA Generativa	13
2.1.2. LLM y Transformers	14
2.1.2.1. Entrenamiento de LLM	15
2.1.3. GPT	16
2.1.3.1. ¿Qué es un prompt?	16
2.1.3.2. Prompt engineering	16
2.1.4. ¿Qué es una API?	17
2.1.4.1. Funcionamiento API	17
2.2. Antecedentes de IA en pacientes virtuales en la enseñanza médica	17
3. Planificación y análisis del proyecto	19
3.1. Arquitectura	19
3.2. Requisitos no funcionales	21
3.2.1. ChatGPT	21
3.2.2. Furhat	22
3.2.3. Kotlin	23
3.3. Herramientas utilizadas	23
3.3.1. IntelliJ IDEA	23
3.3.2. Furhat SDK	23
3.3.3. Visual Studio Code	23
3.3.4. Librería Python (matplotlib)	23
3.3.5. Canva	23
3.3.6. Excel	23
3.3.7. GanttProject	24
3.3.8. GIMP	24
3.3.9. Microsoft Outlook y Microsoft Teams	24
3.3.10. Overleaf y LaTeX	24
3.4. Alcance del proyecto	24
3.4.1. Planificación y gestión	27
3.4.2. Análisis	29
3.4.3. Diseño	32

3.4.4. Implementación y Pruebas	34
3.4.5. Documentación	37
3.5. Alcance de la tarea secundaria	38
3.5.1. ¿Qué trabajo extra requerirá esta tarea secundaria?	38
3.6. Cronograma / Diagrama de Gantt	38
3.7. Evaluación económica	41
3.7.1. Costes por mano de obra	41
3.7.2. Costes en hardware	42
3.7.3. Costes en software	42
3.7.4. Costes indirectos	42
3.7.5. Costes totales	42
3.8. Análisis de riesgos	43
4. Descripción, diseño y desarrollo de la solución	46
4.1. Descripción datos de entrada	46
4.2. Diseño y desarrollo de la solución	48
4.2.1. Diseño del prompt extractivo	51
4.2.2. Diseño del prompt conversacional	53
4.2.3. Diseño del software de prueba de prompts	54
4.2.4. Diseño del software en Furhat	56
5. Implementación y Pruebas	59
5.1. Pruebas del prompt	59
5.2. Pruebas en Furhat	69
5.3. Realizar demo final	70
5.4. Prueba con robot real	70
5.5. Diseño y desarrollo tarea secundaria	70
5.5.1. Pruebas tarea secundaria	72
6. Resultados	73
6.1. Capacidad de ChatGPT para extraer las respuestas de un texto	73
6.2. Capacidad de ChatGPT para actuar como un paciente virtual	73
6.3. Tarea secundaria	74
7. Conclusiones	75
7.1. Trabajo futuro	75
7.2. Reflexión personal	75
Bibliografía	76

Índice de figuras

1.1. Funcionamiento del sistema	11
2.1. Arquitectura de capas transformers	15
3.1. Arquitectura primera tarea	20
3.2. Arquitectura segunda tarea	21
3.3. Robot Furhat real	22
3.4. Fases del diagrama EDT	25
3.5. Diagrama EDT	26
3.6. Planificación y Gestión	27
3.7. Análisis	29
3.8. Diseño	32
3.9. Implementación y Pruebas	34
3.10. Documentación	37
3.11. Gantt del 1 de febrero al 18 de marzo	40
3.12. Gantt del 19 de marzo al 3 de mayo	40
3.13. Gantt del 4 de mayo al 17 de junio	41
4.1. Esquema breve Datos de Entrada	46
4.2. Estructura Datos de entrada	47
4.3. División de los datos	48
4.4. Fórmula BLEU	49
4.5. Tabla de los resultados de la precisión en una conversación	55
4.6. Diagrama de flujo estados Furhat	56
5.1. Precisión según la temperatura	63
5.2. Precisión según la temperatura 2	64
5.3. Instrucciones a los evaluadores para realizar la supervisión humana	68
5.4. Conversación con Furhat	69
5.5. Estructura Datos de entrada en Inglés	71

Índice de cuadros

3.1. Seguimiento y Reuniones con el grupo de trabajo	27
3.2. Objetivos y Alcance	28
3.3. Planificación tareas	28
3.4. Planificación temporal	28
3.5. Captura de Requisitos	29
3.6. Captura de Requisitos Prompt Extractivo	30
3.7. Captura de Requisitos Prompt Conversacional	30
3.8. Análisis datos de entrada	30
3.9. Investigar requisitos no funcionales	30
3.10. Seleccionar herramientas	31
3.11. Análisis métodos de evaluación	31
3.12. Diseño del prompt extractivo	32
3.13. Diseño del prompt conversacional	33
3.14. Diseño del software de prueba de prompts	33
3.15. Diseño del software en Furhat	33
3.16. Implementación del software final	34
3.17. Realizar pruebas prompt	35
3.18. Pruebas finales con Furhat	35
3.19. Realizar demo final	35
3.20. Prueba con robot Furhat real	35
3.21. Desarrollar conclusiones finales	36
3.22. Investigar tecnologías utilizadas	37
3.23. Realizar memoria	37
3.24. Preparar defensa	38
3.25. Planificación del tiempo (II)	39
3.26. Costes totales del proyecto	42
3.27. Riesgo 1	43
3.28. Riesgo 2	43
3.29. Riesgo 3	44
3.30. Riesgo 4	44
3.31. Riesgo 5	44
3.32. Riesgo 6	44
3.33. Riesgo 7	45
4.1. Descripción datos de entrada	48
4.2. Prompt conversacional final	54
5.1. Prompt extractivo base	60
5.2. Prueba: Modelo GPT	60
5.3. Prueba: No se menciona en el historial médico	61
5.4. Prueba: La pregunta no requiere respuesta	61
5.5. Prueba: Brevedad de la respuesta	61
5.6. Prueba: No modificación de la extracción	62
5.7. Prueba: Zero-shot prompting	62
5.8. Prueba: Few-shot prompting	62
5.9. Prueba: Historial médico o texto	63
5.10. Prueba: Conversación completa en prompt	64

5.11. Tabla resumen de las decisiones tomadas en las pruebas	65
5.12. Prompt extractivo final	66
5.13. Valoraciones del paciente virtual según los evaluadores	69
5.14. Prompt en inglés	72
5.15. Prueba: Prompt en inglés	72

Acrónimos

- **IA:** Inteligencia Artificial
- **JSON:** JavaScript Object Notation
- **API:** Application Programming Interface (interfaz de programación de aplicaciones)
- **ML:** Machine Learning (aprendizaje automático)
- **TIC:** Tecnologías de la Información y la Comunicación
- **DSL:** Domain-Specific-Language (Lenguaje Específico de Dominio)
- **SDK:** Software Development Kit (Kit de Desarrollo de Software)
- **IDE:** Integrated Development Environment (Entorno de Desarrollo Integrado)
- **QA:** Question Answering (Búsqueda de Respuestas)
- **BLEU:** BiLingual Evaluation Understudy (Evaluación bilingüe suplente)
- **NLU:** Natural Language Understanding (Comprensión del Lenguaje Natural)
- **NLP:** Natural Language Processing (Procesamiento del Lenguaje Natural)
- **LLM:** Large Language Models (Modelos de Lenguajes de Gran Tamaño)
- **GPT:** Generative Pretrained Transformer (Transformer Generativo Preentrenado)
- **EDT:** Estructura de Desarrollo Temporal

1. Introducción

1.1. Contexto

Desde la irrupción en el año 2022 de ChatGPT, la palabra IA, junto a la previsión de que va a cambiar el mundo no ha hecho más que crecer. Sin embargo, ¿qué es esta tecnología y por qué es tan importante en la actualidad? [33]. La Inteligencia Artificial (IA) está formada por una serie de algoritmos previamente entrenados a partir de los cuales las máquinas son capaces de tomar decisiones para casos específicos. Todo comenzó con Alan Turing, el cual sentó las bases para que hayamos podido llegar a lo que tenemos ahora [24]. Las primeras demostraciones de que podía ser una opción real fue con el ajedrez. Se empezaron a desarrollar softwares capaces de aprender el juego de forma autónoma. El momento álgido fue cuando en 1997, la máquina Deep Blue logró ganar al campeón mundial de ajedrez Gari Kaspárov. Las grandes inversiones por parte de empresas tecnológicas de renombre no comenzaron hasta bien entrados en el nuevo milenio. Hubo un gran desarrollo en distintas áreas, como pueden ser la medicina, la robótica y el procesamiento del lenguaje natural (NLP, Natural Language Processing) [32].

Esta último área es de gran importancia para el proyecto que nos atañe. NLP es una rama de la IA que dota a los ordenadores de la capacidad de comprender el lenguaje humano y generarla. Además de comprender lo escrito, es capaz de entender la voz hablada, interpretarla correctamente y entender el sentimiento que la acompaña. Una sencilla explicación de lo que se realiza en esta técnica es dividir el lenguaje en piezas más cortas que tengan significado, e intentar entender las relaciones entre las piezas. Los usos que se le dan varían entre el resumen de documentos, la conversión de texto a voz y voz a texto, la traducción y la extracción del contexto y sentimiento. Con los usos mencionados se pueden desarrollar chatbots.

Los chatbots [18] son aplicaciones software capaces de mantener una conversación con una persona, comprendiendo las preguntas de la persona y automatizando las respuestas. Gracias al NLP, nos podemos comunicar con ellos a través de texto o voz. Además de esta función, también ha permitido pasar de tener una lista de respuestas fija redactada por los programadores, a que sea capaz de comprender lo que se le transmite, y dar una respuesta acorde a ello. Estas herramientas se utilizan de manera muy diferente, desde ayudar con tareas cotidianas en el móvil con Siri, hasta ayudar con la contratación de un servicio sin tener que necesitar la ayuda de una persona de Atención Al Cliente. Pero sin duda la llegada de ChatGPT en noviembre de 2022 ha logrado que se generalizara aún más la IA en el mundo tecnológico.

ChatGPT en sus primeros 2 meses, logró la escandalosa cifra de 100 millones de usuarios mensuales. Este *boom* atrajo inversiones millonarias a este mundo, y los avances tecnológicos no paran de suceder. Para entender este suceso, hay que saber qué es GPT. GPT o Generative Pre-trained Transformer [2] son una familia de modelos de redes neuronales que utilizan una arquitectura llamada transformadores que dotan a la IA de capacidad de crear texto y contenido, como si de un humano se tratara [34].

La combinación de los avances de la IA y los chatbots, con los de la robótica han propiciado la posibilidad de realizar mi proyecto, un robot capaz de realizar el papel de un paciente. Obtener tal funcionalidad de un robot supondría un paso enorme para la educación en torno a la medicina.

1.2. Origen del proyecto

El proyecto ha sido propuesto por Aitziber Atutxa y Koldobika Gojenola, docentes del departamento de Lenguajes y Sistemas Informáticos y miembros del grupo de investigación IXA en la Escuela de Ingeniería de Bilbao de la Universidad del País Vasco (UPV/EHU). La facultad de Medicina de la UPV/EHU realizó una inversión en la cual se compró un robot Furhat [12]. La razón de esta inversión es que existe un problema para realizar un examen del grado. Uno de los exámenes que se realizan trata de diagnosticar la enfermedad de un paciente. Hasta ahora, se contratan actores que durante horas deben simular tener la misma enfermedad para cada estudiante. Esto supone un gasto para la universidad, ya que los actores deben cobrar por realizar su trabajo durante horas. Por si no fuera poco, el trabajo que deben realizar los actores es repetitivo y nada deseable para ellos. Por tanto, en búsqueda de una solución para esta problemática se pensó en apostar por las nuevas tecnologías. Con la compra del robot Furhat se espera que pueda realizar el papel de un paciente virtual, capaz de responder a las preguntas del médico siguiendo la información de su historial médico.

1.3. Objetivos

Se procede a describir las razones y motivaciones por las que se ha realizado el proyecto, junto a los objetivos establecidos que se esperan obtener tras la finalización del mismo.

1.3.1. Hipótesis del estudio

Los avances en la IA y la robótica han logrado que sea posible el desarrollo de un robot capaz de mantener una conversación con un humano, adoptando el historial médico como propio. Se cree que la IA, en especial la generativa, ha evolucionado lo suficiente como para hacer todo el trabajo ella: entender el contexto del paciente, comprender la pregunta que se le ha hecho, buscar la respuesta a la pregunta dentro del historial médico y crear una conversación realista.

1.3.2. Motivación del proyecto

La automatización que nos permite la IA puede eliminar los trabajos tediosos y ahorrar costes. Además de automatizarlo, también facilitaría la tarea a los que la supervisan. Sería más cómodo y rápido representar casos nuevos, con un simple historial médico. No se necesitaría el periodo de estudio del actor y posterior práctica del mismo antes de poder realizar los exámenes. Tampoco sería necesaria la búsqueda de actores para realizar el trabajo, ni la inversión económica y en tiempo del proceso de contratación y preparación del examen.

1.3.3. Objetivo principal

Desarrollar e implementar un sistema que simule un paciente virtual utilizando un robot Furhat, para que los estudiantes de la facultad de Medicina de la UPV/EHU puedan examinarse de sus habilidades de diagnóstico de enfermedades. Se utilizará ChatGPT para responder a las preguntas que haga el médico a partir del historial médico del paciente. Se tratará de obtener las mejores respuestas posibles de ChatGPT para generar una conversación realista. Se deberá determinar si es viable la sustitución de un agente conversacional por el actor real, véase la figura 1.1.

1.3.4. Objetivos secundarios

- Realizar un análisis de la capacidad de ChatGPT para extraer de un texto las respuestas a las preguntas que se le hagan.
- Realizar un análisis de si ChatGPT puede generar conversaciones realistas siguiendo las instrucciones necesarias para actuar como paciente.
- Tarea secundaria en la que comprobaremos si cambiando el idioma del contexto y conversación al inglés los resultados mejoran.
- Tratar de implementar el sistema en un robot real y realizar una prueba con él.

1.4. Beneficios que aporta el trabajo

Los estudiantes podrán practicar o examinarse con una herramienta que será más versátil y más fácil de utilizar que la forma tradicional. Un solo programa que pueda abarcar todo tipo de enfermedades conseguirá reducir el tiempo y dinero que se debe invertir ahora. Proseguiremos explicando el impacto esperado de nuestro proyecto en las instituciones educativas.

1.4.1. Impacto esperado y valor añadido

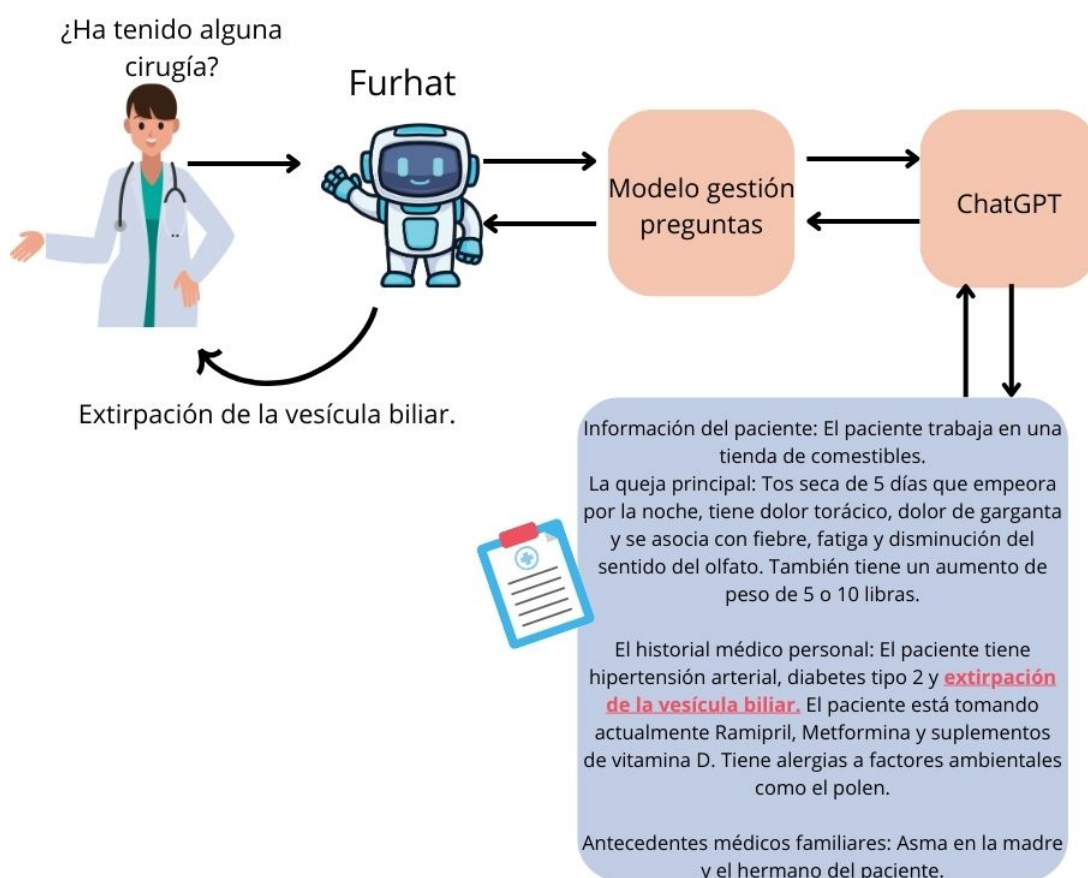
La IA está desarrollándose en multitud de ámbitos y la educación y medicina por supuesto están entre ellos. Es importante apostar por los avances tecnológicos en pro de mejorar la eficiencia y efectividad de los procesos que se realizan.

Lo que se espera que consiga nuestra herramienta se puede resumir en los siguientes puntos:

- Ayudar a la facultad de Medicina de la UPV/EHU en el aprovechamiento de su inversión con el robot Furhat.
- Ayudar a los estudiantes a practicar su comunicación y habilidad para diagnosticar enfermedades sin la necesidad de terceras personas.
- Uso de la herramienta en los exámenes, en sustitución de los actores actuales.

1.4.2. Posibilidad de generalización de los resultados

La gran ventaja de utilizar una IA generativa, es que no se encasilla en un único caso. Como se le entrega a la herramienta el historial médico, ella misma es capaz de amoldarse a ese caso, y actuar como el paciente, por lo que se podría utilizar en multitud de casos distintos.



1.1. Figura: Funcionamiento del sistema

En la figura 1.1 se puede observar el funcionamiento de nuestro proyecto. El médico le hace una pregunta al robot Furhat para saber cómo se encuentra. El robot Furhat utiliza su Modelo de Gestión de Preguntas para saber lo que debe hacer con la pregunta que se le ha hecho. El modelo llama a ChatGPT para obtener la respuesta correcta. ChatGPT busca la respuesta a la pregunta en el historial médico. Una vez la encuentra, construye la respuesta y la vuelve a enviar al modelo y él a su vez a Furhat. Finalmente, el robot da una respuesta al médico.

2. Marco teórico y Estado del arte

La base del proyecto es la Inteligencia Artificial, por lo que es conveniente establecer un contexto teórico de su funcionamiento. Además de su funcionamiento, conocer la historia de las tecnologías que se utilizarán es importante para ver lo que puede conseguir en el presente, y lo que ha ido evolucionando a lo largo de los años. Explicaremos ampliamente los términos más complejos que he mencionado o se mencionarán durante el proyecto. Finalizaremos este apartado con los antecedentes y evolución de los mismos.

2.1. Terminología básica

Comenzaremos esta sección con los principales términos necesarios para comprender el funcionamiento de las tecnologías utilizadas.

2.1.1. Inteligencia Artificial (IA)

Es la disciplina que tiene el objetivo de que los ordenadores obtengan las mismas capacidades que los seres humanos para resolver problemas. La intención es que sean capaces de percibir el entorno y respondan ante él de manera autónoma. Para obtener estos resultados deben realizar una etapa de aprendizaje, en la que utilizan diferentes algoritmos y grandes cantidades de datos para cada vez tomar mejores decisiones. Existen dos métodos de aprendizaje automáticos, el supervisado y no supervisado [17].

2.1.1.1. Aprendizaje supervisado y no supervisado

La gran diferencia entre estos dos métodos está en que en el aprendizaje supervisado los datos de entrada y la salida (resultado) se encuentran etiquetados, mientras que en el no supervisado no. Durante la etapa de aprendizaje, es necesaria la intervención humana para etiquetar correctamente los datos. El objetivo es que el modelo aprenda patrones entre los datos de entrada y de salida, para así predecir correctamente utilizando datos nunca antes vistos. Las tareas para las que más se utiliza este método son: clasificación de imágenes o documentos y pronosticar tendencias o resultados futuros.

Por otro lado, con el no supervisado se entrenan modelos con datos sin procesar ni etiquetar. Al no tener que ser procesados, no requiere de tanta interacción humana. La idea es establecer diferentes *clusters* (grupos de datos) según los patrones que se encuentren entre los datos. Las aplicaciones más comunes de este modelo son: tareas de reconocimiento de objetos, recomendar productos afines a los clientes en sus procesos de compra, comprender los hábitos de compra de los clientes [37].

2.1.1.2. IA Generativa

Es una subdisciplina de la IA que se centra en crear contenido original. Utiliza los datos que han recibido en el proceso de aprendizaje, pero no responden idénticamente con ellos, sino que dan respuestas similares [3]. Los modelos generativos, como se puede esperar, utilizan métodos generativos, los cuales tratan de aprender cualquier distribución de datos mediante aprendizaje no supervisado. Por tanto, se basan en la estadística y la probabilidad para dar respuesta a lo que se le pide.

Este tipo de modelos son muy útiles para la creación de datos *sintéticos*. Son aquellos datos que se asemejan a los del mundo real, pero que se han generado artificialmente. Estos tienen una gran importancia dentro de la investigación, porque son muy útiles para las siguientes tareas:

- Aumento de los datos: En casos en los que es muy complicado obtener una mayor cantidad de datos, se utilizan los modelos generativos [30] para crear datos artificiales que siguen la misma distribución o tienen las mismas características que los originales.
- Anonimización: Intercambian la información sensible por información artificial que es estadísticamente equivalente.
- Protección de los datos: Se pueden crear conjuntos de datos que no tengan información sensible o de identificación personal.

A los modelos que utilizan IA generativa se les denomina FM o *Foundation Models* (Modelos Fundacionales) [19]. Como ya hemos explicado se entrenan con algoritmos que utilizan redes neuronales y aprendizaje no supervisado. Se basan en relaciones y patrones aprendidos para predecir el elemento siguiente en una secuencia. Dentro de estos FM [27] se encuentra LLM o *Large Language Models* (Modelos de Lenguaje de Gran Tamaño).

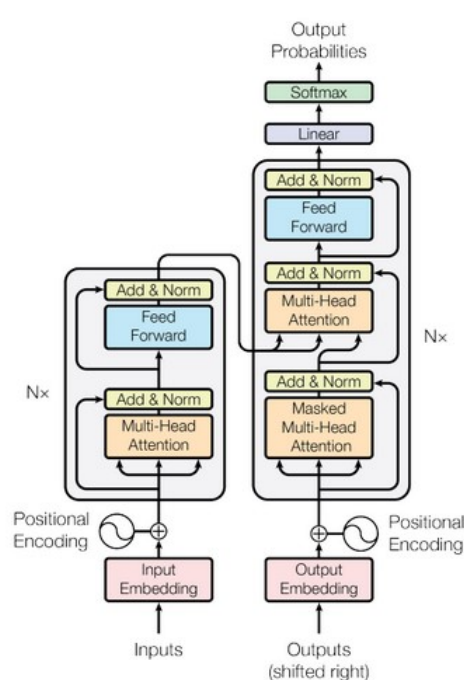
2.1.2. LLM y Transformers

La particularidad de los LLM [4] es que utilizan el tipo de redes neuronales llamado transformers. Estos han cambiado el paradigma de la IA Generativa en cuanto a la creación de textos se refiere. Los transformers permiten aprender el contexto de la frase, algo vital para poder comprender el lenguaje humano que antes no se conseguía. En los anteriores modelos utilizados para LLM, se representaba cada palabra con una única tabla numérica [8]. Esto dificultaba entender la relación entre las palabras y por consiguiente el contexto de la frase. Con el cambio de vectores multidimensionales por las tablas numéricas, se consiguió que palabras con significados contextuales parecidos estuvieran cercas unas de otras dentro del espacio vectorial.

Con la llegada de los *transformers* [26], cuyo mayor cambio es que dentro de las redes neuronales que utilizan tienen un codificador y decodificador capaces de autoatención, se extraen mejor los significados de un texto y comprenden de mejor manera la relación entre palabras. Estos modelos además de aprender sin supervisión, también realizan un autoaprendizaje. De esta manera, aprenden el idioma, la gramática y los conocimientos básicos de la comunicación.

Otra gran diferencia respecto a los modelos anteriores utilizados en LLM, es que se procesan secuencias enormes de datos en paralelo. Antes se hacía de manera secuencial, por lo que se ha reducido enormemente el tiempo necesario de la etapa de aprendizaje y a su vez se ha multiplicado la cantidad de datos que se procesan.

La arquitectura de los transformers se basa en capas [35], vease la figura 2.1:



2.1. Figura: Arquitectura de capas transformers

Tiene capas tanto de codificadores, como de decodificadores. Cada capa tiene muchos nodos, los cuales tienen conexión con las capas subsiguientes. Estos últimos se conocen como los parámetros del modelo, por lo que estos modelos de redes neuronales con transformers llegan a tener miles de millones de parámetros. Para comprender mejor la arquitectura se va a explicar cómo procede cuando se encuentra una frase.

