Implementing Agentic RAG

Assignment

Assignment: Develop an AI-Powered Competitive Analysis Agent with Agentic RAG

Scenario

You work as a data scientist at a consulting company that assists businesses in acquiring a competitive advantage by examining their rivals' tactics. The company has chosen to utilize AI for automating competitive analysis, allowing quicker and more insightful reports for its clients. Your objective is to create an AI-Driven Competitive Analysis Agent that utilizes Agentic RAG to examine competitor information, respond to inquiries, and deliver practical insights. The agent will depend on Cohere's models for embeddings and generation, along with LlamaIndex for effective data indexing and retrieval. The agent must integrate the concepts of Agentic RAG, including reasoning and taking action, to manage intricate inquiries flexibly.

Problem Statement

Companies frequently find it challenging to stay ahead of their rivals because manual competitive analysis is time-intensive. The objective is to develop an AI agent that is capable of:

- Efficiently index and retrieve data on competitors (e.g., marketing strategies, financial reports, product descriptions).
- Respond to natural language inquiries regarding competitors (e.g., "What are the main advantages of Competitor X's marketing approach?").
- Deliver practical insights by analyzing the data and decomposing intricate queries into simpler, achievable sub-goals.
- Modify its actions according to the complexity of the query, employing the ReAct framework to combine reasoning with activities.

Tasks to Be Implemented

- **Set Up the Environment**
    - Create a project directory with necessary files (e.g., main script, requirements, .env for API keys).
    - Set up a virtual environment and install dependencies (e.g., Cohere, LlamaIndex, pandas, numpy, python-dotenv).

- Configure environment variables for the API keys.

- **Data Preparation**
  - Use a dataset of competitor data (e.g., a CSV file with columns like "Competitor Name", "Product Description", "Marketing Strategy", "Financial Summary").
  - Load the data into a pandas DataFrame and preprocess it (e.g., clean text, handle missing values).
  - You can create dummy data or try real-time analysis like in the demo for Market Analysis with Cohere.

- **Implement Agentic RAG with LlamaIndex/Cohere**
  - Use LlamaIndex to index the competitor data for efficient retrieval (e.g., create a VectorStoreIndex with Cohere embeddings).
  - Implement a retrieval system to fetch relevant documents based on user queries.
  - You can use Cohere's embed-english-v3.0 model for generating embeddings and command-a-03-2025 for response generation.

- **Build the Competitive Analysis Agent**
  - Create a class CompetitiveAnalysisAgent that follows the ReAct framework.
  - Implement a reason_and_act method that:
    - Analyzes the query to determine intent (e.g., does it ask for strengths, weaknesses, or comparisons?).
    - Breaks down the query into sub-goals (e.g., retrieve data, analyze strengths, generate insights).
    - Executes sub-goals using appropriate tools (e.g., LlamaIndex retrieval, Cohere generation).
  - Log reasoning steps for transparency.

- **Interactive Interface**

- Create an interactive CLI or UI where users can input queries (e.g., "Compare the marketing strategies of Competitor X and Competitor Y").
- Include features to view query history and exit the program.

Required Feature Suggestions:

- Agentic RAG with ReAct: The agent should demonstrate reasoning (e.g., analyzing query intent) and acting (e.g., retrieving data, generating responses) using the ReAct framework.
- LlamaIndex Integration: Use LlamaIndex for efficient indexing and retrieval of competitor data.
- Cohere Models: Leverage Cohere's embed-english-v3.0 for embeddings and command-a-03-2025 for generation.
- Adaptive Behavior: The agent should adapt its actions based on query complexity (e.g., fetching more data for recent queries, summarizing for overview queries).
- Query History: Allow users to view recent queries and responses.
- Reasoning Transparency: Log reasoning steps for each query to show the agent's decision-making process.

Note:

Data Preparation: While importing the CSV, make sure text fields are sanitized (e.g., eliminate special symbols, standardize case) to enhance embedding quality with Cohere. LlamaIndex Configuration: Think about employing LlamaIndex's VectorStoreIndex alongside Cohere embeddings. You can start it with a custom embedding model by supplying a Cohere client to LlamaIndex. ReAct Implementation: In the reason_and_act function, utilize keywords in the query (e.g., "analyze", "advantages") to identify sub-goals. For instance, if the query includes "compare", gather data for both rivals and examine the distinctions. Cohere Models: Utilize embed-english-v3.0 with the correct input_type (search_document for indexing, search_query for queries). To achieve optimal results

from command-a-03-2025, make sure that prompts are straightforward and succinct during generation.

Error Management: Incorporate try-except structures around API requests to address possible errors (e.g., API rate restrictions, incorrect input texts).

Query History: Save queries and their responses in a list inside the agent class, and create a "history" command to show the most recent 5 queries.