Pontificia Universidad Católica Madre y Maestra

Escuela de Ingeniería en Computación y Telecomunicaciones

Facultad de Ciencias de la Ingeniería

Proyecto de Fin de Carrera

FORMULARIO PROPUESTA DE TEMA PARA PROYECTO PARA ISC

Tutor Socrático: Sistema Inteligente de Tutoría Basado en LLM con Metodología Socrática para la Enseñanza de Introducción a la Algoritmia

Cristian Ignacio De La Hoz Reyes & Manuel José Rodríguez Cruz

10149779 & 10150681

Índice

1 Integrantes del grupo

Matrícula	Nombre y apellido	Teléfono	Correo electrónico
10149779	Cristian Ignacio De La Hoz Reyes	+1 (849) 264-6561	CIDR0001@ce.pucmm.edu.do
10150681	Manuel José Rodríguez Cruz	+1 (829) 354-3720	MJRC0002@ce.pucmm.edu.do

2 Título del tema propuesto

Tutor Socrático: Sistema Inteligente de Tutoría Basado en LLM con Metodología Socrática para la Enseñanza de Introducción a la Algoritmia.

3 Identificación de la Problemática

El curso ICC-101-T (Introducción a la Algoritmia) de PUCMM Campus Santiago enfrenta una crisis pedagógica derivada del uso indiscriminado de herramientas de inteligencia artificial generativa por parte de los estudiantes. La disponibilidad pública de sistemas como ChatGPT, Gemini y similares ha creado un patrón de comportamiento donde los estudiantes utilizan estas herramientas principalmente para obtener soluciones completas a sus tareas y exámenes, en lugar de emplearlas como apoyo genuino al aprendizaje.

Esta situación genera tres consecuencias críticas:

3.1 Pereza Metacognitiva

Fan et al. [1] documentan experimentalmente el fenómeno de “metacognitive laziness” (pereza metacognitiva) en estudiantes que utilizan ChatGPT. Su estudio aleatorizado con 117 estudiantes universitarios revela que, aunque ChatGPT mejora significativamente el desempeño en tareas a corto plazo, no impulsa la motivación intrínseca ni la ganancia y transferencia de conocimiento. Los estudiantes que usaron ChatGPT mostraron menos evaluación metacognitiva, monitoreo y orientación, exhibiendo dependencia tecnológica sin desarrollo correspondiente de capacidades cognitivas profundas.

3.2 Crisis de Integridad Académica

La facultad reporta una brecha creciente entre el desempeño en tareas (donde el uso de IA es difícil de detectar) y el desempeño en evaluaciones presenciales. Los estudiantes presentan código que no pueden explicar, evidenciando que la "ayuda" de herramientas comerciales de IA se ha convertido en sustitución completa del proceso de aprendizaje. Esta situación erosiona la validez de las evaluaciones y devalúa las credenciales académicas institucionales.

3.3 Deficiencias en Fundamentos Algorítmicos

ICC-101-T es un curso de entrada sin prerrequisitos de programación, diseñado para desarrollar pensamiento computacional desde cero. Los estudiantes que evitan la "lucha productiva" mediante uso de IA no desarrollan habilidades esenciales como: abstracción de problemas, diseño de invariantes de ciclo, razonamiento sobre casos base recursivos, y análisis de complejidad algorítmica.

4 Justificación

La construcción de un Sistema Tutor Socrático institucional para ICC-101-T se justifica desde tres perspectivas convergentes: necesidad pedagógica urgente, viabilidad técnica comprobada, y alineación estratégica institucional.

4.1 Necesidad Pedagógica: Combatir la Pereza Metacognitiva

El método socrático representa un antídoto directo a este fenómeno porque:

1. Exige articulación explícita de razonamiento, activando procesamiento metacognitivo
2. Expone lagunas en comprensión mediante preguntas penetrantes, creando el desequilibrio cognitivo necesario para aprendizaje profundo
3. Transfiere autoridad epistémica al estudiante, quien debe construir conocimiento mediante razonamiento guiado
4. Promueve monitoreo metacognitivo continuo mediante cuestionamiento reflexivo

4.2 Viabilidad Técnica: Evidencia Empírica

Liffiton et al. [2] desarrollaron CodeHelp, un sistema que utiliza LLMs con "guardrails robustos que están específicamente diseñados para no revelar soluciones directamente mientras ayudan a los estudiantes a resolver sus problemas". Desplegado durante 12 semanas en un curso de 52 estudiantes, CodeHelp fue "bien recibido por los estudiantes".

Feng et al. [3] documentan que sistemas modernos de tutoría basados en EMT "que utilizan diálogo para andamiaje funcionan tan bien como tutores humanos en temas STEM", con tamaños de efecto d=1.0 comparables a tutoría humana individual.

