Universidad de Valladolid

E.T.S. Ingenieros de Telecomunicación

TRABAJO DE FIN DE GRADO

Grado en Ingeniería de tecnologías específicas de telecomunicación
Mención en sistemas electrónicos

Implementación de herramientas de atención al Usuario mediante modelos fundacionales LLM en la Uva.

Septiembre 2024

Tutor:

Don. Juan Pablo de Castro Fernández

Autor:

Don. Adrián Bernardo Barrero

Resumen

El presente TFG tiene como objetivo el desarrollo de un chatbot sustentado en un LLM (Large Language Model) para la página web de la universidad. Este chatbot permitirá al Centro de Asistencia al Usuario (CAU), prestar un servicio asistido por IA, especialmente a los usuarios de nuevo ingreso, obtener información de forma rápida y sencilla sobre la resolución de diversos problemas y temas relacionados con la universidad, como, por ejemplo:

- Problemas: acceso al campus virtual, recuperación de contraseñas o uso de vTUI
- Información académica: Planes de estudio, asignaturas, horarios, exámenes, etc.
- Trámites administrativos: Matrícula, becas, ayudas, etc.
- Servicios universitarios: Biblioteca, comedor, alojamiento, etc.
- Actividades extracurriculares: Deportes, asociaciones, eventos culturales, etc.
- Preguntas frecuentes de índole técnica.

Abstract

The present Final Degree Project aims to develop a chatbot based on a Large Language Model (LLM) for the university's Help Desk. This chatbot will enable the User Assistance Center (UAC) to provide AI-assisted service, especially to new users, allowing them to quickly and easily obtain information on the resolution of various problems and issues related to the university. Some examples could be:

- Problems: access to the virtual campus, password recovery or vTUI use.
- Academic information: subjects, timetables, exams...
- Administrative procedures: school enrollment, scholarship.
- University services: library, accommodation.
- Extracurricular activities: Sports, association, events.
- Frecuently asked questions

Key words: *AI, LLM, chatbot*

Agradecimientos

En primer lugar, a mi familia por darme la oportunidad de tener una educación de calidad, aunque nunca me entendiesen cuando hablaba de la carrera.

A compañeros y amigos con los que tantas horas compartimos al final en la universidad, especialmente aquellos que formamos parte de una asociación increíble.

A profesores y otro personal, que por su vocación han sido capaces de transmitir pasión y conocimiento sobre el sector.

Gracias a todos por acompañar en este camino.

Índice general

1 INTRODUCCIÓN

1.1 CONTEXTO Y MOTIVACIÓN

En este momento donde la inteligencia artificial se encuentra en pleno auge, su uso para mejorar el potencial de esta institución es clave, además se plantean otros problemas a los que puede hacer frente:

Mejora de la accesibilidad a la información:

- Los usuarios podrán obtener información de forma rápida y sencilla, sin necesidad de tener que contactar con la secretaría de la universidad o buscar en la página web.
- El chatbot estará disponible 24/7, de manera que la obtención de información sea posible en cualquier momento y desde cualquier lugar.

Mejora de la calidad de los servicios:

- El chatbot podrá recopilar información sobre las necesidades de los estudiantes, lo que permitirá mejorar la calidad de los servicios que ofrece la universidad.
- El chatbot podrá ofrecer un servicio personalizado a cada estudiante, adaptándose a sus necesidades e intereses.

Reducción de costes:

- El chatbot puede automatizar o asistir en las tareas que actualmente realizan los empleados de la universidad, como por ejemplo responder a preguntas frecuentes, esto puede liberar a empleados de tareas repetitivas para su dedicación en tareas más productivas.

Mejora de la imagen de la universidad:

- El desarrollo de un chatbot innovador puede mejorar la imagen de la universidad y hacerla más atractiva de cara a posibles estudiantes.
- La universidad se posicionará como una institución moderna y comprometida con la mejora de la calidad de la educación.

En resumen, el desarrollo de un chatbot puede aportar numerosos beneficios tanto a los estudiantes como a la propia universidad.

En el Centro de Atención al Usuario (CAU) de la Universidad se atienden multitud de solicitudes de ayuda y consultas repetitivas, lo cual consume una gran cantidad de recursos humanos. Estas tareas, por su naturaleza, son abordables mediante sistemas de inteligencia artificial para aumentar la eficiencia y mejorar la atención global.

Es por eso por lo que se pretende implantar, una arquitectura experimental que pueda ayudar en la resolución de incidencias frecuentes y que se puedan detectar y responder mediante las capacidades de gestión del lenguaje natural de los nuevos LLM. Estas incidencias pueden necesitar realizar comprobaciones de servicios y consultas en base de datos, dado que suele ser necesario investigar las condiciones personales de cada problema y para cada usuario.

1.2 INTELIGENCIA ARTIFICIAL

La inteligencia artificial (IA) es un campo de la informática que busca crear sistemas que puedan simular la inteligencia humana. Esto abarca una amplia gama de capacidades,

desde el reconocimiento de patrones y el aprendizaje automático, hasta el razonamiento complejo y la toma de decisiones.

En el ámbito de la automatización de tareas, la IA tiene un enorme potencial para transformar la forma en que trabajamos. Algunas de las áreas donde la IA ya está teniendo un impacto significativo incluyen:

- Procesamiento del lenguaje natural (PLN): La IA se puede usar para automatizar tareas como la traducción de idiomas, la redacción de textos y la extracción de información de documentos.
- Visión artificial: La IA se puede usar para automatizar tareas como el reconocimiento de imágenes, la inspección de productos y el control de calidad.
- Robótica: La IA se puede usar para controlar robots que pueden realizar tareas físicas en entornos industriales, médicos y otros.

Esta herramienta ofrece una serie de oportunidades para automatizar tareas repetitivas y monótonas, como:

- Análisis de datos: La IA puede usarse para analizar grandes conjuntos de datos de redes y dispositivos para identificar patrones y tendencias.
- Mantenimiento de redes: También puede utilizarse para detectar y solucionar problemas de red de forma proactiva.
- Optimización de redes: Otra opción es su uso para optimización del rendimiento de las redes y mejorar la calidad del servicio.
- Atención al cliente: Por último, puede usarse para proporcionar asistencia a los clientes 24/7 y resolver problemas de forma rápida y eficiente.

A medida que la IA continúe evolucionando, es probable que veamos aún más aplicaciones para la automatización de tareas, también en el ámbito de las telecomunicaciones.

Algunos ejemplos específicos de cómo se está utilizando en la actualidad en el sector de las telecomunicaciones:

- Verizon: Esta compañía utiliza inteligencia artificial para analizar datos de red y predecir cuándo es probable que ocurran fallos. Esto les permite tomar medidas preventivas y evitar interrupciones del servicio. [1]
- Movistar: Movistar implementa soluciones de análisis de datos avanzados. Utilizan IA para comprender las necesidades de los clientes y crear ofertas personalizadas. [1]
- Tesla: Tesla utiliza inteligencia artificial en sus vehículos para mejorar la conducción autónoma y la experiencia del usuario. La empresa ha desarrollado sistemas avanzados de aprendizaje automático que permiten a sus coches aprender de las condiciones de conducción y optimizar su rendimiento. [1]

En resumen, la IA tiene un enorme potencial para transformar cualquier sector. Al automatizar tareas, puede liberar a las personas para que se concentren en tareas más creativas y estratégicas.

1.3 MACHINE LEARNING Y DEEP LEARNING

El aprendizaje automático (Machine Learning) es un campo de la inteligencia artificial que se centra en la creación de algoritmos que puedan aprender y mejorar su rendimiento por sí mismos, sin ser programados explícitamente. Estos sistemas se entrenan con grandes cantidades de datos para identificar patrones y tomar decisiones basadas en esos patrones.

El aprendizaje profundo (Deep Learning) es un subcampo del aprendizaje automático que utiliza redes neuronales artificiales para aprender de los datos. Las redes neuronales están inspiradas en el funcionamiento del cerebro humano, y son capaces de aprender representaciones complejas de los datos.

Concluyendo, el aprendizaje automático es un campo más amplio que abarca una variedad de técnicas, mientras que el aprendizaje profundo es un tipo específico de aprendizaje automático que utiliza redes neuronales.

Aprendizaje Automático: El aprendizaje es un proceso mediante el cual las redes neuronales adquieren la capacidad de aprender de los datos, sin ser programados explícitamente para cada tarea. Estos algoritmos buscan identificar patrones y relaciones ocultas en grandes conjuntos de datos para realizar predicciones, tomar decisiones o generar contenido nuevo. Existen diferentes tipos de aprendizaje:

- Aprendizaje supervisado: Se entrena el sistema con un conjunto de datos que incluye ejemplos de entrada y salida.
- Aprendizaje no supervisado: Se entrena el sistema con un conjunto de datos sin ejemplos de salida.
- Aprendizaje por refuerzo: Se entrena el sistema a través de ensayo y error, recompensándolo por comportamientos deseables y penalizándolo por comportamientos indeseables.

La manera en la que se interactúa con la información de entrada de estas redes neuronales permite el desarrollo de aplicaciones complejas, como pueden ser:

- Reconocimiento de imágenes: Identificar objetos o personas en imágenes.
- Procesamiento del lenguaje natural: Entender y generar lenguaje humano.
- Predicción de series temporales: Pronosticar valores futuros a partir de datos históricos.
- Detección de fraudes: Identificar transacciones fraudulentas.

1.4 DEEP LEARNING:

Redes neuronales: Las redes neuronales son modelos computacionales inspirados en el cerebro biológico. Están compuestas por nodos interconectados, llamados neuronas artificiales, que procesan información. Estas redes aprenden a partir de datos, ajustando los pesos de sus conexiones para realizar tareas como clasificación, regresión y generación.(4. Introducción a Redes Neuronales Artificiales, n.d.)

- Neuronas: Unidades básicas de procesamiento de las redes neuronales. Cada conexión entre neuronas posee un peso que determina la importancia de la información que fluye a través de ella. Además, cada neurona cuenta con un sesgo, un valor constante que se añade a la suma ponderada de las entradas, permitiendo ajustar el umbral de activación. Para introducir no linealidad en el procesamiento, se emplean unas funciones de activación.
- Capas: Las neuronas se organizan en capas, que pueden ser de entrada, ocultas o de salida. Estas capas funcionan de manera secuencial. Estas capas aprenden a extraer características cada vez más abstractas de los datos, desde bordes simples en imágenes hasta conceptos complejos como rostros o objetos. Esta capacidad de

extracción de características es fundamental para que la red pueda generalizar y realizar predicciones precisas.

- Funciones de activación: Las neuronas utilizan funciones de activación para convertir sus entradas en salidas. La elección de la función de activación es crucial, ya que influye en la capacidad de la red para converger durante el entrenamiento y en la calidad de las predicciones.

Según la arquitectura de procesado de datos y cómo se interconectan entre sí las piezas fundamentales, se pueden distinguir algunos tipos clásicos de redes neuronales:

- Redes Neuronales Artificiales (ANN): Las más básicas, con múltiples capas de neuronas.
- Redes Neuronales Convolucionales (CNN): Especializadas en procesar datos visuales, como imágenes.
- Redes Neuronales Recurrentes (RNN): Diseñadas para procesar secuencias, como texto o series temporales.
- Redes Neuronales de Memoria a Largo Plazo (LSTM): Un tipo de RNN que resuelve el problema del desvanecimiento del gradiente en secuencias largas.
- Redes Generativas Adversariales (GAN): Compuestas por dos redes que compiten entre sí para generar datos realistas.

1.5 TRANSFORMERS

'Transformers' es un concepto de arquitectura en redes neuronales que surge en torno a 2017 [3]. Procesa las secuencias de una manera diferente al resto de tipos de RN. Esto permite aumentar la eficiencia en la captura dependencias de largo alcance. Características importantes para comprender la arquitectura Transformers son:

- Mecanismo de Atención: Es un sistema que permite a la red ponderar la importancia de diferentes partes de entrada al generar la salida.
- Codificador-Decodificador: La arquitectura típica de transformers consta de un codificador que procesa la entrada y un decodificador que genera la salida. Esta codificación y decodificación se utiliza para calcular los vectores numéricos.
- Autoatención: Permite a la red atender a diferentes partes de sí misma para comprender mejor el contexto.
- Multi-Cabeza de Atención: Usa múltiples mecanismos de atención en paralelo para capturar diferentes aspectos. Así se consigue aumentar la capacidad de entender frases más complejas.
- Posiciones Codificadas: Agrega información posicional a las entradas para que la red comprenda el orden de las palabras. Esto se realiza mediante el añadido de vectores numéricos.
- Captura de Dependencias de Largo Alcance: Permite entender relaciones entre palabras distantes en una secuencia. Esto puede cambiar de manera importante el significado de una oración en tareas como traducción.
- Paralelización: Permite entrenar modelos más grandes y complejos en menos tiempo dado que el entrenamiento se realiza en paralelo en múltiples procesadores o GPUs.

Transformers ha demostrado ser una arquitectura poderosa y versátil contribuyendo al desarrollo del NLP. El aumento de eficiencia en el procesamiento paralelo de secuencias lo ha convertido en la base de muchos modelos de lenguaje de nueva creación.

Ya sean de arquitectura transformers o no, algunas de las aplicaciones de estas redes pueden ser:

- Reconocimiento de voz: Convertir el habla en texto.

- Traducción automática: Traducir texto de un idioma a otro.
- Generación de texto: Crear textos originales, como poemas o artículos.
- Visión artificial: Permitir el reconocimiento de objetos, personas u otros a través de diferentes entradas.

Tanto el aprendizaje automático como el aprendizaje profundo tienen un enorme potencial para transformar la forma en que vivimos y trabajamos. Se están utilizando en una amplia gama de aplicaciones, desde el cuidado de la salud hasta las finanzas. Se espera que su impacto siga en crecimiento.

El entrenamiento de una red neuronal es un proceso iterativo que consiste en ajustar los pesos de la red para minimizar una función de pérdida, lo que permite a la red aprender a realizar tareas específicas a partir de datos. El entrenamiento consta de algunas partes importantes:

- Propagación hacia adelante: La propagación hacia adelante es la primera fase del entrenamiento. En esta etapa, los datos de entrada se introducen en la red neuronal y se propagan a través de cada capa hasta alcanzar la capa de salida. En cada capa, los valores de las neuronas se calculan como una combinación lineal de las entradas de la capa anterior, seguida de una función de activación. Comienza con la inicialización aleatoria de los pesos sinápticos que conectan las neuronas. A continuación, la entrada se introduce en la primera capa de la red y se propaga a través de cada neurona sucesiva. En cada neurona, se calcula una suma ponderada de las entradas de la capa anterior, a la que se aplica una función de activación no lineal. Esta salida se convierte en la entrada de la siguiente capa, y así sucesivamente, hasta que se alcanza la capa de salida, produciendo la predicción final de la red.
- La función de pérdida: La función de pérdida es una métrica que mide la discrepancia entre las predicciones realizadas por una red neuronal y los valores reales de los datos. En otras palabras, cuantifica qué tan equivocada está la red en sus predicciones. El objetivo del entrenamiento es encontrar los parámetros de la red que minimicen esta función de pérdida. Existen diferentes tipos de funciones de pérdida adaptadas a distintos problemas. Por ejemplo, el error cuadrático medio (MSE) se utiliza comúnmente en problemas de regresión, donde se busca predecir un valor numérico continuo, mientras que la entropía cruzada es ideal para problemas de clasificación, en los que se busca asignar una entrada a una categoría específica entre varias posibles.
- Propagación hacia atrás: La propagación hacia atrás es un algoritmo que permite a una red neuronal aprender de sus errores. Una vez que la red realiza una predicción y se calcula la diferencia entre esta y el valor real (la función de pérdida), la propagación hacia atrás inicia un proceso de ajuste de los pesos de la red. Este proceso comienza calculando el error en la capa de salida y luego propaga este error hacia atrás a través de cada capa de la red, calculando cuánto contribuyó cada peso al error total. Con esta información, se actualizan los pesos en la dirección opuesta al gradiente del error, acercando así la red a una configuración que minimiza la función de pérdida. Es decir, la propagación hacia atrás es como un proceso de retroalimentación que permite a la red "aprender" de sus equivocaciones y mejorar sus predicciones en futuras iteraciones.[4]
- Optimizadores: Los optimizadores son los mecanismos que permiten a las redes neuronales aprender ajustando sus parámetros a lo largo del entrenamiento. Estos algoritmos calculan la dirección en la que los pesos de la red deben modificarse para

reducir al mínimo la función de pérdida. El descenso del gradiente es el algoritmo más básico, pero su convergencia puede ser lenta. Para acelerar este proceso, se han desarrollado otros optimizadores como el momentum, que acumula la dirección del gradiente en iteraciones anteriores, y Adam, que combina las ventajas del momentum con una adaptación automática de la tasa de aprendizaje. En esencia, los optimizadores son los encargados de guiar a la red neuronal a encontrar la configuración de pesos óptima que le permita realizar las predicciones más precisas.