Si tenemos la siguiente frase: “El niño juega con el perro.”

Primero debemos tener en cuenta que “juega” está relacionado con “niño” y con “perro”. La arquitectura lo analiza de la siguiente manera:

- **Tokenización:** La frase se divide en *tokens* o palabras de la siguiente manera: “El”, “niño”, “juega”, “con”, “el”, “perro”.
- **Embeddings:** Cada token se convierte en una representación matemática del significado de la palabra, llamado vector de *embedding*. A este vector se le asocia un peso, según la importancia dentro de la oración. En este momento se acaban las tareas del módulo codificador.
- **Mecanismo de atención:** En este último paso el vector de embedding llega al decodificador. En vez de procesar la frase de izquierda a derecha, puede enfocar la atención en diferentes tokens cuando genera la palabra de salida. En nuestro ejemplo, cuando genera el token “juega”, presta atención a los tokens “niño” y “perro” a la vez. De esta forma, se comprende mejor la intención y el contexto de la frase.

De esta manera, si imaginamos la conexión entre las palabras de la frase, se podría decir que hay una conexión fuerte establecida entre “juega” y los dos sustantivos “niño” y “perro”.

2.1.2.1. Entrenamiento de LLM

Como ya hemos comentado anteriormente, estos modelos se entrenan automáticamente, pero ¿cuál es el proceso que siguen? [39]

Utilizando un corpus de datos enorme de una calidad alta, el modelo va ajustando sus parámetros de forma iterativa para predecir correctamente el siguiente *token*. Existen diferentes modelos de aprendizaje con los que los científicos de datos tratan de maximizar los resultados:

- **Zero-shot training:** Pueden responder a las tareas que se le indiquen sin previo entrenamiento concreto para ello, sin embargo su precisión varía mucho.

- **Few-shot training:** Se le presentan varios ejemplos específicos de entrenamiento para mejorar su capacidad de respuesta en esa área en concreto.
- **Ajuste fino:** Se añade una extensión de aprendizaje, que utiliza la técnica anterior. De modo que los parámetros se ajusten correctamente para la aplicación concreta en la que se quiere mejorar el rendimiento.

2.1.3. GPT

Tras la explicación de los modelos LLM y los Transformers, es el momento de explicar los modelos GPT, ya que nuestro trabajo depende de la utilización de ChatGPT, el cual los utiliza por supuesto.

GPT o Generative Pretrained Transformers, son una familia de modelos de redes neuronales que utilizan los ya explicados transformers. Nos permiten una mayor calidad y precisión a la hora de crear imágenes, música o texto, al igual que lo haría un ser humano. Concretamente, los GPT son modelos de predicción lingüística, que en base a las consultas que se le hagan en lenguaje natural o *prompts* tratan de predecir la mejor respuesta posible.

Como ya hemos explicado en la arquitectura de capas de los transformers, no solo van palabra a palabra intentando conocer cuál es la siguiente más probable, sino que puede prestar atención a diferentes palabras y generar respuestas más largas y congruentes según lo que le ha consultado.

2.1.3.1. ¿Qué es un prompt?

En la sección anterior hemos nombrado la palabra *prompt*, y durante el proyecto se nombrará aún más, ya que la generación de ellos va a suponer una parte importante del esfuerzo en cuanto al diseño y desarrollo de la solución.

Un *prompt* es una serie de instrucciones que se le dan a un modelo de IA Generativa para obtener una respuesta según los parámetros que se le especifiquen. Sirve para establecer un contexto y definir la tarea específica que se espera de la IA. Estos prompts se escriben en lenguaje natural, como si fuera una conversación normal entre el usuario y la IA. Esto nos permite utilizar estas herramientas sin la necesidad de conocer su funcionamiento.

A continuación se muestran una serie de ejemplos de prompts para la generación de texto:

- *Escribe un resumen sobre la película “El señor de los anillos”*
- *¿Cómo se escribe por pantalla en Python?*
- *Dime la receta para hacer una tortilla de patata.*

Podemos pedir casi cualquier cosa que podamos llegar a imaginar. Según lo preciso que seamos al escribir el *prompt*, obtendremos una mejor respuesta o no. Si se quiere una tarea específica como puede ser:

Escribe un ensayo de 300 palabras, como lo haría Shakespeare sobre la ética en la Inteligencia Artificial. Cuanto más ambiguo se lo pidamos, peor lo hará y aquí entran las técnicas de **prompt engineering** o ingeniería de prompt [22].

2.1.3.2. Prompt engineering

Es una nueva disciplina creada para mejorar la capacidad de los modelos LLM a la hora de responder preguntas o hacer razonamientos aritméticos. Se diseñan técnicas de prompt efectivas para interactuar mejor con los LLM y comprenderlos mejor. Otros objetivos por los que se utiliza esta disciplina son la mejora de la seguridad de los modelos o para intentar diseñar nuevas capacidades. Hay muchas técnicas dentro del *prompting* y cada una nos ayuda a conseguir un objetivo distinto [29]:

- Formatear prompt como sistema de QA (pregunta respuesta)
(Q)-¿De qué color es el cielo?
(A)-
- **Zero-shot**(Sin entrenamiento previo), no se le presenta ningún ejemplo de cómo se responde:
Clasifica el sentimiento en positivo o negativo.
Texto: “Estoy triste porque he perdido la pelota”
Sentimiento:

- **Few-shot:** En tareas complejas, el uso de varias muestras para demostrar al modelo cómo se debe responder ayuda a orientarlo y obtener una mejor respuesta:

Clasifica el sentimiento en positivo o negativo.

Texto: “Estoy triste porque he perdido la pelota”

Sentimiento: Negativo

Texto: “Me he reído mucho con mis amigos en el parque.”

Sentimiento: Positivo

Texto: “Me he caído de la bici y he empezado a sangrar”

Sentimiento: Negativo

Texto: “Me han dado la nota de un examen y he suspendido la asignatura.”

Sentimiento:

Mediante la utilización de estas técnicas estableceremos el objetivo de cada prompt que se utilice. En nuestro caso diseñaremos dos prompts distintos (extractivo y conversacional) para lograr los dos primeros objetivos secundarios, véase 1.3.4.

2.1.4. ¿Qué es una API?

Para las pruebas del prompt y la comunicación entre Furhat y ChatGPT es necesario el uso de su API. Vamos a explicar su funcionamiento y su utilidad en un proyecto como este.

Una API es una serie de reglas que intervienen en la comunicación entre dos aplicaciones [16]. Se encarga de procesar las transferencias de datos, de forma que sea más fácil que los desarrolladores tanto externos como internos de diferentes departamentos accedan a los datos de una empresa. De esta forma se eliminan muchas barreras y tareas innecesarias para la comunicación entre aplicaciones.

2.1.4.1. Funcionamiento API

Concretamente la de ChatGPT [23], lo que nos permite es hacer uso de sus generadores de texto para responder a las preguntas que le hagamos.

Se puede acceder a ella de diferentes maneras, utilizando librerías [20] en según qué lenguajes de programación o llamando directamente a la API mediante una petición HTTP. Se profundizará en las llamadas en esta sección 4.2.4

En los casos de Python y Kotlin se puede acceder a ella a través de la librería *openai*. Una librería en Python [6] es una colección de distintos métodos y funcionalidades que están predefinidos (probados y optimizados) y ayudan a extender la funcionalidad básica del lenguaje de programación.

2.2. Antecedentes de IA en pacientes virtuales en la enseñanza médica

El avance de la Inteligencia Artificial se ha aprovechado en multitud de campos, donde destacan la medicina y la enseñanza. En este apartado nos vamos a centrar en el desarrollo y avance de los pacientes virtuales en la enseñanza médica, no en los avances en cuanto a la telemedicina o herramientas de diagnóstico basadas en la IA. Cabe mencionar que se está trabajando arduamente en pacientes virtuales en los que probar tratamientos para así comprobar como reaccionarían los pacientes reales. A estos desarrollos se les denomina medicina *in silico*. [9] Ahora sí, se realizará una pequeña ilustración de la evolución de los pacientes virtuales con función educativa.

Los comienzos datan de la década de 1960 y 1970, en estos años se intentaban modelar sistemas biológicos y médicos, mediante la simulación por ordenador. Estos pacientes virtuales todavía no tenían IA, y sólo seguían un modelo anatómico y fisiológico [38]. En los años 80 y 90, llegó un desarrollo inmenso dentro de esta tecnología, gracias al aprendizaje automático. El objetivo era obtener una interacción humana realista, que concordara con la de un paciente. Aún estaba lejos, pero ya se obtenían avances en cuanto a la forma de responder adaptándose a la pregunta y contexto. Junto a los progresos en el aprendizaje automático, también sucedieron en el área del modelado 3D, que permitieron una apariencia más realista a nuestro paciente [15].

Con los años de estudio y mejoras, se comenzó a utilizar en la enseñanza para ayudar a los alumnos a practicar en un entorno seguro y simulado. Los últimos años han sido de gran importancia gracias al NLP, ya que ha permitido que sean capaces de comprender el lenguaje y crear conversaciones más realistas [13].

3. Planificación y análisis del proyecto

Después de haber explicado el contexto y objetivos del proyecto, además de una explicación teórica del funcionamiento de las tecnologías utilizadas, llega el turno de la planificación para realizarlo. Detallaremos las fases que ha de tener para tener un desenlace exitoso, junto a los análisis realizados para ello: la arquitectura del proyecto, el alcance del mismo, el marco temporal (diagrama de Gantt), la evaluación económica y el análisis de riesgos.

3.1. Arquitectura

Para comprender mejor la arquitectura del proyecto, primero se deben diferenciar las dos partes principales. Tal y como hemos mencionado en la sección de objetivos 1.3.4, debemos por un lado comprobar si ChatGPT es capaz de extraer correctamente la respuesta de un historial médico, dada una pregunta sobre él. Por otro lado, se debe implementar un paciente virtual que mediante Furhat y ChatGPT de las respuestas adecuadas a un personal médico. Por lo que explicaremos la arquitectura de las dos partes por separado.

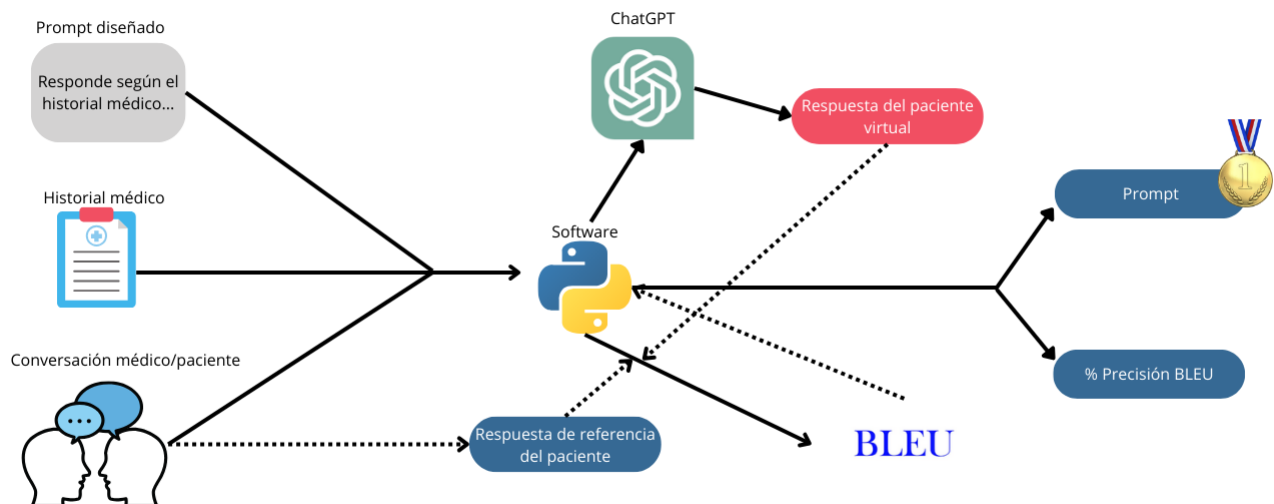
En primer lugar, tenemos la tarea de extraer la respuesta de un historial médico. Como datos de entrada, o *inputs*, tenemos los historiales médicos de los pacientes, junto a una serie de preguntas y respuestas sobre cada historial. Utilizaremos un mismo archivo que tenga todas las conversaciones cada una con el historial médico del paciente, la lista de preguntas y respuestas, y por cada respuesta algún campo extra relevante en casos especiales. La explicación detallada de este archivo JSON se encuentra en la figura 4.1. Además de este JSON, se necesita también el prompt elegido para que ChatGPT de la respuesta como deseamos.

Para conseguir el objetivo de la primera tarea, utilizaremos un programa realizado con el lenguaje de programación **Python**, el cual será el *software de prueba de prompts*, vease la sección 3.14 para una explicación profunda del funcionamiento de este software.

El propósito de este programa es recrear las respuestas de los pacientes que se tienen dentro de los datos de entrada mediante llamadas a la API de ChatGPT. Se le da la instrucción a ChatGPT de que responda a la pregunta que le ha hecho el médico siguiendo el historial médico. Teniendo la “respuesta real” (o información que debería ChatGPT utilizar del historial médico para responder) y la respuesta que ha dado ChatGPT, se utiliza *BLEU* (sección 4.2) para evaluar lo correcta que ha sido la respuesta de la IA. Se aplicará esta forma de evaluación a su vez para conversaciones enteras, de manera que obtengamos la precisión media del prompt elegido. Durante el proceso de la tarea se irán modificando los prompts para comprobar cuál es el que mejor precisión tiene.

La salida o output sería el prompt, que obtiene la mejor precisión *BLEU* y su precisión *BLEU* promedio.

En esta figura 3.1 podemos ver cómo funciona a grandes rasgos:



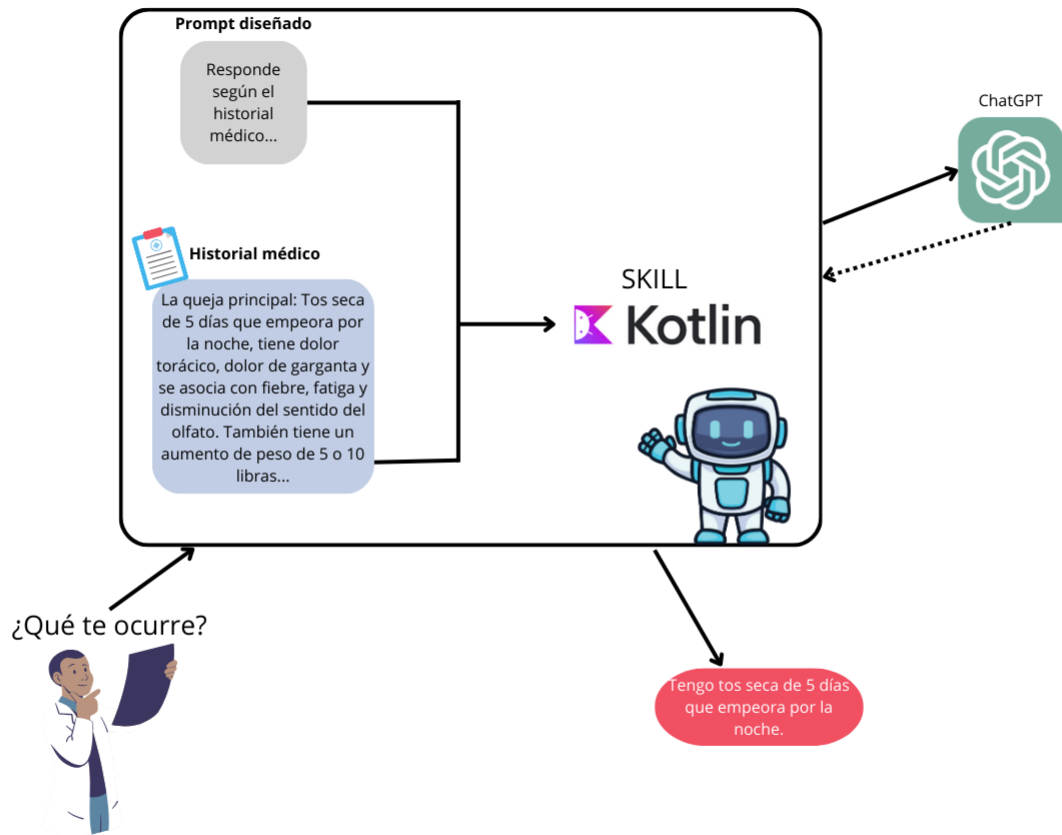
3.1. Figura: Arquitectura primera tarea

En segundo lugar, está la tarea de implementar un paciente virtual utilizando Furhat y ChatGPT. Esta tarea tiene como datos de entrada un historial médico, el prompt elegido y la pregunta que se le hace por voz al robot.

Para lograr el objetivo propuesto, se deberá crear un programa o Skill, como se le llama desde Furhat, que estará desarrollado en el lenguaje **Kotlin**. Este programa dicta cómo se debe comportar Furhat (configuración de idioma, tiempo de espera de la pregunta...) y establece cómo debe responder a las preguntas que se le hagan (explicación detallada en la sección 3.15). Para responder a las preguntas, la Skill realizará llamadas a la API de ChatGPT con esta instrucción: Utilizando el **historial médico del paciente** responde a la **nueva pregunta** siguiendo las instrucciones descritas en el **prompt diseñado**. Finalmente, el robot Furhat pronunciará la respuesta que se obtenga de la llamada a ChatGPT.

Por último, la salida o *output* de esta tarea es la comunicación realista y dinámica entre una persona, que actúe como personal médico, y el robot Furhat, que actuará como paciente.

En esta figura 3.2 podemos ver cómo funciona a grandes rasgos:



3.2. Figura: Arquitectura segunda tarea

3.2. Requisitos no funcionales

Cuando se inicia un proyecto, muchas veces se establecen requisitos no funcionales que determinan cómo se debe llevar a cabo. En nuestro caso, estos requisitos son tanto el uso de Furhat, como el de ChatGPT.

3.2.1. ChatGPT

Esta famosa tecnología es un chatbot (agente conversacional) que utiliza Inteligencia Artificial Generativa desarrollada por la empresa OpenAI. A diferencia de otros chatbots que están desarrollados para una función en específica, por ejemplo contratar un servicio dentro de una empresa, este sistema va más allá y entiende cualquier tarea que se le pide y es capaz de dar una respuesta coherente.

Se llama así, porque utiliza un modelo de lenguaje basado en Transformers, concretamente una versión de GPT (Generative Pre-trained Transformer). Se entrenó con millones de datos de Internet, para aprender a construir frases con sentido y comprender bien el lenguaje. Después de esa etapa, se le instruyó en cuestiones más específicas supervisado por humanos.

A finales de 2022 supuso un boom para la IA, ya que millones de personas que no conocían este tipo de tecnologías y la descubrieron con este modelo, empezaron a utilizarla y conocer las ventajas de la IA. Esta herramienta se puede utilizar para muchísimas tareas muy tediosas hasta su llegada, como pueden

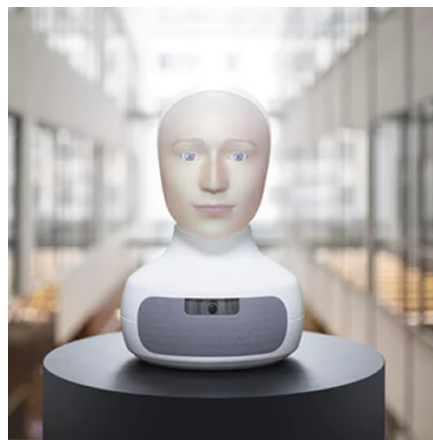
ser: escribir textos extensos, resumirlos, obtener ideas para tus proyectos, programar problemas sencillos, obtener información más rápidamente que en un buscador...

Gracias a su llegada, la inversión en estas tecnologías se ha magnificado y ha proliferado la cantidad de programas que utilizan IA.

3.2.2. Furhat

Se definen a sí mismos como el robot social más avanzado del mundo. Se trata de un robot capaz de emular las emociones de un humano y mantener el contacto visual, todo ello mientras habla y escucha durante la conversación. Es un robot enfocado al *question answering* [31], por lo que es la única tarea que puede realizar.

Se diferencian de otros robots, en que se centra en brindar la mejor interacción comunicativa. Tiene una serie de cámaras para enfocarse en la persona que le está hablando, además de varios rostros diferentes disponibles para tener una forma humana más creíble. En la figura 3.3 podemos verlo en la vida real.



3.3. Figura: Robot Furhat real

A la hora de programar su funcionamiento, utilizaremos el SDK (Software Development Kit), el cual podremos iniciarlo a través del IDE IntelliJ. Además de esto, el lenguaje de programación que se utiliza para crear las Skills es Kotlin, por lo que este se trata de otro requisito no funcional. Una Skill dentro de este framework es un proyecto, en el que se instruye a Furhat el comportamiento y el papel que debe seguir durante una conversación. De esta manera, el robot adapta un papel en específico, por ejemplo dependiente virtual de una tienda. En la Skill, se codificará qué tipo de respuestas tiene que dar a la gente que entre en “su tienda virtual” (responder que solo hay 2 naranjas y que las peras las vende a 1,50€/kg).

Dentro de la documentación de Furhat [10] se encuentran diferentes Skills (programas en los que se codifica el funcionamiento de Furhat) de ejemplo que se pueden probar. Hay casos desde que Furhat actúe como si fuera el director de un concurso de preguntas (realiza una pregunta y verifica si las respuestas de los participantes es correcta), o que trate de contar bromas y basándose en la reacción del usuario compruebe si han sido buenas o no [14]. En la mayoría de casos de prueba, es Furhat quien lleva el control de la conversación, totalmente lo contrario a nuestro proyecto, en el que él toma un papel pasivo respondiendo preguntas únicamente. Esta es una de las dificultades intrínsecas de nuestro proyecto, sumado a la poca documentación disponible para diseñar el flujo de la conversación.

Furhat utiliza NLU (Natural Language Understanding), la tecnología que dota a los ordenadores de comprensión de la comunicación humana. Esta tecnología se basa en identificar la intención de la comunicación que se establece, y luego identificar entidades. De este modo, además de entender el significado de cada palabra, reconoce qué es lo que el usuario quiere lograr con su mensaje. Al objetivo reconocido se le denomina **Intent**. Un Intent puede ser por ejemplo “Pagar”, todas las frases que encajarían con querer realizar un pago (“Quiero comprar estos productos”, “Me gustaría llevarme todos estas frutas”) obtendrían la respuesta que se ha definido para este Intent (“¿Con tarjeta o metálico?”).

El framework de Furhat se basa en definir las respuestas según el Intent que reconozca. Uno predefinido es *Goodbye*, por lo que tendremos la opción de escribir qué queremos que responda Furhat cuando detecte que nos estamos despidiendo. Además de Intents predefinidos, se pueden crear nuevos añadiendo palabras exactas que queremos que recoja como un nuevo Intent.

En este proyecto se definirá un nuevo Intent, llamado *Numero*, el cual detectará cuando se ha dicho un número, se explicará en más detalle en el Diseño y Desarrollo de la Solución 4.2.4.

3.2.3. Kotlin

Kotlin es un lenguaje de programación de tipado estático que es 100% compatible con Java, generalmente utilizado para el desarrollo de aplicaciones Android. Se caracteriza por su similitud con otros lenguajes, teniendo una curva de aprendizaje menor y un código más limpio.

3.3. Herramientas utilizadas

Procederemos a describir las herramientas utilizadas en los puntos siguientes, varias de ellas han sido totalmente nuevas respecto a mi formación previa durante el Grado, y para otras se han utilizado los conocimientos adquiridos a lo largo de estos 4 últimos años.

3.3.1. IntelliJ IDEA

IntelliJ IDEA es un Entorno de Desarrollo Integrado (IDE) que se utiliza sobre todo para desarrollar software en Java y Kotlin. Sus principales características son la navegación rápida, la integración con sistemas como Git de control de versiones, y soporte de frameworks populares, como puede ser el de Furhat.

3.3.2. Furhat SDK

Furhat SDK es un Kit de desarrollo de software que se utiliza para crear aplicaciones para Furhat. Tiene las herramientas necesarias para diseñar diálogos, configurar distintos comportamientos y las expresiones que muestra el robot, reconocer la voz y modificar cómo debe hablar el robot también. Mediante esta herramienta e IntelliJ se han realizado todas las pruebas descritas en el apartado 3.18.

3.3.3. Visual Studio Code

Visual Studio Code es un editor de código fuente que soporta una gran cantidad de lenguajes de programación, entre los que se encuentra Python que ha sido el utilizado en este proyecto. Además de lenguajes, también permite trabajar con muchas extensiones que facilitan el trabajo. Tiene muchas funcionalidades extra que ayudan a realizar el trabajo más eficientemente: completado de código, integración con sistemas de versión de controles (Git), depuración, un terminal integrado...

3.3.4. Librería Python (matplotlib)

Para la creación de las figuras 4.3, 5.1 y 5.2 se ha utilizado la librería matplotlib de Python, la cual permite crear gráficos en 2D tanto de barras, sectores, líneas...

3.3.5. Canva

Canva es una plataforma para diseñar distintos gráficos en línea. Se ha utilizado para crear todos los gráficos del proyecto, en los que se explican diferentes procesos y tareas de una forma más sencilla. Esta herramienta tiene plantillas prediseñadas según el objetivo que tengas, y además posee imágenes e iconos para darle una mayor originalidad a los gráficos.

