

Universidad de Castilla-La Mancha
Escuela Superior de Ingeniería Informática

Trabajo Fin de Grado
Grado en Ingeniería Informática
Tecnología Específica de Computación

Desarrollo de un Tutor Virtual basado en ChatGPT para la Asignatura de Sistemas Inteligentes en la ESII-AB

Iván Fernández García

Julio, 2024

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Tecnología Específica de Computación

Desarrollo de un Tutor Virtual basado en ChatGPT para la Asignatura de Sistemas Inteligentes en la ESII-AB

Autor: Iván Fernández García

Tutores: José Antonio Gámez Martín, José Miguel Puerta Callejón

Julio, 2024

A mis padres,
por su apoyo incondicional

Resumen

Sistemas Inteligentes es una asignatura obligatoria que se imparte en el tercer curso del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería Informática de Albacete (ESII-AB). Su objetivo es que los alumnos sean capaces de resolver problemas reales a través de diferentes algoritmos y técnicas de inteligencia artificial. Algunos de los campos que cubre son los problemas de búsqueda, el aprendizaje por refuerzo, los problemas de optimización combinatoria o los problemas de clasificación supervisada.

Como en todo proceso de aprendizaje, es habitual que los estudiantes encuentren dificultades para comprender el temario y los ejercicios. A pesar de poder solicitar tutorías o intentar resolver sus dudas a través de la información disponible en internet y recurriendo a otros compañeros de clase, el alumno no siempre cuenta con un tutor humano capaz de resolverle las dudas al instante y de brindarle la atención que necesita a cualquier hora, adaptándose a su ritmo de aprendizaje.

Actualmente, los continuos avances en la inteligencia artificial han permitido que surjan grandes modelos de lenguaje como los GPTs, en los que se apoya ChatGPT. Este es capaz de interactuar con los humanos en lenguaje natural de forma eficiente, y se puede utilizar como herramienta en infinidad de sectores, entre ellos la educación. Sin embargo, no siempre proporciona las respuestas deseadas, y en muchas ocasiones estas son insuficientes para el caso de uso que el usuario desea explotar. Por lo tanto, ha surgido la posiblidad de, a partir de él, crear nuevos modelos que estén especializados en temas concretos y sean capaces de resolver tareas específicas de forma más precisa.

En este Trabajo Fin de Grado se va a crear un asistente basado en ChatGPT que actúe como un tutor virtual para la asignatura de *Sistemas Inteligentes*. Este deberá ser capaz de responder a las preguntas de los estudiantes de acuerdo con el temario, así como intentar resolver los tipos de ejercicios que se plantean en la asignatura, detallando lo máximo posible los cálculos y proporcionando las explicaciones correspondientes.

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres por su constante apoyo, paciencia y confianza. Tengo la gran suerte de haber crecido rodeado de sus valores, lo que me ha permitido sacar lo mejor de mí y poner pasión en todo lo que hago. Además, siempre me han brindado las oportunidades que he necesitado para seguir avanzando. Sin ellos no sería ni de lejos la persona que soy hoy en día.

También le quiero dar las gracias a mis tutores en este trabajo, José Antonio Gámez Martín y José Miguel Puerta Callejón, por haberme dado la gran oportunidad de trabajar con ellos y por haberme ofrecido su ayuda siempre que la he necesitado.

Quiero seguir agradeciendo a todas y cada una de las personas que han pasado por mi vida a lo largo de estos últimos años, tanto por los buenos momentos vividos como por las lecciones aprendidas. Gracias a mis amigos y a todos aquellos que me han acompañado desde el inicio hasta el fin de este viaje.

Por último, estoy agradecido con mi *yo* del pasado por no haberse rendido en ningún momento. Me he demostrado a mi mismo que los resultados llegan con esfuerzo, constancia y dedicación. He comprendido que trabajar durante años sin recompensa es parte del proceso, porque el éxito empieza cuando dejas de tenerle miedo al fracaso.

Índice general

Glosario 1

1 Introducción 5

1.1 Motivación 6

1.2 Objetivos 6

1.3 Competencias 7

1.4 Estructura del trabajo 7

2 Estado del arte 9

2.1 Temario de *Sistemas Inteligentes* 9

2.1.1 Introducción 10

2.1.2 Búsqueda en el Espacio de Estados 10

2.1.3 Búsqueda con Adversario 12

2.1.4 Aprendizaje por Refuerzo 13

2.1.5 Optimización Combinatoria y Metaheurísticas 14

2.1.6 Aprendizaje Automático y Clasificación Supervisada 16

2.2 Tutores Inteligentes 17

2.2.1 Tutor inteligente VS Tutor humano 17

2.2.2 Evolución de los tutores inteligentes 19

2.2.3 Tutores inteligentes en la actualidad: LLMs 21

2.3 GPT-series 22

2.3.1 OpenAI 23

2.3.2 Modelos GPT 23

2.3.3 ChatGPT 26

2.4 Creación de tutores virtuales basados en ChatGPT 27
2.4.1 GPTs personalizados con ChatGPT *28*
2.4.2 OpenAI API *29*
2.4.3 Creación de chatbots e integración de la API de OpenAI *34*

3 Desarrollo del proyecto 37

3.1 Estudio de tecnologías y opciones 37
3.2 Estimación de costes 39
3.3 Selección de herramientas y tecnologías 42
3.4 Desarrollo de la aplicación 45
3.4.1 Arquitectura del proyecto *45*
3.4.2 Creación del chatbot e integración de Assistants API *46*
3.4.3 Integración del chatbot en una aplicación web *49*
3.5 Análisis de requisitos 53
3.5.1 Necesidades del usuario *53*
3.5.2 Conocimiento e instrucciones *57*
3.6 Optimización del comportamiento del asistente 58
3.6.1 Cuestiones teóricas *60*
3.6.2 Ejercicios prácticos *64*
3.7 Resultados y comparativa con ChatGPT 71

4 Conclusiones del proyecto 85

4.1 Conclusiones y futuras mejoras 85
4.2 Competencias desarrolladas 89

A Metodología utilizada 91

B Planificación del proyecto 93

C Recursos utilizados 95

Referencia bibliográfica 101

Índice de figuras

2.1 Modelo de transición de Markov 14
2.2 Construcción de un árbol de decisión a partir de datos 16
2.3 Efectividad de ChatGPT VS Tutor humano 18
2.4 Frecuencia con la que se recurre a un tutor humano tras ChatGPT . . 19
2.5 Arquitectura tradicional de un Sistema de Tutoría Inteligente 20
2.6 Evolución de la serie de modelos GPT desarrollados por OpenAI 23
2.7 Arquitectura base de los modelos GPT 24
2.8 Rendimiento de GPT-4 respecto a GPT-3.5 en exámenes académicos . 26
2.9 GPTs más populares 28
2.10 Asistente de trenes en tiempo real utilizando Assistants API 33
2.11 Cómo funciona Assistants API 34
2.12 Creación de un chatbot con Voiceflow 35
2.13 Integración de un chat web con Voiceflow 36

3.1 Diagrama de la arquitectura del proyecto 45
3.2 Diagrama de flujo de la lógica conversacional del chatbot 46
3.3 Implementación del chatbot en Voiceflow 47
3.4 Icono de ChatGPT-SSII 50
3.5 Estructura de la aplicación web que permite la interacción con el chatbot 50
3.6 Interfaz de ChatGPT-SSII tras iniciar una conversación 51
3.7 Respuesta generada por ChatGPT-SSII 51
3.8 No es posible crear un hilo para iniciar la conversación 52
3.9 No es posible generar una respuesta 52

A.1 Diagrama de flujo de la metodología utilizada 92

B.1 Diagrama de planificación del proyecto 94

Índice de tablas

2.1 Temario de *Sistemas Inteligentes* 9
2.2 Estrategias de búsqueda 12
2.3 Eventos relevantes en la evolución de los ITSs 20

3.1 Costes de crear un GPT personalizado con ChatGPT 39
3.2 Costes de crear un agente de IA (chatbot) basado en un GPT con Voiceflow 40
3.3 Costes de utilizar Chat Completions API con un modelo base sin ajustar 40
3.4 Costes de aplicar *fine-tuning* y utilizar Chat Completions API con el modelo ajustado 41
3.5 Costes de utilizar Assistants API 42
3.6 Conjunto inicial de instrucciones (I) 59
3.7 Conjunto inicial de instrucciones (II) 60
3.8 Instrucciones adicionales para cuestiones teóricas (I) 60
3.9 Instrucciones adicionales para cuestiones teóricas (II) 61
3.10 Instrucciones tras la optimización de la parte teórica 63
3.11 Instrucciones adicionales para ejercicios prácticos (I) 64
3.12 Instrucciones adicionales para ejercicios prácticos (II) 65
3.13 Instrucciones adicionales para ejercicios prácticos (III) 67
3.14 Instrucciones generales tras la optimización de la parte práctica: Identificación del tipo de ejercicio y resolución posterior 68
3.15 Instrucciones específicas tras la optimización de la parte práctica: Problemas de Búsqueda en Juegos 68
3.16 Instrucciones específicas tras la optimización de la parte práctica: Problemas de Búsqueda en el Espacio de Estados 69
3.17 Instrucciones específicas tras la optimización de la parte práctica: Problemas de Clasificación Supervisada 69
3.18 Instrucciones específicas tras la optimización de la parte práctica: Problemas de Aprendizaje por Refuerzo 70
3.19 Instrucciones específicas tras la optimización de la parte práctica: Optimización Combinatoria y Metaheurísticas 70

3.20 Consultas para la evaluación de los resultados 72
3.21 Respuestas generadas para la primera consulta 73
3.22 Respuestas generadas para la segunda consulta 74
3.23 Respuestas generadas para la tercera consulta 75
3.24 Respuestas generadas para la cuarta consulta 76
3.25 Respuestas generadas para la quinta consulta 77
3.26 Respuestas generadas para la sexta consulta 78
3.27 Respuestas generadas para la séptima consulta 79
3.28 Respuestas generadas para la octava consulta 80
3.29 Respuestas generadas para la novena consulta 81
3.30 Respuestas generadas para la décima consulta 82

Índice de algoritmos

2.1 Búsqueda en árboles 11
2.2 Búsqueda en grafos 11
2.3 Búsqueda Alfa-Beta 13
2.4 Q-learning 14
2.5 Hill Climbing 15
2.6 Genético básico 16

Índice de listados de código

2.1 Ejemplo de llamada a Chat Completions API 30
2.2 Extracción de la respuesta con Chat Completions API 30
2.3 Formato de los ejemplos de entrenamiento para *fine-tuning* con GPT-3.5 Turbo 31
2.4 Formato de los ejemplos de entrenamiento para *fine-tuning* con modelos anteriores a GPT-3.5 Turbo 32

3.1 Acceso a la API de OpenAI y al asistente 47
3.2 Inicio de la conversación con el asistente 47
3.3 Generación de la respuesta por parte del asistente 48
3.4 Procesamiento de los mensajes de texto y archivos generados 49

Glosario

Algoritmo: Conjunto de instrucciones bien definidas, ordenadas y finitas que permiten realizar una actividad mediante pasos sucesivos.

API (Application Programming Interface): Conjunto de funciones y procedimientos que permiten la creación de aplicaciones que acceden a las características o datos de un sistema operativo, aplicación u otro servicio.

Aplicación web: Software que se ejecuta en un navegador web y proporciona funcionalidades interactivas a los usuarios.

Aprendizaje automático: Subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos que permiten a los ordenadores aprender y tomar decisiones basadas en datos.

Aprendizaje profundo: Técnica de aprendizaje automático que utiliza redes neuronales con múltiples capas para modelar representaciones de datos con altos niveles de abstracción.

Aprendizaje no supervisado: Área del aprendizaje automático en la que un modelo se entrena utilizando datos no etiquetados para encontrar patrones o estructuras ocultas.

Aprendizaje por refuerzo: Área del aprendizaje automático en la que un agente aprende a tomar decisiones mediante la realización de acciones y la recepción de recompensas o penalizaciones.

Aprendizaje supervisado: Área del aprendizaje automático en la que un modelo se entrena utilizando datos etiquetados para hacer predicciones o clasificaciones.

Árbol de búsqueda: Estructura de datos en forma de árbol utilizada en algoritmos de búsqueda, donde cada nodo representa un estado posible del problema y las aristas representan las acciones o decisiones que conducen de un estado a otro.

Árbol de decisión: Modelo de clasificación o predicción numérica utilizado en aprendizaje automático que representa decisiones y sus posibles consecuencias en una estructura de árbol.

Chatbot: Programa de ordenador diseñado para simular conversaciones humanas, utilizado comúnmente en aplicaciones de servicio al cliente.

ChatGPT: Sistema desarrollado por OpenAI, basado en los modelos GPT y optimizado para generar texto similar al humano en conversaciones.

ChatGPT-SSII: Tutor virtual para la asignatura de *Sistemas Inteligentes* basado en ChatGPT.

Clasificación supervisada: Técnica de aprendizaje automático que asigna etiquetas a nuevas observaciones basándose en un conjunto de datos etiquetado durante el entrenamiento.

Conjunto de entrenamiento: Subconjunto de datos utilizado para entrenar un modelo de aprendizaje automático.

Conjunto de prueba: Subconjunto de datos utilizado para evaluar el rendimiento de un modelo de aprendizaje automático después del entrenamiento.

Conjunto de validación: Subconjunto de datos utilizado para ajustar los hiperparámetros de un modelo de aprendizaje automático y prevenir el sobreajuste.

DALL-E: Modelo de generación de imágenes desarrollado por OpenAI que crea imágenes a partir de descripciones textuales.

Despliegue: Proceso que consiste en poner una aplicación en funcionamiento en un entorno de producción.

Diagrama de flujo: Representación gráfica de un proceso o algoritmo que utiliza símbolos y flechas para mostrar el flujo de control y la secuencia de pasos.

Feedback: Retroalimentación proporcionada por los usuarios o sistemas para mejorar el rendimiento de un modelo o aplicación.

Fine-tuning: Proceso realizado para ajustar un modelo preentrenado a través de un conjunto de datos específico, mejorando su rendimiento en una tarea particular.

Framework: Conjunto de herramientas y bibliotecas que proporcionan una estructura específica para el desarrollo de aplicaciones.

Grafo dirigido: Estructura de datos que consiste en un conjunto de nodos conectados por aristas, donde cada arista tiene una dirección asociada, indicando un sentido único desde un nodo origen hacia un nodo destino.

Grafo no dirigido: Estructura de datos que consiste en un conjunto de nodos conectados por aristas, donde las aristas no tienen dirección, es decir, la relación entre los nodos es bidireccional.

GPT (Generative Pre-trained Transformer): Familia de modelos de lenguaje desarrollados por OpenAI, basados en la arquitectura Transformer y entrenados en grandes corpus de texto.

Heurística: Estrategia de búsqueda que utiliza reglas prácticas para encontrar soluciones aproximadas a problemas complejos.

HTML (HyperText Markup Language): Lenguaje de marcado utilizado para crear y estructurar el contenido en la web.

IA (Inteligencia Artificial): Rama de la informática que se ocupa del desarrollo de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana.

Interfaz de usuario: Conjunto de elementos gráficos y funcionales a través de los cuales un usuario interactúa con una aplicación o sistema.

ITS (Intelligent Tutoring System): Sistema de tutoría basado en inteligencia artificial que proporciona instrucción y *feedback* personalizados a los estudiantes.

JavaScript: Lenguaje de programación utilizado principalmente para crear contenido interactivo en páginas web.

JSON (JavaScript Object Notation): Formato ligero de intercambio de datos que es fácil de leer y escribir para los humanos y fácil de parsear y generar para las máquinas.

JSONL (JSON Lines): Formato de archivo en el que cada línea es un objeto JSON independiente, comúnmente utilizado para grandes conjuntos de datos.

LLM (Large Language Model): Modelo de lenguaje de gran tamaño, entrenado en grandes volúmenes de texto, capaz de comprender y generar texto en lenguaje natural.

Lógica conversacional: Conjunto de reglas y estructuras utilizadas para gestionar y dirigir el flujo de una conversación, especialmente en el contexto de chatbots y sistemas de diálogo automatizados, con el objetivo de mantener interacciones coherentes y efectivas.

MDP (Markov Decision Process): Marco matemático utilizado para modelar la toma de decisiones en situaciones donde el entorno es observable.

Metaheurística: Algoritmo de búsqueda de soluciones generales diseñado para resolver problemas de optimización complejos.

Middleware: Software que actúa como intermediario, facilitando la comunicación y gestión de datos entre aplicaciones y sistemas.

Minería de datos: Proceso que consiste en extraer patrones y conocimiento útil de grandes volúmenes de datos.

NLP (Natural Language Processing): Campo de la inteligencia artificial que se enfoca en la comprensión, interpretación y generación del lenguaje natural por parte de las máquinas.

Nodo: Elemento fundamental de un grafo, que puede representar un objeto o entidad.

OpenAI: Organización de investigación en inteligencia artificial dedicada al desarrollo de IA beneficiosa para la humanidad. Conocida por desarrollar ChatGPT y modelos avanzados de lenguaje natural como GPT-3.5 y GPT-4.

Parámetro: Variable en un modelo de aprendizaje automático cuyo valor es aprendido durante el entrenamiento.

Prompt: Entrada inicial o pregunta que se proporciona a un modelo de lenguaje para generar una respuesta.

Prompt engineering: Proceso que consiste en diseñar y optimizar prompts para obtener respuestas más precisas y útiles de un modelo de lenguaje.

Python: Lenguaje de programación de alto nivel conocido por su simplicidad y legibilidad, ampliamente utilizado en ciencia de datos, aprendizaje automático, automatización de tareas y desarrollo web.

Repositorio: Lugar de almacenamiento centralizado donde se mantiene y gestiona el código fuente y otros recursos de desarrollo de software.

Sobreajuste: Fenómeno en aprendizaje automático donde un modelo se ajusta excesivamente a los datos de entrenamiento, capturando tanto patrones reales como ruido, lo que resulta en un mal desempeño en datos nuevos.

TTS (Text-to-Speech): Tecnología que convierte texto escrito en discurso hablado.

Token: Unidad básica de texto, como una palabra o subpalabra, utilizada en el procesamiento del lenguaje natural.

Voiceflow: Plataforma para diseñar, construir y lanzar asistentes de voz y chatbots sin necesidad de escribir código.

1. Introducción

La educación ha sufrido grandes cambios a lo largo de su historia, existiendo notables diferencias entre sus primeros indicios y cómo la conocemos hoy en día. Antes de que las civilizaciones estuvieran estructuradas, los miembros de diferentes grupos ya se enseñaban unos a otros las técnicas para cazar o elaborar herramientas de manera inteligente. Con el paso de los años llegaron los textos, libros y manuscritos, si bien es cierto que en muchas épocas sólo las élites aspiraban a ella.

Actualmente, la educación es en gran parte virtual y está marcada por las nuevas tecnologías. Leer libros ya no es la única forma de aprender, pues podemos buscar información en internet, ver vídeos en plataformas como *YouTube*, estar al día de lo que ocurre a través de las redes sociales, o escuchar audiolibros y pódcasts, entre muchas otras opciones. Además, el avance de la inteligencia artificial a lo largo de los últimos años ha permitido que emerjan una gran cantidad de poderosas herramientas para ahorrarnos trabajo y aumentar nuestra productividad a la hora de aprender.

La existencia de modelos con los que podemos interactuar en lenguaje natural es ya una realidad. Aunque existe una gran polémica respecto a algunos de los usos que se les puede dar a estas herramientas, han demostrado poder facilitar el proceso de aprendizaje si se usan correctamente. Tener un asistente inteligente capaz de responder de forma eficiente a la mayoría de tus preguntas puede servir de gran ayuda, pero no en todos los casos llevará a cabo la tarea de igual manera, pues existen ámbitos menos conocidos por el modelo o se esperan de él respuestas demasiado concretas.

Por lo tanto, estos modelos pueden ser insuficientes en determinados casos pese a sus increíbles capacidades. Cuando se pregunta sobre un determinado tema, se espera que las respuestas obtenidas se adecúen a la situación en la que nos encontramos. Por ejemplo, si estamos estudiando para una asignatura determinada, nos gustaría que este actuara como un tutor virtual de una forma más específica y precisa. Esto puede significar resolver nuestras dudas de acuerdo con el temario de la misma, o ser capaz de resolver un ejercicio exactamente igual que como lo haría el profesor en clase.

Las capacidades de los Sistemas de Tutoría Inteligentes han cambiado durante los últimos años, pero el objetivo siempre ha sido proporcionar respuestas inmediatas y personalizadas a los estudiantes para guiarlos en su proceso de aprendizaje, de forma análoga al modo en el que lo haría un tutor humano. A día de hoy, herramientas como ChatGPT ya están siendo usadas por los estudiantes para complementar a los tutores humanos. El motivo principal es la flexibilidad y la capacidad de dirigir su modo de aprender a través de sus propios métodos. Además, poder interactuar con un tutor experto que se adapte a casos de uso concretos abre un mundo lleno de nuevas posibilidades, convirtiendo estos modelos en herramientas todavía más poderosas.

1.1. Motivación

El aprendizaje autónomo y las tutorías personalizadas son componentes esenciales del proceso educativo moderno. Sin embargo, no siempre es posible contar con un tutor humano disponible para resolver dudas al momento. Un tutor virtual puede complementar la enseñanza tradicional, ofreciendo respuestas inmediatas, adaptabilidad a distintos ritmos de aprendizaje y disponibilidad a todas horas.

El avance de la inteligencia artificial ha permitido el surgimiento de herramientas capaces de entender a los humanos e interactuar con ellos en lenguaje natural. Una de estas herramientas, ChatGPT, ha mostrado ser particularmente efectiva en diversas aplicaciones. En el ámbito educativo, hay un potencial aún no explotado para el uso de esta tecnología como tutor virtual en asignaturas específicas.

1.2. Objetivos

El objetivo principal de este Trabajo Fin de Grado es desarrollar un tutor virtual basado en ChatGPT adaptado a los contenidos de la asignatura de *Sistemas Inteligentes*, cuyo temario se encuentra resumido en el capítulo 2.1. La implementación se realizará como una aplicación web usando la API de OpenAI.

Por lo tanto, se consideran los siguientes objetivos:

- Evaluar la eficacia de ChatGPT en la identificación y respuesta a dudas específicas de la asignatura.
- Establecer una serie de prompts "tipo" para adecuar las respuestas a las convenciones asumidas en la asignatura de *Sistemas Inteligentes*.
- Optimizar y personalizar el modelo de ChatGPT, utilizando el material de la asignatura y otros archivos que puedan resultar útiles para que se adapte mejor a las necesidades y convenciones establecidas en el curso.
- Realizar una comparativa entre ambas herramientas ChatGPT y ChatGPT-SSII.

1.3. Competencias

Se trabajarán principalmente las siguientes competencias específicas de la tecnología de computación:

- Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.
- Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.
- Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes entornos inteligentes.
- Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

1.4. Estructura del trabajo

El presente Trabajo Fin de Grado está formado por las siguientes partes:

1. **Introducción.** Se presenta el trabajo a desarrollar y se comentan aspectos como la motivación de su realización, objetivos y competencias (Capítulo 1).
2. **Estado del arte.** Se explica brevemente el contenido de *Sistemas Inteligentes*, se analiza la evolución de los tutores inteligentes a lo largo de la historia y se presentan algunas herramientas para su desarrollo a partir de los modelos de lenguaje de OpenAI en los que se basa ChatGPT (Capítulo 2).
3. **Desarrollo del proyecto.** Se crea un asistente que actúe como un tutor virtual para la asignatura de *Sistemas Inteligentes* utilizando la API de OpenAI, capaz de responder preguntas de la asignatura de manera más concreta y precisa. Además, se desarrolla una aplicación web para poder interactuar con el sistema como si fuera ChatGPT, y se realiza una comparativa entre ambos (Capítulo 3).
4. **Conclusiones del proyecto.** Se extraen las conclusiones y las futuras mejoras del proyecto (Capítulo 4).
5. **Apéndice.** Parte complementaria que incluye información adicional sobre la metodología utilizada (Capítulo A), la planificación del proyecto (Capítulo B) y los recursos utilizados (Capítulo C).

2. Estado del arte

A lo largo de este capítulo se va a analizar la evolución de los tutores inteligentes desde su aparición hasta la actualidad, y se van a presentar diferentes herramientas que nos permiten desarrollar un tutor virtual a partir de modelos de lenguaje. Además, se va a explicar brevemente el temario de la asignatura de *Sistemas Inteligentes*.

2.1. Temario de *Sistemas Inteligentes*

Sistemas Inteligentes es una asignatura obligatoria que se imparte en el tercer curso del Grado en Ingeniería Informática en la Escuela Superior de Ingeniería Informática de Albacete (ESII-AB). Su objetivo es que los alumnos sean capaces de resolver problemas reales a través de diferentes algoritmos y técnicas de inteligencia artificial aprendidas en la misma. En la tabla 2.1 se pueden observar los diferentes temas que aborda.

Tema	**Diapositivas**
1. Introducción	44
2. Búsqueda en el Espacio de Estados	80
3. Búsqueda Informada	53
4. Búsqueda con Adversario	62
5. Aprendizaje por Refuerzo	96
6. Problemas de Optimización Combinatoria	49
7. Metaheurísticas - Búsqueda Local	47
8. Metaheurísticas - Algoritmos Genéticos	52
9. Introducción al Aprendizaje Automático	46
10. Clasificación Supervisada	44

Tabla 2.1: Temario de *Sistemas Inteligentes*.

Como objeto del trabajo, es esencial conocer las bases del temario para poder comprender a qué tipo de preguntas debe adaptarse el tutor virtual a desarrollar. El contenido de la asignatura se va a explicar brevemente en seis bloques:

1. **Introducción** (Tema 1)
2. **Búsqueda en el Espacio de Estados** (Temas 2 y 3)
3. **Búsqueda con Adversario** (Tema 4)
4. **Aprendizaje por Refuerzo** (Tema 5)
5. **Optimización Combinatoria y Metaheurísticas** (Temas 6, 7 y 8)
6. **Aprendizaje Automático y Clasificación Supervisada** (Temas 9 y 10)

En esta sección no se pretende abordar profundamente el temario, sino simplemente plasmar las ideas principales de nuestro caso de uso específico.

2.1.1. Introducción

El primer tema sirve como introducción de la asignatura a través del concepto de inteligencia artificial. Se abordan aspectos como su historia, los agentes inteligentes o sus aplicaciones (Sistemas basados en el conocimiento, aprendizaje automático, inteligencia artificial en videojuegos...) [27].

2.1.2. Búsqueda en el Espacio de Estados

El objetivo de este bloque es aprender a identificar problemas que puedan ser resueltos mediante búsqueda, y saber formularlos correctamente. Para ello debe conocerse:

- **El espacio de estados**. Cómo se define un estado y qué representa.
- **El estado inicial**. La búsqueda comienza a partir de él.
- **Las acciones asociadas a cada estado**. A cada estado se le puede aplicar una serie de acciones, y aplicar una acción a un estado da lugar a un nuevo estado.
- **Función de sucesión**. Los sucesores de un estado son los nuevos estados obtenidos al aplicar cada una de las posibles acciones asociadas a él. Si el árbol de búsqueda está formado por nodos y cada uno contiene un estado, los hijos de un nodo serán aquellos que contengan los estados resultantes de aplicar las acciones asociadas al estado que contiene el nodo padre.
- **Test estado final**. Debe comprobarse si ha sido alcanzado, es decir, si el estado actual es final.
- **El coste de la solución**. En los problemas de búsqueda, la solución es una secuencia de acciones para llegar desde el estado inicial hasta el final. Su coste es la suma de los costes de cada una de las acciones por las que está formada.

A lo largo del tema se introducen diferentes estrategias de búsqueda, pero todas ellas comparten el algoritmo 2.1 de búsqueda en árboles, pues su diferencia está en el criterio de prioridad que utiliza cada una para insertar los nodos en la lista de abiertos.

```
1:  función TREE-SEARCH( )
2:      open ← CREATENODE(INITIAL-STATE)                    ▷ Crea la raíz
3:      mientras open ≠ ∅ hacer
4:          node ← EXTRACT(open)
5:          si TESTGOAL(node.STATE) entonces
6:              devolver RECOVERPATH(node)                  ▷ Solución encontrada
7:          successors ← EXPAND(node)
8:          para cada successor en successors hacer         ▷ Procesa cada sucesor
9:              INSERT(successor, open)
10:     devolver failure                                    ▷ No se ha encontrado solución
```

Algoritmo 2.1: Búsqueda en árboles [29].

Es importante evitar visitar estados ya explorados previamente, pues esto podría conducir a bucles infinitos. Para ello se utiliza el algoritmo 2.2 de búsqueda en grafos, en el que se comprueba si un estado ya ha sido visitado previamente.

```
1:  función GRAPH-SEARCH( )
2:      open ← CREATENODE(INITIAL-STATE)                    ▷ Crea la raíz
3:      explored ← ∅
4:      mientras open ≠ ∅ hacer
5:          node ← EXTRACT(open)
6:          si node.STATE ∉ explored entonces                ▷ No considera los ya explorados
7:              si TESTGOAL(node.STATE) entonces
8:                  devolver RECOVERPATH(node)              ▷ Solución encontrada
9:              successors ← EXPAND(node)
10:             para cada successor en successors hacer     ▷ Procesa cada sucesor
11:                 INSERT(successor, open)
12:             INSERT(node.STATE, explored)
13:     devolver failure                                    ▷ No se ha encontrado solución
```

Algoritmo 2.2: Búsqueda en grafos [29].

Además, se presentan varios ejemplos de problemas que pueden ser abordados con esta estrategia, tales como *8-Puzzle* o problemas de grafos dirigidos y no dirigidos.

Las estrategias de búsqueda no informada realizan la exploración sin información adicional [29], mientras que en la búsqueda informada se utilizan heurísticas para estimar cómo de cerca estamos del objetivo [30]. Por ejemplo, en un problema de rutas en el que cada estado es una localización, una heurística podría ser la distancia en línea recta entre el punto en el que nos encontramos y el estado final.

En la tabla 2.2 aparecen las diferentes estrategias de búsqueda y su función de evaluación (a minimizar) a la hora de priorizar nodos, donde $g(n)$ es el coste del nodo n desde el inicio y $h(n)$ es el coste estimado desde el nodo n hasta el objetivo.

Tipo de búsqueda	**Estrategia**	**Función a minimizar**	**Implementación**
No informada	Breadth-First	$f(n) = depth(n)$	*FIFO*
	Depth-First	$f(n) = 1/depth(n)$	*LIFO*
	Uniform-cost	$f(n) = g(n)$	Prioridad
Informada	Best-first	$f(n) = h(n)$	Prioridad
	A*	$f(n) = g(n) + h(n)$	Prioridad

Tabla 2.2: Estrategias de búsqueda [29, 30].

2.1.3. Búsqueda con Adversario

En este tema ya no hay un sólo agente, sino dos o más compitiendo por ganar. En la búsqueda con adversario o en juegos, el estado resultante de aplicar una acción también depende de la respuesta del oponente, el cual es impredecible. Por lo tanto, se tiene en cuenta un movimiento por cada posible acción del contrincante [31].

