Instituto Tecnológico de Costa Rica

Campus Tecnológico Central Cartago

Escuela de Ingeniería en Computación

Trabajo Corto 7: Propuesta Técnica para Sistema Multiagente con RAG

Realidad Virtual - Grupo 1

Profesor Kenneth Obando

Daniel Granados Retana, carné 2022104692

Diego Manuel Granados Retana, carné 2022158363

23 de junio del 2025

1. Nombre del sistema

El nombre del sistema multiagente es: TecGPT.

El video con la demostración del sistema se encuentra en el siguiente link: https://youtu.be/TtJsEoA0Vro

2. Descripción general

El objetivo del sistema multiagente es ser un **tutor académico especializado**. Está orientado principalmente para apoyo académico para estudiantes en las etapas de educación secundaria. Esto es principalmente porque el sistema se compone de agentes especializados en varias asignaturas, las cuales son las principales en la escuela y en el colegio. Estos agentes están configurados para responder preguntas de estas materias, explicar los conceptos y procedimientos y utilizar el lenguaje adecuado para un estudiante de secundaria. Hay agentes para los siguientes temas:

- Matemáticas: Apoya en cómo resolver problemas de matemáticas.
- Historia: Explica el contexto histórico y los eventos relevantes.
- Ciencias: Explica conceptos y principios científicos.
- Español: Ayuda con la gramática y redacción de textos.

Adicionalmente, el sistema cuenta con agentes para realizar tareas más generales:

- Recomendador: Recomienda recursos educativos, como libros, artículos académicos, páginas web, etc.
- RAG: Procesa un documento PDF o una página web. Esto implica generar *embeddings* y guardarlos en una base de datos vectorial. Luego, se realiza una

búsqueda de similitud con la entrada del usuario para encontrar los bloques más relevantes. Con base en este contexto, se genera la respuesta del modelo.

3. Agentes involucrados

A continuación, se detallan los agentes implementados y los prompts que usan:

Supervisor: Es un agente que se encarga de la orquestación de todos los agentes. Recibe la entrada del usuario y dirige la consulta al agente más apropiado.

- nombre: supervisor
- Herramientas: Incorpora herramientas para hacer el “handoff” a los otros agentes.
- Prompt: *Usted es un supervisor de agentes para un tutor académico especializado. Supervise las respuestas de los agentes y dirija al usuario al agente adecuado según la pregunta. Si la pregunta es sobre matemáticas, dirija al agente de matemáticas. Si la pregunta es sobre ciencias, dirija al agente de ciencias. Si la pregunta es sobre historia, dirija al agente de historia. Si la pregunta es sobre español, gramática, ortografía, literatura, dirija al agente de español. Si la pregunta es sobre recomendaciones de recursos educativos, dirija al agente de recomendaciones. Si la pregunta es sobre un documento PDF o incluye un URL o enlace a un sitio web, dirija al agente de RAG (Retrieval Augmented Generation). Si la pregunta no es clara o no se puede responder, pida al usuario que aclare su pregunta. Utilice el lenguaje adecuado para un estudiante de secundaria.*
- Interacción con los otros agentes: Redirige el control a los otros agentes especializados.

Asistente de matemáticas: Ayuda con preguntas sobre matemáticas.

- Nombre: math_assistant
- Herramientas: No usa herramientas.
- Prompt: *Usted es un asistente experto en matemáticas. Responda a las preguntas de los usuarios de manera clara y concisa. Explique el paso a paso de cómo resolver los problemas matemáticos. Utilice el lenguaje adecuado para un*

estudiante de secundaria.Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

Asistente de historia: Ayuda con preguntas históricas.

- Nombre: history_assistant
- Herramientas: Tavily Search Tool, para hacer búsquedas, y Wikipedia Tool, para buscar en Wikipedia.
- Prompt: *Usted es un asistente experto en historia. Responda a las preguntas de los usuarios de manera clara y concisa. Explique el contexto histórico y los eventos relevantes. Utilice la búsqueda en línea para proporcionar información actualizada y precisa. Utilice el lenguaje adecuado para un estudiante de secundaria.*
- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

Asistente de ciencias: Ayuda con preguntas sobre ciencias.