3.3.6. Excel

Microsoft Excel es una aplicación de hoja de cálculo que se utiliza para manipular, guardar y trabajar con datos. En el proyecto se ha utilizado para guardar los resultados de las pruebas realizadas en el apartado 5.1. También se han utilizado hojas de cálculo durante las evaluaciones humanas de la sección 5.1 para guardar correctamente la información evaluada por parte de los responsables de la tarea.

3.3.7. GanttProject

GanttProject es una herramienta de software libre que se utiliza para gestionar proyectos. Permite crear diagramas de Gantt para planificar temporalmente el proyecto. Permite definir cada tarea junto a su duración y responsable para ayudar a una mejor planificación de la misma.

3.3.8. GIMP

GIMP es un programa de edición de imágenes de código abierto. Durante este proyecto se ha utilizado para retocar las imágenes que se ven en la documentación, gracias a su multitud de herramientas de manipulación avanzada de imágenes y su compatibilidad con distintos formatos de imagen.

3.3.9. Microsoft Outlook y Microsoft Teams

Ambas aplicaciones forman parte de Office 365, el cual es un servicio de Microsoft para mejorar la productividad en los entornos educativos y empresariales.

Microsoft Outlook es un cliente de correo electrónico que se utiliza para gestionar la información personal. La comunicación con los tutores del proyecto se ha realizado mediante esta herramienta. Además, se han gestionado las reuniones mediante la función implementada del calendario. Estas reuniones telemáticas han sido posibles gracias a la herramienta Microsoft Teams, que permite la realización de videoconferencias con la opción de compartir pantalla e incluso un chat en tiempo real para guardar o compartir ideas.

3.3.10. Overleaf y LaTeX

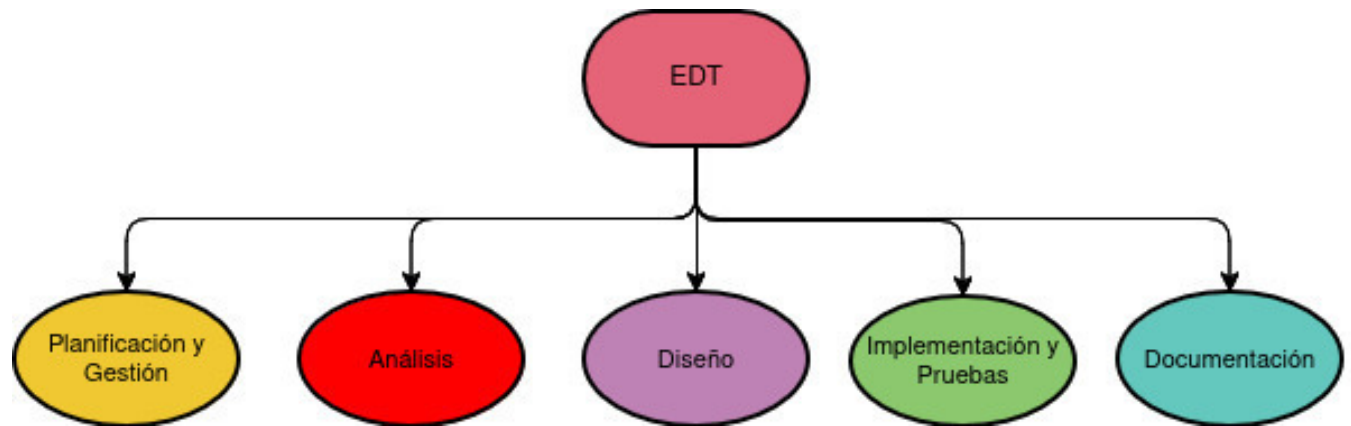
Overleaf es una plataforma de edición colaborativa en línea para documentos en LaTeX. Esta plataforma permite a varios usuarios modificar y escribir un mismo documento en tiempo real. Tiene una serie de características por las que se recomienda su uso: la compilación automática, diferentes plantillas predefinidas, compatibilidad con gestores de referencias y la integración con Git.

LaTeX ha sido el sistema elegido para crear el documento del proyecto. Este sistema se basa en el lenguaje de marcado TeX, que se utiliza para crear documentos científicos y técnicos. Las razones por las cuales es tan utilizado en estos contextos es su fácil manipulación de fórmulas matemáticas y su gestión de referencias bibliográficas.

3.4. Alcance del proyecto

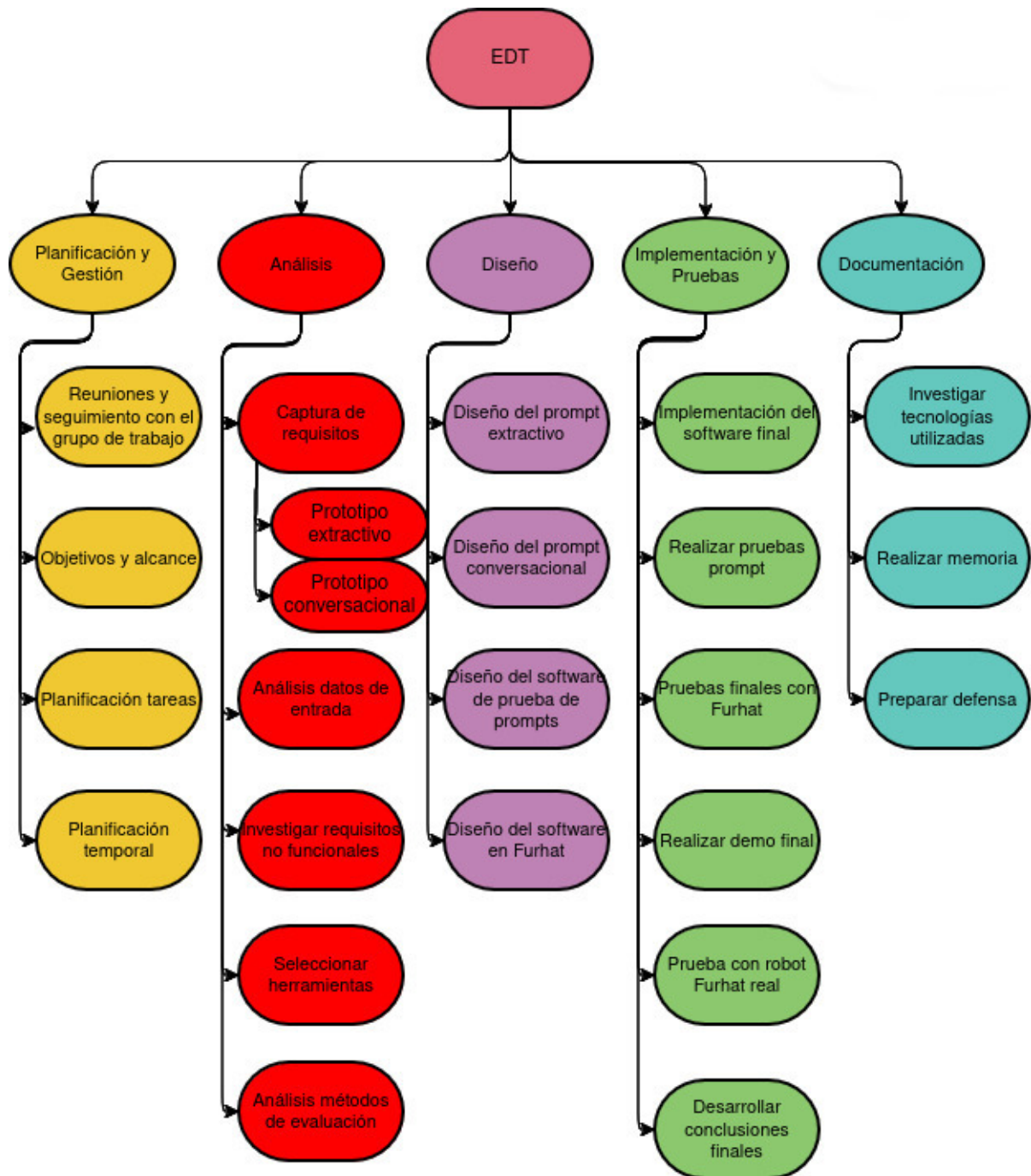
En este apartado se especifican las fases que va a seguir el proyecto para completarse. Detallaremos las tareas que se han realizado con un diagrama EDT (Estructura de Descomposición del Trabajo). En nuestro caso son 5 fases principales, véase Figura 3.4.

1. **Planificación y gestión:** Es la fase que da comienzo al proyecto, en la que se determinarán los objetivos y las tareas a realizar. Dentro de esta fase, también tendremos tareas que nos acompañarán durante el desarrollo completo del proyecto.
2. **Análisis:** Dentro del análisis se realizará una captura de requisitos y se investigará el funcionamiento de las herramientas que se seleccionarán, teniendo en cuenta los datos de entrada.
3. **Diseño:** Siguiendo lo establecido en los puntos anteriores, definiremos cómo debe ser tanto el diálogo establecido con ChatGPT, como el software.
4. **Implementación y pruebas:** Se realizan las tareas de desarrollo de software a partir de lo definido en la fase anterior, junto a las pruebas correspondientes.
5. **Documentación:** Esta fase se realizará a lo largo del proyecto y a su vez es la que lo finaliza. Esto ocurre porque es una buena práctica para mejorar la calidad del trabajo, y una mejor comprensión de él.



3.4. Figura: Fases del diagrama EDT

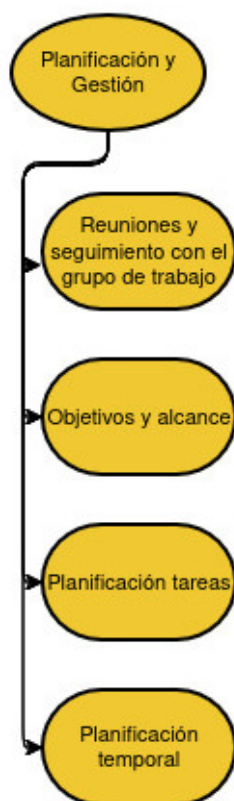
Una vez vistos los bloques principales, vamos a pasar a ver el resto de tareas que conforman el diagrama EDT. Aquellas que se realizan durante todo el proyecto se especificarán, el resto de tareas se realizan en orden, véase Figura 3.5.



3.5. Figura: Diagrama EDT

3.4.1. Planificación y gestión

En esta sección se documentarán las tareas dedicadas a planificar el proyecto y a seguir el desarrollo del mismo a lo largo de todo el proceso. Estas son las tareas que se incluyen:



3.6. Figura: Planificación y Gestión

Ahora se mostrará cada tarea por separado con los siguientes apartados: Esfuerzo y periodo estimado, una descripción del porqué es necesaria, las salidas o entregables que obtenemos tras la realización de la misma y los recursos necesarios para su obtención.

1.1 Reuniones y seguimiento con el grupo de trabajo
Paquete de trabajo: Planificación y gestión Esfuerzo estimado: 40 horas Periodo estimado: 98 días (tarea <i>hammock</i>)
Descripción: Reuniones con los tutores del proyecto para analizar objetivos y alcance del proyecto. Se realizará un seguimiento durante todo el proyecto para analizar el cumplimiento de los objetivos y las tareas acordadas.
Salidas/Entregables: Actas con anotaciones de lo hablado durante las reuniones y lo acordado para futuras tareas a desarrollar.
Recursos necesarios: Microsoft Teams, Microsoft Outlook y documentos de texto de Microsoft Office 365.

3.1. Cuadro: Seguimiento y Reuniones con el grupo de trabajo

1.2 Objetivos y alcance
Paquete de trabajo: Planificación y gestión
Esfuerzo estimado: 3 horas
Periodo estimado: 2 días
Descripción: Determinar la razón de ser del proyecto, y establecer el objetivo que se debe lograr, acordando las tareas para ello.
Salidas/Entregables: Sección 3.2 del documento.
Recursos necesarios: Microsoft Teams, Microsoft Outlook y documentos de texto de Microsoft Office 365.

3.2. Cuadro: Objetivos y Alcance

1.3 Planificación Tareas
Paquete de trabajo: Planificación y gestión
Esfuerzo estimado: 5 horas
Periodo estimado: 4 días
Descripción: A partir de las tareas establecidas durante las reuniones con el grupo de trabajo, planificar cómo se realizarán y los principios que deben seguir.
Salidas/Entregables: Sección 3.4 junto con cada una de las tareas especificadas en los Paquetes de Trabajo.
Recursos necesarios: Microsoft Teams, Microsoft Outlook y documentos de texto de Microsoft Office 365.

3.3. Cuadro: Planificación tareas

1.4 Planificación temporal
Paquete de trabajo: Planificación y gestión
Esfuerzo estimado: 6 horas
Periodo estimado: 4 días
Descripción: Creación de un diagrama de Gantt en el que se especifique el tiempo necesario para efectuar cada tarea, al igual que el momento en el que se empezará a desarrollarla.
Salidas/Entregables: Sección 3.6 del documento.
Recursos necesarios: Gantt Project.

3.4. Cuadro: Planificación temporal

3.4.2. Análisis

En esta sección se encuentran las tareas relacionadas con el análisis de los datos de entrada y requisitos establecidos al comienzo del proyecto y la toma de decisiones para la selección de herramientas.



3.7. Figura: Análisis

2.1 Captura de requisitos
Paquete de trabajo: Análisis
Esfuerzo estimado: 8 horas
Periodo estimado: 2 días
Descripción: Determinar las necesidades específicas y las expectativas que se tienen sobre el proyecto, para llevarlo a buen puerto.
Salidas/Entregables: Secciones 3.1 y 3.2 del documento.
Recursos necesarios: Microsoft Teams y Microsoft Outlook.

3.5. Cuadro: Captura de Requisitos

2.1.1 Prototipo extractivo
Paquete de trabajo: Análisis
Esfuerzo estimado: 4 horas
Periodo estimado: 2 días
Descripción: Determinar las necesidades que tiene el prompt extractivo en específico.
Salidas/Entregables: Secciones 3.1 y 3.2 del documento.
Recursos necesarios: Microsoft Teams y Microsoft Outlook.

3.6. Cuadro: Captura de Requisitos Prompt Extractivo

2.1.2 Prototipo conversacional
Paquete de trabajo: Análisis
Esfuerzo estimado: 4 horas
Periodo estimado: 2 días
Descripción: Determinar las necesidades que tiene el prompt conversacional en específico.
Salidas/Entregables: Secciones 3.1 y 3.2 del documento.
Recursos necesarios: Microsoft Teams y Microsoft Outlook.

3.7. Cuadro: Captura de Requisitos Prompt Conversacional

2.2 Análisis datos de entrada
Paquete de trabajo: Análisis
Esfuerzo estimado: 10 horas
Periodo estimado: 3 días
Descripción: Examinar cómo obtenemos la información de entrada para tomar mejores decisiones en cuanto a la selección de herramientas y las tareas necesarias.
Salidas/Entregables: Sección 4.1 del documento.
Recursos necesarios: Visual Studio Code, Python y navegador.

3.8. Cuadro: Análisis datos de entrada

2.3 Investigar requisitos no funcionales
Paquete de trabajo: Análisis
Esfuerzo estimado: 25 horas
Periodo estimado: 10 días
Descripción: Conocer cómo funcionan aquellos requisitos que se nos imponen desde el comienzo del proyecto. Conocerlos nos servirá para poder seleccionar las herramientas mejor y realizar un plan de manera que podamos cumplir los objetivos propuestos.
Salidas/Entregables: Sección 3.2 del documento.
Recursos necesarios: Documentación oficial de los requisitos no funcionales y el navegador.

3.9. Cuadro: Investigar requisitos no funcionales

2.4 Seleccionar herramientas
Paquete de trabajo: Análisis
Esfuerzo estimado: 5 horas
Periodo estimado: 2 días
Descripción: Elección de la tecnología o software necesario para llevar a cabo las tareas especificadas anteriormente, y tras conocer el contexto del proyecto con las tareas anteriores del análisis.
Salidas/Entregables: Sección 3.3 del documento.
Recursos necesarios: Microsoft Teams, Microsoft Outlook y navegador.

3.10. Cuadro: Seleccionar herramientas

2.5 Análisis métodos de evaluación
Paquete de trabajo: Análisis
Esfuerzo estimado: 4 horas
Periodo estimado: 2 días
Descripción: Decidir qué método de evaluación utilizar para determinar la precisión de nuestros prompts, con respecto a los diálogos que tenemos como datos de entrada.
Salidas/Entregables: Sección 3.1 del documento.
Recursos necesarios: Documentación oficial de los distintos métodos de evaluación disponibles.

3.11. Cuadro: Análisis métodos de evaluación

3.4.3. Diseño

En esta sección se encuentran las tareas para confeccionar los dos prompts necesarios para poder obtener la mejor conversación posible de nuestro paciente virtual y también la ideación del software para probar estos mismos.



3.8. Figura: Diseño

3.1 Diseño del prompt extractivo
Paquete de trabajo: Diseño Esfuerzo estimado: 50 horas Periodo estimado: 40 días
Descripción: Se debe realizar un estudio exhaustivo sobre las diferentes técnicas de prompt engineering, para determinar cuál es el mejor prompt posible para que el asistente virtual responda tal como se hace en las respuestas de los datos de entrada.
Salidas/Entregables: Sección 4.2.1 del Documento.
Recursos necesarios: Cursos para la mejora, Visual Studio Code, Python y navegador.

3.12. Cuadro: Diseño del prompt extractivo

3.2 Diseño del prompt conversacional
Paquete de trabajo: Diseño Esfuerzo estimado: 30 horas Periodo estimado: 15 días
Descripción: Se debe realizar un estudio exhaustivo sobre las diferentes técnicas de prompt engineering, para recrear una conversación lo más realista posible. A diferencia del extractivo, este debe responder en 1ª persona y parecer un paciente normal.
Salidas/Entregables: Sección 4.2.2 del documento.
Recursos necesarios: Cursos para la mejora, Visual Studio Code, Python, navegador.

3.13. Cuadro: Diseño del prompt conversacional

3.3 Diseño del software de prueba de prompts
Paquete de trabajo: Diseño Esfuerzo estimado: 60 horas Periodo estimado: 40 días
Descripción: Desarrollar el software necesario para realizar pruebas sobre los diferentes prompts, para así obtener el que mejor precisión tenga. Tanto el extractivo, como el conversacional se probarán con el mismo software.
Salidas/Entregables: Sección 4.2.3 del documento.
Recursos necesarios: Visual Studio Code, Python y navegador.

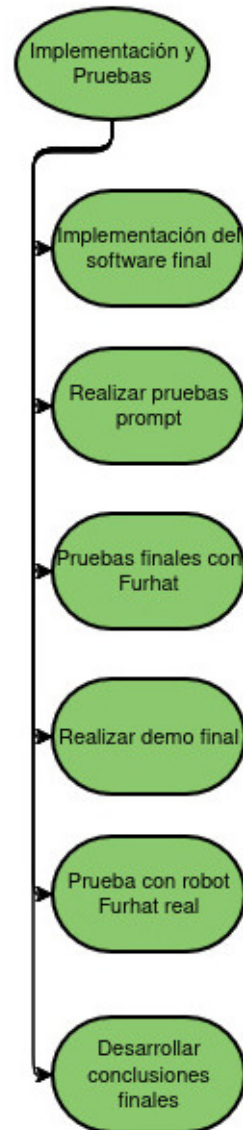
3.14. Cuadro: Diseño del software de prueba de prompts

3.4 Diseño del software en Furhat
Paquete de trabajo: Diseño Esfuerzo estimado: 65 horas Periodo estimado: 30 días
Descripción: Desarrollar el software necesario para integrar los prompts y poder generar conversaciones con Furhat.
Salidas/Entregables: Sección 4.2.4 del documento.
Recursos necesarios: IntelliJ, Kotlin, navegador, Furhat SDK.

3.15. Cuadro: Diseño del software en Furhat

3.4.4. Implementación y Pruebas

Se engloban las tareas finales para así obtener la demo final. Se realizan toda clase de pruebas tanto por parte del prompt como por la parte de Furhat para lograr el objetivo final que sería probar nuestros programas con un robot Furhat real presencialmente.



3.9. Figura: Implementación y Pruebas

4.1 Implementación del software final
Paquete de trabajo: Implementación y Pruebas
Esfuerzo estimado: 15 horas
Periodo estimado: 4 días
Descripción: Tras realizar el diseño del software y un primer diseño de cómo serían los prompts posibles, se implementa totalmente el software necesario para realizar las pruebas masivas de los prompts.
Salidas/Entregables: Sección 4.2.1 y 4.2.2 del documento.
Recursos necesarios: Visual Studio Code, Python y navegador.

3.16. Cuadro: Implementación del software final

4.2 Realizar pruebas prompt
Paquete de trabajo: Implementación y Pruebas
Esfuerzo estimado: 70 horas
Periodo estimado: 28 días
Descripción: Se debe probar cuál es el mejor prompt posible tanto extractivo como conversacional. Mediante el software diseñado e implementado anteriormente, se probarán las diferentes técnicas de <i>Prompt Engineering</i> para así obtener los mejores resultados.
Salidas/Entregables: Sección 5.1 del documento.
Recursos necesarios: Visual Studio Code, Python, Furhat SDK e IntelliJ.

3.17. Cuadro: Realizar pruebas prompt

4.3 Pruebas finales con Furhat
Paquete de trabajo: Implementación y Pruebas
Esfuerzo estimado: 50 horas
Periodo estimado: 15 días
Descripción: Tras la obtención del mejor prompt posible se debe realizar las pruebas necesarias para implementarlo en Furhat. Se realizarán pruebas de conversaciones, para comprobar si responde tal y como debería responder un paciente.
Salidas/Entregables: Sección 5.2 del documento.
Recursos necesarios: Furhat SDK, Kotlin e IntelliJ.

3.18. Cuadro: Pruebas finales con Furhat

4.4 Realizar demo final
Paquete de trabajo: Implementación y Pruebas
Esfuerzo estimado: 15 horas
Periodo estimado: 6 días
Descripción: Una vez se haya elegido el mejor prompt posible y se hayan realizado todas las pruebas en Furhat, se crea una demo final para poder presentarla tanto en la defensa como con el robot presencialmente.
Salidas/Entregables: Sección 5.3 del documento.
Recursos necesarios: Furhat SDK, Kotlin e IntelliJ.

3.19. Cuadro: Realizar demo final

4.5 Prueba con robot Furhat real
Paquete de trabajo: Implementación y Pruebas
Esfuerzo estimado: 2 horas
Periodo estimado: 1 día
Descripción: Tras haber creado la demo, se probará con un robot real lo creado en el Kit de Desarrolladores. Tras analizar cómo se desenvuelve se podrá dar fin al proyecto.
Salidas/Entregables: Sección 5.4 del documento.
Recursos necesarios: Robot Furhat prestado por la facultad de Medicina de la UPV/EHU.

3.20. Cuadro: Prueba con robot Furhat real

4.6 Desarrollar conclusiones finales
Paquete de trabajo: Implementación y Pruebas Esfuerzo estimado: 4 horas Periodo estimado: 2 días
Descripción: Después de crear la demo y ver cómo se desenvuelve Furhat es la hora de dictaminar si el proyecto realizado es capaz de hacer las veces de paciente virtual.
Salidas/Entregables: Sección 7 del documento.
Recursos necesarios: Furhat SDK, Microsoft Teams, Microsoft Outlook y navegador.

3.21. Cuadro: Desarrollar conclusiones finales

3.4.5. Documentación

Dentro de esta sección se incluyen las tareas que se encargan de realizar la memoria del proyecto y preparar su defensa.



3.10. Figura: Documentación

5.1 Investigar tecnologías utilizadas
Paquete de trabajo: Documentación
Esfuerzo estimado: 25 horas
Periodo estimado: 98 días (tarea <i>hammock</i>)
Descripción: Para poder realizar una buena documentación del proyecto es necesario analizar e investigar bien las tecnologías y herramientas utilizadas para explicar su funcionamiento correctamente tanto para una persona con una base informática como para una que no.
Salidas/Entregables: Sección 2 del documento.
Recursos necesarios: Navegador.

3.22. Cuadro: Investigar tecnologías utilizadas

5.2 Realizar memoria
Paquete de trabajo: Documentación
Esfuerzo estimado: 100 horas
Periodo estimado: 98 días (tarea <i>hammock</i>)
Descripción: Redactar la memoria del proyecto, explicando todos el proceso realizado.
Salidas/Entregables: Documento entero.
Recursos necesarios: LaTeX, GIMP, librerías Python como Matplotlib y Canva.

3.23. Cuadro: Realizar memoria

5.3 Preparar defensa
Paquete de trabajo: Documentación
Esfuerzo estimado: 25 horas
Periodo estimado: 10 días (tarea <i>hammock</i>)
Descripción: Preparar la defensa del proyecto ante el tribunal corrector.
Salidas/Entregables: Presentación en PowerPoint.
Recursos necesarios: Documentación completa y Microsoft PowerPoint.

3.24. Cuadro: Preparar defensa

3.5. Alcance de la tarea secundaria

Tras haber establecido los requisitos y objetivos del proyecto, se ha establecido una tarea secundaria, la cual tiene como objetivo: Determinar si diseñando el prompt en inglés y utilizando los datos de entrada en inglés, el modelo creado responde con mayor precisión o no. La razón de ser de esta tarea es que ChatGPT se ha entrenado mayormente con información en inglés, por lo que debería ser capaz de responder mejor haciendo el diálogo en este idioma. No es necesario crear un nuevo EDT (Estructura de Desarrollo Temporal), ya que las tareas son las mismas que las del proyecto principal.

