Generative AI and Large Language Models (LLMs)

Final Project

Assigned: 11/23/2025
Submission: 12/07/2025, 11:59 pm
Max points: 100

Project Brief: Dual-Level Policy Navigator (RAG System Development)
Focus: Retrieval-Augmented Generation (RAG), Grounding, and Evaluation

Project Summary: Equitable Policy Information Access

The goal is to design, implement, and evaluate a Retrieval-Augmented Generation (RAG) system called the "Policy Navigator." This application will act as a reliable, grounded chatbot for a public service organization, providing complex, accurate answers about public policy, health guidelines, or academic policies.

This project applies technical skills (RAG, embeddings, prompt engineering) to the social problem of information access and clarity.

Part 1: Data Preparation and Indexing (Required for ALL Students)

Objective: Create a searchable vector database from publicly available policy documents.

1. **Data Collection (Source Documents):** Acquire and curate a set of at least 5-10 distinct public-domain documents (PDFs, HTML, or plain text) from one of the suggested domains below. The data must be complex enough to require intelligent chunking.

Suggested Specific Datasets (Choose ONE Domain):

Domain	Data Source	Example Content	Link
Public Health/Government Policy	Centers for Disease Control and Prevention (CDC)	Official CDC documents, fact sheets, and FAQs on specific guidelines or immunization schedules.	https://www.cdc.gov/policy/polaris/resources/index.html (Look for downloadable PDFs/HTML on specific topics.)
Academic Policy/Administrative	MIT/Harvard/Stanford Public	Policy reports, white papers, or detailed	https://www.kaggle.com/datasets/arhamfaisal/policy-reports-pdfs (Kaggle dataset with various policy PDFs)

	Policy Papers	administrative documents.	
Federal Benefits (Complexity Test)	Social Security Administration (SSA) FAQs & Handbooks	Detailed FAQs, handbooks, and eligibility guides for Social Security benefits.	https://www.ssa.gov/forms/ (Focus on selecting several related PDF handbooks or large FAQ documents.)

2. **Document Processing:**
    - Implement and test **two distinct chunking strategies** (e.g., fixed-size with overlap vs. sentence-based/semantic chunking) to optimize context coherence.
    - Select a robust open-source embedding model (e.g., all-MiniLM-L6-v2) and vectorize the chunks.
3. **Vector Store Indexing:** Index the vectorized chunks into a simple in-memory vector store (FAISS or chroma-db locally).

Part 2: RAG Pipeline Implementation and Grounding (Required for ALL Students)

Objective: Build the core RAG system and ensure accurate, non-hallucinated responses.

1. **Basic RAG Implementation:** Structure the standard RAG pipeline using Python and a framework like LangChain or LlamaIndex:

   User Query → Embedding → Vextor Search (Top-k) → Context Injection → LLM Generation
2. **Prompt Engineering for Grounding:** Develop a detailed system prompt that enforces strict grounding. The LLM must be instructed to:
    - Strictly adhere to the provided context when generating the answer.
    - If the context does not contain the answer, respond with a predefined, non-hallucinated message (e.g., "The required information is not available in my current resource database.").

Part 3: Evaluation (Differentiated Tasks)

Objective: Validate the system's performance against baselines using quantitative metrics.

Undergraduate Student Tasks

1. **Test Dataset Creation:** Generate 10 complex, scenario-based test questions that **require** detailed, policy-grounded answers.

2. **Comparative Evaluation (Two Systems):** Run the 10 test questions through **two** distinct systems:
    - **System 1:** Vanilla LLM (Zero-shot prompt, no RAG/context).
    - **System 2:** Basic RAG (Simple vector search).
3. **Metrics Calculation:** Evaluate and report on the following metrics for both systems:
    - **Faithfulness/Factuality (Grounding):** What percentage of the generated answer's claims are verifiably supported by the source documents?
    - **Answer Relevancy:** How well did the answer fully address all parts of the user's complex query?

Graduate Student Additional Task (MANDATORY)

Graduate students must complete all Undergraduate tasks **PLUS** the following:

1. **Advanced Retrieval Implementation (The Additional Task):** Implement **one** advanced retrieval method to enhance accuracy beyond simple cosine similarity search. Choose and justify your selection:
    - **Option A: Re-ranking:** Implement a cross-encoder model (e.g., co-condenser-marco-ms) to re-rank the initial top-K search results, selecting only the most relevant documents for the final LLM context.
    - **Option B: Query Transformation/Expansion:** Use a separate LLM call to transform the original complex user query into 3-5 sub-queries (or a HyDE-style hypothetical document) to improve initial vector search coverage before context injection.
2. **Comparative Evaluation (Three Systems):** Run the 10 test questions through **three** distinct systems:
    - System 1: Vanilla LLM
    - System 2: Basic RAG
    - System 3: Advanced RAG (Your implemented improvement)

Deliverables

Deliverable	Undergraduate Students	Graduate Students
Code Repository	Basic RAG Pipeline Code	Basic RAG + Advanced Retrieval Code
Final Report (4-5 pages, PDF)	2-System Comparison Table (Vanilla vs. Basic RAG), discussion of chunking strategies.	3-System Comparison Table (Vanilla vs. Basic vs. Advanced RAG), detailed technical description and justification of the advanced retrieval technique.
Critical Discussion	Discussion of limitations and social impact.	Discussion of limitations, social impact, and the comparative difficulty/benefit of Advanced Retrieval.