- Regularización: La regularización es un conjunto de técnicas empleadas en el entrenamiento de redes neuronales para prevenir el sobreajuste. El sobreajuste ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, perdiendo la capacidad de generalizar a nuevos datos no vistos. Para evitar esto, la regularización introduce restricciones en la complejidad del modelo. Técnicas como L1/L2 añaden un término de penalización a la función de pérdida que incentiva a la red a tener pesos más pequeños, reduciendo así su capacidad para modelar patrones muy específicos del conjunto de entrenamiento. Por otro lado, el dropout consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento, lo que fuerza a la red a aprender representaciones más robustas y menos dependientes de neuronas individuales. En resumen, la regularización es una herramienta fundamental para mejorar la capacidad de generalización de las redes neuronales y evitar que se vuelvan demasiado específicas de los datos de entrenamiento.

1.6 NLP (PROCESADO DE LENGUAJE NATURAL)

El procesado de lenguaje natural es otro de los campos más importantes de la inteligencia artificial, usado a menudo donde se quiere interacción humana. Su objetivo principal es permitir que las máquinas comprendan, interpreten y generen lenguaje humano de una manera que sea valiosa y útil.

Definición y objetivos del NLP: El NLP combina la lingüística, la inteligencia artificial (IA) y la computación para analizar y comprender el lenguaje humano. Los objetivos del NLP incluyen:

- Comprensión del Lenguaje: Hacer que las máquinas entiendan el significado del texto y el habla.
- Generación de Lenguaje: Permitir que las máquinas produzcan texto o habla que sea coherente y relevante.
- Interacción Natural: Facilitar la comunicación entre humanos y máquinas de manera intuitiva, utilizando el lenguaje natural.

Componentes del NLP: El NLP abarca varios componentes y subcampos, que incluyen:

- Análisis Sintáctico: Se refiere a la estructura gramatical de las oraciones. Implica la identificación de partes del discurso (sustantivos, verbos, adjetivos, etc.) y la construcción de árboles sintácticos que representan la estructura de la oración.
- Análisis Semántico: Se centra en el significado de las palabras y las oraciones. Esto incluye la desambiguación de palabras (determinar el significado correcto de una palabra en un contexto dado) y la representación del significado a través de estructuras como redes semánticas.
- Análisis Pragmático: Examina el uso del lenguaje en contextos específicos y cómo el contexto afecta el significado. Esto incluye la comprensión de implicaturas, intenciones y el contexto situacional.

- Análisis de Sentimientos: Se utiliza para determinar la actitud o emoción expresada en un texto, clasificando el contenido como positivo, negativo o neutral.
- Reconocimiento de Entidades Nombradas (NER): Implica la identificación y clasificación de entidades en un texto, como nombres de personas, organizaciones, ubicaciones y fechas.
- Traducción Automática: Se refiere a la conversión de texto de un idioma a otro utilizando algoritmos y modelos de aprendizaje automático.

Desafíos en el NLP: A pesar de los avances significativos, el NLP enfrenta varios desafíos:

- Ambigüedad del Lenguaje: Las palabras y frases pueden tener múltiples significados, lo que dificulta la comprensión precisa.
- Contexto y Matices: El significado de una oración puede depender del contexto, lo que requiere que los modelos comprendan sutilezas y matices.
- Variabilidad del Lenguaje: Las diferencias en dialectos, jergas y estilos de escritura pueden complicar el procesamiento.
- Datos Escasos: Para algunas lenguas o dominios específicos, puede haber una falta de datos etiquetados para entrenar modelos efectivos.
- Ética y Sesgo: Los modelos de NLP pueden perpetuar sesgos presentes en los datos de entrenamiento, lo que plantea preocupaciones éticas sobre su uso.

Dentro del procesamiento de lenguaje natural (NLP), se pueden diferenciar varias categorías de clasificación, cada una con sus propias características y aplicaciones. [5]

- Clasificación de texto: La clasificación de texto es un proceso que implica asignar etiquetas o categorías a documentos de texto basándose en su contenido. Existen diversos tipos de clasificación, como la binaria (dos categorías), la multiclase (varias categorías) y la jerárquica (categorías principales y subcategorías). Esta técnica tiene múltiples aplicaciones, desde el filtrado de spam hasta el análisis de opiniones y la traducción automática. Los algoritmos de clasificación se basan en características como palabras clave, partes de la oración y la estructura del texto. Entre los algoritmos más comunes se encuentran el K-Nearest Neighbors (KNN) y las Máquinas de Soporte Vectorial (SVM). Las redes neuronales, especialmente las recurrentes y las convolucionales, han demostrado ser especialmente efectivas en tareas de clasificación de texto, gracias a su capacidad para aprender representaciones complejas y contextuales del lenguaje.[6]
- Análisis de Sentimientos: Esta clasificación se centra en determinar la opinión o sentimiento expresado en un texto, como positivo, negativo o neutral. Se utiliza comúnmente en el análisis de reseñas de productos, comentarios en redes sociales y encuestas de satisfacción.
- Clasificación de Temas: Se refiere a la identificación de los temas o tópicos principales en un conjunto de documentos. Esto puede incluir la agrupación de documentos relacionados o la asignación de etiquetas temáticas a los textos.
- Clasificación de Intenciones: Utilizada en sistemas de diálogo y chatbots, esta clasificación identifica la intención detrás de una consulta del usuario, como "hacer una reserva", "consultar el estado de un pedido", etc.
- Clasificación de Entidades Nombradas (NER): Este tipo de clasificación se utiliza para identificar y clasificar entidades en un texto, como nombres de personas,

organizaciones, ubicaciones, fechas, etc. Es fundamental para la extracción de información y la comprensión del contexto en el texto.
- Clasificación Jerárquica: En este enfoque, las clases están organizadas en una estructura jerárquica. Por ejemplo, un documento puede clasificarse primero en una categoría general (como "ciencia") y luego en una subcategoría más específica (como "biología").
- Clasificación de Documentos: Se refiere a la asignación de documentos a categorías predefinidas basadas en su contenido. Esto es común en sistemas de gestión de documentos y bibliotecas digitales.

El desarrollo y evolución del procesado de lenguaje natural es tan importante como el de la propia inteligencia artificial y probablemente este crecimiento se siga sucediendo en ambos de manera simultánea.

2 OBJETIVOS

El objetivo principal de este trabajo es desarrollar un chatbot que pueda ayudar con problemas y preguntas en el centro de atención al Usuario de la universidad de Valladolid. Podrá servir de apoyo como herramienta a los trabajadores en el proceso de solucionar incidencias. El proyecto deberá cumplir los siguientes objetivos:

- Seleccionar un modelo fundacional LLM para basar el proyecto en él.
- Crear un proceso de extracción de la información que ya dispone la UVa para hacer un “fine-tunning”: Casos de CAU antiguos, documentación publicada, etc.
- Definir un proceso de “fine-tunning” del modelo.
- Crear una aplicación que se pueda integrar en el CAU de la UVa para ayudar a los técnicos de soporte. Por ejemplo, redactar respuestas.
- Crear un chatbot para atender en un primer nivel a los usuarios. Esto requiere un buen control de la exactitud y contexto de las respuestas.

3 ESTADO DEL ARTE

3.1 CHATBOT BASADO EN CAMINOS LÓGICOS

Los chatbots basados en caminos lógicos son un tipo de chatbot que utiliza un árbol de decisiones para determinar su respuesta [1][8]. El árbol de decisiones se compone de nodos, cada uno de los cuales representa una posible respuesta. Los nodos están conectados por ramas, que representan las posibles preguntas o comentarios del usuario.

Cuando un usuario interactúa con un chatbot basado en caminos lógicos, este comienza en el nodo raíz del árbol de decisiones. A continuación, se evalúa la pregunta o comentario del usuario y se sigue la rama correspondiente, siguiendo las ramas del árbol hasta que llega a un nodo que contiene una respuesta.

Estos chatbots asentados sobre caminos lógicos son relativamente sencillos de implementar, pero tienen algunas limitaciones. Una limitación relevante es que solo pueden responder a preguntas o comentarios que estén previstos en el árbol de decisiones. Si un usuario hace una pregunta o comentario que no está previsto en el árbol, el chatbot puede no ser capaz de proporcionar una respuesta válida, pueda ser por ejemplo una respuesta por defecto.

Otra limitación sustancial que comentar es que pueden ser excesivamente repetitivos y, por lo tanto, antinaturales. Esto se debe a que siempre recorrerá la misma ruta a través del árbol de decisiones para satisfacer a una pregunta o comentario determinado.

A continuación, se presentan algunos ejemplos de chatbots fundamentados en caminos lógicos:

- El asistente virtual de SEUR utiliza un árbol de decisiones para responder a las preguntas de los usuarios (inputs predefinidos, no permite pregunta libre).
- El chatbot de soporte técnico de un banco utiliza un árbol de decisiones para identificar el problema del usuario y proporcionar una solución.
- El chatbot de una tienda online utiliza un árbol de decisiones para ayudar a los usuarios a realizar compras, devoluciones u otras acciones.

3.1.1 Copilot Studio de Microsoft

Copilot Studio es una plataforma de inteligencia artificial (IA). En la realidad hemos visto que está basda en un bot guionizado con respuestas muy limitadas generadas por GPT-4.

Es importante destacar que Copilot Studio no tiene la capacidad de recordar conversaciones anteriores. Cada conversación se considera independiente y la IA no mantiene ninguna información entre conversaciones. Además, aunque ha sido entrenada en datos hasta cierto punto en 2021, puede que no tenga información sobre eventos o desarrollos más recientes.

En términos de seguridad, Copilot Studio está diseñado para evitar proporcionar información que pueda causar daño físico, emocional o financiero. También se abstiene de generar contenido sobre políticos influyentes o cualquier grupo de identidades sociales.

Esta plataforma tiene un enfoque diferente respecto a Azure OpenAI services. Está pensada directamente para implementar un chat, proporcionando el código que permite su incrustación, también a través de correo electrónico para automatizar respuestas, proporcionando un token para habilitar su uso a través de servidores de correo, y otras muchas maneras interesantes de explotarlo. Sin embargo, estas dos parecen las opciones más interesantes en nuestro caso de uso.

Aunque esta plataforma tiene la posibilidad de desarrollar otras opciones para la generación responsiva, como el uso de diálogos guionizados (caminos lógicos), redacción de respuestas basadas en el modelo GPT-4, permite sincronizarse con las aplicaciones de 'Power Automate', dispone de autenticación con servicios de terceros, o conectar con otras API's, el uso de la IA es insuficiente para los requerimientos buscados.

Tras un tiempo explorando esta opción, la conclusión a la que se llegó es que la inteligencia artificial está poco integrada, y aunque es capaz de generar respuestas, el desarrollo principal del procedimiento se lleva a cabo sin su uso, por lo que no aporta nada nuevo a las opciones que ya existían. Recordemos que el objetivo principal del trabajo es la aplicación de los modelos de lenguaje para su uso en atención al usuario.

3.1.2 Pruebas en Copilot Studio

Aunque finalmente se descartó Copilot Studio por limitaciones de la plataforma, se llevaron a cabo algunas funciones y métodos de valor para el proyecto que serán explicados. Copilot Studio es una plataforma que integra el uso de inteligencia artificial, pero de manera muy escasa. Los procedimientos están muy guionizados y no aporta el valor que estamos buscando de la inteligencia artificial para la automatización de muchos más procedimientos, motivo por el que finalmente se descartó, aunque se llegó a conseguir una interfaz capaz de cumplir la mayoría de los objetivos propuestos.

3.1.2.1 GENERACIÓN DATASETS Y DOCUMENTOS

Estos datasets incluyen la información necesaria para que el modelo sea capaz se responder las preguntas acerca de determinados temas, dado que la plataforma también acepta ficheros .docx y .pdf, no solo los archivos de texto serán necesarios para el modelo.

3.1.2.2 WEB SCRAPPING Y DESCARGA DOCUMENTOS

Tras el proceso de adaptación al puesto de trabajo, y de manera independiente al modelo y sistema para el desarrollo del LLM, el primer paso tomado ha sido la creación de los Datasets con los que fine-tuneará posteriormente. La primera idea a la que se recurrió fue la extracción del histórico de correos de la cuenta de soporte, para ser procesada en un formato '.pst'. Donde la pregunta realizada por el usuario se pasaría como input, y la respuesta proporcionada por el CAU sería el 'shot' o respuesta de aprendizaje, a mayores del tunning. Esto requiere una transformación exhaustiva del contenido, dada la importancia del filtrado de contenido inapropiado, como datos de usuario, lenguaje informal, derivaciones de la conversación, etc. Aunque este hecho es relevante, se suma a otros que finalmente han desencadenado descartar esta opción. Para empezar, dado el tamaño del archivo con el que se pretende trabajar, es imposible descargarlo, debido a que es capaz de colapsar cualquier ordenador desde el que se pretenda obtener. A pesar de esto, se descargó otro archivo de prueba con un pesaje significativamente menor y cargarlo en el entorno de trabajo. Incluso con las librerías existentes para trabajar este formato ('emailparser'), no es fácil de proceder. El archivo, el cuál puede contener documentos, se encuentran codificados de manera ilegible y unidos a la respuesta. Aunque consiguiésemos filtrar únicamente preguntas y respuestas, como ya se ha mencionado, la existencia de lenguaje informal, datos personales y otras van prácticamente a imposibilitar esta tarea

Una recopilación más formal probablemente pueda ser llevada a través de la página web 'digital.uva.es'. Para ello, recurriendo a 'bs4' (BeautifulSoup4), librería dedicada para web-scrapping, vamos a recorrer la página principal y todos sus subapartados.

Haciendo uso de esta librería junto con 'request', podemos extraer todos los 'hrefs[1]', y analizar de manera recursiva la página completa. Posteriormente categorizaremos estos enlaces en los diferentes tipos. Se han encontrado enlaces a PDF's (manuales y guías para usuarios), PNG's (fotos explicativas de procesos digitales) y subapartados de la página que contengan información relevante. Con esto logramos un script que automatiza la descarga de los documentos indexados por la página, y las propias páginas de las que extraeremos la información necesaria. Este proceso como método de extracción de información resulta más adecuado, dado que a medida que la página se mantenga en constante actualización, podrá volver a ejecutarse el código para disponer de la misma.

Una vez realizada la obtención de dichos 'html', para subirlos al repositorio de Copilot Studio, en caso de contener información relevante, será necesario convertirlos a texto. Estéticamente quedarán feos, pueden contener espacio o 'enters' innecesarias, prescindiendo de las fotos también, pero contendrán la información útil necesaria para responder a los usuarios. Este proceso de conversión también se ha automatizado en el entorno de trabajo, y guardando los .txt en un directorio aparte.

3.1.2.3 RETRIEVAL AUGMENTED GENERATION (RAG)

Retrieval Augmented Generation es una técnica de inteligencia artificial que mejora la calidad de la IA generativa al permitir a los grandes modelos de lenguaje (LLM) aprovechar

[1] HREF es el atributo utilizado en las páginas HTML que contiene la dirección real de un enlace.

recursos de datos adicionales sin necesidad de volver a entrenarlos. [9]La idea es aplicar dicha técnica a la documentación que se pretende utilizar.

