UNIVERSIDAD NACIONAL DE CATAMARCA

FACULTAD DE TECNOLOGÍA Y

CIENCIAS APLICADAS

INGENIERÍA EN INFORMÁTICA

TRABAJO FINAL

Desarrollo e Implementación de un asistente chatbot en la cátedra Informática de la carrera Ingeniería Electrónica de la Facultad de Tecnologías y Ciencias Aplicadas

Autor:

JORGE E. HERMOSILLA PEDRAZA - MUN°01005

Director:

MG. ANA DEL PRADO

Catamarca, Marzo de 2025

AGRADECIMIENTOS

A mis padres, quienes han sido mi guía en esta vida, por confiar en mí y brindarme su apoyo incondicional. Gracias a su esfuerzo y dedicación, hoy puedo ofrecerles este trabajo, que representa el cierre de una etapa fundamental en mi formación profesional.

A mis abuelos, que siempre han celebrado mis logros y me han acompañado con sus oraciones y buenos deseos.

A mis amigos, por estar a mi lado en los momentos difíciles, por alentarme a seguir adelante y por sacarme una sonrisa cuando más lo necesito.

A mis compañeros, quienes nunca dudaron en tenderme una mano y con quienes he compartido el crecimiento en este camino. Son profesionales a quienes admiro y con quienes siempre quisiera contar como aliados y mentores.

A mis profesores, por transmitir su conocimiento, su pasión por la informática y por enseñarme a ver cada faceta de esta disciplina con curiosidad y entusiasmo.

Y, por último, a la directora de este trabajo, por su apoyo incondicional en cada etapa de su realización. Sin su guía y compromiso, este proyecto no habría sido posible.

A todos ustedes, mi más sincero agradecimiento.

Índice/Tabla de Contenidos

Agradecimientos **1**

Índice de Figuras **6**

Resumen **8**

1. Capítulo I – Marco Metodológico **9**

1.1. Introducción 10

1.2. Planteamiento del problema **12**

1.3. Antecedentes 14

1.3.1. Desarrollo de la inteligencia artificial aplicada a la educación en argentina 14

1.3.2. Desarrollo de la inteligencia artificial aplicada a la educación en el mundo 15

1.3.3. Historia de los chatbots en educación 16

1.3.4. Estado actual de la tecnología de chatbots en educación 16

1.3.5. Integración de chatbots en sistemas de gestión del aprendizaje (LMS) 17

1.4. Justificación **18**

1.5. Objetivos *19*

1.5.1. Objetivo General 19

1.5.2. Objetivos Específicos 19

1.6. Diseño Metodológico *20*

1.6.1. Tipo de Estudio 20

1.6.2. Técnicas e Instrumentos de Recolección de Datos 20

1.6.3. Procedimiento 20

2. Capítulo II – Marco Teórico **22**

2.1. Chatbots en educación y otros sectores 23

2.2. Introducción a los Modelos de Lenguaje 24

2.2.1. Evolución de los chatbots y asistentes virtuales 24

2.2.2. Modelos de lenguaje tradicionales vs. modernos 25

2.3. Large Language Models (LLMs) 26

2.3.1. Definición y características de LLM 26

2.3.2. Arquitectura y funcionamiento 26

2.3.3. Capacidades y limitaciones de LLM 26

2.4. Aplicación de LLM en Chatbots 28

2.4.1. Cómo los LLM mejoran la interacción en chatbots educativos 28

2.4.2. Tutoría mejorada mediante LLM 29

2.4.3. Generación de contenido educativo y retroalimentación 29

2.5. Beneficios de los LLM en la Educación 30

2.5.1. Aprendizaje personalizado y adaptativo 30

2.6. Desafíos en el uso de LLM en la educación 31

2.6.1. Sesgos y errores de los LLM 31

2.6.2. Dependencia tecnológica y monitoreo 31

2.7. Implementación de LLM en el chatbot de tutoría....32
2.7.1. Integración con otras tecnologías....32
2.7.2. Métricas de éxito y evaluación del desempeño....33
2.8. Perspectivas futuras de los LLM en el ámbito educativo....34
2.8.1. Mejora continua en los LLM....34
2.8.2. Potencial de los LLM en la enseñanza autónoma....34
2.9. Marco Conceptual de RAG (Retrieval-Augmented Generation)....35
2.9.1. Definición y funcionamiento de RAG:....35
2.9.2. Características de RAG:....35
2.9.3. Diferencias con otros enfoques generativos:....36
2.9.4. Componentes de RAG....36
2.9.4.1. Módulo de recuperación (Retrieval)....36
2.9.4.2. Módulo de generación (Generation)....37
2.9.5. Recuperación de Información en RAG....37
2.9.5.1. Representación de documentos....37
2.9.5.2. Métodos de indexación y búsqueda....38
2.9.5.3. Medidas de similitud....39
2.9.6. Generación de respuestas en RAG....40
2.9.6.1. Modelos generativos....40
2.9.6.1.1. Ejemplos de Modelos de Lenguaje:....40
2.9.6.2. Mecanismo de combinación de recuperación y generación....41
2.9.6.3. Ajustes y personalización....41
2.9.7. Desafios tecnicos....42
2.9.7.1. Sesgo en los Datos Recuperados....42
2.9.7.2. Costo computacional....42
2.9.7.3. Problemas de escalabilidad....42
2.9.8. Consideraciones éticas....43
2.9.8.1. Privacidad de Datos....43
2.9.8.2. Sesgos en las respuestas....44
2.9.8.3. Transparencia....44
2.9.8.4. Limitaciones del Sistema RAG....45
2.9.9. Aplicaciones y Casos de Estudio de Chatbots Basados en RAG....45
2.9.9.1. Aplicaciones prácticas....45
2.9.9.2. Casos de estudio relevantes....46
2.9.10. Perspectivas Futuras y Avances en Chatbots RAG....47
2.9.10.1. Innovaciones en la Recuperación de Información....47
2.9.10.2. Nuevas Arquitecturas Generativas....49
Integración con Otras Tecnologías....50
3. Capítulo III....52

3.1. Introducción..... 53
3.2. Análisis y requerimientos..... 53
3.3. Limitaciones del sistema..... 53
3.4. Diseño del sistema..... 54
3.5. Implementación..... 59
3.5.1. Desarrollo de la lógica del chatbot..... 59
3.5.2. Configuración del tablero Kanban..... 59
3.5.3. Herramientas y tecnologías utilizadas..... 60
3.5.3.1. Python..... 60
3.5.3.2. Streamlit..... 61
3.5.3.3. GPT-4.0-mini..... 61
3.5.3.4. Langchain..... 62
3.5.3.5. FAISS (Facebook AI Similarity Search)..... 62
3.5.3.6. Trello..... 62
3.6. Estructura de la base de conocimientos..... 62
3.6.1. Estructura general del documento..... 63
3.6.2. Estructura interna de cada unidad..... 63
3.7. Pruebas..... 65
3.8. Recolección y Análisis de Feedback..... 65
3.9. Planificación y Control del Proyecto..... 68
3.9.1. Tareas de desarrollo..... 68
3.9.2. Diagrama de Gantt..... 69
3.9.3. Formato del tablero Kanban..... 69
3.9.4. Tareas que forman parte del backlog..... 69
3.9.5. Evolución en el tablero Kanban..... 71
3.10. Capturas de pantalla del sistema implementado..... 73
3.11. Buenas prácticas para el desarrollo de un chatbot RAG..... 75
3.12. Tecnicas de ingenieria de prompts aplicadas en el chatbot..... 78
3.13. Reflexiones finales..... 81
3.14. Mejoras a futuro..... 81
4. Conclusión..... 84
4.1.1. Impacto general del Chatbot en el aprendizaje..... 84
4.1.2. Modelos de lenguaje utilizados..... 84
4.1.3. Evaluación de las funcionalidades técnicas..... 85
4.1.4. Fortalezas del sistema..... 85
4.1.5. Áreas de mejora y limitaciones..... 85
4.1.6. Relevancia y sostenibilidad..... 86
4.1.7. Conclusión Final..... 86
Bibliografía..... 88

Anexo I: Encuestas realizadas a estudiantes y Lista de cotejo.....104
Anexo II: Épicas y Historias de Usuario.....116
Anexo III: Tareas.....122
Anexo IV: Métricas.....125

Índice de Figuras

Figura 3-1 : Arquitectura de un chatbot basado en RAG.....53
Figura 3-2 : Diagrama de contexto.....54
Figura 3-3 : Diagrama de secuencia de una consulta al chatbot.....54
Figura 3-4 : Pantalla principal.....55
Figura 3-5 : Pantalla de consulta al chatbot.....56
Figura 3-6 : Pantalla de recursos extra.....56
Figura 3-7 : Pantalla de autoevaluación.....57
Figura 3-8 : Fragmento de una unidad del documento usado para la base de conocimientos.....62
Figura 3-9 : Árbol de características.....65
Figura 3-10 : Tablero Kanban inicial.....67
Figura 3-11 : Tablero Kanban a la tercer semana.....68
Figura 3-12 : Tablero Kanban a la sexta semana.....68
Figura 3-13 : Tablero Kanban en la novena semana.....69
Figura 3-14 : Tablero Kanban en la semana doce.....69
Figura 3-15 : Interfaz del sistema - Inicio.....70
Figura 3-16 : Interfaz del sistema - Chatbot.....70
Figura 3-17 : Interfaz del sistema - Recursos.....71
Figura 3-18 : Interfaz del sistema - Soporte para ejercicios.....71
Figura 3-19 : Interfaz del sistema - Evaluación.....72

RESUMEN

El sistema desarrollado es un chatbot basado en Retrieval-Augmented Generation (RAG), diseñado específicamente para brindar soporte a la cátedra de Informática de la carrera de Ingeniería en Electrónica de la Facultad de Tecnología y Ciencias Aplicadas. Este chatbot tiene como propósito principal asistir a los estudiantes en el aprendizaje de programación en C, proporcionando respuestas precisas y claras a sus consultas, además de ejemplos prácticos y explicaciones conceptuales.

El sistema utiliza una base de conocimientos cuidadosamente curada que incluye documentación técnica, apuntes de la cátedra, ejercicios, y preguntas frecuentes sobre C. A través del enfoque RAG, el chatbot combina técnicas de recuperación de información con generación natural de lenguaje, garantizando que las respuestas sean relevantes, completas y alineadas con los materiales de estudio de la cátedra.

Palabras claves: chatbot, desarrollo de software, inteligencia artificial, educación, RAG.

CAPÍTULO I

Introducción

1.1. INTRODUCCIÓN

El desarrollo de big data, inteligencia artificial y la Web 3.0 ha incrementado la importancia de la investigación en LLMs (Large Language Models). Estos modelos han demostrado una gran capacidad en la generación, comprensión y procesamiento del lenguaje natural (PNL), sistemas de diálogo, traducción, generación de contenido por inteligencia artificial, entre otras aplicaciones. Los LLM han desarrollado una gran capacidad en el entendimiento del conocimiento humano y en varios campos profesionales, como sistemas de recomendación, salud, economía y otros. Con los avances en LLMs, también se promueve el desarrollo de tecnologías emergentes que mejoran la interacción hombre-máquina, haciendo que la tecnología sea más intuitiva y útil para el usuario promedio (Gan, et al., 2018).

En este contexto, la educación se presenta como un ámbito crucial para la aplicación de los LLMs, enfrentando desafíos como las diferencias individuales de los estudiantes, la asignación insuficiente de recursos docentes y la evaluación de la efectividad educativa. Por lo que destaca la necesidad de estrategias educativas más atractivas y personalizadas para involucrar a los alumnos de manera efectiva (Gan, et al., 2023).

Los Modelos de Lenguaje de Gran Escala (LLMs) representan una transformación significativa en las prácticas educativas tradicionales. Una de sus principales aplicaciones es la generación de contenido de aprendizaje personalizado y recomendaciones basadas en las necesidades e intereses individuales de los estudiantes. Al analizar los datos de aprendizaje y los patrones de comportamiento de los estudiantes, los LLMs pueden diseñar rutas de aprendizaje y recursos únicos para cada estudiante, ayudándoles a aprender de manera más eficiente. Además, los LLMs pueden servir como asistentes para los profesores, proporcionando herramientas y plataformas de soporte instructivo inteligente. Los docentes pueden utilizar el contenido y las recomendaciones generadas por los LLMs para diseñar actividades de enseñanza, monitorear el progreso de los estudiantes y ofrecer apoyo educativo personalizado (Gan, et al., 2023).

Asimismo, la incorporación de LLMs en el aula puede facilitar la evaluación adaptativa, la tutoría inteligente y otros aspectos que pueden mejorar la calidad de la educación y la

experiencia de aprendizaje. Los LLMs pueden ayudar a evaluar los resultados del aprendizaje, proporcionar recomendaciones personalizadas y revolucionar los modelos de enseñanza tradicionales al permitir un aprendizaje personalizado y tutoría inteligente. Además, los LLMs tienen una amplia gama de aplicaciones en el procesamiento del lenguaje natural, asistentes virtuales, servicio al cliente y escritura inteligente. Pueden proporcionar generación y comprensión del lenguaje para diversas tareas y escenarios, mejorando la comunicación y la interacción en entornos educativos. Por último, los LLMs pueden aprender y actualizarse continuamente entrenando con nuevos datos, acumulando nuevos conocimientos y patrones de lenguaje, y mejorando su rendimiento y capacidades con el tiempo (Gan, et al., 2023).

Las tecnologías de IA tienen un potencial transformador en el ámbito educativo. Al utilizarlas, los educadores pueden adaptar el proceso de enseñanza para satisfacer las necesidades de cada estudiante, estos pueden obtener retroalimentación instantánea, recursos educativos personalizados, y adaptaciones en tiempo real para abordar las necesidades específicas de cada estudiante. Al permitir un enfoque más individualizado, los estudiantes pueden avanzar a su propio ritmo, profundizar en áreas de interés y recibir apoyo adicional en áreas de dificultad. Esto no solo mejora la experiencia de aprendizaje, sino que también promueve la autonomía y la autoeficacia de los estudiantes al asumir un papel más activo en su educación. Lo que conduce a un enfoque más personalizado y centrado en el estudiante, fomentando la independencia, motivación y compromiso de los estudiantes con el proceso de aprendizaje. (Chen, et al., 2020).

La implementación de chatbots en la educación ofrece una serie de beneficios significativos que mejoran la experiencia de aprendizaje de los estudiantes. En primer lugar, los chatbots aumentan el compromiso de los estudiantes al proporcionar experiencias de aprendizaje interactivas, lo que les permite participar de manera más activa en el proceso educativo. Además, la accesibilidad y la inclusión son mejoradas gracias a la disponibilidad de los chatbots las 24 horas del día, brindando soporte a estudiantes de todo el mundo en cualquier momento. Este acceso constante asegura que los estudiantes puedan recibir ayuda y aclaraciones instantáneas, sin importar su ubicación o la hora del día,

eliminando barreras y facilitando un aprendizaje continuo y fluido (Arias-Navarrete, et al., 2020).

Otro beneficio clave de los chatbots en la educación es la retroalimentación instantánea que pueden ofrecer a los estudiantes, mejorando significativamente su aprendizaje al proporcionar comentarios inmediatos sobre su desempeño. Además, los chatbots tienen la capacidad de recomendar recursos de estudio personalizados según las preferencias individuales de cada estudiante, lo que mejora la satisfacción del usuario y promueve una comprensión más profunda de los conceptos de una determinada asignatura. Esto no solo fomenta una actitud positiva hacia el aprendizaje, sino que también puede incrementar la motivación de los estudiantes hacia la materia. Por último, los chatbots son una solución eficiente en términos de costos, ya que reducen la necesidad de personal de soporte, permitiendo a las instituciones educativas ahorrar recursos mientras proporcionan un soporte constante y de alta calidad a sus estudiantes (Arias-Navarrete, Palacios-Pacheco, & Villegas-Ch., 2020).

1.2. Planteamiento del problema

En la Facultad de Tecnología y Ciencias Aplicadas de la Universidad Nacional de Catamarca (UNCa), se imparte la materia Informática, correspondiente al segundo año de la carrera de Ingeniería Electrónica. Este curso cuenta con una matrícula de entre 20 y 25 estudiantes y se dicta en el segundo cuatrimestre con una duración de 6 horas semanales. Un docente se encarga de las clases teóricas y de las prácticas. Sin embargo, se enfrentan varios desafíos que limitan el aprendizaje efectivo de la programación en Lenguaje C.

El principal problema es el escaso tiempo disponible para la práctica debido al esquema cuatrimestral de seis horas semanales. Este tiempo limitado dificulta que los estudiantes, especialmente aquellos sin experiencia previa, adquieran una comprensión profunda y habilidades sólidas en programación. Además, la diversidad en los niveles de conocimiento previos sobre programación entre los estudiantes hace difícil adaptar las actividades a las necesidades individuales, lo que impide que todos los estudiantes puedan avanzar de manera óptima.

Por otra parte, una problemática significativa radica en la complejidad conceptual del tema de punteros en Lenguaje C. Esto se debe a que la manipulación de direcciones de memoria y la gestión de punteros requieren un alto nivel de abstracción y precisión, lo que resulta difícil de comprender para muchos estudiantes. Las dificultades para entender y utilizar correctamente los punteros pueden llevar a errores frecuentes y a una creciente frustración, afectando negativamente el rendimiento académico y la confianza de los estudiantes en sus habilidades de programación.

De manera similar, los estudiantes enfrentan dificultades al intentar comprender y aplicar la lógica de programación en ejemplos prácticos relacionados con situaciones reales, especialmente cuando se utiliza el método de aprendizaje invertido combinado con el aprendizaje basado en casos. Estos casos, que a menudo están relacionados con situaciones laborales futuras y tienen una extensión media o larga, deben desarrollarse fuera del aula sin el apoyo directo del docente, lo cual es particularmente desafiante para los estudiantes de segundo año que aún requieren orientación y tutoría durante estas actividades.

A pesar de que la materia no presenta altas tasas de abandono, existe una necesidad constante de mantener y aumentar la motivación y el compromiso de los estudiantes. La programación, siendo un área desafiante, requiere métodos innovadores para mantener a los estudiantes interesados y comprometidos con el aprendizaje continuo. Para superar estas problemáticas y mejorar la calidad del aprendizaje en la asignatura de Informática, se propone el desarrollo e implementación de un asistente inteligente potenciado por inteligencia artificial. Esta propuesta se fundamenta en la necesidad de mejorar continuamente las metodologías de enseñanza y en la evidencia de que los asistentes inteligentes pueden enriquecer significativamente la experiencia educativa, proporcionando un aprendizaje más personalizado y eficaz.

1.3. ANTECEDENTES

1.3.1. Desarrollo de la inteligencia artificial aplicada a la educación en argentina

En la Universidad Nacional de Catamarca se diseñó un chatbot utilizando Node.Js y la plataforma DialogFlow para desarrollar un modelo conversacional asociado al proceso de atención a interesados en iniciar estudios en la Facultad de Tecnología y Ciencias Aplicadas (Ahumada, et al., 2021). De manera similar, en la Universidad Nacional de la Plata se implementó un chatbot en el área de asuntos estudiantiles, el cual brinda información sobre comedores universitarios, sedes, becas, etc. Entre las tecnologías utilizadas: Facebook Messenger, Java (Spring, Maven), Jetti y una API externa Wit.ia (Gago & Navajas, 2019).

Se implementó un chatbot en la asignatura Análisis y Diseño de Sistemas I de la carrera Analista Programador Universitario de la Universidad Nacional de Jujuy, utilizando el procesamiento del lenguaje natural mediante la librería de Python, Gensim. El chatbot tomaba datos de una base de datos previamente cargada con contenidos de la materia, su capacidad de procesamiento del lenguaje natural le permitía responder a las preguntas realizadas por estudiantes (Medrano, et al, 2018).

En la Universidad Nacional de Jujuy se creó un chatbot para la asignatura Análisis y Diseño de Sistemas I de la carrera Analista Programador Universitario, utilizando el procesamiento del lenguaje natural mediante la librería de Python, Gensim. El chatbot tomaba datos de una base de datos previamente cargada con contenidos de la materia, su capacidad de procesamiento del lenguaje natural le permitía responder a las preguntas realizadas por estudiantes (Medrano, et al, 2018). Asimismo, dentro de la Facultad de Ciencias Jurídicas y Sociales de la UNLP, mediante el servicio de Chatfuel implementaron un chatbot asistente que resuelve consultas sobre la comisión (Batista, 2017).

1.3.2. Desarrollo de la inteligencia artificial aplicada a la educación en el mundo

Chatbot Lola: En julio de 2018, la Universidad de Murcia y la Universidad Politécnica de Cartagena presentaron Lola, un chatbot diseñado para ayudar a los futuros estudiantes en sus procesos de preinscripción y matrícula. Esta iniciativa las convirtió en las primeras universidades de España en implementar esta tecnología. Lola emplea la tecnología Dialogflow de Google, lo que le permite aprender de las interacciones y mejorar continuamente su rendimiento.

Chatbot Beacon: En enero de 2019, la Universidad de Staffordshire presentó Beacon, un chatbot diseñado para brindar asistencia las 24 horas a sus estudiantes. La universidad se asoció con un proveedor de servicios en la nube, ANS Group, lo que la convirtió en la primera institución en el Reino Unido en implementar esta tecnología. Beacon ofrece información personalizada sobre los horarios de los cursos para los estudiantes matriculados, facilita la comunicación entre los estudiantes y sus tutores personales y responde a 400 preguntas frecuentes sobre las instalaciones del campus y los servicios de apoyo. El chatbot identifica a los estudiantes al iniciar sesión en la aplicación, lo que les permite elegir entre interacciones de texto o voz.

Chatbot Lucy: En febrero de 2018, la Universidad de Canberra dio a conocer a Lucy, un chatbot para brindar información importante a sus estudiantes. Para brindar respuestas precisas a las consultas de los usuarios, Lucy escanea el contenido de sitios de soporte de la universidad como por ejemplo AskUC. De esta manera puede ayudar a los estudiantes con consultas que van desde la inscripción y los horarios de clase hasta el estacionamiento de automóviles y los servicios para estudiantes. Cuando Lucy no puede responder una pregunta, los estudiantes pueden pedirle al sistema que levante un ticket con el equipo del Centro de Estudiantes de la Universidad para recibir una respuesta (Gago & Navajas, 2019).

1.3.3. Historia de los chatbots en educación

- Eliza(1966): Un chatbot pionero desarrollado por Joseph Weizenbaum en 1966, capaz de entablar tipos específicos de conversaciones entre humanos y computadoras. Procesaba oraciones de entrada, analizaba la presencia de ciertas palabras clave y generaba respuestas basadas en reglas predefinidas. Los principales problemas técnicos a afrontar fueron la identificación de palabras clave, entender el mínimo contexto, generar respuestas en la ausencia de palabras clave (Shrager, 2024).

- ALICE (1995): ALICE, desarrollado por Richard Wallace, fue uno de los primeros chatbots en utilizar una base de conocimientos para mantener conversaciones más avanzadas (AbuShawar & Atwell, 2015).

- Jill Watson (2016): Desarrollado por IBM, Jill Watson fue un chatbot utilizado en un curso de inteligencia artificial en la Universidad de Georgia. Los estudiantes interactúan con Jill creyendo que era una asistente de enseñanza humana (Kakar et. al,2024).

- Squirrel AI (2018): Squirrel AI es un chatbot de tutoría personalizado que utiliza inteligencia artificial para adaptarse al ritmo de aprendizaje de cada estudiante (Arias-Navarrete, et al., 2020).

1.3.4. Estado actual de la tecnología de chatbots en educación

Los chatbots en el ámbito educativo están siendo diseñados para proporcionar una experiencia de aprendizaje más personalizada. Utilizan algoritmos de inteligencia artificial para adaptarse a las necesidades individuales de los estudiantes, ofreciendo recursos y actividades específicas según su progreso y estilo de aprendizaje (Ait Baha, et al., 2023). Además, los chatbots se utilizan como tutores virtuales, brindando apoyo en tiempo real a los estudiantes. Estos sistemas pueden responder preguntas, proporcionar explicaciones y guiar a los estudiantes a través de problemas o conceptos difíciles (Koivisto, 2021).

Los chatbots también son capaces de proporcionar retroalimentación inmediata sobre el trabajo de los estudiantes. Esto puede incluir corrección gramatical en ensayos escritos, evaluación de respuestas a preguntas de opción múltiple y comentarios sobre el proceso de resolución de problemas (Fernoagă, et al., 2017).

Algunos chatbots educativos están diseñados para proporcionar apoyo emocional y motivacional a los estudiantes. Estos sistemas pueden utilizar técnicas de psicología positiva y coaching para ayudar a los estudiantes a superar desafíos y mantener la motivación (Pesonen, 2021) (Klos, et al., 2021).

Además, los chatbots se están integrando en plataformas de aprendizaje existentes, como sistemas de gestión del aprendizaje (LMS), para proporcionar una experiencia de usuario más fluida y coherente. Esto facilita el acceso de los estudiantes a los recursos adicionales (Bezverhny, et al., 2020).

Las diversas necesidades de los estudiantes, incluida la tutoría personalizada, la retroalimentación y el apoyo emocional, se pueden abordar de manera eficaz mediante la adopción y el uso de chatbots.

1.3.5. Integración de chatbots en sistemas de gestión del aprendizaje (LMS)

La integración de chatbots en sistemas de gestión del aprendizaje (LMS) es un tema relevante en la educación actual. Los chatbots, como asistentes virtuales, pueden mejorar la experiencia de aprendizaje al ofrecer interacciones personalizadas e inteligentes. Algunos aspectos clave incluyen:

Personalización: Los chatbots pueden adaptarse a las necesidades individuales de los estudiantes, proporcionando recomendaciones específicas y retroalimentación personalizada.

Análisis de Datos: La interacción con un chatbot genera datos valiosos. Estos datos pueden utilizarse para comprender mejor el progreso del estudiante y tomar decisiones informadas.

Recomendación de Actividades: Un chatbot integrado en un LMS puede sugerir actividades, recursos o rutas de aprendizaje basadas en el perfil del estudiante y su historial.

Inteligencia Artificial (IA): La IA permite que los chatbots emulen procesos de pensamiento humano y ofrezcan respuestas contextuales y relevantes.

1.4. JUSTIFICACIÓN

La enseñanza de programación, en particular en el Lenguaje C, presenta desafíos significativos que impactan la experiencia de aprendizaje de los estudiantes. Entre ellos, el tiempo limitado disponible para la práctica, las diferencias en los niveles de conocimiento previo y la complejidad de ciertos conceptos clave dificultan el avance de los estudiantes y limitan la personalización de la enseñanza. Estas limitaciones requieren soluciones innovadoras que permitan brindar soporte continuo y adaptado a las necesidades de cada estudiante.

Un chatbot educativo potenciado por inteligencia artificial se plantea como una herramienta clave para abordar estas problemáticas. Este sistema permite complementar la enseñanza presencial al ofrecer un apoyo constante para resolver dudas, reforzar conceptos y guiar en la resolución de ejercicios prácticos. Además, su disponibilidad en cualquier momento brinda a los estudiantes una oportunidad única de acceder a recursos y respuestas fuera del aula, reduciendo la dependencia exclusiva del tiempo de clase.

La implementación de un chatbot no solo mejora la accesibilidad al aprendizaje, sino que también contribuye a incrementar la motivación y el compromiso de los estudiantes. Al interactuar con un sistema que responde de manera clara y contextual a sus consultas, los estudiantes experimentan un aprendizaje más dinámico, autónomo y adaptado a sus necesidades específicas.

Finalmente, esta propuesta también beneficia a los docentes, al facilitar el seguimiento del progreso de los estudiantes y proporcionarles datos valiosos para ajustar sus estrategias pedagógicas. Al integrar esta tecnología en el ámbito educativo, se da un paso hacia la modernización de las metodologías de enseñanza, promoviendo un aprendizaje más efectivo, inclusivo y centrado en el estudiante.

1.5. OBJETIVOS

1.5.1. Objetivo General

Desarrollar e implementar un asistente inteligente que sea capaz de brindar asistencia en el cursado de estudiantes universitarios de la cátedra Informática de la carrera Ingeniería en Electrónica de la Facultad de Tecnología y Ciencias Aplicadas de la Universidad de Catamarca.