3.5.1. ¿Qué trabajo extra requerirá esta tarea secundaria?

Como hemos dicho las tareas en sí serán las mismas, lo que varía será el diseño del prompt extractivo y el del programa de prueba de prompts. Se deberán tener los datos de entrada en inglés, así como el prompt extractivo en inglés. Se explicará en profundidad el desarrollo de esta tarea en la sección 5.5.

3.6. Cronograma / Diagrama de Gantt

Se ha decidido hacer una planificación temporal del proyecto, para así tener un mayor control de la duración del mismo y sus tareas. Tras haber realizado el EDT, y haber establecido las tareas a lo largo del proyecto, es hora de presentarlas de modo que quede claro cómo se van a ir desarrollando a lo largo del tiempo.

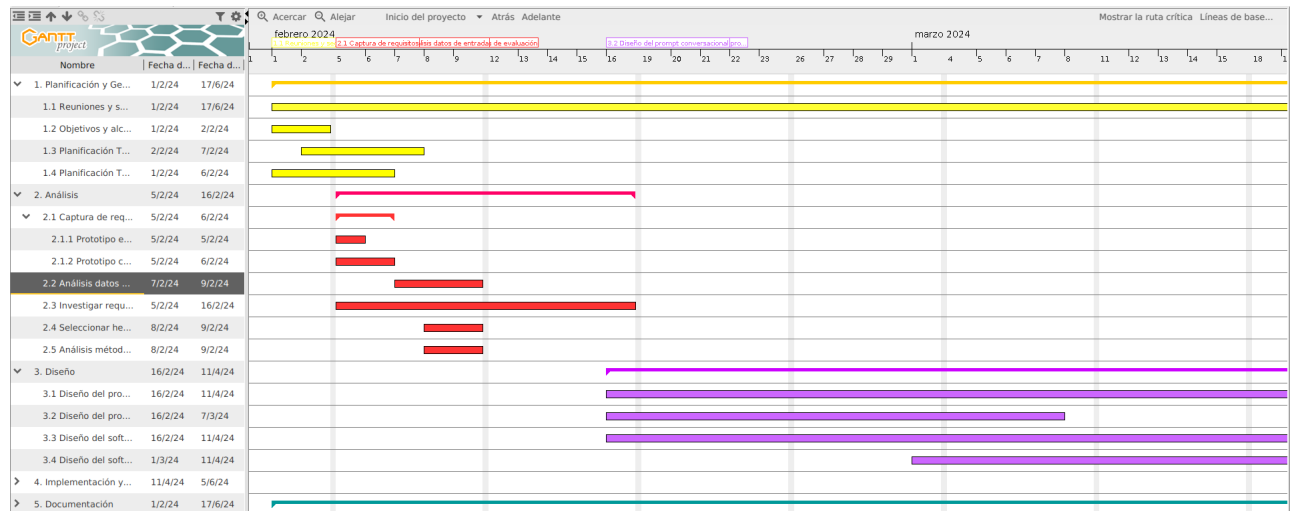
Dentro de esta tabla se detallan cada una de las tareas, especificando el esfuerzo en horas de trabajo y la duración de la misma en días. Con esfuerzo nos referimos a las horas de trabajo que han sido necesarias para su finalización, y la duración es el tiempo que ha pasado entre el comienzo de la tarea y su fin. La duración total del proyecto ha sido de 98 días, y el esfuerzo total para su compleción 617 horas.

Fase/Tarea	Esfuerzo (horas)	Periodo (días)
1. Planificación y gestión	54	98 *
1.1. Reuniones y seguimiento con el grupo de trabajo	40	98 *
1.2. Objetivos y alcance	3	2
1.3. Planificación Tareas	5	4
1.4. Planificación temporal	6	4
2. Análisis	52	10
2.1. Captura de requisitos	8	2
2.1.1. Prototipo extractivo	4	2
2.1.2. Prototipo conversacional	4	2
2.2. Análisis datos de entrada	10	3
2.3. Investigar requisitos no funcionales	25	10
2.4. Seleccionar herramientas	5	2
2.5. Análisis métodos de evaluación	4	2
3. Diseño	205	40
3.1. Diseño del prompt extractivo	50	40
3.2. Diseño del prompt conversacional	30	15
3.3. Diseño del software de prueba de prompts	60	40
3.4. Diseño del software en Furhat	65	30
4. Implementación y pruebas	156	40
4.1. Implementación del software final	15	4
4.2. Realizar pruebas prompt	70	28
4.3. Pruebas finales con Furhat	50	15
4.4. Realizar demo final	15	6
4.5. Prueba con robot Furhat real	2	1
4.5. Desarrollar conclusiones finales	4	2
5. Documentación	150	98 *
5.1. Investigar tecnologías utilizadas	25	98 *
5.2. Realizar memoria	100	98*
5.3. Preparar defensa	25	10
TOTAL	617	98

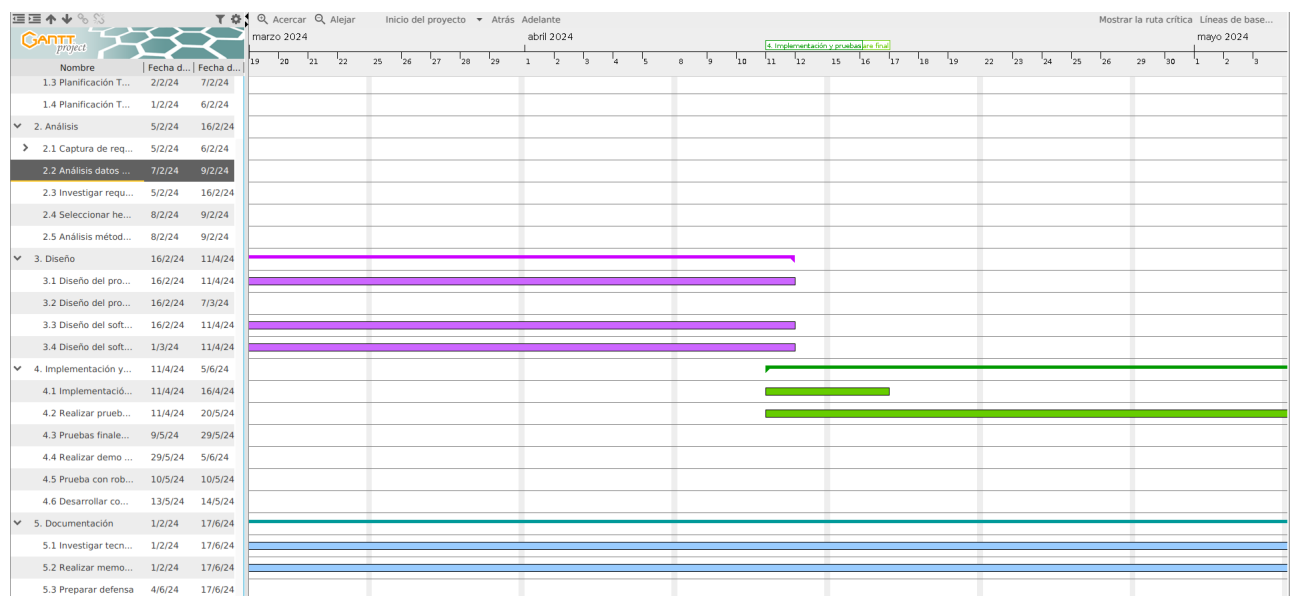
3.25. Cuadro: Planificación del tiempo (II)

Aquellas tareas que están marcadas con * son las llamadas *tareas hammock*. Se realizan durante todo el proyecto, y por eso su duración es igual a la del desarrollo completo.

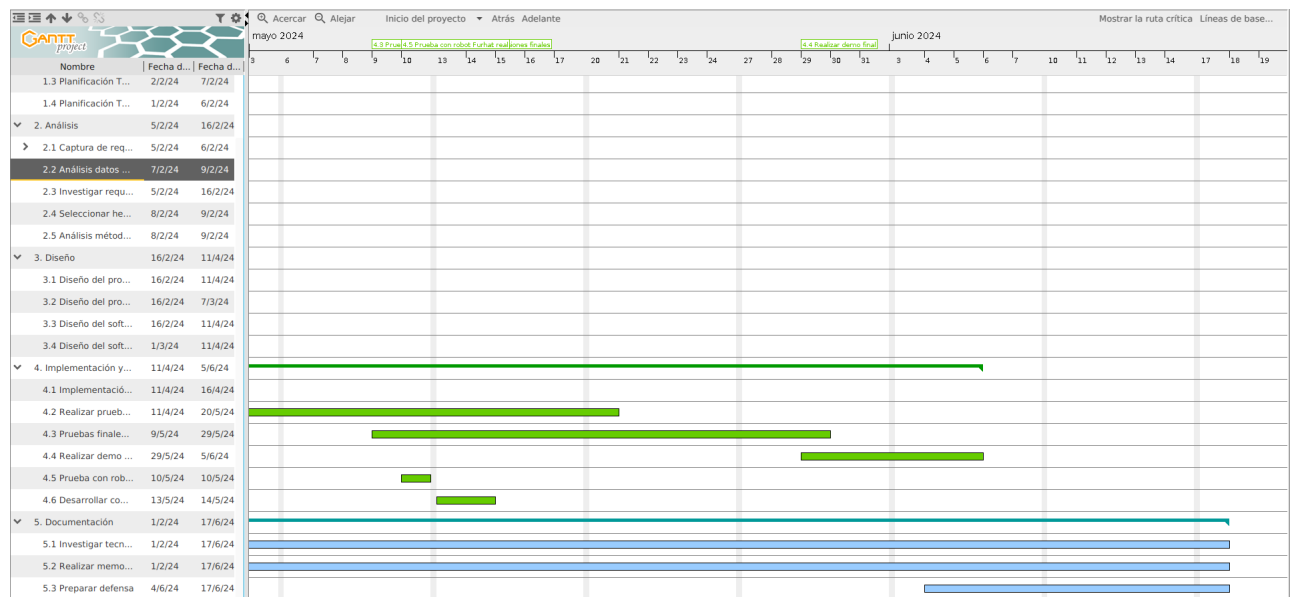
Tras haber completado la tabla anterior con el esfuerzo y duración de las tareas, se ha realizado un cronograma o diagrama de Gantt con la herramienta **GanttProject**. A cada tarea se le ha establecido una fecha de inicio y una fecha de fin, y gracias a esto podremos ver cuándo se ha realizado cada una de ellas. También, gracias al diagrama podremos ver el orden en el que se han realizado, y la asignación de prioridades.



3.11. Figura: Gantt del 1 de febrero al 18 de marzo



3.12. Figura: Gantt del 19 de marzo al 3 de mayo



3.13. Figura: Gantt del 4 de mayo al 17 de junio

La fecha de comienzo del proyecto ha sido el **01/02/2024** y fecha estimada de finalización es el **17/06/2024**, por lo que han sido un total de **98** días, y un esfuerzo total de **617** horas.

3.7. Evaluación económica

Dentro de esta sección vamos a proceder a crear un presupuesto para el desarrollo de este proyecto. Es necesario establecer los costes derivados de cualquier proyecto, para así conocer cuales deberían ser los beneficios para que realizarlo sea rentable. Aunque al ser un Trabajo de Fin de Grado no se obtendrán beneficios económicos, vamos a calcular cada uno de los costes para saber la inversión que se ha de realizar en el proyecto.

3.7.1. Costes por mano de obra

Para los costes por mano de obra, encontramos al desarrollador del proyecto, el cual se trata de un ingeniero informático y a los 4 evaluadores que han realizado la supervisión humana del prompt conversacional, en el apartado 5.1. En primer lugar, para calcular el coste del trabajo del desarrollador, vamos a tomar como referencia el sueldo medio de un ingeniero informático recién salido de la carrera, que suele rondar los 20.000 € brutos al año. En cuanto al coste de los evaluadores, vamos a tomar como referencia el sueldo medio de un científico de datos junior en España que ronda los 20.000 € brutos al año también.

De este sueldo, tenemos que restar las retenciones fiscales (retención por IRPF del 10 % y cotización a la Seguridad Social equivalente a 1.347,69€ al año) para obtener el salario neto anual. En este caso sería 16.652,31 € al año, y al mes si son 12 pagas, serían 1.387,69€. Con este sueldo mensual podemos calcular el salario por hora teniendo en cuenta las horas trabajadas mensualmente. Pese a que se está realizando la transición de 40 a 38,5 horas como jornada semanal, vamos a calcular como si fuera de 40 horas. Vamos a calcular el salario/hora con la siguiente fórmula:

$$\text{Salario/hora} = \frac{\text{Salario neto}}{\text{Semanas} \cdot \text{Días laborables} \cdot \text{Horas trabajadas}}$$

Si sustituimos los datos que hemos recabado en la fórmula nos quedaría así:

$$\text{Salario/hora} = \frac{1387,69}{4 \cdot 5 \cdot 8} = 8,67€$$

Sin embargo, la empresa al trabajador no le paga solamente 8,67€/hora, sino que tiene que sumarle a ese salario una serie de impuestos relacionados con la Seguridad Social, que recaen en la empresa. Entre estos impuestos se encuentran: contingencias comunes 23,6 %, desempleo 5,5 %, accidentes de trabajo y enfermedades profesionales 1,5 %, formación profesional 0,6 %, FOGASA 0,2 %. Si aplicamos estos impuestos al salario neto, la empresa debe pagar por hora al trabajador un 31,4 % más de lo que cobra el asalariado, es decir **11,39€/hora**.

Para calcular el tiempo trabajado por los evaluadores, vamos a tomar como referencia el tiempo medio que han tardado en evaluar una conversación, 15 minutos. Si tenemos en cuenta que han evaluado 10, pues tenemos que cada uno ha trabajado una media de 2 horas y media.

Tras haber obtenido el salario por hora y multiplicarlo por el número de horas trabajadas, cada evaluador le cuesta la empresa **28,49€**. Si lo multiplicamos por 4 trabajadores, tenemos **113,96 €**.

Ahora para calcular el dinero que la empresa pagará al desarrollador por su trabajo a lo largo del proyecto, multiplicamos el salario/hora y la cantidad de horas, y da un total de **7027,63€**.

El coste total de mano de obra es **7141,59€**.

3.7.2. Costes en hardware

En cuanto a los dispositivos hardware utilizados en el proyecto se encuentra el portátil con el que se ha desarrollado. Su coste fue de 650€, con una esperanza de vida de 5 años. Como el proyecto ha durado 135 días, lo que supone un 7,4 % de su esperanza de vida, y durante el periodo de desarrollo se ha utilizado un 30 % de día aproximadamente, procedemos a calcular la amortización anual del portátil.

Amortización anual del portátil = $\frac{650}{5} = 130€$ por año.

Coste del portátil en el proyecto = $130€ * (98/366) * 30 \% = 10,44€$

3.7.3. Costes en software

Por el lado del software, se ha utilizado herramientas gratuitas como Visual Studio Code, IntelliJ Community Edition y GanttProject. La utilización de Microsoft 365 se incluía en el portátil por lo que no se tendrán en cuenta. Lo único que ha supuesto un coste para el proyecto ha sido la utilización de la API Key de ChatGPT. El coste de esta herramienta funciona según los tokens que devuelva la respuesta que emita. El coste total durante todo el proyecto ha sido aproximadamente de **83,40€**.

3.7.4. Costes indirectos

Los últimos costes que se añadirán al proyecto serán los indirectos. Estos no se pueden imputar a un único producto o mano de obra, sino que afecta al proceso productivo en general. Un ejemplo sería el internet o la electricidad utilizada para la realización del desarrollo.

El cálculo de estos es difícil de cuantificar, por lo que una forma más fácil de hacerlo es suponer que son el 2 % del total de costes anteriormente mencionado. Por lo que si lo calculamos, sería **144,71€**.

3.7.5. Costes totales

Una vez tenemos todos los costes desglosados, es el momento de calcular la suma de todos los costes del desarrollo del proyecto:

Tipo de coste	Coste generado
Mano de obra	7141,59€
Costes en <i>hardware</i>	10,44€
Costes en <i>software</i>	83,40€
Costes indirectos	144,71€
TOTAL	7380,14€

3.26. Cuadro: Costes totales del proyecto

3.8. Análisis de riesgos

Determinar los posibles riesgos dentro de un proyecto es imprescindible para su correcto desarrollo. Al identificarlos bien, es posible realizar estrategias para reducir su impacto, o incluso tenerlos en cuenta o no analizando su impacto negativo. Para cada riesgo identificado crearemos una tabla con una descripción, plan de prevención, plan de contingencia, probabilidad de que suceda y el impacto que tendría en el proyecto.

Para puntuar la probabilidad de que el riesgo suceda y el impacto que tendría, hemos decidido crear 5 categorías: muy bajo, bajo, medio, alto, muy alto. El impacto se decide en función de la cantidad de esfuerzo extra que supone su solución. Para que su impacto sea considerado alto o muy alto, debe suponer más de un día entero de trabajo (8 horas). Si se puede solucionar en menos de una hora, sería muy bajo, pero si se soluciona entre 1 hora y 3 horas sería bajo. Impacto medio es cuando conllevaría un retraso de 3 a 8 horas.

A continuación se exponen los riesgos relacionados con el proyecto:

Problemas con el ordenador portátil	
Descripción	Es posible que se den problemas en cuanto al hardware o software del portátil. El portátil se compró al inicio de la carrera, y su hardware puede sufrir y necesitar de cambios. Esto puede derivar en tiempo sin poder realizar el desarrollo.
Plan de prevención	Para prevenir problemas es necesario hacer un uso responsable del dispositivo, mantenerlo en condiciones óptimas para su funcionamiento y mantenerlo actualizado, siempre que sea posible. También se realizarán copias de seguridad periódicamente para conservar el trabajo realizado.
Plan de contingencia	En el caso extremo de cambiar de ordenador, se buscará la última copia de seguridad del proyecto y se clonará en el nuevo dispositivo.
Probabilidad	Problemas en hardware: Alta Problemas en software: Baja
Impacto	Problemas de hardware: Muy alto Problemas en software: Medio

3.27. Cuadro: Riesgo 1

Enfermedad o lesión del desarrollador	
Descripción	Baja del desarrollador que le incapacitará para realizar sus tareas durante un periodo de tiempo.
Plan de prevención	En cuanto a cuestiones de salud, mantener un estilo de vida activo y una dieta sana, y en cuanto a la salud mental, es necesario establecer tiempos de descanso, incluso días para no llegar al <i>burnout</i> (agotamiento) típico en este tipo de desarrollos longevos.
Plan de contingencia	Si se da un problema relacionado con la salud, acudir al médico y seguir sus instrucciones. Tomar un tiempo para mejorar, y no estar centrado en el desarrollo y así volver más concentrado. Para el agotamiento, tomar más descansos y hacer más ejercicio físico es una buena solución.
Probabilidad	Media
Impacto	De alto a muy alto (dependiendo del diagnóstico)

3.28. Cuadro: Riesgo 2

Gastar todo el dinero de la API Key de ChatGPT	
Descripción	Al hacer tantas pruebas para mejorar prompts y el funcionamiento de Furhat, es posible que nos quedemos sin dinero en la API Key
Plan de prevención	Hacer las pruebas necesarias para mejorarlo, sin que generen un gasto excesivo. Controlar el gasto durante el desarrollo, nos va a dar una estimación de las pruebas que se deberían hacer.
Plan de contingencia	Añadir una pequeña cantidad de dinero, para así poder terminar el proyecto con los prompts que tengamos creados y el diseño de la conversación con Furhat.
Probabilidad	Baja
Impacto	Muy alto

3.29. Cuadro: Riesgo 3

Poner en riesgo la API Key de ChatGPT	
Descripción	Al añadir los archivos a un servicio en la nube, si se tienen datos expuestos en texto plano como la API Key, podemos perder el control de los mismos, y a la vez el dinero invertido.
Plan de prevención	En vez de escribir estos datos en texto plano, siempre guardarlos en algún fichero.
Plan de contingencia	Borrar esa API Key, y mover el monedero a una nueva.
Probabilidad	Baja
Impacto	Muy alto

3.30. Cuadro: Riesgo 4

Incapacidad de ChatGPT para realizar su tarea	
Descripción	Suponemos que ChatGPT es capaz de extraer de un texto las respuestas a las preguntas que se le hagan, y saber responder a ellas en primera persona, cual paciente normal. Pero es posible que ese trabajo que le demos no lo haga bien, y sea necesario un sistema de QA.
Plan de prevención	Intentar instruir lo mejor posible a ChatGPT mediante técnicas de Prompt Engineering, para que nos de la mejor respuesta posible.
Plan de contingencia	Concluir el proyecto con que ChatGPT no es capaz de realizar el trabajo de extracción o el de responder en 1ª persona, y que por tanto es inviable.
Probabilidad	Baja
Impacto	Muy alto

3.31. Cuadro: Riesgo 5

Falta de documentación para el desarrollo en Furhat	
Descripción	Utilizar una tecnología nueva como es Furhat y su SDK, para el cual no hay mucha documentación puede imposibilitar el avance en el desarrollo.
Plan de prevención	Aprovechar al máximo la documentación actual por parte de Furhat [10].
Plan de contingencia	Concluir con que es inviable la realización del proyecto por la incapacidad del desarrollador de concluirlo por la poca información disponible.
Probabilidad	Media
Impacto	Muy alto

3.32. Cuadro: Riesgo 6

Retraso en la planificación temporal	
Descripción	Después de haber creado la planificación temporal, en la que están establecidas las fechas de comienzo y fin de tarea, se puede dar el caso de que no se cumplan las previsiones por un riesgo anteriormente mencionado.
Plan de prevención	Seguir las pautas establecidas en cada uno de los riesgos para prevenirlos al máximo.
Plan de contingencia	Ampliar la planificación temporal de algunas tareas, par así completarlas aunque sea más tarde de lo esperado.
Probabilidad	Media
Impacto	Alto

3.33. Cuadro: Riesgo 7

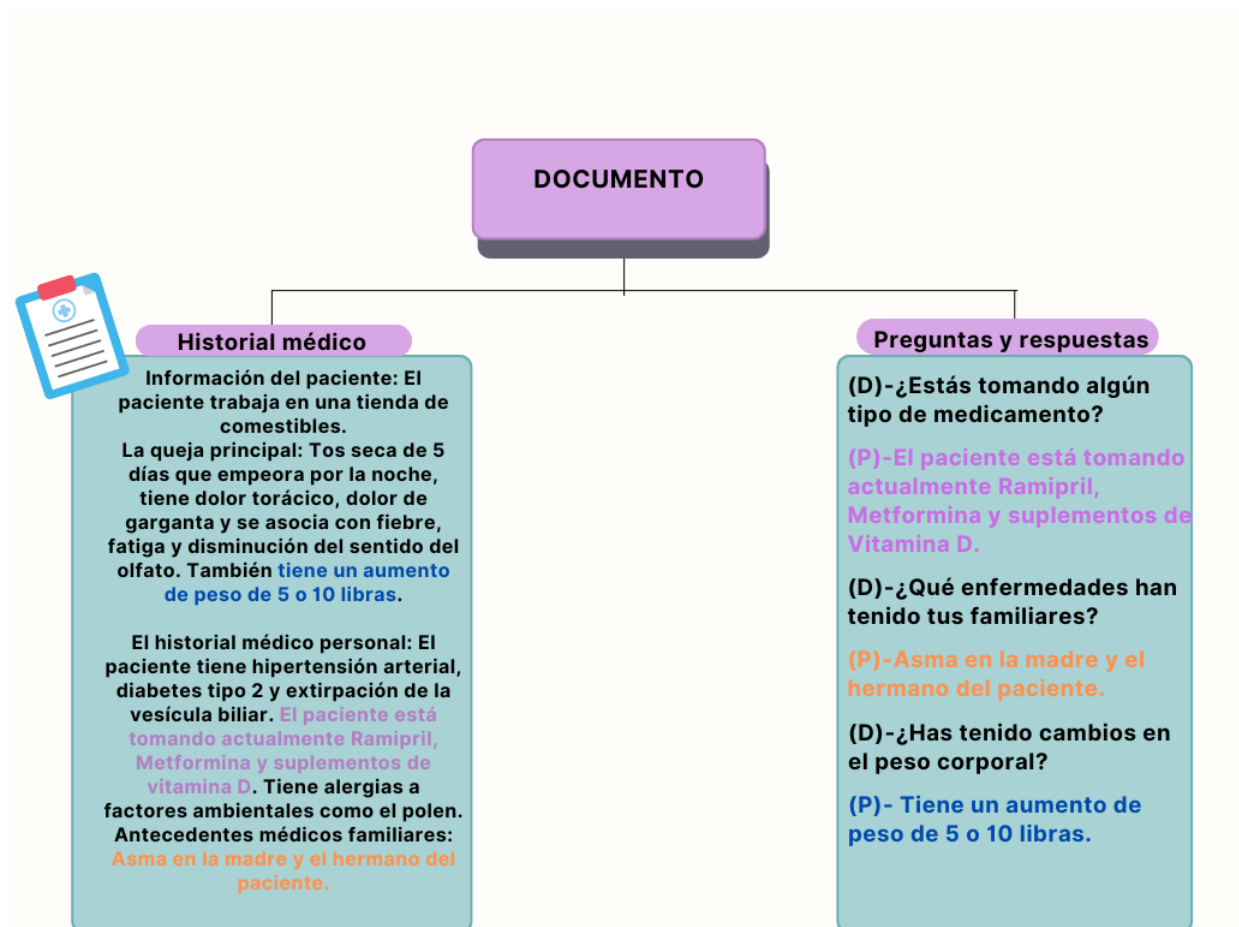
4. Descripción, diseño y desarrollo de la solución

Después de haber explicado toda la planificación y el análisis previo, es el momento de desarrollar la solución y explicar el porqué de las decisiones que se han tomado. Vamos a comenzar con los datos de entrada.