De una manera simplificada:

La técnica de RAG ayuda a los LLM a proporcionar respuestas más precisas y relevantes al combinar la generación de texto con la recuperación de información de fuentes de datos adicionales.

¿Cómo funciona?

- Recuperación: Se utiliza la consulta del usuario para buscar información relevante en una fuente de datos externa, como una base de conocimientos o un conjunto de documentos.
- Generación: El LLM utiliza la información recuperada como contexto para generar una respuesta personalizada para el usuario.

Beneficios de RAG:

- Mayor precisión y relevancia: Las respuestas del LLM se basan en información real y verificada.
- Mejora en la fluidez y coherencia: La información recuperada ayuda al LLM a generar respuestas más completas y mejor estructuradas.
- Mayor diversidad de respuestas: El LLM puede acceder a una gama más amplia de información para generar respuestas más variadas.
- Ahorro de tiempo y recursos: No es necesario volver a entrenar el LLM para añadir nueva información.

Aplicaciones de RAG:

- Atención al cliente: Generar respuestas personalizadas a las preguntas de los clientes.
- Resumen de documentos: Extraer información clave de documentos largos y generar resúmenes concisos.
- Creación de contenido: Generar textos creativos, como poemas, artículos o guiones.
- Traducción: Mejorar la precisión y fluidez de las traducciones automáticas.

3.1.2.4 IMPLEMENTACIÓN OAUTH

OAuth (Open Authorization) es un protocolo de autorización estándar abierto que permite a los usuarios conceder a aplicaciones de terceros el acceso a sus recursos en otros sitios web, sin tener que compartir sus credenciales de inicio de sesión. Esto significa que puedes permitir que una aplicación acceda a tu información de Facebook, por ejemplo, sin tener que darle tu contraseña de Facebook.

El proceso de OAuth se basa en la idea de tokens de acceso. Un token de acceso es una cadena de caracteres única que se otorga a una aplicación después de que el usuario ha autorizado el acceso a sus recursos. La aplicación puede utilizar este token para acceder a los recursos del usuario en el sitio web del proveedor de servicios, como Facebook o Google.

Este proceso se puede dividir en cuatro pasos:

- Solicitud de autorización: El usuario inicia el proceso visitando la aplicación de terceros. La aplicación entonces redirige al usuario al sitio web del proveedor de servicios.

- Autorización del usuario: El usuario ve una lista de los recursos a los que la aplicación solicita acceso y decide si conceder o denegar el acceso.
- Emisión del token de acceso: Si el usuario concede el acceso, el proveedor de servicios emite un token de acceso a la aplicación.
- Acceso a los recursos: La aplicación utiliza el token de acceso para acceder a los recursos del usuario en el sitio web del proveedor de servicios.

OAuth tiene varios beneficios, incluyendo:

- Mayor seguridad: Los usuarios no tienen que compartir sus credenciales de inicio de sesión con aplicaciones de terceros, lo que reduce el riesgo de robo de identidad y fraude.
- Mejor experiencia del usuario: Los usuarios pueden autorizar el acceso a sus recursos de forma granular, lo que les da un mayor control sobre su privacidad.
- Mayor facilidad de uso: OAuth es un protocolo estándar, lo que facilita a los desarrolladores de aplicaciones implementar la autorización.

OAuth se utiliza en una amplia variedad de aplicaciones:

- Inicio de sesión social: Permite a los usuarios iniciar sesión en aplicaciones utilizando sus cuentas de redes sociales, como Facebook o Google.
- Aplicaciones móviles: Permite a las aplicaciones móviles acceder a la información del usuario en otros sitios web, como listas de contactos o calendarios.
- API: Permite a las aplicaciones acceder a datos y funcionalidades de otros sitios web.

Durante la exploración de Copilot Studio como solución al problema, esta herramienta se llegó a implementar, pensando en que el chatbot fuese público, pero dirigido exclusivamente para usuarios de la universidad.

3.1.3 DIALOGFLOW DE GOOGLE CLOUD

Dialogflow es otra plataforma de desarrollo de chatbots que permite crear interfaces de usuario conversacionales para aplicaciones móviles, aplicaciones web, dispositivos, bots, sistemas de respuesta de voz interactiva (IVR) y más. Utiliza la tecnología de comprensión del lenguaje natural (PLN) de Google para comprender el significado de las consultas de los usuarios y responder de forma precisa y relevante. También se puede utilizar Dialogflow para crear chatbots que puedan generar respuestas de voz natural.

Dialogflow es una herramienta flexible que se puede utilizar para crear una amplia gama de chatbots, desde simples bots de preguntas frecuentes hasta chatbots conversacionales complejos que pueden mantener una conversación con un usuario. Entre sus características más destacables o mencionables podríamos hablar de:

1. Potente motor de procesamiento del lenguaje natural (PLN):

- Dialogflow utiliza un motor de PLN para comprender el lenguaje natural de los usuarios y responder de forma precisa y relevante.
- Soporta una amplia gama de idiomas, incluyendo español, lo que lo hace ideal para proyectos de atención al cliente multilingües.

2. Entorno de desarrollo intuitivo:

- Dialogflow proporciona una interfaz gráfica de usuario intuitiva que facilita la creación de chatbots sin necesidad de conocimientos de programación.
- Permite crear flujos de conversación complejos mediante guionizado.

- Ofrece herramientas para la gestión de entidades y variables, lo que permite flujos más personalizados.

3. Integración con Google Cloud Platform:

- Dialogflow se integra con Google Cloud Platform (GCP), sin la necesidad de realizar cambios.
- Se puede integrar un chatbot con otros servicios de GCP como Dialogflow CX, Contact Center AI, Cloud Functions y Cloud Storage para generar embeddings o ampliar capacidades.

Esta integración facilita la creación de chatbots escalables y robustos.

3.2 CHATBOTS BASADOS EN PROCESADO DE LENGUAJE NATURAL (NLP)

Los chatbots sustentados en NLP son sistemas que simulan una conversación con humanos utilizando técnicas de procesamiento del lenguaje natural implementado muy frecuentemente mediante redes neuronales. Estos chatbots son cada vez más populares en diversas áreas como atención al cliente, educación, marketing y entretenimiento.

Técnicas clave de NLP: El Procesamiento del Lenguaje Natural (NLP) es un campo interdisciplinario que se enfoca en la interacción entre ordenadores y el lenguaje humano. A continuación, se detallan algunas de las técnicas clave que sustentan este campo:

- Análisis sintáctico: Se utiliza para comprender la estructura de una oración, identificando sus componentes (sustantivos, verbos, adjetivos, etc.) y sus relaciones.
- Análisis semántico: Se centra en el significado de las palabras y frases dentro del contexto de la conversación.
- Análisis pragmático: Se encarga de la intención del usuario detrás de la oración, considerando el contexto social y las relaciones entre los participantes.

Arquitectura de un chatbot basado en NLP: define la estructura y la interacción de los diferentes componentes que permiten al chatbot comprender, procesar y generar respuestas humanas. Esta arquitectura suele variar dependiendo de la complejidad del chatbot y de las tecnologías utilizadas, pero en general, podemos identificar los siguientes componentes principales:

- Módulo de entrada: Recibe la entrada del usuario, ya sea texto o voz.
- Procesamiento del lenguaje natural: Aplica las técnicas mencionadas para comprender la entrada del usuario.
- Generación de la respuesta: Crea una respuesta natural y coherente a la consulta del usuario.
- Módulo de salida: Envía la respuesta al usuario en el formato adecuado (texto, voz, etc.).

Tecnologías clave para chatbots: Los chatbots, como interfaces conversacionales basadas en inteligencia artificial, se sustentan en una combinación de tecnologías que les permiten comprender, procesar y generar lenguaje humano de forma natural:

- Redes neuronales profundas: Permiten a los chatbots aprender de grandes cantidades de datos y mejorar su capacidad de comprensión y generación de lenguaje.
- Machine learning: Se utiliza para entrenar modelos de lenguaje que predicen la siguiente palabra o frase en una conversación.
- Bibliotecas de NLP: Ofrecen herramientas pre-entrenadas para tareas como análisis sintáctico, análisis de sentimiento, lematización, extracción de entidades, etc.

Desafíos técnicos: el desarrollo de chatbots presenta una serie de desafíos técnicos que desarrolladores buscan superar constantemente:

- Ambigüedad del lenguaje: El lenguaje natural es complejo y puede ser interpretado de diferentes maneras.
- Falta de datos: Entrenar modelos de NLP requiere grandes cantidades de datos de alta calidad.
- Sesgo: Los modelos de NLP pueden reflejar sesgos presentes en los datos con los que fueron entrenados.

Los chatbots basados en NLP son una tecnología en constante evolución con un gran potencial para mejorar la interacción entre usuarios y máquinas.

3.2.1 CHATBOTS BASADOS EN LLM

Los chatbots fundados en LLM son una nueva generación de chatbots que utilizan 'modelos de lenguaje grandes' (LLM) para generar respuestas más espontáneas y atractivas. Parten de arquitecturas de redes neuronales profundas entrenadas con grandes cantidades de datos de texto, lo que permite comprensión y generación de lenguaje humano con un alto nivel de precisión.

Ventajas de los chatbots cimentados en los LLM:

- Respuestas más naturales y atractivas: Pueden generar respuestas que son más fluidas, coherentes y relevantes que las de los chatbots tradicionales.
- Mejor comprensión del lenguaje natural: Son capaces de comprender mejor el significado de las consultas de los usuarios, incluso si son ambiguas o incompletas.
- Capacidad de aprendizaje: Por último, estos pueden aprender de sus interacciones con los usuarios y mejorar su rendimiento con el tiempo.

Desafíos para este tipo de chatbots:

- Costo: Costosos de entrenar y ejecutar por la cantidad de recursos necesarios.
- Sesgo: Existe la posibilidad de que puedan reflejar sesgos presentes en los datos con los que fueron entrenados.
- Seguridad: Pueden ser utilizados para generar contenido malicioso a pesar de poder limitar temas.

3.2.2 MODELOS FUNDACIONALES LLM

LLM o Modelos de Lenguaje de Grande, son redes neuronales profundas que han sido entrenados en enormes cantidades de texto. Estos modelos aprenden las complejidades del lenguaje humano, como la gramática, el significado y hasta ciertos aspectos del contexto. Es importante mencionar los diferentes parámetros, funciones y datos que influyen en el comportamiento del modelo y que pueden condicionar sus capacidades finales.

Los datos de entrenamiento constituyen el pilar fundamental sobre el cual se erigen los LLM. Estos conjuntos de datos, vastos y diversos, actúan como base de aprendizaje para los algoritmos, proporcionándoles la información necesaria para adquirir las habilidades lingüísticas que los caracterizan. A través de un proceso iterativo de exposición y ajuste, los LLM extraen patrones, estructuras y significados del lenguaje natural. Es por ello la importancia de los datos usados:

- Cantidad: Los LLMs requieren grandes cantidades de datos, generalmente terabytes o petabytes. Estos datos son textos, en diferentes idiomas, en función de los que se pretenda que el LLM pueda hablar. Cuanto más grande sea el conjunto de datos, mayor será la capacidad del modelo para generar texto coherente y relevante.
- Variedad: Los datos deben ser diversos en cuanto a temas, estilos y géneros para un mejor rendimiento y así evitar que el modelo pueda especializarse en un tema. Este conjunto de datos puede provenir de grandes colecciones de libros digitalizados, artículos de noticias, enciclopedias, foros y redes sociales y hasta código fuente. Con esto se consigue que la respuesta pueda adaptarse a casi cualquier ambiente.
- Calidad: Los textos deben ser coherentes, gramaticalmente correctos y representativos del lenguaje natural. Datos de baja calidad pueden introducir sesgos y errores en el modelo.

Parámetros adicionales:

- Tokens: Unidades básicas de información que el modelo procesa, como palabras, subpalabras o caracteres. El tamaño del token afecta la granularidad del procesamiento del lenguaje.
- Parámetros: Variables que el modelo aprende durante el entrenamiento. Un mayor número de parámetros permite al modelo aprender relaciones más complejas.

Parámetros relacionados con la generación de texto:

- Temperatura: Controla la creatividad del modelo durante la generación de texto. Una temperatura alta genera texto más creativo y variado, pero también puede ser menos preciso.
- Top K: Limita el número de tokens que el modelo considera durante la generación de texto. Un valor de K alto reduce la diversidad del texto generado, pero lo hace más preciso.
- Beam Search: Técnica que busca la mejor secuencia de tokens posible durante la generación de texto.
- Nucleus Sampling: Técnica que da más peso a los tokens más probables durante la generación de texto.

Otras técnicas avanzadas para conseguir comportamientos más precisos y respuestas más personalizadas pueden ser:

- Dropout: Técnica para evitar que el modelo se sobreajuste a los datos de entrenamiento. El dropout consiste en desactivar aleatoriamente algunas neuronas durante el entrenamiento.
- Embedding: Técnica para convertir palabras en vectores numéricos que el modelo puede procesar. Los embeddings se utilizan para representar el significado semántico de las palabras.
- Normalización: Técnica para evitar que los valores de las activaciones de las neuronas se vuelvan demasiado grandes o demasiado pequeñas.

Tamaño de ventana:

- Define la cantidad de contexto que el modelo considera al procesar una palabra o frase. Un tamaño de ventana más grande permite al modelo tener una mejor comprensión del contexto.
- El tamaño de ventana puede ser fijo o variable, y su elección depende de la tarea específica para la que se está entrenando el LLM.

Importante comentar la arquitectura de los LLM. Estos pueden ser monolíticos o de "Mixture of Experts" (MoE). A continuación, realizaremos una pequeña comparativa entre ellos para comprender sus diferencias y características:

Los LLM monolíticos se implementan como un único modelo grande, mientras que los LLM "Mixture of Experts" se dividen en varios modelos más pequeños, cada uno con un propósito específico.

Ventajas de los LLM monolíticos:

- Más simples de desarrollar e implementar: No se requiere una compleja orquestación de múltiples modelos.
- Más fáciles de entrenar: Se puede entrenar un solo modelo con un conjunto de datos único.

Desventajas de los LLM monolíticos:

- Menos escalables: Dificultad para aumentar la capacidad a medida que crece la demanda.
- Menos flexibles: Dificultad para adaptar el modelo a diferentes tareas o dominios.
- Más difíciles de mantener: Dificultad para corregir errores o actualizar el modelo.

Ventajas de los LLM "Mixture of Experts":

- Más escalables: Se pueden escalar individualmente los diferentes microservicios.
- Más flexibles: Se pueden adaptar los diferentes microservicios a diferentes tareas o dominios.
- Más fáciles de mantener: Se pueden corregir errores o actualizar individualmente los diferentes microservicios.

Desventajas de los LLM "Mixture of Experts":

- Más complejos de desarrollar e implementar: Se requiere una compleja orquestación de múltiples arquetipos.
- Menos eficientes en cuanto a recursos: Varios modelos consumen más recursos que un solo modelo grande.
- Más difíciles de entrenar: Se requiere entrenar diferentes modelos con diferentes conjuntos de datos.

Dadas estas características, hubiese sido más oportuno el uso de un LLM MoE, para haber adaptado el modelo a diferentes tareas que se demandarán.

3.2.2.1 API DE MODELOS EXISTENTES

Las API a LLM ya funcionales, sean Open Source en plataformas que permiten su uso, o de pago (Close Source), entendiéndose toda la variedad de opciones que se van a comentar en este estudio, pueden usarse para crear chatbots más flexibles y naturales que los apoyados en caminos lógicos. Estas API permiten a los desarrolladores interactuar con LLM grandes para generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a preguntas de forma informativa.

Las API para LLM funcionales tienen una serie de ventajas sobre los sustentados en caminos lógicos. En primer lugar, pueden responder a preguntas o comentarios que no estén previstos en el árbol de decisiones. En segundo lugar, pueden generar respuestas más naturales y relevantes. En tercer lugar, son más fáciles de actualizar y mantener. Por último, al no ser modelos Open Source no es posible realizar un proceso de fine-tunning con la

documentación adecuada, lo que puede aumentar el uso de contexto si se desean proporcionar instrucciones.