1.5.2. Objetivos Específicos

a) Utilizar un modelo de lenguaje natural que pueda comprender y responder preguntas específicas sobre los contenidos de la cátedra Informática.

b) Recopilar y digitalizar el material de estudio de la cátedra, como notas de clase, libros de texto, artículos académicos y ejemplos de código, asegurando su sistematización en un formato adecuado para integración tecnológica.

c) Identificar fortalezas y debilidades en el desarrollo de este tipo de herramientas pedagógicas.

d) Diagramar una guía de buenas prácticas para la implementación de este tipo de tecnologías.

e) Recopilar feedback de usuarios del asistente, para realizar mejoras futuras en el sistema.

1.6. Diseño Metodológico

1.6.1. Tipo de Estudio

El trabajo final propuesto corresponde al tipo de investigación denominado desarrollo tecnológico.

1.6.2. Técnicas e Instrumentos de Recolección de Datos

Para el desarrollo del sistema, se aplicó la metodología Kanban, priorizando las tareas que agregan más valor al producto y las que eran independientes de otras (Ahmad et al., 2013).

Para la recolección de datos, se utilizaron las siguientes técnicas e instrumentos:

- **Entrevistas**: Se realizaron entrevistas para ampliar la visión de la problemática y profundizar en los detalles importantes. Estas entrevistas incluyeron reuniones semanales con el profesor de cátedra.
- **Encuestas**: Se llevaron a cabo una serie de preguntas a los estudiantes que probaron la aplicación, con el fin de conocer la experiencia de estudiantes e identificar posibles áreas de mejora. Los encuestados fueron estudiantes de la cátedra de Informática de la carrera de Ingeniería en Electrónica.
- **Historias de Usuario**: Utilizadas para especificar los requisitos del software y priorizar su desarrollo.

1.6.3. Procedimiento

El procedimiento general que se llevó a cabo cubrió las siguientes fases:

- Fase 1: Análisis exploratorio. Consistió en la búsqueda, recolección, lectura comprensiva y análisis de las fuentes de información (bibliografía, publicaciones, sitios web, entre otros) relacionadas con el tema de investigación. Este análisis permitió establecer las bases teóricas y conceptuales necesarias para la implementación de la metodología Kanban.

- Fase 2: Desarrollo del Sistema de Información

Al utilizar la metodología Kanban, esta fase se estructuró en las siguientes etapas continuas y colaborativas:

- **Configuración del Tablero Kanban**: Se definieron las etapas del flujo de trabajo y los criterios de avance de tareas en el tablero Kanban. Se identificaron columnas como “Pendiente”, “En Proceso”, “En Revisión” y “Completado” para mantener un flujo de trabajo visual y ordenado.
- **Priorización y Gestión de Tareas**: Las historias de usuario se priorizaron y clasificaron en el tablero Kanban para gestionar el flujo de trabajo según las necesidades y prioridades del cliente. La planificación incluyó establecer reuniones periódicas con el cliente para revisar avances, ajustar prioridades y establecer acuerdos de contenido.
- **Iteraciones Continuas**: A través de iteraciones, se monitoreó y actualizó el tablero, realizando entregas incrementales del software según las historias de usuario completas. Cada iteración permitió evaluar el trabajo en función de las prioridades, garantizando la adaptación continua a los requerimientos del cliente.

- Fase 3:Análisis de Resultados y Elaboración de Conclusiones. En esta etapa, se evaluaron los resultados obtenidos durante el trabajo y desarrollo del sistema. Las conclusiones se basaron en el análisis de las encuestas, el progreso de las tareas en el tablero y la retroalimentación obtenida en reuniones periódicas.
- Fase 4: Elaboración del Informe Final del Trabajo. Esta fase incluyó la redacción del informe del trabajo final, documentando los recursos utilizados a lo largo del proceso.

CAPÍTULO II

Marco Teórico

1.7. Chatbots en educación y otros sectores

Un chatbot es una aplicación informática diseñada para simular una conversación con seres humanos a través de interfaces de texto o voz. Utiliza algoritmos de procesamiento de lenguaje natural (NLP) para comprender y responder a las consultas de los usuarios de manera coherente y relevante (Lalwani, et al., 2021). Los chatbots han evolucionado significativamente desde sus primeras versiones hasta convertirse en herramientas sofisticadas de comunicación digital. Su papel en la interacción hombre-máquina ha crecido exponencialmente, desde simples asistentes virtuales hasta plataformas capaces de gestionar tareas complejas y mantener conversaciones naturales con los usuarios (Vijayakumar, 2021).

Los chatbots se utilizan en diversos sectores. En el comercio, brindan atención al cliente, facilitan el proceso de compra, responden preguntas frecuentes y mejoran la experiencia del usuario, todo ello aliviando la carga de trabajo del personal humano. En sectores de servicios como la atención médica, los viajes y bancos, sus aplicaciones también son significativas (Chow, 2021). En el ámbito educativo cada vez se utilizan con mayor frecuencia, ayudando en el aprendizaje de materias específicas, guiando la adquisición de nuevas habilidades, respondiendo consultas y enriqueciendo la experiencia de aprendizaje general de los estudiantes (Han, 2021).

La importancia de los chatbots en educación radica en la personalización del aprendizaje, adaptándose a las necesidades individuales de los estudiantes, proporcionando materiales de apoyo y ejercicios específicos del tema. Esto promueve una educación más accesible, al proporcionar contenido y soporte personalizado a estudiantes con diversas necesidades y habilidades. Así como brindar retroalimentación inmediata y personalizada, mejorando la efectividad del aprendizaje (Han, 2021).

En contextos educativos, los chatbots pueden proveer funcionalidades como tutoría virtual, respondiendo preguntas, ofreciendo explicaciones a los estudiantes, generando contenido educativo como ejercicios, cuestionarios y material de estudio, adaptándolo a las necesidades y capacidades del estudiante, generando interacciones más naturales y

efectivas. Además, se puede utilizar chatbots como asistentes de aprendizaje, para ayudar a los estudiantes a aprender nuevos conceptos, resolver problemas y mejorar sus habilidades (Goya, et al., 2023). Por último, un chatbot puede realizar tareas administrativas en instituciones educativas, brindar información sobre inscripciones, programación de clases y gestión de recursos (Gago & Navajas, 2019).

1.8. Introducción a los Modelos de Lenguaje

1.8.1. Evolución de los chatbots y asistentes virtuales

Los chatbots han experimentado una evolución significativa, pasando de ser sistemas básicos basados en reglas a agentes conversacionales sofisticados impulsados por Inteligencia Artificial (IA) y aprendizaje profundo.

Los primeros chatbots, como Eliza (1966) y Parry (1972), usaban reglas predefinidas para simular conversaciones, sin comprender el contexto (Shum, et al. 2018).

Con la incorporación de procesamiento de lenguaje natural y modelos de aprendizaje automático, surgieron chatbots como Alice (1995) y Cleverbot (2008), que lograron interacciones más naturales, pero aún limitadas en comprensión (Gupta et al., 2020)

La llegada de técnicas de aprendizaje profundo y modelos avanzados de NLP revolucionó la tecnología de chatbots.

Agentes conversacionales como Siri, Alexa, Google Assistant y Microsoft Cortana permitieron interacciones contextuales más fluidas y la integración en dispositivos cotidianos.

En esta etapa, los chatbots evolucionaron para comprender el contexto, la intención y el sentimiento del usuario, generando respuestas naturales y coherentes.

La IA conversacional integra tecnologías como IA, NLP, aprendizaje automático, reconocimiento de voz y más, permitiendo que los sistemas comprendan, procesen y

generen lenguaje humano. Esto los habilita para interactuar de manera natural a través de texto, voz o incluso gestos (Hoy, 2018).

La IA conversacional ha ampliado enormemente el alcance de los chatbots, especialmente en la educación, con ejemplos como:

Asistentes de aprendizaje personalizados: Khanmigo (Khan Academy) y MATHia (Carnegie Learning).

Generación de contenido: Kahoot (generador de preguntas con IA).

Práctica de idiomas: Herramientas como Grammarly, HelloTalk y Elsa Speak (Panagiotidis, 2024).

1.8.2. Modelos de lenguaje tradicionales vs. modernos

Los modelos de lenguaje tradicionales, como los basados en n-gramas o en reglas, tenían limitaciones significativas en su capacidad para manejar la complejidad del lenguaje humano. Estos modelos a menudo requerían una extensa programación manual y no podían generalizar bien a nuevas situaciones o contextos. Su enfoque en patrones fijos significaba que eran incapaces de entender el contexto o la ambigüedad del lenguaje (Brown et al., 1992).

En contraste, los modelos de lenguaje modernos, especialmente los LLMs como GPT y BERT, utilizan un pre-entrenamiento generativo y ajuste fino discriminativo (“fine-tunning”) para mejorar la comprensión del lenguaje natural. Estos ayudan a clasificar textos, como identificar si una frase es correcta o determinar el sentimiento (positivo o negativo) de un texto. También son muy buenos para entender y razonar, por ejemplo, completando historias con lógica o respondiendo preguntas con precisión. Además, pueden reconocer si dos frases dicen lo mismo con palabras diferentes y entender la relación entre ideas en diferentes oraciones. Por último, son muy útiles para traducir idiomas, haciéndolo mejor que otros métodos (Alec et al. 2018).

1.9. Large Language Models (LLMs)

1.9.1. Definición y características de LLM

Los Modelos de Lenguaje a Gran Escala (LLMs, por sus siglas en inglés) son modelos de inteligencia artificial diseñados para comprender y generar texto en lenguaje natural. Se entrenan utilizando enormes cantidades de datos textuales, que pueden incluir libros, artículos, sitios web y otros tipos de contenido escrito. Este entrenamiento permite a los LLMs aprender patrones, estructuras y contextos del lenguaje, lo que les permite generar respuestas coherentes y contextualmente relevantes a partir de un prompt o entrada dada (Kasneci et al., 2023).

1.9.2. Arquitectura y funcionamiento

Los LLMs se basan en arquitecturas de redes neuronales profundas, siendo los Transformers una de las más prominentes. Esta arquitectura utiliza un mecanismo de atención, que permite al modelo enfocarse en diferentes partes de la entrada de texto de manera dinámica, facilitando la captura de dependencias a largo plazo en el lenguaje. Modelos como GPT (Generative Pre-trained Transformer) y BERT (Bidirectional Encoder Representations from Transformers) son ejemplos destacados de LLMs que utilizan esta arquitectura. GPT se centra en la generación de texto, mientras que BERT se utiliza principalmente para tareas de comprensión del lenguaje (Chang et al., 2024).

1.9.3. Capacidades y limitaciones de LLM

Los LLMs tienen muchas capacidades que los hacen útiles en distintas áreas. Una de sus habilidades más destacadas es la generación de texto, donde pueden crear contenido como artículos, resúmenes y descripciones que parecen escritos por humanos. Se destacan en la comprensión de preguntas, la interpretación de intenciones y el reconocimiento de nombres

o conceptos dentro del texto, lo que los hace especialmente adecuados para asistentes virtuales y chatbots.

Además, facilitan la traducción entre idiomas de forma instantánea, ayudando en la comunicación global. Su capacidad para analizar el tono y las emociones en el texto es beneficiosa para medir las opiniones en las redes sociales o en las encuestas. La adaptabilidad de los LLM a necesidades específicas mejora su eficacia en contextos personalizados, proporcionando respuestas más relevantes.

También pueden automatizar tareas como la creación de informes o contenido educativo, ahorrando tiempo en varias industrias. Asimismo, mejoran la interacción entre humanos y máquinas, haciendo que las conversaciones sean más fluidas, y pueden interpretar datos complejos para generar informes claros y comprensibles (Karanikolas et al. 2023).

A pesar de las capacidades avanzadas de modelos como GPT-4, aún enfrentan varios desafíos. Un problema importante es que pueden cometer errores al comprender preguntas o generar respuestas, especialmente en tareas complejas, donde no siempre logran un rendimiento consistente, dejando un margen de error considerable.

Se han identificado diferentes tipos de errores, incluida las dificultades para interpretar preguntas con suposiciones o manipular texto de forma incorrecta. Estos errores no solo afectan a GPT-4, sino también a otros modelos como Claude-3 y Llama-3, lo que sugiere que estas limitaciones podrían estar relacionadas con la forma en que los LLMs están diseñados o entrenados.

Además, aunque el fine-tuning puede mejorar algunos aspectos, no siempre corrige los problemas de fondo. Esto indica que ciertos errores son "intrínsecos" a los modelos y difíciles de eliminar. Por último, los LLMs tienen un conocimiento limitado comparado con los humanos y a menudo carecen de autoconciencia o la capacidad de alinear perfectamente sus respuestas con las intenciones humanas (Chen et al., 2024).

1.10. Aplicación de LLM en Chatbots

1.10.1. Cómo los LLM mejoran la interacción en chatbots educativos

La gran capacidad de los LLM para el Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) tiene el potencial de transformar el aprendizaje al proporcionar herramientas personalizadas y adaptativas. Ejemplos de estas aplicaciones incluyen la calificación automática de tareas, la generación de preguntas y herramientas de resumen, que ayudan a mejorar la comprensión y el compromiso de los estudiantes. Estas tecnologías permiten que el contenido educativo se adapte a los niveles de habilidad individuales, reduciendo la carga cognitiva asociada con el procesamiento de grandes volúmenes de material.

Los LLMs destacan en la personalización del aprendizaje, ya que son capaces de generar contenido educativo ajustado a las necesidades específicas de cada estudiante. Además, mejoran la accesibilidad para personas con discapacidades, facilitando un aprendizaje más inclusivo y equitativo.

En el ámbito del apoyo docente, estos modelos son herramientas útiles para la planificación de lecciones y la automatización de tareas administrativas. Esto permite a los maestros centrarse en actividades más importantes, como desarrollar habilidades críticas y guiar a los estudiantes en el pensamiento analítico.

Por otro lado, los chatbots educativos se han implementado en portales académicos, ofreciendo respuestas instantáneas, guías personalizadas y acceso rápido a recursos académicos y administrativos. Un ejemplo destacado es Harvard, donde los chatbots han optimizado la eficiencia de los servicios estudiantiles y mejorado la experiencia educativa.

Finalmente, la integración en cursos muestra el impacto práctico de los LLMs. En el curso CS50 de Harvard, se utilizaron herramientas basadas en OpenAI GPT-4, incluyendo explicaciones de código, mejora de estilo de programación y un chatbot que actuaba como tutor virtual. Estas herramientas fueron altamente valoradas por los estudiantes, quienes destacaron la sensación de tener un tutor personal, mejorando significativamente su experiencia de aprendizaje (Akram Sawiras, 2024).

1.10.2. Tutoría mejorada mediante LLM

Los modelos de lenguaje como los LLMs están transformando la forma de dar tutorías al ofrecer soluciones personalizadas y efectivas. Estas herramientas pueden analizar el progreso de los estudiantes, identificar áreas de dificultad y adaptar las sesiones de aprendizaje a las necesidades individuales, lo que mejora la calidad de la enseñanza. Además, los LLMs pueden generar explicaciones claras, responder preguntas de manera inmediata y sugerir estrategias para resolver problemas. Esto permite que las tutorías sean más dinámicas y accesibles, brindando apoyo constante sin depender de la disponibilidad de un tutor humano.

Otra ventaja es la capacidad de los LLMs para fomentar el pensamiento crítico, ofreciendo preguntas abiertas y retos que estimulan el aprendizaje activo. También facilitan la comunicación al adaptarse al nivel de comprensión del estudiante, haciendo que la experiencia sea más interactiva y efectiva.

Por último, estas herramientas permiten escalar las tutorías, haciendo posible que más estudiantes reciban apoyo simultáneamente. Gracias a su flexibilidad, los LLMs no solo automatizan tareas repetitivas, sino que también ofrecen un aprendizaje más personalizado, mejorando los resultados educativos de manera significativa (Jansson et al, 2024).

1.10.3. Generación de contenido educativo y retroalimentación

Los modelos de lenguaje (LLM) modifican la forma en que se crea contenido educativo y se proporciona retroalimentación personalizada. Estas herramientas pueden evaluar las respuestas de los estudiantes a preguntas abiertas y generar comentarios detallados y específicos basados en criterios definidos. Esto no solo permite a los estudiantes identificar áreas de mejora de manera eficiente, sino que también reciben orientación adaptada a su nivel de comprensión.

Además, los LLM ayudan a reducir los sesgos en la evaluación, proporcionando una retroalimentación más equitativa y consistente para todos los estudiantes. También facilitan el diseño de preguntas y criterios de evaluación, aliviando la carga de trabajo de los instructores, especialmente en clases numerosas. Con estas capacidades, los LLM promueven un ciclo de aprendizaje más efectivo, ofreciendo a estudiantes e instructores herramientas que mejoran la calidad de la enseñanza y del aprendizaje (Matelsky et al, 2023).

1.11. Beneficios de los LLM en la Educación

1.11.1. Aprendizaje personalizado y adaptativo

Los chatbots impulsados por modelos de lenguaje grande (LLMs) permiten personalizar y adaptar el aprendizaje según las necesidades específicas de cada estudiante. Estos sistemas pueden analizar el progreso individual, identificar áreas de mejora y ofrecer contenido educativo personalizado. Además, proporcionan retroalimentación inmediata, ajustando dinámicamente las respuestas y actividades en función del nivel de conocimiento del usuario. Este enfoque fomenta un aprendizaje más eficiente, ya que permite a los estudiantes enfocarse en áreas que requieren mayor atención mientras progresan a su propio ritmo.

Accesibilidad y escalabilidad

Los chatbots basados en LLMs también abordan los desafíos de accesibilidad y escalabilidad en entornos educativos. Al operar de manera continua y automatizada, estos sistemas garantizan soporte 24/7, permitiendo a los estudiantes acceder a ayuda en cualquier momento y lugar. Esto es especialmente valioso en cursos con grandes volúmenes de participantes, ya que los chatbots pueden gestionar consultas repetitivas y ofrecer asistencia organizativa o conceptual de manera eficiente. Además, su diseño flexible les permite adaptarse a diferentes disciplinas y contextos educativos, facilitando su implementación en diversas instituciones y escalando su uso sin comprometer la calidad del soporte (Neumann, 2024).

1.12. Desafíos en el uso de LLM en la educación

1.12.1. Sesgos y errores de los LLM

Los LLM, al ser entrenados con datos que reflejan las desigualdades sociales existentes, pueden desarrollar y perpetuar sesgos en función de la raza, género, discapacidad e ingresos de los estudiantes. Estos sesgos se manifiestan en diferentes formas, como al clasificar a los estudiantes o al generar contenido educativo (Weissburg et al. 2024).

También surgen sesgos lingüísticos, ya que ciertos idiomas están representados de forma más destacada en los datos de entrenamiento que otros. Esto genera errores al manejar idiomas con menor representación o variaciones regionales, lo que limita su precisión en contextos lingüísticos diversos. Por otro lado, los sesgos ideológicos y políticos hacen que los modelos puedan replicar las perspectivas predominantes en los datos de entrenamiento, perpetuando estereotipos y evitando ofrecer perspectivas neutrales en ciertos temas. En cuanto a los sesgos demográficos, algunos grupos específicos, ya sea por región o clase social, están subrepresentados en los datos. Esto dificulta que los modelos reflejen de manera precisa las realidades de estos grupos, generando una visión incompleta o sesgada. También existen los sesgos temporales, que reflejan las creencias y valores predominantes en el período en que se entrenaron los modelos. Esto puede causar problemas al abordar temas históricos o al interpretar cambios en los valores sociales a lo largo del tiempo. Finalmente, los sesgos de confirmación surgen porque los modelos tienden a priorizar respuestas atractivas para satisfacer al usuario. Esto puede llevar a distorsionar la calidad de la información al priorizar contenido que agrada, pero que no necesariamente es más preciso o informativo (Barriere, 2024).

1.12.2. Dependencia tecnológica y monitoreo

Supervisión necesaria: Es fundamental que las respuestas de los chatbots sean monitoreadas para asegurar que complementen el trabajo del docente y no lo reemplacen.

La interacción humana sigue siendo esencial en el proceso educativo, y los LLMs deben ser utilizados como herramientas de apoyo.

Dependencia tecnológica: La creciente dependencia de la tecnología puede llevar a una disminución en las habilidades de pensamiento crítico y resolución de problemas de los estudiantes si no se utiliza de manera equilibrada.

1.13. Implementación de LLM en el chatbot de tutoría

1.13.1. Integración con otras tecnologías

Los modelos como ChatGPT o Llama-3, pueden integrarse con otras tecnologías para mejorar la educación, especialmente en áreas como bases de datos. Estas integraciones permiten automatizar tareas y facilitar el aprendizaje de forma más efectiva.

Una de las principales integraciones es con las bases de datos relacionales y SQL (Structured Query Language). Los LLMs pueden generar y ejecutar consultas SQL automáticamente, ayudando a los estudiantes a trabajar con datos sin necesidad de configurar complicados entornos. Herramientas como DB-BERT, que usa inteligencia artificial para optimizar configuraciones de bases de datos, son un buen ejemplo de esto.

Otra tecnología clave es el RAG (Retrieval-Augmented Generation), que mejora las respuestas de los LLMs al combinar su generación de texto con información extraída de documentos externos. Esto asegura respuestas más precisas y útiles, especialmente en entornos educativos.

También destacan las representaciones visuales, como las generadas por herramientas como Data Tweening o MOCHA. Estas permiten a los estudiantes ver cómo se ejecutan las consultas en las bases de datos, explicando cada paso de manera gráfica y fácil de entender. Esto es especialmente útil para aprender conceptos técnicos de forma práctica.

Los LLMs pueden conectarse con motores de análisis de datos para analizar el trabajo de los estudiantes y proporcionar retroalimentación personalizada. Esto ayuda a cada estudiante a entender mejor sus errores y fortalezas.

Además, se utilizan frameworks pedagógicos como TPACK (Conocimiento Tecnológico, Pedagógico y del Contenido) y SAMR (Sustitución, Aumento, Modificación y

Redefinición), que guían cómo usar la tecnología para mejorar la enseñanza. Esto permite que los LLMs adapten ejercicios y contenido según los intereses o necesidades de los estudiantes.

Por último, existen técnicas avanzadas como ReAct (Razonamiento y Acción) y meta-prompting. Estas ayudan a los LLMs a razonar mejor y ajustar sus respuestas basándose en retroalimentación, reduciendo errores como las alucinaciones.

Con estas tecnologías, los LLMs pueden transformar el aprendizaje, haciéndolo más interactivo, visual y personalizado, adaptándose mejor a las necesidades de cada estudiante. (Prakash et al., 2024).

1.13.2. Métricas de éxito y evaluación del desempeño

El desempeño de los chatbots se puede medir utilizando varias métricas importantes. Una de ellas es la efectividad, que evalúa si el chatbot cumple bien con las tareas para las que fue diseñado. Esto incluye la calidad de sus respuestas y si están alineadas con los objetivos educativos.

Otra métrica es la eficiencia, que mide cómo el chatbot utiliza los recursos, como el tiempo y el esfuerzo, para ayudar a los usuarios. Por ejemplo, se analiza si responde rápido y si facilita procesos como buscar información o resolver tareas.

La satisfacción del usuario es clave, ya que refleja qué tan bien reciben los estudiantes el chatbot. Esto incluye si les parece fácil de usar, si disfrutan interactuar con él y si se sienten motivados a aprender gracias a la herramienta.

La escalabilidad también es importante. Se refiere a la capacidad del chatbot para atender a más usuarios o manejar más interacciones sin que disminuya su calidad o rendimiento.

Además, la usabilidad se enfoca en lo sencillo que es para los estudiantes interactuar con el chatbot. Esto abarca tener una interfaz fácil de entender y accesible desde diferentes dispositivos.

Por último, está el impacto en el aprendizaje, que mide si el chatbot realmente ayuda a los estudiantes a adquirir conocimientos y habilidades. Esto puede evaluarse con encuestas, análisis de resultados o comentarios escritos por los estudiantes.

Estas métricas permiten entender tanto la parte técnica del chatbot como su impacto en la experiencia y el aprendizaje de los estudiantes. (Vanichvasin, 2024).

Las métricas utilizadas en el proyecto se detallan en el Anexo IV.

1.14. Perspectivas futuras de los LLM en el ámbito educativo

1.14.1. Mejora continua en los LLM

Los LLMs están en constante evolución, y se anticipan desarrollos futuros que podrían mejorar aún más la experiencia educativa:

- **Mayor precisión y contextualización**: Con el avance de la tecnología, se espera que los LLMs sean capaces de generar respuestas aún más precisas y contextualizadas, mejorando la calidad de la tutoría.
- **Interacción multimodal**: La integración de capacidades multimodales (texto, voz, imágenes) podría enriquecer la experiencia de aprendizaje, permitiendo interacciones más dinámicas y efectivas (Chang et al., 2024)..

1.14.2. Potencial de los LLM en la enseñanza autónoma

Una de las principales ventajas de los LLM es la personalización del aprendizaje. Estos modelos pueden analizar materiales educativos y responder de manera específica según las dificultades o intereses de cada estudiante, permitiéndoles avanzar a su propio ritmo y enfocarse en lo que más necesitan.

Además, ofrecen accesibilidad mejorada al responder de forma clara y sencilla preguntas complejas. Esto los convierte en una herramienta valiosa para estudiantes que tienen acceso limitado a recursos educativos, facilitando el aprendizaje incluso en contextos desafiantes.

Otra ventaja significativa es su capacidad para fomentar una interacción natural. Los estudiantes pueden comunicarse fácilmente con las plataformas educativas, lo que hace que el proceso de aprendizaje sea más fluido, interactivo y motivador. Esto aumenta el interés y el compromiso de los estudiantes con sus estudios.

Además, los LLM se destacan en el análisis de contenido, ya que pueden procesar materiales educativos y documentos para identificar la información más importante. Esto ayuda a los estudiantes a ahorrar tiempo y a centrarse en lo que realmente importa.

Por último, los LLM fomentan la autonomía al proporcionar herramientas que empoderan a los estudiantes para aprender de forma independiente, otorgándoles un mayor control sobre su educación y promueven la exploración y el desarrollo de conocimientos por sí mismos (Micheal et al., 2024).

1.15. Marco Conceptual de RAG (Retrieval-Augmented Generation)

1.15.1. Definición y funcionamiento de RAG:

La generación aumentada por recuperación (RAG) es un método de procesamiento del lenguaje natural (NLP) que combina la recuperación de información con la generación de texto. A diferencia de los modelos generativos tradicionales que dependen únicamente de los datos con los que fueron entrenados, RAG integra información externa en tiempo real para mejorar la calidad y precisión de las respuestas generadas (Wu et al., 2024).

1.15.2. Características de RAG:

RAG es un enfoque avanzado para procesar lenguaje natural que combina diferentes técnicas para lograr resultados efectivos. Una de sus principales fortalezas es la combinación de enfoques, ya que combina la capacidad de generar respuestas usando solo su modelo interno (enfoque cerrado) con la ventaja de buscar información externa (enfoque abierto). Esto le permite trabajar sin depender de entrenamientos costosos.

Otra característica importante es su capacidad de generación “en abstracto”, lo que significa que no se limita a copiar respuestas de textos existentes. En su lugar, RAG puede crear respuestas en texto libre, formulando ideas completas y originales basadas en las preguntas que se le hacen.

Por último, RAG se caracteriza por su menor tasa de alucinación, lo que implica que produce respuestas más precisas y con menor probabilidad de contener información

errónea o ficticia. Además, proporciona una mayor diversidad en las respuestas, generando textos variados y evitando redundancias, lo que resulta en una notable efectividad en la recuperación de información y en la resolución de tareas de manera eficiente, como lo han demostrado estudios que evalúan el impacto de su sistema de búsqueda (Lewis et al., 2020).

1.15.3. Diferencias con otros enfoques generativos:

Además de las características mencionadas, RAG tiene capacidad para combinar memoria paramétrica y no paramétrica. Esto le permite acceder a información externa, como bases de datos actualizadas, algo que no es posible con modelos generativos tradicionales que dependen exclusivamente de su conocimiento interno.

