RAG-based QA Assistant: Course Project Specification

Project Overview

Project Description

Build a domain-specific conversational QA assistant that combines information retrieval with large language model generation. The system must answer questions based on a curated document corpus, provide citations for its answers, handle multi-turn dialogues, and recognize when it cannot answer a question.

Learning Objectives

By completing this project, you will:

- Understand and implement Retrieval-Augmented Generation (RAG) architectures
- Gain practical experience with embedding models, vector databases, and LLMs
- Design and conduct rigorous evaluation of NLP systems
- Develop multi-turn conversational systems with context management
- Apply software engineering best practices to NLP projects
- Critically analyze system failures and propose improvements

Key Requirements

- **Team size**: 3 students per group
- **Duration**: 10 weeks (5 major milestones)
- **Domain**: Student choice (any well-bounded domain)
- **Corpus size**: Minimum 100 documents, recommended 200-500

Project Milestones

Milestone 1: Project Setup & Domain Selection

Objectives

Establish project foundation, select appropriate domain, and create initial evaluation framework.

Tasks

- Form teams and set up collaboration tools (GitHub, communication channel)
- Select a bounded domain with clear scope
- Identify and collect corpus (ensure legal access/rights)
- Create initial test set: 20-30 question-answer pairs with ground truth
- Research relevant RAG papers and existing systems
- Define project scope and success criteria

Deliverables

Project Proposal Document

- 1-2 pages including:
    - Domain description and motivation (why this domain?)
    - Corpus source and composition (number of docs, topics covered, quality assessment)
    - 5 example questions the system should answer
    - Team member roles and responsibilities
    - Project timeline with individual milestones
    - Risk assessment (what could go wrong?)
    - Initial literature review (5+ relevant papers identified)

Success Criteria

- Domain is well-bounded and evaluable
- Corpus is accessible and sufficient size (100+ docs)
- Test questions reflect realistic use cases
- Clear division of responsibilities
- Proposal approved by instructor

Tips

- Choose domains where documents are: structured, accessible, and legal to use
- Good examples: company documentation, academic papers in subfield, technical manuals, course materials, open government docs

---

Milestone 2: Retrieval System Implementation

Objectives

Build and evaluate a working semantic retrieval system that forms the foundation of your RAG pipeline.

Tasks

- Implement document preprocessing and chunking strategy
- Generate embeddings for entire corpus
- Set up vector database (FAISS, ChromaDB, or Pinecone)
- Implement semantic search retrieval with top-k ranking
- Evaluate retrieval quality using your test set
- Experiment with different chunking strategies and embedding models
- Document design decisions and trade-offs

Deliverables

Retrieval System Package

1. **Working code** with comprehensive documentation
2. **Jupyter notebook** demonstrating:
    - Document processing pipeline with examples
    - Embedding generation process
    - Example queries with top-k retrieved chunks (show 5+ examples)
    - Retrieval quality analysis with visualizations
3. **Technical report** (2-3 pages): **This will be part your final report**
    - Chunking strategy justification (size, overlap, metadata)
    - Embedding model selection and comparison
    - Retrieval results: Precision@k, Recall@k, MRR
    - Error analysis: cases where retrieval fails
    - Design trade-offs discussion

Success Criteria

- Retrieval system runs end-to-end without errors
- Clear documentation of preprocessing decisions
- Systematic comparison of at least 2 approaches (e.g., different chunk sizes or embedding models)
- Insightful analysis of retrieval failures

Technical Specifications

- **Chunking**: 200-500 tokens per chunk with 50-100 token overlap
- **Embedding models**: sentence-transformers

- **Retrieval**: Return top 3-5 chunks per query
- **Metadata**: Preserve document ID, section, title for citation

---

Milestone 3: Generation & End-to-End RAG Pipeline

Objectives

Integrate LLM for answer generation, implement citation mechanism, and build complete RAG pipeline.

Tasks

- Select and integrate LLM (open-source)
- Design prompt template incorporating retrieved context
- Implement citation mechanism to attribute claims to source documents
- Build end-to-end pipeline: query → retrieval → generation → formatted answer
- Create user interface (CLI or basic web interface)
- Test system on diverse question types
- Compare RAG approach to baseline (LLM without retrieval)

Deliverables

Complete QA System Package

1. **End-to-end working system** with user interface
2. **Code repository** with clear documentation and README
3. **Demo notebook** showing:
    - 10+ question-answer examples covering different question types
    - Comparison with baseline (LLM-only answers)
    - Citation accuracy examples
    - Failure cases with analysis
4. **Technical report** (3-4 pages): **This will be part your final report**
    - System architecture diagram
    - LLM choice justification (cost, latency, quality trade-offs)
    - Prompt engineering approach with iterations
    - Citation strategy and implementation
    - Baseline comparison results
    - Quality analysis with examples

Success Criteria

- System produces coherent, relevant answers
- Citations are accurate and consistently formatted
- RAG system outperforms baseline LLM on domain questions

- Clear improvement over Milestone 2 retrieval-only approach
- User interface is functional and intuitive
- Prompt template is well-designed with clear instructions

Technical Specifications

- **LLM options**:
  - Open-source: Llama-2-7B, Mistral-7B, Flan-T5 or other open source LLMs
- **Prompt template must include**:
  - System instructions (role, behavior guidelines)
  - Retrieved context with source identifiers
  - Question
  - Instructions for citation format and handling uncertainty
- **Citation format**: [Source: document_id] or similar clear attribution
- **Response format**: Answer + citations + confidence (optional)

Milestone 4: Multi-turn Dialogue & Advanced Feature

Objectives

Enable conversational interactions and implement one advanced enhancement to improve system capability.