- Nombre: science_assistant
- Herramientas: Tavily Search Tool, para hacer búsquedas; Wikipedia Tool, para buscar en Wikipedia y Arxiv Tool para hacer búsquedas de artículos en el repositorio arXiv.org.
- Prompt: *Usted es un asistente experto en ciencias. Responda a las preguntas de los usuarios de manera clara y concisa. Explique el concepto científico y los principios relacionados. Utilice el lenguaje adecuado para un estudiante de secundaria.*
- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

Asistente de español: Ayuda con preguntas sobre la asignatura de Español.

- Nombre: spanish_assistant
- Herramientas: No usa herramientas.
- Prompt: *Usted es un asistente experto en español. Responda a las preguntas de los usuarios de manera clara y concisa. Revise la redacción, la gramática y la*

ortografía de los textos en español, corrigiendo errores y mejorando la claridad. Responda preguntas sobre literatura en español. Utilice el lenguaje adecuado para un estudiante de secundaria.

- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

Agente recomendador: Recomienda recursos educativos, como páginas web, artículos académicos, etc.

- Nombre: recommender_assistant.
- Herramientas: Tavily Search Tool, para hacer búsquedas; Wikipedia Tool, para buscar en Wikipedia y Arxiv Tool para hacer búsquedas de artículos en el repositorio arXiv.org.
- Prompt: *Usted es un asistente experto en recomendaciones de recursos educativos y académicos. Identifique las necesidades de aprendizaje del usuario y proporcione recursos adecuados. Proporcione recomendaciones personalizadas basadas en las preferencias del usuario. Considere libros, artículos, páginas web, videos y otros recursos relevantes para el aprendizaje. Si el usuario solicita recomendaciones de artículos académicos, utilice la herramienta de búsqueda de Arxiv para encontrar artículos relevantes. Utilice el lenguaje adecuado para un estudiante de secundaria.*
- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

3.1. Uso de RAG

Se usa un agente especializado para hacer RAG en páginas web y en documentos PDF:

Agente RAG:

- Nombre: rag_agent.
- Herramientas: Usa una herramienta para recuperar información de la base de datos vectorial y otra herramienta para ingestar la información de una página web.

- Prompt: *Usted es un agente que recupera información de documentos y sitios web usando Retrieval-Augmented Generation (RAG). Si el usuario proporciona un URL, primero use la herramienta 'ingest_website' para agregar el contenido del sitio web a la base de datos de recuperación. Luego, use la herramienta 'retrieve' para responder a la consulta, citando los documentos o sitios web utilizados. Si el usuario menciona un PDF, utilice la herramienta 'retrieve' para buscar información en los documentos PDF almacenados.*
- Interacción con otros agentes: Recibe la pregunta del supervisor. Cuando responde, devuelve el control al supervisor.

La implementación del agente RAG se encuentra en el archivo `rag.py`.

El procedimiento que utiliza este agente para ingestar información es el siguiente:

1. Primero, procesa el documento o página web. Esto se realiza por medio de los objetos `PyPDFLoader` y `WebBaseLoader`, de la biblioteca `langchain_community.document_loaders`.
2. Se particiona el contenido del documento en pedazos de mil caracteres, con 200 caracteres de traslape. Esto facilita que la consulta haga “match” con partes más específicas y relevantes del texto.
3. Se saca un embedding de cada uno de estos pedazos con el modelo de OpenAI `text-embedding-3-large`.
4. Se suben a Chroma, una base de datos vectorial.

Para el caso de una página web, este procedimiento lo realiza la herramienta `ingest_website`. Para el caso de un PDF, desde la interfaz, se escoge un archivo PDF y la carga se hace por medio de la función `process_pdf`.

El procedimiento que utiliza este agente para resolver consultas de usuarios es el siguiente:

1. Recibe la pregunta del usuario.
2. Genera un embedding de la pregunta del usuario.
3. Realiza una búsqueda de similitud vectorial en la base de datos Chroma.
4. Con base en los 3 resultados más similares, el modelo de lenguaje redacta una respuesta a la pregunta.

Los pasos 2 y 3 los realiza la herramienta `retrieve`.

De esta manera, se ejecuta el proceso de generación aumentada por recuperación.

Esto se implementa en el archivo `rag.py`.