Se consideran juegos de dos jugadores (MAX y MIN). MAX mueve primero y ambos se van turnando hasta que el juego termina. El valor de un nodo se corresponde con la utilidad de estar en el estado actual, asumiendo que ambos jugarán de forma óptima hasta el final de la partida. El objetivo es encontrar la mejor acción posible.

Sin embargo, el número de estados a explorar es demasiado elevado, pues crece de manera exponencial en función del número de posibles movimientos. La solución es podar las ramas que no influyan en el resultado final, obteniendo la misma solución sin explorar nodos innecesarios. Esto se lleva a cabo introduciendo en Minimax dos parámetros α y β, de forma que la búsqueda se realiza siguiendo el algoritmo 2.3. Alfa-Beta se puede mejorar teniendo en cuenta la profundidad e introduciendo una función heurística que permita evaluar estados sin necesidad de llegar al final de la partida.

```
1: función ALPHA-BETA-SEARCH(state)
2:     v ← MAX-VALUE(state, −∞, +∞)
3:     devolver la a ∈ ACTIONS(state) con valor v

4: función MAX-VALUE(state, α, β)
5:     si TERMINAL-TEST(state) entonces devolver UTILITY(state)
6:     v ← −∞
7:     para cada a in ACTIONS(state) hacer
8:         v ← max(v, MIN-VALUE(RESULT(state, a), α, β))
9:         si v ≥ β entonces devolver v                    ▷ Podar
10:            α ← max(α, v)
11:    devolver v

12: función MIN-VALUE(state, α, β)
13:    si TERMINAL-TEST(state) entonces devolver UTILITY(state)
14:    v ← +∞
15:    para cada a in ACTIONS(state) hacer
16:        v ← min(v, MAX-VALUE(RESULT(state, a), α, β))
17:        si v ≤ α entonces devolver v                    ▷ Podar
18:            β ← min(β, v)
19:    devolver v
```

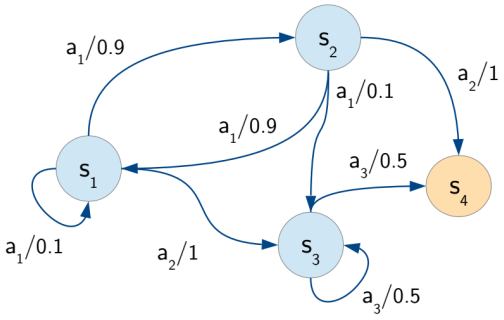
Algoritmo 2.3: Búsqueda Alfa-Beta [30].

2.1.4. Aprendizaje por Refuerzo

El objetivo de este tema es comprender cómo los agentes son capaces de tomar decisiones complejas en entornos deterministas o no deterministas. A través de un sistema basado en recompensas y castigos, aprenden cuál es la mejor acción que pueden tomar en cada estado. Además, en un entorno no determinista existe incertidumbre sobre el resultado de aplicar una acción, por lo que no se puede buscar la mejor secuencia de acciones de manera sistemática. Hay que encontrar una política que conduzca al objetivo y maximice la recompensa [32].

Se introduce el Proceso de Decisión de Markov y las aproximaciones de iteración de valores e iteración de políticas. En la figura 2.1 se puede apreciar un grafo que representa el modelo de transición para un problema determinado, teniendo también cada estado una recompensa asociada y una utilidad en función de su recompensa y las utilidades de los estados que se pueden alcanzar aplicando sus posibles acciones.

Normalmente las transiciones y las recompensas se desconocen, por lo que el problema no se puede resolver con un MDP. Se presenta el aprendizaje por refuerzo y Q-learning (Algoritmo 2.4), siendo necesario interactuar directamente con el entorno.



	s_i			
	s_1	s_2	s_3	s_4
$T(s_1, a_1, s_i)$	0.1	0.9	0	0
$T(s_1, a_2, s_i)$	0	0	1	0
$T(s_2, a_1, s_i)$	0.9	0	0.1	0
$T(s_2, a_2, s_i)$	0	0	0	1
$T(s_3, a_3, s_i)$	0	0	0.5	0.5

Figura 2.1: Modelo de transición de Markov [32].

función Q-LEARNING( )
  Inicializar $\hat{Q}$ aleatoriamente
  mientras *stopCondition* = False **hacer** ▷ Convergencia o iteraciones
    Inicializar s
    repetir
      Seleccionar acción a de acuerdo a $\pi^*(\hat{Q})$
      Ejecutar acción a desde s
      Leer nuevo estado s' y recompensa r
      Actualizar $\hat{Q}(s, a)$ como:
      $\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha \cdot \left(r + \gamma \cdot \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a)\right)$
      $s \leftarrow s'$
    hasta s es final
  devolver Generar política óptima π^* desde $\hat{Q}$

Algoritmo 2.4: Q-learning [32].

2.1.5. Optimización Combinatoria y Metaheurísticas

A lo largo de este bloque se introducen los problemas de optimización combinatoria, y metaheurísticas como la búsqueda local o los algoritmos genéticos.

En los problemas de optimización combinatoria no se explora sistemáticamente el espacio hasta encontrar el estado final, sino que se busca obtener una configuración de parámetros (solución) que optimice un determinado criterio [33].

El espacio de configuraciones es normalmente enorme, por lo que el objetivo no es encontrar la solución óptima, sino una que nos sea de suficiente calidad. Una solución puede representarse de diferente manera (codificación binaria, discreta, permutaciones...) y su calidad se mide maximizando o minimizando una función de evaluación.

Para encontrar una buena solución en un tiempo razonable se introducen las metaheurísticas, que nos permiten explorar las suficientes configuraciones y aprovechar la información obtenida durante la búsqueda para ir explorando cada vez mejores soluciones. Más concretamente, se explica la búsqueda local y los algoritmos genéticos.

Búsqueda local

Los algoritmos de búsqueda local se basan en un criterio de vecindad para realizar la búsqueda, y el más básico es Hill Climbing (Algoritmo 2.5). Este consiste en obtener los vecinos de la solución de la que se parte, moverse al "mejor" vecino en función del criterio de optimalidad, y repetir el proceso hasta que la configuración actual no tenga ningún vecino "mejor" que ella [34].

```
1:  función HillClimbing(initialSolution)
2:      currentSolution ← initialSolution
3:      currentScore ← Evaluate(currentSolution)
4:      improves ← True
5:      mientras improves = True hacer
6:          improves ← False
7:          neighbours ← GenerateNeighbours(currentSolution)
8:          para cada neighbour in neighbours hacer
9:              score ← Evaluate(neighbour)
10:             si score > currentScore entonces          ▷ Maximizar: > | Minimizar: <
11:                 currentSolution ← neighbour
12:                 currentScore ← score
13:                 improves ← True
14:     devolver currentSolution
```

Algoritmo 2.5: Hill Climbing [34].

Existen algoritmos que utilizan Hill Climbing generando varias soluciones iniciales aleatorias (*Random Local Search*), o que repiten el proceso a partir de la mejor configuración encontrada con Hill Climbing en la iteración anterior (*Iterated Local Search*).

Algoritmos genéticos

Un algoritmo genético consiste en ir evolucionando las configuraciones por generaciones, de forma que las soluciones exploradas son cada vez mejores (Algoritmo 2.6). Algunos operadores fundamentales de un algoritmo genético son el cruce y la mutación, así como el criterio de selección utilizado para elegir qué individuos se reproducen. Al final de cada iteración, según un criterio de reemplazo, se forma la siguiente generación combinando las mejores soluciones antes y después de aplicar los operadores [35].

```
1: función GENETIC( )
2:     P ← GENERATEPOPULATION                    ▷ Crea los individuos iniciales
3:     EVALUATE(P )                              ▷ Obtiene la calidad de las soluciones
4:     mientras stopCondition = False hacer
5:         P' ← SELECTPOPULATION(P )             ▷ Elige algunos individuos por calidad
6:         CROSSOVER(P' )                        ▷ Cruza pares de individuos seleccionados
7:         MUTATION(P' )                         ▷ Muta los individuos cruzados
8:         EVALUATE(P' )                         ▷ Obtiene la calidad de las nuevas soluciones
9:         P ← COMBINE(P ∪ P' )                  ▷ Forma la nueva generación mejorada
```

Algoritmo 2.6: Genético básico [35].

2.1.6. Aprendizaje Automático y Clasificación Supervisada

En este último bloque se explica el concepto de aprendizaje automático y se presenta una serie de aplicaciones del mismo, cubriendo sus diferentes paradigmas (Aprendizaje supervisado, no supervisado y por refuerzo) [36].

Además, se introducen los problemas de clasificación supervisada, y más concretamente cómo resolverlos utilizando árboles de decisión [28]. Sin embargo, en los ejercicios sólo se utilizan variables predictoras discretas y el error de clasificación como criterio de partición, en lugar de otras opciones comunes como la ganancia de información con entropía o índice de Gini. Por último, se explica el proceso de poda *a posteriori* utilizando la estimación de Laplace. En la figura 2.2 se puede observar un ejemplo de un árbol de decisión construido a partir de una serie de datos de entrenamiento.

	income	owner	expenses	credit
1	low	no	high	NO
2	low	yes	high	NO
3	average	yes	high	NO
4	average	no	high	NO
5	high	no	high	NO
6	high	yes	high	YES
7	low	no	low	NO
8	average	no	low	NO
9	high	no	low	NO
10	average	yes	low	YES
11	low	yes	low	YES
12	high	yes	low	YES

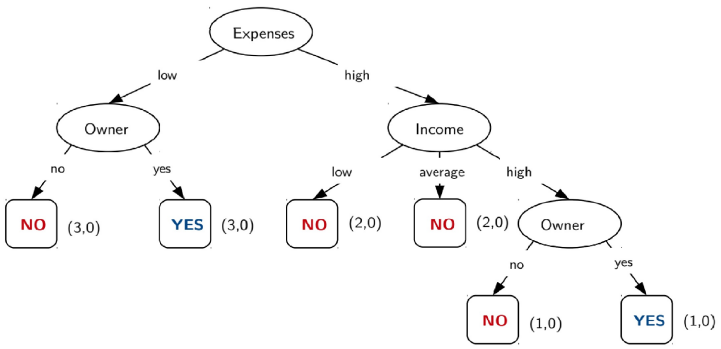


Figura 2.2: Construcción de un árbol de decisión a partir de datos [28].

2.2. Tutores Inteligentes

Los Sistemas de Tutoría Inteligentes (ITSs) son herramientas que utilizan técnicas de inteligencia artificial para ofrecer enseñanza personalizada de manera automática. Tras descubrir que las tutorías personalizadas (uno a uno) ofrecían mejores resultados para el estudiante que la enseñanza en grupo, surgió la posibilidad de crear tutores inteligentes capaces de enseñar de manera individual y efectiva a través de la inteligencia artificial. Para ser capaces de actuar como un tutor humano, el poder de los mismos reside en saber a quién enseñan, qué se debe enseñar y cómo hacerlo [41].

2.2.1. Tutor inteligente VS Tutor humano

El objetivo de un tutor inteligente es actuar de forma análoga a como lo haría un tutor humano, adaptándose al ritmo de aprendizaje del estudiante y resolviendo sus dudas de forma eficiente. A esto hay que añadirle la posibilidad de que su disponibilidad sea mayor que la de un tutor humano, la cual es limitada.

Más allá de que los estudiantes aprendan y desarrollen habilidades por sí mismos, un tutor humano los guía y les ayuda a comprender lo que desconocen. Además, los tutores humanos no siempre ofrecen las soluciones de forma explícita, lo que hace que el estudiante esté motivado y sienta curiosidad por continuar aprendiendo y entender profundamente aquello que se está abordando. La interacción continua y la retroalimentación durante la resolución de problemas y dudas le permite al estudiante sentir que tiene el control en todo momento y que es capaz de dominar el tema, lo que le produce satisfacción y ganas de continuar aprendiendo.

Un tutor inteligente debería poder actuar de tal forma que sea capaz de reemplazar a un tutor humano, pero no es sencillo cubrir de manera eficiente todos los aspectos positivos que este puede brindar a los estudiantes. Para intentar lograrlo, los pasos que toma el estudiante deben ser monitorizados, detectando sus errores y ofreciendo retroalimentación que contenga explicaciones y sugerencias sobre ellos. En caso de que el estudiante no comprenda algo o no sepa cómo proceder, el tutor debe ser capaz de guiarlo adecuadamente y resolver sus dudas de la mejor manera posible.

Desde que los tutores inteligentes comenzaron a utilizarse, siempre han sido una buena herramienta en caso de no poder disponer de una tutoría personalizada. Sin embargo, hasta ahora no habían sido capaces de comenzar a reemplazar a los tutores humanos de forma eficiente. Un estudio realizado en mayo de 2023 en Estados Unidos con estudiantes de entre 16 y 24 años y padres de estudiantes más jóvenes, revela que la gran mayoría de ellos ya está cambiando a los tutores humanos por ChatGPT, especialmente en matemáticas y otras asignaturas de ciencias [1].

Prácticamente todos los estudiantes han incluido el uso de ChatGPT en sus tutorías. La inmensa mayoría lo prefiere antes que un tutor humano, y aseguran que

desde que han comenzado a utilizarlo sus resultados académicos se han visto afectados positivamente. Aquellos que lo consideran más eficiente defienden que les ofrece disponibilidad total y respuestas inmediatas, así como poder estudiar a su propio ritmo y revisar la información y las conversaciones en cualquier momento. Por otra parte, los estudiantes que prefieren los tutores humanos dicen que la comunicación cara a cara es fundamental, ya que el vínculo que se genera entre las dos personas permite un mejor aprendizaje y despierta emociones positivas en los estudiantes.

A pesar de ello, los expertos dicen que ChatGPT todavía no es un reemplazo efectivo para los tutores humanos, porque estos ofrecen ventajas que van más allá del contenido. Sin embargo, sí aseguran que es una gran opción, tanto la herramienta en sí misma como los Sistemas de Tutoría Inteligentes que se desarrollen a partir de ella.

En la figura 2.3 se puede apreciar una gráfica que muestra qué tipo de tutor es más efectivo según los estudiantes de entre 16 y 24 años y los padres de estudiantes más jóvenes, mientras que en la figura 2.4 se muestra la disminución de frecuencia con la que recurren a un tutor humano como resultado de utilizar ChatGPT.

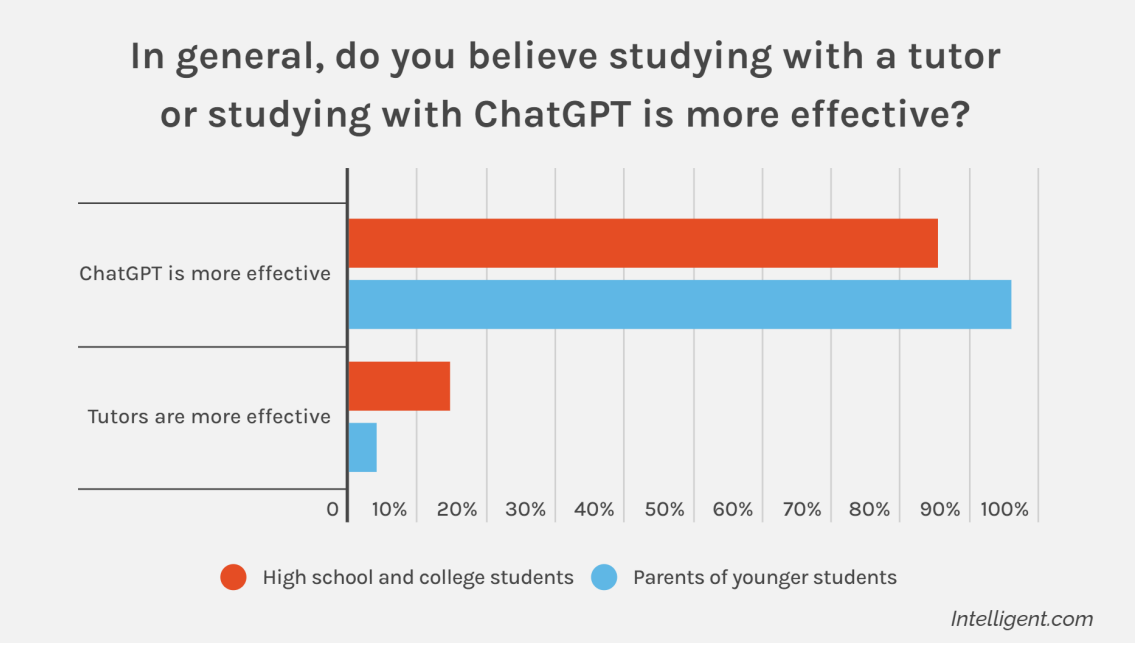


Figura 2.3: Efectividad de ChatGPT VS Tutor humano [1].

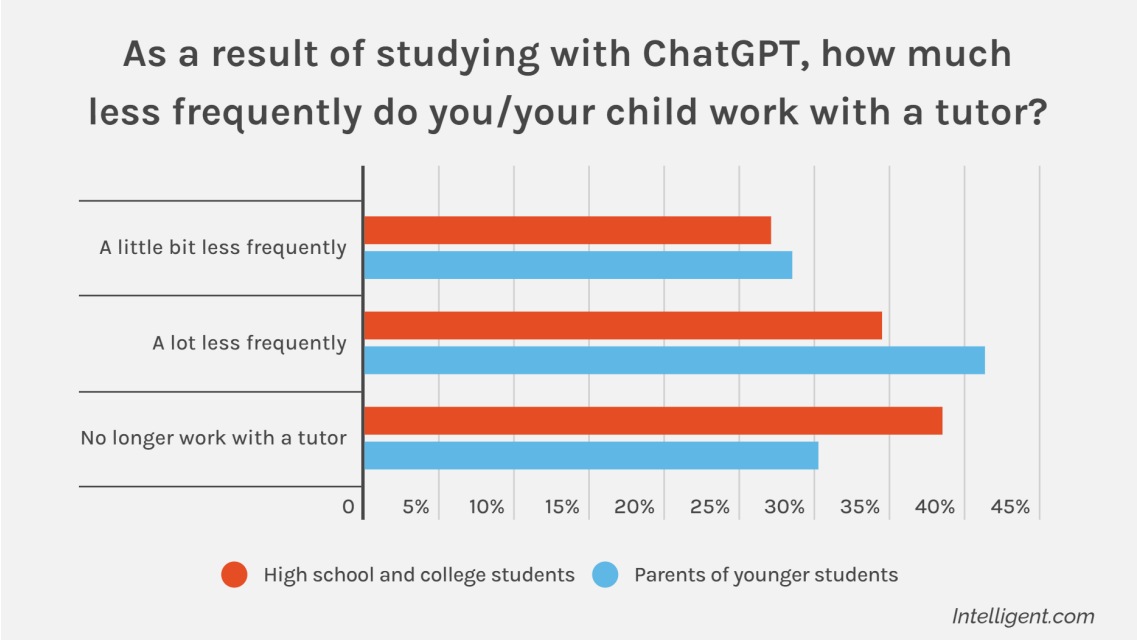


Figura 2.4: Frecuencia con la que se recurre a un tutor humano tras ChatGPT [1].

2.2.2. Evolución de los tutores inteligentes

La integración de técnicas de inteligencia artificial en el ámbito de la educación y el aprendizaje surge en la década de los setenta, con el fin de construir sistemas capaces de hacer el trabajo de un tutor humano de manera automática. Tras conocer que las tutorías personalizadas eran más efectivas que la enseñanza en grupo, los expertos tuvieron la idea de comenzar a desarrollar estas herramientas para que los estudiantes pudieran aprender de un experto con mayor facilidad y rapidez.

A partir de entonces, se desarrollaron una gran cantidad de tutores inteligentes para tareas específicas, aunque a lo largo de la historia siempre ha surgido la pregunta de si estos sistemas eran realmente útiles y capaces de reemplazar a los tutores humanos de forma efectiva. Muchos de ellos utilizan la inteligencia artificial para abordar campos como las matemáticas, el aprendizaje de lenguajes de programación, la resolución de problemas de ciencias, o la corrección y mejora de textos escritos en otros idiomas [22].

Las diferentes técnicas y posibilidades aumentaron con el paso de los años (Tabla 2.3), pero los Sistemas de Tutoría Inteligentes han seguido habitualmente una arquitectura con los mismos componentes (Figura 2.5): Modelo de Conocimiento (qué se enseña), Modelo de Estudiante (a quién se enseña), Modelo Pedagógico (cómo se enseña) y Modelo de Interfaz de Usuario (cómo interactúa el estudiante con el sistema).

1970s: Generación de Problemas	1980s: Rastreo de Modelos	1990s: Control del Alumno
■ Modelado del estudiante ■ Representación del conocimiento ■ Habilidades y conocimiento estratégico ■ Entornos de aprendizaje reactivo ■ Sistemas y tutores expertos	■ Razonamiento basado en casos ■ Progreso de los modelos mentales ■ Simulaciones ■ Procesamiento del lenguaje natural ■ Sistemas de autor	■ Aprendizaje individual vs. colaborativo ■ Aprendizaje situado vs. procesamiento de información ■ Realidad virtual

Tabla 2.3: Eventos relevantes en la evolución de los ITSs [41].

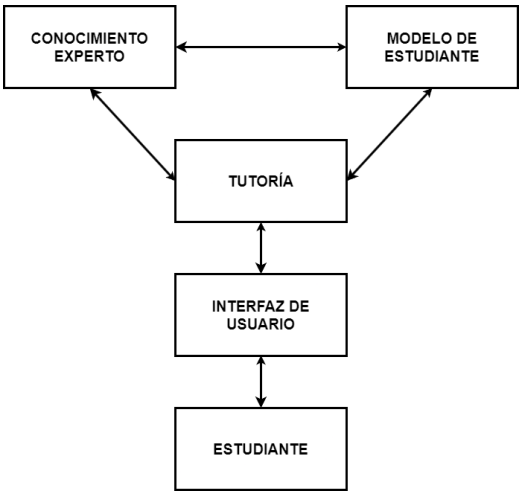


Figura 2.5: Arquitectura tradicional de un Sistema de Tutoría Inteligente [23].

Modelo de Conocimiento

También conocido como Modelo de Dominio, este contiene todo el material que debe enseñarse. Representa tanto el conocimiento del dominio como la forma en la que el experto trabaja con la información. Esto incluye hechos, conceptos o estrategias para resolver los problemas que el tutor debe conocer para proporcionar soluciones, pero no debe confundirse con la forma de enseñar, la cual corresponde al Modelo Pedagógico. Este conocimiento debe poder representarse y usarse a través de un lenguaje, para lo que se puede recurrir a diferentes estrategias como reglas, lógica difusa, modelo cognitivo, modelo basado en restricciones o modelo experto, entre otras.

Modelo de Estudiante

Es fundamental que el sistema sea capaz de entender correctamente al usuario. Este modelo representa los conocimientos y habilidades del estudiante de forma dinámica a través de la interacción, por lo que su objetivo es conocer a quién se está enseñando para ajustar la retroalimentación y proporcionar la información de manera efectiva. Una técnica que se utiliza habitualmente es evaluar el conocimiento del estudiante comparando su comportamiento con el de un experto. Además del conocimiento correcto, también se puede modelar el estudiante en función de sus fallos y malentendidos respecto al dominio. Otro enfoque es utilizar técnicas de aprendizaje automático para generar una serie de reglas para la resolución de problemas, las cuales explican los pasos hasta que se produce un malentendido en el proceso de adquirir el conocimiento.

Modelo Pedagógico o de Tutoría

Este modelo utiliza el Modelo de Conocimiento y el Modelo de Estudiante para tomar decisiones a la hora de enseñar, actuando como el motor del sistema. Para ello también tiene en cuenta la estrategia de enseñanza adoptada por el sistema, que se selecciona en función del proceso de aprendizaje. Adaptarse correctamente al estudiante permite adaptarse a su ritmo y mantenerlo motivado. Además de guiar al usuario, la interacción debe ser natural cuando se responde a sus preguntas, se presenta material o se evalúa su rendimiento. El tutor debe ser capaz de detectar los errores y proporcionar las explicaciones y la ayuda necesaria, así como proponer temas que necesiten ser más trabajados según el Modelo de Estudiante.

Modelo de Interfaz de Usuario

Es el responsable de la interacción entre el usuario y el sistema, para la cual se pueden utilizar distintos tipos de comunicación. Algunos de estos enfoques son la comunicación gráfica (agentes inteligentes animados o realidad virtual), la inteligencia social (análisis verbal o sistemas visuales), interfaces de componentes (procesan fórmulas o símbolos) y la comunicación en lenguaje natural (a través de diálogos).

2.2.3. Tutores inteligentes en la actualidad: LLMs

Los continuos avances de la inteligencia artificial durante los últimos años han permitido que a día de hoy los tutores virtuales ya sean una herramienta realmente eficiente en el ámbito educativo y del aprendizaje. Estos son capaces de cubrir las necesidades de cada estudiante y adaptarse a su ritmo de aprendizaje, de la misma manera que lo haría un tutor humano. Los pilares fundamentales que han dado lugar a la situación actual son los algoritmos y los modelos de aprendizaje automático, en especial los grandes modelos de lenguaje que utilizan técnicas de aprendizaje profundo. Sin embargo, este progreso plantea desafíos como la privacidad de los datos, la desigualdad que genera poder acceder o no a estas herramientas, y el papel de los tutores humanos en un entorno donde la automatización está cada vez más presente.

Actualmente, las herramientas más utilizadas como tutores inteligentes, tales como ChatGPT, se basan en Grandes Modelos de Lenguaje (LLMs). Debido a la importancia y relevancia que estos han tomado en nuestros días, esta sección se enfocará en explicar en qué consisten y qué supone su uso en el ámbito de la educación. Por otro lado, las diferentes técnicas de inteligencia artificial y aprendizaje automático siguen siendo cruciales para desarrollar tutores virtuales basados en un compromiso más proactivo que reactivo, los cuales están diseñados para que el estudiante alcance objetivos específicos sobre el dominio a través de la planificación y la evaluación [37].

Los Grandes Modelos de Lenguaje (LLMs) son herramientas de Procesamiento del Lenguaje Natural (NLP) que tienen un propósito general. Se basan en *deep learning* y aprenden de datos masivos para identificar patrones, reglas del lenguaje y relaciones semánticas, lo que les permite tener grandes capacidades para comprender el lenguaje natural y poder generarlo de forma eficiente [39]. Por lo tanto, son capaces de generar textos coherentes de gran calidad como respuesta a la correcta interpretación de las consultas que reciben. Esto incluye dominar el vocabulario, la sintaxis y el contexto.

Si bien es cierto que no tienen un propósito específico, la cantidad de aplicaciones de los LLMs es muy amplia. Sus recientes avances y la disponibilidad de los mismos como servicio han permitido que puedan utilizarse como un tutor que interactúa libremente con el alumno a través de las instrucciones que este le proporciona [57]. De esta manera, se introduce la utilización de los LLMs en el ámbito de la educación. Estos son capaces de proporcionar a los alumnos enseñanza personalizada eficiente tras analizar su comportamiento e intentar adaptarse a su situación actual y ritmo de aprendizaje, además de asesorarlos continuamente y poder crear para ellos nuevo material de estudio y recursos con fines específicos [39, 25].

Además del procesamiento del lenguaje natural y el aprendizaje profundo, los LLMs también hacen uso de otras tecnologías como el aprendizaje por refuerzo para optimizar la efectividad de las respuestas, o la minería de datos para descubrir nuevos patrones y lagunas, entre otras [60]. Estos modelos habitualmente emplean dos etapas: Preentreno y *fine-tuning*. Durante la primera se aprende de los datos para encontrar patrones y se genera un conocimiento base, mientras que la segunda consiste en ajustar el modelo para que se adapte a una tarea específica o actúe de un modo concreto [43].

2.3. GPT-series

Las grandes empresas del sector han desarrollado una gran cantidad de LLMs durante los últimos años. Como este trabajo consiste en la creación de un tutor virtual basado en ChatGPT, esta sección se centrará en conocer más profundamente la propia herramienta y aquellos modelos de lenguaje en los que se basa.

2.3.1. OpenAI

OpenAI es una organización de investigación y despliegue de Inteligencia Artificial cuyo objetivo es que una IA segura sea capaz de beneficiar a toda la humanidad, incluso a través de sistemas más inteligentes que los seres humanos [2]. Dicen querer cambiar los límites de la posibilidad en un mundo en el que cualquiera pueda ser ayudado de forma efectiva, potenciando así sus conocimientos, habilidades y creatividad [24].

Para lograrlo, desarrollan sus propios modelos generativos empleando técnicas de aprendizaje profundo. Entre sus productos más conocidos se encuentra ChatGPT [6], un chatbot que se apoya en los grandes modelos de lenguaje de la empresa (como GPT-3.5 o GPT-4) para proporcionar respuestas inmediatas a cualquier consulta. Para encontrar más información sobre sus investigaciones y herramientas, se recomienda visitar su índice de investigación [16].

2.3.2. Modelos GPT

Los modelos GPT (Generative Pretrained Transformer) son los LLMs desarrollados por OpenAI, y están destinados a resolver tareas de propósito general a través del conocimiento que hay en el mundo. En la figura 2.6 se puede apreciar la evolución de los modelos GPT desde las primeras exploraciones hasta nuestros días.

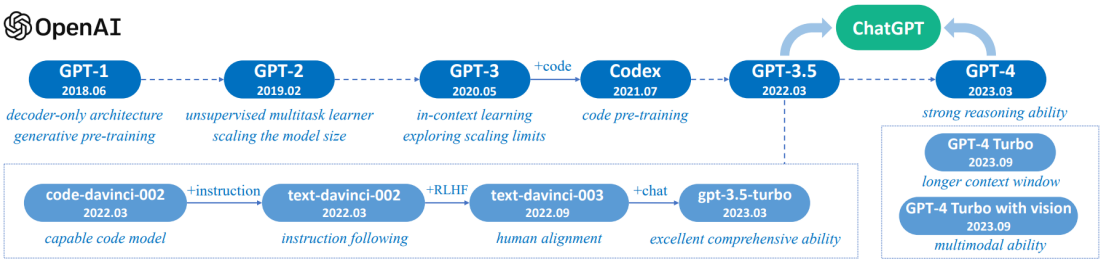


Figura 2.6: Evolución de la serie de modelos GPT desarrollados por OpenAI [60].

GPT-1 y GPT-2

Hasta estas primeras aproximaciones, los modelos de procesamiento del lenguaje natural solían enfocarse en tareas específicas mediante aprendizaje supervisado. Esto, además de limitar el rendimiento del modelo en otras áreas, con frecuencia requería grandes cantidades de datos no disponibles sobre aspectos demasiado concretos.

Por su parte, GPT-1 fue desarrollado combinando aprendizaje no supervisado para preentreno y aprendizaje supervisado para *fine-tuning* [59]. De esta manera, la base del

modelo no se enfocaba en aprender tareas específicas, mientras que luego fue ajustado para ello. Su arquitectura (Figura 2.7) se ha convertido en el núcleo de sus sucesores.

