Escuela de Ingeniería en Computación
Sede Central Cartago
Inteligencia Artificial

Trabajo Corto 7 Sistema MultiAgente con RAG

José David Fernández Salas - 2022045079 José Eduardo Gutiérrez Conejo - 2019073558

Tutor de Seguridad de Software Multiagente con RAG

Propuesta técnica:

Este sistema multiagente inteligente tiene como propósito enseñarles a los estudiantes de temas de seguridad de software. Utiliza técnicas de inteligencia artificial combinadas con recuperación aumentada por búsqueda (RAG), para explicaciones claras, mejor apoyo y ejercicios de práctica. Está diseñado bajo una arquitectura modular de agentes especializados que colaboran para cumplir diferentes funciones: búsqueda de información, generación de respuestas educativas y formulación de ejercicios de repaso.

Se utilizan un total de 3 agentes para el proceso de ayuda al estudiante los cuales se muestran a continuación.

Tutor Agent

Rol: Es el agente principal que se encarga de recibir las preguntas del usuario y coordinar la respuesta

Prompt: “Eres un tutor experto en seguridad de software. Responde de forma clara, educativa y paso a paso usando los apuntes como contexto.”

Interacción con otros agentes: Se envía las preguntas al RAG agente el cual analiza la información importante de los apuntes. La otra interacción es con el agente evaluador el cual se encarga de hacer las prácticas.

Tecnología usada: Se utiliza Ollama + modelo Mistral que fue instalado localmente para la generación de respuestas

RAG Agent

Rol: Recupera partes importantes de apuntes vectorizados (PDF) para construir el contexto de respuesta.
Prompt: Este básicamente se pone a analizar las partes importantes y relacionadas del documento y se las devuelve al tutor como: Apuntes relevantes.

Interacción con otros agentes: Atiende las peticiones de los textos más relevantes de las notas.

Uso de RAG: Este agente implementa recuperación semántica con base vectorial (ChromaDB) sobre apuntes de seguridad. Este responde a las preguntas que le hace el agente tutor devolviéndole los datos más relevantes del tema.

Tecnología usada: Utiliza pdfminer.six para extraer el texto de los archivos PDF.
Se implementa LangChain específicamente "Recursive CharacterText Splitter" el cual divide el contenido del PDF en bloques pequeños para su vectorización.
Para la vectorización se crea un embedding para convertir el text en vectores semánticos reales utilizando el modelo mistral.
Después mediante ChromaDB se pueden guardar los vectores y permite hacer búsqueda por similitud semántica. Finalmente se usa una similitud semántica donde con la ayuda de similarity_search(pregunta, k=3), devuelve los fragmentos más parecidos a la pregunta del estudiante.

EvaluatorAgent
Rol: Genera ejercicios de práctica personalizados sobre los temas clave

Prompt utilizado:
"Eres un generador de ejercicios de práctica sobre seguridad de software.\n"
"Debes crear 2 preguntas variadas para estudiantes:\n"
"- 1 de comprensión teórica\n"
"- 1 de aplicación o análisis práctico\n\n"
"Muestra las preguntas numeradas y bien redactadas.\n"
"No incluyas respuestas, solo las preguntas.\n"

Interacción con otros agentes: Recibe temas desde el TutorAgent cuando el usuario utiliza comandos del tipo practica: <tema>.

Tecnología usada: Se implementa nuevamente Ollama con el Modelo Mistral donde en este caso dos instancias de este modelo se van a estar comunicando para generar preguntas prácticas tipo examen. Además se utiliza request.post() el cual es una petición de un prompt especial enfocado en crear ejercicios a Ollama.