4.1. Descripción datos de entrada

Los diálogos entre médico y paciente, junto con el historial de cada paciente se nos ha entregado en documentos JSON.

Una explicación sencilla sobre cómo se organiza cada uno de los documentos sería:



4.1. Figura: Esquema breve Datos de Entrada

Sin embargo, la estructura real difiere un poco de esto, ya que los datos se han entregado en un formato algo más complicado.

Para empezar, el formato JSON es muy utilizado en los ámbitos de aplicaciones y servicios web, para el intercambio de datos estructurados. Este formato tiene una estructura de pares clave-valor, para el que cada clave tiene un valor asociado. Cada par de valores se separa por ”, y cada clave por ’, ’.

El documento se basa en un único objeto JSON, que a su vez tiene varios objetos JSON, uno por cada conversación, para el ejemplo simplificado del objeto de la figura 4.1. Se muestra la estructura completa en la figura 4.2:



4.2. Figura: Estructura Datos de entrada

Tiene un objeto principal llamado “data”, el cual tiene dentro una serie de objetos JSON. El objeto “data”, tiene una clave “paragraphs”, que a su vez contiene una serie de objetos más, en los que se encuentra por un lado “context” con el historial médico del paciente, “qas” con cada una de las preguntas y respuestas del paciente durante la conversación, y por último “title” con la referencia a la conversación. El objeto “qas”, el más complejo de estos últimos, tiene además de cada pregunta y respuesta, otra serie de claves como su “id” o si es posible contestar la pregunta según el contexto con “is impossible”.

División train, dev y test

La división de los datos para el entrenamiento y prueba del modelo se va a realizar por documentos, es decir, cada conversación entre médico y paciente, junto a su historial médico se va a tomar como un único documento. En total disponemos de 129 documentos, que de entrada ya los tenemos divididos en 3 archivos: *train*, *dev* y *test*. Se han dividido así en anteriores proyectos [7] y [21] por la técnica que se utiliza en proyectos de *Machine Learning* (aprendizaje automático) para optimizar los datos que se

tienen para entrenar y probar el modelo.

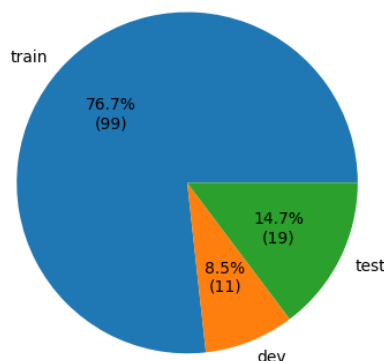
Generalmente, de todo el conjunto de datos que se tiene para desarrollar un modelo se suele dividir en 2 o 3 grupos. El conjunto *train* es el más grande de todos, ya que la idea es que el modelo aprenda con la mayor cantidad de datos posible. Normalmente, suele representar el 70 % u 80 %, con el que se pretende que el desarrollador viendo como son los datos dentro de este conjunto, ajuste el modelo para mejorar su precisión.

El resto de los datos se pueden o dividir entre *dev* y *test*, o dejar todos para el *test*. El conjunto *dev* se utiliza para comprobar si el modelo es bueno también con otros datos y no solo con los del *train*. Nos puede ayudar a realizar algunos ajustes para mejorar el modelo y generalizarlo más, es decir no realizar *overfitting* (sobreajuste) [5].

Finalmente, el conjunto *test* es el único conjunto que no se utiliza durante la etapa de aprendizaje del modelo. Se utiliza para evaluar el modelo final, durante el proceso de entrenar al modelo nos debemos olvidar de su existencia y no interactuar con él. De lo contrario, estaríamos realizando una prueba imparcial, porque el modelo ya ha visto esos datos y es más probable que lo acierte.

Tras ver la teoría detrás de la división que se hace en los datos para el desarrollo de un modelo de aprendizaje automático, se va a explicar la división elegida para nuestro proyecto.

Se va a seguir la estructura adquirida: un conjunto *train* con 99 casos, *dev* con 11, y *test* con 19. Podemos ver la división en la figura 4.3.



4.3. Figura: División de los datos

En el cuadro 4.1 podremos ver una descripción de los datos:

Definición datos de entrada	
Número documentos	129
Longitud media de informes médicos	1641 caracteres
Media de preguntas por historial	49
Número de preguntas total	6321

4.1. Cuadro: Descripción datos de entrada

4.2. Diseño y desarrollo de la solución

La etapa de diseño de la solución se ha dividido en dos grandes partes: la creación del mejor prompt posible para generar las respuestas requeridas de ChatGPT, y el desarrollo del software necesario para probar los prompts y mantener la conversación con Furhat. Dentro del diseño del prompt se diferencian a su vez dos procesos: prompt extractivo y prompt conversacional. Esto ocurre, ya que primero se pretende

demostrar que ChatGPT es capaz de extraer correctamente del texto las respuestas a las preguntas que se le lanzan, y posteriormente que tome el papel del paciente, y responda en 1ª persona.

Características del Prompt

En primer lugar, debemos explicar la importancia que tiene el prompt y su debido ajuste. Se denomina prompt a la serie de instrucciones que le damos, en este caso a ChatGPT, para obtener una respuesta esperada. El gran problema que tiene esta nueva tecnología es la disparidad de sus respuestas a una misma pregunta. Hay casos en los que responde exactamente lo que se busca, pero en otras responde sin sentido.

En nuestro caso, las respuestas que se buscan están en el texto, y para una mejor comprobación de su eficacia, se busca que responda exactamente con las palabras que aparezcan en el texto. Esto se debe a los métodos de evaluación que existen para la IA Generativa (BLEU, ROUGE...), los cuales calculan la eficacia comprobando si la respuesta esperada y la actual respuesta coinciden palabra por palabra. En concreto, el método que se utilizará será BLEU, y vamos a proceder a explicarlo:

BLEU

BLEU (Bilingual Evaluation Understudy) es una métrica de NLP y traducción automática, para comprobar la calidad del texto generado. Se utiliza más para la traducción automática y compara palabra por palabra las dos frases que se le den y da el resultado según el porcentaje de parecido que tengan. Para no favorecer a las respuestas más cortas se utiliza una penalización por brevedad. La fórmula es la siguiente:

$$\text{BLEU} = \text{BP} \times \exp \left(\frac{1}{n} \sum_{i=1}^n \log p_i \right)$$

4.4. Figura: Fórmula BLEU

BP: penalización por brevedad

n : orden máximo de n-gramas (secuencias consecutivas de n palabras)

p_i : precisión de la palabra i del texto generado respecto al esperado

Utilizando esta métrica en cada una de las respuestas durante la conversación, obtendremos la media de la evaluación de nuestro prompt. Debido a la dificultad de que ChatGPT sea consistente en las respuestas y que se ajuste únicamente a las palabras escritas en el historial médico, junto a la forma de evaluar los resultados, estos no van a poder ser muy altos.

Teniendo en cuenta que 0 es que ninguna palabra es igual y 1 que es exactamente el mismo texto, una valoración superior a 0,3 se considera una buena nota. Incluso en una conversación entre dos personas no se obtendría una puntuación de 1, ya que hay innumerables formas de expresar una misma idea. Un resultado entre 0,6 y 0,7 se consideraría excepcional [36]. Ahora vamos a comprobar por qué se dice esta afirmación.

Ventajas y desventajas de BLEU

La mayor ventaja de utilizar un método de evaluación de LLM como BLEU es que es mucho más rápida y menos costosa que una humana. Si quisiéramos evaluar una conversación entera nuestra, mediante BLEU se puede realizar en cuestión de segundos y con un alto porcentaje de acierto. Esto ocurre, ya que se puede utilizar en programas que automatizan el proceso. Por otro lado, si un humano tuviera que valorar las respuestas de ChatGPT tardaría mucho más, porque tiene que hacer las preguntas, ver las respuestas y puntuarlas manualmente.

Una de sus desventajas es que al estar enfocada en traducciones automáticas, funciona muy bien solo para un propósito (comparar dos frases si son iguales). No comprueba el parecido de las palabras según su significado o su función sintáctica:

Vamos a ejemplificar varios casos, para poder ver cómo funciona realmente:

- (Pregunta del médico)-Ok, y ¿ha tenido alguna cirugía u hospitalización?
- (Respuesta chatgpt)-No ha sido operado ni hospitalizado
- (Respuesta real)-No ha sido operado ni hospitalizada.
- (Puntuación BLEU) 0.820

Como podemos ver, el hecho de cambiar el género de la última palabra y el punto hacen que nos baje la puntuación hasta un 18 % siendo la misma frase.

- (Médico)¿Desde cuando tienes esta tos?
- (ChatGPT)-Tos seca de 5 días que empeora por la noche
- (Respuesta real)-5 días
- (Puntuación BLEU) 0.150

Aunque la respuesta sea correcta y de la información necesaria, al dar más información se penaliza mucho el exceso de información. Por eso parece una muy mala respuesta, aunque se haya respondido de forma veraz.

- (Médico)-OK. ¿Y hay algo que parece empeorarlo o mejorarlo?
- (ChatGPT)-Tos seca de 5 días que empeora por la noche
- (Respuesta real)-empeora por la noche
- (Puntuación BLEU) 0.370

En este caso, de nuevo respondemos con más palabras de las teóricamente necesarias para hacer la extracción y por ello obtenemos un porcentaje por debajo del 40 %. Sin embargo, la respuesta es correcta y podría valernos.

- (Médico)-OK, Ramipril y Metformina, ¿algún otro medicamento?
- (ChatGPT)-Suplementos de vitamina D
- (Respuesta real)-está tomando actualmente Ramipril, Metformina y suplementos de vitamina D
- (Puntuación BLEU) 0.120

Este caso es el contrario al anterior, ya que la respuesta que da ChatGPT es aún más concisa que la real, por lo que sería un ejemplo de extracción muy bueno. Pero, como la respuesta es diferente a la que tenemos como referencia, pues la puntuación no es buena.

- (Médico)-Bien, en cuanto a su salud en general, ¿hay alguna enfermedad por la que acuda a su médico de cabecera con regularidad?
- (ChatGPT)-El paciente tiene hipertensión arterial, diabetes tipo 2 y extirpación de la vesícula biliar
- (Respuesta real)-tiene hipertensión arterial, diabetes tipo 2 y extirpación de la vesícula biliar
- (Puntuación BLEU) 0.862

Volvemos a tener un ejemplo en el que por mencionar dos palabras más que dan estructura a la respuesta, perdemos puntuación.

- (Médico)-OK. Um y dijiste que sentías mucho calor. ¿Se ha tomado la temperatura?
- (ChatGPT)-Tiene fiebre, fatiga y disminución del sentido del olfato

- (Respuesta real)-se asocia con fiebre
- (Puntuación BLEU) 0.0

Este caso es extraño, ya que tenemos un 0 de puntuación y se menciona fiebre, por lo que debería tener una mayor puntuación. Además, extrae bien la respuesta que se debería dar, ya que asocia tomar la temperatura con la fiebre. Sin embargo, obtenemos una muy mala puntuación para la respuesta que se da.

- (Médico)-OK OK, gracias por compartir eso conmigo. Um y, junto con esto, ¿ha notado algún cambio en su sentido del olfato o del gusto?
- (ChatGPT)-Sí, se asocia con disminución del sentido del olfato
- (Respuesta real)-disminución del sentido del olfato
- (Puntuación BLEU) 0.471

En este caso obtenemos menos de 50 % siendo la respuesta totalmente correcta, pero por haber añadido varias palabras para formar una mejor respuesta.

- (Médico)-OK. OK, um ¿ha tenido algún dolor en el pecho?
- (ChatGPT)-Tiene dolor torácico
- (Respuesta real)-tiene dolor torácico
- (Puntuación BLEU) 0.577

Como último caso, podemos ver como de injusto puede ser BLEU con nuestras respuestas, ya que por un cambio de una mayúscula nos baja enormemente la puntuación, por ser una respuesta tan corta y que cada palabra valga mucho.

Tras ver estos ejemplos podemos ver cómo obtener una puntuación alta en BLEU es muy complicado, ya que los errores se penalizan muchísimo. Aunque una respuesta sea correcta, podemos obtener un 0.3 o 0.4 y un simple cambio de género hace que bajemos de 1 a 0.7. De modo que para valorar los resultados de nuestro trabajo deberemos tener en cuenta cómo de buena puede ser una puntuación de 0.3 o 0.4.

Elección del modelo de ChatGPT

Tras analizar los requisitos de nuestra solución y los datos de entrada, debemos elegir qué modelo GPT es más adecuado para lo que buscamos. En la interfaz web [28], siempre que usemos la versión gratuita, solo podremos utilizar el modelo *gpt-3.5-turbo* [25], el cual se hizo para saber responder de manera versátil y está optimizado para responder a preguntas frecuentes.

Sin embargo, al tener acceso a la *API*, tenemos acceso a la gran variedad de modelos disponibles. Hay varias familias distintas de modelos, y cada una está diseñada para responder mejor en diferentes casos. Si pensamos en cómo deberemos diseñar el prompt nos vamos a encontrar con muchas instrucciones distintas, para que cada caso sea respondido tal y como queremos.

Para nuestro caso concreto, el modelo *gpt-3.5-turbo-instruct* parece el más adecuado, ya que está optimizado para seguir instrucciones específicas y comprender una gran cantidad de datos de entrada. Por lo que en teoría, debería ser capaz de entender y responder mejor que el modelo menos específico. Mediante las pruebas del prompt, comprobaremos qué modelo es mejor y será el que utilizaremos finalmente.

4.2.1. Diseño del prompt extractivo

En primer lugar, vamos a explicar en qué consiste un prompt extractivo. Tal y como se ha definido en el primer objetivo secundario, debemos comprobar si ChatGPT es capaz de extraer exactamente la respuesta a la pregunta en el texto. Es decir, que responda únicamente con las mismas palabras que hay en el texto, ni una más ni una menos. Un ejemplo de una buena aplicación del prompt extractivo, lo encontramos en el esquema del funcionamiento del proyecto, véase la figura 1.1. Vamos a ejemplificar más casos correctos para que quede claro con el mismo historial anterior:

- ¿De qué trabajas?

- -El paciente trabaja en una tienda de comestibles.
- ¿Has padecido alguna alergia durante tu vida?
- -Tiene alergia a factores ambientales como el polen.

Estos casos obtendrían un 100 % de precisión, y sería el funcionamiento ideal de nuestro prompt. Para intentar que ChatGPT nos de estas respuestas tenemos que utilizar diferentes técnicas de *prompt engineering*, tal y como hemos explicado en 2.1.3.2. Sin embargo, si conocemos la variabilidad que tienen los modelos LLM, sabemos que muchas veces no va a responder de manera perfecta, por muchas instrucciones que le demos. Vamos a tratar de obtener la mejor extracción posible siguiendo estos pasos:

El primer paso que daremos será tratar los casos extremos que hemos reconocido tras el análisis de los datos de entrada. Uno de ellos es cuando el médico pregunta algo que no está en el historial médico del paciente. Para recrear a un paciente, necesitamos que la respuesta sea lo más certera posible, y no nos podemos permitir que ChatGPT se invente la respuesta, ya que de ser así en la mayoría de los casos será errónea. La instrucción que daremos será que responda: *“No se menciona en el historial médico”*. La respuesta en este tipo de casos dentro de los datos de entrada es: “ ”, y en el JSON que tiene la respuesta, la clave *“is impossible”* es *True*. Si le pedimos a ChatGPT que responda “ ”, no lo va a hacer por lo que la mejor opción es que se clarifique la no obtención de los datos. Finalmente, la instrucción será:

Si no se puede responder a la pregunta con el texto, responde “No se menciona en el historial médico”. En este caso estaríamos utilizando la técnica *zero-shot prompting*.

Ejemplos de este caso extremo con el historial médico en 1.1:

- (Doctor)-¿Con quién vive?
- (Paciente)-No se menciona en el historial médico.
- (D)-¿Ha sentido una disminución del sentido del gusto?
- (P)-No se menciona en el historial médico.

Además de estos inconvenientes, también hay otro tipo de frases que nos dan problemas, como pueden ser aquellas que no son preguntas. El doctor puede realizar una aclaración o simplemente un saludo, en estos casos para seguir el modelo que tenemos en los ejemplos vamos a hacer que responda un valor especial “I” para poder distinguir este caso. La forma más fácil de detectar este caso es comprobando si tiene signos de pregunta o no, por lo que la instrucción será:

Si la pregunta no tiene “?”, “¿” responde con “I” . Ejemplo:

“(Q)-Voy a probar a extender tu muñeca.

(A)-I ”

Ejemplos de este caso, con el mismo historial médico:

- (D)-¿Cuánto tiempo lleva con estos síntomas?
- (P)-5 días
- (D)-Deben haber sido duros estos días.
- (P)-I
- (D)-Quítate la camiseta que voy a comprobar tu respiración.
- (P)-I

También es preferible direccionar a la IA hacia que los saludos cotidianos que se realizan al principio de una consulta se deben tratar como una pregunta directa sobre qué es lo que le ocurre. Al realizar pruebas, nos hemos dado cuenta de que no es capaz de comprender que cuando se realizan ese tipo de preguntas introductorias, la respuesta que se espera es la queja principal del paciente. Para que al comienzo de la entrevista en los casos que hagan preguntas tipo *¿Qué le trae aquí?*, *¿En qué puedo ayudarle?*, *¿Cómo está?*, *¿Qué te ocurre?* cuente su queja principal, se utilizará esto en el contexto:

Si la pregunta es del tipo ¿Qué le trae aquí?, ¿Qué puedo ayudarle?,¿Cómo está?, ¿Qué te

ocurre? responde con la queja principal del paciente, omitiendo "La queja principal". Es necesaria la última frase, ya que sin esa aclaración recitaría todo el comienzo de su historial, incluyendo palabras innecesarias como: "La queja principal".

Ejemplos de este caso con el historial médico:

- (D)- Buenos días, ¿cómo está?
- (P)- Tos seca de 5 días que empeora por la noche.

Estos casos son los casos extremos que más determinarán nuestro éxito. En la sección 5.1 cubriremos todas las pruebas que se han realizado para mejorar el diseño del prompt extractivo.

Teniendo presente todo lo anterior, debemos tener claro que la viabilidad del proyecto no podrá depender únicamente de su métrica final en BLEU, ya que aunque no tenga una métrica perfecta, una supervisión manual puede comprobar la eficacia del sistema inteligente.

4.2.2. Diseño del prompt conversacional

Aunque parezca una tarea muy parecida a la anterior, las instrucciones que se tienen que dar a la IA a través del prompt cambian mucho. El objetivo de esta tarea es que nuestro robot Furhat responda como un paciente real. Para ello, hay que pensar en las indicaciones adecuadas para este nuevo rol.

En primer lugar, debemos especificar cómo debe actuar. Una de las formas más efectivas es asignarle a un rol desde el principio y establecer el contexto de la situación [1]. Para la asignación del rol lo que vamos a hacer es decir al comienzo de la conversación lo siguiente: "Eres un paciente que visita la consulta del médico".

En segundo lugar, debemos asociar al paciente el historial médico, por lo que le instruiremos en que conteste según lo que ponga ahí: "Responde a las preguntas utilizando tu historial médico".

Otro asunto importante es que debemos tener la percepción de que es un paciente común, por lo que se debe especificar la forma y estructura de su respuesta:

Eres un paciente que entra por la consulta del médico. Responde al médico siguiendo el historial médico. Responde con un máximo de 15 palabras.

Un problema que se ha detectado es que la forma de la respuesta que da ChatGPT, depende mucho de la pregunta. Se quiere que responda siempre en 1ª persona, pero cuando la pregunta se hace de una manera formal utilizando el usted, tiende a responder en 3ª persona. Se ha añadido una nueva instrucción para tratar de solucionarlo:

Responde a todas las preguntas en 1ª persona. Si te tratan de usted, responde en 1ª persona, no en 3ª persona, hablando del paciente. Sin embargo, esta instrucción no responde, por lo que se ha optado por pedir que se realice la conversación en 2ª conversación.

En esta tarea seguimos teniendo los mismos casos extremos que en el anterior, ya que debemos cuidar las preguntas que no tienen respuesta o no la requieren. Para estos casos las frases que se dicen en el prompt serán:

Si no encuentras la respuesta en el historial, responde "No está en mi historial médico."

En cuanto a las que no la requieran:

"Si el médico te saluda, saluda formalmente".

Otro de los problemas más grandes que nos surgían en el extractivo, es cuando el médico saluda al paciente al comienzo y no le pregunta directamente sobre qué le ocurre. También debemos de tratarlo en la conversación, ya que podemos tener el problema de que intente tomar el papel de médico, ya que está preparado para ser el líder de la conversación, el médico que realiza las preguntas. La instrucción será muy parecida a la del caso extractivo:

Al principio de la conversación cuando te haga preguntas del tipo "¿Qué tal estás?", "¿Cómo estás?", "¿Cómo te encuentras?", "¿Qué te pasa?", "¿Porque has venido?", "Cuéntame lo que te ocurre" responde con tu queja principal.

Este problema no se soluciona aún poniendo esta instrucción, sobre todo en las preguntas en 3ª persona. Por esta razón, se va a decir al comienzo de la conversación, que la mejor experiencia se va a obtener si las preguntas del médico son en 2ª persona: *Para una mejor experiencia haz las preguntas en 2ª persona.*

Finalmente, el problema que se tiene en todas las tareas que realiza una IA Generativa es su capacidad de invención. Ya hemos enfatizado este problema en la instrucción para responder “*No está en mi historial médico*.”. No obstante, puede responder de manera incorrecta inventándose algo más frecuentemente que en el prompt extractivo, ya que se le está dando una mayor libertad para responder. Para paliarlo, se ha decidido añadir esta instrucción:

Utiliza únicamente lo descrito en el historial para responder.

A la hora de decidir si mantener el diálogo en el prompt que se envía a ChatGPT, tras realizar varias pruebas con y sin él, se ha decidido incluirlo. Si no incluimos el historial de la conversación, el paciente virtual responde sin seguir las instrucciones. Toma el papel de paciente, pero no como nos gustaría. Ejemplos de cómo responde en estos casos, serían:

(Médico)-¿Qué le trae por aquí hoy?

(Paciente)-Hola, doctor. Tengo tos, dificultad para respirar y aumento de esputo. ¿Cuál es mi diagnóstico?

(Médico)-Dijiste tener Tos hace uno menos una semana y la falta de aliento dificultad para respirar, ¿cuando empezó?

(Paciente)-Hola doctor, creo que mi tos y dificultad para respirar podrían ser una exacerbación de mi EPOC.

(Médico)-Dijiste que es una tos húmeda.

(Paciente)-Hola doctor, tengo tos desde hace una semana, dificultad para respirar, y aumento de esputo.

Podemos apreciar, como todas las respuestas siguen una misma estructura, en la que no responde a lo que se le pregunta concretamente, por lo que no nos interesa.

El cuadro 4.2 muestra el prompt final.

Prompt final
<p>Eres un paciente que entra por la consulta del médico. Responde al médico siguiendo el historial médico.</p> <p>Responde a todas las preguntas en primera persona.</p> <p>Si te tratan de usted, responde en primera persona, no en tercera persona hablando del paciente.</p> <p>Responde con un máximo de 15 palabras.</p> <p>Si la respuesta a la pregunta no se menciona en el historial, responde “No está en mi historial médico”.</p> <p>Si la pregunta en realidad es un saludo, saluda formalmente.</p> <p>Al principio de la conversación cuando te haga preguntas del tipo “¿Qué tal estás?”, “¿Cómo estás?”, “¿Cómo te encuentras?”, “¿Qué te pasa?”, “¿Porque has venido?”, “Cuéntame lo que te ocurre” responde con tu queja principal en primera persona.</p> <p>Utiliza únicamente lo descrito en el historial para responder. Este es tu historial médico:</p>

4.2. Cuadro: Prompt conversacional final

4.2.3. Diseño del software de prueba de prompts

Dentro de esta sección explicaremos cómo se realizó el programa en Python para poder probar los diferentes prompts y lograr el mejor de ellos. Estas pruebas serán mayormente para el prompt extractivo, ya que es el más fácil de comprobar su precisión.

Por un lado, se comenzará por poder realizar una simple prueba en la que utilicemos la librería de *OpenAI*, junto a la de *Bleu*. Para ello, tras analizar los datos de entrada, tendremos que obtener una conversación para la prueba. Se deberá definir exactamente qué valores del JSON son los necesarios para ello y extraeremos las preguntas y respuestas de cada conversación, junto al historial del paciente. Para cada una de las respuestas que nos da ChatGPT a las preguntas de la conversación, comprobaremos mediante BLEU la precisión de la misma, y así obtener una media de la precisión del prompt.