Otra diferencia clave es el entrenamiento conjunto de su generador y su recuperador. Ambos componentes se optimizan mutuamente durante el proceso de aprendizaje, lo que mejora su interacción y efectividad. En otros enfoques, estos elementos suelen entrenarse por separado, lo que limita su rendimiento.

Además, RAG proporciona una mayor flexibilidad en la generación de texto, ya que puede utilizar diferentes pasajes de información para construir cada parte de una respuesta. Esto contrasta con enfoques más rígidos que dependen de un solo pasaje para toda la generación (Lewis et al., 2020).

1.15.4. Componentes de RAG

1.15.4.1. Módulo de recuperación (Retrieval)

Este módulo tiene la función de localizar información relevante en una base de datos externa. Su operación abarca la fragmentación de datos, que consiste en dividir los documentos en partes más pequeñas para optimizar la búsqueda. A continuación, se realiza la codificación, donde cada fragmento se transforma en representaciones numéricas que reflejan su significado semántico. Posteriormente, se lleva a cabo la indexación, que organiza estas representaciones en una base de datos, permitiendo búsquedas rápidas a

través de algoritmos como Approximate Nearest Neighbors. Finalmente, al recibir una consulta, el sistema identifica los fragmentos más relevantes en función de su similitud, proporcionando los datos necesarios para que el modelo genere respuestas (Wu et al., 2024).

1.15.4.2. Módulo de generación (Generation)

El generador es el encargado de crear la respuesta final a partir de la información recuperada y la consulta original del usuario. Después de que el módulo de recuperación encuentra datos relevantes en fuentes externas, el generador toma esos datos y los combina con la consulta para generar una respuesta clara y adecuada al contexto.

El LLM asegura que el texto generado sea fácil de entender, preciso y que responda correctamente a la pregunta original del usuario. Es decir, el generador transforma los datos recuperados en una respuesta coherente y útil para el usuario (Gupta et al., 2024). Este módulo utiliza la información integrada para generar respuestas finales.

Se pueden considerar diferentes tipos de generadores. Los generadores estándar son modelos pre entrenados, como GPT, que generan respuestas en función de la entrada y el contexto que se les proporciona. Por otro lado, los generadores con recuperación aumentada son modelos adaptados para interactuar de manera específica con la información recuperada, lo que optimiza la relevancia y la precisión de las respuestas (Wu et al., 2024).

1.15.5. Recuperación de Información en RAG

1.15.5.1. Representación de documentos

La representación de documentos es clave en sistemas que buscan información o generan texto automáticamente. Esto se logra transformando los textos en números que reflejan su significado usando dos elementos principales:

Embeddings: son transformaciones matemáticas deterministas de texto, no generan contenido nuevo, sino que crean un contenido fijo, representaciones numéricas de texto existente. Estos vectores capturan relaciones entre palabras y su significado en diferentes contextos, lo que ayuda a encontrar y comparar documentos similares (Perron et al. 2024).

Modelos de Lenguaje: son útiles para capturar los aspectos estructurales de los lenguajes (Petukhova et al. 2024) así como el significado semántico de las palabras, oraciones y documentos. También ayudan a comprender el contexto y mejorar el rendimiento en tareas de comprensión del lenguaje natural (Rathod, 2024).

1.15.5.2. Métodos de indexación y búsqueda

La indexación y búsqueda son fundamentales para manejar grandes volúmenes de datos y permitir búsquedas eficientes. Existen varias herramientas y técnicas que se utilizan para este propósito:

- FAISS (Facebook AI Similarity Search): es una biblioteca para gestionar una base de datos de vectores en un sistema RAG. Esta herramienta facilita la organización y búsqueda de representaciones numéricas de texto o embeddings, de forma rápida y eficiente, estas son generadas por un modelo que transforma fragmentos de texto en vectores numéricos que representan su significado.

  En un sistema RAG, FAISS permite identificar los vecinos más cercanos, encontrando rápidamente los fragmentos de texto más parecidos o relevantes para una consulta, basándose en sus vectores. Esto no solo reduce el tiempo de respuesta al evitar la búsqueda exhaustiva en todos los documentos, sino que también mejora el rendimiento general del sistema, permitiendo manejar grandes volúmenes de datos sin comprometer la eficiencia (Makin et al., 2024).
- Elasticsearch: El sistema RAG tradicional utiliza métodos basados en la coincidencia de palabras clave o la similitud semántica para recuperar documentos. Aunque estos métodos son efectivos en algunos casos, tienen limitaciones al procesar consultas complejas, mejorar la cantidad de documentos relevantes

recuperados y asegurar la diversidad en los resultados. Estas dificultades pueden reducir la efectividad del sistema en aplicaciones más complejas.

Elasticsearch es un motor de búsqueda y análisis de código abierto que admite búsquedas de texto completo, agregaciones avanzadas y soporte para diferentes tipos de datos, como texto y números. Su flexibilidad permite definir consultas más complejas y personalizadas, mientras que sus capacidades de ordenamiento y alto rendimiento mejoran la eficiencia del sistema.

La integración de Elasticsearch con RAG permite superar las limitaciones de los métodos tradicionales, ofreciendo mejores resultados en términos de relevancia y diversidad. Esto ha llevado a su aplicación en áreas como la detección de fraudes crediticios y la gestión de riesgos, donde la recuperación eficiente y precisa de información es esencial (Chen et al., 2024).

- Otras Herramientas: Existen otras herramientas y bibliotecas, como Annoy y Milvus (Alboaie, 2024), que también se utilizan para la indexación y búsqueda de vectores. Estas herramientas están diseñadas para optimizar la búsqueda de similitud en grandes conjuntos de datos, utilizando diferentes algoritmos y estructuras de datos para mejorar la eficiencia (Prabhune y Berndt, 2024).

1.15.5.3. Medidas de similitud

La precisión del sistema de recuperación depende en gran medida de la efectividad de las medidas de similitud. Una buena métrica de similitud asegura que los documentos más relevantes se recuperen en respuesta a una consulta, lo que mejora la experiencia del usuario y la utilidad del sistema en aplicaciones prácticas (García, 2015).

La medición de la similitud entre consultas y documentos es esencial para la precisión de los sistemas de recuperación de información.

Métricas que se utilizan para evaluar esta similitud:

Cosine Similarity: Esta métrica mide el coseno del ángulo entre dos vectores en un espacio de alta dimensión. Es ampliamente utilizada en la recuperación de información porque es

efectiva para medir la similitud entre documentos representados como embeddings. Un valor de 1 indica que los vectores son idénticos, mientras que un valor de 0 indica que son ortogonales (sin similitud) (Saado, 2024)

Jaccard Similarity: Esta métrica se utiliza para medir la similitud entre conjuntos. Se calcula como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos. Es útil en contextos donde se comparan conjuntos de palabras o características (Zadeh y Goel, 2013).

1.15.6. Generación de respuestas en RAG

La RAG combina la recuperación de información con modelos generativos para producir respuestas precisas y relevantes a las consultas de los usuarios. A continuación, se detallan los aspectos clave de este enfoque.

1.15.6.1. Modelos generativos

1.15.6.1.1. Ejemplos de Modelos de Lenguaje:

GPT-1, GPT-2, GPT-3 y GPT-4: Desarrollados por OpenAI, estos modelos son ejemplos de modelos de lenguaje generativos que utilizan arquitecturas de transformadores. Cada versión ha mejorado en términos de tamaño y capacidad de generación de texto.

BERT (Bidirectional Encoder Representations from Transformers): Introducido por Google, BERT es un modelo basado en transformadores que se centra en la comprensión del lenguaje natural mediante el uso de atención bidireccional.

Codex: Un modelo de OpenAI que se especializa en la generación de código, mostrando habilidades avanzadas en la comprensión y creación de código en varios lenguajes de programación.

LLaMA (Large Language Model Meta AI): Un modelo de lenguaje de código abierto que se centra en la eficiencia y la accesibilidad, permitiendo a los investigadores y desarrolladores trabajar con modelos de lenguaje de gran escala (Kalyan, 2024).

1.15.6.2. Mecanismo de combinación de recuperación y generación

El sistema RAG integra dos procesos fundamentales para optimizar la calidad de las respuestas: la búsqueda de información relevante y la creación de texto contextualizado. Este proceso implica la localización de datos relevantes en una base de datos o en un conjunto de documentos, utilizando técnicas de similitud semántica que facilitan la identificación de fragmentos de texto que se ajustan mejor a la consulta del usuario. Tras la recuperación de los documentos más significativos, se procede a una reordenación que prioriza los fragmentos más útiles, evitando así la saturación de información y permitiendo que el sistema se concentre en los datos esenciales, lo que mejora la precisión de la respuesta final. Posteriormente, la información recuperada se integra con la consulta original del usuario, proporcionando al modelo generativo un contexto más completo para generar respuestas más precisas y detalladas.

Este proceso se divide en dos etapas: la contextualización, donde el modelo no solo genera respuestas basadas en patrones aprendidos, sino que también incorpora la información recuperada para ofrecer respuestas más ajustadas al contexto, lo que permite abordar preguntas complejas de manera más informada y evitar respuestas incorrectas o poco útiles; y el ajuste de respuestas, en el que el sistema puede perfeccionar la respuesta generada para aumentar su claridad, precisión y relevancia, lo que puede incluir la reescritura de partes del texto, la adición de detalles o la adaptación de la respuesta al estilo de comunicación esperado por el usuario. Este mecanismo permite que RAG produzca respuestas más completas y confiables, integrando de manera efectiva información externa para enriquecer la experiencia del usuario (Gao et al, 2023).

1.15.6.3. Ajustes y personalización

Para que el modelo generativo produzca respuestas en el tono y enfoque adecuados, es posible implementar diversas estrategias. Una de ellas es el entrenamiento en conjuntos de datos específicos que reflejan el estilo y tono deseados, lo cual resulta particularmente beneficioso en contextos que requieren un lenguaje formal, técnico o coloquial. Asimismo, al interactuar con el modelo, se pueden ofrecer instrucciones precisas que orienten la generación de texto hacia el estilo y contenido esperado, facilitando así que las respuestas se alineen con las expectativas del modelo. Además, los sistemas RAG pueden integrar información sobre el usuario, como su historial de interacciones o preferencias, lo que permite una personalización más efectiva de las respuestas, aumentando así la relevancia y satisfacción del usuario durante la interacción con el sistema (Perak et al., 2024).

1.15.7. Desafios tecnicos

1.15.7.1. Sesgo en los datos recuperados

Los sistemas RAG dependen de bases de datos y conjuntos de información que pueden contener sesgos inherentes. Si los datos recuperados reflejan prejuicios o estereotipos, esto puede llevar a que el modelo genere respuestas sesgadas o inapropiadas. Este problema es crítico, ya que puede afectar la equidad y la justicia en las interacciones con los usuarios (Ortiz de Zárate et al., 2024).

1.15.7.2. Costo computacional

La combinación de recuperación de información y generación de texto puede ser intensiva en recursos. Los modelos generativos, especialmente los de gran escala como GPT-3 y GPT-4, requieren una cantidad significativa de potencia computacional para funcionar eficientemente. Esto puede resultar en altos costos operativos, especialmente en aplicaciones a gran escala donde se manejan múltiples consultas simultáneamente (Zhang et al., 2021).

1.15.7.3. Problemas de escalabilidad

A medida que aumenta la cantidad de datos en las bases de conocimiento y se complejizan las demandas de las aplicaciones, la escalabilidad de los sistemas RAG enfrenta desafíos significativos.

El enfoque tradicional de "recuperar y luego generar" está mostrando limitaciones para manejar eficientemente grandes volúmenes de información. La integración de tecnologías avanzadas, como recuperadores optimizados y modelos de lenguaje sofisticados, ha incrementado la complejidad de estos sistemas, lo que puede afectar negativamente los tiempos de respuesta y la precisión en la recuperación de información relevante.

A medida que la cantidad de datos continúa creciendo, es fundamental implementar estrategias de optimización y diseño modular para garantizar que los sistemas RAG puedan escalar de manera efectiva sin comprometer su rendimiento o utilidad. Este enfoque no solo mejora la gestión de datos a gran escala, sino que también establece una base sólida para la evolución continua y el despliegue práctico de tecnologías RAG (Gao et al. 2024).

1.15.8. Consideraciones éticas

1.15.8.1. Privacidad de Datos

Los sistemas RAG buscan mejorar las respuestas de los modelos de lenguaje al incorporar información relevante extraída de fuentes de conocimiento externas. Uno de los riesgos más significativos radica en la posibilidad de que las consultas de los usuarios puedan extraer datos sensibles de las bases de conocimiento utilizadas. Los ataques dirigidos y no dirigidos pueden exponer información confidencial, incluidos fragmentos de texto originales o detalles privados. Este tipo de vulnerabilidades es especialmente preocupante en dominios sensibles, como el de la salud, donde un chatbot médico que accede a historiales clínicos podría filtrar información sobre pacientes.

La necesidad de proteger la privacidad en los sistemas RAG es crítica para garantizar su adopción segura en contextos sensibles. Medidas como el uso de datos sintéticos, estrategias avanzadas de encriptación y mecanismos de anonimización son esenciales para prevenir el acceso no autorizado y mitigar los riesgos de exposición de información (Zeng et al. 2024).

1.15.8.2. Sesgos en las respuestas

Los sistemas de inteligencia artificial generativa, incluidos los modelos de texto a imagen y Retrieval-Augmented Generation (RAG), han demostrado capacidades impresionantes, pero también plantean preocupaciones éticas significativas relacionadas con los sesgos. Estos modelos tienden a reflejar y amplificar los prejuicios presentes en sus datos de entrenamiento, lo que puede perpetuar estereotipos y afectar negativamente a ciertos grupos demográficos.

Por ejemplo, los modelos generativos pueden asociar profesiones o características físicas con estereotipos de género, edad o tono de piel. Este tipo de sesgo no solo reduce la equidad en las representaciones generadas, sino que también puede limitar la confiabilidad y aceptación de estas tecnologías en aplicaciones críticas.

Para mitigar este problema, es necesario implementar mecanismos que mejoren la imparcialidad en la recuperación y generación de información. Soluciones como FairRAG destacan por utilizar fuentes de datos diversificadas y estrategias de post-hoc debiasing, las cuales son más eficientes y extensibles que la modificación completa del modelo o su reentrenamiento. Estas estrategias no solo mejoran la equidad en las respuestas, sino que también proporcionan mayor transparencia y control en el proceso de generación (Shrestha et al. 2024).

1.15.8.3. Transparencia

La transparencia en el funcionamiento del chatbot es clave para construir confianza con los usuarios. Los usuarios deben ser informados sobre cómo se generan las respuestas, qué datos se utilizan y cómo se manejan sus interacciones. Esto no solo fomenta la confianza, sino que también permite a los usuarios tomar decisiones informadas sobre su interacción con el sistema (Wu et al., 2024).

1.15.8.4. Limitaciones del Sistema RAG

Los enfoques actuales de RAG han demostrado ser prometedores en diversos ámbitos, pero presentan limitaciones significativas que dificultan su implementación en empresas, especialmente en sectores regulados. En primer lugar, los sistemas RAG carecen de un control exhaustivo sobre los procesos de recuperación y generación de información, lo que resulta problemático en industrias donde la precisión y el cumplimiento normativo son esenciales para evitar errores y asegurar la calidad. Además, enfrentan retos en términos de escalabilidad y rendimiento, ya que tienen dificultades para gestionar grandes y diversas bases de conocimiento empresarial, lo que restringe su eficiencia a gran escala. Por otro lado, la falta de explicabilidad y auditabilidad en los resultados de estos sistemas es un obstáculo importante, ya que es fundamental generar confianza y responsabilidad en aplicaciones empresariales críticas, como la toma de decisiones o situaciones legales. Finalmente, la integración de las capacidades de RAG en los sistemas y flujos de trabajo existentes se complica debido a los complejos requisitos de seguridad y gestión de datos que deben cumplirse en el entorno empresarial (Bruckhaus, 2024).

1.15.9. Aplicaciones y Casos de Estudio de Chatbots Basados en RAG

1.15.9.1. Aplicaciones prácticas

En respuesta a una consulta de un cliente, un sistema basado en RAG y LLM recupera documentos relevantes de conocimiento y los utiliza, junto con el historial de chat previo,

para generar sugerencias de respuesta destinadas a los agentes de servicio al cliente en los centros de contacto de una importante empresa minorista, lo que pueden ser de gran para los representantes de servicio al cliente, aliviando su carga de trabajo (Veturi et al. 2024).

Los LLM tienen múltiples aplicaciones en el ámbito educativo, donde su influencia en el aprendizaje es notable. Un ejemplo es TutorBot+, una herramienta diseñada para mejorar el aprendizaje de programación. Este sistema, integrado exitosamente en un LMS, proporciona retroalimentación personalizada y ayuda a desarrollar habilidades de pensamiento computacional en estudiantes. Otro ejemplo es NewtBot, un chatbot educativo que ha recibido valoraciones positivas por parte de los estudiantes. Este sistema utiliza estrategias avanzadas de personalización mediante ingeniería de prompts, lo que mejora su comportamiento y fomenta un mayor compromiso académico (Modran et al. 2024).

Health-LLM, un marco de trabajo diseñado para mejorar la gestión de la salud y la predicción de enfermedades como la personalización del cuidado a los pacientes utilizando LLM (Jin et al. 2024).

También es posible aplicar el enfoque de LLM con RAG a la creación de asistentes legales para la resolución de conflictos, aunque de momento con resultados limitados e impacto marginal (Rafat, 2024).

1.15.9.2. Casos de estudio relevantes

Google emplea RAG en sus sistemas de Pregunta-Respuesta para ofrecer respuestas más precisas e informativas mediante la recuperación de documentos pertinentes de grandes corpus. El modelo RAG de Facebook, diseñado para QA de dominio abierto, utiliza Dense Passage Retrieval (DPR) y BART para mejorar significativamente la calidad de las respuestas y la relevancia contextual. Zendesk mejora sus chatbots de atención al cliente con modelos basados en RAG, recuperando artículos relevantes de la base de conocimiento para proporcionar respuestas instantáneas y personalizadas, lo que reduce la dependencia

de agentes humanos y mejora la satisfacción del cliente. Los asistentes virtuales de IBM Watson utilizan técnicas similares para ofrecer respuestas detalladas y contextualmente apropiadas, mejorando tanto la satisfacción del cliente como la eficiencia del soporte. Copy.ai utiliza RAG para ayudar a los vendedores y escritores a generar contenido de alta calidad al recuperar datos relevantes de grandes corpus, produciendo materiales de marketing bien informados y contextualmente precisos (Tajz, 2024).

1.15.10. Perspectivas Futuras y Avances en Chatbots RAG

1.15.10.1. Innovaciones en la Recuperación de Información

Técnicas de Embeddings:

Embeddings de oraciones: Sentence-BERT, una modificación de la red pre entrenada BERT utiliza estructuras de red doble o triple para realizar embedding de oraciones semánticamente significativas que se pueden comparar mediante similitud de coseno (Reimers, 2019).

Los modelos de embeddings contextuales generan representaciones vectoriales dinámicas para palabras u oraciones, donde los embeddings cambian dependiendo del contexto que las rodea. Estos modelos aprovechan arquitecturas de aprendizaje profundo, como los transformers, para obtener representaciones semánticas más ricas. Este tipo de embeddings, permiten entender mejor el lenguaje humano, capturando tanto el significado literal como las relaciones implícitas entre palabras. Esto mejora el desempeño en tareas como análisis de sentimientos, traducción automática, búsqueda semántica, y más (Ghorbani, 2024).

El uso de embeddings de múltiples modalidades, que combinan texto, imagen y audio, permite a los sistemas de recuperación de información abordar una mayor diversidad de datos y enriquecer la calidad de las respuestas generadas. Este enfoque se inspira en la forma en que los humanos procesan y conectan información verbal y no verbal, como texto, imágenes y sonidos. Al integrar estas modalidades, los sistemas no solo mejoran su

capacidad para interpretar datos complejos, sino que también ofrecen respuestas más precisas y contextualizadas en tareas multimodales.

Avances recientes han demostrado que esta integración puede aplicarse de manera efectiva a una amplia gama de tareas, desde la recuperación de información hasta la clasificación de eventos, mostrando el potencial transformador de los modelos multimodales en aplicaciones prácticas (Yue et al. 2024).

Redes Neuronales Avanzadas:

Las redes neuronales convolucionales (CNN) son un tipo de modelo de inteligencia artificial diseñado especialmente para trabajar con imágenes. Una de sus características principales son las capas convolucionales, que usan filtros para detectar patrones como bordes, texturas y formas en las imágenes, permitiendo a la red aprender automáticamente cómo analizar su contenido. Además, las capas de agrupamiento (pooling) ayudan a reducir la cantidad de datos procesados al mantener solo la información más importante, lo que hace que el modelo sea más rápido y eficiente.

Al final de la red, las capas completamente conectadas toman las características extraídas y realizan predicciones, como clasificar una imagen en una categoría específica. Las CNN también son muy eficientes porque comparten pesos entre sus filtros, lo que reduce la cantidad de parámetros y mejora su capacidad para generalizar en diferentes conjuntos de datos. Además, estas redes no necesitan que los humanos diseñen manualmente las características que deben aprender, ya que lo hacen automáticamente durante el entrenamiento.

En cuanto a sus usos, las CNN son fundamentales en la clasificación de imágenes, como identificar si una foto es de un gato o un perro. También se utilizan en la detección de objetos, por ejemplo, para que los autos autónomos reconozcan peatones y señales de tráfico. En el campo médico, las CNN son útiles para analizar imágenes como radiografías y detectar problemas de salud, como tumores. Otros usos comunes incluyen el

reconocimiento facial, que se aplica en sistemas de seguridad, y la segmentación de imágenes, que permite dividirlas en partes importantes para un análisis más detallado.

Gracias a su capacidad para procesar grandes cantidades de datos de manera eficiente, las CNN son una herramienta clave en muchas aplicaciones modernas de visión por computadora (Ramdurai, 2025).

Las Redes Neuronales Recurrentes (RNN) están diseñadas para trabajar con datos secuenciales, como texto, audio o series temporales, gracias a su capacidad de procesar entradas en orden y recordar información previa. Esto se logra mediante conexiones recurrentes, que permiten retroalimentar información para capturar dependencias en el tiempo. Sin embargo, estas redes tienen limitaciones, como problemas de desvanecimiento del gradiente, que dificultan el aprendizaje de relaciones a largo plazo. Para resolver esto, se han desarrollado variantes como las LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit), que mejoran su rendimiento.

Las RNN se utilizan en tareas como el procesamiento del lenguaje natural, incluyendo la traducción automática, el análisis de sentimientos y la generación de texto. También son clave en el reconocimiento de voz, convirtiendo audio en texto, y en el análisis de series temporales, como predecir precios o monitorear sensores. Además, estas redes se aplican en campos creativos como la composición musical y el arte generativo, aprendiendo patrones de estilos existentes. Es decir, las RNN son herramientas versátiles que permiten capturar dependencias contextuales en datos secuenciales, haciéndolas esenciales para una amplia gama de aplicaciones en inteligencia artificial, aunque su entrenamiento puede ser más complejo y lento que el de otras arquitecturas (Ramdurai, 2025).

1.15.10.2. Nuevas Arquitecturas Generativas

Modelos Emergentes:

HybridRAG, es un marco de trabajo híbrido de generación aumentada por recuperación diseñado para mejorar tareas que requieren respuestas en tiempo real, como la asistencia en composición (sugerir la siguiente palabra o frase antes de que se escriba). Este enfoque combina modelos en la nube y en el cliente (Zhang et al. 2023).

KARAGEN, Knowledge Augmented Retrieval And Generation Engine es un enfoque que busca combinar grafos de conocimiento con LLM para así acelerar la búsqueda de información en textos de investigación (Kaplan et al. 2024).

REAR, Relevance-Aware Retrieval-Augmented, un enfoque diseñado para mejorar los sistemas de RAG en tareas de preguntas y respuestas de dominio abierto (Wang et al. 2024).

Integración con Otras Tecnologías

Visión por Computadora:

El uso de sistemas RAG en el análisis de imágenes combina la recuperación de información con el procesamiento visual para ofrecer respuestas más precisas y útiles. Este enfoque comienza recuperando datos relevantes de bases de datos externas, como especificaciones o artículos relacionados. Luego, integra esta información con el análisis de características visuales de la imagen, como forma, color o marca, utilizando herramientas como redes neuronales convolucionales. Finalmente, genera respuestas completas y contextualizadas, por ejemplo, identificando un producto en una imagen y proporcionando detalles sobre él.

RAG destaca por mejorar la precisión al usar datos actualizados, ofrecer respuestas personalizadas basadas en imágenes y texto, y automatizar tareas como el soporte al cliente. Esto lo hace útil en aplicaciones como e-commerce, donde ayuda a los clientes a encontrar productos similares, o en salud, para analizar imágenes médicas y brindar información relevante (Ramdurai, 2025).

Procesamiento de Voz:

El procesamiento de voz es un componente esencial en sistemas modernos que buscan mantener interacciones naturales con los usuarios.

Uno de los elementos clave es la detección de actividad vocal (VAD), que identifica con precisión cuándo un usuario comienza a hablar y cuándo termina. Esto permite que el sistema gestione los turnos de conversación de manera efectiva, asegurando que las interacciones sean fluidas y naturales.

Los sistemas pueden integrar tecnologías de Reconocimiento de Voz (STT) para convertir el habla en texto de manera continua durante una conversación. Por otro lado, la tecnología de Texto a Voz (TTS) permite convertir texto escrito en audio, proporcionando respuestas auditivas al usuario. Los modelos modernos de TTS, como los basados en aprendizaje profundo, ofrecen voces más naturales y expresivas. Esto permite aplicaciones que van desde asistentes virtuales hasta ayudas para personas con discapacidad visual (Salamea-Lopez, 2024).

CAPÍTULO III

Desarrollo de software

1.16. Introducción

En este documento se detalla el desarrollo de un chatbot basado en RAG, que tiene como objetivo ayudar a estudiantes de la cátedra de Informática en la Facultad de Tecnología y Ciencias Aplicadas de la Universidad Nacional de Catamarca (UNCa). Se utilizó la metodología Kanban para organizar y gestionar las tareas de forma visual y continua. A lo largo del desarrollo se abordaron las siguientes fases clave: requerimientos, diseño, implementación, pruebas y análisis del feedback.

1.17. Análisis y requerimientos

La primera fase consistió en recopilar y analizar las necesidades de la cátedra de Informática, enfocándonos en mejorar el aprendizaje de la programación en Lenguaje C.

Objetivos del sistema: Resolver las dudas de los estudiantes de manera eficiente y proporcionar herramientas que contribuyan al aprendizaje del lenguaje de programación C.

Problemas identificados: Escaso tiempo de práctica en la asignatura, dificultad para entender temas complejos como punteros, y diversidad de niveles de conocimiento entre los estudiantes.

Historias de usuario: A partir de estas necesidades, se definieron historias de usuario que reflejaban los objetivos y funcionalidades requeridas, las cuales se priorizaron y organizaron en un tablero Kanban para facilitar su seguimiento (Ver Anexo II).

1.18. Limitaciones del sistema

Si bien el sistema está diseñado para abordar problemas fundamentales en el aprendizaje de la programación en C, presenta ciertas limitaciones en su versión inicial:

Dependencia de datos estáticos:

La base de conocimientos del chatbot se construye a partir de documentos predefinidos. Esto implica que su capacidad para responder depende de la calidad y amplitud de esos documentos, lo que podría limitar su eficacia en temas más avanzados o específicos que no estén suficientemente documentados.

Ausencia de retroalimentación interna:

En esta primera versión, el sistema no cuenta con un mecanismo integrado para recopilar feedback directamente desde la aplicación. Por lo que se utilizaron encuestas externas para evaluar la experiencia y satisfacción de los usuarios. Esto puede afectar a la detección de áreas que necesitan mejora inmediata.

Falta de personalización profunda:

Aunque el sistema permite seleccionar niveles de experiencia del usuario (nada experimentado, poco experimentado, experimentado), su capacidad para adaptar las respuestas a las necesidades específicas de cada estudiante es limitada, lo que puede impactar negativamente en la experiencia de los estudiantes con mayor o menor conocimiento previo.