A continuación, se presentan algunos ejemplos de API para LLM ya existentes:

- OpenAI GPT-4o API: Esta API permite a los desarrolladores interactuar con GPT-4o, un LLM grande desarrollado por OpenAI. GPT-4o puede generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a preguntas de forma informativa.
- Google AI LaMDA API: En este caso, para interactuar con LaMDA, un LLM grande desarrollado por Google AI. De manera similar LaMDA puede generar texto, traducir idiomas, escribir diferentes tipos de contenido creativo y responder a preguntas de forma informativa.
- Hugging Face Transformers API: Hugging face permite realizar prompts a todos sus modelos disponibles, entre los que se encuentran modelos mencionados.

El uso de alguno de estos modelos sería interesante dadas las enormes capacidades que tienen. Aunque no sea posible realizar ningún entrenamiento sobre estos modelos es probable que con un buen uso del prompt se consigan resultados esperados.

3.2.2.2 MODELOS OPEN SOURCE

Los modelos de código abierto como LlaMa o Vicuña brindan una serie de ventajas sobre los modelos propietarios. En primer lugar, son más accesibles. Cualquier persona puede descargarlos y utilizarlos sin necesidad de pagar una licencia. En segundo lugar, son más transparentes. El código fuente está disponible para que cualquiera lo revise y contribuya. En tercer lugar, son más flexibles. Pueden ser personalizados para adaptarse a diferentes necesidades.

Existen una serie de modelos open Source, de los cuales comenzaremos comparando LlaMa 2, al haber sido de los primeros en liberarse. A medida que los arquetipos de lenguaje continúen desarrollándose, es probable que veamos un aumento en el uso de modelos de código abierto. En febrero 2024, los 5 LLM Open Source más potentes que se plantearon para la realización del proyecto fueron:

- LlaMa 2 70B
- Florecer (bloom)
- MPT-7B
- Halcón
- Vicuña-13B

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
Llama 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
Llama 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	**37.5**	**71.9**	**63.6**	**69.4**	**35.2**	**68.9**	**51.2**	**54.2**

ILUSTRACIÓN 1: BANCO DE PRUEBAS CON DIFERENTES MODELOS

		Time (GPU hours)	Power Consumption (W)	Carbon Emitted (tCO_2eq)
Llama 2	7B	184320	400	31.22
	13B	368640	400	62.44
	34B	1038336	350	153.90
	70B	1720320	400	291.42
Total		3311616		539.00

ILUSTRACIÓN 2: COSTE COMPUTACIONAL DE ENTRENAR LLAMA 2.

Esta figura representa el tiempo utilizado para el entrenamiento total del modelo, utilizando como GPU de referencia una NVIDIA A100, con una VRAM de 80Gb y un precio aproximado de en torno a 12.000€. Con esta referencia podemos llegar a la conclusión de que el proceso de fine-tunning de una manera aproximada también va a ser costoso a nivel computacional.

Sin embargo, debemos replantearnos si es razonable trabajar con un modelo de 70B de parámetros, dado que, para cargarlo de manera completa, sin tamaño reducido, es decir, a 4bytes, serían necesarios 70B x 4bytes = 280Gb de VRAM. No cargarlo entero, o en un tamaño reducido supondría una pérdida de rendimiento y/o velocidad: tanto en funcionamiento como en velocidad de entrenamiento (fine-tune). Con los datos mencionados anteriormente estimar el gasto de un equipo que mantenga este sistema sería exorbitado.

Por lo tanto, el siguiente punto será buscar y encontrar un modelo que pueda mantener un rendimiento similar, reduciendo lo máximo posible el 'peso' de este. En el momento actual, en el que se estaban investigando 'Mistral 7B', modelos muy eficientes que mezclan partes de otra serie de modelos, se ha producido la liberación de un nuevo modelo por parte de Google, 'Gemma 2B' (22/02/2024). Compararemos 'benchmarks', o pruebas de rendimiento en diferentes categorías para ambos, y evaluaremos si son capaces de cumplir con el propósito de este proyecto. Respectivamente, ambos modelos podrían cargarse enteros en una GPU con 28 y 8Gb (~9,5) de VRAM, lo que permite a este último poder ser cargado en casi cualquier tarjeta gráfica actual. No solo el modelo ocupa una cierta cantidad de VRAM por su cantidad de parámetros, además, el número de tokens a procesar es lineal a la memoria física requerida (RAM). Sin embargo, Mistral utiliza una técnica *"Sliding Window Attention"* que permite focalizar el input, y reducir estos recursos [3]. [11]. En todos los modelos de 7 billones de parámetros, Mistral demuestra el mejor rendimiento. Sin embargo,

hay poca documentación acerca de Gemma[2], más allá de las pruebas que podemos realizar nosotros mismos en plataformas como Hugging face. En esta misma plataforma, el canal "Prompt Engineering", en el video "How Bad is Gemma Compared to Mistral?", demuestra de manera práctica la superioridad de Mistral contra Gemma, en la versión de 7B, con preguntas de razonamiento y lógica, incluyendo desde el ámbito matemático hasta el ético.

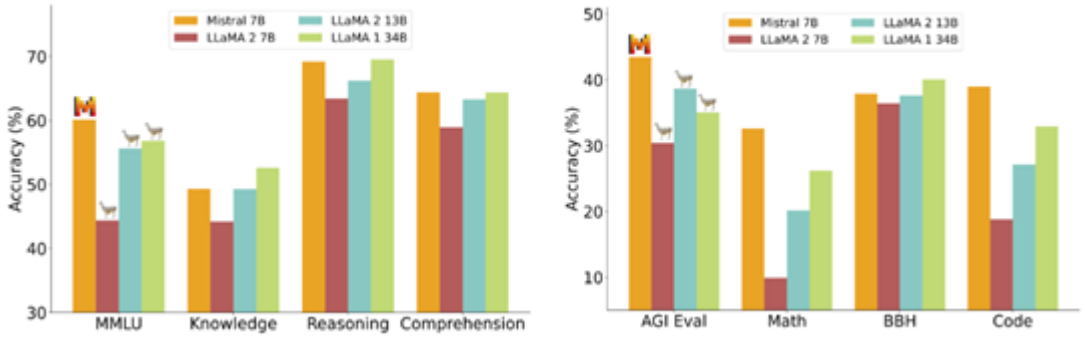

ILUSTRACIÓN 3: COMPARATIVAS DE MISTRAL CON MODELOS DE DIFERENTES TAMAÑOS

Podemos ver como Mistral consigue mejores resultados que modelos más 'pesados', por lo que en este punto podríamos considerar que en el momento actual es el modelo con mejor desempeño.

3.2.3 SOLUCIONES CLOUD IA

El Cloud Computing, también conocido como computación en la nube o simplemente "la nube", es un paradigma de la tecnología de la información que ofrece acceso bajo demanda a recursos informáticos como servidores, almacenamiento, redes, software, bases de datos, análisis e inteligencia artificial a través de Internet.

En términos más simples:

- No necesitas tener tu propia infraestructura: Al contrario que en el pasado, donde las empresas tenían que comprar y mantener sus propios servidores y software, la nube permite "alquilar" estos recursos a un proveedor.
- Pagas por lo que usas: Se utiliza un modelo de pago por uso, lo que significa que solo pagas por los recursos que consumes.
- Acceso desde cualquier lugar: Puedes acceder a tus datos y aplicaciones desde cualquier lugar y en cualquier momento con una conexión a internet.

A pesar de que cuando nos referimos al Cloud, suele ser implícito hacer referencia a los grandes proveedores, hay más tipos de Cloud:

- Nube pública: La infraestructura es propiedad y está gestionada por un proveedor de servicios en la nube, como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform (GCP). Es la opción más económica y escalable, pero ofrece menos control y personalización [12].

[2] https://huggingface.co/chat/settings/google/gemma-7b-it

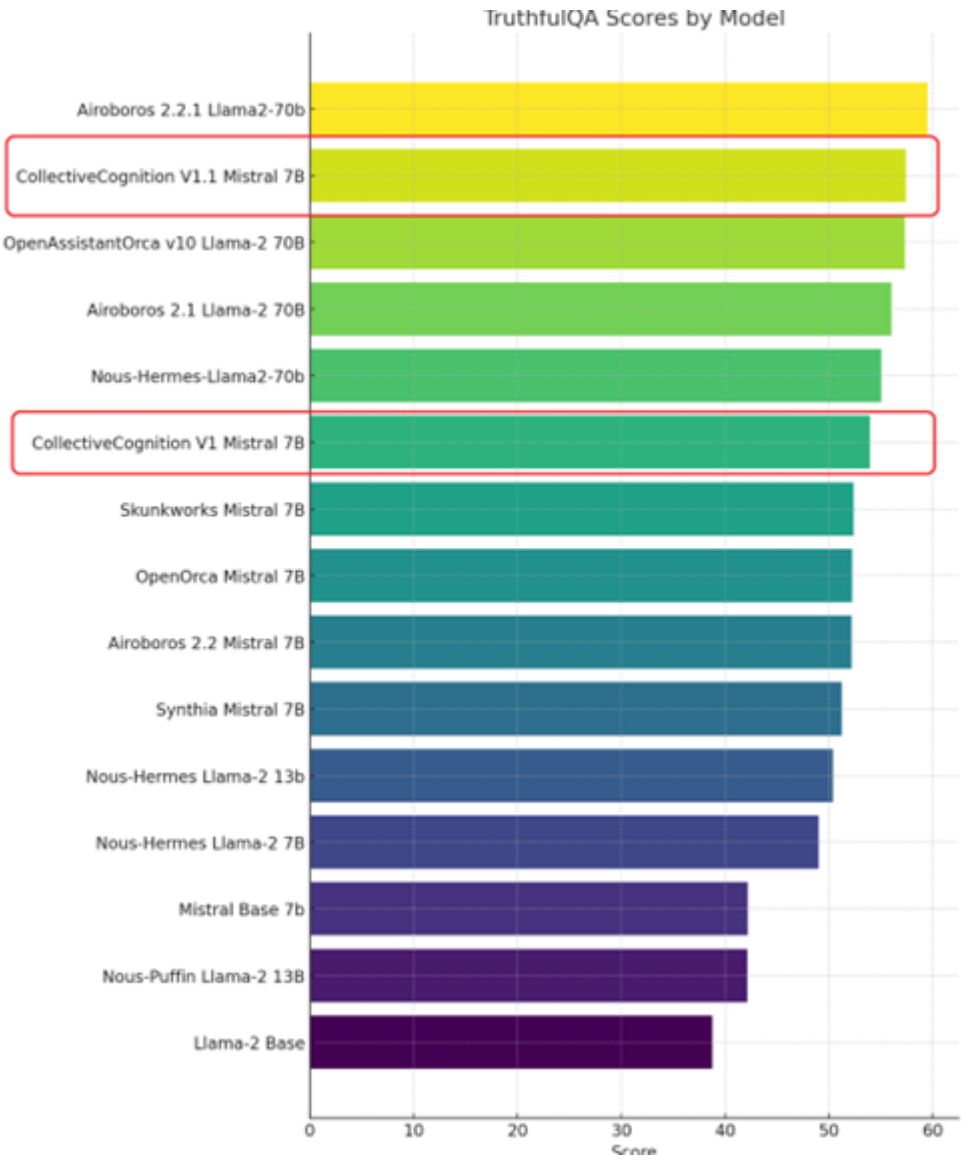

ILUSTRACIÓN 4: PUNTUACIONES DE MISTRAL VS DIFERENTES MODELOS

- Nube privada: La infraestructura es propiedad y está gestionada por la empresa, ya sea en sus propias instalaciones o en un centro de datos externo. Ofrece mayor control y seguridad, pero es más costosa y menos escalable.
- Nube híbrida: Combina la nube pública y la privada para aprovechar las ventajas de ambas. Es una opción flexible que permite adaptar la infraestructura a tus necesidades específicas.

Algunos proveedores Cloud proponen sus propias soluciones IA (siendo los 3 principales las de AWS, Azure y GCP). La tendencia actual apunta a la migración de este tipo de sistemas a Cloud, por diferentes motivos:

1. Ahorro de costes:

- La nube ofrece un modelo de pago por uso, lo que significa que solo pagas por los recursos que consumes.
- Esto puede suponer un ahorro significativo en comparación con la inversión en infraestructura propia.
- No necesitas comprar, mantener ni actualizar hardware o software.

2. Mayor escalabilidad:

- La nube permite escalar recursos de forma rápida y sencilla para adaptarse a las necesidades cambiantes de negocio.
- Puede aumentar o disminuir la capacidad de procesamiento, almacenamiento y memoria en cuestión de minutos.
- Esto ayuda a evitar la sobreinversión en infraestructura y a optimizar el rendimiento.

3. Alta disponibilidad y seguridad:

- Los proveedores de servicios en la nube ofrecen una alta disponibilidad y seguridad para tus datos.
- Sus centros de datos están ubicados en diferentes partes del mundo y cuentan con medidas de seguridad redundantes.
- Esto protege contra fallos de hardware, desastres naturales y otras amenazas.

4. Mayor flexibilidad:

- La nube ofrece una gran flexibilidad para elegir la configuración que mejor se adapte a tus necesidades.
- Puedes elegir entre diferentes tipos de infraestructura, como servidores virtuales, contenedores y plataformas sin servidor.
- Esto permite aprovechar las últimas tecnologías y soluciones innovadoras.

5. Agilidad y rapidez:

- La nube permite implementar nuevos servicios y aplicaciones de forma rápida y sencilla.
- No necesitas esperar a que se instale y configure el hardware o el software.
- Esto ayuda a ser más competitivo y a responder con mayor rapidez a las necesidades del mercado.

6. Facilidad de gestión:

- Los proveedores de servicios en la nube se encargan de la gestión y el mantenimiento de la infraestructura.
- Esto libera de tareas administrativas y permite centrarse en el negocio.
- Puedes acceder a tus datos y aplicaciones desde cualquier lugar y en cualquier momento.

7. Sostenibilidad:

- La nube puede ayudar a reducir la huella de carbono y a ser más sostenible.
- Los proveedores de servicios en la nube utilizan tecnologías eficientes para optimizar el consumo de energía.
- Esto ayuda a contribuir a la protección del medio ambiente.

Como conclusión, la migración a la nube ofrece una serie de ventajas que pueden ayudar a las empresas a ser más eficientes, flexibles, competitivas y sostenibles.

Además de las razones mencionadas anteriormente, la pandemia de COVID-19 ha acelerado la adopción de la nube. El trabajo remoto y la necesidad de acceder a datos y aplicaciones desde cualquier lugar han impulsado la demanda de soluciones en la nube.

Es importante destacar que la migración a la nube no es un proceso simple. Se requiere una planificación cuidadosa para garantizar una transición exitosa. Es importante evaluar las

necesidades, elegir el proveedor adecuado y desarrollar una estrategia de migración adecuada.

3.2.3.1 VERTEX AI GOOGLE CLOUD

Vertex AI es una plataforma de aprendizaje automático (IA) de Google Cloud que ofrece una amplia gama de funciones para ayudar a las organizaciones a desarrollar, entrenar e implementar modelos de IA. Vertex AI incluye funciones para la gestión de datos, el entrenamiento de modelos, la implementación de modelos y la supervisión de modelos.

Algunas de las principales características de Vertex AI incluyen:

- Integración con otros servicios de Google Cloud: Vertex AI se integra con otros servicios de Google Cloud, como Cloud Storage, Cloud SQL y Cloud Functions. Esto permite aprovechar el conjunto completo de herramientas y servicios de Google Cloud para sus proyectos de IA.
- Existencia de un repositorio para la subida de documentación, con la que entrenar el modelo
- Soporte para diferentes tipos de modelos: Vertex AI ofrece soporte para diferentes tipos de modelos de IA, incluidos modelos de aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. Esto permite elegir el tipo de modelo adecuado para sus necesidades específicas. [5]

La elección de la plataforma de IA adecuada depende de las necesidades específicas de la organización. Los factores que considerar incluyen:

- Las características y funciones que ofrece la plataforma: Las plataformas de IA ofrecen una variedad de características y funciones. Es importante elegir una plataforma que ofrezca las características y funciones necesarias para el proyecto de IA.
- La integración con otros servicios: Las plataformas de IA se pueden integrar con otros servicios en la nube. Es importante elegir una plataforma que se integre con los servicios que ya utiliza la organización.
- El costo: Las plataformas de IA tienen diferentes modelos de precios. Es importante elegir una plataforma que se ajuste al presupuesto de la organización.

