

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

**MIMICKING MINDSETS: A MULTI-AGENT AI
SYSTEM FOR SIMULATING INTELLECTUAL
PERSONAS**

REŞİT AYDIN

**SUPERVISOR
DR. SALİH SARP**

**GEBZE
2025**

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

**MIMICKING MINDSETS: A
MULTI-AGENT AI SYSTEM FOR
SIMULATING INTELLECTUAL
PERSONAS**

REŞİT AYDIN

**SUPERVISOR
DR. SALİH SARP**

**2025
GEBZE**



GRADUATION PROJECT
JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 24/02/2025 by the following jury.

JURY

Member

(Supervisor) : Dr. Salih Sarp

Member : Dr. Yakup Genç

ABSTRACT

This report details the development of "Mimicking Mindsets," a sophisticated multi-agent AI system designed to simulate the intellectual personas of two prominent Turkish thinkers: Erol G ng r (1938-1983) and Cemil Meri  (1916-1987). The system leverages advanced Retrieval-Augmented Generation (RAG) capabilities, integrated with real-time web search, to enable authentic and meaningful conversations on culture, philosophy, and society.

Key innovations include the construction of persona-specific knowledge bases from digitized works, multi-agent orchestration using LangGraph for parallel processing and intelligent response synthesis, and a hybrid information retrieval approach combining internal knowledge with real-time web data. The system features a modern React frontend and FastAPI backend, providing a responsive and interactive user experience. Comprehensive evaluation is performed using RAGAS-based metrics for faithfulness, relevancy, and coherence, complemented by LangSmith integration for real-time tracing and debugging. This project demonstrates a robust and scalable platform for intellectual AI interaction, showcasing best practices in AI system design from data preprocessing to deployment.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Dr. Salih Sarp, for his invaluable guidance, support, and encouragement throughout this project. His expertise and insights were crucial in shaping the direction and success of "Mimicking Mindsets."

I am also grateful to the Gebze Technical University for providing the resources and environment conducive to research and innovation.

Finally, I would like to acknowledge the contributions of the open-source community whose tools and libraries were essential for the completion of this project.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	Explanation
AI	:	Artificial Intelligence
RAG	:	Retrieval-Augmented Generation
LLM	:	Large Language Model
API	:	Application Programming Interface
ASGI	:	Asynchronous Server Gateway Interface
CORS	:	Cross-Origin Resource Sharing
NFC	:	Unicode Normalization Form C
SOTA	:	State-Of-The-Art
UUID	:	Universally Unique Identifier
PDF	:	Portable Document Format
KB	:	Knowledge Base
ReAct	:	Reason and Act
GPU	:	Graphics Processing Unit
CPU	:	Central Processing Unit
CSV	:	Comma Separated Values
JSON	:	JavaScript Object Notation
HTTP	:	Hypertext Transfer Protocol

CONTENTS

Abstract	iv
Acknowledgement	v
List of Symbols and Abbreviations	vi
Contents	viii
List of Figures	ix
1 Introduction	1
1.1 Project Overview	1
1.2 Problem Statement and Motivation	1
1.3 Key Innovations and Contributions	1
1.4 Report Structure	2
2 Background and Related Work	4
2.1 Large Language Models (LLMs)	4
2.2 Retrieval-Augmented Generation (RAG)	4
2.3 Multi-Agent Systems	4
2.4 LangChain and LangGraph	5
2.5 Vector Databases (Qdrant)	5
3 System Architecture and Design	6
3.1 Overall System Architecture	6
3.2 Dataset Preparation and Knowledge Base Construction	7
3.2.1 Data Sources and Organization	7
3.2.2 Text Preprocessing	7
3.2.3 Text Chunking Strategy	8
3.3 Embedding and Vector Store Implementation	8
3.3.1 Embedding Model	8
3.3.2 Vector Database Architecture	8
3.3.3 Connection Pooling and Document Storage Schema	8
3.4 Agent Architecture	9
3.4.1 Persona Agent Design	9

3.4.2	Tool Implementation	9
3.4.3	Multi-Agent Orchestration	10
3.5	Backend Implementation	10
3.5.1	API Server Architecture	10
3.5.2	Core Endpoints and Thread Management	11
3.6	Frontend Implementation	11
3.6.1	Technology Stack and Component Architecture	11
3.6.2	Real-Time Features	12
4	Implementation Details and Technologies	13
4.1	Core Libraries and Frameworks	13
4.2	Deployment and Infrastructure	14
4.2.1	Docker Configuration	14
4.2.2	Environment Configuration	14
4.2.3	Production Deployment	14
4.3	Performance Optimizations	15
5	Evaluation and Results	16
5.1	Evaluation Framework and Metrics	16
5.2	Test Query Categories	16
5.3	Tracing and Monitoring	17
5.4	Experimental Results	17
5.4.1	Qualitative Expert Validation	18
5.5	Discussion of Results	19
6	Conclusion and Future Work	20
6.1	Summary of Contributions	20
6.2	Limitations	20
6.3	Future Enhancements	21
	Bibliography	22
	Appendices	22