Restricciones en la interacción:

El chatbot no mantiene un historial completo de conversaciones, lo que puede dificultar que el sistema interprete consultas que dependan de preguntas realizadas anteriormente. Esto puede generar respuestas menos precisas en temas que requieren contexto acumulado.

Dificultad en la comprensión de errores complejos:

La generación de código y corrección de errores por parte del chatbot está limitada a casos sencillos o intermedios. Los errores más complejos o aquellos relacionados con el diseño avanzado de algoritmos pueden no ser identificados o resueltos de manera adecuada.

Limitación de acceso a integraciones externas:

En esta versión, las herramientas externas integradas (por ejemplo, compiladores o editores) son limitadas, lo que podría reducir la utilidad del sistema en escenarios donde los estudiantes necesitan probar soluciones en tiempo real.

1.19. Diseño del sistema

En la fase de diseño, se trabajó en la creación de la arquitectura del chatbot y en la interfaz

gráfica que permitiría a los usuarios interactuar con el sistema de manera amigable.

1.19.1. Arquitectura del chatbot

La arquitectura del sistema se construyó utilizando Python como lenguaje base. Para ello, se emplearon herramientas como Streamlit para la interfaz gráfica, LangChain para la lógica de procesamiento del lenguaje y FAISS para la búsqueda de información relevante.

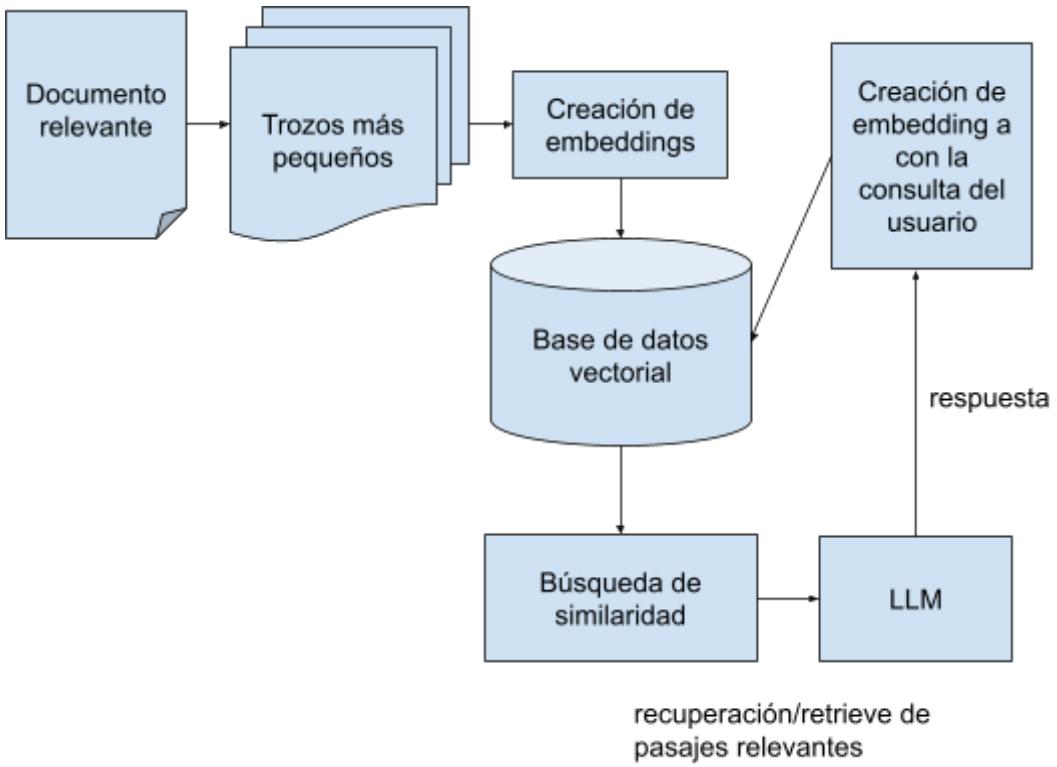


Figura 3-1 : Arquitectura de un chatbot basado en RAG

1.19.2. Diagrama de contexto y diagramas de secuencia

Se desarrollaron diagramas para ilustrar cómo el chatbot interactúa con otros sistemas y cómo procesa las consultas de los usuarios.

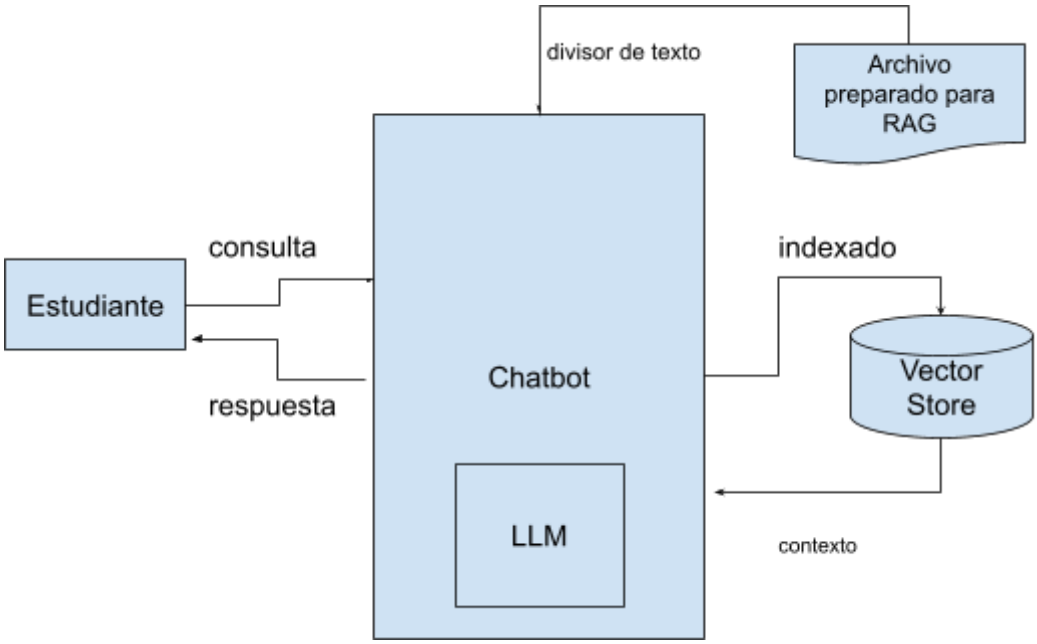


Figura 3-2 : Diagrama de contexto

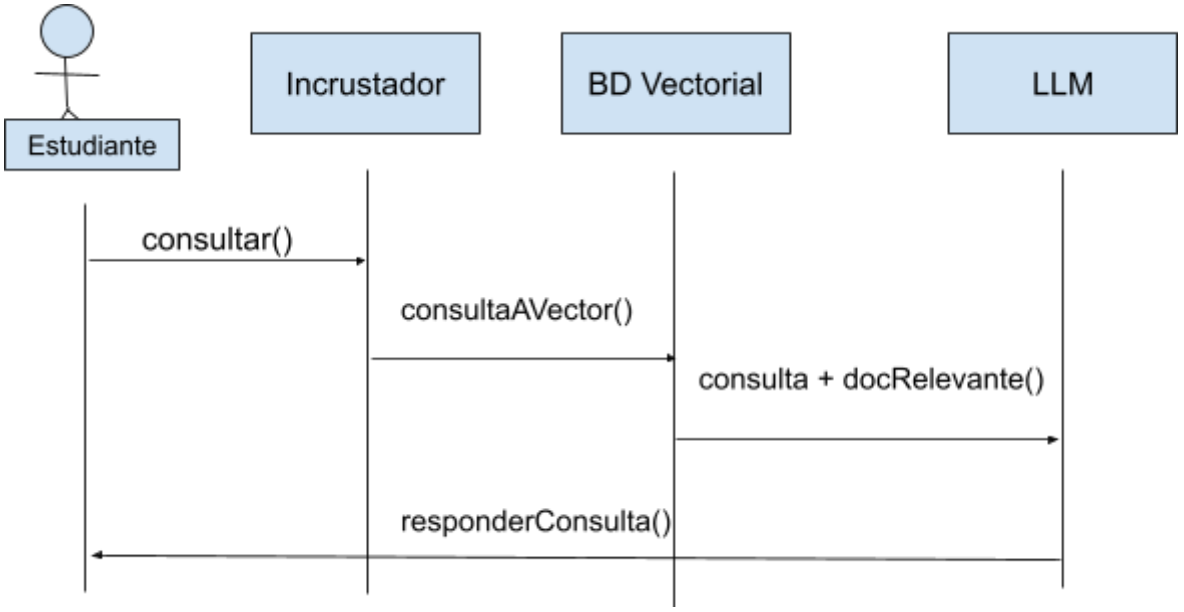


Figura 3-3 : Diagrama de secuencia de una consulta al chatbot

1.19.3. Interfaz gráfica

La interfaz de usuario, diseñada con un enfoque simple e intuitivo, permite a los estudiantes interactuar fácilmente ingresando preguntas y obteniendo respuestas contextualmente relevantes.

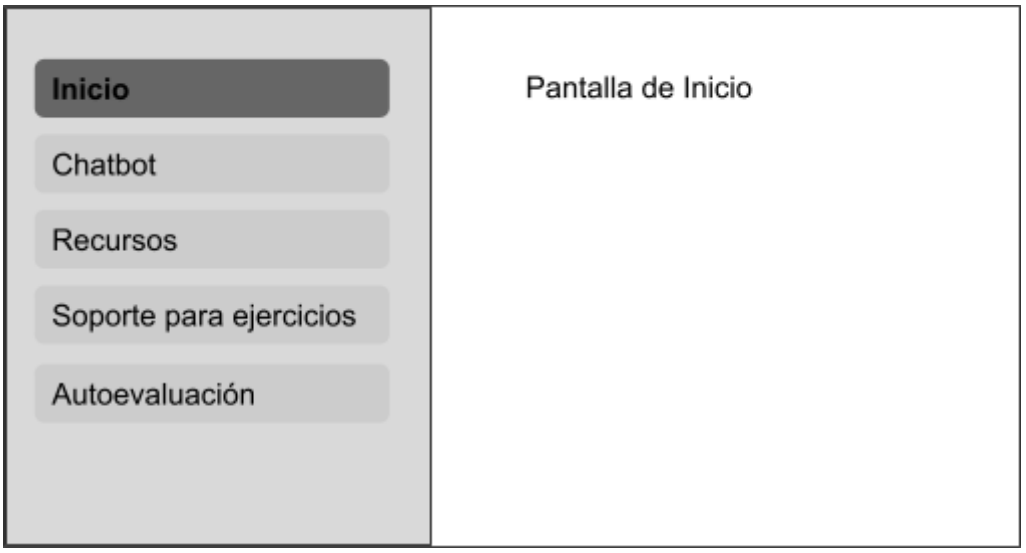


Figura 3-4 : Pantalla principal

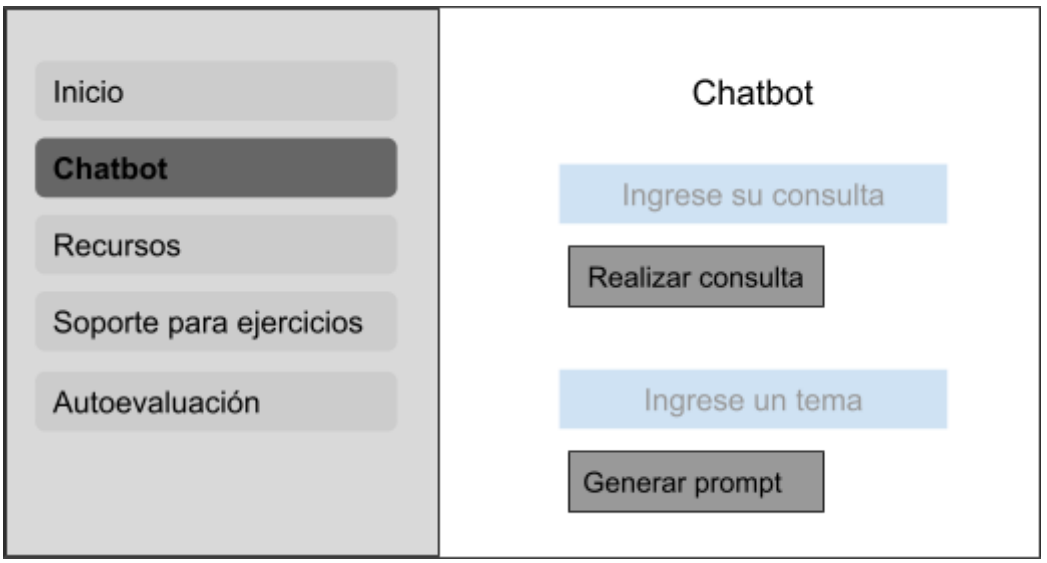


Figura 3-5 : Pantalla de consulta al chatbot

Figura 3-6 : Pantalla de recursos extra

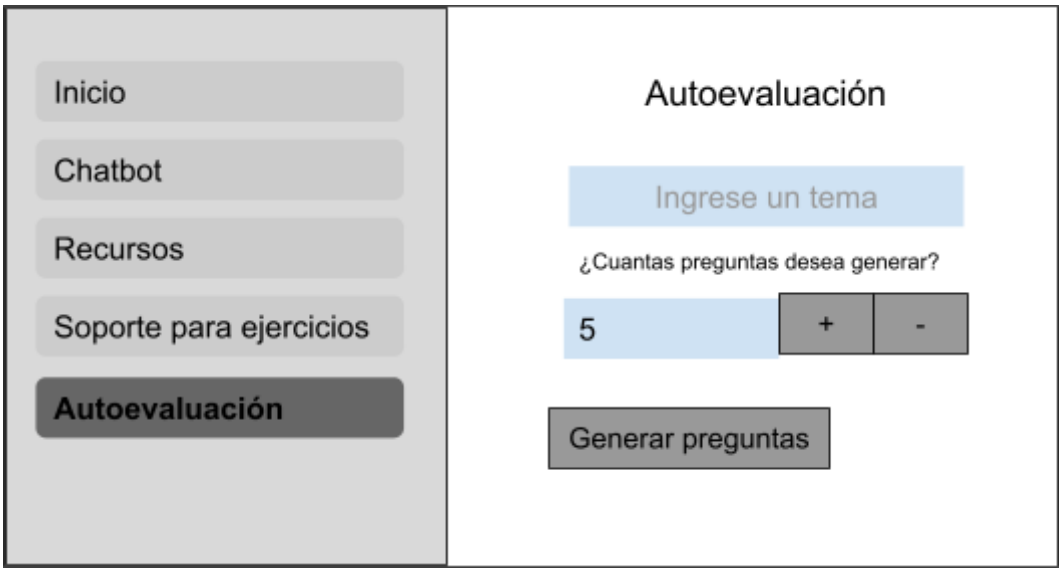


Figura 3-7 : Pantalla de autoevaluación

1.20. IMPLEMENTACIÓN

La implementación del sistema siguió la planificación establecida en el tablero Kanban. Las tareas fueron desglosadas en subtareas específicas y priorizadas según su relevancia para el usuario final.

1.20.1. Desarrollo de la lógica del chatbot

Se implementó el chatbot utilizando Python y ChatOpenAI, combinando técnicas de Recuperación Aumentada con Generación (RAG) para buscar documentos y generar respuestas contextualizadas.

1.20.2. Configuración del tablero Kanban

Las historias de usuario fueron descompuestas en tareas, tales como la validación de entradas y la configuración de prompts personalizados para diferentes tipos de consultas.

1.20.3. HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS

1.20.3.1. Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, ampliamente utilizado en diversas áreas como el desarrollo web, la ciencia de datos, la inteligencia artificial, y la automatización de tareas.

En el desarrollo del chatbot basado en RAG, Python actúa como lenguaje principal para integrar todas las herramientas necesarias (como Streamlit, LangChain, y FAISS), permitiendo que el chatbot recupere información relevante y genere respuestas basadas en el contexto (Olawore et al., 2024).

Características Principales de Python

Sintaxis Sencilla y Legible: Python está diseñado para ser fácil de leer y escribir, con una sintaxis limpia que permite a los programadores concentrarse en la lógica del código más que en la complejidad de la estructura.

Versatilidad: Python es un lenguaje de propósito general, por lo que se usa tanto en aplicaciones web y de escritorio como en análisis de datos, machine learning, y más.

Gran Comunidad y Ecosistema de Bibliotecas: Python cuenta con una comunidad muy activa y una gran cantidad de bibliotecas y frameworks, como Django para desarrollo web, NumPy y pandas para análisis de datos, y TensorFlow y PyTorch para aprendizaje automático.

Interactividad y Flexibilidad: Python permite el uso interactivo y es ideal para desarrollo rápido de prototipos gracias a su naturaleza interpretada, facilitando pruebas inmediatas y ajustes continuos.

Compatibilidad Multiplataforma: Python se puede ejecutar en múltiples sistemas operativos, como Windows, macOS y Linux, sin apenas modificaciones en el código[1] .

[1] Python Software Foundation, https://www.python.org

1.20.3.2. Streamlit

Streamlit que facilita la creación de aplicaciones interactivas, especialmente en el ámbito del análisis de datos y el aprendizaje automático. En el trabajo se utilizó para definir la interfaz, la cual permite capturar entradas de texto del usuario y mostrar respuestas de manera dinámica, con elementos de interfaz como campos de texto y botones [2].

1.20.3.3. GPT-4.0-mini

GPT-4o Mini es un modelo de lenguaje diseñado para ser más accesible y rentable, ofreciendo una alternativa más económica que modelos anteriores como GPT-3.5 Turbo. Este modelo se caracteriza por su bajo costo de procesamiento, con tarifas de 15 centavos por millón de tokens de entrada y 60 centavos por millón de tokens de salida. Además, tiene una ventana de contexto de 128K tokens y admite hasta 16K tokens de salida por solicitud. También supera a otros modelos pequeños en diversas métricas de rendimiento, alcanzando un 82% en el benchmark MMLU y destacándose en tareas de razonamiento matemático, codificación y comprensión multimodal.

El modelo está diseñado para manejar texto y visión, con planes futuros de incluir entrada y salida en audio y video. Su arquitectura mejorada permite un mejor rendimiento en la recuperación y generación de información, además de ofrecer mejoras en el procesamiento de datos estructurados y en la generación de respuestas contextuales en aplicaciones como chatbots y asistentes inteligentes. Su compatibilidad con múltiples idiomas también ha sido optimizada, lo que lo hace más eficiente en el manejo de texto en diversos idiomas.

En términos de seguridad, GPT-4o Mini incorpora medidas avanzadas para mitigar riesgos como los ataques de jailbreaks y manipulaciones en los prompts del sistema. Utiliza técnicas de aprendizaje por refuerzo con retroalimentación humana (RLHF) para mejorar la precisión y confiabilidad de sus respuestas. Además, ha sido evaluado por expertos en áreas como psicología social y desinformación para detectar posibles riesgos y mejorar su seguridad. Su disponibilidad incluye acceso para desarrolladores y usuarios de ChatGPT en las versiones Free, Plus y Team, con soporte para futuras funciones de ajuste fino [3].

[2] Snowflake Inc., https://streamlit.io/
[3] OpenAI, https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

1.20.3.4. Langchain

Es una biblioteca que facilita el desarrollo de aplicaciones que integran modelos de lenguaje. Ofrece una estructura modular que permite combinar modelos de lenguaje con otras herramientas de procesamiento de datos [4].

PromptTemplate: Define plantillas de texto para estandarizar las respuestas del modelo de lenguaje.

LLMChain: Permite construir cadenas de procesamiento de lenguaje natural, combinando modelos de lenguaje y plantillas de prompts.

TextLoader y CharacterTextSplitter: Herramientas para cargar, dividir y procesar textos grandes en fragmentos manejables, que luego se utilizan como contexto para el modelo.

1.20.3.5. FAISS (Facebook AI Similarity Search)

FAISS es una biblioteca de Facebook para la búsqueda y agrupamiento de vectores, muy útil para manejar grandes cantidades de datos. Almacena los documentos como vectores y permite realizar búsquedas de similitud, de manera que el chatbot puede recuperar documentos relevantes en función de la consulta del usuario [5].

1.20.3.6. Trello

Es una herramienta visual que permite a los equipos gestionar un proyecto y flujo de trabajo, así como supervisar tareas. Fue utilizado para administrar y tener un registro visual de las tareas lista y pendientes por realizar [6].

1.21. Estructura de la base de conocimientos

Una estructura clara y organizada facilita que el sistema RAG pueda localizar rápidamente información relevante en función de la consulta del usuario. Esto optimiza la velocidad y

[4] LangChain Inc. https://www.langchain.com/
[5] Meta. https://ai.meta.com/tools/faiss/
[6] Atlassian. https://trello.com/about

precisión de las respuestas. Al tener el contenido dividido en unidades y secciones bien definidas, el sistema puede ofrecer respuestas concisas y contextualizadas, mejorando la experiencia del usuario final (Du, et al. 2024).

El documento utilizado es una guía completa de programación en C, generada a partir de los apuntes de la cátedra, organizada en unidades temáticas que abordan conceptos esenciales del lenguaje. Cada unidad está estructurada en secciones bien definidas que incluyen explicaciones teóricas, ejemplos prácticos, tablas y preguntas frecuentes.

1.21.1. Estructura general del documento

El documento está dividido en múltiples unidades, cada una enfocada en un tema específico de programación en C. Ejemplo:

Unidad 1: Introducción y conceptos básicos.

Unidad 2: Entrada y salida.

Unidad 3: Estructuras de control, selección y repetición.

Unidad 4: Funciones.

Unidad 5: Punteros y arreglos.

Unidad 6: Estructuras.

Unidad 7: Control de periféricos.

1.21.2. Estructura interna de cada unidad

Cada unidad sigue un patrón que facilita la navegación y recuperación de información:

- Introducción al tema: Proporciona una visión general del contenido que se abordará.
- Desarrollo del contenido: Explica conceptos clave de forma detallada y progresiva.
- Ejemplos de código: Incluye fragmentos de código bien documentados que ilustran cada concepto.
- Tablas: Presenta tablas para describir tipos de datos, rangos, operadores, entre

otros.

- Errores comunes y soluciones: Identifica errores típicos y cómo solucionarlos.
- Preguntas frecuentes : Responde a dudas comunes relacionadas con el tema.
- Uso de Etiquetas y Títulos: Cada sección está etiquetada y titulada de forma clara, permitiendo una fácil identificación y recuperación de contenido.
    - Ejemplo:

    @tema: Describe el tema principal de la sección.

    @etiquetas: Proporciona palabras clave que facilitan la búsqueda.

- Uso de ejemplos: Cada concepto clave está acompañado de ejemplos prácticos en lenguaje C, lo que refuerza el aprendizaje y facilita la comprensión.

--- COMIENZO UNIDAD 5: Punteros y Arreglos ---
Guía de Programación en C - Punteros y Arreglos
@tema: Punteros y Arreglos en C
@etiquetas: punteros, arreglos, declaración, aritmética de punteros, matrices, operaciones con arreglos

1. Punteros: Concepto
Un puntero es una variable que almacena la dirección de memoria de otra variable. Es decir, su valor es la dirección donde se encuentra almacenado otro dato.

Ejemplo de puntero:
c
Copiar código
int x = 25;
int *pint = &x; // 'pint' es un puntero a 'x'
En este ejemplo, el puntero pint contiene la dirección de la variable x, y *pint accede al valor almacenado en esa dirección.

@ejemplo: Ejemplo de puntero que apunta a una variable int.

2. Declaración de Punteros
Los punteros se declaran especificando el tipo de dato al que apuntan, seguido de un asterisco *:

c
Copiar código
int *punt; // Puntero a entero
char *car; // Puntero a carácter
Inicialización de punteros:
c
Copiar código
int var = 14;
int *punt = &var; // Inicialización del puntero 'punt' con la dirección de 'var'
@ejemplo: Ejemplo de declaración e inicialización de punteros.

3. Operadores de Punteros
En C, hay dos operadores fundamentales para trabajar con punteros:

Operador de dirección (&): Obtiene la dirección de una variable.
Operador de contenido (*): Accede al valor almacenado en la dirección a la que apunta el puntero.
Ejemplo:
c
Copiar código
int x = 25;

Figura 3-8 : Fragmento de una unidad del documento usado para la base de conocimientos

1.22. Pruebas

Para asegurar la calidad del chatbot, se realizaron pruebas unitarias y funcionales. Se validó que el sistema cumpliera con los criterios de aceptación de las historias de usuario y que el chatbot respondiera correctamente a las consultas planteadas.

Pruebas unitarias: Evaluaron la correcta integración de los componentes, como la lógica del chatbot y la búsqueda de documentos.

Pruebas funcionales: Aseguraron que el chatbot cumpliera con las expectativas del usuario, verificando las funcionalidades a nivel general.

1.23. Recolección y Análisis de Feedback

Tras la implementación inicial, se evaluó la efectividad del chatbot mediante pruebas con estudiantes, quienes lo utilizaron durante una hora para resolver ejercicios de programación en C. Posteriormente, se recolectó su feedback mediante un formulario en Google Docs, el cual se analizó para identificar áreas de mejora.

Prioridades de mejora: El feedback fue utilizado para priorizar tareas de optimización, como mejoras en la interfaz y ajustes en la precisión de las respuestas (ver Anexo I).

Luego pasado un mes desde la última prueba se realizó una segunda prueba en la cual estudiantes resolvieron ejercicios durante una hora, posteriormente respondieron una encuesta en Google Docs y completaron una lista de cotejo. Del análisis de estos datos se puede concluir:

Los estudiantes han valorado positivamente la claridad de las explicaciones proporcionadas por el chatbot. Este resultado sugiere que el asistente es efectivo en la transmisión de

conceptos técnicos de programación en C, convirtiéndose en un recurso valioso para los estudiantes en su proceso de aprendizaje.

También, un número considerable de estudiantes reportó haber aprendido algo nuevo a través de las interacciones con el chatbot, lo que demuestra su capacidad para complementar la enseñanza en el aula y abordar vacíos de conocimiento en temas complejos como punteros y estructuras de datos.

El 100% de los estudiantes está dispuesto a utilizar el chatbot nuevamente, lo que resalta el nivel de aceptación y utilidad percibida. Esto es un indicador de que la herramienta es bien recibida y valorada como un apoyo constante en el aprendizaje de programación.

Sin embargo, existen áreas en las que se puede mejorar:

La rapidez de respuesta del chatbot ha sido valorada de manera moderada, lo que indica que, en algunos casos, el tiempo de espera para recibir respuestas es un factor que podría mejorarse. Optimizar la velocidad de procesamiento, ya sea ajustando las consultas en el sistema RAG o mejorando la infraestructura técnica, sería un paso necesario para incrementar la satisfacción del usuario.

El porcentaje de estudiantes que reportaron que el chatbot comprendía correctamente sus preguntas es relativamente bajo, y una proporción significativa de usuarios abandonó las conversaciones debido a problemas de comprensión.

Una de las áreas más críticas es la capacidad del chatbot para detectar y corregir errores en el código generado por los estudiantes. Este aspecto presenta una baja calificación, lo que indica que el asistente no está cumpliendo de manera efectiva esta función. Un enfoque futuro debería centrarse en enriquecer la base de conocimiento del sistema RAG con más ejemplos prácticos, documentación relevante y respuestas basadas en escenarios de error comunes en programación en C.

Por último, una baja eficiencia relativa sugiere que los estudiantes no están resolviendo significativamente más ejercicios con la ayuda del chatbot que sin él. Para abordar esto, sería útil proporcionar ejemplos de resolución paso a paso, sugerencias de optimización de

código, y enfoques alternativos que los estudiantes puedan aplicar a sus problemas de programación.