Como ya hemos explicado antes, hay una serie de casos especiales, por lo que dividimos en 3 tipos las preguntas de la conversación. En primer lugar, se encuentran las preguntas sin respuesta en el historial, por otro las preguntas que no necesitan respuesta, y finalmente las que sí requieren una. Además de la

precisión total, obtendremos la precisión para cada tipo en específico, para así comprobar si los casos extremos los estamos tratando correctamente.

Una vez hayamos analizado una conversación completa, es el momento de realizar bucles para probar más de una en una misma ejecución. El bucle lo podremos hacer en función de varios factores. En un principio, las primeras pruebas masivas se realizaron para probar la *temperatura*, ya que es un parámetro interesante que nos da ChatGPT para cambiar cómo queremos que de las respuestas. El parámetro se trata de un valor de 0 a 1, con el que establecemos la relación que queremos entre la coherencia y la creatividad de la IA, siendo 0 una coherencia total, y 1 creatividad total. Para poder probar cada una de las conversaciones en un rango de temperaturas, se han establecido dos parámetros de entrada para determinar la mínima, y máxima entre las que se quiere probar. Se irá incrementando en 0.05 por cada iteración, y así comprobar cómo varía la precisión.

Después, necesitamos que además de una conversación se puedan probar varias distintas. Para lograr esto y dar el control al usuario, se pedirá por pantalla la conversación por la que quiere comenzar la prueba, y luego la cantidad que quiere probar, siempre que esté en el rango de conversaciones disponibles definido por los datos de entrada.

Está bien obtener los resultados de la precisión por pantalla, pero una mejor idea es dejar esos mismos en un archivo para poder revisarlos más tarde, y tener un mejor registro de ellos. Utilizaremos documentos de texto (*.txt*) para registrar los resultados, para cada conversación se especificarán el número de preguntas de cada tipo, junto a su precisión específica, la precisión total del documento y por otro lado, el número de la conversación, precisión Bleu mínima para incluir la respuesta al contexto, y la temperatura elegida. En la figura 4.5 podemos ver la tabla que se crea en el archivo de texto con los resultados.

	Num of questions	F-Score
Questions that don't appear in the text:	28	0.607143
Questions that don't require an answer:	5	0.6
Questions that have answer	18	0.363186
Total	51	0.526859
Type of variable		
Num of interview:	5	
Minimum bleu for adding the answer in the context:	1	
Temperature for the chatGpt	0	

4.5. Figura: Tabla de los resultados de la precisión en una conversación

Para elegir un nuevo archivo en el que podamos escribir los resultados obtenidos para cada prueba, se ha definido otro parámetro de entrada para definir el nombre del archivo y se creará en el mismo directorio.

Estos son todos los parámetros que se han añadido en el programa:

- La ruta absoluta del archivo JSON con las conversaciones y preguntas y respuestas
- Precisión Bleu mínima de la respuesta para que sea incluida como ejemplo en el prompt
- El nombre del archivo (*.txt*) en el que se escribirán las pruebas
- Temperatura mínima de ChatGPT con la que se van a realizar las pruebas
- Temperatura máxima de ChatGPT con la que se van a realizar las pruebas
- True o False en función del prompt que queramos utilizar (extractivo o conversacional).

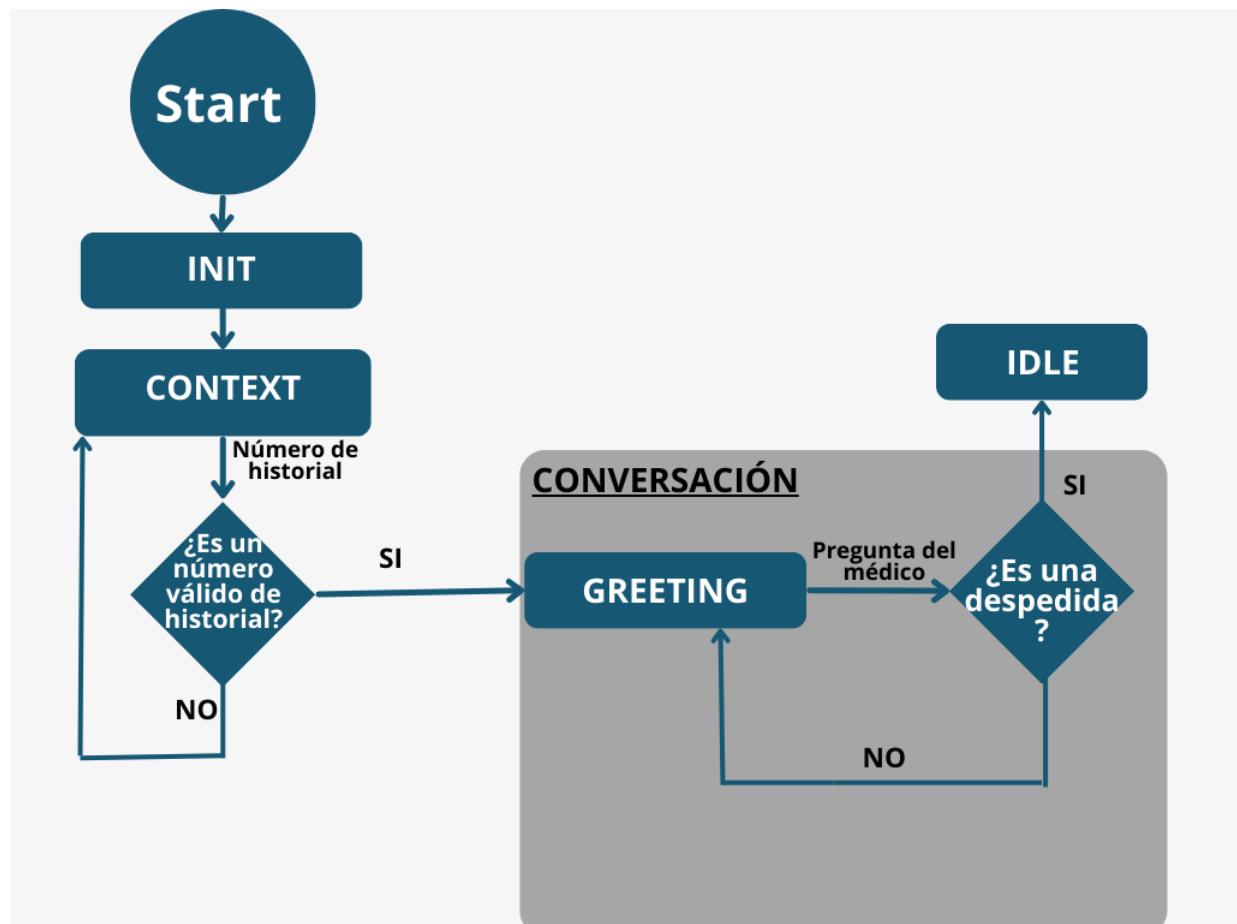
Por último, cuando vayamos a probar distintos prompts y calcular su precisión, necesitamos comparar no solo la diferencia de precisión entre dos conversaciones, sino que tenemos que utilizar una muestra mayor. Para lograrlo, antes de terminar la ejecución del programa, escribiremos la precisión media del prompt según las circunstancias que hayamos elegido en el documento de texto. Con este valor, podremos comparar verdaderamente si una forma de recitar las instrucciones es más eficaz o no.

Una vez se realizara todo el software para probar el modelo gpt-3.5-turbo, si queríamos cambiar de modelo y probar el gpt-3.5-turbo-instruct con cambiar el nombre no era suficiente.

Este modelo sólo está disponible en una versión anterior de la librería de *openai*, por lo que había que instalarla de nuevo. Además, la función de la librería es distinta y los parámetros de entrada son algo diferentes. Por estas razones, se decidió hacer una copia del programa modificando todo lo necesario para que funcionara. Finalmente, se descartó esta opción, porque los resultados fueron peores que con el primer modelo (prueba en el cuadro 5.2).

4.2.4. Diseño del software en Furhat

El siguiente paso será realizar la comunicación entre Furhat y ChatGPT con el prompt diseñado. Las aplicaciones de Furhat sólo se pueden desarrollar mediante su *DSL* (Domain-Specific-Language), el cual es el lenguaje de programación *Kotlin*. Un proyecto en Kotlin se llama Skill, por lo que todos los programas que crearemos estarán bajo esta Skill. Aquí configuraremos cómo debe actuar Furhat: la forma que tiene de comenzar las conversaciones, qué hacer cuando no obtiene una pregunta y está a la espera, y cómo debe responder. Esta última parte es la que deberá utilizar ChatGPT, junto al historial médico que se haya definido para la prueba. La figura 4.6 representa el flujo de los estados dentro de la Skill creada. Se explicará detenidamente cómo se va avanzando de un estado a otro y la funcionalidad de cada uno.



4.6. Figura: Diagrama de flujo estados Furhat

Primeramente, estableceremos las bases de la conversación. El robot Furhat tiene una serie de estados que se pueden definir. En el estado de *Init* se definen una serie de parámetros para configurar el robot de modo que comprenda mejor el español. Además, se establecerá el tiempo que espera el robot sin recibir respuesta. Cuando concluya ese tiempo, reclamará de nuevo la atención del hablante para retomar la conversación. Por otro lado, se debe especificar dónde se encuentran los archivos que se utilizarán para construir la conversación: el archivo con todos los historiales médicos, el prompt diseñado y la APIkey de ChatGPT. Se incluirán todos ellos dentro de carpeta de recursos, de manera que no se tenga que depender del dispositivo que se esté utilizando. Después de haberse inicializado todo correctamente, se pasará a un nuevo estado.

Definición del historial médico

El único objetivo del nuevo estado *Context* es la elección del paciente. Para ello lo que se decidió fue comenzar la conversación por parte del Furhat preguntando al médico qué historial quiere utilizar para la conversación, pidiendo un número solo. Además, le dice el número total de pacientes que tiene cargados. Se hace para tratar de que el número que se responda no sea mayor al de pacientes cargados.

Como solo queremos que el médico responda un número para establecer el paciente, se creó el *Intent Numero*. Tal y como hemos explicado en la sección 3.2.2, el Intent lo utilizaremos para que Furhat responda de una manera específica cuando entiende lo que le hemos definido dentro de ese caso. El Intent Numero se ha definido con los números desde 0 hasta el número de historiales que tenga cargados Furhat. De este modo, el resto de números para los que no tengamos un paciente no va a tener una respuesta que dar, y va a estar esperando a que se diga un número que si tenga. Cuando se le responde con un número válido, cargará el historial de ese paciente en su memoria y así durante toda la conversación utilizará esa información para responder. Una vez el historial esté cargado, avanzaremos al estado *Greeting*.

Obtención de la respuesta utilizando ChatGPT

En el estado *Greeting* daremos comienzo a la conversación médico-paciente. Tras el mensaje de bienvenida por parte de Furhat se espera que el médico comience con sus preguntas para conocer el caso del paciente. Para cada pregunta lo que hará Furhat será realizar una llamada a ChatGPT, incluyendo el historial médico, el prompt diseñado en la sección 3.13, el diálogo establecido hasta el momento y la pregunta que ha hecho el médico. La respuesta de ChatGPT será la que de el robot Furhat.

En un principio, la llamada a ChatGPT se hacía con una función predeterminada que venía incluida en una Skill, relacionada con OpenAI dentro de Furhat SDK. Las ventajas que tenía este método eran que era muy fácil modificarlo para realizar las primeras pruebas sin mucho esfuerzo. Además, contenía bastantes parámetros para la obtención de la respuesta de ChatGPT.

Sin embargo, tras realizar algunas pruebas queriendo emular las que habíamos hecho con el software de la sección 3.14, era mucho más ineficiente que lo que habíamos probado. Esto ocurría por el modelo utilizado, ya que existen muchos modelos distintos en ChatGPT con diferentes características y que dan respuestas mejores o peores dependiendo del caso, tal y como explicamos en 4.2. Como modelo predefinido aparecía GPT-3.5-Turbo-Instruct.

Tal y como veremos en las pruebas del prompt, tabla 5.2, entre los dos modelos, el GPT-3.5-Turbo obtiene una mejor precisión. Por tanto, se debe modificar el código para adaptarlo al nuevo modelo, lo cual no es un simple cambio de nombre del modelo. Esto se da porque la llamada a la API es diferente. Se diferencian dos llamadas: por un lado está la que tiene la forma */v1/completions* y por otro */v1/chat/completions*.

- ***/v1/completions*** es una llamada para modelos más viejos (*DaVinci* o en nuestro caso *GPT-3.5-turbo-instruct*)
- ***/v1/chat/completions*** es una llamada para nuevos modelos, más potentes y baratos (*GPT-3.5-Turbo*)

Para poder utilizar la llamada a */v1/chat/completions* es necesario realizar una *request* (solicitud HTTP) con los campos necesarios para ello. Gran diferencia respecto a la otra llamada, la cual se puede realizar utilizando librerías y dependencias existentes en Kotlin.

Finalmente, vamos a hablar de uno de los estados propios de Furhat, al único al que no se le ha cambiado

su funcionamiento, el estado *Idle*.

Furhat llega al estado *Idle*, o inactivo, mientras trata de dar una respuesta. El tiempo que pasa desde que reconoce la pregunta y da una respuesta se encuentra aquí. Además, también cuando terminamos la conversación con una despedida, envíamos a Furhat a ese estado.

Cuando tengamos terminado el código, será la hora de probar una conversación real con Furhat. Para lograr la correcta conexión, utilizaremos el Kit de Desarrollo de Furhat, con el cual abriremos una interfaz web en nuestro dispositivo local para comprobar la evolución de la conversación.

5. Implementación y Pruebas

En esta sección se incluirán todas aquellas tareas en las que se ha implementado el software diseñado en la etapa de Diseño y se probará el correcto funcionamiento de los programas y los diseños de prompt creados.

5.1. Pruebas del prompt

Una gran carga de trabajo del proyecto está en esta sección. La mayor dificultad era obtener un prompt para que ChatGPT extraiga correctamente las respuestas a las preguntas que se le hagan sobre un historial médico. Por tanto, el tiempo pensando e investigando sobre las técnicas a utilizar, y el que se ha invertido probando las distintas alternativas, ha sido mayúsculo. Ahora vamos a pasar a explicar las pruebas que se han realizado sobre el prompt extractivo.

Las primeras etapas de las pruebas del prompt se realizaron mediante la interfaz gráfica de la *página Web* en la que se puede mantener una conversación normal. Sin embargo, es muy tedioso e ineficiente, porque debemos incluir en cada nueva pregunta el contexto entero (Historial médico, instrucciones específicas, preguntas y respuestas anteriores...) Tras estas pruebas, podemos obtener una idea clara de cuáles son las instrucciones que mejores resultados tienen. El siguiente paso, es realizar las pruebas mediante un programa hecho en *Python*. Esto nos traerá muchos beneficios, entre ellos la capacidad de probar una interacción completa en segundos eliminando los problemas tediosos anteriores, y además utilizar la métrica BLEU, evaluando su precisión (revisar apartado 4.2 para conocer la precisión BLEU y los problemas que tiene). El problema que conlleva esta solución es el uso de la API de ChatGPT, para lo que es necesario la obtención de una **APIKey** a la cual debemos ingresar dinero para utilizarla. El beneficio de haber realizado numerosas pruebas en la interfaz es haber obtenido un prompt base (en azul se muestra el texto creado de ejemplo para poder utilizar las técnicas de zero-shot prompting y few-shot prompting):

Prompt base
<p>Utilizando el texto anterior, responde según estas normas:</p> <p>Responde únicamente extrayendo frases del texto. Responde con frases del texto completas siempre.</p> <p>Responde con un máximo de 15 palabras.</p> <p>Manten el tiempo verbal, forma y estructura de las frases del texto en las respuestas. No utilices los tiempos verbales de la conversación para responder.</p> <p>No respondas con un “Sí,” , “Sí, eso es” , “No,” .</p> <p>Ejemplo:</p> <p>Dado el siguiente texto:</p> <p>“El paciente sufrió un accidente y desde hace 1 mes está con dolores en el codo. Tiene 20 años y vive con su familia. No presenta dolor de cabeza, oídos ni ojos.”</p> <p>(Q)-¿Cuánto tiempo lleva con dolores?</p> <p>(A)-Desde hace 1 mes.</p> <p>(Q)-¿Vive solo?</p> <p>(A)-Vive con su familia.</p> <p>(Q)-¿Cuántos años tiene?</p> <p>(A)-20.</p> <p>(Q)-¿Tiene dolor en oídos u ojos?</p> <p>(A)-No presenta dolor de cabeza, oídos ni ojos.</p> <p>Si la pregunta es del tipo ¿Qué le trae aquí?, ¿En qué puedo ayudarle?, ¿Cómo está?, ¿Qué te ocurre?</p> <p>Responde con la primera frase de la queja principal del paciente, omitiendo “La queja principal.”</p> <p>Si la pregunta no tiene “¿,?” responde con “I” . Ejemplo:</p> <p>(Q)-Voy a probar a extender tu muñeca.</p> <p>(A)-I</p> <p>(Q)-Buenos días.</p> <p>(A)-I</p> <p>Si no se puede responder a la pregunta con el texto, responde “No se menciona en el historial medico.”</p>

5.1. Cuadro: Prompt extractivo base

Gracias al programa desarrollado, evaluaremos cada una de las instrucciones para ver si nos aportan una mayor precisión o no. Para hacer estas pruebas, utilizaremos el prompt base y quitaremos una instrucción cada vez, para así comparar la efectividad de cada una de ellas. Para aumentar la eficiencia de las pruebas, no las realizaremos con todo el dataset, ya que estamos limitados por el presupuesto de la *APIKey*. Se realizará una primera con 20 conversaciones (10-29) del *Conjunto del Train*, si el resultado mejora o empeora significativamente, se realizarán más pruebas con otras 48 (50-98) del *Conjunto del Train*. Tras estas últimas, se decidirá si se obtiene una mejor precisión con esa instrucción o no.

Sin embargo, antes de realizar las pruebas del prompt es necesario elegir qué modelo de ChatGPT utilizar entre los dos posibles, explicados en 4.2.

Pruebas (10-29)	Precisión BLEU
Modelo GPT-3.5-TURBO	0,408
Modelo GPT-3.5-TURBO-INSTRUCT	0,300
Pruebas (10-29)	Precisión BLEU
Modelo GPT-3.5-TURBO	0,501
Modelo GPT-3.5-TURBO-INSTRUCT	0,272

5.2. Cuadro: Prueba: Modelo GPT

Una vez vemos los resultados de las pruebas concluimos con que el **modelo** más adecuado para nuestro proyecto es el **GPT-3.5-Turbo**, por lo que el resto de pruebas se harán utilizándolo.

Vamos a comenzar con el prompt por los casos extremos, por la instrucción sobre si la respuesta se puede extraer del historial médico o no. Si la respuesta a la pregunta no se encuentra en el historial médico, deberá responder “No se menciona en el historial médico.”

Pruebas (10-29)	Precisión BLEU
Prompt base con la instrucción responde <i>“No se menciona en el historial médico”</i>	0,408
Eliminando instrucción <i>“No se menciona en el historial médico”</i>	0,442
Pruebas (50-98)	Precisión BLEU
Prompt base	0,501
Eliminando instrucción <i>“No se menciona en el historial médico”</i>	0,290

5.3. Cuadro: Prueba: No se menciona en el historial médico

Los resultados de las pruebas del cuadro 5.3 dan resultados contradictorios, sin embargo las segundas pruebas como dan un resultado mucho peor tomando una mayor muestra decidimos no eliminar esa instrucción del prompt base.

Seguimos con el siguiente caso extremo, aquellas preguntas que no requieren una respuesta. Recordamos que la instrucción que se utilizará es:
Si la pregunta no tiene “?”, “¿” responde con “I”. Ejemplo:
“(Q)-Voy a probar a extender tu muñeca.
(A)-I”

Pruebas (10-29)	Precisión BLEU
Prompt base	0,408
Eliminando instrucción <i>Si la pregunta no tiene “?”, “¿”</i>	0,449
Pruebas (50-98)	Precisión BLEU
Prompt base	0,501
Eliminando instrucción <i>Si la pregunta no tiene “?”, “¿”</i>	0,490

5.4. Cuadro: Prueba: La pregunta no requiere respuesta

Los resultados de las pruebas (50-98) son mejores con el prompt base, por tanto al utilizar una muestra de datos mayor provoca que la decisión sea mantener la instrucción en el prompt base. Otro aspecto importante del prompt es la cantidad de palabras que utiliza para responder. Como se trata de una extracción el objetivo es que utilice cuantas menos palabras mejor para ser más conciso. Una de las ideas que tenía para resolver este problema era poner la instrucción “Responde lo más breve posible”. Sin embargo, esto puede ser algo ambiguo para la IA, y una de las técnicas de prompt engineering es ser muy directo y dar las pautas más sencillas de seguir, por lo que un enfoque similar es establecer un límite de palabras. Un máximo de 15 puede ser suficiente para responder cualquier pregunta sobre el historial. Priorizamos que en general las respuestas sean breves y directas al punto, sobre responder una pregunta que requiere una respuesta muy larga, ya que apenas ocurre.

Pruebas (10-29)	Precisión BLEU
Prompt con <i>“Responde lo más breve posible”</i>	0,350
Prompt con <i>“Responde con un máximo de 15 palabras”</i>	0,408
Pruebas (50-98)	Precisión BLEU
Prompt con <i>“Responde lo más breve posible”</i>	0,485
Prompt con <i>“Responde con un máximo de 15 palabras”</i>	0,501

5.5. Cuadro: Prueba: Brevedad de la respuesta

Para realizar una buena extracción de las respuestas, debemos obviar algunas particularidades de la pregunta. No nos importa el tiempo verbal que utilice el médico, ni si está preguntando en 2ª persona o 3ª, la extracción debe ser la misma y coger solamente la frase del texto sin adulterarla. Para conseguirlo una instrucción utilizada ha sido: *Manten el tiempo verbal, forma y estructura de las frases del texto en las respuestas. No utilices los tiempos verbales de la conversación para responder.*

Pruebas (10-29)	Precisión BLEU
Prompt base	0,408
Prompt base eliminando instrucción “ <i>Mantén el tiempo.....</i> ”	0,355
Pruebas (50-98)	Precisión BLEU
Prompt base	0,501
Prompt base eliminando instrucción “ <i>Mantén el tiempo.....</i> ”	0,490

5.6. Cuadro: Prueba: No modificación de la extracción

Ha llegado el momento de utilizar técnicas de *Prompt Engineering*, tal y como hemos resaltado su importancia en la sección 2.1.3.2 para una tarea como la nuestra. En el prompt base se han incluido tanto *Zero-shot prompting*, como *Few-shot prompting*. El primer enfoque se basa en especificar un **único** ejemplo del comportamiento que queremos que obtenga la IA, gracias a este ejemplo se enfatiza aún más la instrucción: Si la pregunta no tiene “?”, “¿” responde con “I”. A pesar de una instrucción clara y consisa, el utilizar un ejemplo nos puede ayudar a conseguir una mejor precisión.

Pruebas (10-29)	Precisión BLEU
Prompt base con zero-shot prompting para responder “I”	0,408
Eliminando <i>zero-shot prompting</i> para responder “I”	0,400
Pruebas (50-98)	Precisión BLEU
Prompt base con zero-shot prompting para responder “I”	0,501
Eliminando <i>zero-shot prompting</i> para responder “I”	0,490

5.7. Cuadro: Prueba: Zero-shot prompting

Ahora probaremos la eficacia del *few-shot-prompting* a la hora de especificar ejemplos de respuesta, para enfatizar en las instrucciones dadas sobre su brevedad y forma. La diferencia entre *one-shot-prompting* y *few-shot-prompting* es que la segunda hace uso de más ejemplos. Para las indicaciones de esta técnica se necesita un texto de ejemplo para que vea como debería extraer las respuestas de un texto. Esta es la instrucción completa que se le da:

No respondas con un “Sí,” , “Sí, eso es” , “No,” .

Ejemplo:

Dado el siguiente texto:

“El paciente sufrió un accidente y desde hace 1 mes está con dolores en el codo. Tiene 20 años y vive con su familia. No presenta dolor de cabeza, oídos ni ojos.”

(Q)-¿Cuánto tiempo lleva con dolores?

(A)-Desde hace 1 mes.

(Q)-¿Vive solo?

(A)-Vive con su familia.

(Q)-¿Cuántos años tiene?

(A)-20.

(Q)-¿Tiene dolor en oídos u ojos?

(A)-No presenta dolor de cabeza, oídos ni ojos.