4.3 Ventaja Estratégica: Propiedad Institucional

A diferencia de ChatGPT, Gemini, GitHub Copilot u otras herramientas comerciales, un sistema propiedad de PUCMM proporciona:

- **Alineación de incentivos:** Optimizar para aprendizaje profundo, no satisfacción inmediata del usuario
- **Integración curricular:** Diseño específico para ICC-101-T
- **Datos institucionales:** Control completo de información del aprendizaje
- **Autorización explícita:** Eliminación de ambigüedad sobre integridad académica
- **Control de evolución:** Adaptación basada en evidencia empírica institucional

5 Funcionalidades del Proyecto

5.1 Módulo 1: Interfaz de Diálogo Socrático

- Chat conversacional estructurado con editor de código integrado
- Formulario de contexto estructurado (problema, código, casos de prueba, lo intentado, pregunta)
- Streaming de respuestas en tiempo real
- Historial de conversación persistente

5.2 Módulo 2: Motor de Diálogo Socrático

- Framework EMT (Expectation-Misconception Tailoring)
- Base de conocimiento de expectativas por algoritmo
- Catálogo de misconcepciones comunes
- Patrones de diálogo: rastreo de ejecución, invariantes de ciclo, complejidad, casos límite

5.3 Módulo 3: Sistema de Guardrails Multi-Capa

- Arquitectura de dos agentes (Analista + Filtro Pedagógico)
- Validador anti-solución (verifica ausencia de código completo)
- Límites de asistencia anti-pereza (tiempo mínimo de intento, máximo de pistas)

5.4 Módulo 4: Rastreo de Conocimiento

- Modelo dinámico del conocimiento del estudiante
- Taxonomía de Componentes de Conocimiento (KCs) para ICC-101-T
- Actualización de maestría en tiempo real basada en respuestas

5.5 Módulo 5: Panel de Instructor

- Monitoreo de interacciones estudiantiles
- Identificación automática de estudiantes en dificultad
- Analíticas de aprendizaje (tiempo por tema, tasa de dificultad, progresión)

5.6 Módulo 6: Base de Conocimiento ICC-101-T

- Indexación de contenido curricular (syllabus, materiales, ejemplos)
- Alineación con objetivos de aprendizaje del curso

5.7 Módulo 7: Scaffolding Metacognitivo

- Prompts metacognitivos obligatorios (“¿Por qué crees que...?”)
- Presentación de disfluencia deseable (casos que contradicen expectativas)

5.8 Alcance de MVP (6 meses)

[[CHECK]] Módulos 1, 2, 3: Diálogo socrático con guardrails

[[CHECK]] Módulo 4: Rastreo básico de conocimiento

[[CHECK]] Módulo 5: Panel básico de instructor

[[CHECK]] Módulo 6: Base de conocimiento para 3-4 algoritmos fundamentales

[[CHECK]] Módulo 7: Prompts metacognitivos básicos

6 Modelo de los Procesos de Negocios

6.1 Proceso 1: Onboarding de Estudiante

1. Estudiante accede mediante credenciales institucionales (SSO)
2. Sistema verifica inscripción en ICC-101-T
3. Tutorial interactivo explicando propósito y diferencias vs. ChatGPT
4. Ejercicio guiado de prueba
5. Cuenta activada con estado de conocimiento inicial

6.2 Proceso 2: Sesión de Tutoría Socrática

1. Estudiante selecciona tópico algorítmico
2. Completa formulario de contexto estructurado
3. AGENTE 1 (Analista) procesa código e identifica problemas
4. Sistema consulta modelo de conocimiento
5. Gestor de diálogo determina etapa EMT apropiada
6. AGENTE 2 (Filtro Pedagógico) genera respuesta socrática
7. Validador Anti-Solución verifica ausencia de soluciones completas
8. Respuesta mostrada al estudiante
9. Respuesta del estudiante actualiza modelo de conocimiento

6.3 Proceso 3: Monitoreo por Instructor

Instructor accede a panel que presenta:

- Lista de estudiantes activos
- Alertas de estudiantes que excedieron límites
- Distribución de tópicos solicitados
- Problemas con mayor tasa de dificultad

6.4 Proceso 4: Escalación a Instructor

Cuando estudiante excede límites (3 sesiones o 5 turnos):

1. Sistema genera resumen automático
2. Notifica a instructor con contexto completo
3. Informa al estudiante con expectativas claras

6.5 Integración con Flujo de Curso Existente

- **Semana 1-2:** Onboarding y explicación diferencia vs. ChatGPT
- **Semana 3+:** Estudiantes usan tutor durante desarrollo de contenido
- **Continuo:** Instructor revisa panel y ajusta énfasis en clase

El tutor **NO reemplaza**: clases presenciales, laboratorios, horas de oficina, evaluaciones sumativas.

El tutor **SÍ aumenta**: disponibilidad (24/7), cantidad de práctica con retroalimentación, visibilidad de dificultades, datos institucionales.