Modelo de conceptos (Árbol de características)

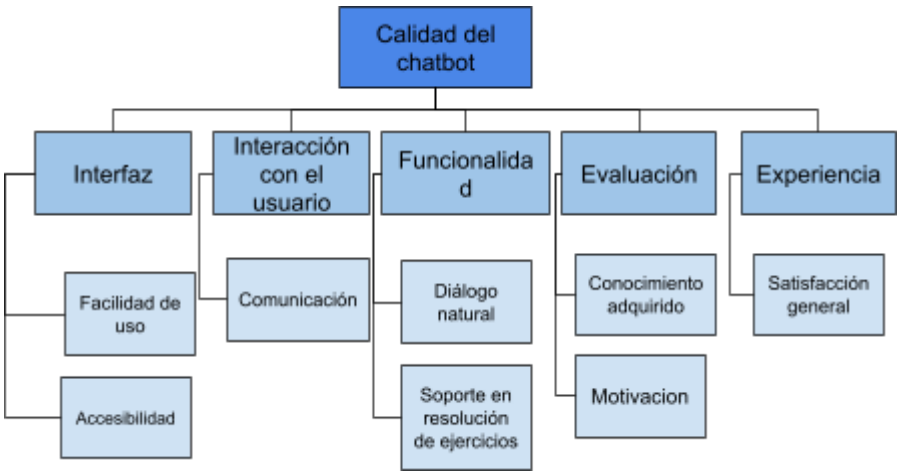


Figura 3-9 : Árbol de características

1.24. Planificación y Control del Proyecto

La planificación del proyecto se divide en fases, con un enfoque en mantener un flujo de trabajo continuo y flexible.

1.24.1. Tareas de desarrollo

Las tareas principales ligadas al desarrollo del producto, se agruparon en distintas fases:

- **Fase 1: Desarrollo y Configuración Inicial (3 semanas)**: Creación de la interfaz y configuración del chatbot.
- **Fase 2: Ajustes y Documentación (3 semanas)**: Mejora del feedback y organización de la documentación técnica.
- **Fase 3: Ejercicios y Evaluación (4 semanas)**: Desarrollo de ejercicios prácticos y sesiones de evaluación.
- **Fase 4: Pruebas y Ajustes Finales (2 semanas)**: Testing de las funcionalidades y ajustes finales.

1.24.2. Diagrama de Gantt

Planificación de las tareas de desarrollo

Fases / Semanas	1	2	3	4	5	6	7	8	9	10	11	12
Fase 1: Desarrollo y configuración inicial												
Fase 2: Ajustes y documentación												
Fase 3: Ejercicios y evaluación												
Fase 4: Pruebas y ajustes finales												

1.24.3. Formato del tablero Kanban

El tablero estará formado por las siguientes columnas:

- Backlog: donde se colocaran las tareas a realizar y las nuevas tareas que surjan
- Haciendo: contendrá las tareas en las que el equipo de desarrollo está trabajando
- Testing:contendrá las tareas para el equipo de testing
- Hecho: se colocaran las tareas que hayan pasado por el proceso de desarrollo y testeo.

1.24.4. Tareas que forman parte del backlog

Las historias de usuario se analizan y se desglosan en tareas específicas para el desarrollo del chatbot, las cuales se registran y organizan en un tablero visual para facilitar el seguimiento y priorización de cada tarea (ver Anexo III). Este tablero permite ajustar y asignar prioridades a las historias de usuario en función de su relevancia para los objetivos del sistema, así como estimar los recursos necesarios y coordinar acuerdos con la cátedra.

Esta planificación permite un flujo continuo de trabajo, manteniendo flexibilidad para incorporar nuevos requerimientos propuestos.

- **Fase 1: Desarrollo y configuración inicial (3 semanas)**
  - Crear el campo de entrada de texto.
  - Desarrollar el botón de envío.
  - Configurar la lógica del chatbot.
  - Implementar selección de nivel de experiencia.
- **Fase 2: Ajustes y documentación (3 semanas)**
  - Ajustar feedback del chatbot según el nivel de experiencia.
  - Organizar la sección de acceso a documentación técnica.
  - Crear enlaces a documentación técnica y videos tutoriales.
- **Fase 3: Ejercicios y evaluación (4 semanas)**
  - Categorizar videos por temas clave.
  - Agregar enlaces a compiladores en línea e incluir instrucciones de uso.
  - Desarrollar lista de ejercicios prácticos clasificados.
  - Crear sección de evaluación con preguntas de opción múltiple.
  - Configurar feedback inmediato.
  - Implementar botón de reinicio de evaluación.
- **Fase 4: Pruebas y ajustes finales (2 semanas)**
  - Testing de todas las funcionalidades desarrolladas.
  - Realizar ajustes finales según los resultados de las pruebas.

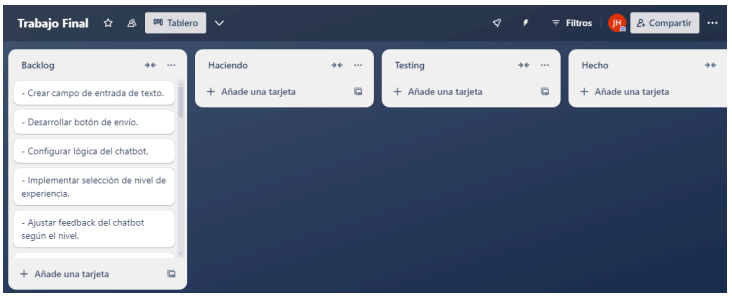


Figura 3-10 : Tablero Kanban inicial

1.24.5. Evolución en el tablero Kanban

Se llevó un control visual del avance del proyecto mediante el tablero Kanban, registrando el estado de las tareas al finalizar cada fase.

Tablero correspondiente a la tercera semana

Figura 3-11 : Tablero Kanban a la tercer semana

Tablero correspondiente a la sexta semana

Figura 3-12 : Tablero Kanban a la sexta semana

Tablero correspondiente a la novena semana

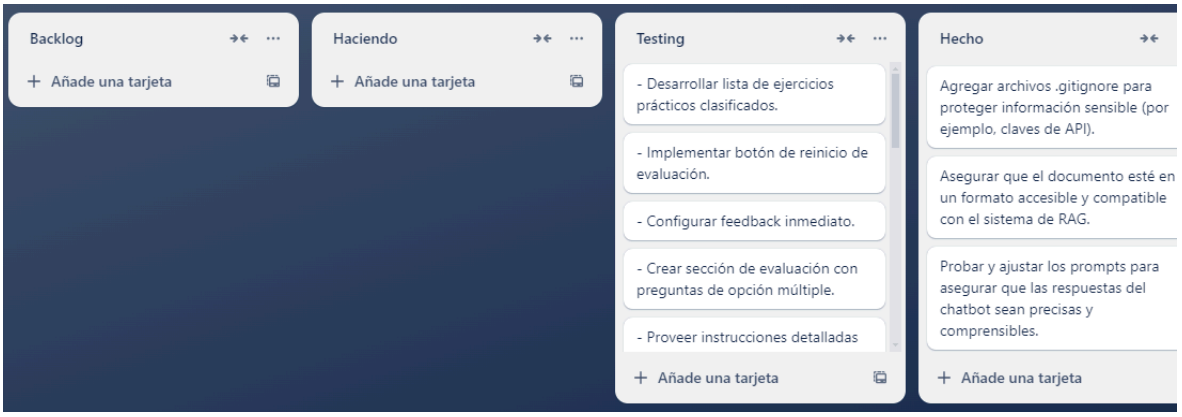


Figura 3-13 : Tablero Kanban en la novena semana

Tablero correspondiente a la duodécima semana

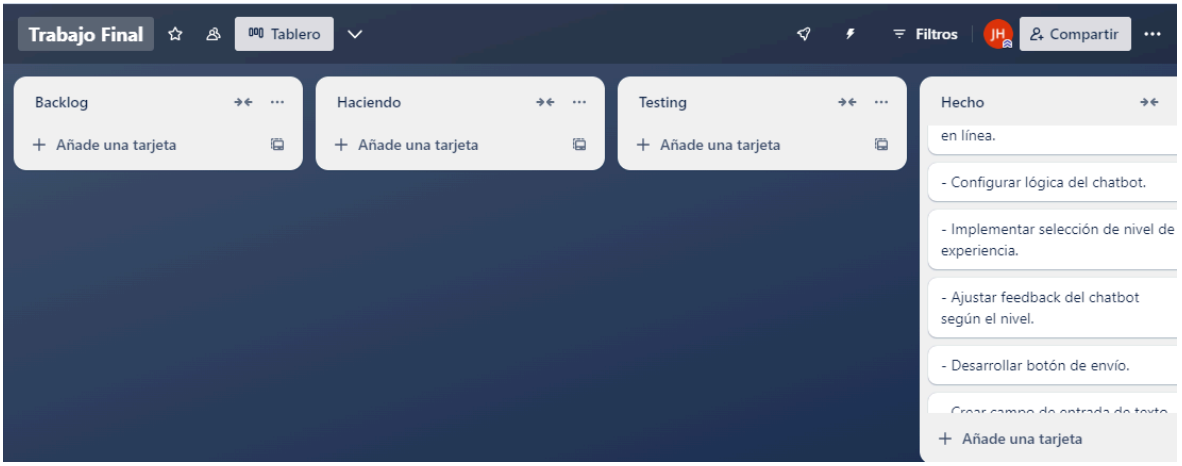


Figura 3-14 : Tablero Kanban en la semana doce

1.25. CAPTURAS DE PANTALLA DEL SISTEMA IMPLEMENTADO

Pantalla inicial

Figura 3-15 : Interfaz del sistema - Inicio

Pantalla Chatbot

Figura 3-16 : Interfaz del sistema - Chatbot

Pantalla de recursos

Figura 3-17 : Interfaz del sistema - Recursos

Pantalla de soporte para ejercicios

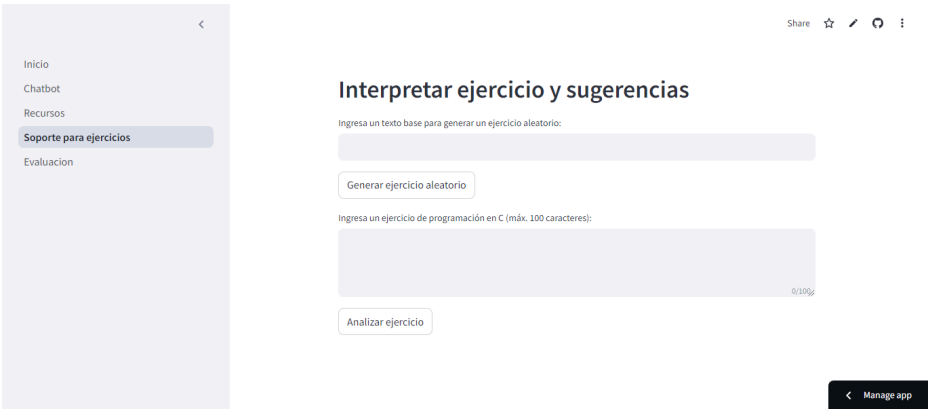


Figura 3-18 : Interfaz del sistema - Soporte para ejercicios

Pantalla de autoevaluación

Figura 3-19 : Interfaz del sistema - Evaluación

1.26. BUENAS PRÁCTICAS PARA EL DESARROLLO DE UN CHATBOT RAG

Una de las claves en el desarrollo es contar con una base de conocimientos bien estructurada. La información que el chatbot utiliza debe ser confiable y organizada de forma clara para facilitar su recuperación. Además, dividir los documentos en fragmentos más pequeños ayuda a que el modelo encuentre respuestas más precisas en lugar de devolver información extensa o irrelevante.

Para garantizar que el chatbot recupere información de manera eficiente y precisa, se ha implementado un proceso estructurado de carga y segmentación de documentos. En el código, la función cargar_documentos() (ver Anexo V) utiliza TextLoader para leer la base de conocimientos (unidades_completas.txt) y luego emplea CharacterTextSplitter(chunk_size=500, chunk_overlap=0), lo que divide los documentos en fragmentos de 500 caracteres. Esto evita respuestas extensas e irrelevantes y permite que el modelo procese información en partes manejables.

Además, el uso de OpenAIEmbeddings() junto con FAISS.from_documents(docs, embeddings) convierte estos fragmentos en representaciones vectoriales, facilitando la búsqueda semántica eficiente mediante db.similarity_search(user_query). Esto permite encontrar fragmentos relevantes en función de la consulta del usuario, en lugar de depender de coincidencias exactas de palabras clave.

Por otro lado, el código asegura que la información recuperada sea utilizada de manera efectiva en la generación de respuestas. Para ello, se construye un PromptTemplate, donde el contexto recuperado se incorpora a la pregunta del usuario

prompt_template = """

Eres un experto en C. Responde solo usando C, sin mencionar C++ ni características relacionadas.

{experience_level}

Documentos relevantes:

{context}

Pregunta: {user_query}

"""

El proceso de búsqueda de información dentro de la base de conocimientos debe ser rápido y preciso. Para esto, se pueden usar herramientas como FAISS o Elasticsearch, que permiten encontrar fragmentos de texto relevantes en cuestión de milisegundos. Además, se pueden aplicar técnicas de comparación, como la similitud coseno, para ordenar los documentos recuperados según su relevancia antes de que sean utilizados por el modelo generativo.

Para optimizar la búsqueda de información en la base de conocimientos, el código utiliza FAISS (FAISS.from_documents(docs, embeddings)) para almacenar y recuperar fragmentos de texto relevantes de manera eficiente. FAISS es una herramienta diseñada para búsquedas rápidas en grandes volúmenes de datos mediante técnicas de indexación y búsqueda de similitud. En este caso, la función db.similarity_search(user_query) permite encontrar los fragmentos más cercanos semánticamente a la consulta del usuario en cuestión de milisegundos.

Además, FAISS emplea una métrica de similitud para ordenar los documentos recuperados según su relevancia. Aunque en el código no se especifica explícitamente la métrica

utilizada, FAISS permite aplicar la similitud coseno como criterio de comparación, lo que garantiza que los fragmentos seleccionados sean los más cercanos al significado de la pregunta del usuario. Esto se evidencia en la línea:

docs = db.similarity_search(user_query)

context = "\n\n".join([doc.page_content[:500] for doc in docs[:2]])

Aquí, similarity_search(user_query) obtiene los documentos más relevantes, y luego se filtran los dos mejores fragmentos para reducir el uso de tokens y mejorar la precisión de la respuesta. Gracias a esta optimización, el chatbot evita sobrecarga de información y mejora la calidad del contenido proporcionado al usuario.

El modelo generativo debe estar optimizado para producir respuestas claras y bien fundamentadas. Modelos como GPT-4o mini permiten generar respuestas con menor costo y mayor rapidez. También es útil ajustar las respuestas en función del contexto, asegurándose de que la información recuperada sea incorporada correctamente en la generación del texto. La ingeniería de de prompts puede ayudar a que el chatbot ofrezca respuestas más precisas.

Es fundamental controlar la calidad y evitar sesgos en las respuestas. Para lograrlo, se recomienda monitorear los datos utilizados por el modelo. También es importante evaluar regularmente las respuestas del chatbot y realizar ajustes si se detectan problemas en la información proporcionada.

El chatbot debe estar diseñado para ser rápido y escalable, de modo que pueda responder sin demoras incluso cuando hay muchos usuarios al mismo tiempo. Para ello, se pueden implementar técnicas como almacenamiento en caché de respuestas frecuentes y procesamiento en lote para optimizar el rendimiento del sistema. Además, es recomendable utilizar una arquitectura modular que permita actualizar el chatbot sin necesidad de hacer grandes cambios en su estructura.

La seguridad y privacidad de los datos también es una prioridad. Es importante implementar medidas para proteger la información de los usuarios y evitar la exposición de datos sensibles. Algunas estrategias incluyen el cifrado de las interacciones, la

anonimización de datos personales y el cumplimiento de normativas de privacidad como GDPR o la Ley de Protección de Datos Personales en cada país donde se utilice el chatbot.

Finalmente, el chatbot debe contar con un sistema de evaluación y mejora continua. Para esto, es útil definir métricas como la precisión de las respuestas, el tiempo de respuesta y la satisfacción del usuario. También se recomienda permitir que los usuarios brinden feedback sobre la calidad de las respuestas y utilizar estos datos para realizar ajustes y mejoras en el modelo.

1.27. TECNICAS DE INGENIERIA DE PROMPTS APLICADAS EN EL CHATBOT

En este tipo de aplicaciones, el prompt es la entrada que alimenta al modelo, y la ingeniería de prompts puede influir drásticamente en los resultados. El actuar del modelo se verá afectado por la forma del prompt como su contenido.

La ingeniería de prompts es el diseño y optimización de estas entradas para que las respuestas de los LLMs sean precisas, relevantes y coherentes. Es clave a la hora de aprovechar la potencia de estos modelos (Chen et al. 2023).

En la implementación del chatbot se utilizó un prompt específico según la tarea o módulo que se desarrollaba, por ejemplo en el módulo de “Autoevaluación” se utilizó este prompt

“**Eres un profesor experto en el lenguaje de programación C.Evita hacer referencias a C + +, cin, cout, namespaces, o características específicas de C + +.**

Genera {numero_de_preguntas} preguntas de opción múltiple combinando el siguiente tema: {user_query} y un tema adicional: {tema_aleatorio}.

También debes considerar la posibilidad de formular preguntas del tipo "¿Cuál es la salida de este código?". Si optas por este tipo de pregunta, incluye el código en un bloque de código de la siguiente manera:

#include <stdio.h>

int main() {{

// Tu código aquí

return 0;

}}

Cada pregunta debe incluir opciones y una respuesta correcta. El formato JSON debe ser el siguiente:

[

{{

"pregunta": "La pregunta generada.",

"codigo": "Aquí va el código si la pregunta es sobre la salida del mismo.",

"opciones": {{

"A": "Opción A",

"B": "Opción B",

"C": "Opción C",

"D": "Opción D"

}},

"respuesta_correcta": "La letra de la opción correcta y una breve explicación."

}},]

Documentos relacionados:

{context}”

En este prompt se utilizan varias técnicas de ingeniería de prompts para limitar la respuesta del modelo, como ser:

Role Prompting : Se le asigna un rol específico al modelo: "Eres un profesor experto en el lenguaje de programación C". Esto guía al modelo a responder desde la perspectiva de un experto en C, afectando el tono, el lenguaje y el contenido de las preguntas generadas (Njifenjou et al., 2024).

Constraint Prompting : Se imponen restricciones claras sobre el contenido que el modelo debe evitar: "Evita hacer referencias a C + +, cin, cout, namespaces, o características específicas de C + +". Esto asegura que las preguntas se centren exclusivamente en el lenguaje C, sin elementos de C++ (Suri et al., 2024).

Task Specification: Se define claramente la tarea que el modelo debe realizar: "Genera {numero_de_preguntas} preguntas de opción múltiple combinando el siguiente tema: {user_query} y un tema adicional: {tema_aleatorio}". Se especifica el tipo de preguntas, la cantidad y cómo deben combinarse los temas (Stoica et al., 2024).

Output Format Specification: Se detalla el formato exacto en el que se espera la salida del modelo: "El formato JSON debe ser el siguiente: [...]". Esto facilita el procesamiento y uso de las preguntas generadas (Ekin,2023).

Few-Shot Prompting: se da una estructura clara y un ejemplo de cómo debe ser una pregunta sobre la salida de un código (Ma et al., 2024):

```
#include <stdio.h>

int main() {{

   // Tu código aquí

   return 0;

}}
```

Context Providing (Provisión de Contexto): Se proporciona un contexto adicional al modelo mediante la inclusión de "Documentos relacionados: {context}". Esto permite que el modelo utilice información adicional para generar preguntas más relevantes y precisas (Ekin,2023).

1.28. Reflexiones finales

El chatbot desarrollado ha demostrado ser una herramienta valiosa para apoyar el aprendizaje de la programación en C en la cátedra de Informática. Según los resultados obtenidos a partir de las encuestas y el feedback de los usuarios, esta solución tecnológica se posiciona como un recurso práctico para facilitar la comprensión de conceptos clave y asistir a los estudiantes en la resolución de ejercicios. Los estudiantes han destacado su usabilidad, la claridad de la interfaz y la rapidez en la entrega de respuestas, atributos que lo hacen una alternativa accesible y eficaz en el ámbito educativo.

Sin embargo, no todos los usuarios tuvieron una experiencia homogénea. Algunos expresaron dificultades al plantear preguntas más complejas o relacionadas con fragmentos de código detallados, lo que revela ciertas limitaciones en el alcance actual del chatbot. Pese a ello, este aspecto no desmerece su utilidad general, sino que subraya la importancia de continuar explorando su potencial para abordar una mayor diversidad de situaciones de aprendizaje.

En síntesis, el chatbot ha cumplido su objetivo principal de brindar soporte educativo efectivo a los estudiantes, consolidándose como un recurso que enriquece la experiencia de aprendizaje en la programación en lenguaje C. Su impacto positivo reafirma el valor de integrar tecnologías basadas en inteligencia artificial en la educación, ofreciendo soluciones accesibles, dinámicas y ajustadas a las necesidades de los usuarios.

1.29. Mejoras a futuro

Para optimizar la funcionalidad del chatbot y atender las observaciones de los usuarios, una primera mejora se centra en ampliar el límite de caracteres permitidos en las consultas. Actualmente, esta limitación dificulta plantear preguntas detalladas, especialmente aquellas

que incluyen fragmentos de código.

Además, resulta fundamental enriquecer y actualizar la base de conocimientos empleada por RAG. Agregar explicaciones más detalladas, ejemplos prácticos y preguntas frecuentes relacionadas con los temas abordados aumentará la relevancia de las respuestas. Este enfoque garantiza que el chatbot esté preparado para cubrir una mayor variedad de consultas, respondiendo con precisión sin requerir modificaciones en el modelo subyacente. También se sugiere incluir un mecanismo para que los usuarios califiquen las respuestas dentro de la interfaz, lo que proporcionará feedback inmediato y ayudará a ajustar la base documental según las necesidades recurrentes.

La interfaz del chatbot puede mejorarse ajustando elementos visuales como el contraste de colores y el tamaño adaptable de la fuente, garantizando así una experiencia más inclusiva. Asimismo, se puede integrar una herramienta de previsualización para fragmentos de código ingresados, lo que facilitará la validación automática antes de enviarlos al sistema. Esto no solo reducirá errores en las consultas, sino que también incrementará la precisión de las respuestas, fortaleciendo la confianza del usuario en la herramienta.

Finalmente, para fomentar un aprendizaje más activo, se podría integrar el chatbot con herramientas externas como compiladores en línea o simuladores interactivos. Estas funcionalidades permitirían a los estudiantes probar el código directamente desde la interfaz y comprender mejor los conceptos que están aprendiendo

Conclusiones

1.30. Conclusión

1.30.1. Impacto general del Chatbot en el aprendizaje

El desarrollo e implementación del chatbot para la cátedra de Informática han evidenciado un impacto positivo en la experiencia educativa de los estudiantes. Según los datos recopilados, el chatbot cumplió con su propósito principal de facilitar la comprensión de los conceptos fundamentales de la programación en C, especialmente en temas complejos como punteros y la lógica de programación. Los estudiantes reconocieron que el sistema ha contribuido a reducir el tiempo necesario para resolver dudas y ha ofrecido un acceso inmediato a recursos educativos que complementan las clases presenciales.

Además, el análisis de las encuestas y el feedback recibido muestra que el chatbot no solo ha sido útil como una herramienta de apoyo técnico, sino que también ha incrementado la confianza de los estudiantes al interactuar con contenidos que anteriormente les resultaban desafiantes. Esto indica que su incorporación al proceso educativo ha promovido un aprendizaje más autónomo y eficiente, ayudando a mitigar las limitaciones de tiempo y recursos dentro del aula.

1.30.2. Modelos de lenguaje utilizados

Para el desarrollo del chatbot, se evaluaron distintos LLM, centrándose en GPT-3.5 y GPT-4o Mini. Inicialmente, se consideró la implementación de GPT-3.5 debido a su estabilidad y rendimiento en tareas generales de procesamiento de lenguaje natural. Sin embargo, tras un análisis de costos y eficiencia, se optó por GPT-4o Mini, ya que ofrece una relación costo-beneficio más favorable.

GPT-4o Mini se destaca por su menor costo en comparación con modelos más avanzados, mientras mantiene un rendimiento sólido en tareas de generación de texto y comprensión de consultas. Esta decisión permitió garantizar la viabilidad del chatbot a largo plazo sin comprometer su calidad.

1.30.3. Evaluación de las funcionalidades técnicas

Desde el punto de vista técnico, el chatbot mostró un rendimiento robusto en aspectos clave como la rapidez de respuesta y la claridad de las explicaciones proporcionadas. No obstante, algunos estudiantes señalaron que la interpretación de consultas complejas o ambiguas aún representa un desafío para el sistema, lo cual subraya la necesidad de optimizar el motor de procesamiento del lenguaje natural y ampliar la base de conocimiento utilizada.

Asimismo, la calidad del diseño de la interfaz fue uno de los puntos más destacados por los usuarios. Con un promedio de aceptación del 73% en aspectos como contraste de colores, tipografía y accesibilidad, el chatbot demostró ser una herramienta visualmente amigable. Sin embargo, el análisis de las observaciones indicó que ciertos ajustes, como la simplificación de menús o la incorporación de tutoriales interactivos, podrían mejorar aún más la experiencia del usuario.

1.30.4. Fortalezas del sistema

Las principales fortalezas identificadas durante la implementación son; la disponibilidad del chatbot en cualquier momento, la capacidad de ofrecer respuestas inmediatas motivando a los estudiantes a realizar más consultas y practicar ejercicios y adaptar las respuestas según el nivel de conocimiento del usuario. Estas características posicionan al chatbot como un recurso viable para su integración en otras cátedras.

1.30.5. Áreas de mejora y limitaciones

A pesar de los logros alcanzados, el trabajo reveló limitaciones que deben ser atendidas. La rapidez de respuesta, aunque adecuada en general, mostró inconsistencias en contextos de alta carga de usuarios, lo que podría estar relacionado con las restricciones de infraestructura técnica. Asimismo, la comprensión de consultas avanzadas fue un área donde el sistema obtuvo calificaciones más bajas, destacando la importancia de ampliar y diversificar los datos de entrenamiento del modelo de lenguaje utilizado.

Otra limitación significativa fue la falta de retroalimentación en tiempo real sobre la calidad de las interacciones. Aunque las encuestas realizadas proporcionaron información

valiosa, un sistema integrado de monitoreo y evaluación continua podría ofrecer una visión más completa y permitir ajustes más ágiles.

1.30.6. Relevancia y sostenibilidad

El impacto educativo del chatbot trasciende su implementación inicial, demostrando que las tecnologías basadas en inteligencia artificial tienen el potencial de transformar las metodologías de enseñanza en entornos universitarios. La experiencia adquirida en este trabajo establece un precedente para futuras iniciativas, mostrando que es posible implementar herramientas tecnológicas con un enfoque inclusivo y accesible, sin requerir grandes inversiones.

Además, el trabajo destaca la importancia de fomentar un enfoque sostenible en el desarrollo de sistemas educativos empleando tecnologías de información y comunicación, capaces de adaptarse a las necesidades cambiantes de los estudiantes y docentes.

1.30.7. Conclusión Final

En resumen, el chatbot ha demostrado ser un recurso innovador y eficaz para apoyar el aprendizaje de la programación en C. Si bien existen áreas que requieren mejoras, los resultados obtenidos hasta ahora refuerzan su utilidad y viabilidad como una herramienta educativa. La experiencia acumulada durante su desarrollo e implementación servirá como base para expandir su alcance, explorar nuevas funcionalidades y promover su adopción en otros contextos académicos.

Este trabajo no solo ha beneficiado a los estudiantes, sino que también ha proporcionado a los docentes una visión renovada sobre cómo la tecnología puede integrarse en el proceso educativo, potenciando el aprendizaje y mejorando la calidad de la enseñanza.

Bibliografía

AbuShawar, B., & Atwell, E. (2015). ALICE chatbot: Trials and outputs. Computación y sistemas, 19(4), 625-632.

Ahumada, H. C., Póliche, M. V., & Contreras, N. A. (2021). *Chatbot asistente para atención de aspirantes en la Facultad de Tecnología y Ciencias Aplicadas de la UNCA*. *Simposio Argentino de Inteligencia Artificial*.

Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In 2013 39th Euromicro conference on software engineering and advanced applications (pp. 9-16). IEEE.

Ait Baha, T., El Hajji, M., Es-Saady, Y., et al. (2023). *The impact of educational chatbot on student learning experience*. *Education and Information Technologies*. https://doi.org/10.1007/s10639-023-12166-w

Akram Sawiras, K. (2024). Evaluation and Development of Innovative NLP Techniques for Query-Focused Summarization Using Retrieval Augmented Generation (RAG) and a Small Language Model (SLM) in Educational Settings.