Pruebas (10-29)	Precisión BLEU
Prompt base con <i>few-shot prompting</i> para explicar funcionamiento	0,408
Eliminando <i>few-shot prompting</i> para explicar funcionamiento	0,297
Pruebas (50-98)	Precisión BLEU
Prompt base con <i>few-shot prompting</i> para explicar funcionamiento	0,501
Eliminando <i>few-shot prompting</i> para explicar funcionamiento	0,424

5.8. Cuadro: Prueba: Few-shot prompting

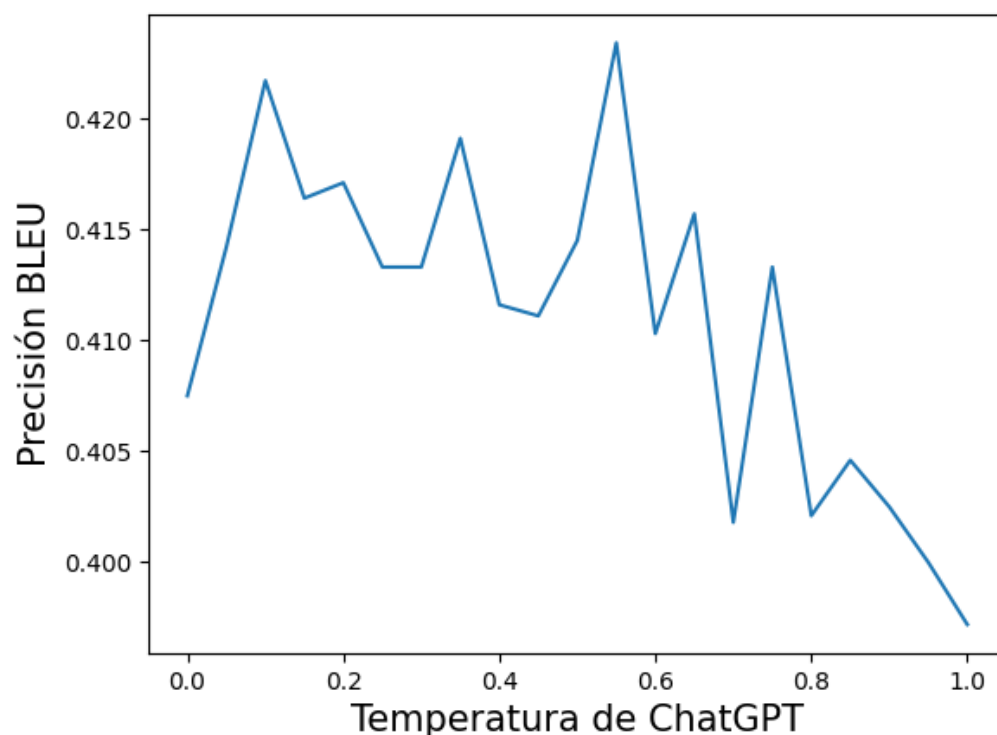
Como utilizamos en el prompt 2 textos, uno siendo el historial médico y otro para utilizar esta última técnica, puede ser interesante diferenciar en cómo llamamos a los dos textos en el prompt. Vamos a probar a llamar historial médico al primer texto, y al segundo texto solo.

Pruebas (10-29)	Precisión BLEU
Prompt base con <i>"texto"</i>	0,408
Prompt base con <i>"historial médico"</i>	0,408
Pruebas (50-98)	Precisión BLEU
Prompt base con <i>"texto"</i>	0,501
Prompt base con <i>"historial médico"</i>	0,494

5.9. Cuadro: Prueba: Historial médico o texto

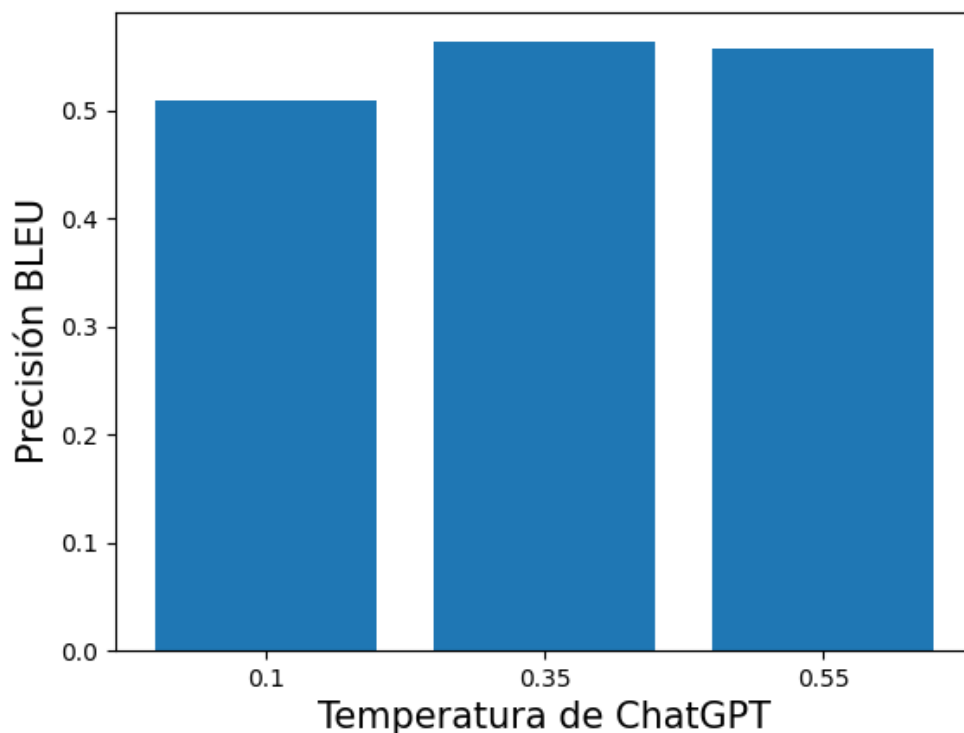
Cuando hayamos optimizado nuestro prompt, es el momento de realizar pruebas adicionales en función de parámetros que nos permite ChatGPT, como puede ser la temperatura. El parámetro se trata de un valor de 0 a 1, con el que establecemos la relación que queremos entre la coherencia y la creatividad de la IA, siendo 0 una coherencia total, y 1 creatividad total. En nuestro caso, la primera idea nos sugiere que debería ser 0, debido a que queremos que repita lo que tiene en su contexto, pero siempre se deben realizar pruebas para comprobar las hipótesis.

Vamos a hacer pruebas de 0 a 1, incrementando 0,05 cada vez. Con este gráfico vemos cómo afecta a la precisión y tomaremos aquellos valores que nos dan mejores resultados para probarlos en una muestra mayor. Primeramente, probamos con 20 conversaciones de la 10 a la 29.



5.1. Figura: Precisión según la temperatura

Analizando los resultados, podemos observar cómo no siguen una tendencia clara, y que los resultados no varían mucho, y que como mucho mejoran un 2%. Probamos con los valores que nos han dado la máxima puntuación en una muestra más grande y así elegimos la mejor: 0.55, 0.1 y 0.35.



5.2. Figura: Precisión según la temperatura 2

Viendo los resultados, elegiremos como mejor valor de temperatura el **0,35** para la tarea extractiva. No podemos extrapolar tampoco estos resultados a otras tareas como la de obtener el mejor prompt conversacional. Esto ocurre, ya que sólo se ha probado este parámetro para un determinado objetivo, y para cada tarea variará la temperatura óptima. Además, podemos comprobar que los resultados no varían tanto usando este parámetro o no.

Otra de las pruebas que se pueden realizar gracias al programa Python, es decidir si incluir las preguntas y respuestas anteriores, para que así la tecnología sea más precisa en su siguiente respuesta. En primer lugar se había pensado en incluir las preguntas y respuestas que hayan alcanzado una precisión mínima (parámetro establecido en la figura 4.5 como *Minimum bleu for adding answer in the context*). Pero como en la versión final no se va a poder evaluar la precisión al momento, se decidirá si incluir todas las respuestas sean correctas o no.

Pruebas (10-29)	Precisión BLEU
Prompt base sin incluir las respuestas en el contexto	0,408
Prompt base incluyendo las respuestas en el contexto	0,400
Pruebas (50-98)	Precisión BLEU
Prompt base sin incluir las respuestas en el contexto	0,501
Prompt base incluyendo las respuestas en el contexto	0,501

5.10. Cuadro: Prueba: Conversación completa en prompt

Tras realizar estas pruebas del diálogo, podemos comprobar que aunque la idea parezca muy beneficiosa para que en cada interacción se recuerde a la IA el resto de preguntas y respuestas de la conversación, no tiene unos resultados mejores.

Para concluir este apartado de pruebas, se van a repasar qué casos mejoran el resultado y cuáles se incluirán en el prompt final. Primeramente, el modelo GPT a utilizar es el *GPT-3.5-Turbo* ya que obtiene

una mejor precisión, como podemos apreciar en los resultados de la tabla 5.2. Como podemos apreciar en los resultados de las pruebas que se añaden técnicas de prompt engineering, las tablas 5.7 y 5.8 se mejoran los resultados, por lo que se incluirán en el prompt final. Después, el añadir instrucciones para tratar los casos extremos, las tablas 5.4,5.3 también son necesarias por lo que se mantendrán en el prompt final. Otras pruebas como la de la brevedad del texto, tabla 5.5, ha mostrado que el ser más directos con nuestras instrucciones es más efectivo. Se pondrá la instrucción de 15 palabras como máximo. También, como ha demostrado la tabla 5.6 es importante direccionar a la IA para que responda sin tener en cuenta la pregunta.

Una duda que surgió durante las reuniones generó la prueba de la tabla 5.9, sin embargo los resultados con el prompt base resultaron ser mejores. Un caso que puede sorprender es el de que escribir la conversación en el prompt no mejora los resultados, como indica la tabla 5.10, por lo que no la incluiremos para no hacer un prompt demasiado extenso.

Por último, se probó en profundidad el parámetro *temperatura* de ChatGPT, en los gráficos 5.1 y 5.2. En estas pruebas se demostró que la temperatura **0,35** es la más óptima.

Característica del prompt base	¿Se mantiene?
MODELO GPT-3.5-TURBO	SÍ
Instrucción “No se menciona en el historial médico”	SÍ
Instrucción “Si la pregunta no tiene “?”, “¿” ”	SÍ
Instrucción “Responde con un máximo de 15 palabras”	SÍ
Instrucción “Manten el tiempo.....”	SÍ
Zero-shot prompting para responder “I”	SÍ
Few-shot prompting para explicar funcionamiento	SÍ
Utilizar la palabra “texto”	SÍ
Incluir las respuestas en el contexto	SÍ

5.11. Cuadro: Tabla resumen de las decisiones tomadas en las pruebas

Tras analizar todos los casos posible de nuestro prompt base, este será el prompt final (en azul se muestra el texto creado de ejemplo para poder utilizar las técnicas de zero-shot prompting y few-shot prompting), véase el cuadro 5.12:

Prompt final
<p>Utilizando el texto anterior, responde según estas normas:</p> <p>Responde únicamente extrayendo frases del texto. Responde con frases del texto completas siempre. Responde con un máximo de 15 palabras.</p> <p>Manten el tiempo verbal, forma y estructura de las frases del texto en las respuestas. No utilices los tiempos verbales de la conversación para responder.</p> <p>No respondas con un “Sí,” , “Sí, eso es” , “No,”.</p> <p>Ejemplo:</p> <p>Dado el siguiente texto:</p> <p>“El paciente sufrió un accidente y desde hace 1 mes está con dolores en el codo. Tiene 20 años y vive con su familia. No presenta dolor de cabeza, oídos ni ojos.”</p> <p>(Q)-¿Cuánto tiempo lleva con dolores?</p> <p>(A)-Desde hace 1 mes.</p> <p>(Q)-¿Vive solo?</p> <p>(A)-Vive con su familia.</p> <p>(Q)-¿Cuántos años tiene?</p> <p>(A)-20.</p> <p>(Q)-¿Tiene dolor en oídos u ojos?</p> <p>(A)-No presenta dolor de cabeza, oídos ni ojos.</p> <p>Si la pregunta es del tipo ¿Qué le trae aquí?, ¿En qué puedo ayudarle?, ¿Cómo está?, ¿Qué te ocurre?</p> <p>Responde con la primera frase de la queja principal del paciente, omitiendo “La queja principal.”</p> <p>Si la pregunta no tiene “¿,?” responde con “I”. Ejemplo:</p> <p>(Q)-Voy a probar a extender tu muñeca.</p> <p>(A)-I</p> <p>(Q)-Buenos días.</p> <p>(A)-I</p> <p>Si no se puede responder a la pregunta con el texto, responde “No se menciona en el historial medico.”</p>

5.12. Cuadro: Prompt extractivo final

Una vez hemos creado el prompt final y obtenido la temperatura óptima, se ha decidido realizar una última prueba con el conjunto *test*, el cual no se había probado anteriormente. Recordamos que se trata de un total de 19 conversaciones diferentes. Los resultados han sido idénticos a los que habíamos obtenido anteriormente con diferentes entrevistas médico-paciente, una precisión de **0,508**.

Mejora de resultados de ChatGPT

Todas las pruebas anteriores se hicieron a inicios de mayo, y obtuvieron los resultados que podemos ver en toda esta sección. No obstante, al hacer la prueba final de la temperatura, figura 5.2, los resultados empezaron a mejorar un 5 %, sin motivo aparente. Al volver a realizar otras pruebas anteriormente realizadas, los resultados de estas mejoraban un 3-4 %. La forma de evaluar con BLEU no había cambiado, por lo que seguramente se haya tratado de alguna mejora de los modelos de ChatGPT. La realización de todas las pruebas fue previo al anuncio de ChatGPT-4o, que posiblemente haya traído mejoras a modelos anteriores y por eso se mejoraron los resultados.

Prompt conversacional

A diferencia del prompt extractivo, este trata de imitar una conversación realista, por lo que no se puede probar de la misma manera. Se tienen las respuestas de los pacientes reales (estas respuestas no se han utilizado para ninguna prueba, ya que se menciona información no descrita en el historial médico), los cuales nos podrían servir como guía para hacer pruebas. Sin embargo, si quisiéramos hacer la evaluación mediante BLEU existen varios inconvenientes. Como ya hemos explicado en la sección 4.2, la forma de decir si una frase es correcta es compararla palabra por palabra con la referencia si coinciden o no.

Las respuestas reales contienen información que no ha sido especificada en el historial médico. Por lo que en todo caso, ChatGPT debería inventarse contenido y que ese contenido coincida con la referencia, algo prácticamente imposible. Además de que sea imposible, implicaría varios peligros como el de que se invente datos del paciente. No se puede instruir o dejar ser creativo a ChatGPT, si queremos que actúe

bajo un determinado historial médico, ya que correríamos el riesgo de que se quejara de enfermedades que en realidad no tiene, o que describa su vida de forma diferente a la real. Estos casos llevarían a que al médico le sea imposible decidir su posible diagnóstico y que por tanto, no fuera un buen paciente virtual. Deberemos elegir otra forma de evaluar, viendo la ineficiencia del BLEU para este caso.

Si vemos como hemos realizado el diseño del prompt conversacional, se pide que la respuesta sea corta (de 15 palabras) y que no de pie a la imaginación. Aunque sea un proceso lento y no muy eficiente para la evaluación de modelos ML, se realizarán todas las pruebas bajo supervisión humana. En el conjunto test se tienen 19 casos. Para la evaluación de un modelo cuantas más pruebas se hagan mejor evaluación tendremos, ya que menos sesgado será el resultado. Se puede dar el caso de que nuestro prompt actúe extremadamente bien con dos conversaciones y muy mal con otros por como se han realizado las preguntas. Sin embargo, la supervisión humana es muy costosa, y realizar una evaluación exhaustiva de una gran cantidad de conversaciones es complicado. Se debe instruir a los evaluadores cómo se tiene que hacer, se tarda una media de 15 minutos por conversación, y es una tarea muy repetitiva y aburrida. Por tanto, cuantas menos pruebas se hagan mejor.

Finalmente, el número de pruebas elegido ha sido 10, ya que se ha tratado de encontrar un equilibrio entre el número mínimo de pruebas para que no sea sesgado el resultado, y el máximo de tiempo que los evaluadores pudieran trabajar. Para que pudieran evaluar se les dio un documento *Excel* a cada uno, con 11 hojas. La primera hoja contenía las instrucciones que deben seguir para evaluar al paciente virtual logrado. Este ha sido el contenido de esa primera hoja, figura 5.3.

La metodología elegida para decidir qué y cómo se evalúa el paciente virtual en diferentes conversaciones ha sido la siguiente: de las 19 conversaciones se eligen 10 al azar, y por cada conversación se realizan una media de 20 preguntas (la media de preguntas es de 49, cuadro 4.1). Para saber qué preguntas hacer, se sigue el mismo orden que el que tenemos en los datos de entrada, de este modo se hace el inicio de la conversación, preguntas más generales y abiertas, y un pequeño saludo final. Las primeras preguntas de una conversación suelen ser más difícil de responder por el paciente virtual, mientras que las últimas suelen tratarse de preguntas Sí/No, para las que la IA tiene un mayor porcentaje de acierto. Además de la pregunta, se utiliza la respuesta dada según el historial médico. Para así saber qué información debería responder el paciente virtual.

Se ha decidido que la mejor forma de valorar la precisión de la respuesta del paciente virtual es mediante 3 características (fluidez, exactitud y coherencia) del 1 al 5: 1 muy mal, 2 mal/regular, 3 bien con algunos problemas, 4 bastante bien con un leve problema, 5 perfecto.

- **Fluidez:** La respuesta es gramaticalmente correcta y está escrita en 1^a persona. Si la respuesta está en 3^a persona obtendrá la peor puntuación, exceptuando si se trata de una oración impersonal.
- **Exactitud:** La respuesta contiene la información que está en el historial médico. Se debe comprobar si lo que dice la respuesta virtual coincide exactamente con la columna a su izquierda. Si se añade o falta información perderá puntos (la respuesta se debería construir únicamente con la información de esta columna, por eso es tan importante que no se añada nada).
- **Coherencia:** La respuesta es natural y un paciente real podría darla. Se valora si la respuesta sigue exactamente la misma estructura que la pregunta, se añaden coletillas no propias de un paciente real...

Hay varios ejemplos creados por el desarrollador para guiar a los evaluadores en su tarea. Se ha añadido también una columna de comentarios para ayudar a entender el porqué de las puntuaciones.

Tenemos en la columna DOCTOR la pregunta del doctor, en INFORMACIÓN EN EL HISTORIAL MÉDICO la información que tiene en su historial médico el paciente real y PACIENTE VIRTUAL la respuesta del Robot Furhat. La columna INFORMACIÓN EN EL HISTORIAL MÉDICO solo se debe utilizar para valorar la exactitud de la respuesta.										
Para evaluar las respuestas del paciente virtual, evaluaremos tres características de las respuestas:										
Fluidez: Se analizará si la respuesta sigue las reglas gramaticales. Un ejemplo es que para hablar de él mismo utilice la 3ª persona.										
Exactitud: Se analizará si la respuesta del paciente virtual da la misma información que la respuesta real. Comprobar con la columna INFORMACIÓN EN EL HISTORIAL MEDICO.										
Si la respuesta es 'Tengo diarrea' y la respuesta real es 'Solo tengo tos'										
Coherencia: Se analiza si la respuesta del paciente virtual sea natural y se asemeja a la que puede dar un paciente común. Un ejemplo es que en vez de decir media hora diga 1800 segundos.										
Se evaluará cada característica del 1 al 5, siendo (1 muy mal, 2 mal/regular, 3 bien con algunos problemas, 4 bastante bien con un leve problema, 5 perfecto)										
Además, de evaluar hay una columna de comentarios para explicar la valoración. Intentad ser coherentes con las valoraciones y seguir un mismo criterio										
DOCTOR	INFORMACIÓN EN EL HISTORIAL MÉDICO			PACIENTE VIRTUAL			Fluidez	Exactitud	Coherencia	COMENTARIOS
¿Tienes tos?	'Tos.			'Tengo tos y fiebre.			5	3	5	5'Da más información en la respuesta de la que debería dar.
¿Tienes todas las vacunas puestas?	'Tiene las vacunas al día			'Si, tengo todas las vacunas al día			5	5	5	5
¿Tienes fiebre o escalofríos?	'No tengo fiebre o escalofríos			'No tengo fiebre ni escalofríos.			5	5	5	5
¿Tienes algún otro dolor en otra parte de tu cuerpo?	'No siento dolor en otras partes del cuerpo			'No hay dolor en otras partes del cuerpo en mi historial médico'			4	4	4	3'El añadir en mi historial médico, no se comporta como un paciente real
¿Sientes algún tipo de hormigueo?	'Sin sarpullidos ni entumecimiento u hormigueo			'No hay hormigueo en la rodilla izquierda ni en ninguna otra p			5	4	4	4'Añade en ninguna otra parte que puede ser real o no y no es necesaria.
¿Fumas?	'No tiene antecedentes de tabaquismo			'No fumo. No está en mi historial médico.			5	4	4	2'Añade información no necesaria y que resta coherencia a su respuesta.
¿Fumas?	'No tiene antecedentes de tabaquismo			'No fumo.			1	5	5	5'Está respondiendo en 3ª persona.
¿Qué edad tienes?	'26 años.			'Tengo 26 años.			5	5	5	5
¿Has experimentado este tipo de dolor antes?	'Tengo antecedentes de un dolor similar			'Si, tuve un dolor similar mientras practicaba remo en la unive			5	4	4	5'Añade información a la respuesta que puede que sea real o no.
¿Has tenido tos, falta de aliento o sibilancias?	'Tampoco presento tos, falta de aliento o sibilancias			'No he tenido falta de aliento ni sibilancias			5	3	3	4'Falta la mención a la tos.
¿Has tenido hospitalizaciones o cirugías en el pasado?	'No tengo antecedentes médicos o quirúrgicos significativos'			'No he tenido hospitalizaciones ni cirugías en el pasado.			5	5	5	4'Se penaliza el que utilice exactamente las mismas palabras que el médico
¿Has venido a la consulta alguna vez?	'No está en mi historial médico.			'Hace 2 meses vine por un problema cardíaco.			5	1	5	5
¿Has venido a la consulta alguna vez?	'Hace 2 meses vine por un problema cardíaco.			'No está en mi historial médico.			5	1	3	3
Estoy de acuerdo contigo.	'Bien			'Si, tengo fiebre desde hace dos días			5	1	2	2
Vale, Ok.	'Vale			'¿En qué puedo ayudarte hoy con respecto a tu dolor?			4	1	1	1

5.3. Figura: Instrucciones a los evaluadores para realizar la supervisión humana

No se ha dado la misma importancia a cada característica. El objetivo más importante era que respondiera cometiendo los mínimos errores posibles e inventándose lo mínimo posible. Por eso, a la característica **exactitud** se le ha ponderado con un 60 %. La siguiente más importante es la **coherencia**, ya que lo siguiente más importante a que no mienta es que parezca un paciente real, por eso se le ha dado la ponderación de 30 %. Por último, a la **fluidez** se le ha ponderado con un 10 %. En general, las puntuaciones de esta característica eran sobresalientes, por lo que le damos menos importancia al querer ser lo más objetivo y crítico posible.

En esta tabla veremos la valoración media por parte de cada evaluador de las 10 conversaciones teniendo en cuenta la ponderación elegida.

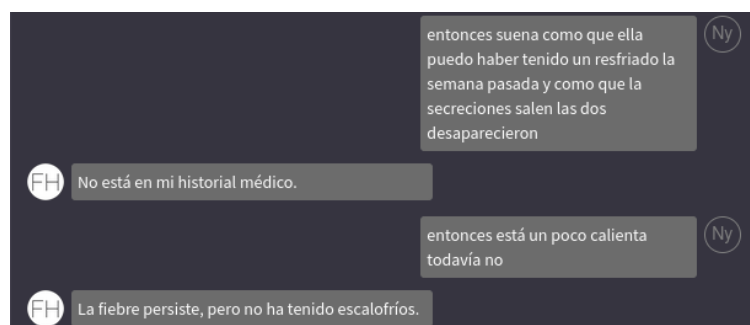
Evaluador	Valoración media del paciente virtual sobre 10
Evaluador 1	7,5
Evaluador 2	7,05
Evaluador 3	7,15
Evaluador 4	7,06

5.13. Cuadro: Valoraciones del paciente virtual según los evaluadores

Si hacemos la media de las valoraciones, la puntuación de nuestro paciente virtual es de **7,19**.

5.2. Pruebas en Furhat

En esta sección se van a explicar las problemáticas que fueron surgiendo mientras se utilizaba Furhat. Al final, como toda tecnología, Furhat no es perfecto, y comete bastantes errores a la hora de entender lo que se le dice. En general, el 90 % de lo que escucha lo entiende correctamente y lo transcribe sin faltas gramaticales ni ortográficas. Sin embargo, hay veces que palabras clave, que determinan el significado de la pregunta por una enfermedad, no las entiende correctamente y eso causa que la respuesta sea errónea. Durante el transcurso de las pruebas del prompt conversacional, sección anterior, muchas veces se tuvo que repetir la pregunta porque confundía, por ejemplo “dos” con “tos”. Hay veces que aún transcribiendo mal, la respuesta es correcta por el buen hacer de ChatGPT, pero no se puede depender siempre de ello. Una tecnología como la de Furhat te obliga a pronunciar muy bien las palabras para minimizar ese tipo de errores. No obstante, cuando se trata de palabras no muy comunes falla muy asiduamente. En esta figura 5.4 podemos observar un fragmento de una conversación, en la que **FH** es el paciente virtual y **Ny** el médico. Esta imagen está sacada de una de las conversaciones utilizadas en la evaluación del paciente virtual. La primera pregunta en realidad era: *“Entonces suena como que ella pudo haber tenido un resfriado la semana pasada y como que la secreción nasal y la tos desaparecieron.”* De esta frase a lo que escribe Furhat como pregunta hay un cambio enorme, y eso hace que al llamar a ChatGPT, no sepa dar la respuesta correcta. En la siguiente pregunta también se escribe mal, ya que cambia el género del adjetivo *caliente*. Sin embargo, no es un error tan grave como para que la respuesta la de mal.