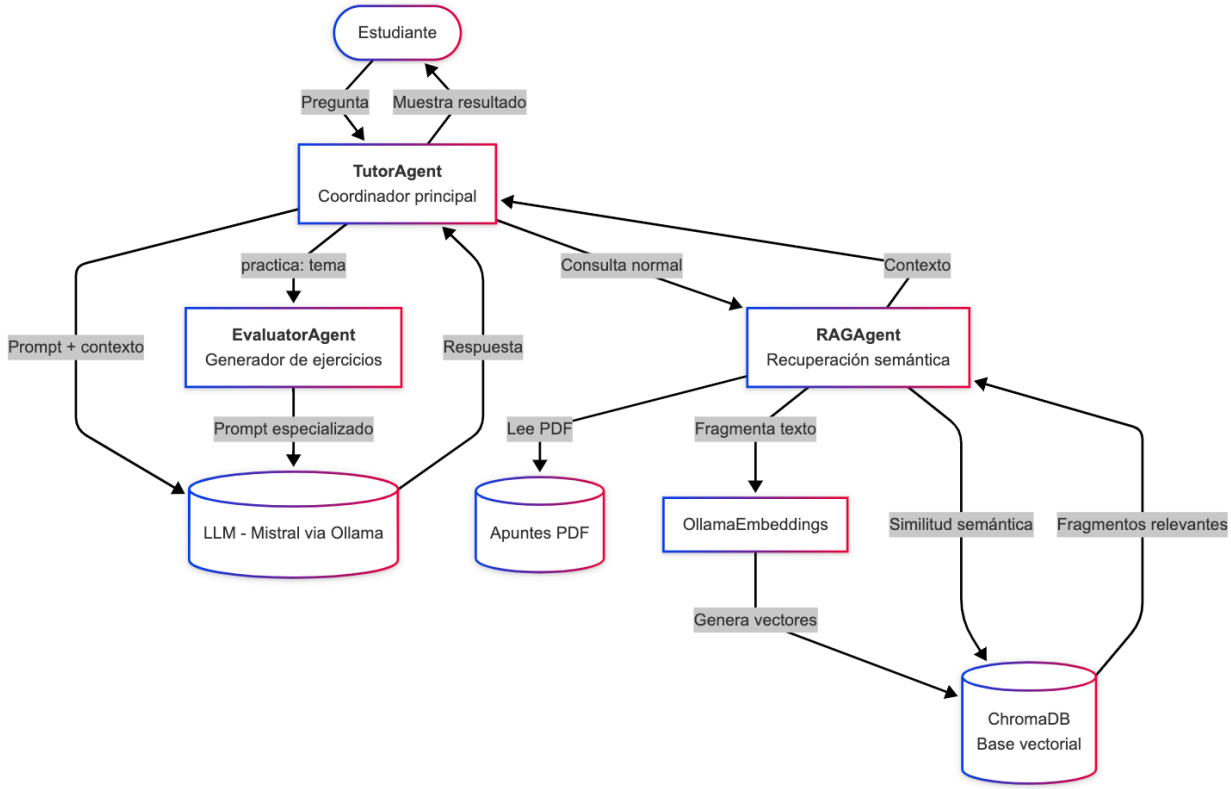
Introducción del RAG
Se conoce como Retrieval-Augmented generation (RAG) que es una arquitectura que combina la recuperación de información desde una base de datos o documentos y mediante un modelo de lenguaje lo convierte en lenguaje natural (Zhao et al, 2024). Como objetivo tiene mejorar la precisión, contextualización y robustez de las respuestas generadas por los modelos de lenguaje. RAG es importante porque permite que los LLM puedan tener más información necesaria para el análisis.

Funcionamiento del RAG

1. Carga y fragmenta el contenido del archivo PDF de apuntes de seguridad utilizando pdfminer.six y RecursiveCharacterTextSplitter.
2. Genera vectores semánticos reales (embeddings) para cada fragmento usando el modelo de lenguaje local Mistral a través de Ollama, mediante el componente OllamaEmbeddings.
3. Al recibir una pregunta del estudiante, el sistema realiza una búsqueda semántica en la base de datos vectorial (ChromaDB).
4. Recupera los fragmentos más similares a la pregunta y los entrega como contexto al TutorAgent.
5. El Tutor usa ese contexto para generar una respuesta educativa y paso a paso mediante IA generativa.

En el caso de la implementación como tutor de seguridad este carga los apuntes mediante un PDF. Se divide en fragmentos donde se crean embeddings con Mistral. Cuando se hace una pregunta el sistema se dedica a buscar los 3 fragmentos más parecidos para que el sistema formule su respuesta. Las ventajas de usar RAG tienen que ver con que se puede utilizar información actualizada y correcta porque al proporcionar los datos se garantiza más confiabilidad. Otra ventaja es la privacidad de las consultas porque al correr local no requiere de conexión. Finalmente esto es muy escalable al permitir leer múltiples documentos en distintos formatos para mejores respuestas.

Diagrama de arquitectura del sistema entre los agentes y componentes externos:



Link de Video: https://youtu.be/H20Ag2bEzkQ

Referencias:

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., ... & Cui, B. (2024). Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*. https://arxiv.org/abs/2402.19473