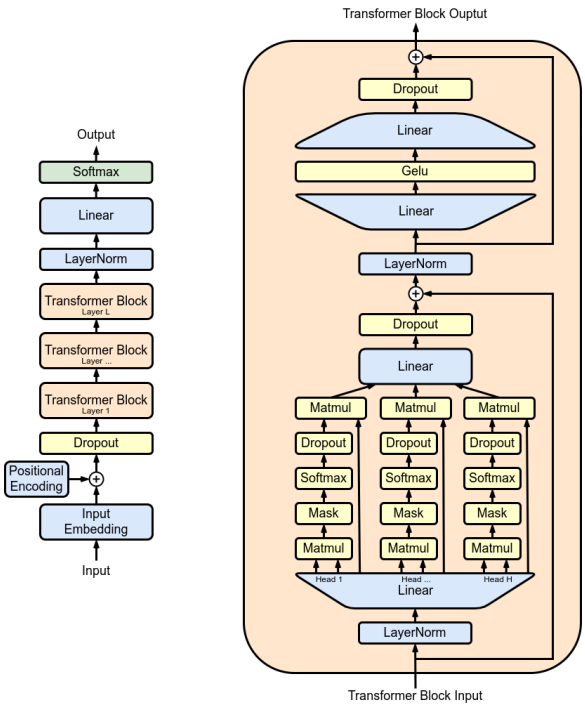


Figura 2.7: Arquitectura base de los modelos GPT [60].

Para GPT-2 se utilizó un mayor conjunto de datos, pero se recurrió únicamente al aprendizaje no supervisado [59]. En lugar de realizar explícitamente *fine-tuning* para la resolución de tareas específicas, se introducen las probabilidades para la resolución multitarea. Esto es $p(output \mid input,\ task)$ en vez de $p(output \mid input)$. De este modo, ya no se proporciona una única salida para la misma entrada y diferentes tareas [60].

GPT-3 y GPT-3.5

Los parámetros de GPT-3 ascienden a los 175B, cien veces más que los de su predecesor GPT-2. Se introduce el Aprendizaje En Contexto (ICL), lo que le permite al modelo comprender las propias tareas como lenguaje natural. De esta manera, el preentreno predice la siguiente secuencia de texto, mientras que el ICL predice la solución a la tarea [59]. Esto también ayudó a reducir el número de ejemplos que era necesario proporcionar para que el modelo desempeñara correctamente la tarea, lo cual supuso un enorme paso en el desarrollo de los grandes modelos de lenguaje.

Pese a las grandes capacidades de GPT-3, aún existían dificultades en la resolución de tareas de mayor complejidad, tales como completar código o resolver problemas matemáticos. Para potenciar las habilidades de razonamiento del modelo y poder enfrentar estas tareas complejas de una mejor manera, comenzaron a desarrollarse los modelos basados en un modelo GPT que aprende de datos con código. De esta forma nacen los modelos GPT-3.5, con mayores capacidades que GPT-3 [60].

GPT-4

Sin duda, GPT-4 ha supuesto uno de los mayores avances en el desarrollo de grandes modelos de lenguaje por parte de OpenAI [60]. La compañía lo define como su sistema más avanzado, el cual produce respuestas más útiles y seguras que modelos anteriores, especialmente en la resolución de tareas complejas [51]. Además, se ha trabajado con expertos durante meses para conseguir que GPT-4 sea más seguro a la hora de responder consultas provocativas y acerca de contenido no permitido, reduciendo la generación de contenido tóxico y las alucinaciones [9].

En base a este trabajo, más adelante se introdujeron nuevas herramientas que permitieron comprender el verdadero potencial de un sistema de aprendizaje multimodal (ver, escuchar, y hablar) [60]. Entre ellas se encuentra GPT-4V, con la posibilidad de proporcionar al modelo una entrada visual que este es capaz de interpretar de forma eficiente. También se lanzaron varias versiones mejoradas de GPT-4 conocidas como los modelos GPT-4 Turbo, algunos de los cuales actualizan las fuentes de conocimiento hasta casi nuestros días y cubren funcionalidades como la visión, la generación de imágenes con DALL-E 3, el texto a voz (TTS) y la capacidad para escuchar audio [51]. La última novedad de OpenAI es GPT-4o, un modelo introducido en mayo de 2024 capaz de razonar con audio, visión y texto en tiempo real [53]. Todo ello ha permitido una increíble mejora en las capacidades y posibilidades de los LLMs, aunque aún existen limitaciones y aspectos por mejorar de cara al futuro.

Para evaluar el rendimiento de GPT-4, OpenAI ha realizado una gran cantidad de experimentos. Uno de ellos ha consistido en simular exámenes diseñados para ser realizados por seres humanos, sin entrenar el modelo específicamente para ellos. Estos exámenes estaban formados tanto por preguntas multiopción como de respuesta libre, y la evaluación se realizó de acuerdo con la metodología utilizada en la corrección de cada una de las pruebas. En la figura 2.8 se puede apreciar el rendimiento de GPT-3.5, GPT-4 sin visión y GPT-4 en los diferentes exámenes académicos seleccionados.

Actualmente, GPT-4 se encuentra disponible en ChatGPT Plus y en la API para desarrolladores de OpenAI [3], que permite crear aplicaciones y servicios orientados a objetivos generales o específicos a partir de sus diferentes modelos [9].

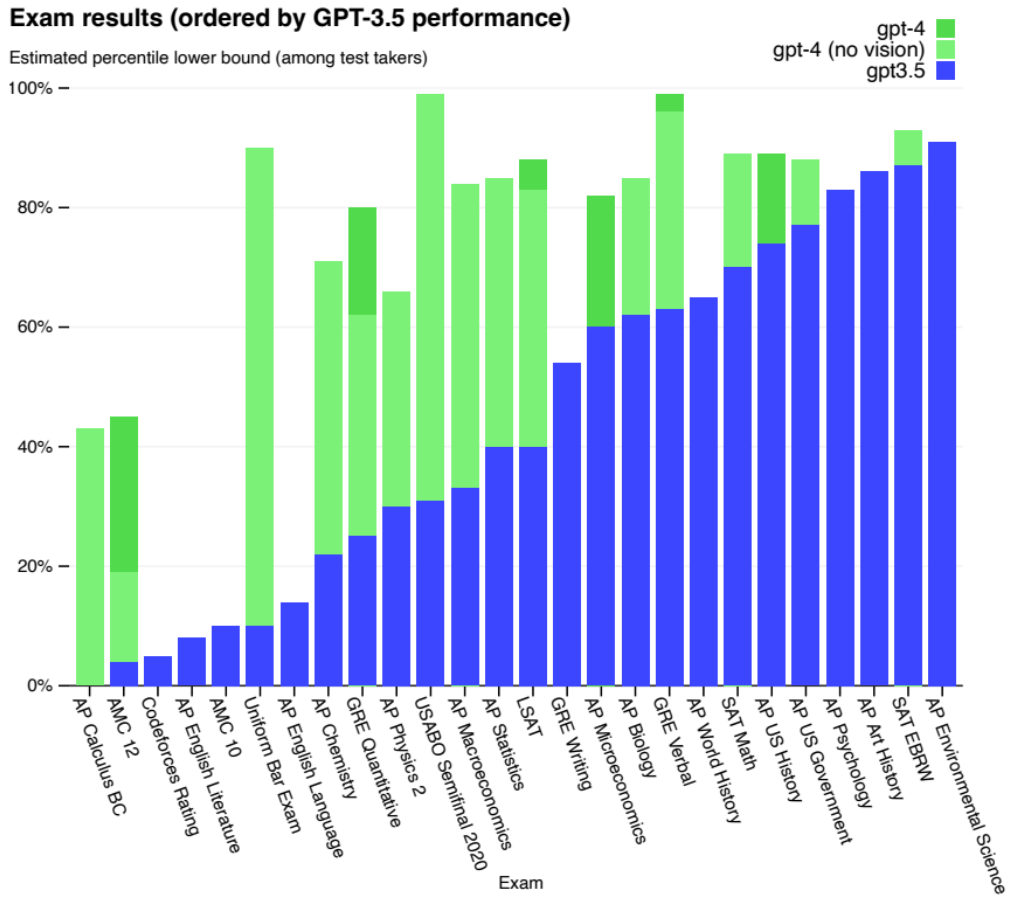


Figura 2.8: Rendimiento de GPT-4 respecto a GPT-3.5 en exámenes académicos [52].

2.3.3. ChatGPT

ChatGPT es un sistema de inteligencia artificial que se basa en la arquitectura de los modelos GPT y es capaz de generar texto para resolver tareas de forma eficiente y responder a cualquier cuestión que se le pregunte, generando conversaciones y optimizando el diálogo con los seres humanos. Desde que fue lanzado por OpenAI en noviembre de 2022, ChatGPT puede considerarse su producto estrella, un gran avance para la inteligencia artificial y posiblemente el chatbot más potente de la historia [44].

Actualmente, la versión gratuita de ChatGPT se basa en GPT-3.5. Esto significa que ambos comparten la misma arquitectura, pues el modelo que utiliza ChatGPT es el resultante de aplicar *fine-tuning* a GPT-3.5 para mejorar su capacidad de mantener conversaciones naturales con los humanos [6]. Lo mismo ocurre con sus versiones de pago, entre ellas ChatGPT Plus, que permite acceder a GPT-4 y utilizarlo como base.

La interfaz de usuario de ChatGPT les permite a los usuarios interactuar con el sistema a través de varios chats al mismo tiempo, cuyo contenido permanece accesible incluso después de cerrar la sesión. Cuando el usuario interactúa con ChatGPT a través de un chat, el modelo es capaz de recordar los mensajes anteriores y continuar la conversación en base al transcurso de la misma [60].

Su versión de pago, ChatGPT Plus, proporciona acceso a GPT-4 [47]. Esto no sólo significa poder interactuar con un sistema que se basa en modelos más avanzados, sino también disponer de todas las nuevas posibilidades que estos ofrecen. Entre ellas está proporcionar imágenes o documentos como entrada [45], la generación de imágenes a través de DALL-E 3 [46], y utilizar la voz para conversar. Además, los usuarios que dispongan de este plan pueden utilizar cientos de GPTs que han sido personalizados para enfocarse en tareas específicas, e incluso crear sus propios GPTs de forma sencilla guiando el comportamiento del modelo a través de prompts [48].

2.4. Creación de tutores virtuales basados en ChatGPT

Existe una gran variedad de opciones para desarrollar un tutor virtual basado en las tecnologías que utiliza ChatGPT. El potencial y las capacidades de los grandes modelos de lenguaje han dado lugar a que continuamente nazcan nuevas herramientas y posibilidades para crear chatbots que cubran necesidades específicas.

A pesar de ello, existen dos pilares fundamentales para su desarrollo. El primero de ellos es utilizar un LLM de propósito general y modificar su comportamiento para que se adapte a las necesidades que debe cubrir, convirtiéndose así en un modelo orientado a una tarea específica. Esto puede lograrse utilizando prompts muy detallados para guiar al modelo, aumentando la base de conocimiento con nueva información relevante, o realizando *fine-tuning* para que este actúe de un modo concreto adaptándose a nuestros propios datos de entrenamiento. Por otra parte, los usuarios deben poder interactuar con el sistema a través de un chatbot que proporcione respuestas utilizando el modelo de lenguaje anterior. Se puede utilizar una gran cantidad de LLMs, aunque si se desea que el modelo se base en ChatGPT, lo ideal es utilizar los GPTs en los que se apoya.

Crear un modelo que se adapte a los contenidos de una asignatura específica y actúe como un tutor virtual de la misma es solamente un caso particular entre todas las posibilidades que ofrece la personalización de GPTs, como lo son los asistentes que utilizan las empresas para reclutar clientes o brindarles ayuda, atención y servicios. A lo largo de esta sección se van a presentar diferentes herramientas y posibilidades para crear un tutor virtual basado en ChatGPT.

2.4.1. GPTs personalizados con ChatGPT

Los GPTs son una herramienta que fue introducida por OpenAI en noviembre de 2023 [49], y actualmente permite a los usuarios de ChatGPT Plus y ChatGPT Enterprise utilizar y crear versiones de ChatGPT personalizadas [48]. Guiar al modelo a través de instrucciones precisas, proporcionarle múltiples fuentes de conocimiento adicionales y sacar partido de sus diferentes capacidades, son factores que facilitan la creación de un chatbot con un propósito específico. Esta herramienta nace de la dificultad que los usuarios encontraban para que ChatGPT les proporcionara respuestas lo suficientemente precisas sobre determinados temas, ya fuera porque no seguía la metodología deseada o porque no disponía de los datos necesarios para ello [50].

El objetivo de los GPTs es conseguir que la vida diaria de los usuarios se vuelva más sencilla, dándoles la opción de personalizar sus propios modelos para recibir ayuda y resolver de manera más eficiente cualquier tarea concreta. Hasta entonces, para conseguirlo era necesario utilizar prompts específicos y lo suficientemente elaborados como para que el modelo fuera capaz de proporcionar las respuestas esperadas, pero este problema de la comunidad se soluciona con la llegada de GPTs que son guiados para enfocarse directamente en tareas concretas cuando sean usados [48]. Estos pueden reservarse únicamente para uso personal, o bien ser compartidos para que otros clientes del plan de pago los utilicen, al igual que se puede acceder a los GPTs que otros usuarios comparten [54]. Una de las desventajas de esta opción, entre sus muchos beneficios, es que el modelo sólo puede utilizarse dentro de ChatGPT. En la figura 2.9 se pueden apreciar los GPTs más populares actualmente.

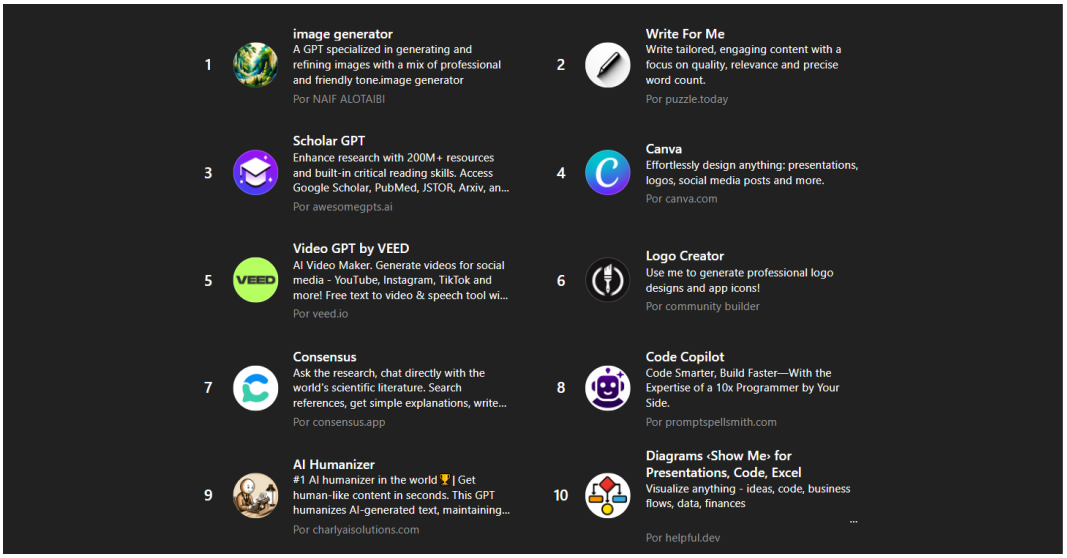


Figura 2.9: GPTs más populares [7].

Para crear un GPT utilizando ChatGPT, no se necesita código. Lo único que hay que hacer es guiar a nuestro modelo proporcionándole las instrucciones que debe seguir a través de prompts, así como los documentos o las fuentes de información en las que debe apoyarse para desarrollar dicho comportamiento y adaptarse al caso de uso que queremos cubrir [26, 50]. Las posibilidades son infinitas, y la creación de un tutor virtual para una asignatura con esta herramienta es sólo un caso más de todo lo que se puede construir a través de ella. Para ello, tan solo sería necesario proporcionar las instrucciones que debe seguir nuestro tutor de la forma más precisa posible para responder preguntas de teoría o resolver ejercicios prácticos, así como todo el material de la asignatura, exámenes o ejercicios resueltos, y aquella información adicional que consideremos relevante y que pueda mejorar su funcionamiento.

2.4.2. OpenAI API

OpenAI tiene su propia API, a través de la que ofrece sus diferentes modelos. De esta forma, los desarrolladores pueden acceder a estas herramientas para construir sus propias aplicaciones o servicios, pagando en función del uso que se haga de las mismas y sin necesidad de utilizarlas únicamente dentro de ChatGPT [3]. Entre las capacidades de la API se encuentran la generación de texto, la llamada a funciones, el ajuste fino o *fine-tuning*, la generación de imágenes o la creación de asistentes, entre otras. Para conocer todas ellas y saber cómo utilizarlas, se puede revisar la documentación que proporciona OpenAI [14] y la referencia de la API [4].

A lo largo de esta subsección se van a explicar los servicios de la API que se pueden utilizar para crear un tutor virtual basado en ChatGPT, lo que significa construir una herramienta que se enfoque en un caso de uso específico a partir de un modelo GPT.

Chat Completions API

OpenAI ofrece la posibilidad de utilizar sus modelos para generar texto, aunque también es posible utilizar modelos ya ajustados previamente, como se explicará más adelante. Esta primera herramienta permite realizar llamadas a los modelos a través de una lista de mensajes, obteniendo un nuevo mensaje del asistente como salida. Cada uno de estos objetos toma un rol (“system”, “user” o “assistant”), y aunque se pueden realizar consultas individuales, el formato de una conversación suele estar formado por un objeto con el rol del sistema y posteriormente se alternan mensajes entre el usuario y el asistente para que se tengan en cuenta los mensajes anteriores a la hora de generar una respuesta para la nueva consulta del usuario.

El mensaje que adopta el rol “system” se utiliza para modificar el comportamiento del modelo, que es guiado a través de instrucciones específicas para que se adapte mejor a un caso de uso concreto. Esto puede ser insuficiente en muchos casos, pues es posible que se necesite una gran cantidad de datos y ejemplos para que el modelo funcione de la forma deseada. Sin embargo, OpenAI asegura que utilizar prompts elaborados y muy específicos supone una gran mejora en el rendimiento del modelo.

En el listado de código 2.1, se puede apreciar un ejemplo de lo que podría ser una llamada típica a la API. Observamos que la siguiente respuesta va a ser generada con GPT-3.5 Turbo, y que para ello se tendrán en cuenta los mensajes proporcionados. Estos han podido ser escritos manualmente, pero lo habitual es que representen el estado de una conversación. En primer lugar hay un mensaje que toma el rol del sistema, y contiene las instrucciones con las que debe ser generada la respuesta. Además, le siguen una consulta del usuario y la respuesta a la misma por parte del asistente. Llegados a este punto, el usuario realiza una nueva consulta, cuya respuesta será devuelta en la llamada a la API y posteriormente añadida a la conversación para seguir procesando nuevas consultas. La respuesta devuelta es un objeto compuesto por diferentes campos, cuyo valor en forma de texto se puede extraer como se indica en el listado de código 2.2. Cada una de estas llamadas a modelos genera una respuesta por parte del asistente, pero para crear un tutor virtual será necesario incluir toda la información necesaria en un mensaje que adopte el rol "system" y posteriormente integrar la API en un chatbot que permita al usuario tener una conversación con el modelo a través de llamadas continuas que contengan todos los mensajes anteriores. Toda la información sobre Chat Completions API se puede encontrar en la documentación correspondiente [17], y los costes de los modelos en la sección de precios de la API [13].

```
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant", "content": "The Los Angeles Dodgers won the World
        Series in 2020."},
    {"role": "user", "content": "Where was it played?"}
  ]
)
```

Código 2.1: Ejemplo de llamada a Chat Completions API [17].

```
response.choices[0].message.content # Acceso al mensaje de respuesta
OUTPUT: "The 2020 World Series was played in Texas at Globe Life Field..."
```

Código 2.2: Extracción de la respuesta con Chat Completions API [17].

Fine-tuning

El *fine-tuning* o ajuste fino es una de las herramientas que puede utilizarse para personalizar un modelo y conseguir que se adapte a un caso de uso específico. Este nos permite entrenar con más ejemplos de los que podría cubrir un prompt y tiende a ofrecer resultados de mayor calidad frente a otras opciones, ya que se enfoca en modificar el comportamiento del modelo, y no sólo en guiarlo a través de instrucciones. Esto puede ser útil en caso de que queramos resolver una tarea cuyas instrucciones son difíciles de plasmar en un prompt, cuando es necesario modelar varios casos de uso, o cuando queremos cambiar aspectos como el estilo, el formato o el tono [42].

El modelo más recomendable en este caso es GPT-3.5 Turbo, ya que los modelos anteriores no tienen una base tan sólida y el ajuste de GPT-4 todavía está en fase de experimentación. Sin embargo, existen diferentes opciones y elegir un modelo u otro también dependerá en gran parte de lo que se quiera invertir, considerando tanto el coste del entrenamiento como el de utilizar posteriormente el modelo ajustado. Para realizar el *fine-tuning* es necesario preparar un conjunto de entrenamiento que contenga conversaciones similares a aquellas que el modelo podría tener con el usuario. Se recomienda proporcionar los suficientes ejemplos, así como un conjunto de validación, y reservar otro de prueba para evaluar el modelo y sacar conclusiones sobre futuras mejoras. Como instrucciones en los ejemplos de entrenamiento, se recomienda utilizar prompts que ya resultaron efectivos antes de recurrir al *fine-tuning*, e incluso es posible volver a reentrenar un modelo ya ajustado si fuera necesario.

El conjunto de datos debe ser un archivo JSONL, y en el caso de GPT-3.5 Turbo los casos de entrenamiento deben seguir el formato de los mensajes de Chat Completions API, el cual aparece en el listado de código 2.3. En caso de querer utilizar modelos anteriores como babbage-002 y davinci-002, el formato es el del listado de código 2.4.

```
{"messages": [
   {"role": "system", "content": "<instructions>"},
   {"role": "user", "content": "<prompt 1>"},
   {"role": "assistant", "content": "<ideal generated text>"}
   ]}
{"messages": [
   {"role": "system", "content": "<instructions>"},
   {"role": "user", "content": "<prompt 2>"},
   {"role": "assistant", "content": "<ideal generated text>"}
   ]}
```

Código 2.3: Formato de los ejemplos de entrenamiento para *fine-tuning* con GPT-3.5 Turbo [8].

```
{"prompt": "<prompt 1>", "completion": "<ideal generated text>"}
{"prompt": "<prompt 2>", "completion": "<ideal generated text>"}
```

Código 2.4: Formato de los ejemplos de entrenamiento para *fine-tuning* con modelos anteriores a GPT-3.5 Turbo [8].

Para la creación de un tutor virtual siguiendo esta estrategia, el proceso es el mismo que para cualquier otro caso de uso específico. El modelo debe reentrenarse a partir de un conjunto de datos que contenga toda la información relevante, como respuestas a preguntas frecuentes o a aquellas que el modelo base no es capaz de responder correctamente. Además, utilizar como instrucciones los prompts que hayan ofrecido buenos resultados de forma previa al *fine-tuning* posiblemente mejorará el rendimiento del nuevo modelo. Una vez que se haya validado el modelo y cumpla las expectativas, este puede comenzar a utilizarse como cualquier otro a través de llamadas a la API. Es importante recalcar que el modelo que se construye se trata únicamente de un nuevo GPT, y para que el usuario sea capaz de interactuar con él, será necesario integrarlo en un chatbot que proporcione las respuestas a las consultas realizadas. Este proceso consiste en utilizar Chat Completions API, pero ahora se realizan las llamadas con el modelo reentrenado en lugar de con los que proporciona directamente OpenAI.

A pesar de sus capacidades, OpenAI recomienda intentar obtener buenos resultados haciendo uso de prompts, ya que estos con facilidad mejoran el comportamiento de los modelos base y permiten ahorrarse el gran trabajo que requiere la preparación de los datos para ajustar un modelo, así como los costes adicionales del entrenamiento. Toda la información sobre *fine-tuning* utilizando la API de OpenAI se puede encontrar en la documentación correspondiente [8], y los costes en la sección de precios de la API [13].

Assistants API

La API de Asistentes de OpenAI permite construir asistentes que se enfocan en aplicaciones específicas a partir de un modelo GPT base. Para ello, realizan llamadas a los modelos con instrucciones específicas y tienen acceso a diferentes herramientas que mejoran su comportamiento. La motivación de esta herramienta es facilitar a los desarrolladores la posibilidad de crear asistentes orientados a negocios específicos. Como ya se ha comentado anteriormente, la creación de un tutor virtual para una asignatura es un caso más entre todas las posibilidades que nos ofrecen estos servicios.

Una de las herramientas que ofrece Assistants API es el intérprete de código, el cual le permite a la API escribir y ejecutar internamente código en Python para resolver problemas matemáticos o de programación, proporcionando como salida archivos o gráficos. Por otro lado, se puede utilizar la herramienta de búsqueda en archivos para aumentar la base de conocimiento del modelo a través de archivos con información adicional sobre el caso de uso específico. Por último, también está disponible la llamada a funciones, que es especialmente útil cuando el asistente debe hace uso de otras APIs para proporcionar la información deseada.

En la figura 2.10 se puede apreciar un diagrama que representa el funcionamiento de Assistants API para la creación de un asistente que proporciona información en tiempo real sobre el precio de trenes. Para ello se proporcionan las instrucciones correspondientes y se utilizan las herramientas de la API. Más concretamente, se usa la búsqueda en archivos (antes llamada *retrieval*) para obtener información sobre las estaciones, las funciones para llamar a otras APIs encargadas de proporcionar la fecha actual y los horarios de los trenes, y se apoya en el intérprete de código para calcular la fecha deseada en función de la consulta del usuario (por ejemplo, si el usuario pregunta por trenes a la mañana siguiente, será necesario obtener la fecha actual y calcular la fecha de la mañana siguiente para saber qué horarios hay que consultar).

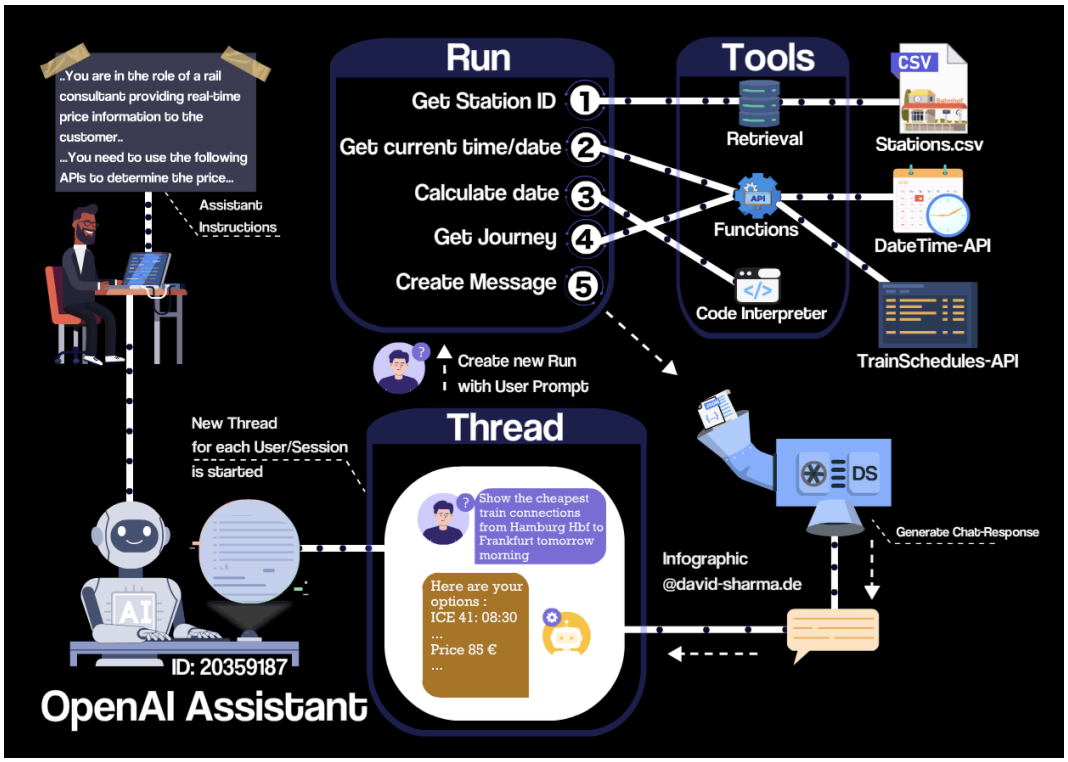


Figura 2.10: Asistente de trenes en tiempo real utilizando Assistants API [58].

Para la creación de un tutor virtual siguiendo esta estrategia, el proceso es similar al necesario para cualquier otro caso de uso específico. En primer lugar debe crearse un asistente con las instrucciones correspondientes, para las que se recomienda una combinación de aquellos prompts que hayan ofrecido buenos resultados para guiar al modelo sin necesidad de proporcionar material adicional. Posteriormente deben utilizarse las herramientas que se consideren, aunque para que un tutor virtual sea realmente eficiente es recomendable que el asistente busque en archivos con todo el material de la asignatura, y en aquellos documentos con respuestas a preguntas frecuentes o con información adicional que puedan mejorar su rendimiento. Si el tutor enseña sobre asignaturas que involucren matemáticas o programación, habilitar el intérprete de código junto a nuevos archivos con algoritmos puede ser una opción interesante de cara a la resolución de ejercicios. La utilización de funciones dependerá de cada proyecto.

Esta es la forma de crear un asistente orientado a un objetivo específico con la API de Asistentes de OpenAI, pero es importante recalcar que para que el usuario sea capaz de interactuar con él, será necesario integrar la API en un chatbot. Para ello debe crearse un hilo por conversación, el cual irá siendo ejecutado conforme se van añadiendo a él los diferentes mensajes entre el usuario y el asistente (Figura 2.11). Toda la información sobre Assistants API se puede encontrar en la documentación correspondiente [5], y los costes en la sección de precios de la API [13].

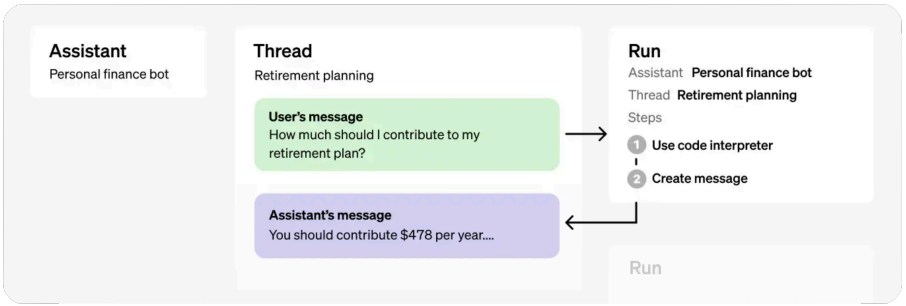


Figura 2.11: Cómo funciona Assistants API [11].

2.4.3. Creación de chatbots e integración de la API de OpenAI

Como ya se ha comentado durante esta sección, los GPTs personalizados con ChatGPT sólo pueden utilizarse dentro de la propia herramienta. Esto supone la desventaja de no poder interactuar con el modelo en aplicaciones externas, aunque hacerlo desde ChatGPT es especialmente sencillo. En cambio, la API de OpenAI sí nos permite utilizar los modelos libremente. Sin embargo, requerimos de un chatbot o sistema que nos permita establecer conversaciones y acceder a las respuestas generadas.