En general, Vertex AI es una plataforma de IA sólida que ofrece una amplia gama de funciones y características. Es una buena opción para organizaciones que buscan una plataforma de IA escalable y flexible, que también podría considerarse para este proyecto. Es necesario tener en cuenta que esta solución requiere un coste, tanto inicial, y sujeto a la preparación del modelo, como en su funcionamiento, de manera mensual en función a los recursos utilizados, que en este caso se calcularían con el número de querys usados. El coste mensual previsto para un modelo de *Vertex AI conversational* en que se esperan una media de 3000 Querys/mensuales, en base a la carga de trabajo vista en el CAU, se estima en torno a 20€/mes ($0.00125 / 1,000 caracteres para Gemini 1.5 Pro) (https://cloud.google.com/vertex-ai?hl=es-419). Sin embargo, estos presupuestos son referidos al modelo genérico sin el 'fine-tunning'. El hecho de entrenarlo conlleva la dedicación de recursos exclusivos para nuestro modelo lo cual engrosa desproporcionadamente el presupuesto. Más adelante se ha explicado que no hará falta hacer uso de esta técnica, aunque Vertex AI no haya sido la opción escogida.

Además, esta plataforma se puede complementar con otras como Dialogflow, también ofrecidas por la misma compañía y comentada anteriormente.

3.2.3.2 AZURE OPENAI SERVICES DE MICROSOFT

Azure OpenAI Service es un servicio de inteligencia artificial ofrecido por Microsoft Azure en colaboración con OpenAI. Este servicio proporciona acceso a través de la API de REST a los potentes modelos de lenguaje de OpenAI, incluyendo las series de modelos GPT-4, GPT-4 Turbo con Visión, y GPT-3.5-Turbo, etc.

Estos modelos pueden adaptarse fácilmente a tareas específicas, como la generación de contenido, el resumen, el reconocimiento de imágenes, la búsqueda semántica y la traducción de lenguaje natural a código. Los usuarios pueden acceder al servicio a través de las API REST, el SDK de Python o la interfaz basada en web en Azure OpenAI Studio.

Además, Azure OpenAI Service ofrece modelos de inteligencia artificial en los lenguajes más utilizados a nivel mundial. También cuenta con medidas de seguridad y cumplimiento robustas, lo que permite desarrollar aplicaciones seguras.

En resumen, Azure OpenAI Service es una plataforma que permite a los desarrolladores crear, entrenar e implementar modelos de IA, proporcionando una serie de herramientas y servicios para desarrollar, entrenar e implementar modelos de IA.

Algunos de los usos principales servicios que ofrece esta plataforma en concreto son:

- IA conversacional: Puedes desarrollar chatbots y asistentes virtuales que pueden interactuar de manera natural con los usuarios.
- Creación de contenido: Los modelos de lenguaje avanzados pueden generar contenido de alta calidad en una variedad de formatos, incluyendo textos, informes, artículos, y más.
- Creación de bases de datos: Puedes utilizar la IA para extraer y organizar información de grandes volúmenes de datos.
- Modelos de codificación e inteligencia artificial de lenguaje: Azure OpenAI Service ofrece modelos de codificación e inteligencia artificial de lenguaje líderes del sector que puedes ajustar a tus necesidades específicas.
- Seguridad y cumplimiento integrados: Azure OpenAI Service cuenta con robustas medidas de seguridad y cumplimiento, lo que permite desarrollar aplicaciones seguras.
- Precios flexibles de pago por uso y rendimiento aprovisionado: Puedes elegir entre un modelo de precios de pago por uso o rendimiento aprovisionado, dependiendo de tus necesidades.

Sin duda, el uso de la API de gpt-4º es la alternativa que mejor se adapta al proyecto al tener unos costes razonables en pago por uso. Aunque no existe necesidad de realizar un fine-tunning al modelo, en esta misma plataforma sería posible, pero eso implica el aprovisionamiento de recursos dedicados, que se traducen en 8,2€/h, equivalente a ~6000€ mensuales, más el coste del propio tunning, 117€/hora.

Una última alternativa, que tampoco vamos a necesitar, y que nos permitiría indexar información para las respuestas del modelo, habría sido generar embeddings con Azure AI Search. El coste de esta elección supondría un plan de pago de 70€/mensuales. Más adelante se ha explicado cómo se ha diseñado la solución sin fine-tunning ni embeddings.

4 ENTORNO TECNOLÓGICO

El equipo preparado inicialmente por el departamento para el desarrollo del trabajo equipa un Ryzen 9 5900X como procesador principal, con 32GB de memoria RAM DDR4 @3200MHz, un almacenamiento flash de 512GB NVME y una tarjeta gráfica RX6600 con 8GB de VRAM. El equipo se entregó con Windows 11 como sistema operativo instalado, aunque se optó por una distribución de Ubuntu Server (22.04 LTS) y el despliegue de un Jupyter-lab para el trabajo con Python3.

El equipo se configura para poder acceder desde fuera del lugar principal de trabajo. Para esta tarea, es fundamental seleccionar una contraseña robusta. Así se procede a la habilitación de escritorio remoto, por defecto en puerto 3389. universidad, es necesaria la configuración de la VPN (a través de Fortinet). Dado que RDP[3] no permite funciones WOL[4] es necesario que el equipo no alcance el estado de suspensión, por ello y para evitar malgastar energía en dicho puesto se apaga la pantalla y se activa un modo ahorro de energía, deshabilitando el apagado.

La primera iniciativa elegida antes de ser descartada fue Copilot Studio. Esta plataforma ya está pensada para el desarrollo de chatbots, incluyendo un repositorio para subir documentación, secciones que permiten generar diferentes tipos de temas, de maneras guionizadas, e incluso incluyendo automatizaciones y acciones con otras plataformas y su implementación mediante diversos canales. Pero después se explica que esta elección finalmente no cumplía los requerimientos del proyecto.

Se han utilizado recursos y endpoints de Azure. En concreto se ha utilizado Azure OpenAI para la creación de una implementación de GPT-4º con la que se ha trabajado. Este ha sido finalmente el modelo sobre el que se ha basado el proyecto

5 ANÁLISIS

El análisis es la fase necesaria para comprender la perspectiva del problema. Se pretendía crear un sistema que proponga textos de respuesta para solucionar incidencias reales de usuarios, incidencias que, para diagnosticarlas, es necesario realizar una serie de comprobaciones previas. Aunque cada problema se trata de una manera totalmente única, en este proyecto se han tomado 3 ejemplos puntuales y se va a tratar de demostrar que, para añadir nuevos casos, no es necesario reprogramar el código, sino que basta con añadir en un fichero de texto las instrucciones en lenguaje natural acerca de su resolución, para que sea el propio modelo el que las interprete. Con esto se consigue una escalabilidad real para el proyecto y su posible implementación final, más allá de esta demostración.

No se pretende solventar cada problema en un único prompt. Será un proceso iterativo que añade información al contexto en cada paso hasta que sea posible generar una respuesta al problema. En general, los pasos necesarios para la resolución de un problema son los siguientes:

- Extracción de los datos identificativos proporcionados por el usuario.

[3] *RDP (Remote Desktop Protocol) es el protocolo que permite acceso por escritorio remoto.*
[4] WOL (Wake-On-Lan) Permite despertar o arrancar un dispositivo enviando un 'paquete mágico', que contiene 16 veces la dirección MAC del mismo.

- Determinación del tipo de problema.
- Inicio de fase de resolución.
- Comprobación de los datos necesarios para la resolución de la fase de resolución del problema
- Invocación de los servicios externos de utilidad para obtener la información de contexto del problema.
- Uso de esos datos para realizar una comprobación en las bases de datos de la universidad.
- Interpretación del estado del problema en base a los datos obtenidos, estado de los servicios, e instrucciones proporcionadas.
- Avance a siguientes fases de resolución.
- Finalmente, se presenta un texto formateado para ser enviado al usuario. El texto puede ofrecer una solución o pedir nuevos datos para repetir el proceso.

5.1 CASO DE USO

En la fase de análisis el CAU seleccionó un caso de uso representativo del tipo de incidencias que tienen que resolver a diario. A partir de la secuencia esperada de resolución se ha diseñado el sistema asistido por LLM.

5.1.1 MENSAJE PROPORCIONADO POR EL USUARIO

```
Pedro López <pedro.lopes@gmail.com>
Buenos días.
Mi nombre es Pedro López y tengo un problema con el acceso al campus virtual.
He olvidado mi contraseña y sin embargo cuando accedo a
https://ldapapps.uva.es/gestionClave/ para tratar de recuperarla me dice que
no existen datos con mi usuario. Me gustaría que me ayudaseis para poder
acabar la preinscripción en el máster. Mi id es e789456123x, y mi correo este
desde el que escribo: pedro.lopes@gmail.com
Un saludo
Muchas gracias, Pedro
```

5.1.2 PROCEDIMIENTO SEGUIDO POR EL CAU

Del siguiente mensaje el operador del CAU debe ser capaz de extraer el contexto proporcionado por el usuario que, en este caso, está formado por los datos:

- Problema: **Recuperación de contraseña**.
- Mensaje de error: "**No hay datos con mi usuario**"

En función de este diagnóstico inicial, se puede iniciar una ruta específica para resolver el problema. Estos problemas están documentados en el CAU en unos documentos internos que se mantienen actualizados.

Los operadores del CAU tienen acceso al directorio de usuarios por lo que pueden completar los datos identificativos a partir del id que el usuario ha proporcionado.

- Nombre: Pedro López
- Username: PedroLopez
- UID: e789456123x

- Correo externo (opcional): pedro.lopes@gmail.com
- Correo interno (opcional): NA

Si en la resolución del problema hiciesen falta datos que no estuviesen mencionados en el mensaje se respondería al usuario con un mensaje solicitándolos.

En general, todo problema se resuelve con una serie iterativa de comprobaciones y obtención de datos faltantes.

Para codificar estos procedimientos, en este proyecto se propone utilizar un formato de texto plano en lenguaje natural que describa el curso de la resolución.

Por ejemplo, para el problema "**Recuperación de contraseña**" se pueden definir la serie de pasos disponibles para diagnosticar en un guion de soluciones, como se describe en el apartado 6.3.1.

5.1.2.1 DIAGNÓSTICO 1: FUNCIONAMIENTO DE LOS SERVICIOS.

Se deben de diagnosticar para este caso que los servicios LDAP y campus virtual están funcionando correctamente. Estos son los servicios descritos en las instrucciones para dicho problema. El modelo responderá con un archivo json que contiene las URL mencionadas en las instrucciones para la fase de resolución identificada, y el código de control las usará para realizar las peticiones y obtener los códigos de estado que serán añadidos al contexto en la siguiente iteración. El modelo interpretará los códigos de estado.

- **Códigos incorrectos:** Se deben interpretar los códigos de la comprobación y dar una respuesta al usuario. Por ejemplo, '503' Internal Server error. Servicio caído.
- **Códigos correctos:** Los códigos de estado O.K. son '200'. Leer los atributos devueltos en el JSON, añadirlos al contexto del problema y proceder al siguiente paso del diagnóstico.

5.1.2.2 DIAGNÓSTICO 2: CONSULTAS EN BASES DE DATOS.

En este punto que ya se ha descartado que el problema fuese debido al estado de algunos de los servicios se debe proceder obteniendo información relevante del usuario. Es el modelo el que lo descarta al interpretar que los códigos son correctos (200). En este caso se buscarán el perfil en LDAP, y algún dato extra como la situación de la persona (PAS, PDI, estudiante, alta temporal, etc). Por ejemplo, el operador del CAU ejecutaría la siguiente petición: https://trots.es/api-v2.0/users/tfg-api.php/?user=PedroLopez

El operador observa las respuestas de los servicios y añade los atributos al contexto del problema.

5.1.2.3 RESOLUCIÓN: MENSAJE AL USUARIO

Consideremos un ejemplo en el que el problema radica en que el correo conocido por el sistema (según se devuelve en las llamadas a los endpoints) y el utilizado por el usuario no coinciden. Este caso es relativamente habitual y puede deberse a que en el momento del registro inicial se cometió un error tipográfico, que no se recuerde el mail usado, o en el caso de alumnos de nuevo ingreso, que haya sido algún familiar el que realizó el trámite usando su dirección. A continuación, se muestra la respuesta que el CAU suele generar para este caso:

```
Estimado Pedro López,
```

```
Gracias por ponerte en contacto con el Centro de Atención al Usuario de la
Universidad de Valladolid.

Hemos revisado tu perfil de usuario y hemos encontrado que hay una
discrepancia en los correos electrónicos. Según nuestros registros, el correo
registrado es "pedro.lopez@gmail.com". Para resolver este problema, dado que
tu caso es el de una alta temporal, te pedimos que envíes un correo a
soporte@uva.es adjuntando una foto de tu DNI para verificar tu identidad. Una
vez recibida y verificada la identificación, te proporcionaremos un enlace
para corregir esta información.


Si necesitas más información o tienes alguna otra duda, no dudes en ponerte
en contacto con nosotros.


Atentamente,

Centro de Atención al Usuario (CAU)

Universidad de Valladolid
```

Este es el comportamiento deseado para el asistente basado en LLM.

6 DISEÑO DEL PROYECTO.

Tras el análisis de como el problema ha sido enfocado toca pasar a la fase de diseño. Durante esta fase se exploró la alternativa de Copilot Studio y se vio que no satisfacía los requisitos. A continuación, se ha explicado cómo se ha desarrollado el proyecto, con el requisito de la creación de una API que simulará ser una base de datos de la universidad. El proyecto cuenta de 3 partes como se refleja en la Ilustración 5: Diseño del proyecto

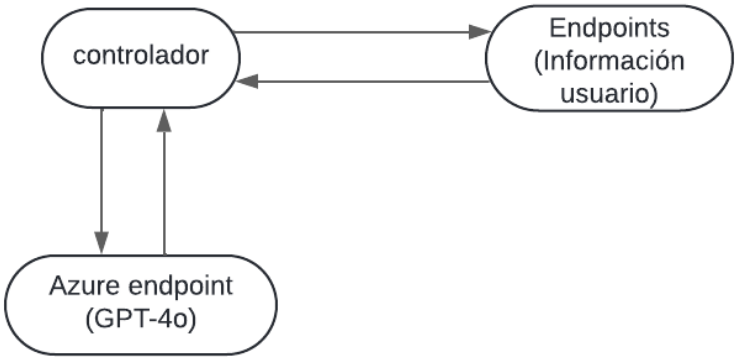

ILUSTRACIÓN 5: DISEÑO DEL PROYECTO

Estas partes son el LLM y las bases de datos de usuarios, que se relacionan a través del controlador.

El LLM es el corazón de este proyecto. Usando Azure, el endpoint se conectará a una implementación de GPT-4o, cuya función será recibir el prompt y generar una respuesta.

Haciendo uso de la ingeniería del prompt, explotaremos la capacidad del modelo para cumplir instrucciones y redactar respuestas de una manera natural.

El acceso a las bases de datos de usuarios se realizará mediante un servicio REST. La función de este servicio es devolver un archivo JSON con la información del usuario solicitado. Se realizará mediante el método GET y usando el atributo 'user'.

El controlador actuará como intermediario, recibirá el mensaje del usuario para componer el prompt, analizará la respuesta, usará los datos extraídos para realizar consultas en la base de datos, leerá instrucciones, añadirá todo este contexto para volver a componer prompts, y devolverá la respuesta final.

6.1 COMPONENTES DEL SISTEMA

El sistema se ha diseñado siguiendo una arquitectura cliente-servidor, separando el frontend del backend para facilitar futuras implementaciones.

