RAG Assignment 1: Basic RAG Implementation and Comparison

Objective

Implement a basic Retrieval-Augmented Generation (RAG) system and compare its performance with local LLMs using your existing interface.

Tasks

Set up the LLMs

- Ensure Llama3 is set up and running via Ollama (mandatory for all students).
- Use the second LLM you set up in your previous project.

Implement the basic RAG system

- Use the provided code (`https://colab.research.google.com/drive/1mAJKlSDpucygH0V-ohUqpoQSF5F-xXwd?usp=sharing`) as a starting point.
- Modify the code to work with Llama3 and your second LLM instead of Claude or GPT.
- Ensure the RAG system can run entirely on your local machine.
- Document any changes you needed to make to adapt the RAG system for use with local LLMs.

Integrate RAG with your existing interface

- Adapt the user interface you developed in your previous project to incorporate the RAG system.
- Ensure users can switch between using the RAG system and standalone LLMs.
- If you didn't create a UI in the previous project, develop a basic command-line interface for this purpose.
- To integrate the RAG system with your existing interface:
  - Analyze your current interface code and identify areas that handle query input and response output.
  - Modify your input handling to allow users to choose between RAG and non-RAG modes.
  - Update your query processing logic to route queries through the RAG system when enabled.
  - Adjust your response display to accommodate the additional context or information provided by the RAG system.
  - Add any RAG-specific features or settings you want to expose to the user.
  - Ensure proper error handling and performance optimization for RAG queries.
  - Thoroughly test the integrated system to ensure smooth operation in both RAG and non-RAG modes.

Comparative analysis

- Use the test queries from your previous project, and add additional ones if necessary to reach at least 10 diverse queries.
- Include a mix of domain-specific and general knowledge queries to test the RAG system's versatility.
- Run these queries through:
  1. The RAG system
  2. Standalone Llama3
  3. Your second LLM
- Document and compare responses, focusing on relevance, accuracy, and context-awareness.
- Analyze differences in response times and resource usage.

Reflection and documentation

- Write a 500-word reflection on how the RAG system changes or enhances the capabilities of your chatbot compared to using standalone local LLMs.
- Document your implementation process, including any challenges faced and solutions devised.

Deliverables

- GitHub repository containing your RAG system implementation, comparison code, and updated interface.
- A comprehensive report including:
  - Detailed comparison results and analysis
  - Interaction logs and screenshots using your interface
  - Your reflection on the RAG system vs. local LLMs in the context of your chatbot
- Updated README file with clear instructions for setting up and running your enhanced system.
- A brief user guide explaining how to use the new RAG features in your chatbot interface.

Video Demonstration

Create a 5-10 minute YouTube video showcasing your enhanced RAG-powered chatbot.

Video Demonstration Guidelines

Your video should include:

- **Brief overview of your RAG system:**
  - Explain how you've integrated RAG with your local LLMs (Llama3 and your chosen second LLM).
  - Highlight any new setup processes you've implemented.
- **Demonstration of key features:**
  - Show how users can switch between RAG and non-RAG modes.
  - Demonstrate the chatbot answering queries using the enhanced knowledge base.

  - Showcase any conversation improvements or advanced features you've implemented.

- **Performance comparison:**
  - Briefly compare responses from your RAG system vs. standalone LLMs for a couple of queries.
  - Highlight improvements in relevance, accuracy, or context-awareness.

- **User interface walkthrough:**
  - Guide viewers through your updated UI, pointing out new features.
  - Demonstrate how users can provide feedback or rate responses (if implemented).

- **Challenges and solutions:**
  - Discuss 1-2 significant challenges you faced during development.
  - Explain how you overcame these challenges or what you learned from them.

- **Future improvements:**
  - Briefly mention any ideas you have for further enhancing your RAG-powered chatbot.

Tips for your video

- Keep explanations concise but clear.
- Use screen recordings to demonstrate your chatbot in action.
- Ensure good audio quality for your narration.
- You may use simple video editing to add text overlays or highlights if desired.

Grading Rubric for RAG Assignment 1

- **RAG System Implementation and Integration (30%)**
  - Successful setup and modification of RAG system with local LLMs.
  - System functionality on local machine.
  - Integration with existing interface, including switching mechanism.
  - Code quality and organization.

- **Comparative Analysis and Reflection (25%)**
  - Quality and diversity of test queries (at least 10).
  - Thoroughness of response comparison and performance analysis.
  - Insight in 500-word reflection on RAG system capabilities.
  - Documentation of implementation process, challenges, and solutions.

- **Video Demonstration (15%)**
  - Clear explanation of RAG system integration and features.
  - Effective demonstration of chatbot functionality.
  - Discussion of challenges, solutions, and future improvements.
  - Overall presentation quality and clarity.

- **Deliverables Quality (20%)**

  - Completeness of GitHub repository.
  - Clarity of report, README, and user guide.
  - Overall organization and presentation of materials.
- **User Experience and Functionality (10%)**
  - Usability of the RAG-enhanced chatbot.
  - Proper error handling and performance.
  - Additional RAG-specific features or innovations.

Bonus Points (up to 5 extra points)

- Implementation of additional RAG-specific features not explicitly required (2 points).
- Exceptional creativity or innovation in RAG integration or analysis (3 points).

Deductions

- Late submission: -5% per day.
- Missing major components: up to -10% per component.
- Plagiarism or code copying without attribution: potential for zero grade and academic disciplinary action.

Total: 100%

Due Date: Oct. 27