Movie Question Answering System

Using Retrieval-Augmented Generation and Large Language Models

1. Overview

1.1 Project Goal

Build an intelligent question-answering system that can answer natural language questions about a dataset of 10,000 IMDB movies. Your system must handle two fundamentally different types of queries:

1. **Semantic/Conceptual Queries**: Questions requiring understanding of movie content, themes, and context
   - Examples: "What are some alien-related movies?", "Which films explore time travel themes?", "What movies feature strong female protagonists?"
2. **Factual/Statistical Queries**: Questions requiring numerical computations, aggregations, or filtering
   - Examples: "What's the average rating of James Bond movies?", "Which director has the highest-grossing film?", "How many sci-fi movies were released after 2010?"

1.2 Dataset

You will work with an IMDB movie dataset containing ~10,000 movies.

Dataset Link:
https://drive.google.com/file/d/1MtthDLw4VXobg9EoUf0Nk5uZIn01e0CR/view?usp=sharing

---

2. Technical Requirements

2.1 Suggested Technologies

- **LLM**: Qwen, Llama, or similar models.
- **Embeddings**: sentence-transformers

- **Platform**: Google Colab Pro (required for GPU access)
- **Key Libraries**: transformers, torch, pandas, numpy, llama-index, faiss

2.2 Suggested System Architecture

Your system can include the following components:

1. **Data Preprocessing Module**
    - Load and clean the movie dataset
    - Create rich text representations for embedding
    - Handle missing values appropriately
2. **Vector Index Construction**
    - Generate embeddings for movie descriptions
    - Build and persist a vector index using LlamaIndex
    - Implement efficient similarity search
3. **Query Classification System**
    - Automatically classify incoming queries as semantic or factual
    - Route queries to appropriate processing pipeline
4. **Semantic Query Pipeline (RAG)**
    - Retrieve relevant movie documents using vector similarity
    - Generate contextual answers using the LLM
    - Provide source attribution (which movies informed the answer)
5. **Factual Query Pipeline (Code Generation)**
    - Generate pandas/SQL code to answer statistical questions
    - Execute code safely with appropriate error handling
    - Format numerical results into natural language using the LLM
6. **Unified Interface**
    - Single function to answer any question
    - Clear logging of query type and processing steps
    - User-friendly output formatting

---

3. Deliverables

3.1 Code Submission

Submit a well-documented Jupyter notebook containing:

Part 1: Setup and Data Loading (10 points)

- Installation of all required packages
- Model loading with proper quantization
- Dataset loading and preprocessing
- Initial data exploration and statistics

Part 2: Vector Index Construction (10 points)

- Document creation from dataframe
- Embedding generation
- Vector index building
- Index persistence and loading

Part 3: Semantic Query Implementation (25 points)

- RAG pipeline implementation
- Query engine configuration
- Retrieval quality testing
- At least 5 example semantic queries with results

Part 4: Factual Query Implementation (35 points)

- Code generation prompts
- Safe code execution framework
- Result formatting
- At least 5 example factual queries with results

Part 5: Query Classification and Integration (20 points)

- Query classification logic
- Unified question-answering interface
- Comprehensive testing with diverse queries
- Error handling and edge cases