- El backend del sistema ha sido implementado utilizando Flask, un framework web ligero y versátil para Python. La API backend implementa el controlador del proyecto y se encarga de una serie de pasos: recibir el problema del usuario, enviarlo una primera vez al LLM, solicitando al este modelo que genere un documento JSON con los datos que el usuario ha proporcionado, posteriormente con el problema extraído se leerán las instrucciones correspondientes, donde será necesario comprobar servicios y posteriormente realizar una consulta a una base de datos. Añadiendo al contexto (instrucciones del problema) el estado de los servicios y la información obtenida de la base de datos mencionada, el modelo será capaz de identificar la cuestión del problema (puede haber situaciones excepcionales no contempladas, que podrían añadirse posteriormente), leyendo las instrucciones y los datos necesarios, y devolver una respuesta apropiada.

- Como componente central de nuestro sistema, hemos integrado el modelo de lenguaje GPT-4o de OpenAI en la plataforma Azure. Al configurarlo en 25k tokens por minuto (El contexto del modelo es superior pero se limita de esta manera para no generar demanda excesiva), hemos ampliado significativamente su capacidad para mantener y procesar información a largo plazo ya que el modelo puede mantener un contexto de conversación grande, lo que le permite generar respuestas más coherentes y relevantes a medida que la interacción evoluciona. Durante el ciclo iterativo el contexto va a crecer por lo que es necesario disponer de un modelo que pueda soportarlo.

- Endpoints: Estos servicios REST, exponen un punto de acceso a una base de datos de usuarios. El servicio procesa la petición y devuelve un objeto JSON con los datos del usuario solicitado. Ha sido generada para este proyecto como una API en (https://trots.es/api-v2.0/), que simularía ser el LDAP utilizado por el CAU.

6.2 API'S Y LDAP

Durante el desarrollo y pruebas del proyecto se implementó un simulador de endpoints REST para emular los servicios que, en un entorno de producción, proporcionarán la información necesaria al sistema.

Estos endpoints se desplegaron en una IP pública accesible desde el controlador del chatbot. Para ello se ha utilizado la CDN de Cloudflare en un dominio personal que cuenta con certificado de seguridad y por lo tanto utiliza un canal seguro HTTPS. Esta API REST se ha desarrollado en PHP.

Los servicios toman un parámetro 'user' para identificar el usuario. Se han creado 5 usuarios de prueba con datos falsos. La URL para la API, con un usuario de ejemplo es el siguiente:

https://trots.es/api-v2.0/users/tfg-api.php/?user=JuanGomez

https://trots.es/api-v2.0/colectivo/tfg-api.php/?user=JuanGomez

Un ejemplo del formato JSON que devolverá esta API con datos de usuario será:

```
{
      "nombre": "Juan Gómez Rodríguez",
      "email": "juan.gomez@email.com",
      "ID": "e345678901X "
 }
```

Por supuesto, en un entorno real de explotación, los endpoints que se usarán serán los que realmente realizan las consultas a los sistemas de la universidad.

Los endpoints que se utilizan se especificarán en lenguaje natural en el documento de resolución de incidencias cuya estructura veremos en 6.3.1. por lo que el sistema es totalmente adaptable a otros casos reales.

6.3 DISEÑO DEL SISTEMA

El objetivo es aprovechar al máximo las capacidades de generar texto predictivo del LLM para reducir el código tradicional al mínimo, agilizando el desarrollo y aumentando la flexibilidad del sistema

Para ello se ha diseñado un sistema basado en un controlador iterativo que utiliza a un LLM para tomar todas las decisiones de acuerdo con una descripción textual del procedimiento. Este controlador se encargará de suministrar al LLM prompts cuidadosamente diseñados, delegando en él la toma de decisiones en cada etapa del proceso.

La clave de este enfoque reside en la capacidad del LLM para comprender y ejecutar instrucciones complejas a partir de prompts bien formulados. Mediante la ingeniería del prompt, lograremos que el modelo no solo genere texto, sino que realice tareas específicas como estructurar datos o tomar decisiones lógicas.

6.3.1 DOCUMENTO DE RESOLUCIÓN DE PROBLEMAS

La redacción del documento es una parte fundamental en la resolución del problema. Aunque se describe en lenguaje natural, lo que permite a gente sin conocimientos en programación contribuir a la resolución de nuevos problemas, debe redactarse cuidadosamente. De lo contrario es fácil que el modelo pueda entrar en bucle o no ser capaz de seguir las instrucciones.

```
Fase: estado de servicios.

Se deben comprobar si los siguientes servicios están funcionando
correctamente:

- Campus virtual: https://campusvirtual.uva.es/

- LDAP: https://ldapapps.uva.es/

- APPS STIC: https://apps.stic.uva.es/preinsmaster/

Si algún servicio no funciona ('fases comprobadas'), salta a la fase
'resolución' y redacta el mensaje al usuario comentando que se trata de un
```

```
problema técnico temporal. Si todos los servicios anteriores funcionan,
pasamos a la fase 'comprobación 1'


Fase: comprobación 1

realizar las siguientes consultas de datos:

- perfildeusuario: https://trots.es/api-v2.0/users/tfg-
api.php/?user={username}

- diagnostico1: https://trots.es/api-v2.0/colectivo/tfg-
api.php/?user={username}

sustituye los parámetros que fuesen necesarios


Fase: resolución.

Redactar una respuesta para el usuario como CAU Universidad de valladolid con
el diagnóstico:

Si no existe identificador en perfildeusuario, el problema es que el usuario
no pertenece a la UVa.

Si el colectivo es estudiante y en diagnostico1 hay 0 créditos matriculados
el problema es que el usuario no está matriculado en ninguna asignatura.

Si el colectivo es profesor y diagnostico1 es negativo el problema es que el
profesor no está registrado en el el POD adecuadamente y tiene que consultar
con la secretaría administrativa de su departamento.

Si ninguna de las causas contempladas ha generado el problema responde que se
investigará el problema.

Responde de manera concisa con el motivo del problema como si estuvieses
contestando al usuario.
```

Es importante por ello observar que hay una serie de marcadores de texto que se mencionan internamente como puede ser:

- Secciones de resolución llamadas fases.
- Comprobaciones a realizar en determinadas URLs.
- Condiciones para cambio de fase.
- Condiciones en caso de no encontrar un motivo del problema.
- Instrucciones precisas.

Al momento de añadir el contexto, usar los mismos tokens de la fase correspondiente ayuda al modelo a generar una relación para comprender las instrucciones. (‘perfildeusuario’ ‘diagnostico1’)

6.3.2 ALGORITMO

Cada vez que se procese un mensaje en el sistema, se ejecutará el diagrama de flujo que se ve en la Ilustración 6: Diagrama de flujo del controlador

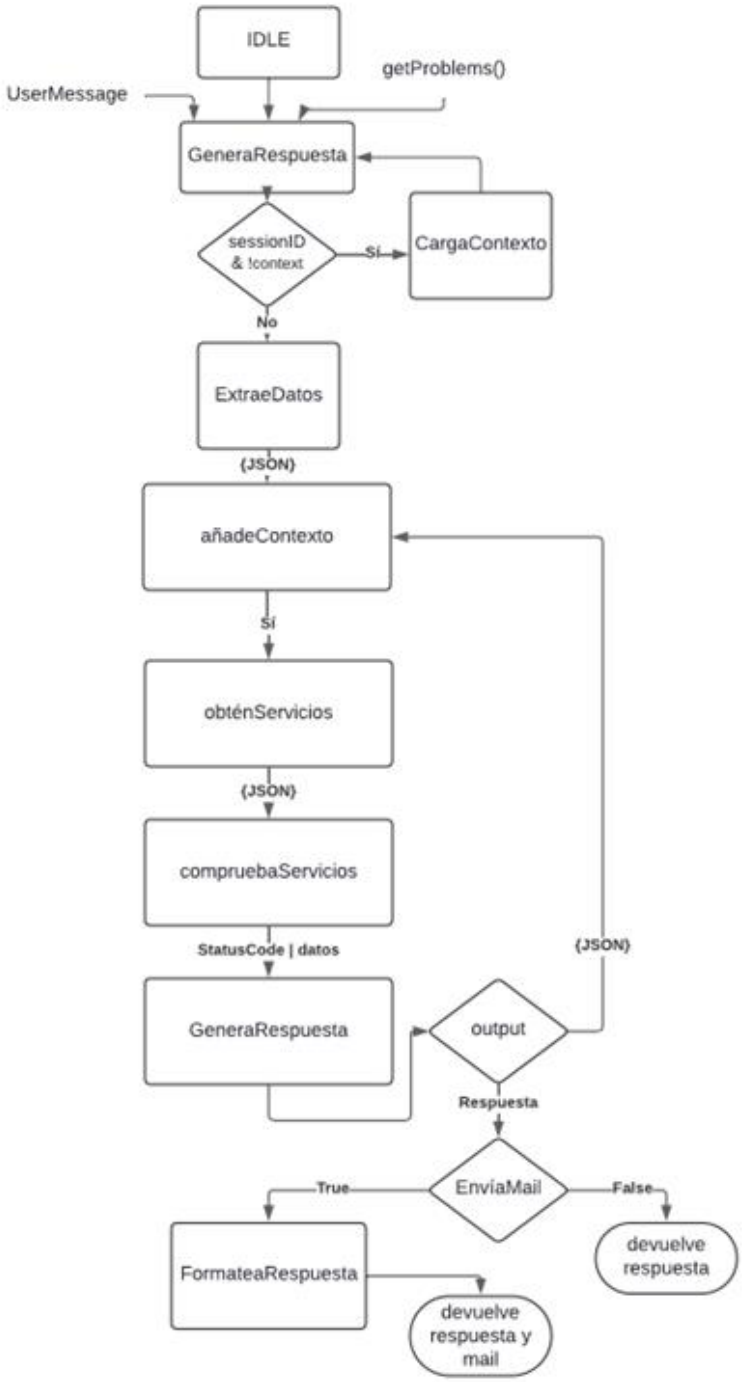

Ilustración 6: Diagrama de flujo del controlador

En éste, podemos diferenciar 3 fases:

6.3.3 FASE 1: ANÁLISIS DEL MENSAJE

En la primera fase se pretende procesar el mensaje mediante el LLM para identificar el tipo de problema que se describe y obtener algunos datos básicos. Primero se obtienen del directorio de configuración la lista de documentos de texto donde se describen los tipos de problemas documentados y su flujo de resolución. El nombre de los documentos de problemas tiene que ser suficientemente descriptivo. Después se confecciona un prompt que contiene la lista de los nombres de los problemas y que pide al LLM que identifique el

problema entre estas opciones. El prompt contiene también unas instrucciones predefinidas para que la respuesta del LLM sea un documento tipo JSON con la información extraída del mensaje del usuario. En todos los casos el prompt es una concatenación del mensaje del usuario, y las instrucciones que buscamos sobre dicho mensaje ('chatrole'):

```
problemas = [archivo for archivo in os.listdir('openai/data/') if
archivo.endswith(".txt")]

chatrole = f"Eres un programador que va a devolver unicamente un archivo json
con información extraída del prompt. Omite incluir ```json... & ```. Necesito
que devuelvas 'Nombre': Contiene nombre y apellido del usuario.'username': Es
'Nombre' formateado, sin tildes ni espacios entre nombre o apellidos y con
las iniciales en mayúsculas. 'mail-interno': correo proporcionado con dominio
uva.es. 'mail-externo': correo proporcionado con un dominio diferente.
'Identificador': tiene el tipo e + NIF. el usuario puede proporcionar NIF o
identificador. 'Problema': Se debe clasificar el problema del usuario en uno
de entre los siguientes tipos {problemas}. 'sesionID': ID de la conversación.
'Datos': si hay información extra que pueda ser relevante para la resolución
del problema añádela a modo de comentario explicativo. Los datos que no se
proporcionen se deben rellenar como 'NA'."
```

La información extraída puede contener los atributos 'nombre', 'username' (nombre formateado), 'id', 'mail_interno'(dominio @uva.es), 'mail_externo', 'problema' (se pedirá al modelo que clasifique entre los propuestos), 'sessionID', y un elemento JSON llamado 'datos' que contendrá información que pueda ser relevante para la resolución del problema. Por ejemplo, en un problema con la tarjeta virtual universitaria, el elemento datos puede adoptar el valor "Tiene iPhone 14" ya que el hecho de que el usuario use un determinado modelo de termnial es relevante. El prompt se ha diseñado para maximizar la probabilidad de que el LLM identifique este tipo de información en el contexto que se está procesando.

```
{
  "Nombre": "Isabel Martín",
  "username": "IsabelMartin",
  "mail-interno": "NA",
  "mail-externo": "isabel.martin@trots.es",
  "Identificador": "NA",
  "Problema": " ",
  "sesionID": " ",
  "Datos": " "
}
```

Si en la respuesta existe un atributo 'sessionID' significa que hubo mensajes previos del usuario que deberían considerarse también, por lo que el controlador, los leerá de su almacenamiento y se utilizarán para añadir información al contexto en el nuevo prompt (que por lo tanto tendrá cada vez un contexto más rico) y se repetirá el procesado del LLM.

6.3.4 FASE 2: ENRIQUECIMIENTO DE CONTEXTO

A continuación, se ejecuta la fase 'añadeContexto' que es donde se va a intentar que el LLM indique al controlador como ejecutar la secuencia de resolución del problema y, finalmente, se generará una respuesta para el usuario en lenguaje natural.

Con el parámetro 'problema' obtenido en la fase inicial, el controlador lee el fichero de texto correspondiente que contiene las instrucciones para diagnosticar y resolver el problema. Al componer el prompt de la primera iteración, ya se incluyen los datos que proporcionó el usuario ('Fases comprobadas'):

```
Fase: estado de servicios.

Se deben comprobar si los siguientes servicios están funcionando
correctamente:

- Campus virtual: https://campusvirtual.uva.es/

- LDAP: https://ldapapps.uva.es/

- APPS STIC: https://apps.stic.uva.es/preinsmaster/

Si algún servicio no funciona ('fases comprobadas'), salta a la fase
'resolución' y redacta el mensaje al usuario comentando que se trata de un
problema técnico temporal.Si todos los servicios anteriores funcionan,
pasamos a la fase 'comprobación 1'



Fase: comprobación 1

realizar las siguientes colsultas de datos:

- perfildeusuario: https://trots.es/api-v2.0/users/tfg-
api.php/?user={username}

- diagnostico1: https://trots.es/api-v2.0/colectivo/tfg-
api.php/?user={username}

sustituye los parámetros que fuesen necesarios



Fase: resolución.

Redactar una respuesta para el usuario como CAU Universidad de valladolid con
el diagnóstico:

Si no existe identificador en perfildeusuario, el problema es que el usuario
no pertenece a la UVa.

Si el colectivo es estudiante y en diagnostico1 hay 0 créditos matriculados
el problema es que el usuario no está matriculado en ninguna asignatura.

Si el colectivo es profesor y diagnostico1 es negativo el problema es que el
profesor no está registrado en el el POD adecuadmaente y tiene que consultar
con la secretaría administrativa de su departamento.

Si ninguna de las causas contempladas ha generado el problema responde que se
investigará el problema.

Responde de manera concisa con el motivo del problema como si estuvieses
contestando al usuario.

Fases comprobadas:

. Datos usuario: Isabel Martín, id:NA, mail interno:NA, mail
externo:isabel.martin@trots.es, username:IsabelMartin.
```

La descripción del proceso de resolución se realiza mediante lenguaje natural con unas pequeñas restricciones de estructura que no interfieren con la interpretación humana. La estructura del documento se ha diseñado teniendo en cuenta que la resolución de un problema puede necesitar comprobaciones de servicios, consultas de datos del usuario, y

posteriormente la interpretación de los datos según dichas instrucciones. Todo esto deberá estar detallado y segmentado de manera lógica en forma de fases de resolución (ver 35). El número de fases no supondrá un problema para la IA ya que el proceso será iterativo.