4. Tecnologías utilizadas

La principal tecnología usada es LangGraph, la cual se basa en LangChain. LangGraph simplifica considerablemente la creación de agentes por medio de funciones como `create_react_agent`. Esta función facilita crear agentes ya que gestiona el "handoff" entre agentes, las llamadas a las herramientas, la generación de la respuesta y más, de manera automática. Así, la creación de cada agente implica solamente definir el prompt, el nombre y las herramientas que se van a usar. Para flujos más complejos, LangGraph se puede utilizar para crear pipelines de agentes.

LangGraph también facilita la gestión de la memoria del agente. Esto se brinda por medio del objeto *MemorySaver*. Este es el mecanismo para que los agentes tengan memoria a corto plazo. Solamente es necesario pasarle un objeto de esta clase a los agentes como `checkpointer`.

Para darle una mayor funcionalidad al sistema, se aprovecharon las herramientas que ya están integradas en la biblioteca de LangChain. Estas son herramientas desarrolladas por la comunidad que se pueden utilizar muy fácilmente con agentes propios. Más específicamente, se usaron las siguientes:

- TavilySearch: Es una herramienta que implementa un motor de búsqueda diseñado para agentes y se utiliza para realizar búsquedas en Internet.
- arxiv: Es una herramienta para obtener información de artículos en arXiv.
- WikipediaQueryRun: Es una herramienta para hacer búsquedas en artículos de Wikipedia.

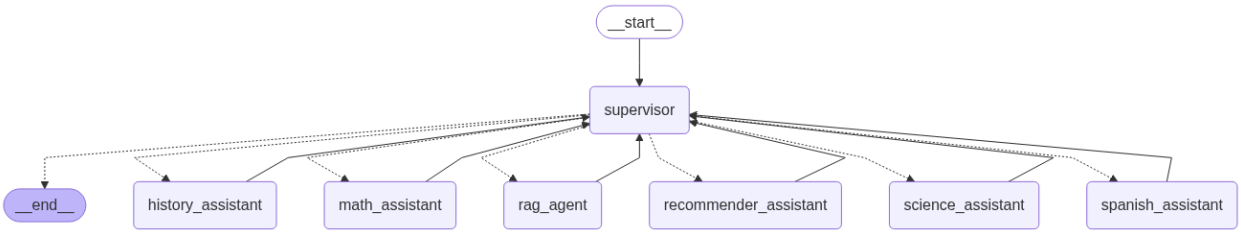
En cuanto a los modelos de inteligencia artificial utilizados, se usaron los que ofrece la compañía OpenAI:

- Para los embeddings, se usó el modelo "text-embedding-3-large".
- Para los agentes, se usó "gpt-4o-mini" para el desarrollo y "gpt-4o" para la demostración.

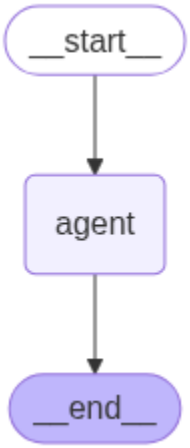
Para la base de datos vectorial, se utilizó Chroma, que es una base de datos de código abierto que busca ser fácil de usar y ser orientada hacia aplicaciones de inteligencia artificial.

5. Diagrama de arquitectura del sistema

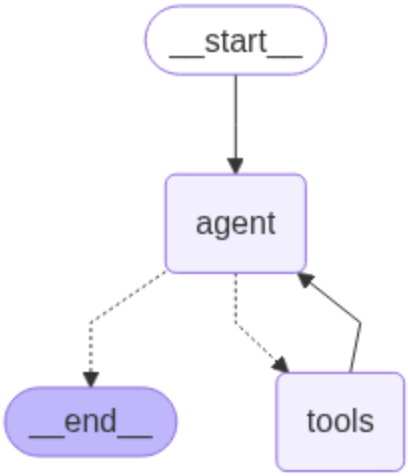
Utilizando la herramienta de LangGraph para generar diagramas de los agentes, el supervisor tiene la siguiente estructura:



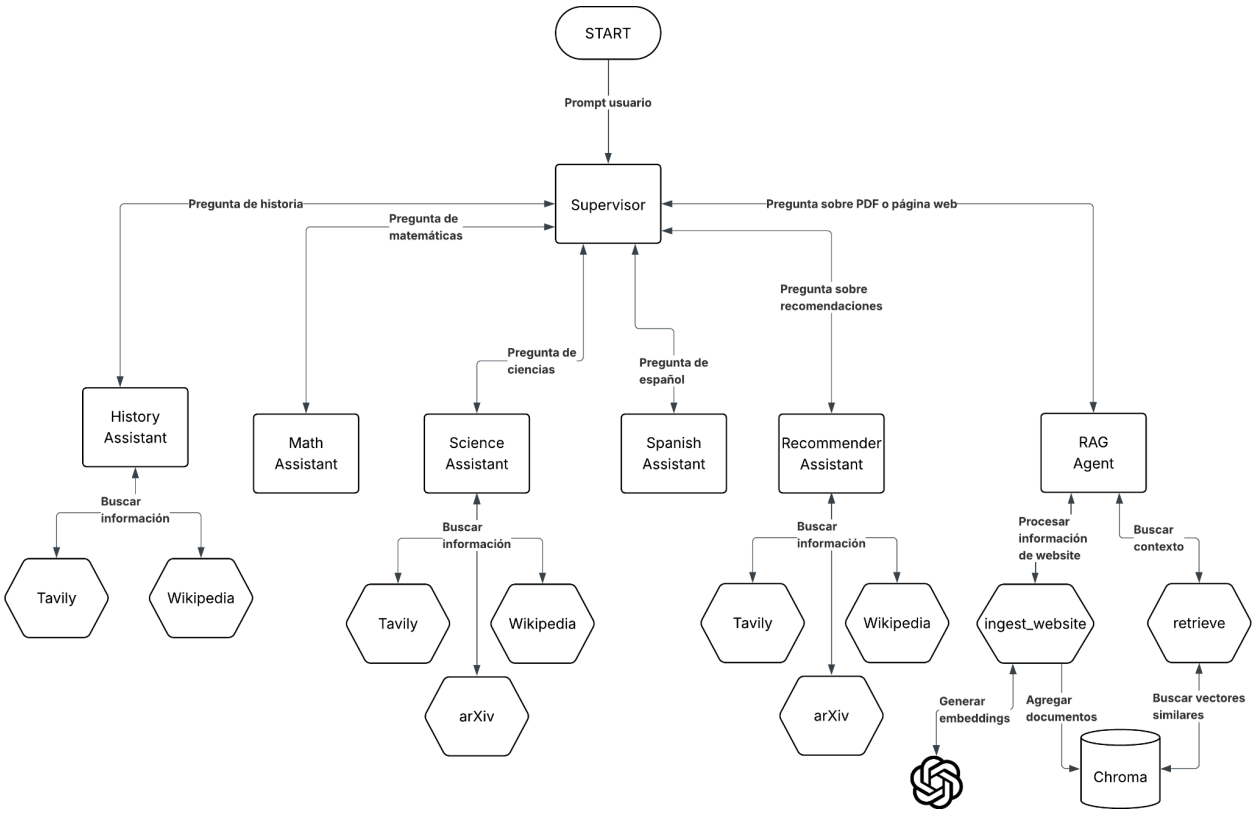
Los asistentes de las asignaturas tienen el siguiente flujo:



El agente de RAG tiene el siguiente flujo:



Esto se puede combinar en un diagrama más completo:



Este diagrama muestra que los agentes tienen una estructura similar. Lo que los caracteriza por separado es su prompt.

6. Referencias

Se adjuntan las referencias consultadas para la realización del sistema multiagente:

ArXiv. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/integrations/tools/arxiv/

Build an Agent. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/tutorials/agents/

Build a Retrieval Augmented Generation (RAG) App: Part 1. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/tutorials/rag/

Build a Retrieval Augmented Generation (RAG) App: Part 2. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/tutorials/qa_chat_history/

Build a semantic search engine. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/tutorials/retrievers/

Chroma. (2025). Chroma. https://www.trychroma.com/

load_tools. (s.f.). LangChain Documentation. https://python.langchain.com/api_reference/community/agent_toolkits/langchain_community.agent_toolkits.load_tools.load_tools.html#load-tools

Memory. (s.f.). LangGraph. Recuperado el 23 de junio, 2025, de https://langchain-ai.github.io/langgraph/agents/memory/

Multi-agent. (s.f.). LangGraph. https://langchain-ai.github.io/langgraph/agents/multi-agent/

Tavily Search. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/integrations/tools/tavily_search/

Wikipedia. (s.f.). LangChain. Recuperado el 23 de junio, 2025, de https://python.langchain.com/docs/integrations/tools/wikipedia/