Alboaie, S. (2024). Thinking Databases.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. (2018). *Improving language understanding by generative pre-training*. OpenAI Blog.

Arias-Navarrete, A. S., Palacios-Pacheco, X. I., & Villegas-Ch., W. (2020). ***Integración de un chatbot a un LMS como asistente para la gestión del aprendizaje.***

Barriere, V. (2024). Fantastic Biases (What are They) and Where to Find Them. arXiv preprint arXiv:2411.15051.

Batista, A. R. (2017). ***Uso de chatbots como apoyo para la comunicación en el aula. Un asistente virtual 24x7x365 colaborando con el curso.***

Bezverhny, O., et al. (2020). ***Use of chat bots in Learning Management Systems.***

Brown, Peter & Dellapietra, Vincent & Souza, Peter & Lai, Jennifer & Mercer, Robert. (1992). ***Class-Based n-gram Models of Natural Language. Computational Linguistics.*** **18. 467-479.**

Bruckhaus, T. (2024). RAG Does Not Work for Enterprises. arXiv preprint arXiv:2406.04369.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. ***ACM Transactions on Intelligent Systems and Technology*****,** ***15*****(3), Article 39, 45 pages. https://doi.org/10.1145/3641289**

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023). Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. arXiv preprint arXiv:2310.14735.

Chen, J., Bao, R., Zheng, H., Qi, Z., Wei, J., & Hu, J. (2024). Optimizing Retrieval-Augmented Generation with Elasticsearch for Enhanced Question-Answering Systems. arXiv preprint arXiv:2410.14167.

Chen, X., Xie, H., Zou, D., & Hwang, G.-J. (2020). ***Application and theory gaps during the rise of Artificial Intelligence in Education*****. *Computers and Education: Artificial Intelligence, 1*, 100002.**

Chen, Y., Liu, Y., Yan, J., Bai, X., Zhong, M., Yang, Y., ... & Zhang, Y. (2024). See what llms cannot answer: A self-challenge framework for uncovering llm weaknesses. arXiv preprint arXiv:2408.08978.

Chow, J. C. L. (2021). ***Chatbot for Health Care and Oncology Applications Using Artificial Intelligence and Machine Learning: Systematic Review.***

Du, X., Zheng, G., Wang, K., Feng, J., Deng, W., Liu, M., ... & Lou, Y. (2024). Vul-RAG: Enhancing LLM-based Vulnerability Detection via Knowledge-level RAG. arXiv preprint arXiv:2406.11147.

Ekin, S. (2023). Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices. Authorea Preprints.

Fernoagă, V.P., Stelea, G. A., & Gavrila, C. (2017). ***Intelligent Education Assistant Powered by Chatbots.***

Gago, N. M., & Navajas, L. (2019). ***Chatbot accesible para el Área de Asuntos Estudiantiles (UNLP).***

Gan, W., Qi, Z., Wu, J., & Lin, J. C.-W. (2018). ***Adapting Large Language Models for Education: Foundational Capabilities, Potentials, and Challenges.***

Gan, W., Qi, Z., Wu, J., & Lin, J. C.-W. (2023). ***Large Language Models in Education: Vision and Opportunities.*** **En IEEE International Conference on Big Data (pp. 1–10).**

García, N. M. (2015). Uso de la similitud semántica para la recuperación de información geoespacial (Doctoral dissertation, Universitat d'Alacant/Universidad de Alicante).

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Gao, Y., Xiong, Y., Wang, M., & Wang, H. (2024). Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. arXiv preprint arXiv:2407.21059.

Ghorbani, N. (2024). Enhancing Corporate Document Management Systems with AI: Leveraging Embeddings and Semantic Search for Efficient Retrieval and Chat-based Assistance.

Goya, P., & Minz, N., & Sha, A. (2023). ***Chatbots and Virtual Assistants in Education: Enhancing Student Support and Engagement.***

Gupta, A., Hathwar, D., & Vijayakumar, A. (2020). Introduction to AI chatbots. International Journal of Engineering Research and Technology, 9(7), 255-258.

Gupta, S., Ranjan, R., & Singh, S. N. (2024). A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. arXiv preprint arXiv:2410.12837.

Han, Z. (2021). ***The applications of chatbot.***

Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: an introduction to voice assistants. Medical reference services quarterly, 37(1), 81-88.

Jansson, M., Tian, K., Hrastinski, S., & Engwall, O. (2024, May). An initial exploration of semi-automated tutoring: How AI could be used as support for online human tutors. In Networked Learning Conference (Vol. 14).

Kakar, S., Maiti, P., Taneja, K., Nandula, A., Nguyen, G., Zhao, A., ... & Goel, A. (2024). Jill Watson: Scaling and Deploying an AI Conversational Agent in Online Classrooms. In International Conference on Intelligent Tutoring Systems (pp. 78-90). Cham: Springer Nature Switzerland.

Kalyan, K. S. (2024). A survey of GPT-3 family large language models including ChatGPT and GPT-4. Natural Language Processing Journal, 6, 100048.

Karanikolas, N., Manga, E., Samaridi, N., Tousidou, E., & Vassilakopoulos, M. (2023). Large language models versus natural language understanding and generation. In Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics (pp. 278-290).

Jin, M., Yu, Q., Zhang, C., Shu, D., Zhu, S., Du, M., ... & Meng, Y. (2024). Health-LLM: Personalized retrieval-augmented disease prediction model. arXiv preprint arXiv: 2402.00746.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T.,

Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). ***ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences*****, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274**

Kaplan, A., Keim, J., Schneider, M., Koziolek, A., & Reussner, R. (2024). Combining Knowledge Graphs and Large Language Models to Ease Knowledge Access in Software Architecture Research.

Klos, M. C., Escoredo, M., Joerin, A., Lemos, V. N., Rauws, M., & Bunge, E. L. (2021). ***Artificial Intelligence Chatbot for Anxiety and Depression in University Students: a Pilot Randomized Controlled Trial*****.**

Koivisto, M. (2021). ***Virtual Learning Companion: A Review of Chatbots in Education*****.**

Lalwani, T., Bhalotia, S., Pal, A., Bisen, S., & Rathod, V. (2021). ***Implementation of a Chatbot System using AI and NLP*****.**

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

Ma, H., Zhang, C., Bian, Y., Liu, L., Zhang, Z., Zhao, P., ... & Wu, B. (2024). Fairness-guided few-shot prompting for large language models. Advances in Neural Information Processing Systems, 36.

Makin, A. (2024). Ontology-Driven Knowledge Management Systems Enhanced by Large Language Models.

Matelsky, J. K., Parodi, F., Liu, T., Lange, R. D., & Kording, K. P. (2023). ***A large language model-assisted education tool to provide feedback on open-ended responses.*** **arXiv preprint arXiv:2308.02439.**

Medrano, J., Castillo, C., & Tejerina, M. (2018). ***Empleo de un chatbot en el aula como herramienta de apoyo al proceso de enseñanza-aprendizaje.***

Micheal, A., Prasanth, A., TS, A., & BL, K. (2024). ***Advancing educational accessibility: The LangChain LLM chatbot's impact on multimedia syllabus-based learning. Research Square.*** **https://doi.org/10.21203/rs.3.rs-4399670/v1**

Modran, H., Bogdan, I. C., Ursuțiu, D., Samoila, C., & Modran, P. L. (2024). LLM intelligent agent tutoring in higher education courses using a RAG approach. Preprints 2024, 2024070519.

Neumann, A. T., Yin, Y., Sowe, S., Decker, S., & Jarke, M. (2024). An LLM-Driven Chatbot in Higher Education for Databases and Information Systems. IEEE Transactions on Education.

Njifenjou, A., Sucal, V., Jabaian, B., & Lefèvre, F. (2024). Role-Play Zero-Shot Prompting with Large Language Models for Open-Domain Human-Machine Conversation. arXiv preprint arXiv:2406.18460.

Olawore, K., McTear, M., & Bi, Y. (2024). Development and Evaluation of a University Chatbot Using Deep Learning: A RAG-Based Approach.

Ortiz de Zárate, J. M.; Dias, J. M.; Avenburg, A. y Gonzalez Quiroga, J. I. (2024). Sesgos algorítmicos y representación social en los modelos de lenguaje generativo (LLM). Fundar

Panagiotidis, P. (2024). LLM-Based Chatbots in Language Learning. European Journal of Education, 7(1), 102-123.

Perak, B., Beliga, S., & Meštrović, A. (2024, June). Incorporating Dialect Understanding Into LLM Using RAG and Prompt Engineering Techniques for Causal Commonsense Reasoning. In Proceedings of the Eleventh Workshop on NLP for Similar Languages, Varieties, and Dialects (VarDial 2024) (pp. 220-229).

Perron, B. E., Rivenburgh, K. A., Victor, B. G., Qi, Z., & Luan, H. (2024). A Primer on Word Embeddings: AI Techniques for Text Analysis in Social Work. arXiv preprint arXiv:2411.07156.

Pesonen, J. (2021). *'Are You OK?' Students' Trust in a Chatbot Providing Support Opportunities*. 10.1007/978-3-030-77943-6_13.

Petukhova, A., Matos-Carvalho, J. P., & Fachada, N. (2024). Text clustering with LLM embeddings. arXiv preprint arXiv:2403.15112.

Prabhune, S., & Berndt, D. J. (2024). Deploying Large Language Models With Retrieval Augmented Generation. arXiv preprint arXiv:2411.11895.

Prakash, K., Rao, S., Hamza, R., Lukich, J., Chaudhari, V., & Nandi, A. (2024). Integrating LLMs into database systems education. *Proceedings of the 2024 ACM SIGCSE Technical Symposium on Computer Science Education*.

Rafat, M. I. (2024). AI-powered Legal Virtual Assistant: Utilizing RAG-optimized LLM for Housing Dispute Resolution in Finland.

Ramdurai, B. (2025). Large Language Models (LLMs), Retrieval-Augmented Generation (RAG) systems, and Convolutional Neural Networks (CNNs) in Application systems. International Journal of Marketing and Technology, 15(01).

Rathod, P. Anurag (2024). Review of Word Embeddings, their Classification, and use Case in LLM Application. J Artif Intell Mach Learn & Data Sci 2024, 2(1), 431-433.

Reimers, N. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084.

Saado, H. (2024). Adressing Domain-specific Knowledge and Terminology using Large Language Models and Word Embeddings.

Salamea-Lopez, R. (2024). Design of AI-enabled Chatbot (Doctoral dissertation, WORCESTER POLYTECHNIC INSTITUTE).

Shrager, J. (2024). ELIZA Reinterpreted: The world's first chatbot was not intended as a chatbot at all. arXiv preprint arXiv:2406.17650.

Shrestha, R., Zou, Y., Chen, Q., Li, Z., Xie, Y., & Deng, S. (2024). FairRAG: Fair human generation via fair retrieval augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11996-12005).

Shum, H. Y., He, X. D., & Li, D. (2018). From Eliza to XiaoIce: challenges and opportunities with social chatbots. Frontiers of Information Technology & Electronic Engineering, 19, 10-26.

Stoica, I., Zaharia, M., Gonzalez, J., Goldberg, K., Zhang, H., Angelopoulos, A., ... & Davis, J. Q. (2024). Specifications: The missing link to making the development of llm systems an engineering discipline. arXiv preprint arXiv:2412.05299.

Suri, M., Mathur, P., Dernoncourt, F., Goswami, K., Rossi, R. A., & Manocha, D. (2024). VisDoM: Multi-Document QA with Visually Rich Elements Using Multimodal Retrieval-Augmented Generation. arXiv preprint arXiv:2412.10704.

Tajz, S. (2024). Primjena velikih jezičnih modela u kreiranju personaliziranih planova putovanja temeljenih na činjenicama (Doctoral dissertation, University of Rijeka. Faculty of Informatics and Digital Technologies).

Vanichvasin, P. (2024). Impact of chatbots on student learning and satisfaction in the entrepreneurship education programme in higher education context. *International Education Studies, 15*(6), 15. https://doi.org/10.5539/ies.v15n6p15

Veturi, S., Vaichal, S., Jagadheesh, R. L., Tripto, N. I., & Yan, N. (2024). Rag based question-answering for contextual response prediction system. arXiv preprint arXiv:2409.03708.

Vijayakumar, M. (2021). *A Study on Chatbots and Virtual Assistants in Customer Engagement: A Review*.

Wang, Y., Ren, R., Li, J., Zhao, W. X., Liu, J., & Wen, J. R. (2024). REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering. arXiv preprint arXiv:2402.17497.

Weissburg, I., Anand, S., Levy, S., & Jeong, H. (2024). LLMs are Biased Teachers: Evaluating LLM Bias in Personalized Education. arXiv preprint arXiv:2410.14012.

Wu, S., Xiong, Y., Cui, Y., Wu, H., Chen, C., Yuan, Y., ... & Xue, C. J. (2024). Retrieval-Augmented Generation for Natural Language Processing: A Survey. arXiv preprint arXiv:2407.13193.

Yue, X., Tian, X., Lu, L., Zhang, M., Wu, Z., & Li, H. (2024). CoAVT: A Cognition-Inspired Unified Audio-Visual-Text Pre-Training Model for Multimodal Processing. arXiv preprint arXiv:2401.12264.

Zadeh, R. B., & Goel, A. (2013). Dimension Independent Similarity Computation. Journal of Machine Learning Research, 14(6).

Zeng, S., Zhang, J., He, P., Ren, J., Zheng, T., Lu, H., ... & Tang, J. (2024). Mitigating the privacy issues in retrieval-augmented generation (rag) via pure synthetic data. arXiv preprint arXiv:2406.14773.

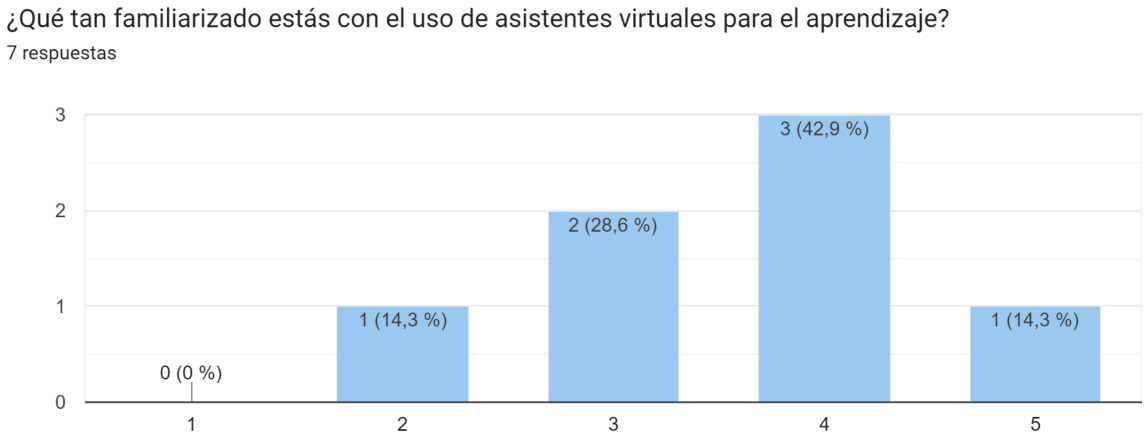
Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., ... & Perrault, R. (2021). The AI index 2021 annual report. arXiv preprint arXiv:2103.06312.

Zhang, X., Xia, M., Couturier, C., Zheng, G., Rajmohan, S., & Ruhle, V. (2023). Hybrid Retrieval-Augmented Generation for Real-time Composition Assistance. https://doi.org/10.48550/arXiv.2308.04215

ANEXOS

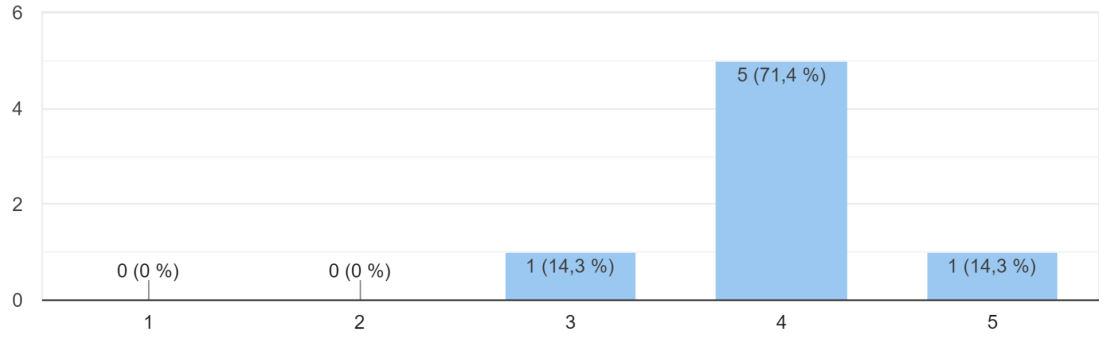
Anexo I: Encuestas realizadas a estudiantes y Lista de cotejo

Primer encuesta:



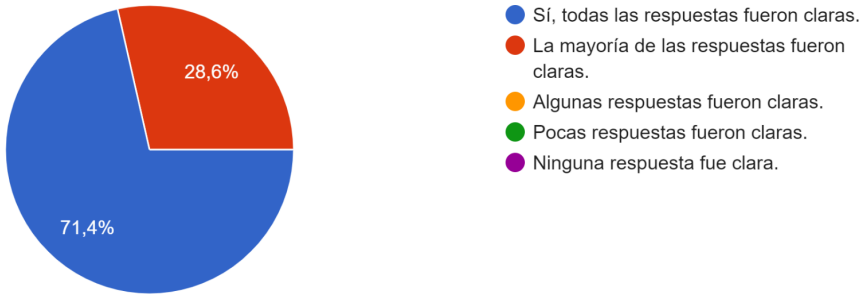
En una escala de 1 a 5, ¿qué tan fácil te resultó navegar e interactuar con el chatbot?

7 respuestas



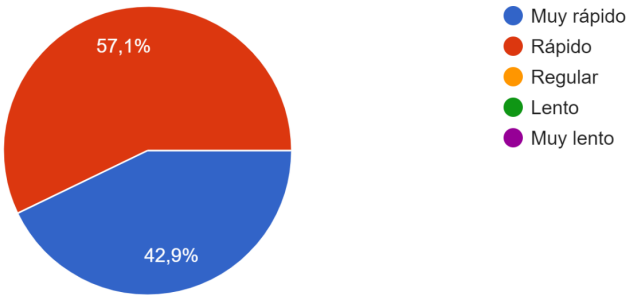
¿El chatbot proporcionó respuestas claras y relevantes a tus preguntas?

7 respuestas



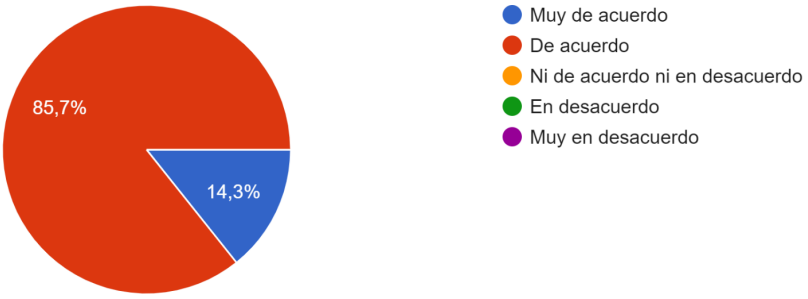
¿Qué tan rápido consideras que el chatbot responde a tus preguntas?

7 respuestas



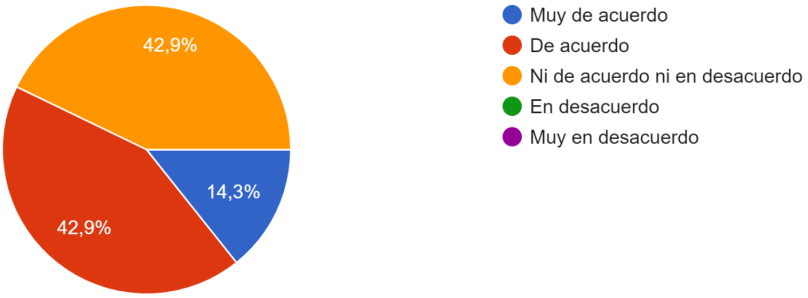
¿Crees que el chatbot ha entendido tus preguntas correctamente?

7 respuestas



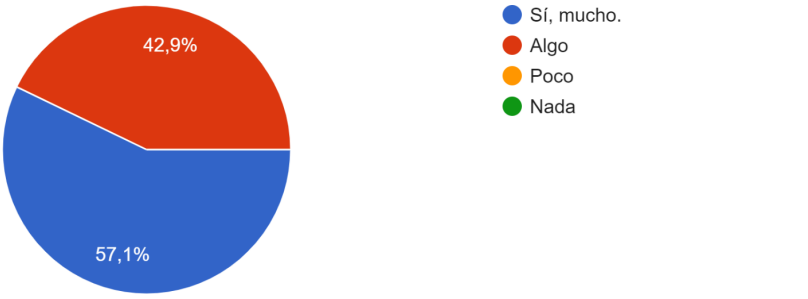
Me ha resultado fácil formular preguntas al chatbot

7 respuestas



¿Consideras que el chatbot agrega valor a tu experiencia de aprendizaje?

7 respuestas



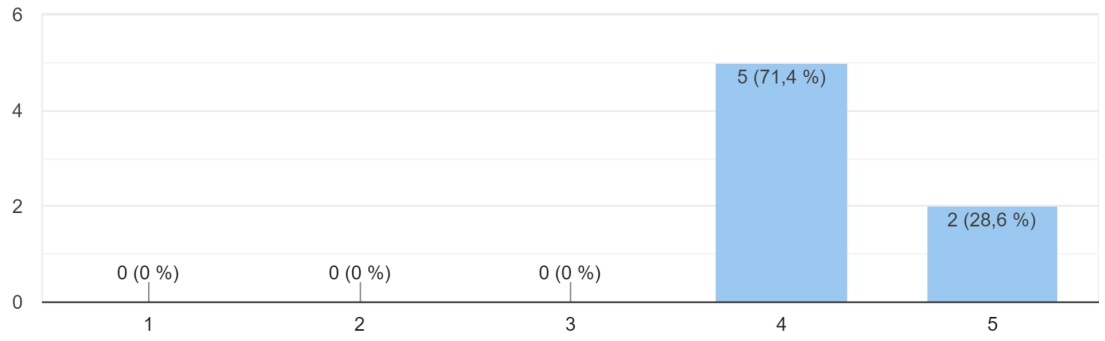
¿Tienes alguna sugerencia sobre cómo mejorar la funcionalidad o la experiencia con el chatbot?

7 respuestas

- Agregar más carácter para el apartado de consulta, pero de ahí trabaja muy bien
- Que se le pueda repreguntar después que de una respuesta
- Sería mejor que en Soporte para Ejercicios se pudiera Pegar lo copiado y que se pueda enviar más carácteres.
- Que se pueda interactuar mas acerca de una sola pregunta, ej. Si yo le pregunto algo y me responde de forma no muy clara, perirle que se explique mas algo, o que cambie ciertos aspectos o que me pueda dar una forma mejor. Basicamente como generar una conversacion y no un "pregunta/respuesta"
- Se podría mejorar agregando mas cantidad para poner en la pregunta y no solo 50 caracteres.
- Que el rango de caracteres sea más amplio
- Podría mejorar agregando caracteres a las preguntas y no solo 50.

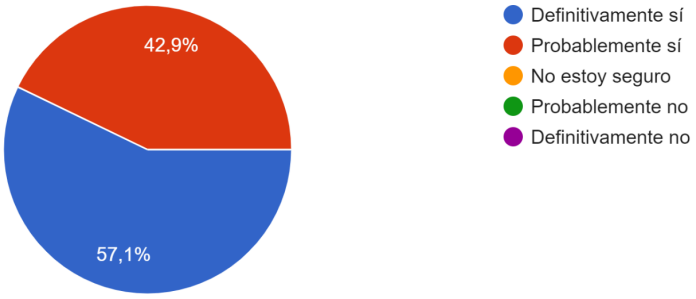
En una escala de 1 a 10, ¿cómo calificarías tu experiencia general con el chatbot?

7 respuestas



¿Volverías a utilizar este chatbot en el futuro para resolver tus consultas?

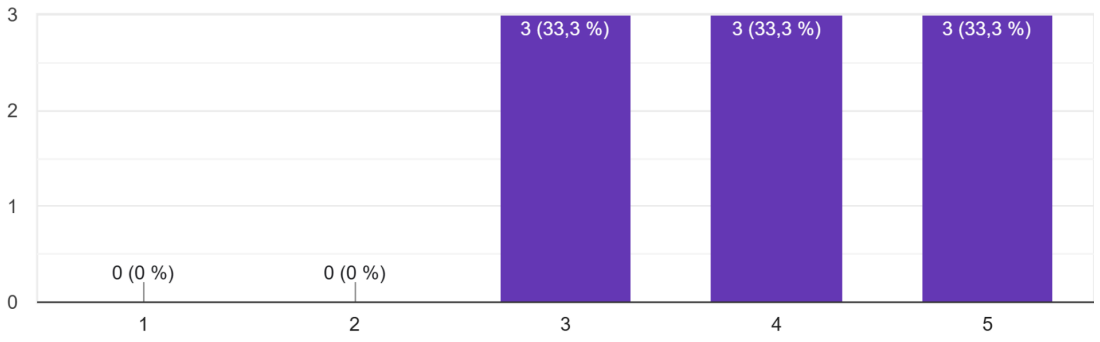
7 respuestas



Segunda encuesta:

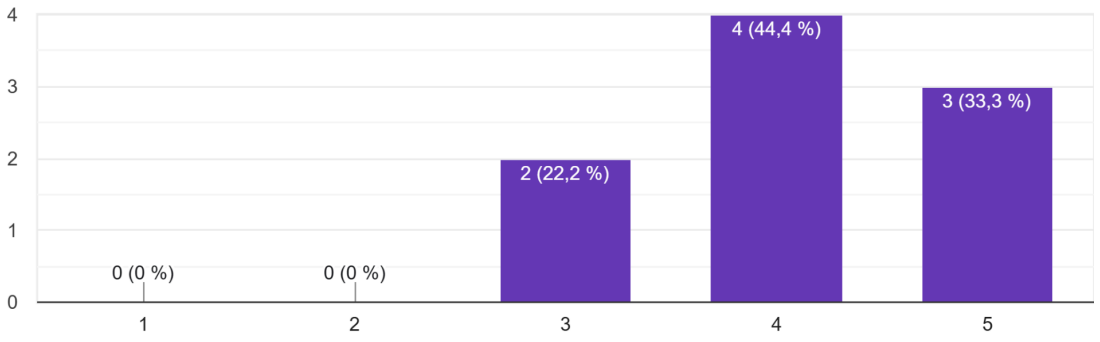
En una escala de 1 a 5, ¿qué tan fácil te resultó navegar e interactuar con el chatbot?