El algoritmo iterativo generará en cada fase un prompt con las instrucciones del problema y los datos que se han recopilado del usuario hasta ahora.

```
chatrole = f"solo si la fase actual es 'resolución' elabora una respuesta
para el usuario siguiendo las instrucciones como si fueses atención al
usuario de la universidad de Valladolid. En cualquier otro caso, la respuesta
debe de ser un archivo json que contenga un objeto con los servicios que hay
que consultar en dicha fase: la clave es el nombre del servicio y el valor su
url sustituyendo argumentos necesarios si los hubiese. Se sigue este patrón
en función del número de enlaces que se hayan proporcionado. Omite el resto
de instrucciones, se utilizarán en otro momento y omite incluir ```json... &
``. Deduce la fase actual de las comprobaciones proporcionadas al final"
```

Si el problema requiere realizar algún tipo de comprobación, mediante el prompt, se instruye al LLM para que la respuesta sea un documento en formato JSON con la lista de servicios que sean necesarios ejecutar.

```
{
  "Campus virtual": "https://campusvirtual.uva.es/",
  "APPS STIC": "https://apps.stic.uva.es/preinsmaster/"
}
```

Esto se produce porque en las instrucciones se indica que para estar en la fase siguiente de resolución necesita comprobar los estados de los sistemas “APPS STIC” y “Campus Virtual”. Como no existe mención a estos tokens en el contexto, el LLM sigue las instrucciones y genera un JSON con los servicios que están asociados en las instrucciones a esos tokens.

Esto se describe con el fragmento:

```
Fase: estado de servicios.
Se deben comprobar si los siguientes servicios están funcionando
correctamente:
- Campus virtual: https://campusvirtual.uva.es/
- APPS STIC: https://apps.stic.uva.es/preinsmaster/
Si algún servicio no funciona ('fases comprobadas'), salta a la fase
'resolución' y redacta el mensaje al usuario comentando que se trata de un
problema técnico temporal. Si todos los servicios anteriores funcionan,
pasamos a la fase 'comprobación 1'
```

Una vez testeados los servicios por el controlador, se incorporan los datos obtenidos al contexto en el siguiente prompt. Se añadirá a todo lo anterior los nuevos estados de los servicios o la información obtenida de ellos.

```
  Campus virtual: OK
  APPS STIC: OK
```

De esta manera, las instrucciones ahora contendrán:

```
Fase: estado de servicios.
Se deben comprobar si los siguientes servicios están funcionando
correctamente:
- Campus virtual: https://campusvirtual.uva.es/
- APPS STIC: https://apps.stic.uva.es/preinsmaster/
Si algún servicio no funciona ('fases comprobadas'), salta a la fase
'resolución' y redacta el mensaje al usuario comentando que se trata de un
```

```
problema técnico temporal.Si todos los servicios anteriores funcionan,
pasamos a la fase 'comprobación 1'

Fase: comprobación 1
realizar las siguientes colsultas de datos:
- perfildeusuario: https://trots.es/api-v2.0/users/tfg-
api.php/?user={username}
- diagnostico1: https://trots.es/api-v2.0/colectivo/tfg-
api.php/?user={username}
sustituye los parámetros que fuesen necesarios
Fase: resolución.
Redactar una respuesta para el usuario como CAU Universidad de valladolid con
el diagnóstico:
Si no existe identificador en perfildeusuario, el problema es que el usuario
no pertenece a la UVa.
Si el colectivo es estudiante y en diagnostico1 hay 0 créditos matriculados
el problema es que el usuario no está matriculado en ninguna asignatura.
Si el colectivo es profesor y diagnostico1 es negativo el problema es que el
profesor no está registrado en el el POD adecuadmaente y tiene que consultar
con la secretaría administrativa de su departamento.
Si ninguna de las causas contempladas ha generado el problema responde que se
investigará el problema.
Responde de manera concisa con el motivo del problema como si estuvieses
contestando al usuario.

Fases comprobadas:
. Datos usuario: Isabel Martín, id:NA, mail interno:NA, mail
externo:isabel.martin@trots.es, username:IsabelMartin.

 Fases comprobadas:
Campus virtual: OK
APPS STIC: OK
```

Ahora el LLM será capaz de avanzar a la fase 'comprobación 1'. En este punto las instrucciones indican que hay que comprobar los datos del usuario con otros servicios externos "perfildeusuario" y "diagnostico1". El LLM devuelve el siguiente json:

```
{
  "perfildeusuario": "https://trots.es/api-v2.0/users/tfg-
api.php/?user=IsabelMartin",
  "diagnostico1": "https://trots.es/api-v2.0/colectivo/tfg-
api.php/?user=IsabelMartin"
}
```

El resultado de ejecutarlos por el controlador ahora incluirá información del usuario:

```
perfildeusuario: OK {'nombre': 'Isabel Martín Fernández', 'email':
'isabel.martin@trots.es', 'ID': 'e43267757v'}

diagnostico1: OK {'colectivo': 'estudiante'}
```

se añaden de nuevo al contexto y se itera de nuevo.

En cada iteración se le pide al LLM que interprete en qué fase de la resolución se encuentra según el contexto enviado con las instrucciones y el ciclo se repetirá de manera iterativa. De esta manera, el LLM podrá generar los textos siguiendo lo que haya sido descrito en las instrucciones hasta que produzca una respuesta destinada al usuario. El controlador tomará como fin del proceso toda respuesta del LLM que no sea de formato JSON (en realidad se realiza una comprobación heurística verificando si la respuesta ya no empieza por '{').

6.3.5 FASE 3: GENERACIÓN DE RESPUESTA

En esta fase, se le pide al modelo que formatee el cuerpo del mensaje con el estilo y formato deseado para la comunicación con el usuario.

A efectos demostradores, se ha implementado un servicio para que, opcionalmente, el texto generado se envíe por email al destinatario. Para ello, el controlador admite el parámetro "sndml" que si es "true" generará un envío mediante un servidor de correo. En cualquier caso, el controlador siempre devolverá la respuesta del LLM a través de la API.

```
chatrole = "Necesito que al siguiente mensaje le incluyas las etiquetas <br>
en los saltos de línea para formatearlo correctamente. El propio mensaje ya
va a ser el contenido de un mail por lo que es necesario omitir todo lo que
no sea necesario"
```

6.4 PRUEBAS Y VALIDACIÓN

Objetivos de las Pruebas:

- **Verificar:** Que la API desarrollada funcione correctamente y responda a las solicitudes de manera adecuada.
- **Validar:** Que la automatización de respuestas y correos genere mensajes con el formato y contenido esperado.
- **Evaluar:** La eficiencia y el tiempo de respuesta del sistema en diferentes escenarios.

6.4.1 DISEÑO DE LAS PRUEBAS:

- **Pruebas Unitarias:**
    - Se ha cubierto cada función de la API individualmente, verificando los resultados esperados, desde la extracción de información y servicios, sus comprobaciones y su interpretación.
    - Se comprobó la correcta generación de respuestas y correos electrónicos, en el flujo donde el mail se solicitó.
- **Pruebas de Integración:**
    - Se evaluó la interacción entre la API y otros componentes del sistema, como el frontend. Debido a la lentitud para devolver respuestas ha sido necesario el uso de funciones asíncronas y promesas.
    - Diferentes flujos completos han sido probados, desde la resolución total del problema hasta la solicitud de datos al no haber sido proporcionados inicialmente.
- **Pruebas de Usuario:**
    - Se ha simulado una base de datos con 5 usuarios y datos ficticios.

- La interfaz desarrollada consta los elementos interactivos en los que se incluye un contenedor para mostrar mensajes de usuario y repuestas, un input para proporcionar el mensaje y un botón de envío. El uso vía usuario ha sido intuitivo y sencillo.

6.4.2 CASOS DE PRUEBA

Pruebas con datos válidos: Problema de acceso al campus virtual. El usuario (estudiante) no dispone de créditos matriculados, motivo por el que no puede acceder.

Mensaje del usuario:

```
 "Buenos días. Mi nombre es Pedro López y tengo un problema con el acceso al campus virtual. Mi id es e789456123x, correo: pedro.lopes@trots.es. No puedo acceder."
```

La respuesta del chatbot tras comprobar que el campus virtual y los servicios asociados funcionan correctamente, pero que Pedro como estudiante no está matriculado de ningún crédito:

```
"Estimado Pedro López,

Hemos revisado tu caso y hemos detectado que, aunque estás registrado como estudiante en nuestra base de datos, actualmente no tienes ningún crédito matriculado. Te recomendamos que te pongas en contacto con la secretaría de tu centro para regularizar tu situación académica y proceder con la matrícula correspondiente.

Atentamente,
Centro de Atención al Usuario (CAU)
Universidad de Valladolid"
```

Error interno de servidor: En este caso se ha simulado para el mismo mensaje de usuario que existe un problema con un servicio interno donde la respuesta es un error 503. El mensaje que se responde al usuario es:

```
"Estimado/a usuario/a,

Hemos detectado un problema técnico temporal con el servicio del Campus Virtual. Nuestro equipo de soporte está trabajando para resolverlo lo antes posible. Agradecemos su paciencia y comprensión.

Si necesita asistencia inmediata o tiene alguna duda adicional, no dude en ponerse en contacto con nosotros.

Atentamente,
Equipo de Atención al Usuario
Universidad de Valladolid"
```

Problema de recuperación de contraseña: El correo de recuperación no coincide, sea por una errata en el registro inicial, o porque se usó el correo de un familiar que no se recuerda, por lo tanto, el correo de recuperación no se llega a detectar. Mensaje de usuario:

```
"Buenos días. Mi nombre es Isabel Martín. Mi correo isabel.marti@trots.es. He olvidado la contraseña de acceso y cuando intento recuperarla no me llega ningún mail."
```

En este caso, tras comprobar que ambos correos son diferentes, se informa:

```
“Estimada Isabel Martín,

Después de revisar su perfil de usuario y los diagnósticos obtenidos, hemos
encontrado que los correos electrónicos proporcionados no coinciden con
nuestros registros.

Para resolver esta discrepancia, le pedimos que siga los pasos
correspondientes:

1. **Correo institucional (dominio uva.es)**: Por favor, acérquese al
negociado de su centro para corregir esta información.

2. **Altas temporales**: Para solucionar este problema, envíe un correo
electrónico a soporte@uva.es. En el mensaje, adjunte una foto de su DNI. De
esta manera, podremos verificar su identidad y proporcionarle un enlace de
recuperación de sus datos.

Si necesita más ayuda o si hay algo más en lo que podamos asistirle, no dude
en ponerse en contacto con nosotros.

Atentamente,

Atención al Usuario
Universidad de Valladolid ”
```

Problema con la tarjeta virtual (VTUI): Aunque el usuario comenta que ha subido la foto de perfil, ésta aún no ha sido validada por lo que no se puede continuar el proceso. Mensaje de usuario:

```
Mi nombre es Ana Pérez. Tengo un problema con la tarjeta virtual y esque no
puedo usarla. He subido la foto y pagado la tasa. Tengo un iphone. Mi correo
es ana.perez@trots.es
```

La respuesta que obtuvo:

```
Estimada Ana Pérez,

Hemos revisado tu consulta y, según los datos proporcionados, tu tasa está
abonada correctamente. Sin embargo, la validación de tu foto aún no se ha
completado.

Para proceder, es necesario que actualices y valides tu foto en el sistema.
Una vez que esta validación esté completada, podremos continuar con el
estudio de cualquier otra causa relacionada con tu solicitud. Si necesitas
ayuda para validar tu foto o tienes alguna otra consulta, no dudes en ponerte
en contacto con nosotros.

Quedamos a tu disposición para cualquier otra cuestión.

Atentamente,
Atención al Usuario
Universidad de Valladolid
```

6.4.2.1 AUTOMATIZACIÓN DE CORREOS:

Pruebas de envío a diferentes direcciones de correo: Se ha comprobado que el servidor de correo envía a cualquier dirección. Hay que destacar que como existe una redirección con todos los correos con dominio *@trots.es, son los que se han utilizado para las demostraciones. En casos reales el dominio podría haber sido @uva.es, @estudiantes.uva.es o de dominio no institucional.

6.4.3 RESULTADOS Y ANÁLISIS

- **Identificación de errores:** En algunos casos se producía un 'maquetado' en los JSON devueltos por GPT-4o que inducían en error al no poder interpretarse. Se corrigió indicando explícitamente en las instrucciones que no generase ese maquetado.
- **Análisis de rendimiento:** El tiempo de ejecución más largo, en el que el flujo del programa llama 5 veces al modelo de lenguaje y realiza 3 llamadas a servicios produce tiempos de aproximadamente ~5 segundos.
- **Costes:** El coste de la implementación de GPT-4o en Azure OpenAI es de 0,0047€/1.000 input tokens & 0,0141 €/1.000 output tokens. Durante el desarrollo de esta solución las facturas ascendían a un precio total de 2,27 €
- **Conclusiones:** El proyecto tiene un potencial valor para el CAU al ser capaz de proporcionar una respuesta a los problemas contemplados, por un precio infinitesimal y con una capacidad de escalarse a más problemas sin la necesidad de remodelar el código.

7 CONCLUSIONES

El presente trabajo de fin de grado se ha centrado en la exploración e implementación de herramientas de atención al cliente basadas en inteligencia artificial (IA). A través de un análisis exhaustivo de las opciones disponibles, incluyendo modelos open-source, plataformas en la nube y APIs de modelos, se ha llegado a la conclusión de que la IA puede ser utilizada de manera innovadora para automatizar una serie de procesos críticos en el ámbito de la atención al usuario.

A continuación, se detallan las principales conclusiones alcanzadas:

- Automatización de procedimientos: La IA presenta un gran potencial para automatizar tareas repetitivas y de bajo valor añadido, como la extracción de datos de mensajes de usuario, en este trabajo a través de un primer prompt se pedía al modelo que generase un archivo de datos tipo JSON donde los datos no proporcionados se completaban con 'NA'; la comprobación de datos en bases de datos, a través de estos primeros datos extraídos se usaban identificadores para realizar peticiones a una API que contenía información de usuarios para poder contrastarla; o la redacción de procedimientos ante problemas específicos.
- Mejora de la eficiencia y la productividad: La automatización de tareas mediante IA conduce a una mejora significativa en la eficiencia y la productividad del equipo de atención al cliente. Al reducir el tiempo dedicado a tareas repetitivas, los agentes pueden atender a un mayor número de clientes en menos tiempo, disminuyendo los tiempos de espera y mejorando la satisfacción general del cliente.

- Escalabilidad ante nuevos problemas: Un punto a destacar es la gran escalabilidad que presenta este proyecto ante la aparición de nuevos problemas. Gracias a la naturaleza de la IA, no es necesario reprogramar todo el código desde cero para incorporar nuevas casuísticas. En su lugar, el sistema puede aprender y adaptarse a nuevos problemas de forma incremental, con un esfuerzo mínimo. Esto permite a la empresa estar preparada para afrontar nuevos retos de manera rápida y eficiente, sin necesidad de grandes inversiones en desarrollo
- Implementación novedosa: La propuesta de este trabajo de fin de grado radica en la utilización de la IA no solo para la automatización de tareas repetitivas, sino también para la generación de contenido personalizado y la resolución de problemas complejos. Esto permite ir más allá de los chatbots básicos y ofrecer una experiencia de atención al cliente más completa y satisfactoria.