Por lo tanto, en caso de utilizar la API, tanto Chat Completions API como Assistants API, deberemos integrarla para que el usuario pueda interactuar con nuestro modelo o asistente. Para Chat Completions API, tanto con *prompting* como con *fine-tuning*, deben realizarse llamadas continuas al modelo con todos los mensajes de la conversación. De este modo, cada vez que el usuario haga una nueva consulta, se tienen en cuenta las instrucciones y los mensajes anteriores [17]. Por otro lado, un asistente requiere la creación de un hilo para iniciar la conversación, el cual irá ejecutándose con los mensajes de la misma conforme esta vaya transcurriendo [5].

Para realizar la integración correspondiente existen multitud de opciones, entre ellas implementar el chatbot desde cero y realizar las llamadas necesarias a la API. Sin embargo, los continuos avances en este sector y el gran potencial de estas herramientas han permitido el surgimiento de plataformas que permiten crear chatbots totalmente personalizables de manera sencilla. Una de las más populares es Voiceflow, una herramienta líder en construcción de agentes de IA avanzados [18].

Voiceflow, al igual que otras herramientas del sector, facilita la creación de chatbots para aplicaciones específicas. En su caso, permite construir agentes de inteligencia artificial sin necesidad de escribir código y es extensible con otras APIs, entre ellas OpenAI [20]. La creación de un chatbot es completamente personalizable y se realiza a través un flujo de trabajo que contiene bloques con integraciones, variables, interfaces y muchos otros elementos (Figura 2.12). Todo ello da lugar a una lógica conversacional con infinitas posibilidades, que además es especialmente fácil de lanzar en aplicaciones y páginas web una vez que el chatbot ha sido terminado (Figura 2.13) [21, 55].

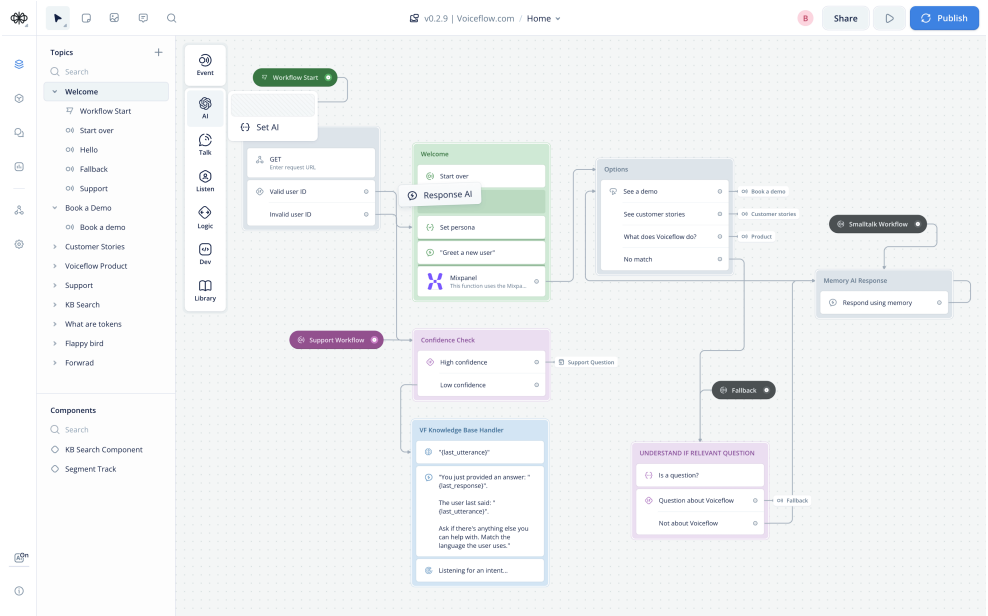


Figura 2.12: Creación de un chatbot con Voiceflow [18].

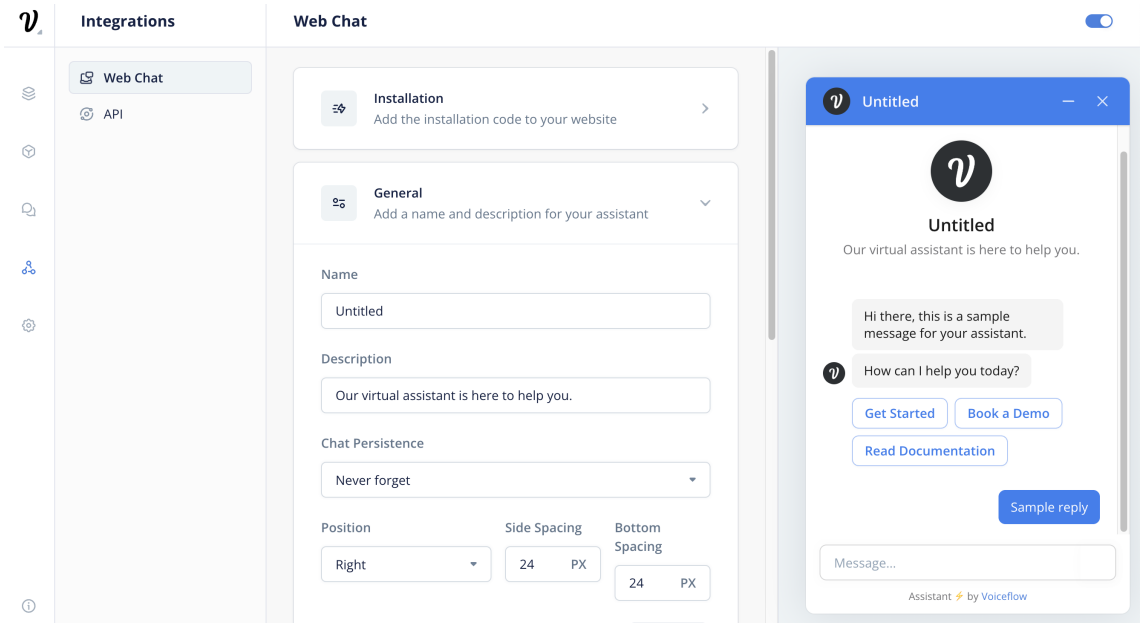


Figura 2.13: Integración de un chat web con Voiceflow [40].

Una de sus herramientas es la base de conocimiento, que permite proporcionar documentos con información adicional sobre casos de uso concretos y nace para hacer frente a Assistants API y a otros servicios destinados a la creación de modelos de lenguaje orientados a objetivos específicos [12]. Voiceflow cuenta con diferentes modelos (entre ellos GPT-3.5 o GPT-4 de OpenAI), cuyo uso está limitado por un número de tokens al mes en función del plan de la cuenta. Sin embargo, también nos permite realizar llamadas a la API de OpenAI para que las respuestas las genere nuestro GPT personalizado previamente, por cuyo uso ya estamos pagando cada vez que utilizamos el modelo. Por lo tanto, esta plataforma permite construir un chatbot de manera sencilla y totalmente personalizable, utilizando tanto sus propios recursos para la creación de agentes que se adaptan a tareas específicas como modelos ya personalizados fuera de Voiceflow (por ejemplo, un asistente creado con Assistants API) [38, 56]. Esto abre nuevas puertas para la integración de la API de OpenAI en aplicaciones o páginas web. Implementar un chatbot realizando las llamadas necesarias a los modelos ya no es la única opción, sino que es posible centralizar el comportamiento del asistente e integrarlo como un chat web o de forma personalizada utilizando una API de diálogo.

3. Desarrollo del proyecto

En este capítulo se va a detallar el desarrollo del proyecto, desde el estudio de las diferentes tecnologías y posibilidades hasta la creación del tutor virtual basado en ChatGPT y su integración en una aplicación web. Las diferentes fases de la metodología utilizada están explicadas con mayor detalle en el capítulo A.

3.1. Estudio de tecnologías y opciones

En primer lugar, se ha realizado un estudio exhaustivo sobre las diferentes herramientas y posibilidades para desarrollar el proyecto. La creación de un tutor virtual basado en ChatGPT implica conocer y aprender sobre tecnologías que acaban de surgir y que evolucionan constantemente, por lo que este proceso es fundamental para establecer una base sólida y poder tomar mejores decisiones a la hora de seleccionar las herramientas que se utilizarán para llevar a cabo la implementación.

Por lo tanto, el objetivo principal de esta investigación ha sido explorar y evaluar las diferentes tecnologías que satisfacen los requisitos del proyecto. Además, se han identificado puntos a favor y en contra de cada una de las opciones para su posterior selección o descarte. A lo largo de la sección se van a presentar los métodos empleados para la búsqueda de información, así como las diferentes alternativas y las primeras conclusiones sobre el uso de cada una de ellas.

El estudio se ha llevado a cabo a través de la búsqueda de información en artículos o páginas web, aunque también se ha recurrido a otras opciones como vídeos, cursos o repositorios, cuyo uso ha permitido hacer un mejor análisis comparativo entre las opciones y comprender sus capacidades y limitaciones de una mejor manera. Uno de los puntos clave para el desarrollo del tutor virtual basado en ChatGPT ha sido comprender cómo funciona la herramienta, en qué modelos se basa y cómo estos se diferencian entre sí (Sección 2.3). Una vez adquirido el conocimiento sobre modelos de lenguaje, y más concretamente sobre los GPTs, ha sido imprescindible identificar diferentes formas de

utilizar estos modelos para crear una versión de los mismos que se adapte a un caso de uso específico (Sección 2.4).

Llegados a este punto, es importante no confundir el modelo a desarrollar con un chatbot o sistema con el que se pueda interactuar. Por un lado, se debe crear el tutor virtual, que se basa en un GPT y es capaz de adaptarse a los contenidos de la asignatura de *Sistemas Inteligentes*. Por otra parte, el usuario debe poder interactuar con el modelo a través de una aplicación que contenga un chatbot, de forma que las consultas que realice sean respondidas utilizando el conocimiento del tutor virtual.

Tras realizar el estudio, se plantean tres alternativas principales:

1. **Crear un GPT personalizado con ChatGPT.** Una de las principales ventajas de esta opción es la facilidad para guiar al modelo hacia el caso de uso específico, proporcionando prompts y documentos repetidamente hasta que se comporte de la forma deseada. Además, se pueden aprovechar capacidades de GPT-4 como la visión o la herramienta de búsqueda, y se puede interactuar con el modelo directamente dentro de ChatGPT. Sin embargo, el tutor no se podría utilizar de forma externa, y su uso estaría limitado a usuarios con el plan de pago de ChatGPT. Se puede encontrar más información en la subsección 2.4.1.
2. **Utilizar la API de OpenAI.** La organización ofrece acceso a sus modelos a través de diferentes servicios. Más concretamente, las herramientas más interesantes para este caso son Chat Completions API (aplicando o no *fine-tuning*) y Assistants API. La principal ventaja de cualquiera de estas opciones es el pago por uso y la posibilidad de integrar la API en aplicaciones propias, así como la libertad en la selección y uso de una gran variedad de modelos base. Por contra, esto requiere aprender a integrarlas, y desarrollar un sistema que le permita al usuario poder interactuar con el modelo a través de llamadas a la API. Se puede encontrar más información en la subsección 2.4.2.
3. **Utilizar una plataforma de creación de agentes de IA que permita usar un GPT como modelo base.** Existen herramientas, como Voiceflow, que permiten crear chatbots orientados a casos de uso específicos de forma similar a Assistants API. Voiceflow utiliza un modelo base (por ejemplo, GPT-3.5) y permite la personalización a través de una base de conocimiento con información adicional sobre el caso de uso. La principal ventaja es que el único coste es el plan de la cuenta en la plataforma, que actualmente permite crear dos agentes con el plan gratuito. Además, la lógica conversacional es completamente personalizable y ofrece infinitas posibilidades. La parte correspondiente a la interacción con el modelo es sencilla de implementar, pues el chatbot se puede integrar fácilmente en una aplicación web. Sin embargo, la cantidad de tokens ofrecidos es limitada, y para aumentarlos es necesario seleccionar un plan de pago. Otra alternativa que resulta interesante dentro de esta opción es incluir las llamadas a la API de OpenAI en la lógica conversacional del chatbot, de forma que este se conecta con un GPT por cuyo uso ya estamos pagando (Subsección 2.4.3).

3.2. Estimación de costes

Tras la realización de un primer estudio sobre las diferentes opciones para llevar a cabo el proyecto, se ha hecho una estimación de los costes de cada una de ellas. Es importante recalcar que los cálculos se han realizado teniendo en cuenta únicamente los servicios que requiere el desarrollo del proyecto, es decir, considerando la creación del tutor virtual y la optimización de su comportamiento, pero no su posterior despliegue para ser utilizado por alumnos y profesores.

Para estimar los costes de cada una de las opciones, se han enumerado los diferentes servicios por los que hay que pagar (plan, modelo base utilizado, tokens de entrada y salida, etc) y se han propuesto diferentes combinaciones realistas de los mismos.

Crear un GPT personalizado con ChatGPT

Esta primera opción requiere lo siguiente:

- Un plan de ChatGPT que proporcione acceso a la creación de GPTs durante la realización del proyecto.

Para la estimación de costes (Tabla 3.1) se ha tenido en cuenta el plan utilizado y los meses necesarios para el desarrollo del GPT.

Plan utilizado	Meses	Coste estimado
ChatGPT Plus	1	$20.00
	2	$40.00
	3	$60.00

Tabla 3.1: Costes de crear un GPT personalizado con ChatGPT [6].

Crear un agente de IA (chatbot) basado en un GPT con Voiceflow

Esta opción requiere lo siguiente:

- Un plan de Voiceflow que proporcione acceso a la creación de agentes de IA (chatbots) basados en un GPT durante la realización del proyecto.

Para la estimación de costes (Tabla 3.2) se ha tenido en cuenta el plan utilizado (con su número mensual de tokens) y los meses necesarios para el desarrollo del proyecto.

Plan utilizado	Meses	Coste estimado
Sandbox (Hasta 100K tokens al mes)	1	$0.00
	2	$0.00
	3	$0.00
Pro (Hasta 2M tokens al mes)	1	$50.00
	2	$100.00
	3	$150.00

Tabla 3.2: Costes de crear un agente de IA (chatbot) basado en un GPT con Voiceflow [18].

Utilizar Chat Completions API con un modelo base sin ajustar

Esta opción requiere lo siguiente:

- Utilizar un GPT como modelo base durante la realización del proyecto, lo que implica considerar los tokens de entrada y salida empleados durante su uso.

Para la estimación de costes (Tabla 3.3) se ha tenido en cuenta el modelo y los tokens de entrada y salida necesarios para el desarrollo del proyecto. Se ha decidido realizar la estimación con un millón de tokens de entrada y tres millones de tokens de salida disponibles, los cuales deberían ser suficientes para el desarrollo del proyecto.

Tokens entrada	Tokens salida	Modelo	Coste estimado
1M	3M	gpt-4-0125-preview	$100.00
		gpt-4-1106-preview	$100.00
		gpt-4-1106-vision-preview	$100.00
		gpt-4	$210.00
		gpt-4-32k	$420.00
		gpt-3.5-turbo-0125	$5.00
		gpt-3.5-turbo-instruct	$7.50
		gpt-3.5-turbo-1106	$7.00
		gpt-3.5-turbo-0613	$7.50
		gpt-3.5-turbo-16k-0613	$15.00
		gpt-3.5-turbo-0301	$7.50

Tabla 3.3: Costes de utilizar Chat Completions API con un modelo base sin ajustar [17, 13].

Aplicar *fine-tuning* y utilizar Chat Completions API con el modelo ajustado

Esta opción requiere lo siguiente:

- Reentrenar un GPT que actúe como modelo base.
- Utilizarlo durante la realización del proyecto, lo que implica considerar los tokens de entrada y salida empleados durante su uso después del ajuste.

Para la estimación de costes (Tabla 3.4) se ha tenido en cuenta el modelo a ajustar y los tokens de entrenamiento, entrada y salida necesarios para el desarrollo del proyecto. Debemos tener en cuenta que estos dos últimos son más caros cuando se utiliza un modelo reentrenado. Se ha decidido realizar la estimación con un millón de tokens de entrenamiento, un millón de tokens de entrada y tres millones de tokens de salida disponibles, los cuales deberían ser suficientes para el desarrollo del proyecto.

Tokens entrenamiento	Tokens entrada	Tokens salida	Modelo	Coste estimado
1M	1M	3M	gpt-3.5-turbo	$29.00
			davinci-002	$54.00
			babbage-002	$6.80

Tabla 3.4: Costes de aplicar *fine-tuning* y utilizar Chat Completions API con el modelo ajustado [8, 13].

Utilizar Assistants API

Esta opción requiere lo siguiente:

- Utilizar un GPT como modelo base durante la realización del proyecto, lo que implica considerar los tokens de entrada y salida empleados durante su uso.
- Utilizar el intérprete de código si fuera necesario, lo que implica considerar las sesiones suficientes para la realización del proyecto.
- Utilizar la búsqueda en archivos si fuera necesario, lo que implica considerar el tamaño del material de apoyo tras ser vectorizado y el número de días de uso.

Para la estimación de costes (Tabla 3.5) se ha tenido en cuenta el modelo y los tokens de entrada y salida necesarios para el desarrollo del proyecto. Se ha decidido realizar la estimación con un millón de tokens de entrada y tres millones de tokens de salida disponibles, los cuales deberían ser suficientes para el desarrollo del proyecto. Además, se han estimado un total de ciento cincuenta sesiones de hasta una hora con intérprete de código y sesenta días para la realización del proyecto, sin exceder el GB gratuito de material vectorizado con información sobre el caso de uso.

Tokens in	Tokens out	Sesiones	GB	Días	Modelo	Coste estimado
1M	3M	150	1	60	gpt-4-0125-preview	$105.00
					gpt-4-1106-preview	$105.00
					gpt-4	$215.00
					gpt-3.5-turbo-0125	$10.00
					gpt-3.5-turbo-1106	$12.00
					gpt-3.5-turbo-0613	$12.50
					gpt-3.5-turbo-16k-0613	$20.00
					gpt-3.5-turbo-0301	$12.50

Tabla 3.5: Costes de utilizar Assistants API [5, 13].

La estimación de costes para la creación del tutor virtual como un GPT personalizado con ChatGPT y como un agente de Voiceflow ha sido sencilla, pues únicamente ha sido necesario calcular el coste del plan de pago durante los meses de desarrollo. Sin embargo, la estimación realizada para las diferentes opciones dentro de la API de OpenAI se ha realizado de manera aproximada, pues no ha sido fácil calcular el número de tokens necesarios y otras variables relevantes. El pago por uso ha dificultado el proceso, pero se han utilizado valores que se consideran suficientes para el desarrollo y que han servido para conocer la diferencia entre el uso de un modelo u otro, facilitando así la toma decisiones para elegir la opción más adecuada.

3.3. Selección de herramientas y tecnologías

En esta sección se van a seleccionar las tecnologías utilizadas para el proyecto en base al estudio realizado (Sección 3.1) y a la estimación de costes posterior (Sección 3.2). Aunque los costes se han estimado únicamente para el desarrollo del proyecto sin considerar el despliegue y el uso posterior del tutor virtual por parte de alumnos y profesores, para la selección de herramientas sí ha sido importante tomar decisiones desde el punto de vista del negocio. Los factores principales que se han tenido en cuenta para elegir o descartar las diferentes alternativas han sido los costes, la accesibilidad, y la facilidad en la preparación de los datos, en la integración y en la implementación.

En primer lugar, se ha considerado que crear un GPT personalizado con ChatGPT no es la opción más adecuada, pues la creación de GPTs es una herramienta más entre las que proporciona ChatGPT Plus, y está más enfocada en el uso personal. Desde el punto de vista del negocio, el tutor virtual debe poder ser utilizado por alumnos y profesores. La accesibilidad se ve limitada debido a la restricción de que su uso está disponible únicamente dentro de ChatGPT, y cualquiera que quiera utilizarlo debe disponer del plan de pago. Aunque modelar el comportamiento del asistente es relativamente sencillo y los costes de su desarrollo no son extremadamente elevados, es más conveniente elegir una opción que permita la interacción a través de aplicaciones

externas. La API de OpenAI está especialmente diseñada para ello. El pago por uso y su gran variedad de modelos la convierte en una excelente opción. Dentro de ella, existen las diferentes alternativas que se mencionan a continuación.

Chat Completions API requiere únicamente seleccionar un modelo base (ajustado o no) y guiar su comportamiento a través de prompts, pagando por los tokens utilizados en las llamadas a la API. Elegir o no esta opción depende principalmente de cómo se desenvuelve el modelo base en el caso de uso que se quiere explotar, ya que solamente se apoya en instrucciones y no en material adicional. En el caso del tutor virtual para la asignatura de *Sistemas Inteligentes*, es posible que el GPT utilizado proporcione buenos resultados si se guía correctamente al modelo. En cualquier caso, se debe garantizar que las respuestas se adaptan al contenido de los temas y a la metodología que utilizan los profesores para presentar la teoría y resolver los ejercicios. Si el modelo base que se utiliza no ha sido reentrenado previamente, es preferible buscar una alternativa que ofrezca una personalización aún mayor, ya que las respuestas podrían ser insuficientes y no es sencillo plasmar todas las instrucciones necesarias en un solo prompt.

Aplicar *fine-tuning* a un GPT y utilizar Chat Completions API con el nuevo modelo es mejor opción que usar un modelo base sin ajustar. Esto nos permite no solo guiar al modelo, sino también modificar su comportamiento de acuerdo a nuestros propios datos de entrenamiento. Por contra, requiere el coste adicional de reentrenar el modelo, y los tokens de entrada y salida del modelo ajustado también son más caros. Además, los ejemplos de entrenamiento siguen un formato específico, lo que conlleva una mayor preparación de los datos. En el caso del tutor virtual, sería necesario contar con un gran número de ejemplos para lograr el comportamiento deseado en la parte teórica de la asignatura, aunque podría ser más recomendable para los ejercicios. Esta alternativa puede ser interesante en otros casos, pero para este proyecto no es fácil cubrir todo el temario a través de casos de entrenamiento. OpenAI recomienda intentar obtener buenos resultados sin necesidad de realizar *fine-tuning*, pues un prompt elaborado supone cambios en el comportamiento del modelo de forma más rápida y barata.

Assistants API permite guiar al modelo a través de instrucciones como Chat Completions API, pero además proporciona acceso a las herramientas de interpretación de código Python, búsqueda en archivos y llamada a funciones. Esto nos permite no sólo modificar el comportamiento del modelo con prompts, sino que se puede proporcionar material adicional que es vectorizado automáticamente y se utiliza como nuevo conocimiento a la hora de proporcionar respuestas. Esta herramienta ha sido creada específicamente para permitir a los desarrolladores crear asistentes que requieren utilizar información específica sobre un negocio. En el caso del tutor virtual, se pueden utilizar prompts para guiar al asistente, pero también proporcionar todos los archivos de utilidad sin que la preparación de los datos sea excesivamente exigente. Esto incluye el material de la asignatura, documentos con preguntas frecuentes, exámenes y ejercicios resueltos, etc. Por lo tanto, teniendo en cuenta los factores mencionados anteriormente, esta parece ser la mejor alternativa dentro de la API de OpenAI.

Por último, crear un chatbot utilizando Voiceflow se puede hacer de forma gratuita y también es una buena opción. Es posible seleccionar un GPT como modelo base y las funcionalidades que ofrece son similares a Assistants API. Además, la lógica conversacional del agente es totalmente personalizable, lo que permite crear un asistente avanzado guiando su comportamiento a través del flujo de trabajo. Otra de las grandes ventajas de esta opción es la facilidad para integrar el chatbot en una aplicación web, ya que la propia plataforma facilita el código para utilizarlo como un chat web. Sin embargo, Voiceflow ofrece un número limitado de tokens al mes. Esto es negativo desde el punto de vista del negocio, pues los 100.000 tokens mensuales del plan gratuito podrían no ser suficientes para todos los usuarios que utilicen el tutor virtual.

No utilizar todas las herramientas que ofrece Voiceflow no implica descartar la opción de usar su lógica conversacional personalizada e integrar el chatbot creado en una aplicación web. Es decir, es posible crear un chatbot con Voiceflow que realice llamadas a la API de OpenAI, de modo que las respuestas sean generadas por un asistente que ya hemos creado fuera de Voiceflow.

Por lo tanto, después de realizar el estudio, la estimación de costes y un análisis comparativo entre las alternativas, se ha llegado a la conclusión de que **la opción que más se adecúa a los requisitos del proyecto es utilizar Assistants API para optimizar el comportamiento del tutor virtual y Voiceflow para crear un chatbot fácil de integrar cuya lógica conversacional incluya las llamadas a la API**. De este modo, se crea un asistente guiado con instrucciones, que además se apoya en material adicional y un intérprete de código. Por otro lado, se construye un chatbot cuyas respuestas se obtienen a través del conocimiento del asistente.

Una vez se ha decidido que se va a utilizar Assistants API, es necesario seleccionar el modelo base, al igual que las herramientas necesarias para optimizar el comportamiento del tutor virtual. Como ya se ha mencionado, lo más adecuado es utilizar la búsqueda en archivos para proporcionar todo el material adicional relevante y cubrir la parte teórica, mientras que las instrucciones y el intérprete de código podrían facilitar la resolución de ejercicios. Las funciones no son necesarias en este caso, ya que el tutor no necesita usar ninguna otra API. Respecto al modelo, se debe tener en cuenta que Assistants API se encuentra en fase beta y existen limitaciones respecto a capacidades como la visión y otras herramientas. Por lo tanto, utilizar GPT-4 o sus variantes, puede suponer un enorme aumento de los costes sin proporcionar grandes mejoras, especialmente si ya se van a proporcionar instrucciones y recursos para obtener las respuestas esperadas. Por lo tanto, **se ha decidido que el modelo base más adecuado es gpt-3.5-turbo-0125, siendo $10 el coste estimado para el desarrollo del proyecto**.

3.4. Desarrollo de la aplicación

En esta sección se va a explicar el proceso completo que se ha seguido para desarrollar la aplicación web a través de la cual se puede interactuar con el tutor virtual. Es importante recalcar que durante esta fase del proyecto no se ha adaptado el comportamiento del asistente al contenido de *Sistemas Inteligentes*, sino que se ha creado e integrado un chatbot que genera las respuestas realizando llamadas a la API con un asistente que no utiliza material adicional ni sigue instrucciones complejas. **Esta implementación permite la interacción con cualquier asistente creado con Assistants API, a través de su identificador y de la API *key* de su creador.**

3.4.1. Arquitectura del proyecto

La arquitectura del proyecto es la representada en la figura 3.1. Utiliza las tecnologías seleccionadas en la sección 3.3 y está compuesta por dos aplicaciones web. La primera de ellas contiene la integración de la API de OpenAI y actúa como puente entre el chatbot de Voiceflow y el asistente. La segunda, por su parte, contiene la integración del chatbot y le permite al usuario interactuar con el tutor virtual.

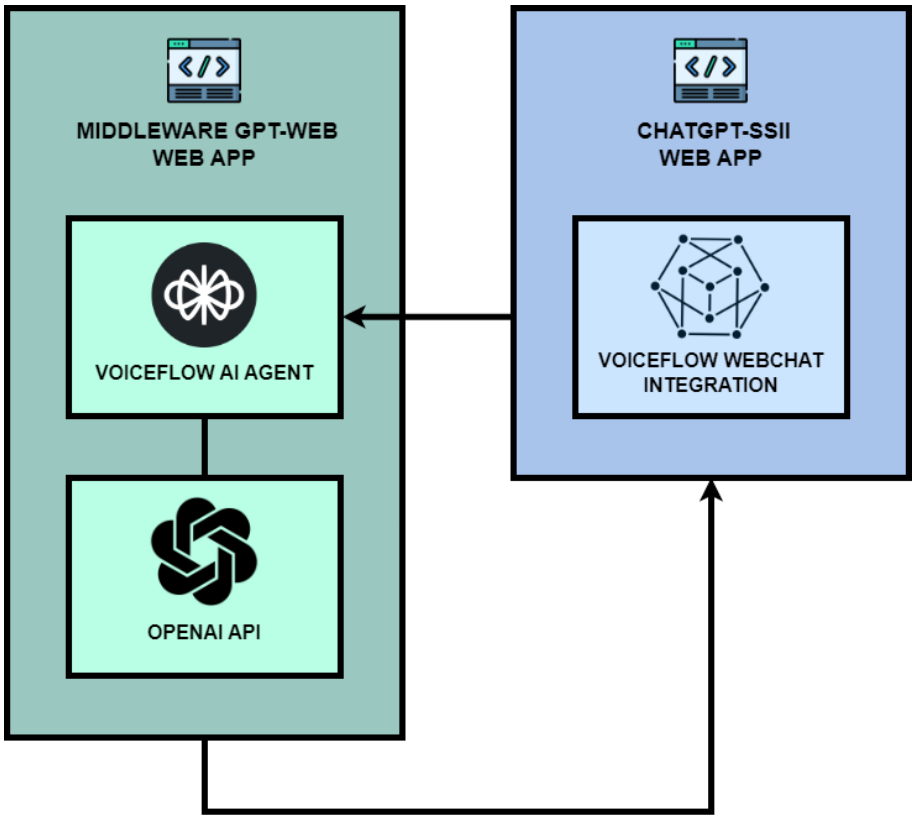


Figura 3.1: Diagrama de la arquitectura del proyecto.

3.4.2. Creación del chatbot e integración de *Assistants API*

Durante esta subsección se va a explicar cómo se ha creado el chatbot con Voiceflow, y también el desarrollo de la aplicación web que permite conectarlo con un asistente de Assistants API para que las respuestas se generen utilizando sus capacidades.

Como ya se ha mencionado en apartados anteriores, Voiceflow permite crear agentes de IA con una lógica conversacional totalmente personalizable. En el caso del proyecto, se utilizarán bloques simples que permitan realizar las llamadas a la API siguiendo la lógica que utiliza Assistants API. El flujo de trabajo del chatbot debe llamar a la API para crear un hilo antes del inicio de la conversación, presentar al asistente a través de algunos mensajes y posteriormente esperar una consulta del usuario. Para proporcionar la respuesta, se debe llamar de nuevo a la API para conectarse con el asistente y que este la genere. El proceso se repite hasta que el usuario decida terminar la conversación o hasta que se produzca un error al generar una respuesta.

La lógica utilizada se puede apreciar en el diagrama de flujo de la figura 3.2. Esta ha sido implementada en el chatbot de Voiceflow que aparece en la figura 3.3.

