Assignment: Industry-Specific RAG System with Tool Use and MCP

Overview

In this project, you will design, implement, and deploy a Retrieval-Augmented Generation (RAG) application tailored to a specific industry domain. This assignment will challenge you to integrate modern LLM capabilities including tool use/function calling and the Model Context Protocol (MCP) to create a sophisticated AI assistant with domain-specific expertise.

Learning Objectives

- Design and implement a production-quality RAG system
- Integrate vector databases for efficient knowledge retrieval or use an existing API
- Implement tool use/function calling for enhanced capabilities
- Utilize the Model Context Protocol for structured interactions
- Develop a responsive UI with streaming capabilities
- Understand domain-specific considerations for AI applications

Industry Selection

Choose ONE of the following industry domains for your RAG application or pick an industry of your liking:

1. Healthcare (clinical decision support, medical research)
2. Legal (case law research, contract analysis)
3. Financial services (investment research, regulatory compliance)
4. Education (personalized tutoring, research assistance)
5. Environmental science (climate data analysis, sustainability research)

Project Components

Part 1: API Integration for Domain Knowledge (15%)

Objective: Leverage external APIs to provide real-time domain knowledge to your RAG

application.

Requirements:

- Integrate at least 1 industry-relevant APIs to augment your RAG system
- Design effective prompting strategies that incorporate API-retrieved information
- Implement caching mechanisms to optimize API usage
- Build adapters that convert API responses into formats suitable for your foundational model

Suggested APIs by Industry:

- **Healthcare**: PubMed API, RxNorm API, FHIR API for healthcare data
- **Legal**: LexisNexis API, Caselaw Access Project API, Court Listener API
- **Financial**: Alpha Vantage API, Financial Modeling Prep API, SEC EDGAR API
- **Education**: Khan Academy API, Open Library API, Coursera API
- **Environmental**: NOAA Climate Data API, EPA API, NASA Earth Data API

Resources:

- LangChain GitHub Repository
- LlamaIndex GitHub Repository
- Requests Library Documentation
- FastAPI Documentation

Deliverables:

- API integration code with proper error handling
- Documentation of API selection rationale
- Prompt engineering strategies incorporating API data
- Caching and optimization mechanisms

Part 2: DSPy Integration and Prompt Programming (20%)

Objective: Leverage DSPy to create optimized and programmable prompting strategies.

Requirements:

- Implement the DSPy framework for systematic prompt programming
- Create domain-specific modules using DSPy's programming model
- Design and optimize prompts using DSPy's teleprompters
- Implement signature chaining for complex reasoning tasks
- Integrate DSPy with your API-based knowledge retrieval

Resources:

- DSPy GitHub Repository
- LangChain Documentation
- Stanford AI Lab

Deliverables:

- Implementation of DSPy modules relevant to your domain
- Documentation of prompt programming strategies
- Optimization results for different prompting approaches
- Integration of DSPy with your RAG pipeline

Part 3: Retrieval Strategies Implementation (15%)

Objective: Develop efficient information retrieval strategies for your RAG application.

Requirements:

- Choose one of the following retrieval approaches:
    - **Option A**: Implement a vector database-based retrieval system (Pinecone, Weaviate, Chroma, etc.)
    - **Option B**: Implement keyword-based or lightweight retrieval methods without a vector database
- Design effective chunking and context management strategies
- Create query formulation techniques to enhance retrieval accuracy
- Implement re-ranking or hybrid retrieval approaches
- Design citation and source attribution mechanisms

Vector Database Approach Resources:

- Pinecone Documentation
- Weaviate Documentation

- Chroma Documentation
- LlamaIndex Documentation

Alternative Retrieval Approach Resources:

- FAISS Library
- Rank-BM25 Library
- LangChain Documentation

Deliverables:

- Implementation of your chosen retrieval strategy
- Context management and query planning code
- Documentation justifying your retrieval approach decision
- Evaluation comparing different retrieval methods for your domain

Part 4: Tool Use and Function Calling (15%)

Objective: Enhance your RAG system with specialized tools relevant to your industry domain.