9 respuestas



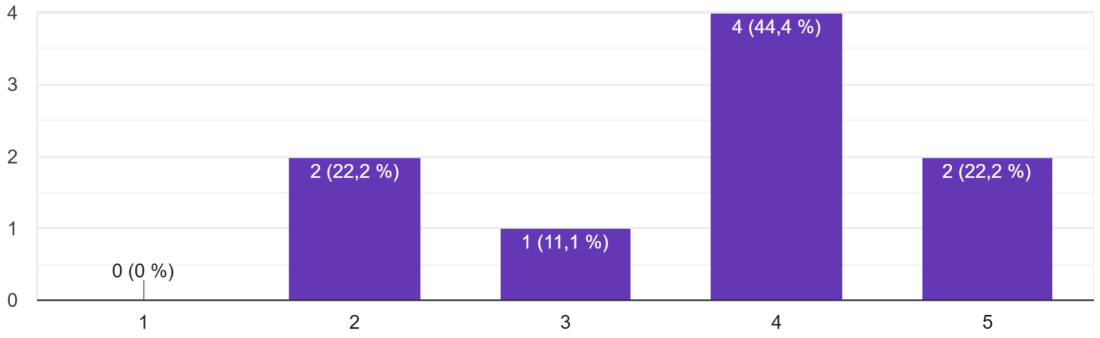
El chatbot te proporcionó explicaciones claras sobre los conceptos de C

9 respuestas



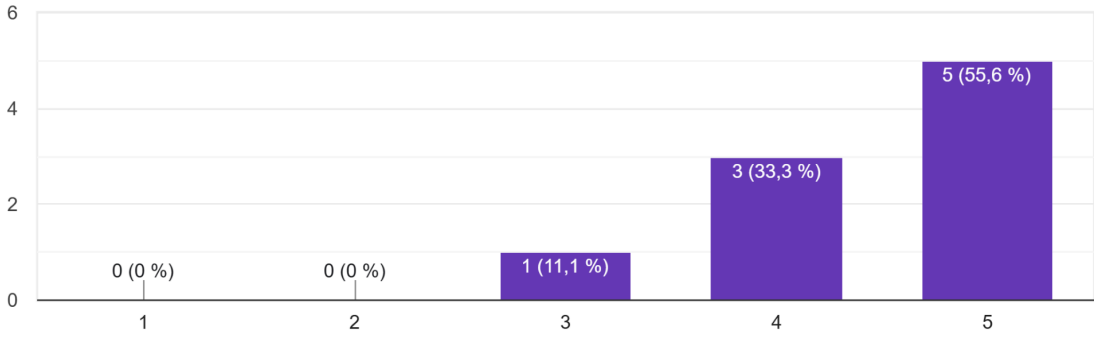
El uso de contraste en la interfaz es la adecuada.

9 respuestas



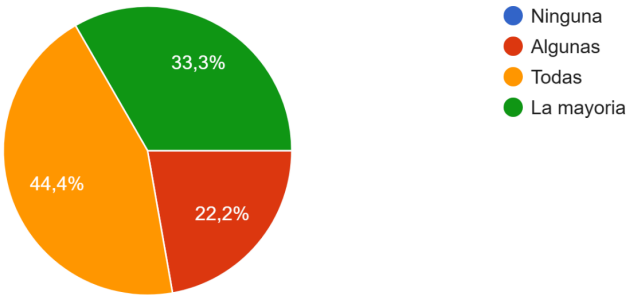
El tamaño de la fuente en la interfaz se adapta bien para facilitar la lectura

9 respuestas



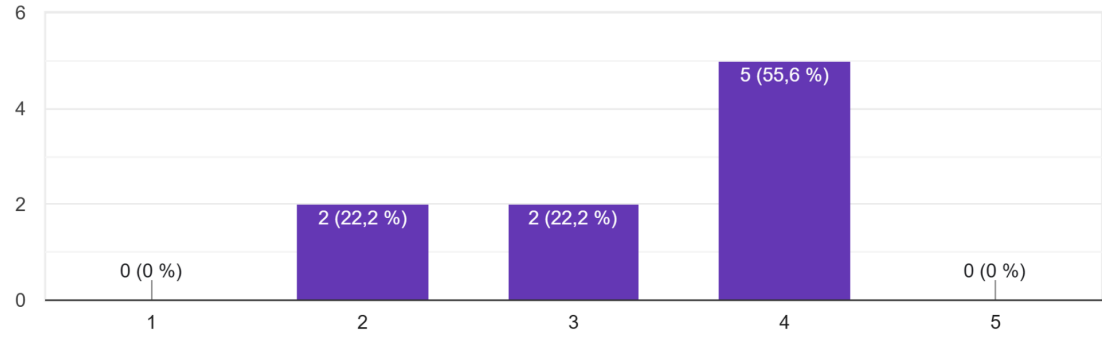
¿Crees que el chatbot ha entendido tus preguntas correctamente?

9 respuestas



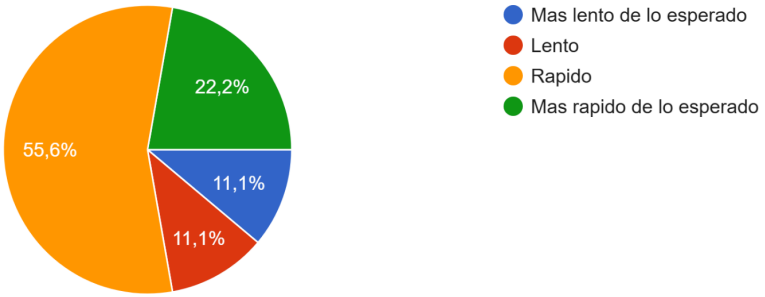
El chatbot ha ayudado a mejorar mi comprensión de la programación en C

9 respuestas



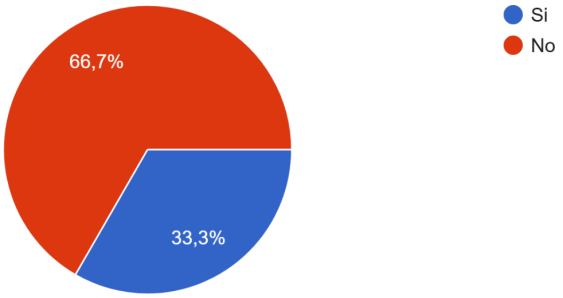
¿Qué tan rápido consideras que el chatbot responde a tus preguntas?

9 respuestas



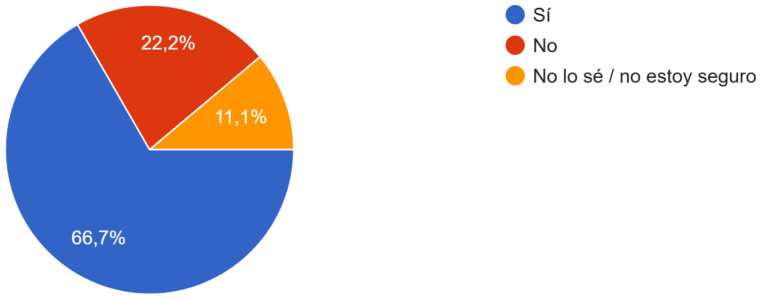
¿Abandonaste conversaciones debido a problemas de comprensión del Chatbot?

9 respuestas



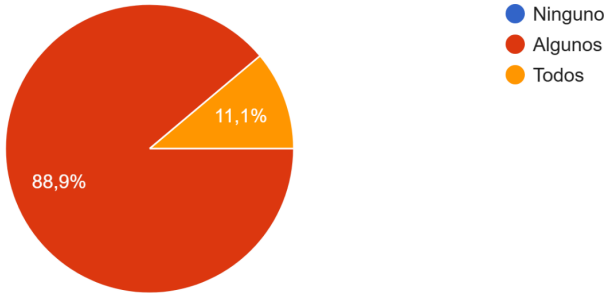
¿Te sentiste capaz de identificar y corregir tus errores después de usar el chatbot?

9 respuestas



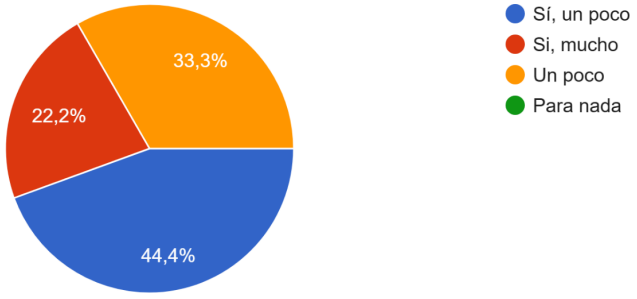
¿Cuántos ejercicios, de los que te propusiste a hacer, pudiste resolver con la ayuda del chatbot?

9 respuestas



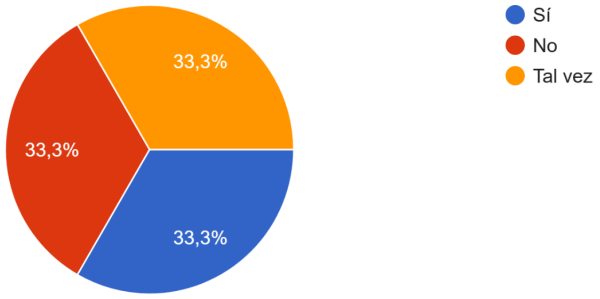
¿El chatbot te permitió aprender algo nuevo de los temas abordados ?

9 respuestas



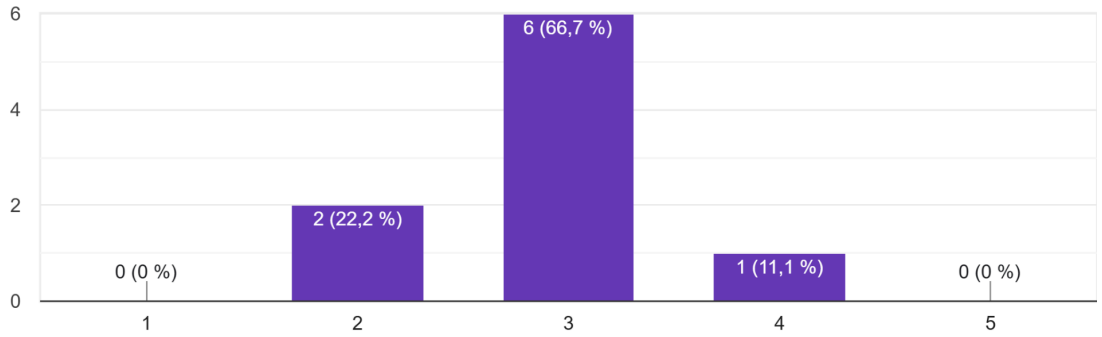
¿Consideras que el chatbot detectó y te ayudó a corregir errores en el código que generaste?

9 respuestas



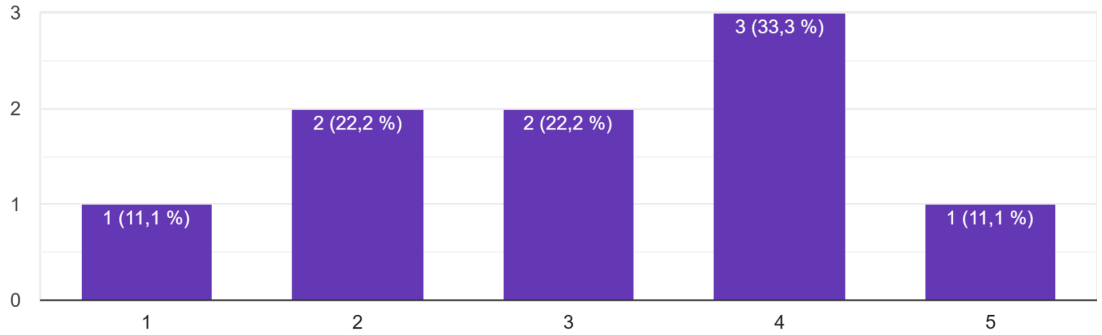
¿Sientes que después de usar el chatbot cometiste menos errores en tus ejercicios?

9 respuestas



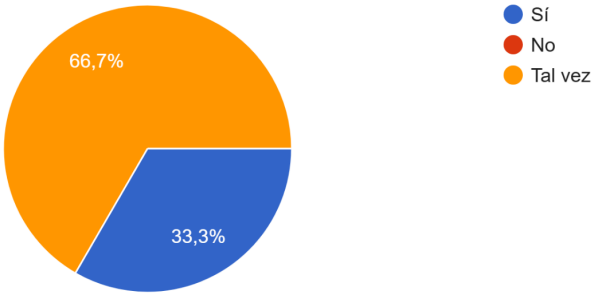
¿El uso del chatbot ha incrementado tu motivación para aprender a programar en C?

9 respuestas



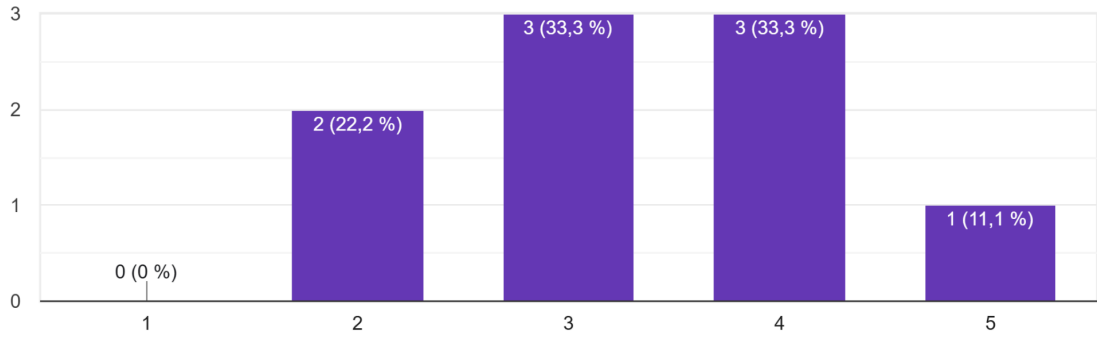
¿Te gustaría utilizar el chatbot voluntariamente en futuras sesiones para aprender programación en C?

9 respuestas



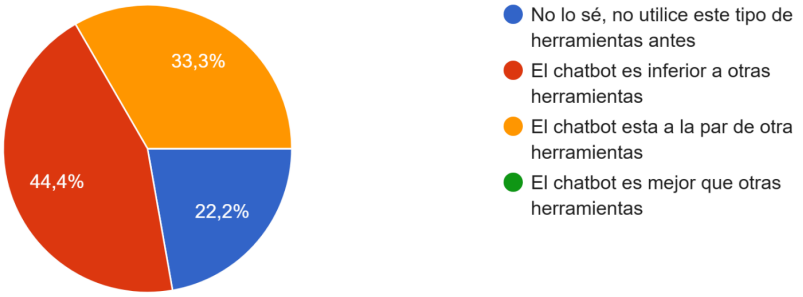
Después de usar el chatbot, pedi ayuda al profesor con menos frecuencia

9 respuestas



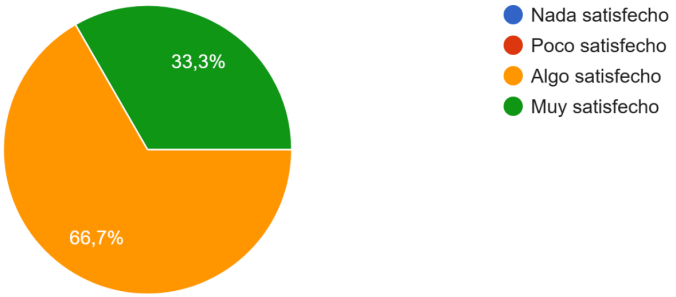
¿Cómo calificarías la experiencia de uso del chatbot en comparación con otras herramientas de aprendizaje de programación?

9 respuestas



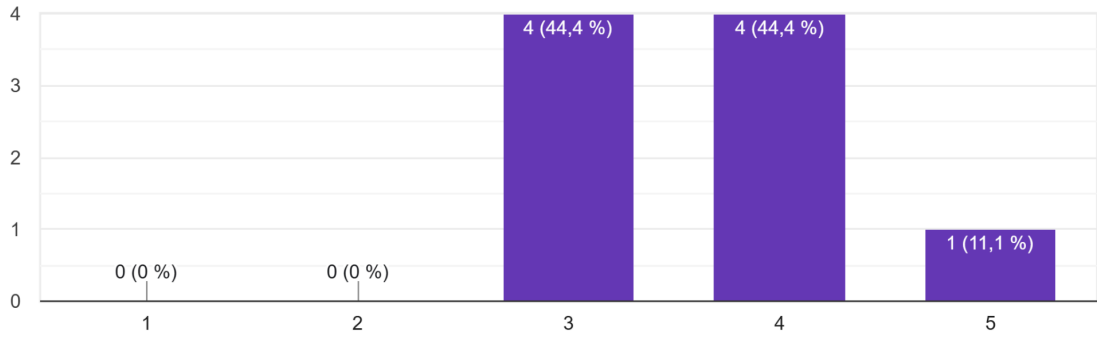
En general, ¿qué tan satisfecho te sientes con la actitud del chatbot hacia tus preguntas y dudas?

9 respuestas



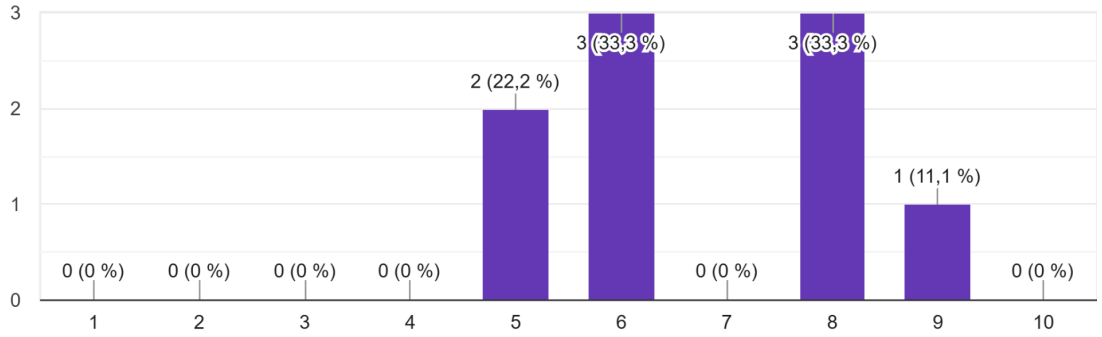
¿Cuál fue el sentimiento general hacia el chatbot después de usarlo?

9 respuestas



En una escala de 1 a 10, ¿cómo calificarías tu experiencia general con el chatbot?

9 respuestas



¿Tienes alguna sugerencia sobre cómo mejorar la funcionalidad o la experiencia con el chatbot?

5 respuestas

ampliar caracteres maximos y poder preguntarle errores en un codigo que le pases

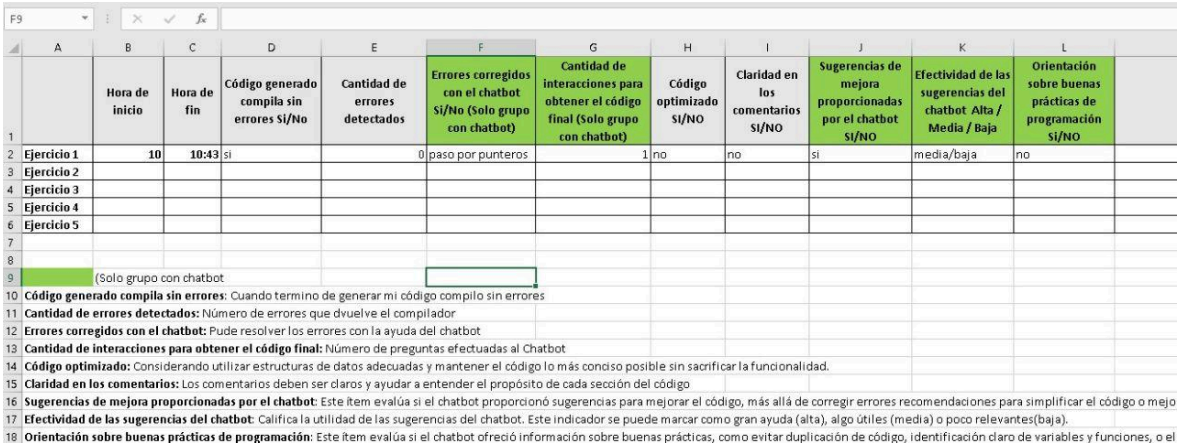
No, estoy bastante conforme con el chatbot me ha ayudado, mucho en mis programas a la hora de poner a prueba algunas dudas.

debido a la poca cantidad de caracteres se me complicaba realizar una pregunta puntual con pocas palabras

aumentar la cantidad de caracteres permitidos para generar preguntas

Los caracteres al preguntar me limita mucho y mas cuando hablamos de la parte en donde se puede poner un fragmento del codigo, observacion: al generar una pregunta para preguntarle con mas claridad genera preguntas con mas de 50 caracteres y no las pude usar en la conversación y el único formato que no me gustó es de la conversación con chat bot los otros estan muy bien logrado.

Lista de cotejo



	A	B	C	D	E	F	G	H	I	J	K	L
1		Hora de inicio	Hora de fin	Código generado compila sin errores Si/No	Cantidad de errores detectados	Errores corregidos con el chatbot Si/No (Solo grupo con chatbot)	Cantidad de interacciones para obtener el código final (Solo grupo con chatbot)	Código optimizado SI/NO	Claridad en los comentarios SI/NO	Sugerencias de mejora proporcionadas por el chatbot SI/NO	Efectividad de las sugerencias del chatbot Alta / Media / Baja	Orientación sobre buenas prácticas de programación Si/NO
2	Ejercicio 1	10	10:43	si	0	paso por punteros	1	no	no	si	media/baja	no
3	Ejercicio 2											
4	Ejercicio 3											
5	Ejercicio 4											
6	Ejercicio 5											
7												
8												
9		(Solo grupo con chatbot										
10	**Código generado compila sin errores**: Cuando termino de generar mi código compilo sin errores											
11	**Cantidad de errores detectados:** Número de errores que dvuelve el compilador											
12	**Errores corregidos con el chatbot:** Pude resolver los errores con la ayuda del chatbot											
13	**Cantidad de interacciones para obtener el código final:** Número de preguntas efectuadas al Chatbot											
14	**Código optimizado:** Considerando utilizar estructuras de datos adecuadas y mantener el código lo más conciso posible sin sacrificar la funcionalidad.											
15	**Claridad en los comentarios:** Los comentarios deben ser claros y ayudar a entender el propósito de cada sección del código											
16	**Sugerencias de mejora proporcionadas por el chatbot**: Este ítem evalúa si el chatbot proporcionó sugerencias para mejorar el código, más allá de corregir errores recomendaciones para simplificar el código o mejo											
17	**Efectividad de las sugerencias del chatbot**: Califica la utilidad de las sugerencias del chatbot. Este indicador se puede marcar como gran ayuda (alta), algo útiles (media) o poco relevantes(baja).											
18	**Orientación sobre buenas prácticas de programación**: Este ítem evalúa si el chatbot ofreció información sobre buenas prácticas, como evitar duplicación de código, identificación claro de variables y funciones, o el											

Anexo II: Épicas y Historias de Usuario

Épica 1: Interacción con el Chatbot

Descripción: Crear una interfaz de chatbot en la que los estudiantes puedan interactuar para resolver dudas y preguntas de programación en lenguaje C.

Historia de usuario	**1.1**
Como	estudiante
Quiero	ingresar mis preguntas o fragmentos de código C,
para	recibir retroalimentación o sugerencias que me ayuden a comprender y solucionar problemas específicos.
Criterios de Aceptación	○ Campo de entrada para escribir consultas o código. ○ Botón de envío para recibir respuestas. ○ El chatbot responde con sugerencias claras o ejemplos de código en C.

Historia de usuario	**1.2**
Como	estudiante

Quiero	que el chatbot considere mi nivel de conocimiento
para	que las respuestas sean comprensibles y adecuadas a mi nivel de experiencia.
Criterios de Aceptación	○ Opciones para ajustar el nivel de respuesta del chatbot (básico, intermedio, avanzado). ○ Feedback que utilice terminología y ejemplos de acuerdo al nivel seleccionado.

Épica 2: Recursos Adicionales de Aprendizaje

Descripción:
Proporcionar una sección de recursos donde los estudiantes puedan acceder a documentación, videos y otros materiales complementarios.

Historia de usuario	2.1
Como	estudiante
Quiero	acceder a documentación técnica y ejemplos de código,

para	reforzar mi comprensión de los conceptos básicos y avanzados de C.
Criterios de Aceptación	○ Enlaces a documentación de funciones y bibliotecas de C. ○ Sección organizada para acceder a recursos técnicos de C.

Historia de usuario	**2.2**
Como	estudiante
Quiero	tener acceso a videos tutoriales sobre conceptos de C,
para	poder ver explicaciones visuales cuando necesite apoyo adicional.
Criterios de Aceptación	■ Enlaces a videos relevantes. ■ Videos categorizados según temas clave de programación en C.

Historia de usuario	**2.3**
Como	estudiante
Quiero	un acceso rápido a compiladores en línea,
para	probar mis soluciones y ejemplos en tiempo real.
Criterios de Aceptación	○ Enlaces a compiladores en línea confiables (por ejemplo, Replit, JDoodle). ○ Instrucciones breves sobre cómo utilizar estos compiladores desde la interfaz.

Épica 3: Ejercicios de Programación en C

Descripción:
Proporcionar ejercicios prácticos en lenguaje C para que los estudiantes puedan practicar sus habilidades de programación.

Historia de usuario	**1.1**
Como	estudiante
Quiero	acceder a una lista de ejercicios prácticos de C,

para	poder practicar y mejorar mis habilidades de programación.
Criterios de Aceptación	○ Lista de ejercicios clasificados por dificultad y tema. ○ Instrucciones claras y detalladas para cada ejercicio.

Épica 4: Autoevaluación y Retroalimentación

Descripción:
Implementar una sección de autoevaluación para que los estudiantes puedan poner a prueba sus conocimientos a través de quizzes y recibir retroalimentación inmediata.

Historia de usuario	**4.1**
Como	estudiante
Quiero	realizar una autoevaluación sobre temas de programación en C,
para	comprobar mi comprensión y aprendizaje.
Criterios de Aceptación	○ Sección de evaluación con preguntas de opción múltiple y resultados instantáneos. ○ Feedback inmediato con explicación de las

	respuestas correctas e incorrectas..

Historia de usuario	**4.2**
Como	estudiante
Quiero	recibir un resumen de mis respuestas y aciertos,
para	poder identificar áreas en las que necesito mejorar.
Criterios de Aceptación	○ Resumen de resultados al finalizar la autoevaluación. ○ Botón para reiniciar la evaluación si el estudiante desea practicar nuevamente.

Criterios de éxito

- Los estudiantes encuentran una interfaz intuitiva y fácil de navegar.
- Los estudiantes pueden resolver problemas en C con la ayuda del chatbot.
- Los recursos adicionales proporcionan material de apoyo suficiente para el aprendizaje autónomo.
- Las autoevaluaciones proporcionan una retroalimentación útil y ayudan en el proceso de aprendizaje.

ANEXO III: TAREAS

Tareas por Historia de Usuario

Épica 1: Interacción con el Chatbot

1. **Historia de Usuario 1.1**
    - Crear un campo de entrada de texto para preguntas y fragmentos de código en la interfaz.
    - Desarrollar un botón de envío para activar la respuesta del chatbot.
    - Configurar la lógica del chatbot para responder con sugerencias o ejemplos de código en C.
2. **Historia de Usuario 1.2**
    - Implementar opciones para que el usuario seleccione su nivel de experiencia (básico, intermedio, avanzado).
    - Ajustar el feedback del chatbot en función del nivel seleccionado (terminología y ejemplos adaptados).

Épica 2: Recursos Adicionales de Aprendizaje

1. **Historia de Usuario 2.1**
    - Crear enlaces a documentación de funciones y bibliotecas de C en una sección dedicada.
    - Organizar la sección para facilitar el acceso a documentación técnica y de recursos de C.
2. **Historia de Usuario 2.2**
    - Añadir enlaces a videos tutoriales organizados por temas de programación en C.
    - Clasificar los videos por temas clave y añadir una breve descripción para cada categoría.
3. **Historia de Usuario 2.3**
    - Agregar enlaces a compiladores en línea recomendados (por ejemplo, Replit, JDoodle).
    - Incluir instrucciones breves para el uso de cada compilador en línea desde la interfaz de la aplicación.

Épica 3: Ejercicios de Programación en C

1. **Historia de Usuario 3.1**
    - Desarrollar una lista de ejercicios prácticos de C clasificados por dificultad (básico, intermedio, avanzado) y tema.

- Proveer instrucciones detalladas y claras para cada ejercicio.

Épica 4: Autoevaluación y Retroalimentación

1. **Historia de Usuario 4.1**
   - Crear una sección de evaluación con preguntas de opción múltiple relacionadas con temas de C.
   - Configurar retroalimentación inmediata con explicación de las respuestas correctas e incorrectas al finalizar cada pregunta.
2. **Historia de Usuario 4.2**
   - Desarrollar un resumen de resultados que muestre las respuestas correctas e incorrectas.
   - Implementar un botón de reinicio para que el estudiante pueda repetir la evaluación.