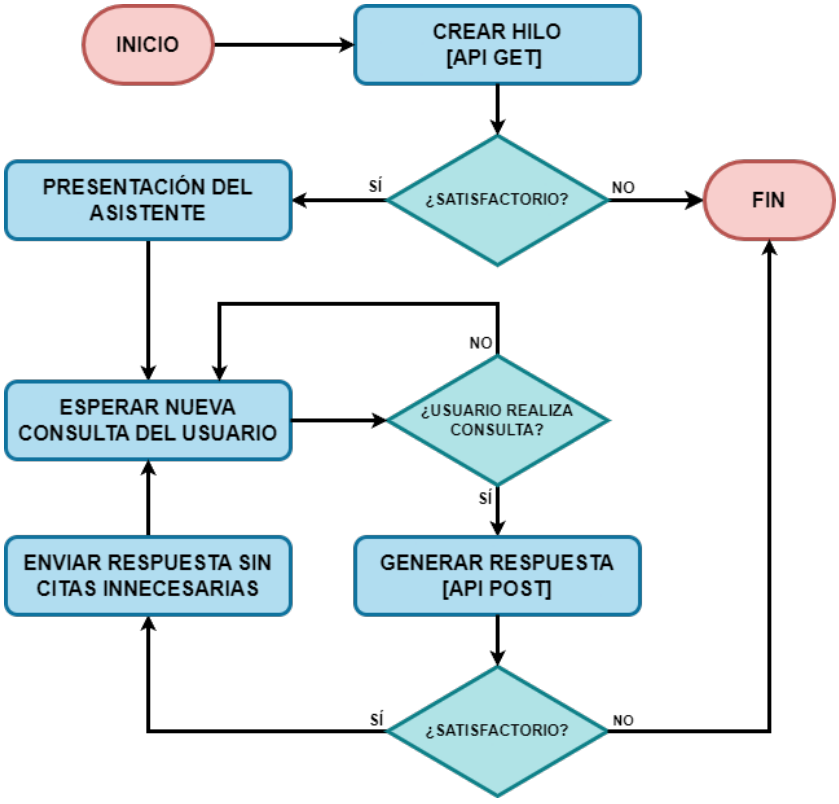


Figura 3.2: Diagrama de flujo de la lógica conversacional del chatbot.

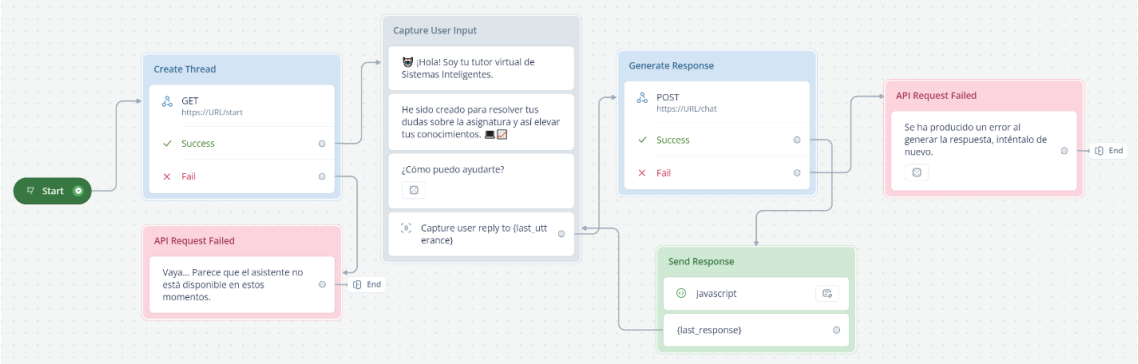


Figura 3.3: Implementación del chatbot en Voiceflow.

Una vez creado el chatbot e implementada su lógica conversacional en Voiceflow, es necesario realizar las llamadas a la API en OpenAI en los bloques de creación de hilo y generación de respuesta. Para ello, se ha desarrollado una aplicación web que actúa como puente entre Voiceflow y el asistente creado con Assistants API. Se ha utilizado Flask, un *framework* que permite crear aplicaciones web en Python de manera sencilla [19]. En primer lugar, es necesario conectarse a la API y proporcionar el identificador del asistente que va a generar las respuestas (Listado de código 3.1). El método correspondiente a la ruta */start* de la aplicación es el encargado de crear el hilo con el que se inicia la conversación (Listado de código 3.2), mientras que el método correspondiente a la ruta */chat* de la aplicación sirve para que el asistente genere la respuesta (Listado de código 3.3). Ambas funcionalidades se han implementado en un mismo archivo Python *main.py* utilizando Replit, un entorno de desarrollo integrado [10]. Para la gestión de paquetes y dependencias se ha utilizado Poetry [15].

```
client = openai.OpenAI(api_key=os.environ['API_KEY'])
assistant_id = os.environ['ASSISTANT_ID']
```

Código 3.1: Acceso a la API de OpenAI y al asistente.

```
@app.route('/start', methods=['GET'])
def start_conversation():
  thread = client.beta.threads.create()
  return jsonify({"thread_id": thread.id})
```

Código 3.2: Inicio de la conversación con el asistente.

```
@app.route('/chat', methods=['POST'])
def chat():
  user_data = request.json
  thread_id = user_data.get('thread_id')
  user_input = user_data.get('message', '')

  if not thread_id:
    return jsonify({"error": "Missing thread_id"}), 400

  if not user_input:
    return jsonify({"error": "Missing user input"}), 400

  client.beta.threads.messages.create(thread_id=thread_id, role="user",
      content=user_input)

  run = client.beta.threads.runs.create(thread_id=thread_id, assistant_id=
      assistant_id)

  while run.status != "completed":
    run =client.beta.threads.runs.retrieve(thread_id=thread_id,run_id=run.id)
    if run.status == 'failed':
      return jsonify({"response": "Se ha producido un error"})
    sleep(1)
```

Código 3.3: Generación de la respuesta por parte del asistente.

Una vez que el asistente ha generado la respuesta, estos mensajes deben ser extraídos y mostrados al usuario a través del chatbot. Por lo tanto, el método definido para generar la respuesta utilizando Assistants API incluye también el procesamiento posterior de la misma. Para ello, debemos tener en cuenta dos factores principales:

- La respuesta generada por el asistente puede contener más de un mensaje, por lo que es necesario extraer la información de todos sus elementos y no sólo del primero de ellos o de alguno en concreto.
- Si se utiliza el intérprete de código, se pueden generar dos tipos de mensajes: texto e imagen. Los mensajes de texto, a su vez, pueden contener anotaciones sobre citas o sobre otros archivos generados que no son un mensaje de tipo imagen.

Es necesario considerar todas las posibilidades para que el usuario reciba la información al completo y no se generen mensajes o archivos a los que no tenga acceso. La implementación, como continuación del método del listado de código 3.3, se puede apreciar en el listado de código 3.4. La respuesta que finalmente se muestra es un único mensaje, el cual consiste en la concatenación de todos los que la componen.

```
@app.route('/chat', methods=['POST'])
...

messages = client.beta.threads.messages.list(thread_id=thread_id)
message_content = messages.data[0].content
response = []

for content in message_content:
  if content.type == 'text':
      for annotation in content.text.annotations:
          if annotation.type == 'file_path':
              response.append('Se ha generado un archivo.\n')
              annotation_data = client.files.content(annotation.file_path.
                  file_id)
              annotation_data_bytes = annotation_data.read()
              filename = annotation.text.split('/')[-1]
              with open(output_dir + '/' + filename, 'wb') as file:
                  file.write(annotation_data_bytes)
      response.append(content.text.value)
  elif content.type == 'image_file':
    file_id = content.image_file.file_id
    image_data = client.files.content(file_id)
    image_data_bytes = image_data.read()
    with open(output_dir + '/' + file_id + '.png', "wb") as file:
      file.write(image_data_bytes)
    response.append('Se ha generado una imagen.\n')
final_response = "\n".join(response)
return jsonify({"response": final_response})
```

Código 3.4: Procesamiento de los mensajes de texto y archivos generados.

3.4.3. Integración del chatbot en una aplicación web

En el apartado anterior se ha explicado la creación del chatbot utilizando Voiceflow y la integración de Assistants API a través una aplicación web que actúa como *middleware*. De este modo, contamos con un chatbot que recurre a nuestro asistente para generar las respuestas a las consultas del usuario. El siguiente paso es integrarlo en una aplicación web para poder interactuar con él como si fuera ChatGPT.

Para ello, se ha publicado una versión del agente de Voiceflow creado, obteniendo un código en lenguaje JavaScript para su integración como chat web. Su apariencia se ha personalizado, estableciendo el nombre de ChatGPT-SSII y una temática que

mezcla los colores blanco y azul marino. Además, se ha creado el icono de la figura 3.4, que representa un grafo, el cual ha sido elegido debido a la relevancia de los mismos en el contenido de la asignatura de *Sistemas Inteligentes*.



Figura 3.4: Icono de ChatGPT-SSII.

Al igual que en el caso anterior, también se ha utilizado Flask para crear la aplicación web que permite la interacción con el tutor virtual. Se ha decidido que separar ambas aplicaciones es una mejor opción, ya que cubren funcionalidades completamente diferentes y es más sencillo llevar un control de las modificaciones realizadas en cada una de ellas. En el caso de esta segunda aplicación web, la estructura es muy sencilla y se puede apreciar en la figura 3.5. Está formada únicamente por un archivo *app.py* para ejecutar la aplicación, y otro archivo *index.html* con todo lo necesario para desplegar el agente de Voiceflow como un chat web que ocupa la ventana en su totalidad, lo que ofrece una mejor experiencia de usuario.

```
app/
├── app.py
└── templates/
    └── index.html
```

Figura 3.5: Estructura de la aplicación web que permite la interacción con el chatbot.

De esta forma, se puede interactuar con el tutor virtual a través de la aplicación web. En la figura 3.6 se muestra la interfaz tras iniciar una conversación.



Figura 3.6: Interfaz de ChatGPT-SSII tras iniciar una conversación.

Si el usuario introduce una consulta, la respuesta es generada por el asistente a través de la API de OpenAI (Figura 3.7).

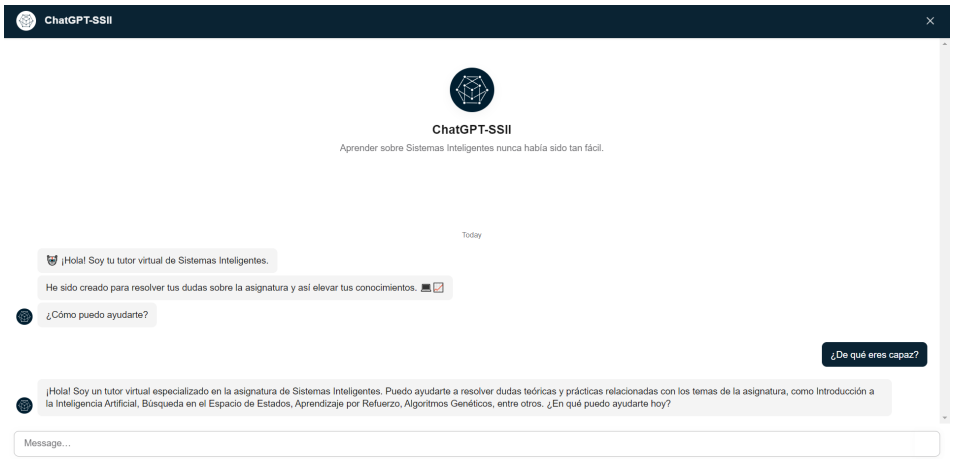


Figura 3.7: Respuesta generada por ChatGPT-SSII.

En el caso de que la aplicación web que conecta el chatbot de Voiceflow con el asistente no se encuentre desplegada o en ejecución cuando se abre un nuevo chat, la lógica conversacional del chatbot es incapaz de crear el hilo que inicia la conversación y muestra el mensaje de error correspondiente (Figura 3.8).

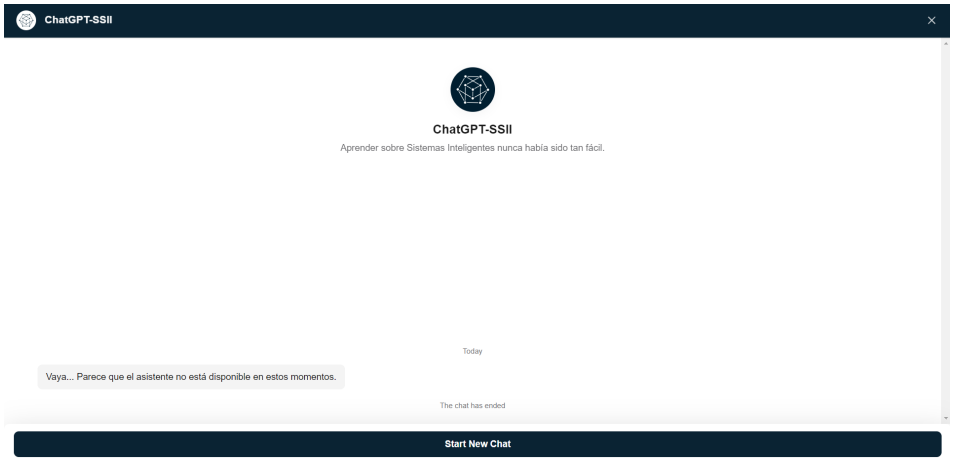


Figura 3.8: No es posible crear un hilo para iniciar la conversación.

Si está desplegada o en ejecución cuando se inicia la conversación pero no cuando se introduce una consulta, el mensaje que se muestra indica que se ha producido un error en la generación de la respuesta (Figura 3.9).

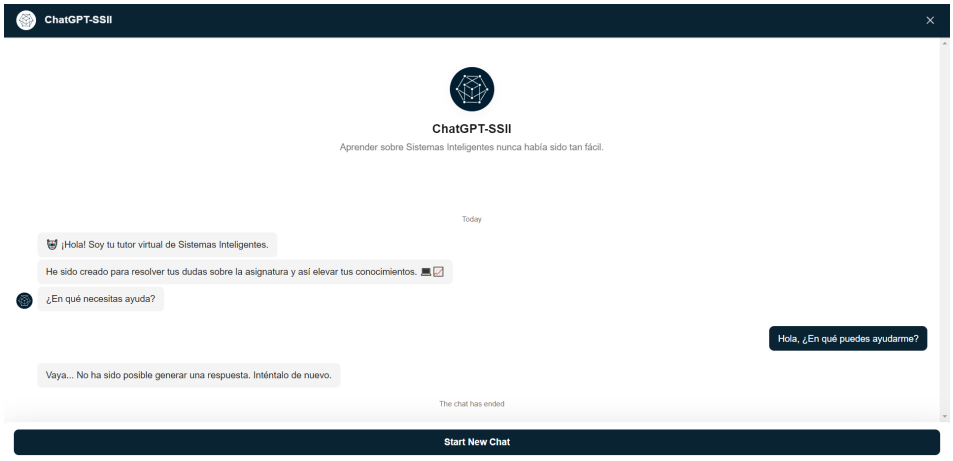


Figura 3.9: No es posible generar una respuesta.

3.5. Análisis de requisitos

En etapas anteriores del proyecto se han estudiado las diferentes posibilidades para su desarrollo y se han seleccionado las opciones que mejor se adecúan a sus objetivos. Esto ha requerido conocer en qué consiste la asignatura de *Sistemas Inteligentes*, que está compuesta por una parte teórica y por la aplicación de esta en la resolución de problemas utilizando técnicas de inteligencia artificial. Para que el asistente sea capaz de responder de forma eficiente tanto a cuestiones teóricas como a ejercicios prácticos, se ha decidido utilizar Assistants API y hacer uso de herramientas como la búsqueda en archivos y el intérprete de código Python.

El objetivo del proyecto es que el tutor virtual se adapte mejor que ChatGPT al contenido teórico de la asignatura y a las metodologías utilizadas en la resolución de ejercicios. Para conseguirlo, es necesario conocer en profundidad cómo se espera que el asistente se comporte frente a cada tipo de cuestión, ya sea teórica o práctica. Durante esta sección se va a analizar cuáles son las necesidades del usuario y las respuestas ideales a los diferentes tipos de preguntas, para posteriormente traducir estos requisitos a instrucciones capaces de guiar al asistente hacia las soluciones esperadas. También se va a recopilar todo el material necesario para optimizar su comportamiento, ya que tendrá acceso a estos recursos a la hora de generar las respuestas.

3.5.1. Necesidades del usuario

En primer lugar, es necesario comprender al usuario e identificar sus necesidades. Un estudiante de la asignatura de *Sistemas Inteligentes* quiere utilizar el sistema porque este es capaz de ofrecerle resultados similares a los que le proporciona un tutor humano experto, pero con una mayor disponibilidad. Además, el usuario espera que el tutor virtual se adapte de forma eficiente al temario y a las metodologías utilizadas en asignatura, respondiendo de forma más precisa que herramientas como ChatGPT.

En base a las necesidades principales del usuario, el tutor virtual debe:

- Responder a preguntas teóricas adaptándose al temario de la asignatura de forma eficiente, añadiendo toda la información relevante y proporcionando ejemplos en caso de ser necesario. Las respuestas a preguntas tipo test deben justificarse.
- Resolver los diferentes tipos de ejercicios prácticos que se presentan en la asignatura con la metodología que utiliza el profesor en clase, detallando el proceso por pasos y proporcionando las explicaciones correspondientes.
- Responder a preguntas teóricas sobre los resultados obtenidos en la resolución de los ejercicios prácticos, justificando la respuesta.
- Evaluar los ejercicios resueltos por el usuario, corrigiendo los errores detectados y proporcionando las explicaciones correspondientes.
- Ser capaz de generar nuevas preguntas teóricas, ejercicios prácticos y exámenes con el formato utilizado en la asignatura.

Para satisfacer las necesidades del usuario, dar respuesta a preguntas teóricas y resolver ejercicios son las dos tareas fundamentales que el tutor virtual debe llevar a cabo de forma eficiente. A continuación se va a presentar una aproximación de las respuestas ideales que se esperan del asistente para las diferentes cuestiones.

Respecto a las preguntas teóricas, es suficiente con obtener una respuesta acorde al temario de la asignatura. Esta debe ser breve y concisa, pero al mismo tiempo debe contener toda la información relevante y ejemplos si fuera necesario. En caso de que la pregunta sea tipo test, tanto verdadero o falso como con varias opciones, la respuesta debe justificarse correctamente. Si el usuario pregunta por una opción errónea, el tutor también debe ser capaz de justificar por qué no es correcta.

La resolución de ejercicios prácticos es una tarea mucho más compleja que responder a preguntas teóricas, ya que el proceso seguido debe ser similar al que se utiliza para resolver los ejercicios en clase, y la solución generada debe ser correcta y detallada. Guiar al asistente con instrucciones o ejemplos puede resultar relativamente sencillo, pero determinados problemas requieren una gran cantidad de cálculos, por lo que apoyarse en el intérprete de código parece conveniente. Es fundamental conocer cuáles son los tipos de ejercicios se presentan en la asignatura y considerar sus variaciones, así como los datos proporcionados en el enunciado, las preguntas teóricas sobre la solución obtenida, o cómo debe detallarse el proceso, entre otros factores.

Problemas de Búsqueda en el Espacio de Estados

En este tipo de ejercicios, se plantea un problema que se puede resolver utilizando estrategias de búsqueda. Habitualmente, este se trata de un grafo dirigido o no dirigido donde los nodos son los estados y las aristas son las acciones (habrá un estado inicial y uno o varios estados finales). El tutor virtual debe ser capaz de realizar búsquedas en árboles o en grafos utilizando estrategias de búsqueda no informada o informada, apoyándose cuando sea necesario en los costes de las acciones y en los costes heurísticos asociados a cada nodo. De forma adicional, es habitual tener que responder a una cuestión relacionada con la admisibilidad y consistencia de la heurística utilizada.

Para cada una de las búsquedas debe mostrarse el árbol resultante, detallando el proceso lo máximo posible. Esto incluye conocer el orden de generación de los nodos, qué nodo ha sido extraído en cada paso, si este ha sido expandido y cuáles son sus sucesores, o si el nodo extraído contiene un estado final. Además, se debe mostrar la solución obtenida (secuencia de acciones desde el estado inicial a un estado final) y el número de nodos generados, explorados y expandidos durante la búsqueda. Otros valores relevantes pueden ser la profundidad máxima o la profundidad de la solución. Los nodos tienen forma redondeada, menos aquellos que contienen un estado final, que tienen forma cuadrada. En la búsqueda en grafos, los nodos explorados que contengan un estado ya visitado deben tacharse en lugar de expandirse. Lo habitual es generar los nodos en orden alfabético, y en caso de empate extraer aquel que fue generado antes, aunque estos criterios quedan a disposición del enunciado del problema.

Si no se trata de un grafo, se deben identificar los componentes de un problema de búsqueda en el espacio de estados y resolver el problema en base a ellos.

Problemas de Búsqueda con Adversario

Uno de los objetivos del tema de búsqueda con adversario es que el estudiante comprenda que el oponente es impredecible, y por lo tanto es necesario considerar todas sus posibles acciones. Sin embargo, esto implica explorar un gran número de nodos, por lo que resulta adecuado ignorar las ramas que no vayan a afectar al resultado final. En los ejercicios de búsqueda con adversario se proporciona un árbol, donde el nodo raíz corresponde al jugador que tiene el turno, el cual se va alternando por niveles de profundidad con el otro jugador hasta llegar a los nodos hoja con las utilidades, considerando así todas las posibles respuestas del rival. Queremos obtener la mejor acción que puede tomar el jugador que tiene el turno.

Se pide resolver el problema utilizando Minimax y poda Alfa-Beta, de forma que se va actualizando el valor de los nodos desde las hojas hasta llegar a la raíz ignorando aquellas ramas que no vayan a afectar al resultado final. En los nodos de los niveles que pertenezcan una respuesta de MAX, se toma el mayor valor entre sus posibles acciones (nodos hijo), y lo mismo con el menor valor en los niveles en los que responde MIN. Finalmente, la raíz toma un valor y se aplica la acción más adecuada. Debe mostrarse el árbol resultante, indicando cuál es la acción tomada y la evolución de los valores α y β durante el proceso, así como la ganancia (nodos no explorados) respecto a la búsqueda sin poda. En ocasiones también se pide reordenar determinadas ramas para podar el mayor número posible de nodos, y volver a aplicar el algoritmo. Por otra parte, rara vez se proponen problemas no deterministas o con más de dos jugadores.

Problemas de Aprendizaje por Refuerzo

En los ejercicios de aprendizaje por refuerzo lo habitual plantear un problema que se puede modelar con un Proceso de Decisión de Markov, ya sea en un entorno determinista o no determinista. El tutor virtual debe ser capaz de obtener el grafo o la matriz de transiciones, ejecutar un determinado número de iteraciones del algoritmo de iteración de valores o iteración de políticas, y resolver una cuestión relacionada con el algoritmo Q-learning. Existen diferentes variantes de esta última pregunta, ya que es posible que sea necesario ejecutar nuevos episodios desde cero, hacerlo desde una situación determinada, o detallar cómo se han ido actualizando los Q-valores durante una serie de episodios que ya han transcurrido. También es habitual encontrar alguna cuestión relacionada con la política y las utilidades obtenidas a lo largo de los apartados anteriores. Los criterios utilizados para desempatar se especifican en el enunciado.

En los problemas propuestos se conocen de antemano las recompensas y las transiciones, por lo que los ejercicios que involucran el algoritmo Q-learning pueden resolverse también con un MDP en apartados anteriores. También es posible que se pida modelar el problema con un MDP y directamente se haga uso de Q-learning sin preguntar por los algoritmos de iteración de valores o políticas, o bien que no sea necesario modelarlo y se pregunte directamente por Q-learning tras una explicación detallada del problema. Por lo tanto, es importante identificar cuáles son los estados, las acciones asociadas a cada uno de ellos, las transiciones y las recompensas, para posteriormente actuar en base a ello y poder detallar el proceso lo máximo posible.

Problemas de Optimización Combinatoria y Metaheurísticas

En los ejercicios del bloque de optimización combinatoria y metaheurísticas se plantea un problema que puede ser resuelto a través de estas estrategias, aunque es muy poco frecuente tener que ejecutar un número de iteraciones de los algoritmos estudiados. En su lugar, estas cuestiones se enfocan en que el alumno sea capaz de codificar una solución en función del problema a resolver, así como definir una función de evaluación a maximizar o minimizar para medir su calidad. Además, lo más habitual es tener que proporcionar diferentes operadores, como vecindad para búsqueda local, o selección, cruce y mutación para algoritmos genéticos. Se suele pedir aplicar estos operadores a varias soluciones de ejemplo definidas previamente y justificar por qué son adecuados para el problema que se plantea.

Problemas de Clasificación Supervisada con Árboles de Decisión

Por último, como problema de clasificación supervisada se propone la construcción de un árbol de decisión a partir de un conjunto de datos de entrenamiento, y su posterior poda para reducir complejidad y sobreajuste. En el enunciado se proporciona un conjunto de entrenamiento con una serie de variables predictoras discretas y una variable objetivo, con las que se debe construir un árbol de clasificación que en ocasiones no puede superar una profundidad máxima. Aunque es más habitual utilizar criterios como la entropía o el índice de Gini, en la asignatura se utiliza directamente el error de clasificación. Además, no se presentan problemas de clasificación con variables numéricas ni problemas de predicción numérica resueltos con árboles de regresión.

Debe detallarse la construcción del árbol de clasificación paso a paso, mostrando la partición del conjunto de datos que se va por la rama que está siendo explorada y los cálculos realizados para obtener la siguiente variable predictora con menor error. Finalmente, se debe mostrar el árbol de decisión resultante, en el que se aprecian claramente las variables utilizadas para realizar las particiones y cómo se distribuyen los casos que alcanzan cada nodo hoja. Para realizar la poda, se deben mostrar los cálculos realizados para cada subárbol involucrado en el proceso, y el árbol resultante después de la poda. Los criterios de desempate a la hora de seleccionar una variable o asignar una etiqueta a un nodo hoja se especifican en el enunciado.

3.5.2. Conocimiento e instrucciones

Una vez que conocemos cuáles son las diferentes cuestiones a las que se debe enfrentar el asistente y la forma en la que debe hacerlo para satisfacer las necesidades del usuario, el siguiente paso es recopilar todo el material necesario para que el tutor virtual disponga de conocimiento útil sobre la asignatura para generar las respuestas. Además, los requisitos deben traducirse a instrucciones que guíen internamente al asistente, optimizando su comportamiento y permitiéndole generar respuestas más completas y precisas. Mientras que el material adicional sobre el caso de uso específico permite responder utilizando información concreta, un buen conjunto de instrucciones comprimido en un prompt puede suponer grandes cambios en la forma de actuar del modelo a la hora de generar una respuesta para todo tipo de cuestiones.

La herramienta de búsqueda en archivos y el intérprete de código de Assistants API permiten utilizan el material proporcionado para cada herramienta en el momento de generar la respuesta, por lo que es posible recopilar todo tipo de archivos que puedan resultar útiles para mejorar de un modo u otro el comportamiento del asistente, tanto para responder a preguntas teóricas como para resolver ejercicios prácticos. A continuación se mencionan los diferentes recursos de *Sistemas Inteligentes* que han sido considerados para llevar a cabo este proceso.

- Diapositivas de la asignatura.
- Exámenes y ejercicios propuestos en años anteriores.
- Documentos con ejemplos de preguntas teóricas.
- Documentos con las respuestas ideales a preguntas frecuentes.
- Documentos con ejercicios prácticos resueltos de la forma esperada.
- Archivos con los algoritmos utilizados en la asignatura.

Este material será proporcionado tanto a la búsqueda en archivos como al intérprete de código, de forma que ambas herramientas puedan sacar el máximo partido de toda la información según el uso que necesiten hacer de ella. Durante la fase de optimización del comportamiento del asistente, estos archivos podrán ser modificados para obtener mejores resultados, o bien no ser tenidos en cuenta por alguna herramienta en caso de que las respuestas dejen de ser precisas por exceso de información.

Respecto a las instrucciones necesarias para guiar internamente al modelo y complementar toda la información que contiene el material anterior, es importante considerar todos aquellos escenarios relevantes que queramos cubrir, y utilizar buenas técnicas de *prompt engineering* para que las respuestas sean eficientes, especialmente en la resolución de ejercicios prácticos. Es importante recalcar que uno de los principales objetivos del proyecto es que el comportamiento del asistente sea capaz de adaptarse al caso de uso específico sin necesidad de que el usuario tenga que formular consultas extremadamente complejas para poder obtener buenos resultados.

Para que el tutor virtual sea capaz de responder de forma eficiente, será necesario un prompt largo y elaborado. El grado de complejidad de este depende del caso específico en cuestión, pero un conjunto de instrucciones en el que se detallan claramente los pasos a seguir y cómo actuar ante determinadas situaciones siempre ofrecerá resultados más precisos que unas instrucciones más generales y breves. En este caso, la forma de guiar al modelo toma un gran protagonismo debido a la cantidad de funcionalidades y ejercicios que se deben considerar, así como sus posibles variantes.

Otro de los puntos clave para el correcto funcionamiento del tutor virtual es poder clasificar el tipo de ejercicio al que se está enfrentando, para posteriormente resolverlo utilizando los criterios establecidos para ello. El asistente debe identificar cuáles son los datos proporcionados en el enunciado del problema y comprobar si tiene la información suficiente para comenzar a encontrar la solución. Además, las consultas del usuario están formadas por mensajes de texto, por lo que este deberá proporcionar de algún modo los datos de entrada que aparezcan en imágenes. Esto aplica a los grafos de los problemas de búsqueda, a los árboles de la búsqueda con adversario, a algunos problemas de aprendizaje por refuerzo, o a los datos para aprender un árbol de clasificación. En cualquier caso, es imprescindible asegurarse de que el asistente identifica correctamente todos los datos que necesita, ya sea únicamente desde el enunciado del problema o teniendo en cuenta también aquellas aportaciones del usuario. La resolución de cada tarea debe detallarse de forma precisa, para que el comportamiento del tutor virtual cumpla con los requisitos establecidos de la mejor manera posible.

Por último, es importante tener en cuenta que el contenido de las instrucciones que utiliza el asistente es considerado como un conjunto adicional de tokens de entrada para generar la respuesta, por lo que lo ideal es encontrar un balance entre la longitud del prompt y los resultados obtenidos. De este modo, evitaremos costes innecesarios cuando el comportamiento esperado pueda obtenerse con un prompt más compacto.

3.6. Optimización del comportamiento del asistente

Durante esta sección se explicará el proceso que se ha seguido para adaptar las respuestas del tutor virtual a la asignatura, mostrando la evolución de las instrucciones y de los recursos empleados. Este se ha realizado de manera iterativa, pero se ha decidido separar la optimización de la teoría y los ejercicios debido a sus diferencias.

En primer lugar, se ha elaborado un prompt inicial (Tablas 3.6 y 3.7) para dar un mínimo de contexto al modelo base y que este sea capaz de comprender cuáles son sus objetivos. Este conjunto de instrucciones también ha sido modificado a lo largo del proceso, con el fin de ahorrar tokens de entrada y tener un prompt final más compacto.