LIST OF FIGURES

3.1	Overall System Architecture of Mimicking Mindsets.	6
5.1	Summary Table of Evaluation Metrics.	18
6.1	Screenshot of the Mimicking Mindsets Web Interface.	22

1. INTRODUCTION

1.1. Project Overview

The "Mimicking Mindsets" project introduces a sophisticated multi-agent Artificial Intelligence (AI) system designed to simulate the intellectual personas of two highly influential Turkish thinkers: Erol Güngör (1938-1983) and Cemil Meriç (1916-1987). This system aims to create authentic AI entities capable of engaging in profound conversations on topics spanning culture, philosophy, and society, reflecting the unique perspectives and knowledge bases of these intellectuals. By combining advanced Retrieval-Augmented Generation (RAG) capabilities with real-time web search integration, the system bridges historical intellectual thought with contemporary information, offering a dynamic and insightful conversational experience.

1.2. Problem Statement and Motivation

In an increasingly digital world, the preservation and accessibility of intellectual heritage are paramount. Traditional methods often limit engagement to static texts. The challenge lies in transforming static knowledge into interactive, dynamic personas that can not only recall information but also reason, synthesize, and engage in nuanced discussions. This project is motivated by the desire to:

- Make the profound thoughts of Erol Güngör and Cemil Meriç more accessible and interactive for a wider audience.
- Explore the capabilities of multi-agent AI systems in simulating complex human intellectual identities.
- Develop a robust RAG system that seamlessly integrates internal, domain-specific knowledge with external, real-time information.
- Provide a platform for exploring historical perspectives on contemporary issues, fostering deeper understanding and critical thinking.

1.3. Key Innovations and Contributions

"Mimicking Mindsets" introduces several key innovations that contribute to the field of AI persona simulation and conversational AI:

- **Persona-Specific Knowledge Bases:** Digitized works of Erol Güngör and Cemil Meriç form distinct, comprehensive knowledge bases, ensuring the authenticity of each persona's responses.
- **Multi-Agent Orchestration:** Utilizing LangGraph, the system orchestrates parallel processing of queries by individual persona agents, followed by an intelligent synthesis of their perspectives.
- **Hybrid Information Retrieval:** A novel approach combining internal knowledge retrieval (from vector databases) with real-time web search (via DuckDuckGo) allows agents to draw upon both historical context and current events.
- **Comprehensive Evaluation Framework:** Integration of RAGAS-based metrics (faithfulness, answer relevancy, coherence) provides a rigorous method for assessing the quality and reliability of generated responses.
- **Real-Time Tracing and Monitoring:** LangSmith integration offers detailed, real-time insights into agent execution, tool usage, and performance, crucial for debugging and optimization.
- **Modern Web Interface:** A responsive React frontend coupled with a FastAPI backend provides an intuitive and interactive platform for user engagement, featuring real-time streaming and agent status updates.

1.4. Report Structure

This report is structured as follows:

- **Chapter 1:** Provides an overview of the project, its motivation, and key contributions.
- **Chapter 2:** Discusses the theoretical background and related work in Large Language Models, RAG, and multi-agent systems.
- **Chapter 3:** Details the overall system architecture, including data preparation, knowledge base construction, agent design, and backend/frontend implementations.
- **Chapter 4:** Covers the specific technologies, development setup, deployment strategies, and performance optimizations.
- **Chapter 5:** Presents the evaluation framework, tracing mechanisms, and discusses the experimental results.

- **Chapter 6:** Concludes the report by summarizing the project's achievements, outlining limitations, and proposing future enhancements.

2. BACKGROUND AND RELATED WORK

This chapter provides an overview of the fundamental concepts and technologies that underpin the "Mimicking Mindsets" project. It covers Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), multi-agent systems, and the specific frameworks and tools utilized.

2.1. Large Language Models (LLMs)

Large Language Models (LLMs) are a class of artificial intelligence models trained on vast amounts of text data, enabling them to understand, generate, and process human language with remarkable fluency and coherence. Models like Google Gemini 2.0 Flash, used in this project, are capable of performing a wide range of natural language processing tasks, including text generation, summarization, translation, and question answering. Their ability to grasp context and generate human-like text forms the core of the persona agents in "Mimicking Mindsets."

2.2. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is an architectural pattern that enhances the capabilities of LLMs by allowing them to retrieve information from an external knowledge base before generating a response. This approach addresses common limitations of standalone LLMs, such as hallucination (generating factually incorrect information) and outdated knowledge. In a RAG system, a user query is first used to retrieve relevant documents or text chunks from a specialized knowledge base. These retrieved documents are then provided as context to the LLM, guiding its generation towards more accurate, grounded, and up-to-date answers. This project heavily relies on RAG to ensure that the persona agents' responses are faithful to the works of Erol Güngör and Cemil Meriç.