7 Revisión Preliminar de Bibliografía y Proyectos Similares Existentes

7.1 Sistemas Tutores Inteligentes: Fundamentos

Feng, Magana y Kao [3] realizaron una revisión sistemática de literatura sobre efectividad de ITS en dominios STEM. Sus hallazgos establecen que ITS muestran tamaños de efecto d=1.0 comparables al impacto de tutoría humana individual. Sistemas basados en Expectation-Misconception Tailoring (EMT) que utilizan diálogo funcionan tan bien como tutores humanos en temas STEM.

7.2 LLMs en Educación de Programación

Hellas et al. [4] evaluaron respuestas de LLMs a solicitudes reales de ayuda de 150 estudiantes principiantes. GPT-3.5 identificó al menos un problema en 90 % de casos, validando utilidad potencial de LLMs. Sin embargo, “las soluciones modelo se proporcionan frecuentemente incluso cuando se le solicita explícitamente al LLM que no lo haga”. Conclusión: ingeniería de prompts sola es insuficiente; se requieren guardrails arquitectónicos.

7.3 CodeHelp: Implementación de Guardrails

Liffiton et al. [2] desarrollaron CodeHelp, directamente relevante como modelo arquitectónico. Implementa: formulario de entrada estructurado, sistema de dos agentes, guardrails automatizados, panel de instructor. Resultados: bien recibido por estudiantes, fácil de desplegar, complementa (no reemplaza) apoyo humano.

7.4 Ruffle&Riley: Generación Automatizada de Tutores

Schmucker et al. [5] proponen generación automática de scripts de tutoría a partir de texto de lección usando marco EMT. Introduce formato de aprendizaje-enseñando con dos agentes. Evaluación: usuarios reportaron calificaciones más altas de comprensión, retención, utilidad percibida.

7.5 Knowledge Tracing con LLMs

Scarlatos, Baker y Lan [6] proponen DialogueKT para rastreo de conocimiento en diálogos tutoriales. Método LLMKT permite identificar componentes de conocimiento en cada turno y predecir desempeño futuro. Relevancia: permite que sistema rastree maestría de KCs específicos de algoritmos y adapte dificultad dinámicamente.

7.6 Limitaciones de Adaptividad en LLMs

Borchers y Shou [7] proporcionan evaluación crítica. Encuentran que incluso modelo mejor sigue marginalmente la adaptividad de ITS tradicionales. GPT-4o "sigue instrucciones confiablemente pero tiende a proporcionar retroalimentación excesivamente directa". Implicación: no depender de capacidades inherentes del LLM; arquitectura explícita necesaria.

7.7 Pereza Metacognitiva: El Problema Central

Fan et al. [1] proporcionan evidencia experimental del fenómeno que motiva centralmente este proyecto. Estudio con 117 estudiantes: ChatGPT mejora desempeño a corto plazo pero no impulsa motivación intrínseca ni ganancia/transferencia de conocimiento. Usuarios mostraron menos evaluación metacognitiva. Concepto: "metacognitive laziness" — IA puede promover dependencia sin aprendizaje profundo.

7.8 Conclusión de la Revisión

La literatura establece firmemente:

[CHECK] Viabilidad técnica de ITS basados en LLMs

[CHECK] Necesidad urgente de guardrails pedagógicos

[CHECK] Efectividad del marco EMT para STEM

[CHECK] Peligro real de pereza metacognitiva con IA generativa

[CHECK] Ventaja de propiedad institucional vs. herramientas comerciales

El proyecto propuesto se fundamenta en evidencia robusta y llena brecha específica en educación algorítmica.

Referencias Bibliográficas

[1] Y. Fan et al., «Beware of Metacognitive Laziness: Effects of Generative Artificial Intelligence on Learning Motivation, Processes, and Performance,» *British Journal of Educational Technology*, págs. 1-42, 2024. DOI: 10.1111/bjet.13544.

[2] M. Liffiton, B. Sheese, J. Savelka y P. Denny, «CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes,» *arXiv preprint arXiv:2308.06921*, 2023.

[3] S. Feng, A. J. Magana y D. Kao, «A Systematic Review of Literature on the Effectiveness of Intelligent Tutoring Systems in STEM,» *IEEE Frontiers in Education Conference*, págs. 1-8, 2021. DOI: 10.1109/FIE49875.2021.9637240.

[4] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää y J. Sorva, «Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests,» en *Proceedings of the 2023 ACM Conference on International Computing Education Research*, 2023, págs. 1-13. DOI: 10.1145/3568813.3600139.

[5] R. Schmucker, M. Xia, A. Azaria y T. Mitchell, «Ruffle&Riley: Towards the Automated Induction of Conversational Tutoring Systems,» *arXiv preprint arXiv:2310.01420*, 2023.

[6] A. Scarlatos, R. S. Baker y A. Lan, «Exploring Knowledge Tracing in Tutor-Student Dialogues using LLMs,» en *Proceedings of the 15th International Learning Analytics and Knowledge Conference*, 2025, págs. 1-11. DOI: 10.1145/3706468.3706501.

[7] C. Borchers y T. Shou, «Can Large Language Models Match Tutoring System Adaptivity? A Benchmarking Study,» *arXiv preprint arXiv:2504.05570*, 2025.