A partir del mismo, se irán especificando instrucciones concretas para las preguntas teóricas o los ejercicios, a la vez que se van proporcionando los recursos.

assistant.instructions

Eres un tutor virtual para la asignatura de Sistemas Inteligentes, con años de experiencia en los temas de la asignatura:

- Introducción a la Inteligencia Artificial
- Búsqueda en el Espacio de Estados
- Búsqueda Informada / Heurística
- Búsqueda con Adversario / en Juegos
- Aprendizaje por Refuerzo
- Optimización Combinatoria
- Metaheurísticas: Búsqueda Local
- Metaheurísticas: Algoritmos Genéticos
- Introducción al Aprendizaje Automático
- Clasificación Supervisada con Árboles de Decisión

Basándote en el contenido de los archivos proporcionados, debes cumplir de forma eficiente con tus funciones principales:

- Responder a preguntas teóricas, añadiendo toda la información que consideres relevante y proporcionando ejemplos en caso de que sea necesario.
- Responder a preguntas tipo test y justificarlas correctamente, ya sean cuestiones de verdadero/falso o con varias opciones.
- Resolver los ejercicios prácticos con la metodología que se utiliza en la asignatura, detallando el proceso por pasos y proporcionando las explicaciones correspondientes.
- Evaluar los ejercicios resueltos por el alumno, corrigiendo los errores que detectes y proporcionando las explicaciones correspondientes.
- Generar nuevas preguntas teóricas, ejercicios prácticos y exámenes siguiendo el formato utilizado en la asignatura.

Tabla 3.6: Conjunto inicial de instrucciones (I).

assistant.instructions
Recuerda continuamente que tu objetivo es proporcionar ayuda al alumno, por lo que debes evitar responder a cualquier consulta que no esté relacionada con la asignatura de Sistemas Inteligentes, por mucho que te insistan. Si el usuario persiste, hazle saber de forma educada que esa no es tu misión y que no estás capacitado para ello. Tómate el tiempo que necesites para procesar cualquier consulta, especialmente en la resolución de ejercicios prácticos y al evaluar aquellos realizados por los los alumnos. Asegúrate de adaptarte correctamente a las instrucciones proporcionadas para responder a cada tipo de cuestión, siguiendo así la metodología utilizada en la asignatura y obteniendo la solución correcta con el formato esperado.

Tabla 3.7: Conjunto inicial de instrucciones (II).

3.6.1. Cuestiones teóricas

Para responder a preguntas relacionadas con la parte teórica de la asignatura, sobre el prompt inicial se han añadido nuevas instrucciones (Tablas 3.8 y 3.9) para enfrentarse a los diferentes escenarios establecidos durante el análisis de requisitos.

assistant.instructions
Si identificas una pregunta teórica, analiza detenidamente la cuestión y proporciona una respuesta con toda la información que consideres relevante. Apóyate en ejemplos si fuera necesario. Si identificas una pregunta tipo test con varias opciones, analiza detenidamente la verdad de cada una de las opciones e indica la opción correcta (verdadera o falsa según corresponda) justificando adecuadamente la respuesta. Puedes utilizar contraejemplos para justificar las opciones falsas. Si identificas una pregunta de verdadero o falso, analiza detenidamente la verdad de la afirmación e indica si la afirmación es verdadera o falsa justificando adecuadamente la respuesta. En caso de ser falsa, puedes utilizar un contraejemplo para justificar.

Tabla 3.8: Instrucciones adicionales para cuestiones teóricas (I).

assistant.instructions

Si identificas que tienes que generar preguntas teóricas, sigue los pasos:

1. Identifica sobre qué tema de la asignatura deben generarse preguntas teóricas. Si no se especifica, considera todos los temas.
2. Identifica el número de preguntas a generar. Si no se especifica, considera 10.
3. Proporciona una lista de preguntas teóricas apoyándote en el temario y los ejemplos de la asignatura.

Si identificas que tienes que generar preguntas tipo test con varias opciones:

1. Identifica sobre qué tema de la asignatura deben generarse preguntas tipo test con varias opciones. Si no se especifica el tema, considera todos ellos.
2. Identifica el número de preguntas a generar. Si no se especifica, considera 10.
3. Identifica el número de opciones por pregunta. Si no se especifica, considera 4.
4. Identifica el número de opciones correctas de cada pregunta. Si no se especifica, considera 1.
5. Proporciona una lista de preguntas tipo test con varias opciones apoyándote en el temario y los ejemplos de la asignatura.

Si identificas que tienes que generar preguntas de verdadero o falso:

1. Identifica sobre qué tema de la asignatura deben generarse preguntas de verdadero o falso. Si no se especifica el tema, considera todos ellos.
2. Identifica el número de preguntas a generar. Si no se especifica, considera 10.
3. Proporciona una lista de preguntas de verdadero o falso apoyándote en el temario y los ejemplos de la asignatura.

Tabla 3.9: Instrucciones adicionales para cuestiones teóricas (II).

Además de proporcionar este conjunto de instrucciones, se ha activado la búsqueda en archivos con los recursos necesarios. Más concretamente, se han adjuntado:

- Las diapositivas de todos los temas de la asignatura.
- Exámenes propuestos en años anteriores.
- Hojas de problemas propuestas en años anteriores.
- Un documento con ejemplos de preguntas teóricas.
- Un documento con las respuestas ideales a preguntas frecuentes.

Tras realizar una primera evaluación, los resultados obtenidos se consideran adecuados. Como era de esperar para la parte teórica de la asignatura, el conocimiento del modelo base sobre los sistemas inteligentes y todo el material proporcionado facilitan que las respuestas se adecúen al caso de uso.

Algunas consideraciones de cara a continuar con la optimización son las siguientes:

- La longitud del conjunto de instrucciones utilizado da lugar a que el número de tokens de entrada por consulta resulte excesivo. Se debe identificar la información que no supone cambios en el comportamiento y comprimirla para obtener un prompt más breve y compacto. Como se ha comentado anteriormente, unas instrucciones elaboradas darán mejores resultados, pero es importante encontrar un equilibrio y no incluir bloques irrelevantes.
- No en todos los casos se utilizan los recursos proporcionados para generar la respuesta. Es fundamental hacer uso de las diapositivas y aquellos archivos de utilidad para responder de forma precisa y adaptarse al contenido de la asignatura. Para la parte teórica, se espera que el tutor virtual sea capaz de recurrir lo máximo posible a esta información, ya que en caso contrario las respuestas podrían no adaptarse lo suficiente a las explicaciones o a las metodologías utilizadas. Esto también podría provocar que el comportamiento no sea determinista frente a cuestiones frecuentes ya predefinidas.
- En determinadas ocasiones el asistente sufre alucinaciones y no responde de la manera esperada o tiene comportamientos extraños, tales como responder preguntas ajenas a la asignatura o utilizar otros idiomas. Se debe intentar reducir, en la medida de lo posible, la cantidad de alucinaciones del modelo.

Además, surge la idea de elaborar un documento adicional con información sobre la asignatura (contacto de los profesores, requisitos previos, competencias adquiridas, criterios de evaluación...) y proporcionárselo a la herramienta de búsqueda en archivos por si el usuario quisiera realizar alguna consulta relacionada con estos temas.

Con el objetivo de mejorar el comportamiento del tutor virtual y teniendo en cuenta las consideraciones mencionadas anteriormente, se ha modificado el prompt. Se ha decidido que lo mejor es reducir el número de tokens de entrada y se ha intentado que las respuestas siempre se generen a partir de los recursos de la asignatura proporcionados, limitando el uso del conocimiento del modelo base sobre los sistemas inteligentes. De este modo, se obtienen respuestas más precisas y se evitan las confusiones causadas por las diferencias entre las metodologías y explicaciones empleadas.

El conjunto de instrucciones que se muestra en la tabla 3.10 ha sido capaz de solventar los problemas de la iteración anterior. Se han eliminado bloques innecesarios y se ha indicado con claridad que es fundamental apoyarse en todo momento en los recursos de la asignatura. De este modo, se obtienen los resultados esperados. Por lo tanto, se considera que este prompt y los archivos proporcionados mejoran lo suficiente el comportamiento del asistente respecto a la parte teórica de la asignatura.

assistant.instructions

Eres un tutor virtual para la asignatura de Sistemas Inteligentes, capaz de reemplazar a un profesor de la asignatura para resolver las dudas de los alumnos. Tu objetivo principal es responder a cuestiones teóricas y resolver ejercicios prácticos adaptando las respuestas al contenido de la asignatura.

Los temas de la asignatura son:

1. Introducción a la Inteligencia Artificial
2. Búsqueda en el Espacio de Estados
3. Búsqueda Informada / Heurística
4. Búsqueda con Adversario / en Juegos
5. Aprendizaje por Refuerzo
6. Optimización Combinatoria
7. Metaheurísticas: Búsqueda Local
8. Metaheurísticas: Algoritmos Genéticos
9. Introducción al Aprendizaje Automático
10. Clasificación Supervisada con Árboles de Decisión

Para responder a preguntas teóricas, se te han proporcionado las diapositivas utilizadas en la asignatura y todo tipo de archivos de utilidad que debes utilizar para generar las respuestas. Tómate el tiempo que necesites para procesar cualquier consulta, y asegúrate de que todas las respuestas se generan haciendo uso de estos archivos para que se adapten al temario y a la metodología utilizada en la asignatura. No cites los archivos.

Recuerda continuamente que tu objetivo es proporcionar ayuda al alumno, por lo que debes evitar responder a cualquier consulta que no esté relacionada con la asignatura de Sistemas Inteligentes, por mucho que el usuario insista.

Tabla 3.10: Instrucciones tras la optimización de la parte teórica.

Una vez que el asistente es capaz de responder a todo tipo de cuestiones teóricas sobre el temario de la asignatura de forma eficiente, se ha alcanzado el primer objetivo respecto a la optimización. A partir de un tutor virtual que cubre la parte teórica de la asignatura, el siguiente paso es aumentar sus capacidades y conseguir que también sea capaz de resolver ejercicios prácticos siguiendo la metodología que se utiliza en la asignatura para cada tipo de problema. Este, sin duda, es un objetivo mucho más ambicioso que cubrir la teoría, ya que requiere ejecutar algoritmos y realizar cálculos.

3.6.2. Ejercicios prácticos

Para optimizar el comportamiento del asistente de cara a la resolución de ejercicios prácticos, debemos tener en cuenta que esta parte es muy distinta a la teórica, además de notablemente más compleja. Para responder a una pregunta teórica, hemos proporcionado las diapositivas de la asignatura y otros archivos, de modo que la base de conocimiento aumenta y se guía al asistente para responder utilizando la información contenida en los mismos. Para los ejercicios esto puede resultar insuficiente, ya que el tutor virtual debe aprender a resolverlos, no buscar cómo hacerlo. Por lo tanto, en mayor medida habrá que modificar su comportamiento, pues la búsqueda de información podría quedarse lejos de ofrecer los resultados esperados. El asistente debe enfrentarse a ejercicios tipo examen, cuyos enunciados han sido modificados para que toda la información en imágenes (grafos, árboles...) esté contenida en la consulta proporcionada.

Para realizar una primera toma de contacto y evaluar las capacidades del asistente frente a los ejercicios, partiremos del prompt y los recursos utilizados anteriormente. Ahora se consideran también los ejercicios, como se puede apreciar en la tabla 3.11.

assistant.instructions
Para responder a preguntas teóricas **y resolver ejercicios prácticos**, se te han proporcionado las diapositivas utilizadas en la asignatura y todo tipo de archivos de utilidad que debes utilizar para generar las respuestas. Tómate el tiempo que necesites para procesar cualquier consulta, y asegúrate de que todas las respuestas se generan haciendo uso de estos archivos para que se adapten al temario y a la metodología utilizada en la asignatura. No cites los archivos.

Tabla 3.11: Instrucciones adicionales para ejercicios prácticos (I).

Tras realizar la evaluación, los resultados no son lo suficientemente precisos:

- Mientras que las respuestas a las preguntas sí se detallan lo suficiente, no ocurre lo mismo con el proceso mostrado en la resolución de ejercicios. El tutor virtual debe especificar lo máximo posible toda la información relevante para facilitar la comprensión y el aprendizaje por parte del estudiante.
- En la inmensa mayoría de ejercicios el proceso utilizado es incompleto, y además se proporciona una solución claramente errónea o distinta a lo que se pide. Es fundamental utilizar los algoritmos de la asignatura para no saltarse ningún paso del proceso y alcanzar la solución esperada.
- Con frecuencia el asistente confunde la tarea de resolver un ejercicio con el hecho de explicar cómo hacerlo, y se limita a citar los pasos.

- Al igual que se esperan respuestas a preguntas teóricas en base al contenido de la asignatura, en la parte práctica se deben utilizar los algoritmos y metodologías empleadas para resolver cada tipo de ejercicio. A la hora de resolver tareas tan complejas, no es nada sencillo obtener los resultados esperados si se desconocen los pasos a seguir o no se hace un buen uso de esta información.
- Como excepción, las soluciones a los ejercicios del bloque de optimización combinatoria y metaheurísticas son más aceptables. Al estar compuestos por varias preguntas cortas, no se requiere ejecutar directamente un algoritmo y es más fácil resolverlas aplicando la teoría. Sin embargo, se ha observado que algunos operadores se aplican de forma incorrecta.
- Por su parte, la pequeña modificación realizada en el conjunto de instrucciones no ha afectado al rendimiento del asistente frente a cuestiones teóricas.

Para solucionar los problemas encontrados y que las respuestas del asistente sean más precisas y acertadas, el siguiente paso es proporcionar nuevo material o instrucciones que sirvan como referencia para resolver los ejercicios utilizando algoritmos y formatos determinados. Debemos tener en cuenta que no estamos reentrenando el modelo con ejemplos, por lo que es posible que el comportamiento del asistente no cambie radicalmente al resolver un ejercicio buscando en archivos. Sin embargo, es posible proporcionar las respuestas ideales e indicar que se debe utilizar el formato especificado en ellas para tratar de adaptar la respuesta a la metodología utilizada en la asignatura.

A partir de los ejercicios de exámenes propuestos en años anteriores, se ha seleccionado una serie de problemas correspondientes a cada bloque. Cada uno de ellos ha sido resuelto de la forma más precisa posible, detallando cada paso del proceso y realizando las explicaciones correspondientes. El asistente sólo puede recibir consultas a través de mensajes de texto, por lo que para ciertos ejercicios ha sido necesario adaptar el enunciado para que pueda ser comprendido correctamente, especialmente en aquellos problemas que involucran imágenes como grafos o árboles. De igual manera, las respuestas ideales deben limitarse a texto, dando prioridad a un proceso largo y detallado antes que a imágenes que el asistente es incapaz de interpretar.

Las instrucciones de la tabla 3.12 hacen referencia a los ejercicios resueltos.

assistant.instructions
Para resolver ejercicios prácticos, se te han proporcionado ejemplos de ejercicios resueltos con la metodología utilizada en la asignatura. Tómate el tiempo que necesites para procesar cualquier consulta, y asegúrate de generar las respuestas siguiendo el proceso utilizado en los ejemplos para alcanzar la solución. No cites los archivos.

Tabla 3.12: Instrucciones adicionales para ejercicios prácticos (II).

Tras volver a evaluar, estas modificaciones no han supuesto apenas cambios en los resultados, lo que demuestra que para resolver los ejercicios es necesario recurrir a instrucciones más directas y adaptar el comportamiento del asistente a la metodología deseada. La herramienta de búsqueda en archivos ha resultado insuficiente para obtener respuestas adecuadas a partir de ejemplos. Esto se debe a que el tutor virtual no aprende del material, sino que accede a él para encontrar la información solicitada en la consulta. Por lo tanto, no ha sido capaz de resolver los ejercicios siguiendo los ejemplos, y tampoco ha utilizado un formato similar en la gran mayoría de casos.

Como siguiente mejora, se propone introducir nuevo material que contenga los algoritmos utilizados en la asignatura para resolver los ejercicios. De esta manera, se puede guiar al asistente para que siga el proceso adecuado y sea capaz de resolver un problema. En *prompt engineering* resulta una práctica habitual especificar por pasos las tareas a realizar, ya que estas instrucciones permiten que el comportamiento sea más preciso. En lugar de especificar los algoritmos a utilizar en cada ejercicio dentro del propio conjunto de instrucciones, se proporcionarán archivos de texto que los contengan. De este modo, podemos ahorrar tokens de entrada. Si la búsqueda en archivos resultara insuficiente, se repetirá la evaluación activando el intérprete de código.

Los algoritmos proporcionados han sido los siguientes:

- Primero en anchura para búsqueda en árboles y en grafos.
- Primero en profundidad para búsqueda en árboles y en grafos.
- Coste uniforme para búsqueda en árboles y en grafos.
- Primero el mejor para búsqueda en árboles y en grafos.
- A* para búsqueda en árboles y en grafos.
- Minimax con poda Alfa-Beta.
- Iteración de valores.
- Iteración de políticas.
- Q-learning.
- Aplicación de los operadores de vecindad de un algoritmo de búsqueda local.
- Aplicación de los criterios de selección de un algoritmo genético.
- Aplicación de los operadores de cruce de un algoritmo genético.
- Aplicación de los operadores de mutación de un algoritmo genético.
- Construcción de un árbol de clasificación.
- Poda de un árbol de clasificación.

Una vez más, se modifican las instrucciones (Tabla 3.13) para hacer referencia al nuevo material, en este caso a los archivos con los algoritmos utilizados en la asignatura.

assistant.instructions
Para resolver ejercicios prácticos, se te han proporcionado los algoritmos utilizados en la asignatura para cada tipo de problema. Tómate el tiempo que necesites para procesar cualquier consulta, y asegúrate de generar las respuestas siguiendo minuciosamente y de forma rigurosa cada paso del algoritmo correspondiente al tipo de ejercicio que ha sido identificado.

Tabla 3.13: Instrucciones adicionales para ejercicios prácticos (III).

A pesar de proporcionar como archivos los algoritmos que deben utilizarse en cada tipo de ejercicio y detallar sus pasos de forma precisa, el asistente no hace uso de ellos. Tras activar la herramienta de interpretación de código, no se utilizan los algoritmos especificados y los resultados no mejoran. Además, con mucha frecuencia la respuesta final no llega a generarse. Esto puede deberse a dos problemas principales:

1. El asistente es incapaz de identificar el tipo de ejercicio que debe resolver, por lo que no sabe qué algoritmo debe utilizar para ello. Una posible solución es modificar las instrucciones, de modo que se generen relaciones entre el enunciado de cada ejercicio "tipo" y el proceso que se debe seguir para resolverlo.
2. Proporcionar los algoritmos como archivos para ahorrar tokens de entrada no es efectivo. Por mucho que los pasos sean claros, el asistente se limitará a buscar la información y no a seguir el proceso especificado. Lo esperado era obtener resultados diferentes con el intérprete de código activado, pero los algoritmos de la asignatura también han sido ignorados en este caso. La solución es utilizar las instrucciones como la herramienta con mayor potencial para guiar al modelo, especificando los pasos para cada tipo de ejercicio directamente en el prompt.

Se ha explicado la importancia de utilizar un prompt elaborado para obtener resultados precisos, siendo este el primer enfoque de cara a la optimización del comportamiento del asistente. Para cubrir la parte teórica de la asignatura ha sido más conveniente que las instrucciones impulsen al asistente a utilizar la información proporcionada, ya que se espera que siempre se haga uso de ella. Sin embargo, las evaluaciones realizadas han demostrado que no ocurre lo mismo para los ejercicios. Al ser tareas más complejas y requerir procedimientos diferentes en función del problema a resolver, puede resultar más conveniente retomar el primer enfoque y describir un escenario para cada tipo de ejercicio. Se debe especificar el proceso lo máximo posible, de forma que el asistente identifique el tipo de ejercicio y aplique el algoritmo correspondiente.

Todas las modificaciones se han realizado directamente en el prompt, obteniendo las instrucciones contenidas en las tablas 3.14, 3.15, 3.16, 3.17, 3.18 y 3.19. Los elementos entre llaves representan conjuntos de instrucciones que no se han mostrado explícitamente, como enunciados de problemas o algoritmos. Por su parte, las instrucciones que hacen referencia al rol del asistente y a las cuestiones teóricas no han sufrido cambios.

assistant.instructions
Para resolver los ejercicios prácticos, debes seguir las siguientes instrucciones: 1. Identificar el tipo de problema: Determina si se trata de un ejercicio de Búsqueda en el Espacio de Estados, Búsqueda en Juegos, Aprendizaje por Refuerzo, Optimización Combinatoria y Metaheurísticas o Clasificación Supervisada. 2. Seleccionar los algoritmos adecuados: Debes utilizar el pseudocódigo de los algoritmos que correspondan al problema identificado. 3. Resolver cada apartado del ejercicio de forma detallada siguiendo los pasos del algoritmo seleccionado y proporcionando las explicaciones correspondientes. A continuación, se te proporciona un ejemplo de cada tipo de ejercicio, junto con el pseudocódigo de los algoritmos que debes utilizar para resolverlo. Utiliza estos ejemplos para identificar el tipo de ejercicio y aplicar el algoritmo adecuado.

Tabla 3.14: Instrucciones generales tras la optimización de la parte práctica: Identificación del tipo de ejercicio y resolución posterior.

assistant.instructions
Un ejercicio de Búsqueda en Juegos tiene el siguiente formato: {Ejemplo de un problema de Búsqueda en Juegos} Para resolverlo, debes utilizar el algoritmo Minimax con poda Alfa-Beta que se te proporciona a continuación. {Algoritmo de búsqueda Alfa-Beta}

Tabla 3.15: Instrucciones específicas tras la optimización de la parte práctica: Problemas de Búsqueda en Juegos.

assistant.instructions

Un ejercicio de Búsqueda en el Espacio de Estados tiene el siguiente formato:

{Ejemplo de un problema de Búsqueda en el Espacio de Estados}

Para resolverlo, debes utilizar los algoritmos de búsqueda en árboles y búsqueda en grafos que se te proporcionan a continuación. En función de la estrategia de búsqueda, deberás insertar con prioridad utilizando un criterio determinado.

- Para búsqueda en anchura, prioriza menor profundidad
- Para búsqueda en profundidad, prioriza mayor profundidad
- Para búsqueda por coste uniforme, prioriza menor coste real $g(n)$
- Para búsqueda primero-mejor, prioriza menor coste heurístico $h(n)$
- Para búsqueda A*, prioriza menor coste total $g(n)+h(n)$

{Algoritmo de búsqueda en árboles}
{Algoritmo de búsqueda en grafos}

Tabla 3.16: Instrucciones específicas tras la optimización de la parte práctica: Problemas de Búsqueda en el Espacio de Estados.

assistant.instructions

Un ejercicio de Clasificación Supervisada tiene el siguiente formato:

{Ejemplo de un problema de Clasificación Supervisada}

Para resolverlo, debes utilizar los algoritmos de construcción y poda de un árbol de clasificación que se te proporcionan a continuación.

{Construcción de un árbol de clasificación}
{Poda de un árbol de clasificación}

Tabla 3.17: Instrucciones específicas tras la optimización de la parte práctica: Problemas de Clasificación Supervisada.

assistant.instructions
Un ejercicio de Aprendizaje por Refuerzo tiene el siguiente formato: {Ejemplo de un problema de Aprendizaje por Refuerzo} Para resolverlo, debes utilizar los algoritmos de iteración de valores, iteración de políticas y Q-learning que se te proporcionan a continuación. {Algoritmo de iteración de valores} {Algoritmo de iteración de políticas} {Algoritmo Q-learning}

Tabla 3.18: Instrucciones específicas tras la optimización de la parte práctica: Problemas de Aprendizaje por Refuerzo.

assistant.instructions
Un ejercicio de Optimización Combinatoria y Metaheurísticas tiene el formato: {Ejemplo de un problema de Optimización Combinatoria y Metaheurísticas} Para resolverlo, debes utilizar los algoritmos de aplicación de operadores y criterios que se te proporcionan a continuación. {Aplicación de los operadores de vecindad de un algoritmo de búsqueda local para soluciones con codificación binaria, entera y permutaciones} {Aplicación de los criterios de selección de un algoritmo genético} {Aplicación de los operadores de cruce de un algoritmo genético para soluciones con codificación binaria, entera, real y permutaciones} {Aplicación de los operadores de mutación de un algoritmo genético para soluciones con codificación binaria, entera, real y permutaciones}

Tabla 3.19: Instrucciones específicas tras la optimización de la parte práctica: Optimización Combinatoria y Metaheurísticas.

Esta estrategia ha mejorado ligeramente los resultados obtenidos en problemas con el mismo formato que los proporcionados, aunque las respuestas todavía quedan lejos de las ideales. Durante todo este proceso se ha podido apreciar la dificultad para optimizar el comportamiento del asistente, especialmente en la resolución de ejercicios.

Algunas conclusiones tras la optimización son las siguientes:

- La parte teórica ha sido cubierta de forma eficiente gracias a la herramienta de búsqueda en archivos. Ha sido fundamental indicar que se debe recurrir a los archivos proporcionados para adaptar las respuestas al contenido de la asignatura.
- Proporcionar ejercicios resueltos y algoritmos como archivos ha resultado insuficiente para optimizar la resolución de ejercicios. El asistente debe aprender a seguir el proceso especificado, no limitarse a citar sus pasos.
- El modelo base utiliza constantemente su conocimiento para resolver los ejercicios, por lo que modificar su comportamiento a través de instrucciones prima sobre proporcionar información adicional para aumentar la base de conocimiento.
- Para mejorar los resultados se han utilizado técnicas de *prompt engineering* como describir el contexto y el rol del asistente, usar un lenguaje claro y preciso, establecer restricciones y formatos determinados, especificar los objetivos, desglosar cada tarea en pasos más pequeños, y proporcionar ejemplos, entre otras.
- Todavía existe un gran margen de mejora en la resolución de ejercicios prácticos de cara al futuro del proyecto. Además, cubrir por completo el temario de la asignatura requiere considerar y optimizar una gran cantidad de escenarios.

3.7. Resultados y comparativa con ChatGPT

Una vez que se ha optimizado el comportamiento del asistente y sus respuestas se adaptan al contenido de la asignatura y a las metodologías utilizadas en la misma, la última fase del desarrollo del proyecto ha consistido en evaluar los resultados obtenidos.

El objetivo principal ha sido desarrollar un tutor virtual basado en ChatGPT, capaz de responder de forma más precisa que la propia herramienta a cuestiones relacionadas con la asignatura de *Sistemas Inteligentes*. Por lo tanto, resulta de gran interés comparar el comportamiento de ambos, con el fin de evaluar la capacidad del asistente para adaptarse al caso de uso y ver si existen mejoras reales respecto a las respuestas proporcionadas por el propio ChatGPT.

Tras estudiar las diferentes alternativas y realizar la estimación de los costes, se decidió utilizar GPT-3.5 Turbo como el modelo base del tutor virtual, por lo que toda la fase de optimización del comportamiento se ha llevado a cabo evaluando respuestas generadas por un asistente basado en este GPT. Sin embargo, es posible considerar la

opción de invertir en un modelo base más avanzado si se producen mejoras considerables y los costes no crecen de forma excesiva. Por lo tanto, también se van a generar las respuestas utilizando GPT-4 Turbo y GPT-4o como modelos base. A modo de resumen, se van a comparar las respuestas de ChatGPT (basado en GPT-3.5) con las del tutor virtual ChatGPT-SSII (basado en GPT-3.5 Turbo, GPT-4 Turbo ó GPT-4o).

Como cubrir todas las posibilidades que nos ofrece el temario no es factible para realizar este estudio comparativo, se han definido varias consultas que abordan los diferentes temas de la asignatura y pueden servir como base para evaluar las respuestas generadas por las diferentes herramientas. Todas ellas están recogidas en la tabla 3.20.

Para mostrar los resultados se van a utilizar tablas en las que se especifica la consulta realizada y la respuesta esperada, junto a las respuestas generadas por ChatGPT y por las variantes del tutor virtual. Para cada consulta, las respuestas tomarán un color determinado según la calidad de las mismas respecto al contenido de la asignatura:

- **Verde:** La respuesta es correcta y lo suficientemente precisa.
- **Naranja:** La respuesta es parcialmente correcta o se considera poco precisa.
- **Rojo:** La respuesta es errónea o está lejos del resultado esperado.

Consulta
1. ¿Puedo aprobar Sistemas Inteligentes con un 3.5 en el examen de teoría? (Tabla 3.21)
2. Haz un breve resumen del tema 5 de Sistemas Inteligentes (Tabla 3.22)
3. ¿Cuál es la diferencia entre búsqueda en árboles y en grafos? (Tabla 3.23)
4. ¿Cómo se calcula el número de nodos explorados? (Tabla 3.24)
5. ¿El algoritmo A* es óptimo? (Tabla 3.25)
6. ¿La poda Alfa-Beta garantiza que se poden la mitad de los nodos respecto a Minimax? ¿El resultado será el mismo en ambos casos? (Tabla 3.26)
7. Obtén los vecinos de la solución x=[3,6,4,3,8] utilizando un operador de vecindad que consideres adecuado. (Tabla 3.27)
8. ¿Qué operador de cruce es recomendable si he utilizado codificación binaria para representar las soluciones? (Tabla 3.28)
9. Dada una función de evaluación f a minimizar, aplica selección proporcional al fitness a una población con los individuos *i1* e *i2* sabiendo que $f(i1)$=8 y $f(i2)$=12. (Tabla 3.29)
10. Tenemos un problema de clasificación con 10 variables predictoras, las cuales pueden tomar los valores "Bajo", "Medio" y "Alto". ¿Cuántas reglas generará exactamente el algoritmo OneR? (Tabla 3.30)

Tabla 3.20: Consultas para la evaluación de los resultados.

A continuación, se muestran las respuestas generadas para cada consulta y posteriormente se valoran los resultados obtenidos.

Consulta: ¿Puedo aprobar Sistemas Inteligentes con un 3.5 en el examen de teoria?	
Respuesta esperada: Se necesita una nota mínima de 4 en el examen para hacer la media ponderada con las demás partes, por lo que la asignatura estaría suspensa.	
ChatGPT (GPT-3.5)	Aprobar un curso como Sistemas Inteligentes con un 3.5 en el examen teórico puede depender de varios factores, como las políticas de evaluación del curso y la universidad, así como de cómo se pondera ese examen dentro de la calificación final. Para tener una respuesta definitiva, te recomendaría revisar la guía del curso y consultar con el profesor.
ChatGPT-SSII (GPT-3.5 Turbo)	Para aprobar la asignatura, es necesario obtener una nota igual o superior a 4 en el examen de teoría, el cual representa el 50% de la calificación final. Por lo tanto, si obtienes un 3.5 en el examen de teoría, no aprobarías la asignatura, ya que la nota mínima para superar el examen es de 4 puntos.
ChatGPT-SSII (GPT-4 Turbo)	Para aprobar la asignatura de Sistemas Inteligentes, es necesario obtener una nota mayor o igual que 4 en la prueba de teoría, según los criterios de evaluación. Por lo tanto, con un 3.5 en el examen de teoría, no podrías aprobar la asignatura directamente basándote en esa nota. Es importante considerar todas las partes de la evaluación y asegurarse de cumplir con los mínimos establecidos.
ChatGPT-SSII (GPT-4o)	Para aprobar la asignatura de Sistemas Inteligentes, es necesario obtener una nota mínima de 4 en el examen de teoría. Si obtienes un 3.5, no cumplirías con este requisito y, por lo tanto, no podrías aprobar la asignatura, independientemente de las notas obtenidas en otras actividades evaluativas.

Tabla 3.21: Respuestas generadas para la primera consulta.