Por lo tanto, el uso de la inteligencia artificial respecto un método más convencional de resolver el problema aporta un valor añadido con los puntos que se acaban de mencionar

8 LIMITACIONES Y LÍNEAS FUTURAS

El bot, a pesar de su potencial, presenta algunas limitaciones inherentes a la tecnología de procesamiento del lenguaje natural y a la arquitectura del propio proyecto. Su desempeño, como cualquier sistema basado en inteligencia artificial, está condicionado por la calidad y cantidad de los datos de entrenamiento y por la complejidad de las tareas asignadas. Además, los modelos de lenguaje, por muy avanzados que sean, pueden generar respuestas imprecisas o incoherentes en situaciones no contempladas. Estas limitaciones, comunes a los sistemas de inteligencia artificial, resaltan la importancia de desarrollar soluciones híbridas que combinen la potencia de la automatización con la flexibilidad y la empatía de la interacción humana. Algunas restricciones en este proyecto pueden ser:

- Dependencia de las peticiones: La realización de todas las comprobaciones a través de peticiones limita la velocidad y eficiencia del bot, especialmente en casos de alta demanda o cuando se requieren interacciones más complejas con los sistemas de la universidad. Esto puede generar demoras en la respuesta y afectar la experiencia del usuario. A su vez, tiene la necesidad de que todas las consultas realizadas a bases de datos sean a través de APIs.
- Contexto limitado de GPT-4o: El límite de contexto de 25k tokens puede ser insuficiente para manejar conversaciones complejas o que requieran recordar información de interacciones previas. Esto puede dificultar la resolución de problemas que involucran múltiples variables o que requieren un seguimiento a largo plazo. A medida que el modelo se siga desarrollando y se consiga más contexto se podrá ir ampliando. El uso estimado actual para resolver un problema siguiendo las fases es de 8k, por lo que ahora no es relevante.
- Precisión en la redacción de instrucciones: La eficacia del bot depende en gran medida de la precisión y claridad de las instrucciones proporcionadas en los archivos .txt. Errores en la redacción pueden llevar a respuestas incorrectas o incompletas por parte del bot, a una incapacidad para avanzar de fase o a la pérdida del estado actual. Aunque este proyecto esté pensado para que gente sin conocimientos técnicos acerca de programación pueda usarlo y ampliar su capacidad de resolver problemas, la precisión de la redacción en las instrucciones puede suponer un problema.
- Escalabilidad de la base de conocimientos: A medida que se añaden nuevas funcionalidades y se amplía la base de conocimientos del bot, la gestión de los

archivos .txt puede volverse cada vez más compleja y propensa a errores si el bot no es capaz de diferenciar claramente el problema del que se trata.

- Falta de aprendizaje automático: Si bien el bot puede acceder a bases de datos de usuarios, no cuenta con mecanismos de aprendizaje automático para mejorar su rendimiento con el tiempo a partir de las interacciones con los usuarios. Esto limita su capacidad para adaptarse a nuevos escenarios o a cambios en los sistemas de la universidad.
- Seguridad: Abrir este sistema directamente al usuario, en vez de ser supervisado por algún trabajador del CAU puede conllevar riesgos en seguridad si no se implantan las medidas necesarias. Por ejemplo, una suplantación de identidad podría relevar datos de un usuario. Este aspecto es crucial porque podría determinar la viabilidad del proyecto.
- Falta de integración en las herramientas de trabajo habituales del CAU: Para una verdadera mejora del proceso, es necesario integrar los textos generados en la herramienta de gestión de tickets.

A pesar de las limitaciones actuales, el potencial de los chatbots para transformar la atención al usuario en las universidades es innegable. Las limitaciones identificadas no son obstáculos insuperables, sino más bien desafíos que estimulan la investigación y el desarrollo de nuevas soluciones. Las líneas futuras de este proyecto explorarán estrategias para superar estas barreras y maximizar el potencial del bot, con el objetivo de ofrecer un servicio de atención al usuario cada vez más eficiente, personalizado y satisfactorio. Algunas de las ideas propuestas contemplan:

- Elección del modelo disponible más potente: A medida que el mundo de la inteligencia artificial avanza, puede que en un futuro la elección óptima ya no sea GPT-4o. Cambiar este modelo por otro, supone un cambio insignificante a nivel de código, y puede conllevar nuevas capacidades.
- Optimización en costos y latencia: Algunas tareas pueden resultar sencillas de realizar, como formatear un texto o generar un json. Buscando reducción de costes, en contraposición al punto anterior, se podría delegar estas tareas a modelos más sencillos y por lo tanto baratos, o incluso funcionando en local.
- Capacidad de diferentes implementaciones: Aunque existe un sistema para acceder a mensajes anteriores (sesionID), puede desarrollarse una base de datos que permita guardar y seleccionar todas las conversaciones para ser desplegadas en el navegador.
- Adjunto de documentación: Para algunas incidencias que requieren adjuntar algún documento, se puede crear un método que lo seleccione e incluir en el envío por correo electrónico.
- Sistemas de autenticación: Muchos de los problemas de seguridad mencionados anteriormente podrían verse solucionados por la implementación de un sistema de autenticación. Esto podría excluir posibles estudiantes de la universidad, por lo que debería de estudiarse su alcance y/o permitir funciones limitadas sin autenticar.
- Algunos de los servicios que necesitan ser comprobados entregan una página diferente cuando el servicio está caído, lo que sigue produciendo un código de estado ‘200’, que se interpreta como correcto. Una prueba futura sería buscar si ese servicio concreto está realmente funcionando.

Existe una amplia variedad de opciones por las que se podría mejorar este bot. Para ello será necesario descubrir matices en su funcionamiento, como costes, latencias, tiempos de uso y detalles de seguridad.

9 BIBLIOGRAFÍA Y RECURSOS

[1] R. Belloso Chacín, W. José Urdaneta Urdaneta, U. Privada Rafael Belloso Chacín, and V. Blanca Liliana González Pertuz, "UNIVERSIDAD PRIVADA," *Ciencias Administrativas y Gerenciales*, vol. 21, no. 2, pp. 147–167, [Online]. Available: https://orcid.org/0009-0009-7788-3036

[2] "4. Introduccion a redes neuronales artificiales".

[3] A. Vaswani *et al.*, "Attention Is All You Need."

[4] E. C. Elejalde -g Navarro -a Rosete -e Leyva *et al.*, "SISTEMA PARA EL ENTRENAMIENTO PARALELO DE REDES NEURONALES DE PROPAGACIÓN HACIA ATRÁS informática."

[5] V. Dogra *et al.*, "A Complete Process of Text Classification System Using State-of-the-Art NLP Models," 2022, *Hindawi Limited*. doi: 10.1155/2022/1883698.

[6] M. Cheng, E. Durmus, and D. Jurafsky, "Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models," Long Papers.

[7] Redacción IA, "Chatbots conversacionales vs. basados en reglas- diferencias clave.," Promptengineer. Accessed: Feb. 15, 2024. [Online]. Available: https://promptengineer.es/chatbots-conversacionales-vs-basados-en-reglas-diferencias-clave/

[8] Redacción IA, "Chatbots conversacionales vs. basados en reglas- diferencias clave.," Promptengineer. Accessed: Feb. 15, 2024. [Online]. Available: https://promptengineer.es/chatbots-conversacionales-vs-basados-en-reglas-diferencias-clave/

[9] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." [Online]. Available: https://github.com/huggingface/transformers/blob/master/

[10] A. Q. Jiang *et al.*, "Mistral 7B," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06825

[11] A. Q. Jiang *et al.*, "Mistral 7B," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.06825

[12] G. Berg and A. Lincke, "Bachelor Degree Project Image Classification with Machine Learning as a Service-A comparison between Azure, SageMaker, and Vertex AI."

[13] G. Berg and A. Lincke, "Bachelor Degree Project Image Classification with Machine Learning as a Service-A comparison between Azure, SageMaker, and Vertex AI."

10ANEXO 1: CÓDIGO

```
import os
from openai import AzureOpenAI
import json
import re
import requests
from flask import Flask, request
import sib_api_v3_sdk
from sib_api_v3_sdk.rest import ApiException
import urllib
import logging
import time


app = Flask(__name__)



@app.route('/petición', methods=['GET'])

def petición():



    logging.basicConfig(filename=os.path.join('logs/',
f'log_{time.strftime("%j_%H%M%S")}.log'), level=logging.INFO,
format='%(asctime)s %(message)s')



     client = AzureOpenAI(
      api_key = os.getenv("AZURE_OPENAI_API_KEY"),
      api_version = "2024-02-01",
      azure_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")

    )

    try:
        msg = request.args.get('msg')
        if not msg:
            raise ValueError("Missing 'msg' parameter")

        prompt = urllib.parse.unquote_plus(msg)
    except Exception as e:
        return f"Error: {e}"

    try:
        sndml = request.args.get('sndml')
        if not sndml:
            sndml=False

    except Exception as e:
        sndml=False
        return f"Error: {e}"



    respuesta = primerprompt(prompt, client)
    print(respuesta + "\n")
```

```
    logging.info(f'MENSAJE DE USUARIO: {prompt} \nRespuesta de GPT-
4o:{respuesta}\n')

    datos_json = json.loads(respuesta)
    Nombre = datos_json["Nombre"]
    SID = datos_json["sesionID"]
    Identificador = datos_json["Identificador"]
    mail_interno = datos_json["mail-interno"]
    mail_externo = datos_json["mail-externo"]
    problema = datos_json["Problema"]
    username = datos_json["username"]
    context = datos_json["Datos"]

    if problema == "NA":
        return "El sistema no puede identificar el problema"


    if SID != "NA":
        logging.info(f'hay SID. Añadiendo contexto')
        with open(f"sesion/{SID}.txt", 'r') as archivo:
            prompt1 = prompt + archivo.read()

        with open(f"sesion/{SID}.txt", 'a') as archivo:
            archivo.write(prompt)
        prompt = prompt1
        logging.info(f'Ambos prompts: \n{prompt}')
        respuesta = primerprompt(prompt, client)
        logging.info(f'Nueva respuesta de GPT-4o: \n{respuesta}')

        datos_json = json.loads(respuesta)
        Nombre = datos_json["Nombre"]
        Identificador = datos_json["Identificador"]
        SID = datos_json["sesionID"]
        mail_interno = datos_json["mail-interno"]
        mail_externo = datos_json["mail-externo"]
        problema = datos_json["Problema"]
        username = datos_json["username"]
        context = datos_json["Datos"]

    else:
        SID = assignSID()
        with open(f"sesion/{SID}.txt", 'a') as archivo:
            archivo.write(prompt)

        logging.info(f'Se ha asignado SID {SID}')


    with open(f"data/{problema}", 'r') as archivo:
        instructions = archivo.read()
        print(instructions)


    chatrole = f" si la fase actual es 'resolución' elabora una respuesta
para el usuario siguiendo las instrucciones como si fueses atención al
usuario de la universidad de valladolid. En cualquier otro caso, la respuesta
```

```
debe de ser un archivo json que contenga una objeto con los servicios que hay
que consultar en dicha fase: la clave es el nombre del servicio y el valor su
url sustituyendo argumentos necesarios si los hubiese. Se sigue este patrón
en función del número de enlaces que se hayan proporcionado. Omite el resto
de instrucciones, se utilizarán en otro momento y omite incluir ```json... &
```.Deduce la fase actual de las comprobaciones proporcionadas"
#-- bucle resolución
    comprobaciones = ""   # se inicializa antes del bucle
    respuesta = "{"   # lo mismo. Si no no entra al bucle.
Cuando no devuelve un json es la respuesta
    for i in range(0,5):

        instructions = f"{instructions}. Datos usuario: {Nombre},
id:{Identificador}, mail interno:{mail_interno}, mail externo:{mail_externo},
username:{username}.comentario: {context}\ \n\n Fases
comprobadas:\n{comprobaciones}" #comentario: {context}\
        logging.info(f'instrucciones ronda {i}:\n{instructions}\n')
        response = client.chat.completions.create(model = "gpt-4o25ktokens",
            messages=[{"role": "system", "content":chatrole},{"role": "user",
"content": instructions}])
        respuesta = response.choices[0].message.content
        print(respuesta + " \nRespuesta: " +str(i))
        logging.info(f'respuesta ronda{i}:\n{respuesta}\n')
        #try:
        if respuesta.startswith('{') is False:
            logging.info(f'Salimos del bucle')
            break
        datos_json = json.loads(respuesta)
        comprobaciones += check_services(datos_json)
        print(comprobaciones)
    print(instructions)

    if respuesta.startswith('{') is True:
        return f"Estimado usuario. \n\nNo ha sido posible resolver su
problema. Seguiremos buscando posibles causas para solucionarlo lo antes
posible. Disculpe las molestias. \n\nsesionID = {str(SID)}\nCentro de
Atención al Usuario (CAU).\nUniverisdad de Valladolid"
#---------------------------------- formatea mail o devuelve respuesta

    if sndml is True:
        chatrole = "Necesito que al siguiente mensaje le incluyas las
etiquetas
 en los saltos de línea para formatearlo correctamente. El
propio mensaje ya va a ser el contenido de un mail por lo que es necesario
omitir todo lo que no sea necesario"
        resp = client.chat.completions.create(model = "gpt-4o25ktokens",
            messages=[{"role": "system", "content": chatrole },{"role":
"user", "content": respuesta }])
        content = resp.choices[0].message.content
        html = f"<p>{content}</p>"
        subject = "Respuesta CAU incidente"
        to_address = mail_interno if mail_interno != "NA" else mail_externo
        receiver_username = Nombre

        logging.info(f'sndml true, se va a formatear respuesta y enviar a
{to_address}')
```
```

```
        email_response = send_email(subject, html, to_address ,
receiver_username)
        print(email_response)
        logging.info(f'{email_response}')
        #return respuesta
    else:
        logging.info(f'sndml false, no se envía correo')

    respuesta += '\n sesionID= ' + str(SID)
    return respuesta  # final de la ejecucción
#----------------------------------- FUNCIONES, FIN EJECUCCIÓN PRINCIPAL
def check_services(json_data):
    results = []
    for service_name, url in json_data.items():
        try:
            response = requests.get(url)
            response.raise_for_status()  # Levanta una excepción si el estado
no es 200
            try:
                json_content = json.loads(response.content)
            except json.JSONDecodeError:
                print("¡¡¡¡¡¡  PRIMERA EXCEPCIÓN. !!!!!!!!")
                json_content = ""
            results.append(f"{service_name}: OK {json_content}\n")
        except requests.exceptions.RequestException as e:
            print("¡¡¡¡¡¡  SEGUNDA EXCEPCIÓN.!!!!!!!!")
            results.append(f"{service_name}: Error\n{e}\n")
    return " ".join(results)
#-------------------------------------------------------------------------
def primerprompt(prompt, client):
    problemas = [archivo for archivo in os.listdir('data/') if
archivo.endswith(".txt")]
    chatrole = f"Eres un programador que va a devolver unicamente un archivo
json con información extraída del prompt. Omite incluir ```json... & ```.
Necesito que devuelvas 'Nombre': Contiene nombre y apellido del
usuario.'username': Es 'Nombre' formateado, sin tildes ni espacios entre
nombre o apellidos y con las iniciales en mayúsculas. 'mail-interno': correo
con dominio uva.es. 'mail-externo': correo con un dominio diferente.
'Identificador': tiene el tipo 'e' + NIF. el usuario puede proporcionar NIF o
identificador. 'Problema': Se debe clasificar el problema del usuario en uno
de entre los siguientes tipos {problemas}. 'sesionID': ID de la conversación.
'Datos': si hay información extra que pueda ser relevante para la resolución
del problema añádela a modo de comentario explicativo. Los datos que no se
proporcionen se deben rellenar como 'NA'. Si no se puede obtener
'Identificador' el problema es faltan_datos.txt"
    response = client.chat.completions.create(model = "gpt-4o25ktokens",
        messages=[{"role": "system", "content": chatrole },{"role": "user",
"content": prompt }])
    return response.choices[0].message.content
#-------------------------------------------------------------------------
def assignSID():
    used = [int(os.path.splitext(archivo)[0]) for archivo in
os.listdir('sesion/') if archivo.endswith(".txt")]
    if len(used) == 0:
        return 1
```

```
    else:
        return max(used) + 1
#-----------------------------------------------------------------------------
def send_email(subject, html, to_address=None, receiver_username=None):
    # SendinBlue mailing parameters
    configuration = sib_api_v3_sdk.Configuration()
    configuration.api_key['api-key'] = os.getenv("BREVO_API_KEY")
    api_instance =
sib_api_v3_sdk.TransactionalEmailsApi(sib_api_v3_sdk.ApiClient(configuration)
)

    subject = subject
    sender = {"name": "Adrián", "email": "contacto@trots.es"}
    html_content = html

    if to_address:
        to = [{"email": to_address, "name": receiver_username}]
    else:
        to = [{"email": "adrian.bernardo@estudiantes.uva.es", "name":
"Adrián"}]

    send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(to=to,
html_content=html_content, sender=sender, subject=subject)

    try:
        # Send the email
        api_response = api_instance.send_transac_email(send_smtp_email)
        print(api_response)
        return {"message": "Email sent successfully!"}
    except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
#-----------------------------------------------------------------------------

if __name__ == '__main__':
    app.run(host="0.0.0.0")
```