Tasks

- Implement conversation history management
- Add follow-up question handling (coreference resolution, context maintenance)
- Test multi-turn dialogue scenarios (create 5-10 multi-turn test conversations)
- Conduct ablation study to measure impact of enhancement
- Optimize system latency and performance

Deliverables

Enhanced System Package

1. **Updated system** with multi-turn capability and chosen enhancement
2. **Ablation study report** comparing:
   - System with vs. without chosen enhancement
   - Different conversation memory strategies (if applicable)
   - Quantitative improvement metrics
3. **Demo video** (3-5 minutes):
   - Multi-turn conversation demonstrations
   - Enhancement feature showcase
   - Edge case handling examples

4. **Technical report** (4-5 pages): (**This will be part your final report**)
    - Multi-turn dialogue architecture
    - Conversation history management approach
    - Enhancement description and motivation
    - Implementation details
    - Ablation results with statistical analysis
    - Performance profiling (latency, throughput)
    - Limitations and trade-offs

Success Criteria

- System handles multi-turn conversations coherently
- Context is preserved across turns (test with follow-up questions)
- Chosen enhancement shows measurable improvement
- Ablation study demonstrates enhancement value with statistics
- System meets latency requirements (p95 < 5 seconds)
- Demo clearly showcases new capabilities

Technical Specifications

Conversation Memory

- Keep last 3-5 turns OR sliding window of last 1000 tokens
- Track document sources across turns for citation consistency
- Handle context overflow gracefully

Milestone 5: Comprehensive Evaluation & Final Delivery

Objectives

Conduct rigorous evaluation, perform deep error analysis, and deliver complete project packages.

Tasks

Evaluation & Analysis

- Expand test set to 50-100 questions (mix single-turn and multi-turn)
- Run comprehensive evaluation suite
- Conduct detailed error analysis with categorization
- Profile system performance and identify bottlenecks
- Compare with baselines and state-of-the-art (if applicable)
- Prepare all visualizations and results tables

Final Report & Presentation

- Write comprehensive final report
- Prepare presentation and demo
- Finalize code repository with full documentation
- Create reproducibility package
- Prepare supplementary materials

Deliverables

1. Comprehensive Evaluation Report

- Complete evaluation results measuring:
    - **Accuracy**: Answer correctness (manual or automated evaluation)
    - **Retrieval quality**: Precision@k, Recall@k, MRR, NDCG
    - **Citation accuracy**: % of claims with correct citations
    - **Dialogue coherence**: Context preservation across turns
    - **Latency**: p50, p95, p99 response times (optional)
- Error analysis with categorization (20-30 failures minimum):
    - Retrieval failures (wrong chunks retrieved)
    - Generation errors (hallucination, off-topic)
    - Citation errors (missing or incorrect attributions)
    - Context loss (multi-turn failures)
    - Tool errors (if applicable)
- Statistical significance testing (paired t-test, Wilcoxon)
- Baseline comparisons

2. Final Report (8-16 pages, conference format)

- Abstract
- Introduction (motivation, problem statement, contributions)
- Related Work (RAG systems, conversational QA, attribution, 10+ papers)
- System Design & Architecture (with diagrams)
- Implementation Details (corpus, models, hyperparameters)
- Experimental Setup (evaluation data, metrics, baselines)
- Results & Analysis (tables, figures, statistical tests)
- Error Analysis & Discussion (categorized failures with examples)
- Limitations & Future Work (honest reflection)
- Conclusion
- References

3. Code Repository (complete package)

- Clean, well-documented code with type hints
- Comprehensive README with:
    - Project overview

    - Setup instructions (step-by-step)
    - Usage examples
    - System requirements
    - Known limitations
- requirements.txt or environment.yml
- Organized directory structure
- Example scripts and notebooks
- Evaluation notebooks with reproducible results
- Tests (unit tests for key components recommended)

4. Presentation (10 min + 5 min Q&A)

- 10-12 slides covering:
    - Problem motivation (1-2 slides)
    - System architecture (2 slides)
    - Key technical decisions (2-3 slides)
    - Results & evaluation (2-3 slides)
    - Error analysis insights (1 slide)
    - Conclusions & lessons learned (1 slide)
- Clear, professional delivery