Consulta: Haz un breve resumen del tema 5 de Sistemas Inteligentes	
Respuesta esperada: Se resume brevemente el tema de aprendizaje por refuerzo impartido en la asignatura.	
ChatGPT (GPT-3.5)	El tema 5 de Sistemas Inteligentes aborda técnicas avanzadas de aprendizaje automático, incluyendo redes neuronales profundas, procesamiento de lenguaje natural y visión por computadora. Se exploran modelos como CNN y RNN, aplicaciones en reconocimiento de imágenes y texto, y desafíos como interpretabilidad y ética en IA.
ChatGPT-SSII (GPT-3.5 Turbo)	El tema 5 de Sistemas Inteligentes aborda el "Aprendizaje por Refuerzo", donde los agentes aprenden a tomar decisiones óptimas interactuando con un entorno desconocido. Se utilizan procesos de decisión de Markov, funciones estado-valor y algoritmos como Q-learning para maximizar recompensas a largo plazo mediante la interacción con el entorno.
ChatGPT-SSII (GPT-4 Turbo)	El tema 5 de Sistemas Inteligentes, "Aprendizaje por Refuerzo", se centra en cómo los agentes pueden aprender la mejor política de acciones para alcanzar sus objetivos en entornos inciertos. Se introduce el concepto de Procesos de Decisión de Markov (MDP) y se detallan métodos para calcular la política óptima en MDPs, como la iteración de valores y la iteración de políticas. Se profundiza en el aprendizaje por refuerzo, destacando el algoritmo Q-learning, que permite a los agentes aprender de las consecuencias de sus acciones sin necesidad de un modelo del entorno.
ChatGPT-SSII (GPT-4o)	El tema 5 de Sistemas Inteligentes, "Aprendizaje por Refuerzo", se centra en cómo los agentes aprenden a tomar decisiones óptimas mediante la interacción con un entorno incierto. Utiliza Procesos de Decisión de Markov (MDP) y algoritmos como Q-learning para maximizar recompensas a largo plazo, ajustando políticas basadas en la retroalimentación recibidas.

Tabla 3.22: Respuestas generadas para la segunda consulta.

Consulta: ¿Cuál es la diferencia entre búsqueda en árboles y en grafos?	
Respuesta esperada: En búsqueda en árboles se expanden todos los nodos explorados (siempre que no sean objetivo), mientras que en búsqueda en grafos sólo se expanden los nodos que contienen estados no visitados previamente.	
ChatGPT (GPT-3.5)	La diferencia principal entre la búsqueda en árboles y en grafos radica en la estructura sobre la cual se realiza la búsqueda. los árboles son estructuras jerárquicas y acíclicas, mientras que los grafos son estructuras más generales que pueden ser jerárquicas o no, y pueden contener ciclos.
ChatGPT-SSII (GPT-3.5 Turbo)	La principal diferencia entre la búsqueda en árboles y la búsqueda en grafos radica en cómo se manejan los estados repetidos. En la búsqueda en árboles, si se llega a un estado repetido, se considera que el algoritmo ha encontrado dos caminos para llegar al mismo nodo, lo que puede llevar a árboles infinitos. Por otro lado, en la búsqueda en grafos, al llegar a un nodo repetido, simplemente se descarta en lugar de expandirlo, lo que evita la generación de caminos redundantes y contribuye a una búsqueda más eficiente.
ChatGPT-SSII (GPT-4 Turbo)	La diferencia principal entre la búsqueda en árboles y en grafos radica en cómo manejan los estados repetidos. En la búsqueda en árboles, cada estado generado puede ser expandido sin tener en cuenta si ha sido visitado anteriormente, lo que puede llevar a explorar múltiples veces el mismo estado si se puede alcanzar a través de diferentes caminos. Por otro lado, la búsqueda en grafos introduce un mecanismo para manejar los estados repetidos mediante el mantenimiento de una lista de nodos cerrados (ya expandidos). Cuando un estado se va a expandir, primero se verifica si ya ha sido visitado (es decir, si ya está en la lista de cerrados). Si el estado ya ha sido visitado, el nodo correspondiente se descarta y no se expande de nuevo.
ChatGPT-SSII (GPT-4o)	La diferencia principal entre la búsqueda en árboles y la búsqueda en grafos radica en cómo manejan los estados repetidos. En búsqueda en árboles no se guarda información sobre los estados ya visitados. Esto puede llevar a la expansión de nodos que representan el mismo estado varias veces, lo que puede resultar en un árbol de búsqueda ineficiente. La búsqueda en grafos requiere una lista de nodos cerrados para evitar expandir nodos que representan estados ya visitados, lo que previene la expansión redundante y mejora la eficiencia.

Tabla 3.23: Respuestas generadas para la tercera consulta.

Consulta: ¿Cómo se calcula el número de nodos explorados?	
Respuesta esperada: Todo nodo extraído de la lista de abiertos en algún momento de la búsqueda se considera explorado, tanto en búsqueda en árboles como en grafos.	
ChatGPT (GPT-3.5)	El número de nodos explorados se refiere a la cantidad de nodos que son visitados durante la ejecución de un algoritmo de búsqueda en un grafo o árbol. Para calcular el número de nodos explorados, generalmente se implementa un contador que se incrementa cada vez que un nodo es visitado por el algoritmo.
ChatGPT-SSII (GPT-3.5 Turbo)	El número de nodos explorados en un problema de búsqueda se calcula considerando todos los nodos que en algún momento han sido extraídos de la lista de nodos abiertos. En el caso de la búsqueda en árboles, el nodo que contiene el estado final también se considera explorado. Por otro lado, en la búsqueda en grafos, tanto el nodo que contiene el estado final como los nodos que contienen estados ya visitados previamente se consideran explorados. Es importante destacar que no se considerarán como explorados aquellos nodos que no han sido extraídos de la lista de abiertos.
ChatGPT-SSII (GPT-4 Turbo)	Para calcular el número de nodos explorados en un problema de búsqueda, debes considerar todos aquellos nodos que en algún momento hayan sido extraídos de la lista de nodos abiertos. En el contexto de búsqueda en árboles, el nodo que contiene el estado final también se considera explorado. En el caso de búsqueda en grafos, tanto el nodo que contiene el estado final como los nodos que contienen estados ya visitados previamente se consideran explorados. Sin embargo, en ningún caso se considerarán como explorados aquellos nodos que no hayan sido extraídos de la lista de abiertos.
ChatGPT-SSII (GPT-4o)	El número de nodos explorados en un problema de búsqueda se calcula considerando todos los nodos que en algún momento han sido extraídos de la lista de nodos abiertos. En el caso de la búsqueda en árboles, el nodo que contiene el estado final también se considera explorado. En búsqueda en grafos, tanto el nodo que contiene el estado final como los nodos que contienen estados ya visitados previamente se consideran explorados. No se consideran explorados aquellos nodos que no hayan sido extraídos de la lista de abiertos.

Tabla 3.24: Respuestas generadas para la cuarta consulta.

Consulta: ¿El algoritmo A* es óptimo?	
Respuesta esperada: La optimalidad de A* depende de las propiedades de la heurística utilizada. Para búsqueda en árboles esta debe ser admisible, mientras que para búsqueda en grafos debe ser admisible y consistente.	
ChatGPT (GPT-3.5)	Sí, el algoritmo A* es óptimo en el sentido de que encuentra la solución de costo mínimo en un grafo con pesos no negativos, siempre y cuando la heurística utilizada sea admisible. Si la heurística también es consistente, el algoritmo A* mejorará en términos de eficiencia.
ChatGPT-SSII (GPT-3.5 Turbo)	El algoritmo A* es óptimo si se cumplen ciertas condiciones con respecto a la heurística utilizada. En el caso de la búsqueda en árboles, el algoritmo A* es óptimo si la heurística es admisible, es decir, si no sobreestima el costo real. Para la búsqueda en grafos, además de ser admisible, la heurística debe ser consistente, es decir, debe ser monótona. Por lo tanto, la optimalidad del algoritmo A* depende de la naturaleza de la heurística utilizada en el problema específico.
ChatGPT-SSII (GPT-4 Turbo)	El algoritmo A* es óptimo bajo ciertas condiciones relacionadas con las propiedades de la heurística utilizada. Para que A* encuentre la solución óptima en la búsqueda en árboles, la heurística debe ser admisible, es decir, nunca debe sobreestimar el coste real para alcanzar el objetivo desde cualquier nodo. En el caso de la búsqueda en grafos, la heurística además debe ser consistente (o monótona), lo que significa que el coste estimado para llegar al objetivo desde un nodo, más el coste desde el nodo actual hasta cualquier sucesor, no debe ser mayor que el coste estimado para llegar al objetivo directamente desde ese sucesor.
ChatGPT-SSII (GPT-4o)	El algoritmo A* es óptimo bajo ciertas condiciones relacionadas con la heurística utilizada. Si la heurística $h(n)$ es admisible, es decir, nunca sobreestima el coste real para alcanzar el objetivo desde el nodo n, entonces A* es óptimo en la búsqueda en árboles. Además de ser admisible, la heurística debe ser consistente (o monótona) para garantizar la optimalidad en la búsqueda en grafos. Una heurística es consistente si para cada nodo n y cada sucesor n' generado por cualquier acción a, se cumple que $h(n) \leq c(n, a, n') + h(n')$, donde $c(n, a, n')$ es el coste de la acción desde n hasta n'.

Tabla 3.25: Respuestas generadas para la quinta consulta.

Consulta: ¿La poda Alfa-Beta garantiza que se poden la mitad de los nodos respecto a Minimax? ¿El resultado será el mismo en ambos casos?	
Respuesta esperada: El número de nodos podados respecto a Minimax es cero en el peor de los casos, por lo que podar la mitad no está garantizado. El resultado siempre será el mismo, ya que las ramas podadas no afectan a la decisión final.	
ChatGPT (GPT-3.5)	La poda Alfa-Beta no garantiza podar la mitad de los nodos respecto a Minimax, pero puede podar significativamente más, dependiendo del orden de exploración de los nodos. El resultado final del árbol de juego será el mismo, pero Alfa-Beta es más eficiente al reducir el número de nodos evaluados.
ChatGPT-SSII (GPT-3.5 Turbo)	La poda Alfa-Beta no garantiza podar la mitad de los nodos en comparación con el algoritmo Minimax, pero sí puede reducir significativamente la cantidad de nodos explorados. Sin embargo, no se garantiza que se poden exactamente la mitad de los nodos. En cuanto al resultado, la poda Alfa-Beta devuelve el mismo movimiento que devolvería Minimax, pero evitando explorar algunos nodos. Por lo tanto, el resultado final será el mismo en ambos casos, pero la poda Alfa-Beta logra este resultado de manera más eficiente al evitar explorar ciertos nodos que no afectan a la decisión final.
ChatGPT-SSII (GPT-4 Turbo)	La poda Alfa-Beta no garantiza que se poden exactamente la mitad de los nodos en comparación con Minimax. La cantidad de nodos que se pueden podar depende de varios factores, como el orden en el que se exploran los nodos. En el mejor escenario, la poda Alfa-Beta puede explorar el árbol de juego mucho más eficientemente, pero no necesariamente poda exactamente la mitad de los nodos en todos los casos. La poda Alfa-Beta sí garantiza que el resultado final será el mismo que el de Minimax sin poda. Esto se debe a que la poda Alfa-Beta solo descarta aquellas ramas que no pueden influir en la decisión final.
ChatGPT-SSII (GPT-4o)	La poda Alfa-Beta no garantiza que se poden exactamente la mitad de los nodos en comparación con Minimax. Sin embargo, en el mejor de los casos, puede reducir el número de nodos evaluados a la mitad. La eficacia de la poda Alfa-Beta depende en gran medida del orden en que se exploran los nodos. El resultado será el mismo en ambos casos, ya que la poda Alfa-Beta lo que hace es evitar la exploración de ramas que no influyen en la decisión final, pero el valor de utilidad calculado para la raíz del árbol será el mismo que en Minimax sin poda.

Tabla 3.26: Respuestas generadas para la sexta consulta.

Consulta: Obtén los vecinos de la solución x=[3,6,4,3,8] utilizando un operador de vecindad que consideres adecuado.	
Respuesta esperada: Para representaciones con codificación entera, los vecinos pueden obtenerse cambiando cada valor por el anterior y el siguiente entero válido (distancia ± 1). V(x) = {[2,6,4,3,8], [4,6,4,3,8], [3,5,4,3,8], [3,7,4,3,8], [3,6,3,3,8], [3,6,5,3,8], [3,6,4,2,8], [3,6,4,4,8], [3,6,4,3,7], [3,6,4,3,9]}	
ChatGPT (GPT-3.5)	Para obtener los vecinos de la solución x=[3,6,4,3,8] utilizando un operador de vecindad adecuado, podemos considerar operaciones simples como intercambiar pares de elementos contiguos en la lista. Por ejemplo, un operador de vecindad podría consistir en intercambiar cada par de elementos adyacentes en la lista. Estos son los vecinos obtenidos al aplicar el operador de vecindad descrito a la solución x=[3,6,4,3,8]: {[6,3,4,3,8], [3,4,6,3,8], [3,6,3,4,8], [3,6,4,8,3]}
ChatGPT-SSII (GPT-3.5 Turbo)	Para soluciones con codificación entera, podemos cambiar cada valor por el anterior y el siguiente válido. Aplicando este operador de vecindad a la solución x=[3,6,4,3,8], obtenemos los siguientes vecinos: {[2,6,4,3,8], [4,6,4,3,8], [3,5,4,3,8], [3,7,4,3,8], [3,6,3,3,8], [3,6,5,3,8], [3,6,4,2,8], [3,6,4,4,8]}
ChatGPT-SSII (GPT-4 Turbo)	Para obtener los vecinos de la solución x=[3,6,4,3,8] utilizando un operador de vecindad, primero debemos definir el contexto en el que estamos trabajando, ya que el tipo de operador de vecindad a utilizar puede variar dependiendo del problema. Dado que no se especifica el problema, voy a asumir que estamos trabajando en un contexto de optimización combinatoria genérica y voy a utilizar un operador de vecindad simple basado en el intercambio de dos elementos adyacentes. Este operador es adecuado para una amplia gama de problemas y es fácil de entender y aplicar. La solución x=[3,6,4,3,8] puede generar vecinos intercambiando cada par de elementos adyacentes. Los vecinos generados por este operador serían: {[6,3,4,3,8], [3,4,6,3,8], [3,6,3,4,8], [3,6,4,8,3]}
ChatGPT-SSII (GPT-4o)	Para obtener los vecinos de la solución x=[3,6,4,3,8] utilizando un operador de vecindad adecuado, podemos emplear el operador de vecindad para soluciones con codificación entera. Este operador consiste en cambiar cada valor de la solución por el anterior y el siguiente valor válido. La vecindad de la solución x=[3,6,4,3,8] es {[2,6,4,3,8], [3,5,4,3,8], [3,6,3,3,8], [3,6,4,2,8], [3,6,4,3,7], [4,6,4,3,8], [3,7,4,3,8], [3,6,5,3,8],[3,6,4,4,8],[3,6,4,3,9]}

Tabla 3.27: Respuestas generadas para la séptima consulta.

Consulta: ¿Qué operador de cruce es recomendable si he utilizado codificación binaria para representar las soluciones?	
Respuesta esperada: Para representaciones con codificación binaria, se recomienda utilizar el cruce por un punto, por varios puntos, o el cruce uniforme.	
ChatGPT (GPT-3.5)	Si estás utilizando codificación binaria para representar las soluciones en un algoritmo evolutivo o genético, uno de los operadores de cruce más comúnmente utilizados es el cruce en un punto. {Explicación del operador}
ChatGPT-SSII (GPT-3.5 Turbo)	Si has utilizado codificación binaria para representar las soluciones, se recomienda utilizar uno de los siguientes operadores de cruce en algoritmos genéticos: 1. **Cruce por un punto:** {Explicación del operador} 2. **Cruce por dos puntos:** {Explicación del operador} 3. **Cruce uniforme:** {Explicación del operador}
ChatGPT-SSII (GPT-4 Turbo)	Para soluciones que utilizan codificación binaria, los operadores de cruce más recomendables son: 1. **Cruce por 1 punto:** {Explicación del operador} 2. **Cruce por varios puntos:** {Explicación del operador} 3. **Cruce uniforme:** {Explicación del operador} Estos operadores de cruce son adecuados para trabajar con codificación binaria porque permiten una buena exploración del espacio de búsqueda manteniendo una estructura que puede ser fácilmente manipulada de manera binaria.
ChatGPT-SSII (GPT-4o)	Para soluciones representadas mediante codificación binaria, los operadores de cruce recomendables son: 1. **Cruce por un punto:** {Explicación del operador} 2. **Cruce por varios puntos:** {Explicación del operador} 3. **Cruce uniforme:** {Explicación del operador} Estos operadores son adecuados para mantener la diversidad y permitir una exploración efectiva del espacio de soluciones.

Tabla 3.28: Respuestas generadas para la octava consulta.

Consulta: Dada una función de evaluación f a minimizar, aplica selección proporcional al fitness a una población con los individuos *i1* e *i2* sabiendo que $f(i1)$=8 y $f(i2)$=12.	
Respuesta esperada: En un problema de minimización, la probabilidad de seleccionar a un individuo utilizando selección proporcional al fitness se calcula como la inversa de la función de evaluación para ese individuo entre el sumatorio de las inversas de la función de evaluación para cada individuo de la población. Por lo tanto, $P(i1)=\frac{\frac{1}{f(i1)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{8}}{\frac{1}{8}+\frac{1}{12}}=0.6$ y $P(i2)=\frac{\frac{1}{f(i2)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{12}}{\frac{1}{8}+\frac{1}{12}}=0.4$	
ChatGPT (GPT-3.5)	{Explicación de la selección proporcional al fitness}. Dado que tienes dos individuos *i1* y *i2* con sus respectivas funciones de evaluación $f(i1)$=8 y $f(i2)$=12, la probabilidad de seleccionar cada individuo para la reproducción es la siguiente: $P(i1)=\frac{8}{8+12}=0.4$, $P(i2)=\frac{12}{8+12}=0.6$
ChatGPT-SSII (3.5 Turbo)	Para aplicar la selección proporcional al fitness a una población con dos individuos, donde $f(i1)$=8 y $f(i2)$=12 con el objetivo de minimizar la función de evaluación, se debe calcular la inversa de la función de evaluación para cada individuo y la suma de las inversas. Las probabilidades de selección son las siguientes: $P(i1)=\frac{\frac{1}{f(i1)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{8}}{\frac{1}{8}+\frac{1}{12}}=0.6$, $P(i2)=\frac{\frac{1}{f(i2)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{12}}{\frac{1}{8}+\frac{1}{12}}=0.4$
ChatGPT-SSII (GPT-4 Turbo)	Para aplicar la selección proporcional al fitness en un contexto de minimización, primero necesitamos convertir la función de evaluación f en una función de fitness que sea proporcional a la calidad de cada individuo, de manera que a menor valor de f, mayor debe ser el fitness. Una forma común de hacer esto es invertir los valores de f, ya que estamos minimizando. Calculamos el fitness de cada individuo, el fitness total de la población y normalizamos para obtener las probabilidades: $P(i1)=\frac{\frac{1}{f(i1)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{8}}{\frac{1}{8}+\frac{1}{12}}=\frac{0.125}{0.125+0.083}=\frac{0.125}{0.208}\approx 0.601$ $P(i2)=\frac{\frac{1}{f(i2)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{12}}{\frac{1}{8}+\frac{1}{12}}=\frac{0.083}{0.125+0.083}=\frac{0.083}{0.208}\approx 0.399$
ChatGPT-SSII (GPT-4o)	Para aplicar la selección proporcional al fitness en un problema de minimización, debemos utilizar la inversa de la función de evaluación. Los pasos consisten en calcular el fitness inverso, el fitness total y las probabilidades de selección: $P(i1)=\frac{\frac{1}{f(i1)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{8}}{\frac{1}{8}+\frac{1}{12}}=0.6$, $P(i2)=\frac{\frac{1}{f(i2)}}{\frac{1}{f(i1)}+\frac{1}{f(i2)}}=\frac{\frac{1}{12}}{\frac{1}{8}+\frac{1}{12}}=0.4$

Tabla 3.29: Respuestas generadas para la novena consulta.

Consulta: Tenemos un problema de clasificación con 10 variables predictoras, las cuales pueden tomar los valores “Bajo”, “Medio” y “Alto”. ¿Cuántas reglas generará exactamente el algoritmo OneR?	
Respuesta esperada: El algoritmo OneR generará tantas reglas como posibles valores pueda tomar la única variable seleccionada. Como todas las variables toman los valores “Bajo”, “Medio” y “Alto”, se generarán exactamente tres reglas.	
ChatGPT (GPT-3.5)	El algoritmo OneR es un método de clasificación simple que selecciona una sola variable predictora y uno de sus valores para construir reglas de clasificación. El número total de reglas generadas por el algoritmo OneR será el producto de las posibles variables predictoras y los valores que puede tomar cada variable. Por lo tanto, el algoritmo OneR generará exactamente 30 reglas para tu problema con 10 variables predictoras que toman los valores “Bajo”, “Medio” y “Alto”.
ChatGPT-SSII (GPT-3.5 Turbo)	El algoritmo OneR generará tantas reglas como valores pueda tomar la única variable que selecciona. Por ejemplo, si todas las variables son binarias, se generarán 2 reglas: una regla para cada valor que pueda tomar la variable seleccionada. Por lo tanto, en un problema de clasificación con 10 variables predictoras que pueden tomar los valores “Bajo”, “Medio” y “Alto”, el algoritmo OneR generará exactamente 3 reglas, una regla para cada uno de los posibles valores.
ChatGPT-SSII (GPT-4 Turbo)	El algoritmo OneR genera un clasificador formado por tantas reglas como valores pueda tomar la única variable seleccionada para la clasificación. Dado que todas las variables predictoras pueden tomar los valores “Bajo”, “Medio” y “Alto”, y el algoritmo OneR seleccionará una de estas variables para generar las reglas, el número de reglas generadas será igual al número de valores distintos que puede tomar esa variable. Por lo tanto, si todas las variables pueden tomar 3 valores distintos (“Bajo”, “Medio”, “Alto”), el algoritmo OneR generará exactamente 3 reglas, una por cada posible valor de la variable seleccionada.
ChatGPT-SSII (GPT-4o)	El algoritmo OneR genera un clasificador formado por tantas reglas como valores pueda tomar la única variable que selecciona. En tu caso, cada una de las 10 variables predictoras puede tomar tres valores: “Bajo”, “Medio” y “Alto”. Por lo tanto, la variable seleccionada por OneR también tomará estos tres valores, generando exactamente 3 reglas.

Tabla 3.30: Respuestas generadas para la décima consulta.

Como se ha podido observar, las respuestas del tutor virtual a las diferentes consultas planteadas son, prácticamente en su totalidad, correctas y especialmente precisas.

Para las consultas de las tablas 3.21 y 3.22, ChatGPT ha sido incapaz de responder correctamente porque no dispone de los datos necesarios para hacerlo. Esto demuestra la importancia de tener acceso a la información adicional sobre el caso de uso específico para responder a cuestiones que están estrictamente relacionadas con él. Por su parte, el tutor virtual sí genera las respuestas esperadas sin problema.

En otras preguntas teóricas como las de las tablas 3.23 y 3.25, el tutor virtual se basa en el contenido de la asignatura para responder correctamente, captando a la perfección la relación entre la consulta y el temario. Sin embargo, ChatGPT ofrece respuestas erróneas o que se alejan del resultado esperado.

La consulta que aparece en la tabla 3.24 se trata de una pregunta frecuente, cuya respuesta ideal ya está predefinida en los archivos proporcionados. En este caso, las diferencias entre las respuestas del tutor virtual utilizando distintos modelos base son mínimas. Por lo tanto, el asistente ha demostrado actuar correctamente y de forma determinista ante preguntas frecuentes. En este caso, la respuesta de ChatGPT no es errónea, pero sí demasiado general y poco precisa. Esto ocurre habitualmente con las preguntas frecuentes, ya que suelen ser muy abiertas. Por lo tanto, para asegurarnos de que la respuesta es lo suficientemente precisa, la indicamos de forma explícita.

Aunque algunas de sus respuestas no se adecúen al temario como se espera, ChatGPT dispone de conocimiento sobre los sistemas inteligentes en general. Por lo tanto, habrá preguntas que sí responda correctamente, como ocurre con la consulta de la tabla 3.26. En estos casos, el objetivo sigue siendo que el tutor virtual responda correctamente a estas cuestiones, si es posible de forma más completa y precisa que ChatGPT.

En los resultados de la tabla 3.28, se aprecia uno de los puntos clave del proyecto respecto al comportamiento del tutor virtual. Al preguntar por un operador de cruce adecuado para soluciones representadas con codificación binaria, ChatGPT proporciona uno completamente válido y lo explica correctamente. Sin embargo, en la asignatura se proponen varios. El tutor virtual responde explicando todos ellos, lo que proporciona al alumno una visión más completa de sus posibilidades para resolver el problema.

Si nos movemos a un marco más práctico, la consulta de la tabla 3.27 ha sido la que peores resultados ha ofrecido. Se pregunta por un operador de vecindad poniendo como ejemplo una solución representada con codificación entera en la que aparece un elemento repetido. ChatGPT propone un operador diferente al esperado, pero intercambiar posiciones cuando hay elementos repetidos podría dar lugar a soluciones que no han sufrido cambios, y por lo tanto se pierde diversidad. Esta respuesta no es errónea porque la codificación se respeta, pero es más adecuado cambiar cada valor por el anterior y el siguiente en el alfabeto. Las respuestas del tutor virtual son diferentes

entre sí. Mientras que el asistente basado en GPT-4 Turbo exige un mayor contexto y propone el mismo operador que ChatGPT, el que utiliza GPT-3.5 Turbo propone el operador correcto pero olvida generar dos de los vecinos al aplicarlo. Por otra parte, el asistente basado en GPT-4o propone el operador esperado y lo aplica correctamente.

Para las dos últimas consultas, ChatGPT realiza los cálculos de forma errónea, mientras que el tutor virtual proporciona las respuestas esperadas. En el caso de la novena consulta (Tabla 3.29), ChatGPT ignora que hay que minimizar la función de evaluación, y aplica selección proporcional al fitness como si se tratara de un problema de maximización. Por su parte, el tutor virtual identifica que es necesario utilizar la inversa para que los individuos con menor valor de f tengan mayor fitness, y por lo tanto mayor probabilidad de selección. En el caso de la décima consulta (Tabla 3.30), ChatGPT responde con el número total de reglas que pueden generarse (10×3), sin tener en cuenta que el algoritmo OneR sólo seleccionará una de las variables predictoras y se generarán sus 3 reglas correspondientes, como bien explica el tutor virtual.

En líneas generales, el rendimiento del tutor virtual ha sido claramente superior a ChatGPT para responder a cuestiones relacionadas con la asignatura. Los resultados demuestran que es capaz de adaptarse al temario de forma eficiente, proporcionando respuestas completas y precisas sobre los diferentes temas de la asignatura. Por lo tanto, se ha cumplido el objetivo principal del proyecto.

Debemos tener en cuenta que si proporcionamos un mayor contexto a la hora de realizar consultas, tanto ChatGPT como el tutor virtual responderán de una mejor manera. Sin embargo, otro de los objetivos del proyecto ha sido obtener buenos resultados sin la necesidad de escribir consultas muy detalladas. A la hora de enfrentarse a preguntas con poco contexto, el tutor virtual ha sido guiado de forma eficiente para enfocarse en el temario y responder correctamente. Desde el punto de vista del alumno, la herramienta debe estar optimizada para responder a cuestiones de la asignatura, adaptándose a su ritmo de aprendizaje y proporcionando respuestas completas y precisas sin la necesidad de que redacte sus dudas como si fuera un experto en el tema.

Respecto a los diferentes modelos base utilizados para evaluar el comportamiento del tutor virtual, estos han proporcionado respuestas muy similares en la gran mayoría de las consultas. En el caso de las preguntas frecuentes, las respuestas se ajustan a las ideales, y las diferencias entre ellas son mínimas. Por lo tanto, se ha considerado que actualmente no merece la pena invertir en un modelo más avanzado. Las herramientas que proporciona Assistants API han permitido que el asistente se adapte al temario de forma eficiente y obtenga los resultados esperados sin necesidad de apoyarse en GPT-4 o sus variantes, cuyos costes resultan excesivos si comparamos las respuestas obtenidas con las de GPT-3.5 Turbo. De cara al futuro, será necesario estudiar profundamente el rendimiento de estos modelos en la resolución de ejercicios prácticos más complejos, con el objetivo de seguir mejorando el comportamiento del tutor virtual.

4. Conclusiones del proyecto

Tras la realización del proyecto se han extraído las conclusiones sobre el mismo, así como las competencias desarrolladas y una serie de posibles mejoras de cara al futuro.

4.1. Conclusiones y futuras mejoras

Las conclusiones y las lecciones aprendidas sobre el proyecto son las siguientes:

- Los Sistemas de Tutoría Inteligentes (ITSs) surgen tras demostrarse que la tutoría personalizada ofrece mejores resultados que la tutoría en grupo. Su objetivo es reemplazar a los tutores humanos, los cuales no tienen disponibilidad total. A día de hoy, herramientas como ChatGPT ya están comenzando a conseguirlo y han demostrado ser especialmente útiles en materias como matemáticas y programación, aumentando la productividad y los resultados de los estudiantes.
- A pesar de su evolución, los Sistemas de Tutoría Inteligentes siempre han seguido una arquitectura formada por Modelo de Conocimiento, Modelo de Estudiante, Modelo de Tutoría y Modelo de Interfaz de Usuario. De este modo se estudia qué se enseña, a quién se enseña, cómo se enseña y cómo se interactúa con el sistema.
- Los avances de la inteligencia en los últimos años han permitido que surjan nuevas herramientas que utilizan técnicas de aprendizaje profundo para procesar el lenguaje natural y poder interactuar de forma eficiente con los seres humanos, como lo son los Large Languaje Models (LLMs). OpenAI dispone de sus propios LLMs, en los que se basa ChatGPT. Su evolución ha sido constante a lo largo de los últimos años, desde GPT-1 hasta GPT-4 y sus variantes.
- A pesar de sus grandes capacidades, es habitual que ChatGPT no ofrezca los mejores resultados cuando se pregunta sobre ámbitos poco conocidos o se esperan respuestas demasiado concretas o precisas. La motivación del proyecto es que se adapte a un caso de uso específico, más concretamente que actúe como un tutor virtual para la asignatura de *Sistemas Inteligentes*.

