IT Support Multi-Agent Bot

Nombre del Sistema

IT Support Multi-Agent Bot

Descripción

IT Support Multi-Agent Bot es un sistema inteligente de soporte técnico automatizado, basado en una arquitectura de agentes colaborativos y tecnología RAG (Retrieval-Augmented Generation). Su objetivo es proporcionar asistencia técnica inmediata y contextualizada ante preguntas frecuentes de usuarios, utilizando una base documental compuesta por FAQs, manuales técnicos y documentación oficial.

El sistema descompone el flujo de interacción en distintos agentes especializados, responsables de entender, diagnosticar y resolver las solicitudes del usuario final. Esto permite una estructura modular, extensible y escalable, ideal para escenarios de soporte en entornos corporativos, educativos o domésticos.

Agentes Involucrados

1. UserAgent

- **Rol:** Interfaz principal con el usuario. Recibe preguntas técnicas y coordina el flujo de interacción.
- **Prompt usado:**
  - "Haz tu pregunta técnica: "
- **Interacción:**
  - Recibe la pregunta del usuario y la envía al `DiagnosticAgent`.

2. DiagnosticAgent

- **Rol:** Diagnóstico inicial de la pregunta técnica. Decide si es necesario consultar la base de conocimiento (RAG).
- **Prompt usado:**
  - Recibe la pregunta del usuario y la pasa al sistema RAG mediante `ask_question`.
- **Interacción:**
  - Determina si una pregunta requiere acceso a conocimiento externo.
  - Llama a `ask_question` (RAG) para obtener una respuesta relevante.
  - Pasa la respuesta al `ResolutionAgent`.

3. ResolutionAgent

- **Rol:** Refinamiento y entrega de la respuesta final al usuario.
- **Prompt usado:**
  - Recibe la respuesta del RAG y la retorna tal cual (puede ser extendido para post-procesamiento).
- **Interacción:**
  - Recibe la respuesta del `DiagnosticAgent` y la retorna al `UserAgent`.

Uso del RAG (Retrieval-Augmented Generation)

- **Agente que lo usa:**
  - `DiagnosticAgent`
- **Cómo lo usa:**
  - Llama a `ask_question` que utiliza un vector store (`Chroma`) para buscar en la base documental (`faqs.txt`, PDFs, etc.) y retorna la respuesta más relevante.
- **Componentes involucrados:**
  - `vector_store.py`: carga documentos, crea el vector store, realiza la búsqueda y retorna la respuesta.

Flujo del ask_question

1. Carga los documentos desde /documents.
2. Fragmenta los textos en chunks con TextSplitter.
3. Vectoriza los chunks con OpenAIEmbeddings.
4. Recupera los más similares a la pregunta (k=3).
5. Usa RetrievalQA para combinar documentos con una respuesta generada.

Tecnologías Utilizadas

- **Python 3.12**
- **LangChain** (vector store, text splitter, document loaders)
- **Chroma** (almacenamiento de vectores)
- **OpenAI Embeddings** (vectorización de textos)
- **dotenv** (gestión de variables de entorno)
- **Poetry** (gestión de dependencias)
- **VS Code** (entorno de desarrollo recomendado)

Diagrama de Arquitectura

```
flowchart TD
    U[Usuario] -->|Pregunta| UA[UserAgent]
    UA -->|Pregunta| DA[DiagnosticAgent]
    DA -->|Pregunta| RAG[RAG (ask_question)]
    RAG -->|Respuesta relevante| DA
    DA -->|Respuesta| RA[ResolutionAgent]
    RA -->|Respuesta final| UA
    UA -->|Respuesta| U
    RAG -->|Documentos| Docs[Base de Conocimiento (faqs.txt, PDFs)]
    RAG -->|Embeddings| OpenAI[OpenAI Embeddings]
    RAG -->|Vector Store| Chroma[Chroma DB]
```

Estructura del Proyecto

```
.
├── .env
├── poetry.lock
├── pyproject.toml
├── README.md
├── docs/
│   └── _IAR_TC06__Multiagentes (1).pdf
├── src/
│   └── it_support_bot/
│       ├── __init__.py
│       ├── main.py
│       ├── agents/
│       │   ├── __init__.py
│       │   ├── diagnostic_agent.py
│       │   ├── resolution_agent.py
│       │   └── user_agent.py
│       └── rag/
│           ├── __init__.py
│           ├── vector_store.py
│           ├── chroma_db/
│           └── documents/
│               └── faqs.txt
└── tests/
    └── __init__.py
```

Cómo Ejecutarlo

1. **Clona el repositorio:**

```
git clone https://github.com/tu-usuario/it-support-multiagent-bot.git
cd it-support-multiagent-bot
```

2. **Instala dependencias:**

```
poetry install
```

3. **Configura tus variables de entorno:**

   - Crea un archivo `.env` con tu clave de OpenAI:

```
OPENAI_API_KEY=tu_clave_openai
```

4. **Ejecuta el sistema:**

```
poetry run python src/it_support_bot/main.py
```

Repositorio

https://github.com/tu-usuario/it-support-multiagent-bot