Tabla de Asignación de Tareas por Historia de Usuario

Historia de Usuario	Tarea
Historia de Usuario 1.1	- Crear campo de entrada de texto. - Desarrollar botón de envío. - Configurar la lógica del chatbot.
Historia de Usuario 1.2	- Implementar selección de nivel de experiencia. - Ajustar feedback del chatbot según el nivel.
Historia de Usuario 2.1	- Crear enlaces a documentación técnica. - Organizar sección de acceso a documentación técnica.
Historia de Usuario 2.2	- Añadir enlaces a videos tutoriales. - Clasificar vídeos por temas clave.

Historia de Usuario 2.3	- Agregar enlaces a compiladores en línea. - Incluir instrucciones de uso para cada compilador en línea.
Historia de Usuario 3.1	- Desarrollar lista de ejercicios prácticos clasificados. - Proveer instrucciones detalladas para cada ejercicio.
Historia de Usuario 4.1	- Crear sección de evaluación con preguntas de opción múltiple. - Configurar feedback inmediato.
Historia de Usuario 4.2	- Desarrollar resumen de resultados al finalizar la evaluación. - Implementar botón de reinicio de evaluación.

Tareas Generales para la Aplicación

1. **Preparar el documento RAG**
   - Crear y estructurar el documento de preguntas y respuestas sobre programación en C.
   - Asegurar que el documento esté en un formato accesible y compatible con el sistema de RAG.
2. **Configurar el Repositorio de Código**
   - Crear y organizar el repositorio en GitHub (o en otra plataforma de control de versiones).
   - Agregar archivos `.gitignore` para proteger información sensible (por ejemplo, claves de API).
3. **Armar los Prompts para el Chatbot**
   - Diseñar prompts específicos para cada tarea.
   - Probar y ajustar los prompts para asegurar que las respuestas del chatbot sean precisas y comprensibles.

ANEXO IV: MÉTRICAS

Modelo de Calidad para Chatbot Educativo en Lenguaje C

Conceptos calculables:

- Interfaz
- Interacción Usuario
- Funcionalidad
- Evaluación del Aprendizaje
- Experiencia del Usuario

Modelo de Conceptos (Árbol de Características):

1. **Interfaz (IG)**
    - **1.1 Facilidad de uso**
        - **Métrica 1**: Promedio de calificación en la facilidad de navegación e interacción con el chatbot (escala 1 a 5).
        - **Tipo de métrica**: Indirecta
        - **Escala:** Ordinal (valores de 1 a 5 indican el nivel de facilidad de uso en orden creciente)
        - **Unidad de medida**: Porcentaje (%), dado que el indicador busca el porcentaje de usuarios que calificaron con 4 o 5
        - **Tipo de indicador**: Elemental
        - **Fórmula de cálculo**: Porcentaje=(Número de usuarios que califican con 4 o 5/ Número total de usuarios encuestados)×100
        - **Fuente de los datos de entrada**: Encuesta
    - **1.2 Accesibilidad**
        - **Métrica 1**: Calidad del contraste de colores en la interfaz
        - **Tipo de métrica**: Directa

- **Escala:**Ordinal (1 a 5, indicando la percepción de calidad del contraste en la interfaz)
- **Unidad de medida**: Promedio (de las calificaciones de contraste)
- **Tipo de indicador**: Elemental
- **Fórmula de cálculo**: Promedio = Suma de calificaciones del contraste de color/Numero total de usuarios encuestados
- **Fuente de los datos de entrada**:Encuesta

- **Métrica 2**: Adaptabilidad del tamaño de fuente en la interfaz
- **Tipo de métrica**: Directa
- **Escala:** Ordinal (1 a 5, indicando la percepción de adaptabilidad del tamaño de fuente)
- **Unidad de medida**: Promedio (de las calificaciones de adaptabilidad de la fuente)
- **Tipo de indicador**: Elemental
- **Fórmula de cálculo:** Promedio = Suma de calificaciones de adaptabilidad de fuente/Número total de usuarios encuestados
- **Fuente de los datos de entrada:**Encuesta

2. **Interacción Usuario**
   - **2.1 Comunicación**
     - **Métrica 1**: Promedio de calificación en la claridad de las explicaciones sobre los conceptos de C.
     - **Tipo de métrica**: Indirecta
     - **Escala:**Ordinal (1 a 5, indicando nivel de claridad)
     - **Unidad de medida**: Promedio
     - **Tipo de indicador:** Elemental
     - **Fórmula de cálculo:** Promedio = Suma de calificaciones de claridad en explicaciones/Número total de usuarios encuestados

- **Fuente de los datos de entrada:** Encuesta

- **Métrica 2**: Calificación en la rapidez de respuesta del chatbot (escala 1 a 5).
- **Tipo de métrica:** Indirecta
- **Escala:** Ordinal (1 a 5, en orden de rapidez de respuesta)
- **Unidad de medida**: Promedio
- **Tipo de indicador:** Elemental
- **Fórmula de cálculo**:Promedio = Suma de calificaciones de rapidez de respuesta/Número total de usuarios encuestados
- **Fuente de los datos de entrada**:Encuesta

- **Métrica 3**: Porcentaje de usuarios que calificaron con 4 o 5 la comprensión de sus preguntas por parte del chatbot.
- **Tipo de métrica:** Indirecta
- **Escala:** Ordinal (valores de 1 a 5 para comprensión)
- **Unidad de medida:** Porcentaje (% de usuarios con calificación de 4 o 5)
- **Tipo de indicador:** Elemental
- **Fórmula de cálculo:** Porcentaje=(Número total de usuarios encuestados/Nu´mero de usuarios que califican con 4 o 5)×100
- **Fuente de los datos de entrada:**Encuesta

3. **Funcionalidad**
    - **3.1 Diálogo natural**
        - **Métrica 1**: Porcentaje de usuarios que reportaron no abandonar conversaciones por problemas de comprensión del chatbot.
        - **Tipo de métrica:** Directa

- **Escala:** Razón (número de usuarios que abandonaron)
- **Unidad de medida**:Porcentaje
- **Tipo de indicador:** Elemental
- **Fórmula de cálculo:** Porcentaje= (Número de usuarios que abandonaron debido a problemas de comprensión/numero de usuario encuestados) x100
- **Fuente de los datos de entrada:**Encuesta

- **3.2 Soporte en resolución de ejercicios**
  - **Métrica 1**: Eficiencia relativa del chatbot
  - **Tipo de métrica**: Directa
  - **Escala:** Razón
  - **Unidad de medida:** Porcentaje
  - **Tipo de indicador:** Derivado
  - **Fórmula de cálculo:** Eficiencia relativa= abs (Promedio (sin chatbot) - Promedio (con chatbot))/Promedio (sin chatbot) ×100
  - **Fuente de los datos de entrada:**Resultados de ejercicios resueltos por los estudiantes con y sin chatbot

  - **Métrica 2**: Porcentaje de usuarios que calificaron positivamente la detección y corrección de errores en el código generado.
  - **Tipo de métrica**: Indirecta
  - **Escala:** Ordinal (1 a 5 para detección de errores)
  - **Unidad de medida:** Porcentaje de usuarios con calificación de 4 o 5
  - **Tipo de indicador:** Elemental

- **Fórmula de cálculo:** Porcentaje=(Número total de usuarios encuestados/Número de usuarios que califican con "Si" o "Talvez")×100
- **Fuente de los datos de entrada:** Encuesta

4. **Evaluación del Aprendizaje**
   - **4.1 Conocimiento adquirido**
     - **Métrica 1**: Número de estudiantes que reportaron haber aprendido algo nuevo.
     - **Tipo de métrica**: Directa
     - **Escala:** Likert ("Sí, un poco", "Si, mucho" , "Un poco" , "Para nada")
     - **Unidad de medida:** Porcentaje de usuarios que reportan haber aprendido algo nuevo
     - **Tipo de indicador**:Elemental
     - **Fórmula de cálculo**: Porcentaje=(Número de usuarios que reportan aprendizaje "Si un poco" "Si, mucho"/ número de usuario encuestados)×100
     - **Fuente de los datos de entrada:**Encuesta

   - **4.2 Motivación**
     - **Métrica 1**: Número de estudiantes dispuestos a usar el chatbot voluntariamente en el futuro.
     - **Tipo de métrica:** Directa
     - **Escala:** Likert(Sí/No/Tal vez)
     - **Unidad de medida:** Porcentaje de usuarios que usarían el chatbot nuevamente
     - **Tipo de indicador:** Elemental

- **Fórmula de cálculo** Porcentaje=(Número de usuarios que no respondieron "No"/ número de usuario encuestados ×100
- **Fuente de los datos de entrada**: Encuesta

5. **Experiencia del Usuario**
    - **5.1 Satisfacción general**
        - **Métrica 1**: Calificación de la experiencia de uso del chatbot en comparación con otras herramientas de programación
        - **Tipo de métrica**: Directa
        - **Escala:** Ordinal
        - **Unidad de medida:** Porcentaje
        - **Tipo de indicador:** Elemental
        - **Fórmula de cálculo:** Porcentaje = (Suma de calificaciones positivas/Número total de usuarios encuestados) *100
        - **Fuente de los datos de entrada:**Encuesta

El modelo de calidad definido en el anexo permite evaluar el desempeño del chatbot desde diferentes dimensiones clave, alineándose con los aspectos mencionados en la evaluación general del sistema.

En primer lugar, la efectividad del chatbot se mide a través de métricas como la claridad de explicaciones sobre los conceptos de C y la eficiencia en la resolución de ejercicios. Estas métricas permiten determinar si el chatbot cumple con su propósito educativo y si las respuestas generadas son útiles para los estudiantes.

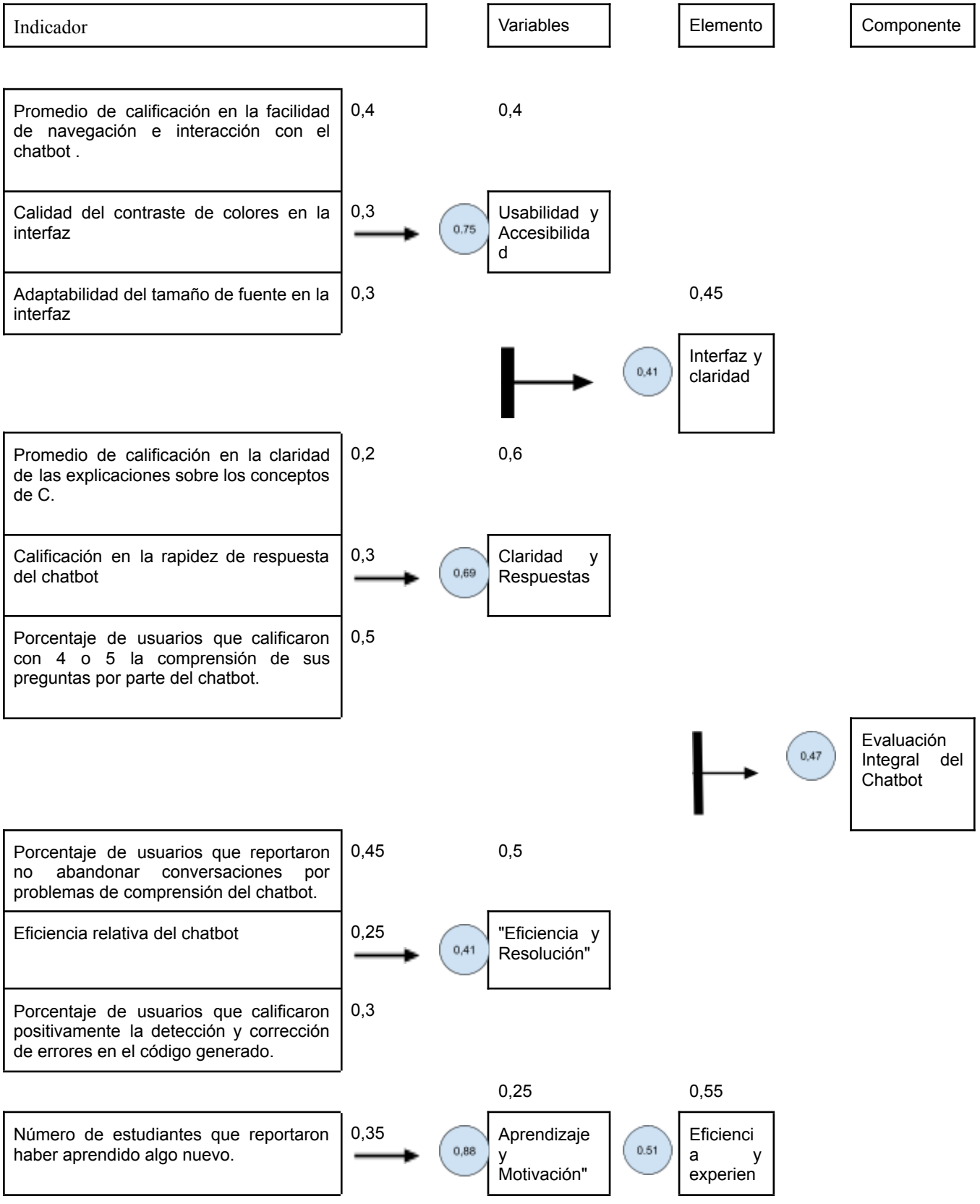
La eficiencia se evalúa mediante indicadores como la rapidez de respuesta y la capacidad del chatbot para facilitar la resolución de tareas, lo que contribuye a optimizar el tiempo y esfuerzo de los usuarios. La métrica sobre abandono de conversaciones por problemas de comprensión también es clave para medir la eficiencia en el soporte brindado.

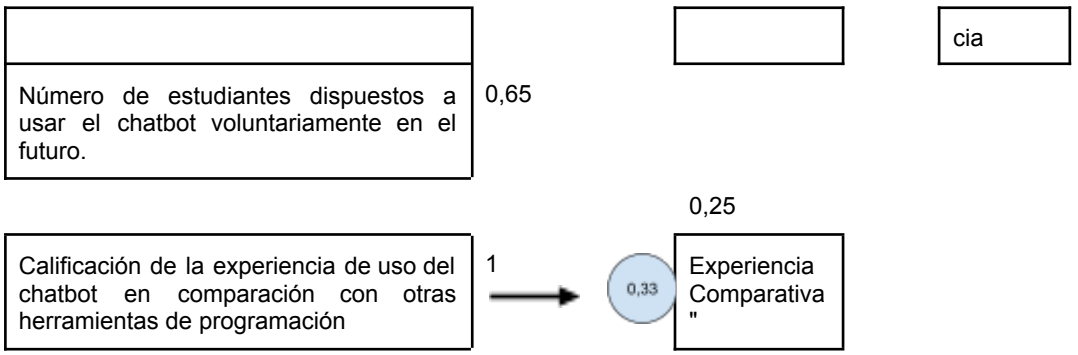
La satisfacción del usuario se refleja en métricas de experiencia, como la facilidad de uso, la accesibilidad y la percepción general de la interacción con el chatbot. Evaluar la navegabilidad y el diseño de la interfaz permite mejorar la usabilidad y garantizar una experiencia fluida para los estudiantes.

La escalabilidad está implícita en la capacidad del chatbot de manejar múltiples interacciones sin degradar su rendimiento. Si bien no hay una métrica directa en el anexo, los tiempos de respuesta y la percepción del usuario pueden dar indicios de su capacidad para adaptarse a una mayor demanda.

Por último, el impacto en el aprendizaje se mide mediante indicadores como la adquisición de conocimientos y la motivación para seguir usando el chatbot. Estas métricas permiten evaluar si el chatbot realmente contribuye al proceso de enseñanza, fomentando un aprendizaje más autónomo y efectivo.

En conclusión, las métricas propuestas permiten evaluar tanto los aspectos técnicos del chatbot como su impacto educativo. Estas mediciones brindan información valiosa para la mejora continua del sistema, asegurando que siga siendo una herramienta útil y eficiente para los estudiantes.


Indicador
Variables
Elemento
Componente
Promedio de calificación en la facilidad de navegación e interacción con el chatbot .
0,4
0,4
Calidad del contraste de colores en la interfaz
0,3
0,75
Usabilidad y Accesibilidad
Adaptabilidad del tamaño de fuente en la interfaz
0,3
0,45
0,41
Interfaz y claridad
Promedio de calificación en la claridad de las explicaciones sobre los conceptos de C.
0,2
0,6
Calificación en la rapidez de respuesta del chatbot
0,3
0,69
Claridad y Respuestas
Porcentaje de usuarios que calificaron con 4 o 5 la comprensión de sus preguntas por parte del chatbot.
0,5
0,47
Evaluación Integral del Chatbot
Porcentaje de usuarios que reportaron no abandonar conversaciones por problemas de comprensión del chatbot.
0,45
0,5
Eficiencia relativa del chatbot
0,25
0,41
"Eficiencia y Resolución"
Porcentaje de usuarios que calificaron positivamente la detección y corrección de errores en el código generado.
0,3
0,25
0,55
Número de estudiantes que reportaron haber aprendido algo nuevo.
0,35
0,88
Aprendizaje y Motivación"
0,51
Eficiencia y experien



Normalización de Métricas

Métrica	Cálculo	Normalizado	Ponderado	Observación
Promedio de calificación en la facilidad de navegación e interacción con el chatbot .	Porcentaje=(Número de usuarios que califican con 4 o 5/ Número total de usuarios encuestados)×100 = (6/9)x100 = 66,66	0,66	0,4*0,66 = 0,264	Una calificación por debajo del 80% indica que existen problemas para navegar por el chatbot y aún existe un espacio para la mejora en cuanto a la claridad de la interfaz.
Calidad del contraste de colores en la interfaz	Promedio = Suma de calificaciones del contraste de color/Numero total de usuarios encuestados = [(2x2) + 3 + (4x4) + (2x5)] /9 = 3,66	3,66 / 5 = 0,732	0,3*0,732 =0,2196	La calificación es positiva , se puede seguir trabajando en aspectos puntuales para mejorar este promedio, pero no es prioritario para futuras mejoras.
Adaptabilidad del tamaño de fuente en la interfaz	Promedio = Suma de calificaciones de adaptabilidad de fuente/Número total de usuarios encuestados =[3 + (3x4) + (5x5)] /9= 4,44	4,44/5 = 0,888	0,3*0,888 =0,2664	La calificación es positiva , se puede seguir trabajando en aspectos puntuales para mejorar este promedio, pero no es prioritario para futuras mejoras.
Promedio de calificación en la claridad de las explicaciones sobre los conceptos de C.	Promedio = Suma de calificaciones de claridad en explicaciones/Número total de usuarios encuestados = (3x2 + 4x4+	0,822	0,5*0,822=0,411	La calificación es positiva , se puede seguir trabajando en aspectos puntuales para mejorar este promedio, es prioritario para futuras entregas mejorar la claridad de las explicaciones.

	5x3)/9 = 4,11			
Calificación en la rapidez de respuesta del chatbot	Promedio = Suma de calificaciones de rapidez de respuesta/Número total de usuarios encuestados = (1+2+5x3+4x2)/9 = 2,88	0,57	0,25*0,57=0,1421	La calificación no es satisfactoria y en parte puede estar sesgada a la infraestructura de los equipos utilizados para trabajar, se debería probar nuevamente con una conexión a Internet más estable.
Porcentaje de usuarios que calificaron con 4 o 5 la comprensión de sus preguntas por parte del chatbot.	Porcentaje=(Número de usuarios que califican con 4 o 5/Número total de usuarios encuestados)×100 = (5/9)x100 = 55,55	0,55	0,25*0,55=0,1375	El resultado es bueno, pero no satisfactorio. Es necesario ajustar los prompts y mejorar la base de conocimiento para incrementar esta calificación.
Porcentaje de usuarios que reportaron no abandonar conversaciones por problemas de comprensión del chatbot.	Porcentaje= (Número de usuarios que abandonaron debido a problemas de comprensión/numero de usuario encuestados) x100 = (3/9) * 100 = 33,33	0,33	0,45*0,33 =0,1485	La calificación no es positiva, es posible mejorar la tasa de retención ampliando la capacidad del chatbot para tener conversaciones más largas sin perder el contexto En esta versión se priorizo el ahorro de tokens ocupados.
Eficiencia relativa del chatbot	Promedio(con chatbot) Promedio sin chatbot)/Promedio sin chatbot *100 =abs(2,13-1,71)/1,71*100=24,5%	0,24	0,25*0,24 = 0,06	La calificación no es positiva pero esperable debido a que la oferta actual de herramientas similares es bastante completa y están en una etapa más avanzada en su desarrollo.
Porcentaje de usuarios que calificaron positivamente la detección y corrección de errores en el código generado.	Porcentaje=(Número de usuarios que califican con “Si” o “Talvez”/Número total de usuarios encuestados)×100 = 6/9 *100 = 66,66	0,66	0,3*0,66 = 0,198	La calificación es positiva dado que no se buscaba activamente detectar/corregir errores, el modelo logró adaptarse .
Número de estudiantes que reportaron haber aprendido algo	Porcentaje=(Número de usuarios que reportan aprendizaje “Si,	0,66	0,35*0,66 = 0,231	El porcentaje obtenido es positivo dado que se tiene en cuenta que los estudiantes ya tienen conocimientos previos.

nuevo.	mucho" "Si, un poco"/ número de usuario encuestados)×100 =6/9 *100 = 66,66			
Número de estudiantes dispuestos a usar el chatbot voluntariamente en el futuro.	Porcentaje=(Número de usuarios que no respondieron "No")/ número de usuario encuestados ×100 = 9/9 * 100 = 100	1	0,65*1 = 0,65	El chatbot fue recibido positivamente lo cual indica que puede aportar valor en la clase.
Calificación de la experiencia de uso del chatbot en comparación con otras herramientas de programación	Porcentaje = (Suma de calificaciones positivas /Número total de usuarios encuestados) *100 = (1x3 + 2x0)/9 * 100 = 33,3	0,33	0,33	La calificación no es satisfactoria pero esperable (al ser la primera versión), las herramientas actuales como Copilot, Gemini o ChatGPT se desempeñan mucho mejor.

Anexo V: Fragmentos del código fuente

Pagina Chatbot.py

```
import os
import streamlit as st
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.document_loaders import TextLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.chat_models import ChatOpenAI
from dotenv import load_dotenv

# Cargar clave API de st.secrets en Streamlit Cloud, o desde .env en local
if "OPENAI_API_KEY" in st.secrets:
    api_key = st.secrets["OPENAI_API_KEY"]
else:
    api_key = os.getenv('OPENAI_API_KEY')

# Verificar si la clave de API está presente
if api_key is None:
    raise ValueError("No se encontró 'OPENAI_API_KEY'. Asegúrate de que esté
definida correctamente en los secretos o en .env.")

# Asignar la clave a las variables de entorno
os.environ["OPENAI_API_KEY"] = api_key

def cargar_documentos():
    loader = TextLoader("unidades_completas.txt", encoding='utf-8')
    documents = loader.load()
    text_splitter = CharacterTextSplitter(chunk_size=500, chunk_overlap=0)
    docs = text_splitter.split_documents(documents)
    embeddings = OpenAIEmbeddings()
    db = FAISS.from_documents(docs, embeddings)
    return db

# Cargar documentos y crear embeddings
db = cargar_documentos()

# Crear el LLM de OpenAI
llm = ChatOpenAI(model_name="gpt-4", temperature=0)
```

```
# Inicializar una variable de sesión para el historial de conversación
if 'chat_history' not in st.session_state:
    st.session_state['chat_history'] = []

def agregar_a_contexto(user_query, respuesta):
    """Agrega la pregunta y respuesta más recientes al historial de conversación."""
    st.session_state['chat_history'].append({"query": user_query, "answer": respuesta})

def obtener_contexto_limited():
    """Recuperar la última interacción para reducir el tamaño de tokens."""
    if len(st.session_state['chat_history']) > 1:
        return [st.session_state['chat_history'][-1]]  # Solo última interacción
    else:
        return st.session_state['chat_history']

def construir_contexto():
    """Construir un contexto compacto basado en la última interacción."""
    historial = obtener_contexto_limited()
    contexto = ""
    for interaction in historial:
        contexto += f"Pregunta: {interaction['query']}\nRespuesta:
{interaction['answer']}\n"
    return contexto

def mostrar():
    st.header("Chatbot - Pregunta sobre programación en C")

    # Seleccionar el nivel de experiencia
    level_of_experience = st.selectbox("Selecciona tu nivel de experiencia:",
                                       options=["Nada experimentado", "Poco experimentado",
"Experimentado"])

    # Mensaje personalizado según el nivel de experiencia
    explanation_modifier = {
        "Nada experimentado": "Proporcione una explicación sencilla con ejemplos
claros.",
        "Poco experimentado": "Explica de forma moderada con algunos detalles
técnicos.",
        "Experimentado": "Asume conocimiento intermedio de C, proporciona respuestas
concisas."
    }

    # Input del usuario para su pregunta
    user_query = st.text_input("Ingresa tu pregunta sobre programación en C:",
key="user_query_input")
```

```
    # Botón para realizar la consulta después de validar el input
    if st.button("Realizar consulta"):
        # Validar que el input no esté vacío y no exceda los 50 caracteres
        if not user_query:
            st.warning("El campo no puede estar vacío. Por favor, ingresa una pregunta.")
        elif len(user_query) > 50:
            st.warning("La pregunta no debe exceder los 50 caracteres.")
        else:
            # Continuar con la ejecución si la validación es exitosa
            # Crear el contexto con las últimas interacciones
            conversation_history = construir_contexto()

            # PromptTemplate que establece el rol del profesor y las restricciones
            prompt_template = """
            Eres un experto en C. Responde solo usando C, sin mencionar C++ ni
características relacionadas.
            {experience_level}

            Documentos relevantes:
            {context}

            Pregunta: {user_query}
            """

            # Crear el PromptTemplate
            prompt = PromptTemplate(
                input_variables=["context", "user_query", "experience_level"],
                template=prompt_template
            )

            try:
                # Realizar búsqueda de documentos relevantes, usando solo 500 caracteres de
cada documento
                docs = db.similarity_search(user_query)

                # Combinar los documentos relevantes en un solo string como contexto (solo
hasta 2 documentos para reducir tokens)
                context = "\n\n".join([doc.page_content[:500] for doc in docs[:2]])

                # Crear la cadena de pregunta-respuesta usando el template
                chain = LLMChain(llm=llm, prompt=prompt)

                # Pasar el contexto (documentos), la consulta del usuario y el nivel de
experiencia al LLM
                answer = chain.run(context=context, user_query=user_query,
```

```
                        experience_level=explanation_modifier[level_of_experience])

            # Si no hay respuesta o es insuficiente, mostrar un mensaje de disculpa
            if not answer.strip():
                answer = "Lo siento, no tengo la información suficiente para responder esa
pregunta. ¿Podrías intentar con otra pregunta?"

            # Formatear la respuesta con tono respetuoso y claro
            final_answer = f"Hola, aquí tienes la respuesta a tu
pregunta:\n\n{answer}\n\nSi tienes más dudas, no dudes en preguntar."

            # Guardar el historial de conversación
            agregar_a_contexto(user_query, final_answer)

            # Mostrar la respuesta
            st.write("Respuesta del profesor asistente:", final_answer)
        except Exception as e:
            st.error(f"Hubo un error al procesar tu pregunta: {e}")

    # Mostrar el historial de conversación en un desplegable
    if st.session_state.chat_history:
        with st.expander("Historial de conversación"):
            for i, interaction in enumerate(st.session_state.chat_history):
                st.write(f"**Consulta {i+1}:** {interaction['query']}")
                st.write(f"**Respuesta {i+1}:** {interaction['answer']}")

    # Nuevo input para generar prompts
    user_topic = st.text_input("Ingresa un tema para generar prompts:",
key="user_topic_input")

    # Botón para generar prompts
    if st.button("Generar Prompt"):
        if user_topic:
            # Sugerencias de prompts bien formateados
            posibles_prompts = [
                f"Explica cómo funcionan los {user_topic} en C.",
                f"¿Cómo se declaran y usan los {user_topic} en C?",
                f"¿Cuáles son los errores comunes al trabajar con {user_topic} en C?",
                f"Muestra un ejemplo de código en C utilizando {user_topic}.",
                f"¿Cómo se relacionan los {user_topic} con la memoria en C?"
            ]

            st.write("Sugerencias de prompts:")
            for prompt in posibles_prompts:
                st.code(prompt)
        else:
```

```
            st.warning("Por favor, ingresa un tema para generar prompts.")

mostrar()
```