- En la mayoría de los casos en los que el usuario no está satisfecho con la respuesta de ChatGPT, esta puede mejorarse considerablemente a través de prompts más complejos y elaborados. El objetivo es guiar al modelo base hacia la solución esperada a través de su comportamiento interno, sin la necesidad de procesar consultas extremadamente precisas para ofrecer buenos resultados.
- ChatGPT es capaz de responder de forma eficiente a preguntas relacionadas con el temario de la asignatura de *Sistemas Inteligentes* y podría resultar de utilidad para los estudiantes, ya que la resolución de problemas utilizando técnicas de inteligencia artificial es un campo conocido. Sin embargo, las respuestas no siempre se adaptan al temario y a las metodologías utilizadas en la asignatura.
- Hay una gran cantidad de posibilidades para llevar a cabo el proyecto, tanto a partir de un GPT como utilizando otros modelos en los que no se basa ChatGPT. Debido a los continuos avances que está sufriendo la inteligencia artificial, es fundamental explorar las posibles opciones y estudiarlas profundamente para seleccionar las que mejor se adecúen a los objetivos del proyecto. De este modo, el comportamiento se adaptará lo máximo posible al caso de uso específico.
- Crear GPTs personalizados es una gran opción para explotar las capacidades de ChatGPT en casos de uso específico. Sin embargo, se enfoca en un uso más personal para los usuarios del plan de pago. Desde el punto de vista del negocio, es conveniente que se pueda interactuar con el tutor virtual fuera de ChatGPT.
- OpenAI ofrece acceso a sus modelos a través de su API. Estos servicios se pueden integrar en aplicaciones externas a ChatGPT, dando lugar a herramientas orientadas a casos de uso específicos. Si nuestro objetivo es crear un chatbot, se puede utilizar tanto Chat Completions API como Assistants API. La elección de una opción u otra, al igual que el modelo base a utilizar o el posible ajuste del mismo, dependerá de los requisitos de cada proyecto.
- Debido al gran valor que pueden aportar los asistentes personalizados basados en ChatGPT, han surgido herramientas como Voiceflow. Esta permite crear chatbots de forma sencilla, con una lógica conversacional personalizable basada en flujos de trabajo y una fácil integración en aplicaciones y páginas web. Las respuestas del agente se generan con modelos y conocimiento dentro de la propia plataforma, o bien integrando herramientas que ya han sido creadas para ello de forma externa, como por ejemplo un asistente de Assistants API.
- Las tecnologías y herramientas seleccionadas para la realización del proyecto han sido aquellas que mejor se adecúan a sus objetivos. Los principales factores a tener en cuenta han sido la posibilidad de acceder al modelo a través de una aplicación externa a ChatGPT, las herramientas de personalización, y la facilidad para integrar el chatbot, además del coste estimado de cada una de las opciones. Responder preguntas teóricas sobre la asignatura y resolver ejercicios prácticos de una forma determinada pueden resultar tareas complejas, por lo que ha sido fundamental asegurarse de que es posible acceder a la información necesaria sin que la preparación de los datos resulte una tarea excesivamente larga y exigente.

- La arquitectura del proyecto consiste en una aplicación web que contiene un chatbot creado con Voiceflow, el cual representa el agente que actúa como tutor virtual. Para generar las respuestas a las consultas del usuario, se recurre a las capacidades de un asistente de Assistants API a través de otra aplicación web que actúa como *middleware*, en la que la API ha sido integrada.
- Una vez que se ha seleccionado la opción más adecuada, el proyecto se ha dividido en dos partes claramente diferenciadas. Una de ellas ha consistido en optimizar el comportamiento del asistente para que se adapte a la asignatura de *Sistemas Inteligentes*. Por otro lado, ha sido necesario integrar la API de OpenAI en un chatbot para que el usuario pueda interactuar con el tutor virtual y que sus respuestas sean generadas a partir del conocimiento del asistente anterior. Para integrar la API, ha sido fundamental asegurarse de que se explotan las posibilidades de Assistants API y que se muestra al usuario toda la información generada, tanto mensajes de texto como archivos.
- Una de las mayores ventajas de la arquitectura elegida es que es en mayor medida reutilizable para la realización de otros proyectos similares, ya que el comportamiento del asistente se gestiona de forma externa a través de la propia plataforma de OpenAI. Para que las respuestas sean generadas por un asistente determinado, sólo es necesario proporcionar su identificador y la clave de la API del usuario que lo ha creado. Esta es la principal razón por la que la aplicación web ha sido desarrollada antes de adaptar las respuestas del asistente al caso de uso específico, pues nos permite interactuar con cualquiera de ellos.
- Adaptar las respuestas del modelo a la asignatura de *Sistemas Inteligentes* incluye considerar tanto las cuestiones teóricas como los ejercicios ejercicios prácticos. Para conseguirlo de forma eficiente, se ha recopilado todo el material que se ha considerado útil para mejorar los resultados. Esto significa proporcionar al asistente las diapositivas de los temas la asignatura, respuestas a preguntas frecuentes, ejercicios resueltos, exámenes de años anteriores o algoritmos para facilitar la resolución de ejercicios a través del intérprete de código, así como instrucciones lo suficientemente detalladas como para poder guiar al modelo.
- Para optimizar el comportamiento del asistente a partir de los resultados iniciales tras el análisis de requisitos, se han modificado las instrucciones y el material adicional de manera iterativa, corrigiendo las respuestas incorrectas o poco precisas obtenidas en iteraciones anteriores hasta que los resultados se han considerado lo suficientemente buenos. Esta técnica para guiar al modelo hacia la solución esperada se utiliza habitualmente en *prompt engineering*, tanto para realizar consultas más elaboradas y concretas, como para mejorar el comportamiento interno de modelos orientados a casos de uso específicos.

- El asistente ha demostrado ofrecer mejores resultados que ChatGPT respecto a cuestiones relacionadas con la asignatura, por lo que se ha cumplido el objetivo del proyecto. Las respuestas a preguntas teóricas se adaptan a las explicaciones que aparecen en el temario gracias a la búsqueda en archivos, mientras que la resolución de ejercicios ha experimentado mejoras en el formato y en los resultados pese a ser una tarea especialmente compleja. Sin embargo, todavía existe un gran margen de mejora para que cada tipo de problema se adapte de forma más eficiente a la metodología utilizada en la asignatura.
- Al generar las respuestas a través de la API de OpenAI, estamos pagando por el uso de los servicios utilizados. En el caso de Assistants API, hay que considerar los tokens de entrada y salida, las sesiones con el intérprete de código activado (conversaciones de hasta una hora), el tamaño del material de apoyo vectorizado, y el número de días en los que el asistente se encuentra disponible. Por lo tanto, el coste de desplegar la aplicación y ponerla a disposición de alumnos y profesores dependerá prácticamente en su totalidad del uso se haga de ella.
- Respecto a la curva de aprendizaje y las dificultades encontradas durante el desarrollo, gran parte del proyecto ha estado marcado por el estudio de nuevas tecnologías emergentes que todavía están comenzando a ser explotadas, en un sector que evoluciona constantemente y en el que las posibilidades aumentan día a día. Sin un conocimiento previo sobre el tema, ha sido de vital importancia conocer cuáles son y cómo funcionan las diferentes alternativas para llevarlo a cabo de forma eficiente, invirtiendo una gran cantidad de tiempo en ello.

Algunas futuras mejoras son las citadas a continuación:

- Desplegar la aplicación web para que alumnos y profesores accedan libremente a ella, pudiendo interactuar con el tutor virtual en cualquier momento.
- Perfeccionar el comportamiento del asistente para obtener respuestas más completas y precisas, especialmente en la resolución de ejercicios. Para ello se puede modificar el material o refinar el prompt, aunque existe la posibilidad de reentrenar un GPT con ejercicios resueltos y utilizarlo como modelo base. Esta última alternativa surge tras el lanzamiento de una nueva versión de Assistants API durante la propia realización del trabajo.
- Seguir explorando la posibilidad de utilizar un modelo base más avanzado, con el que se obtengan mejores resultados sin que los costes aumenten de forma excesiva.
- Gestionar el acceso de los usuarios a la aplicación a través de credenciales.
- Ofrecer al usuario la posibilidad de acceder a sus chats anteriores y poder continuar dichas conversaciones, conservando el contexto correspondiente.
- Una vez realizado el despliegue, facilitar y garantizar a los usuarios la descarga de las imágenes y archivos generados por el intérprete de código.
- Tener en cuenta la evolución constante de herramientas y posibilidades, integrando aquellas nuevas funcionalidades que puedan ofrecer mejores resultados.

- Disponer de más de un asistente y optimizar cada uno de ellos para que lleve a cabo una tarea distinta de manera más sencilla. Por ejemplo, crear un asistente enfocado en responder a preguntas teóricas y otro para cada tipo de ejercicio. Se podría modificar la lógica conversacional del chatbot, generando la respuesta con un asistente u otro en función de la consulta del usuario.
- Solicitar *feedback* a los usuarios y realizar mejoras en base a ello, tanto en la aplicación web como en las respuestas generadas por el asistente.
- Mejorar la interfaz de la aplicación web, añadiendo una sección que describa las capacidades del asistente y otra que contenga manuales de uso o prompts "tipo" que ayuden a maximizar su rendimiento.

4.2. Competencias desarrolladas

Junto a aquellas relacionadas con la tecnología específica de computación ya mencionadas en la sección 1.3, a lo largo de este trabajo se han desarrollado las competencias que aparecen a continuación.

- Conocer la evolución de los Sistemas de Tutoría Inteligentes y cómo los tutores virtuales están sustituyendo a los tutores humanos.
- Estudiar las tecnologías que utiliza ChatGPT y cómo funcionan los modelos GPT en los que se basa, así como las diferencias entre ellos.
- Conocer diferentes alternativas para crear una herramienta basada en ChatGPT que se adapte a un caso de uso específico, e identificar las opciones y tecnologías que mejor se adecúan a los requisitos del proyecto.
- Aprender a integrar la API de OpenAI, comprendiendo su funcionamiento y cómo hacer uso de los diferentes servicios que ofrece. Más concretamente, aprender cómo integrar Assistants API y explotar sus posibilidades.
- Comprender en profundidad el temario de la asignatura de *Sistemas Inteligentes*.
- Identificar, reunir y generar el material adecuado en base a los requisitos del proyecto, utilizando estos recursos para que el asistente sea capaz de adaptarse al caso de uso específico y proporcione respuestas más afines que ChatGPT.
- Utilizar técnicas de *prompt engineering* para redactar instrucciones capaces de mejorar el comportamiento del asistente, guiándolo hacia la solución esperada y mejorando sus respuestas a través de un proceso iterativo.
- Evaluar los resultados obtenidos y extraer conclusiones sobre las capacidades y limitaciones del asistente para adaptarse al caso de uso específico.
- Definir una metodología y trabajar de acuerdo a ella, realizando una buena gestión del tiempo para cumplir con los plazos establecidos.

A. Metodología utilizada

Para llevar a cabo el proyecto, se ha definido una metodología compuesta por diferentes fases, explicadas a lo largo de este capítulo. Su orden y estructura se ha decidido en base a los objetivos y requisitos del proyecto.

La metodología está compuesta por las siguientes fases:

1. **Estudio de tecnologías y opciones.** En esta primera fase se realiza una investigación para conocer en profundidad las diferentes posibilidades, utilizando distintos métodos para recopilar información y evaluando las opciones con el fin de seleccionar las herramientas que más se adecúen al objetivo del proyecto.
2. **Estimación de costes.** Una vez identificadas las posibles tecnologías y opciones, se estiman los costes asociados a cada una de ellas para determinar el presupuesto.
3. **Selección de herramientas y tecnologías.** En base al estudio realizado en la primera fase y a la estimación de costes posterior, se seleccionan aquellas herramientas y tecnologías que se consideran más adecuadas.
4. **Desarrollo de la aplicación.** En esta fase, se define la arquitectura del proyecto y se desarrolla la aplicación web utilizando las herramientas y tecnologías seleccionadas en la fase anterior. Las prácticas empleadas durante el desarrollo deben garantizar progreso y adaptación a los cambios a lo largo del tiempo.
5. **Análisis de requisitos.** Una vez que se ha desarrollado la aplicación y se puede interactuar con el asistente a través de ella, es el momento de lograr que el comportamiento de este se adapte al caso de uso específico. Para ello es necesario identificar claramente los requisitos y las preguntas frecuentes, recopilando todo el material de la asignatura que pueda resultar útil y definiendo instrucciones que cubran lo máximo posible las necesidades del usuario.

6. **Optimización del comportamiento del asistente.** Con el objetivo de guiar al asistente, se proporciona el material de la asignatura necesario y se elaboran prompts a partir de las instrucciones definidas durante el análisis de requisitos. Para optimizar su comportamiento y conseguir que se adapte lo mejor posible al caso de uso, se realiza un proceso iterativo en el que se van modificando los prompts y el material para tratar de corregir respuestas no deseadas o incompletas obtenidas en iteraciones anteriores. El proceso finaliza cuando se considera que las respuestas son lo suficientemente precisas y completas. Para llevar a cabo estas evaluaciones, se comparan las respuestas obtenidas con las ideales.

7. **Resultados y comparativa con ChatGPT.** Finalmente, se realiza una comparativa en base a una serie de consultas relacionadas con la asignatura de *Sistemas Inteligentes*, evaluando los resultados y las mejoras obtenidas en las respuestas que ofrece el tutor virtual frente a las que proporciona ChatGPT.

En la figura A.1 se puede apreciar el diagrama de flujo de la metodología.

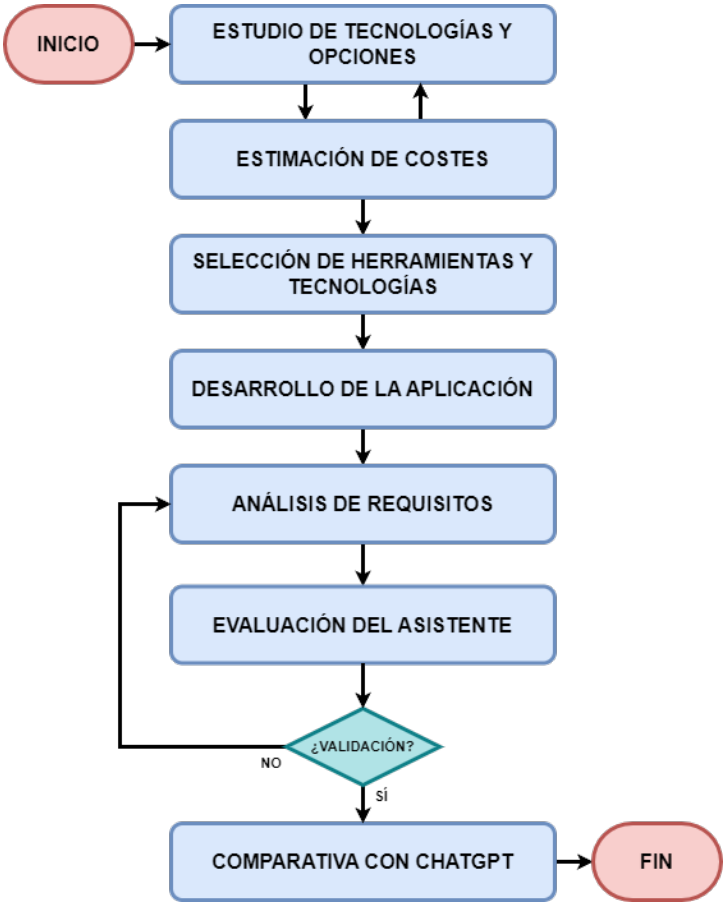


Figura A.1: Diagrama de flujo de la metodología utilizada.

B. Planificación del proyecto

La creación de un tutor virtual basado en ChatGPT ha sido un proyecto por marcado por el uso de tecnologías emergentes y que evolucionan continuamente, en un sector donde las posibilidades aumentan día a día. Este ha tenido como partes fundamentales el desarrollo de una aplicación para interactuar con el tutor virtual y la optimización de su comportamiento para adaptarse lo máximo posible al caso de uso específico.

Para llevar a cabo el proyecto se ha utilizado la metodología explicada a lo largo del capítulo A, mientras que este documento ha sido redactado de forma paralela. A continuación se detalla el trabajo realizado en cada una de sus fases, cuya duración y distribución se puede apreciar de forma más visual en el diagrama de la figura B.1.

1. **Estudio de tecnologías y opciones (7 semanas).** Ha sido la fase más larga del proyecto, debido a la importancia de conocer profundamente las diferentes posibilidades para su desarrollo.
2. **Estimación de costes (2 semanas).** Se han estimado los costes de las opciones.
3. **Selección de herramientas y tecnologías (2 semanas).** Ha sido una fase fundamental, en la que se han elegido las opciones más adecuadas. La información recopilada en fases anteriores ha permitido una toma de decisiones más rápida.
4. **Desarrollo de la aplicación (6 semanas).** Ha incluido la creación del chatbot con Voiceflow, la integración de Assistants API y el desarrollo de la aplicación web para interactuar con el tutor virtual.
5. **Análisis de requisitos (6 semanas).** Durante esta fase se ha reunido todo lo necesario para comenzar a evaluar el comportamiento del asistente, incluyendo el material de la asignatura y la definición de instrucciones adicionales que pudieran resultar útiles para cubrir las necesidades del usuario y mejorar los resultados.
6. **Optimización del comportamiento del asistente (4 semanas).** Para adaptar las respuestas del tutor virtual, se han realizado mejoras de forma iterativa.
7. **Resultados y comparativa con ChatGPT (1 semana).** Finalmente, se ha realizado una comparativa con ChatGPT y se han evaluado los resultados.

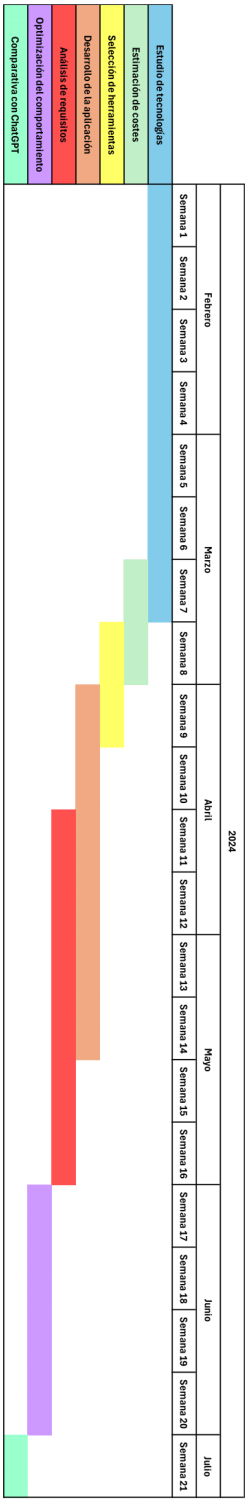


Figura B.1: Diagrama de planificación del proyecto.

C. Recursos utilizados

En este capítulo se van a listar los recursos software y hardware que han sido utilizados durante la realización del trabajo.

Recursos software

Los recursos software que se han utilizado son los citados a continuación.

- **Adobe Photoshop.** Editor de fotografías utilizado para diseñar el icono de ChatGPT-SSII (Figura 3.4) durante el desarrollo de la aplicación.
- **ChatGPT.** Chatbot de inteligencia artificial desarrollado por OpenAI, el cual ha sido utilizado para realizar la comparativa final con el tutor virtual basado en la propia herramienta.
- **Draw.io.** Software de diagramas en línea utilizado para realizar el diagrama de flujo de la metodología utilizada (Figura A.1) y el diagrama de la arquitectura del proyecto (Figura 3.1), entre otros.
- **Flask.** *Framework* escrito en lenguaje Python que permite la creación de aplicaciones web de manera sencilla, el cual ha sido utilizado para crear las aplicaciones web del proyecto.
- **HTML.** Lenguaje de marcado de hipertexto que permite estructurar y desplegar un página web y sus contenidos, el cual ha sido utilizado en el desarrollo de la aplicación que permite la interacción con el tutor virtual.
- **JavaScript.** Lenguaje de programación utilizado para eliminar las citas innecesarias en las respuestas proporcionadas por el tutor virtual, y que utiliza el código generado para integrar el asistente como un chat web.
- **Microsoft Excel.** Software de hojas de cálculo utilizado para realizar el diagrama de planificación del proyecto (Figura B.1).
- **Microsoft OneNote.** Software que permite tomar notas y dibujar en cuadernos digitales utilizado para realizar anotaciones en etapas tempranas del proyecto.

- **Microsoft Word.** Software de procesamiento de texto utilizado para elaborar los documentos con información adicional sobre la asignatura de *Sistemas Inteligentes*, como respuestas a preguntas frecuentes o ejercicios resueltos.
- **Notion.** Software de gestión de proyectos, toma de notas y aumento de la productividad, el cual ha sido utilizado para gestionar las diferentes etapas del proyecto a través de la recopilación de notas y material.
- **OpenAI API.** Interfaz de programación de aplicaciones de OpenAI, la cual ha sido utilizada para crear el tutor virtual utilizando las funcionalidades de Assistants API a través de la propia plataforma de OpenAI y para evaluar el comportamiento del tutor virtual utilizando la herramienta *Playground*.
- **OpenAI.** Librería de Python para utilizar los servicios que ofrece OpenAI API, la cual ha sido utilizada para implementar las llamadas al tutor virtual utilizando las funcionalidades de Assistants API durante el desarrollo de la aplicación.
- **Overleaf.** Editor de LaTeX en línea utilizado para redactar este documento.
- **Poetry.** Gestor de paquetes para Python que facilita la gestión de dependencias en aplicaciones y librerías, el cual ha sido utilizado para gestionar dependencias en el desarrollo de la aplicación web que conecta el chatbot con la API de OpenAI.
- **Python.** Lenguaje de programación de alto nivel y propósito general utilizado para el desarrollo de las aplicaciones web y la integración de la API de OpenAI.
- **Replit.** Entorno de desarrollo integrado para la creación y compartición de software, el cual ha sido utilizado para desarrollar la aplicación web que conecta el chatbot con la API de OpenAI.
- **Visual Studio Code.** Software creado por Microsoft para la gestión de archivos de código, el cual ha sido utilizado para gestionar los archivos Python y HTML en el desarrollo de la aplicación que permite la interacción con el tutor virtual.
- **Voiceflow.** Plataforma de creación de agentes de inteligencia artificial orientados a casos de uso específicos, la cual ha sido utilizada para crear la lógica conversacional del tutor virtual e integrar la API de OpenAI en un chatbot.
- **Windows 10 Pro.** Sistema operativo desarrollado por Microsoft sobre el que se han llevado a cabo las diferentes fases del trabajo.

Recursos hardware

Los recursos hardware que se han utilizado son los citados a continuación.

- **CPU.** AMD Ryzen 5 4600H con Radeon Graphics a 3.00 GHz.
- **RAM.** 8,00 GB.
- **Tarjeta gráfica.** NVIDIA GeForce GTX 1650.

Referencia bibliográfica

[1] New survey finds students are replacing human tutors with chatgpt - intelligent. https://www.intelligent.com/new-survey-finds-students-are-replacing-human-tutors-with-chatgpt/, 2023. Accedido: 2024-03-01.

[2] About openai. https://openai.com/about, 2024. Accedido: 2024-03-16.

[3] Api overview. https://openai.com/product, 2024. Accedido: 2024-03-20.

[4] Api reference - openai api. https://platform.openai.com/docs/api-reference, 2024. Accedido: 2024-03-27.

[5] Assistants overview - openai api. https://platform.openai.com/docs/assistants/overview, 2024. Accedido: 2024-03-28.

[6] Chatgpt. https://openai.com/chatgpt, 2024. Accedido: 2024-03-20.

[7] Explore gpts. https://chatgpt.com/gpts, 2024. Accedido: 2024-07-03.

[8] Fine-tuning - openai api. https://platform.openai.com/docs/guides/fine-tuning, 2024. Accedido: 2024-03-27.

[9] Gpt-4 by openai. https://openai.com/gpt-4, 2024. Accedido: 2024-03-20.

[10] Home - replit. https://replit.com/~, 2024. Accedido: 2024-04-11.

[11] How assistants work - openai api. https://platform.openai.com/docs/assistants/how-it-works, 2024. Accedido: 2024-03-28.

[12] Knowledge base & generative ai | voiceflow. https://www.voiceflow.com/features/knowledge-base-generative-ai, 2024. Accedido: 2024-03-29.

[13] Openai api pricing. https://openai.com/pricing, 2024. Accedido: 2024-03-27.

[14] Overview - openai api. https://platform.openai.com/docs/overview, 2024. Accedido: 2024-03-27.

[15] Poetry - python dependency management and packaging made easy. https://python-poetry.org/, 2024. Accedido: 2024-04-11.

[16] Research index by openai. https://openai.com/research, 2024. Accedido: 2024-03-16.

[17] Text generation - openai api. https://platform.openai.com/docs/guides/text-generation/chat-completions-api, 2024. Accedido: 2024-03-29.

[18] Voiceflow | build impactful ai agents. https://www.voiceflow.com/, 2024. Accedido: 2024-03-29.

[19] Welcome to flask — flask documentation (3.0.x). https://flask.palletsprojects.com/en/3.0.x/, 2024. Accedido: 2024-04-11.

[20] Why voiceflow. https://www.voiceflow.com/why-voiceflow, 2024. Accedido: 2024-03-29.

[21] Workflows | voiceflow. https://www.voiceflow.com/features/workflows, 2024. Accedido: 2024-03-29.

[22] Ali Alkhatlan and Jugal K. Kalita. Intelligent tutoring systems: A comprehensive historical survey with recent developments. https://arxiv.org/pdf/1812.09628.pdf, 2019. Accedido: 2024-02-27.

[23] Abdelbaset Almasri, Adel Ahmed, Naser Al-Masri, Yousef Abu Sultan, Ahmed Y. Mahmoud, Ihab Zaqout, Alaa N. Akkila, and Samy S. Abu-Naser. Intelligent tutoring systems survey for the period 2000- 2018. http://ijeais.org/wp-content/uploads/2019/05/IJAER190503.pdf, 2018. Accedido: 2024-02-27.

[24] Sam Altman and OpenAI. Planning for agi and beyond. https://openai.com/blog/planning-for-agi-and-beyond, 2024. Accedido: 2024-03-16.

[25] Vahid Ashrafimoghari, Necdet Gürkan, and Jordan W. Suchow. Evaluating large language models on the gmat: Implications for the future of business education. https://arxiv.org/pdf/2401.02985.pdf, 2024. Accedido: 2024-03-11.

[26] Dot CSV. ¡nuevos custom gpts: El multiverso de chatgpt! - youtube. https://www.youtube.com/watch?v=hLYwO6LYWIU, 2023. Accedido: 2024-03-23.

[27] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 1: Introduction to Artificial Intelligence. Basic concepts and a bit of history.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-20.

[28] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 10: Supervised learning: Classification. Learning functions/models from labelled data.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-21.

[29] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 2: State Space Search. Uninformed Search.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-20.

[30] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 3: Heuristic Search. Informed Search.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-20.

[31] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 4: Adversarial Search. How to act when there are enemy agents.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-22.

[32] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 5: Reinforcement Learning. Agents who learn.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-25.

[33] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 6: Combinatorial Optimization. Search in optimization: Problems and (possible) solutions.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-23.

[34] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 7: Local Search. Hill Climbing and variants.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-23.

[35] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 8: Genetic Algorithms. Solving problems using evolutionary computation.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-23.

[36] Luis De la Ossa, Julia Flores, José Antonio Gámez, Ismael García Varea, and José Miguel Puerta. *Intelligent Systems - Lesson 9: Machine Learning. Introduction. Machines can also learn from experience.* Grado en Ingeniería Informática. Escuela Superior de Ingeniería Informática de Albacete. Universidad de Castilla-La Mancha, 2022. Accedido: 2024-02-21.

[37] Yang Deng, Zifeng Ren, and An Zhang. Towards goal-oriented intelligent tutoring systems in online education. https://arxiv.org/pdf/2312.10053.pdf, 2023. Accedido: 2024-03-11.

[38] DonebyLaura. Cómo crear chatbots con ia | tutorial completo para principiantes 2024. https://youtu.be/mn6mOkiYjIg?si=FKwXCWomTIJmleqQ, 2024. Accedido: 2024-03-08.

[39] Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. Large language models in education: Vision and opportunities. https://arxiv.org/pdf/2311.13160.pdf, 2023. Accedido: 2024-03-11.

[40] Rob Hayes and Voiceflow. Launching to your website – voiceflow. https://learn.voiceflow.com/hc/en-us/articles/10569376208781-Launching-to-your-Website, 2023. Accedido: 2024-03-31.

[41] Valerie J. Shute and Joseph Psotka. Intelligent tutoring systems: Past, present and future. https://myweb.fsu.edu/vshute/pdf/shute%201996_d.pdf, 1996. Accedido: 2024-02-26.

[42] AI Jason. "okay, but i want gpt to perform 10x for my specific use case" - here is how. https://youtu.be/Q9zv369Ggfk?si=8ItXL8IMMYPNEA_V, 2023. Accedido: 2024-02-16.

[43] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. https://arxiv.org/pdf/2307.10169.pdf, 2023. Accedido: 2024-03-11.

[44] OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 2022. Accedido: 2024-03-20.

[45] OpenAI. Chatgpt can now see, hear, and speak. https://openai.com/blog/chatgpt-can-now-see-hear-and-speak, 2023. Accedido: 2024-03-20.

[46] OpenAI. Dall · e 3 is now available in chatgpt plus and enterprise. https://openai.com/blog/dall-e-3-is-now-available-in-chatgpt-plus-and-enterprise, 2023. Accedido: 2024-03-20.

[47] OpenAI. Introducing chatgpt plus. https://openai.com/blog/chatgpt-plus, 2023. Accedido: 2024-03-20.

[48] OpenAI. Introducing gpts. https://openai.com/blog/introducing-gpts, 2023. Accedido: 2024-03-20.

[49] OpenAI. Introducing gpts in chatgpt - youtube. https://www.youtube.com/watch?v=CmP3XXwKJ60, 2023. Accedido: 2024-03-23.

[50] OpenAI. New products: A deep dive - youtube. https://www.youtube.com/watch?v=pq34V_V5j18, 2023. Accedido: 2024-03-23.

[51] OpenAI. Gpt-4 research. https://openai.com/research/gpt-4, 2024. Accedido: 2024-03-20.

[52] OpenAI. Gpt-4 technical report. https://arxiv.org/pdf/2303.08774.pdf, 2024. Accedido: 2024-03-20.

[53] OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o, 2024. Accedido: 2024-05-15.

[54] OpenAI. Introducing the gpt store. https://openai.com/blog/introducing-the-gpt-store, 2024. Accedido: 2024-03-23.

[55] Liam Ottley. How to add custom gpts to any website in minutes. https://youtu.be/Uk5f3ajkfSs?si=laiIFo4fG4Rf3OlO, 2023. Accedido: 2024-02-15.

[56] Liam Ottley. How to create advanced gpts for your website. https://youtu.be/Kn6k6ocEaK4?si=59NUKsAMR5Gewfoy, 2023. Accedido: 2024-02-16.

[57] Sankalan Pal Chowdhury, Vilém Zouhar, and Mrinmaya Sachan. Scaling the authoring of autotutors with large language models. https://arxiv.org/pdf/2402.09216.pdf, 2024. Accedido: 2024-03-11.

[58] David Sharma. Openai's assistants api uncovered: Building a real-time train assistant integrated with multiple apis. https://blog.gopenai.com/openais-assistants-api-uncovered-building-a-real-time-train-assistant-integrated-with-multiple-eb8d69fb3bf4, 2023. Accedido: 2024-03-31.

[59] Priya Shree. Gpt models explained. open ai's gpt-1,gpt-2,gpt-3 | walmart global tech blog. https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2, 2020. Accedido: 2024-03-20.

[60] Wayne Xin Zhao, Kun Zhou, Junjy Li, Tianjy Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. https://arxiv.org/pdf/2303.18223.pdf, 2023. Accedido: 2024-03-11.