DETAILED RUBRICS:

Deliverable 1: Code Repository and Implementation (30 Points)

Criteria	Max Points	Undergraduate Students (Basic RAG)	Graduate Students (Advanced RAG)
A. RAG Pipeline Functionality	10	The Basic RAG pipeline executes successfully: data loading, chunking, embedding, vector search, context injection, and LLM generation.	The Basic RAG pipeline executes successfully AND the chosen Advanced Retrieval method (Re-ranking or Query Expansion) is fully implemented and integrated.
B. Data Preparation & Chunking	10	Implements two distinct chunking strategies. Code is clean and modular. Data source is clearly documented.	Implements two distinct chunking strategies. Code is clean and modular. Critical analysis within code comments explaining *why* the chosen chunking strategy was selected for the final pipeline.
C. Grounding & Error Handling	5	System prompt successfully enforces grounding (avoids simple hallucinations). Non-	System prompt successfully enforces grounding. Non-found answers trigger the defined error message. Implementation includes robust error handling (e.g., try-

		found answers trigger the defined error message.	except blocks for API calls, retrieval failures).
D. Advanced Retrieval (Grad Only)	5	Not applicable.	The implementation of the Advanced Retrieval technique (Option A or B) is technically sound, runnable, and utilizes appropriate external libraries/models (e.g., a cross-encoder model for Re-ranking).
Subtotal	**30**	**30 Points**	**30 Points**

Deliverable 2: Final Report (40 Points)

The report must be 4-5 pages (excluding appendices and references) and clearly structured. Font: Times New Roman, size 12.

Criteria	**Max Points**	**Undergraduate Students (2-System Comparison)**	**Graduate Students (3-System Comparison)**
A. Project Overview & Data	5	Clear introduction, documentation of the chosen data source, and a brief justification of chunking choices.	Clear introduction, documentation of the chosen data source, and detailed quantitative discussion of the two chunking strategies and their impact on context quality.
B. Evaluation Setup	10	The 10 complex test questions are clearly defined and relevant to the source documents. The Vanilla LLM and Basic RAG setup are explained.	The 10 complex test questions are clearly defined. All three system setups (Vanilla, Basic RAG, and Advanced RAG) are clearly explained and documented.
C. Comparative Results Table	15	Presents a neat and accurate **2-System Comparison Table** showing Faithfulness and Relevancy metrics for the Vanilla LLM and the Basic RAG system.	Presents a neat and accurate 3-System Comparison Table showing Faithfulness and Relevancy metrics for all three systems (Vanilla, Basic RAG, Advanced RAG). Includes visualization (graph/chart) of the comparative performance.

D. Advanced Technical Justification	10	Not applicable.	**Mandatory:** Detailed technical description and justification (citing relevant literature) for the chosen Advanced Retrieval technique (Option A or B). Analysis of *why* this technique should theoretically outperform basic similarity search.
Subtotal	**40**	**40 Points**	**40 Points**

Deliverable 3: Critical Discussion and Analysis (30 Points)

Criteria	**Max Points**	**Undergraduate Students**	**Graduate Students**
A. Limitations & Failure Modes	10	Identification of at least three key limitations of the Basic RAG system (e.g., context window size, data freshness, embedding quality).	Identification of at least three key limitations of the RAG system, **including specific failure modes observed during the advanced retrieval testing** (e.g., Query Expansion introducing irrelevant noise, Re-ranker bias).
B. Social Impact & Ethical Reflection	10	Thoughtful discussion on how the Policy Navigator can improve information equity and accessibility in urban settings (the "why").	**Deep reflection** on the ethical implications of using LLMs for policy advice, including risks of misinformation, bias amplification, and the responsibility of the system's human custodians.
C. Comparative Benefit Analysis (Grad Only)	10	Not applicable. You will be given extra 10 points here.	**Mandatory:** Critical analysis of the trade-off between the implementation difficulty/computational cost of the Advanced Retrieval method versus its empirical benefit (the gain in performance from System 2 to System 3). Should address whether the benefit justified the complexity.
Subtotal	**30**	**30 Points**	**30 Points**