Requirements:

- Implement at least 1 domain-specific tools/functions
- Create a proper function calling schema with appropriate validation
- Design tools that enhance the capabilities of your RAG system
- Implement error handling and fallback mechanisms

Suggested Industry-Specific Tools:

- **Healthcare**: Medical calculator tools, drug interaction checker, diagnostic flowchart navigator
- **Legal**: Citation formatter, case law similarity analyzer, legal timeline generator
- **Financial**: Financial calculator, market data retrieval, regulatory requirement checker
- **Education**: Citation generator, curriculum standards mapper, quiz/assessment generator
- **Environmental**: Carbon footprint calculator, climate data visualizer, sustainability metric analyzer

Resources:

- Anthropic Claude Documentation
- LangChain GitHub Repository
- LlamaIndex GitHub Repository
- Pydantic Documentation

Deliverables:

- Implementation of domain-specific tools
- Function schemas with proper validation
- Integration of tools with RAG system
- Documentation of tool design decisions and use cases

Part 5: Model Context Protocol Integration (10%)

Objective: Implement MCP for structured interactions with the underlying LLM.

Requirements:

- Implement the Model Context Protocol pattern
- Create structured prompts for different types of queries
- Design appropriate XML schemas for your domain
- Implement response parsing and validation

Resources:

- Anthropic Claude Documentation
- Claude API Reference
- XML Best Practices

Deliverables:

- Implementation of MCP integration
- Documentation of XML schemas and their purpose
- Prompt templates for different query types
- Evaluation of MCP effectiveness in your application

Part 6: User Interface and Response Streaming (10%)

Objective: Create an intuitive UI with streaming response capabilities (OPTIONAL)

Requirements:

- Develop a web-based UI for interacting with your RAG system
- Implement streaming responses for improved user experience
- Create appropriate visualization components for your domain
- Design effective source citation display

Resources:

- Streamlit Documentation
- Gradio Documentation
- React Documentation
- Claude API Reference

Deliverables:

- Functional web-based UI
- Implementation of response streaming
- Domain-specific visualization components
- User experience documentation and design decisions

Part 7: Comprehensive Evaluation and Verification (15%)

Objective: Rigorously evaluate the performance of your RAG system and verify the quality of responses.

Requirements:

- Implement an evaluation harness using DSPy's evaluation capabilities
- Create automated verification mechanisms for factual accuracy
- Design domain-specific evaluation metrics (relevance, correctness, helpfulness)
- Implement self-critique and self-correction mechanisms
- Perform comparative analysis against baseline approaches

Resources:

- DSPy GitHub Repository
- RAGAS GitHub Repository
- LangChain GitHub Repository
- Claude Documentation

Deliverables:

- Comprehensive evaluation framework with automated metrics
- Fact-checking and verification system implementation
- Documentation of evaluation methodology and results
- Analysis comparing different prompting and retrieval strategies
- User study design and results (if applicable)

Technical Requirements

- **Programming Language**: Python 3.9+
- **LLM Integration**
- **Documentation**: Comprehensive documentation of all components
- **Testing**: Unit tests and integration tests for critical components

Additional Resources

DSPy and Prompt Programming:

- DSPy GitHub Repository
- DSPy: Programming with Foundation Models (arXiv Paper)

RAG Architecture:

- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (arXiv Paper)
- Self-RAG: Learning to Retrieve, Generate, and Critique (arXiv Paper)

API Integration:

- Python Requests Library

- FastAPI Documentation

LLM Resources:

- Anthropic Claude Documentation
- OpenAI API Documentation

UI Frameworks:

- Streamlit Documentation
- Gradio Documentation

Evaluation Tools:

- RAGAS GitHub Repository
- TruLens GitHub Repository

Deployment Options:

- Docker Documentation
- GitHub Actions Documentation