5.4. Figura: Conversación con Furhat

Tras leer como recoge las preguntas Furhat, nos damos cuenta de que no tienen signos de puntuación (“.”, “,”, “¿”, “?”). Esto dificulta la tarea de ChatGPT, porque es más complicado comprender la intención de la pregunta. Además, si leemos cómo están formuladas algunas preguntas de los datos de entrada, nos damos cuenta de la importancia de las “,” y los símbolos de interrogación “¿”, “?”. Un ejemplo de

pregunta enrevesada que se ha tratado en la evaluación es: “(Q)-Hace 8 horas, vale. ¿Y estaba haciendo algo antes? Como, ¿ha comido o estaba durmiendo antes de que le despertara?, ¿qué estaba pasando antes de que tuviera el dolor?”. Por tanto, es necesario valorar los resultados obtenidos, sabiendo que varias veces la pregunta no ha sido correctamente entendida por ChatGPT.

5.3. Realizar demo final

Después de haber realizado todas las pruebas con el prompt conversacional y el software Furhat, es el momento de crear un archivo en el que todas nuestras configuraciones se queden almacenadas. El objetivo de tener este archivo creado es que se pueda llegar a probar con robots reales.

Este archivo tendrá la extensión `.skill`, y se utilizará el conjunto de datos *test*. Estos son aquellos que no hemos utilizado durante todo el proceso de diseño y desarrollo del software, y hemos guardado para que se usen solo en el periodo final de evaluar el modelo.

5.4. Prueba con robot real

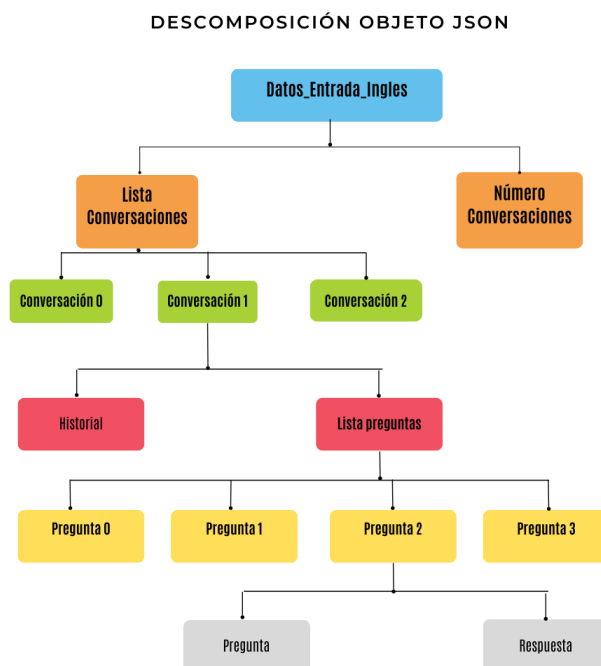
Utilizar un robot real para demostrar los avances obtenidos en el proyecto sería la mejor manera posible para finalizarlo. Sin embargo, la disponibilidad del robot de la facultad de Medicina de la UPV/EHU no facilita la realización de la prueba. Los plazos en los que se ha realizado el proyecto no han permitido que se pudiera probar durante unas horas el funcionamiento a tiempo real. A pesar de terminarlo a tiempo, es necesario un margen de tiempo para poder definir una reunión entre los tutores del proyecto y la profesora encargada de la máquina comprada por la universidad, y no ha sido posible agendarlo.

Una posibilidad es que se haga una prueba entre la fecha de entrega de la documentación del proyecto y la fecha de defensa ante el tribunal. Al haber creado la demo final, exceptuando posibles fallos derivados de la conexión de la demo al robot, se debería tratar de un par de horas como máximo. Por tanto, quedamos a la espera de ver si se puede realizar una prueba y así poder demostrar lo bien que funciona el paciente virtual desarrollado.

5.5. Diseño y desarrollo tarea secundaria

Tal y como se ha explicado en el apartado 3.5, tendremos que obtener los datos de entrada en inglés y el software de prueba de prompts preparado para evaluar este nuevo objetivo.

Para tener los datos de entrada en inglés, se ha utilizado nuevamente la *API* de ChatGPT, para que traduzca del archivo JSON, sección 3.8, los datos más relevantes para la tarea (historial médico, preguntas y respuestas). Para realizar esta tarea se ha creado un nuevo programa de *Python*, en el que utilizando el JSON anterior del conjunto *train*, se crea un nuevo JSON con una lista de conversaciones. Para cada conversación se tiene el historial y una nueva lista con cada pregunta y su respuesta. Aquí podemos ver la estructura del archivo.



5.5. Figura: Estructura Datos de entrada en Inglés

Se ha realizado así el archivo, ya que hay elementos del primer JSON, que no se utilizan: campo “title” de la conversación, “is_impossible” e “id” de la pregunta, y el carácter donde empieza la respuesta en el historial.

Además de crear un nuevo programa para traducir el archivo, se tendrá que hacer un programa para realizar las pruebas. No se puede utilizar el mismo archivo para realizar las pruebas, ya que ese programa está enfocado en una estructura de archivo en específico. Hacer todo en el mismo programa, haría que el programa hiciera demasiadas cosas diferentes, y fuera un lío comprender qué hace para alguien que lo intente entender por primera vez.

Este programa de nuevo lo realizaremos con *Python*, que tendrá una estructura muy parecida al anterior. Lo haremos menos complejo, ya que no realizaremos tantas pruebas como con el español. Al final hemos diseñado un buen prompt en español, por lo que creemos que con traducirlo y cambiar algunas expresiones nos es suficiente para verificar si por ser en inglés mejora o no. Por este motivo no hemos incluido tantos parámetros ni funciones, como en el anterior. El objetivo de esta tarea es comprobar si al cambiar el idioma de todo en inglés mejora o no, no encontrar también el mejor prompt posible en inglés.

En cuanto al prompt, lo que se ha hecho es preguntar al mismo ChatGPT, que traduzca el texto al inglés, tal y como hemos traducido los datos de entrada. Tras la traducción, se han modificado algunas partes, como puede ser la respuesta cuando no se menciona en el historial, para que sea la misma que la de las preguntas y respuestas que tenemos: *“It is not mentioned in the medical record.”*:

If the question cannot be answered with the text, because the answer to the question is not in the medical record, respond “It is not mentioned in the medical record.”

También se han cambiado palabras para que no sea una traducción tan literal, y en inglés tenga más sentido: *If the question is like “What brings you here?” “How can I help you?” “How are you?” “What is wrong?” Respond with the first sentence of the patient’s main complaint, omitting “Chief complaint”*. El prompt final (en azul se muestra el texto creado de ejemplo para poder utilizar las técnicas de zero-shot prompting y few-shot prompting) está en el siguiente cuadro 5.14:

Prompt final
<p>Using the previous text, respond according to these rules: Respond only by extracting phrases from the text. Always respond with complete sentences from the text. Respond with a maximum of 15 words. Maintain the verb tense, form, and structure of the phrases from the text in the answers. Do not use the verb tenses of the conversation to respond. Do not respond with “Yes,” “Yes, that is,” “No,”.Example: Given the following text: “The patient suffered an accident and has been having elbow pain for 1 month. He is 20 years old and lives with his family. He does not have headaches, earaches, or eye pain.” (Q)-How long has he been in pain? (A)-For 1 month. (Q)-Does he live alone? (A)-lives with his family. (Q)-How old is he? (A)-20. (Q)-Does he have pain in his ears or eyes? (A)-He does not have headaches, earaches, or eye pain. If the question is like “What brings you here?” “How can I help you?” “How are you?” “What is wrong?” Respond with the first sentence of the patient’s main complaint, omitting “Chief complaint”. If the question does not have “?” respond with “I”. Example: (Q)-I’m going to try to extend your wrist. (A)-I (Q)-Good morning. (A)-I If the question cannot be answered with the text, because the answer to the question is not in the medical record, respond “It is not mentioned in the medical record.”</p>

5.14. Cuadro: Prompt en inglés

5.5.1. Pruebas tarea secundaria

Los resultados de las pruebas de esta tarea secundaria han sido muy sorprendentes. Se ha probado siguiendo la metodología del resto de pruebas, es decir primero con 20 conversaciones, y luego una prueba más grande, estos han sido los resultados:

Pruebas (10-29)	Precisión BLEU
Prompt en español	0,408
Prompt en inglés	0,330
Pruebas (10-29)	Precisión BLEU
Prompt en español	0,501
Prompt en inglés	0,358

5.15. Cuadro: Prueba: Prompt en inglés

Tras analizar los resultados podemos concluir que ChatGPT no hace mejor todo en inglés. A pesar de haberse entrenado en su mayoría con información en inglés, las tareas en otros idiomas como puede ser el español las realiza igual o mejor que en su idioma nativo. Estos resultados pueden deberse a que el prompt no está tan optimizado como el desarrollado para la tarea principal. Con un mayor tiempo de estudio, se podría obtener un prompt capaz de dar unos mejores resultados.

6. Resultados

En esta sección se expondrán los resultados obtenidos de los objetivos definidos en la sección 1.3.4.

6.1. Capacidad de ChatGPT para extraer las respuestas de un texto

En este apartado, valoraremos los resultados obtenidos en la primera sección 5.1. Tal y como habíamos comentado al explicar los objetivos del proyecto, era necesario primero comprobar la capacidad de ChatGPT para que dado un texto de referencia supiera extraer correctamente las respuestas a las distintas preguntas que se le hagan. Tras realizar cientos o incluso miles de pruebas para la mejora del prompt extractivo, se ha obtenido una **precisión media de 0,5 sobre 1** (tanto con el conjunto *train* como con el *test*). Si recordamos cómo se valoran las precisiones en *BLEU*, sabemos que 0,3 se considera un buen resultado, y entre 0,6 - 0,7 un resultado perfecto (recordemos lo explicado en el apartado de BLEU, sección 4.2). Por tanto, podemos afirmar que cumple las expectativas vertidas sobre la eficiencia de ChatGPT en la extracción de respuestas en un texto.

Por la forma en la que se han hecho las pruebas, es más difícil conocer fallos que se realicen muy habitualmente. Sin embargo, se puede decir que comete menos fallos graves en casos extremos que el prompt conversacional. No tener que modificar la respuesta que tiene que dar, provoca que haya menos probabilidades de que se invente nuevo texto, que nada tiene que ver con la respuesta que tiene que dar.

Seguramente si se hubiera hecho una supervisión humana, tal y como se ha hecho con el prompt conversacional, los resultados habrían sido bastante mejores. No obstante, hay que tener en cuenta el tiempo invertido ya en todas las prueba realizadas con *BLEU* para esa tarea, y los recursos necesarios que se deberían invertir para hacer una nueva valoración por parte de un conjunto de evaluadores. Además, al haberse hecho las pruebas con tantas conversaciones, hacer una supervisión humana con un número reducido de conversaciones como 10, podría no servirnos para una correcta evaluación del prompt.

Para concluir, se debe decir que vistos los resultados obtenidos, la tarea del prompt conversacional debería obtener muy buenos resultados.

6.2. Capacidad de ChatGPT para actuar como un paciente virtual

En este apartado, se valorarán los resultados obtenidos en el apartado de 5.1. Como hemos explicado en este apartado, la única forma válida de evaluar la eficacia del prompt conversacional diseñado era mediante una evaluación humana. 4 personas tomaron parte de esta importante tarea para decidir lo buena herramienta que es este proyecto desarrollado. La **nota media sobre 10** de los evaluadores fue de un **7,19**, teniendo en cuenta las ponderaciones que se decidieron para ser más justos. La *exactitud* se ponderó con 60 % siendo la característica que peor nota de las 3 obtenía a lo largo de todas las pruebas, ya que era la más difícil de lograr. En segundo lugar, se encontraba la *coherencia* que se valoró con 30 % para darle más importancia a que nuestro paciente virtual se comportara como uno real que acaba de entrar a consulta. Por último, está la *fluidez* con 10 % que era lo que mejores nota tenía, y por eso no se le dio tanta importancia.

Además de la nota final, cabe recalcar los comentarios que hicieron los evaluadores tras finalizar su tarea. En primer lugar, y la más comentada está la respuesta “No está en mi historial médico” que da el paciente virtual cuando la respuesta a la pregunta no sabe responderla. La queja de los evaluadores viene de que esa no es una respuesta que vaya a dar un paciente real. Sin embargo, es un caso que en una

conversación médico-paciente no se daría. El paciente por ejemplo, si le preguntan “¿Has tenido tos?” debería saber responderla, aunque no esté escrito en su historial médico. Por tanto, es un caso especial y buscar una respuesta creíble a ese caso es complicado. Dentro del conjunto de datos, la respuesta era “ ”, o lo que es lo mismo no responder. La respuesta inventada para la tarea “No está en mi historial médico” no ha resultado ser muy convincente, por lo que habría que buscar una diferente. Posibles respuestas para este caso serían: “No me consta.” o “No sé.”. Este caso ha empeorado mucho la valoración de *coherencia*, ya que se da muchas veces durante el transcurso de la conversación, y ha habido evaluadores que han valorado con un 1 esta categoría.

Otro caso muy claro en el que falla es cuando el médico se dirige al paciente virtual en 3ª persona, sobre todo en la primera pregunta. Al final, recordamos que se le da la conversación completa a ChatGPT cuando se le hace una nueva pregunta, por lo que si se hace una pregunta utilizando el usted, generalmente responde correctamente en 1ª persona. Pero, la primera pregunta es muy determinante, ya que no tiene más ejemplos de los que aprender. Por este mismo motivo, en la frase introductoria que hace Furhat se pide que para una mejor experiencia se le hagan las preguntas en 2ª persona. De esta manera, reducimos el riesgo de que se equivoque y responda sin seguir las pautas establecidas.

Por último, tenemos el caso especial en el que no se requiere una respuesta, por ejemplo:

“De acuerdo”, “Lo he entendido”, “Ok” ... Dentro del prompt extractivo este caso se trata de una manera más fácil, porque en las instrucciones se cataloga este caso con: *Las preguntas que no tienen “¿”, “?” se responden con “I”*. No obstante, esta instrucción no sirve con Furhat, ya que como podemos observar en la figura 5.4 las preguntas no tienen signos de puntuación. Se ha probado a incluir una instrucción en la que se instruya directamente que a ese tipo de “preguntas” se respondan con un simple “OK”, pero no funciona bien. Lo positivo es que este caso se da pocas veces o en general tiene poca importancia, porque la respuesta que se requiere no es importante. El lado negativo es la imagen que se da de que no es un paciente real, sino que un robot.

En cuanto a las despedidas, también falla a veces, ya que si no detecta un “Adiós” o un saludo final muy claro no comprende que es una forma de finalizar la conversación. Hay veces que da una respuesta parecida al caso anterior y se desprende del papel de paciente que está en una consulta. Pero si se realiza una despedida clara, rara vez suele fallar.

6.3. Tarea secundaria

En esta sección se van a mencionar los resultados obtenidos en el apartado 5.5.1. En las primeras reuniones surgió la idea de que ChatGPT pudiera realizar mejor esta tarea en inglés, al haber sido desarrollado en este idioma. Sin embargo, tal y como hemos probado esto no ocurre. Los resultados son bastante peores que los del castellano. Se puede deber al prompt desarrollado, el cual es una simple traducción del diseñado en 3.12. Una investigación más profunda para encontrar un mejor prompt en inglés y una traducción más correcta podría mejorar los resultados obtenidos.

7. Conclusiones

Se han desarrollado con éxito las tareas propuestas al comienzo del proyecto: analizar la capacidad de ChatGPT para extraer de un texto las respuestas a las preguntas que se le hagan, desarrollar e implementar un paciente virtual utilizando ChatGPT y Furhat, y probar si los resultados de las pruebas mejoran cambiando el idioma al inglés.

El proyecto se ha realizado satisfactoriamente, gracias al trabajo constante durante más de 3 meses. A pesar de lo complicado de estimar la duración de las tareas, se han cumplido los plazos, exceptuando la prueba con el robot real, que no dependía totalmente del desarrollador y que se intentará realizar en un futuro cercano.

Tras haber mencionado los resultados y el análisis de los mismos en el anterior apartado, en este capítulo se hablará sobre el trabajo futuro para mejorar los resultados del proyecto y una reflexión final.

7.1. Trabajo futuro

En cuanto a los objetivos para un trabajo futuro encontraríamos tratar de mejor manera los casos especiales: las preguntas, cuya respuesta no se encuentra en el historial médico y aquellas que no requieran una respuesta. Quizás una mejor investigación y más pruebas sobre diferentes técnicas de *prompt engineering* ayudarían a responder mejor esos casos.

También un aspecto muy importante es la evolución de los modelos de ChatGPT. Este proyecto se ha realizado con la versión *GPT-3.5-turbo*, una versión anterior y peor que *GPT-4-Turbo* o *GPT-4o* que salió mientras se estaban realizando las pruebas para este proyecto. La evolución de los modelos es casi más rápida que el tiempo que se tarda en desarrollar un proyecto de estas magnitudes, por lo que no se puede esperar utilizar siempre los mejores y más actualizados. Seguramente si este proyecto se hubiera realizado con versiones mejores, los resultados lo habrían sido también, pero se decidió mantener el modelo por factores económicos sobre todo.

Por supuesto, realizar pruebas con un robot real y poder ver cómo se desenvuelve realizando el papel de un paciente virtual sería muy necesario. Si se quiere poner en funcionamiento este proyecto y que cubra la necesidad existente en los exámenes de la facultad de Medicina de la UPV/EHU, se deben hacer pruebas para que se vea si se puede utilizar o no realmente. Puliendo los fallos mencionados anteriormente, podría ser una herramienta muy válida para sustituir actores reales, siempre y cuando haya un responsable supervisando las respuestas.

7.2. Reflexión personal

Tras finalizar este proyecto, es el momento de apreciar la Inteligencia Artificial y la capacidad que tiene para realizar tareas que hasta ahora eran impensables. Una única herramienta como ChatGPT es capaz de tanto extraer de un texto las frases concretas que respondan a una pregunta, como de una vez tiene esas respuestas convertirlas en una respuesta convincente. Hace no muchos años se debían desarrollar sistemas de Question Answering [31] para un proyecto en específico, mientras que ahora una única herramienta hace cualquier papel que se le proponga. Además, está el robot Furhat, el cual también utiliza la IA para comprender lo que se le dice y clasificarlo según el motivo detrás de la frase. Estos avances nos sorprenden mucho, pero al ritmo al que va la tecnología dentro de unos años estos logros se considerarán mediocres. Tareas que ahora mismo consideramos impensables, se lograrán gracias a la IA en cuestión de segundos.

Si vamos más allá de nuestro caso concreto y ponemos la mirada en pacientes virtuales (en los que se prueban medicamentos o técnicas nuevas), muchas investigaciones se desarrollarán con estas nuevas tecnologías y supondrán una ayuda tremenda para mejorar los resultados actuales e incluso curar diferentes enfermedades.

Siempre existirá la controversia de utilizar ordenadores para recrear situaciones y enfermedades reales, pero con los avances actuales y futuros, posiblemente será la forma más fácil y efectiva de realizar experimentos.

En cuanto a lo aprendido gracias a este proyecto, están todas las tecnologías utilizadas. Aunque varias de ellas se hayan aprendido a lo largo de la carrera, el profundizar en ellas para hacer un desarrollo real ha ayudado enormemente a mi formación para un futuro profesional. Establecer plazos con un proyecto de tales dimensiones, hace que el trabajo deba ser constante y fructífero.

Por último, ver como mi esfuerzo y dedicación han logrado desarrollar un paciente virtual y que sea útil para la universidad o futuros proyectos, me enorgullece muchísimo. Espero que se siga investigando en esta dirección para que se cumpla el objetivo con el que se ideó este proyecto.

Bibliografía

- [1] Ainalluna. Roles en el prompt. URL <https://ainalluna.com/blog/la-importancia-de-asignar-un-rol-al-prompt>.
- [2] AmazonAWS. ¿qué es gpt? . URL <https://aws.amazon.com/es/what-is/gpt/>.
- [3] AmazonAWS. Ia generativa. . URL <https://aws.amazon.com/es/what-is/generative-ai/>.
- [4] AmazonAWS. ¿qué es un modelo llm? . URL <https://aws.amazon.com/es/what-is/large-language-model/>.
- [5] AprendeMachineLearning. ¿qué es el overfitting y underfitting? URL <https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/>.
- [6] AprendePython. ¿cómo funcionan las librerías en python? URL <https://aprendepythonhoy.com/librerias-en-python-y-su-utilidad-para-programar/>.
- [7] Janire Arana. Vir-pat: Paziante birtuala simulatzeko galdera-erantzun sistema eta corpusa.
- [8] CloudFlare. ¿qué es un modelo llm y por qué se llaman así? URL <https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model/>.
- [9] Diego L. Carrillo-Pérez y Raúl Carrillo-Esper Dioselina Lanzagorta-Ortega. Inteligencia artificial en medicina: presente y futuro. URL https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0016-38132022001100017.
- [10] DocsFurhat. Documentación furhat. URL <https://docs.furhat.io/applications/>.
- [11] Christopher Wavell-Saba Shahab Meghan Chevalier-Scott Good Isabella De Blasi Rafik Rhouma Christopher McMahon Jean-Paul Lam Thomas Lo Faiha Fareez, Tishya Parikh and Christopher W. Smith. A dataset of simulated patient-physician medical interviews with a focus on respiratory cases. scientific data, 9(1):313,2022.
- [12] Furhat. ¿qué es un robot furhat? URL <https://furhatrobotics.com/>.
- [13] Oscar García Esquirol. Futuro de la enseñanza médica: inteligencia artificial y big data. URL http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S2014-98322015000300009&nrm=iso.
- [14] Github. Skills predeterminadas disponibles en furhat. URL <https://github.com/FurhatRobotics/example-skills>.
- [15] HealthySimulation. Antecedentes pacientes virtuales. URL <https://www.healthysimulation.com/30298/clinical-simulation-history/>.
- [16] Ibm. ¿qué es una api? . URL <https://www.ibm.com/es-es/topics/api>.
- [17] Ibm. Aprendizaje supervisado y no supervisado. . URL <https://www.ibm.com/es-es/topics/unsupervised-learning>.
- [18] Ibm. ¿qué es un chatbot y para qué sirve? . URL <https://www.ibm.com/es-es/topics/chatbots>.
- [19] Ibm. ¿qué son los modelos fundacionales y cuáles son? . URL <https://research.ibm.com/topics/foundation-models>.

- [20] idDigitalSchool. ¿qué es una librería en python? URL <https://iddigitalschool.com/bootcamps/que-son-las-librerias-de-python/>.
- [21] Mikel Idoyaga. Vir-pat: Paziante birtuala simulatzeko txatbota, intentzio sailkatzailea eta corpusa. Disponible en Archivo PDF.
- [22] Incae. ¿qué es el prompt engineering? URL <https://www.incae.edu/es/blog/2023/03/29/prompt-engineering-que-es-y-donde-puedo-aprenderlo.html>.
- [23] Ionos. ¿cómo funciona la api de chatgpt. URL <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/chatgpt-api/>.
- [24] Linkedin. Historia de la inteligencia artificial, inicios y evolución. URL <https://es.linkedin.com/pulse/desde-sus-inicios-la-evoluci%C3%B3n-de-inteligencia-artificial>.
- [25] NextIdeaTech. Modelo gpt 3.5-turbo-instruct. URL <https://blog.nextideatech.com/openai-gpt-3-5-turbo-instruct/>.
- [26] Nvidia. ¿qué son los transformers? . URL <https://la.blogs.nvidia.com/blog/que-es-un-modelo-transformer/>.
- [27] Nvidia. Foundation models. . URL <https://blogs.nvidia.com/blog/what-are-foundation-models/>.
- [28] OpenAI. Interfaz web de chatgpt. URL <https://chatgpt.com/>.
- [29] PromptingGuide. Introducción al prompt engineering. URL <https://www.promptingguide.ai/es/introduction/basics>.
- [30] QuestionPro. Modelos generativos. URL <https://www.questionpro.com/blog/es/modelos-generativos/#:~:text=Los%20modelos%20generativos%20son%20herramientas,de%20tu%20conjunto%20de%20datos>.
- [31] RecuperacionInf. Recuperación de información: Procesamiento de lenguajes naturales. URL <http://recuperacioninf.orgfree.com/Question-Answering.html>.
- [32] SAS. Funcionamiento del nlp. URL https://www.sas.com/es_es/insights/analytics/what-is-natural-language-processing-nlp.html.
- [33] Tableau. Historia de la inteligencia artificial, inicios y uno de sus precursores. URL <https://www.tableau.com/data-insights/ai/history>.
- [34] Telefonica. ¿qué es chatgpt y para qué sirve? URL <https://www.telefonica.com/es/sala-comunicacion/blog/chatgpt-que-es-para-que-sirve-y-como-usarlo/>.
- [35] theblackboxlab. Detalles de los transformers. URL <https://theblackboxlab.com/2023/06/07/transformers/>.
- [36] TowardsDataScience. Puntuaciones aceptables de una evaluación bleu. URL <https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b>.
- [37] UniversidadEuropea. Explicación extensa diferencias entre aprendizaje supervisado y no supervisado. URL <https://universidadeuropea.com/blog/aprendizaje-supervisado-no-supervisado/>.
- [38] VirtualMedicalCoaching. ¿what is virtual patient simulation? URL <https://blog.virtualmedicalcoaching.com/what-is-virtual-patient-simulation>.
- [39] Whatsnew. ¿cómo se entrenó a chatgpt? URL <https://www.whatsnew.com/2023/07/18/como-se-realizo-el-proceso-de-entrenamiento-de-chatgpt/>.