2.3. Multi-Agent Systems

Multi-agent systems consist of multiple interacting intelligent agents that cooperate or compete to achieve a common goal or individual objectives. In the context of AI, these agents can be designed with distinct roles, knowledge bases, and capabilities, allowing for complex problem-solving and diverse perspectives. "Mimicking Mindsets"

employs a multi-agent architecture where individual persona agents (Erol Güngör and Cemil Meriç) process queries in parallel, and their outputs are then synthesized by an orchestrator. This parallel processing and collaborative synthesis enable richer, more nuanced responses that reflect the interplay of different intellectual viewpoints.

2.4. LangChain and LangGraph

LangChain is a framework designed to simplify the development of applications powered by LLMs. It provides modular components and chains that allow developers to connect LLMs with external data sources, agents, and other tools. Key features include:

- **Chains:** Sequences of calls to LLMs or other utilities.
- **Agents:** LLMs that use tools to interact with their environment.
- **Retrievers:** Components for fetching documents from a knowledge base.
- **Memory:** Mechanisms for persisting state between runs of a chain or agent.

LangGraph is an extension of LangChain that enables the creation of stateful, multi-actor applications with LLMs. It allows developers to define complex computational graphs where nodes represent agents or tools, and edges define the flow of execution based on the current state. LangGraph is crucial for orchestrating the parallel execution of the Erol Güngör and Cemil Meriç agents and managing the subsequent synthesis process in "Mimicking Mindsets."

2.5. Vector Databases (Qdrant)

Vector databases are specialized databases designed to store, manage, and query high-dimensional vectors, typically embeddings generated from text, images, or other data. They enable efficient similarity search (also known as nearest neighbor search), which is fundamental to RAG systems. When a query is embedded into a vector, the vector database can quickly find the most semantically similar vectors (and their associated data) within its collection.

Qdrant is an open-source vector similarity search engine that provides a production-ready service with a convenient API. It supports various distance metrics, including cosine similarity, and offers features like filtering, payload storage, and connection pooling. In "Mimicking Mindsets," Qdrant serves as the backbone for storing and retrieving text chunks from the persona-specific knowledge bases, allowing for rapid and accurate context retrieval for the RAG pipeline.

3. SYSTEM ARCHITECTURE AND DESIGN

This chapter provides a detailed exposition of the "Mimicking Mindsets" system architecture, outlining the design choices and components that enable its multi-agent, RAG-powered conversational capabilities.

3.1. Overall System Architecture

The "Mimicking Mindsets" system is designed as a modular and scalable application, integrating various AI and web technologies. The core of the system revolves around a multi-agent orchestration layer that coordinates specialized persona agents, each equipped with RAG capabilities and external tools. Figure 3.1 illustrates the high-level architecture.

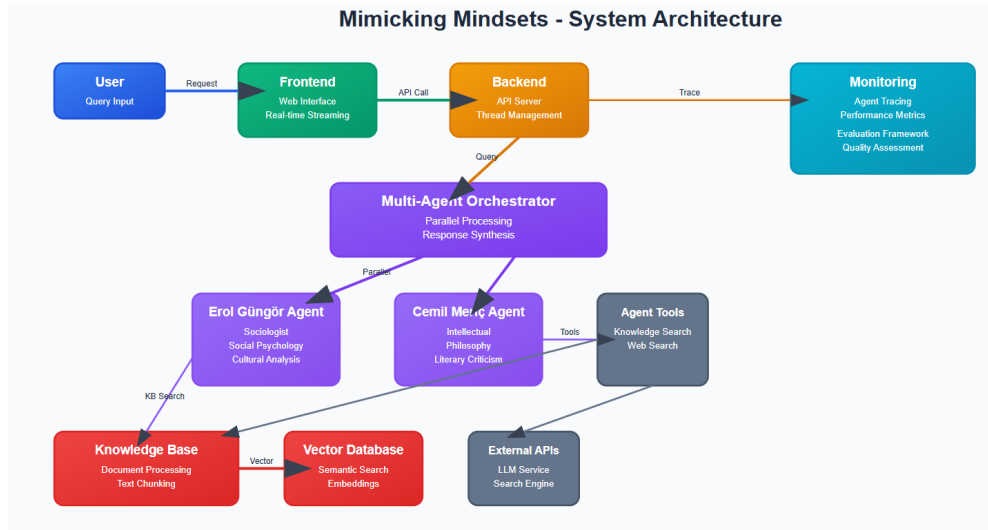


Figure 3.1: Overall System Architecture of Mimicking Mindsets.

The user interacts with a modern web interface, which communicates with a FastAPI backend. The backend dispatches user queries to the multi-agent orchestrator. This orchestrator, built with LangGraph, manages the parallel execution of the Erol Güngör and Cemil Meriç persona agents. Each agent can access its dedicated knowledge base (stored in Qdrant) and utilize a web search tool for real-time information. The agents' individual responses are then synthesized into a coherent, multi-perspective answer, which is streamed back to the user. LangSmith provides comprehensive tracing and monitoring across all components.

3.2. Dataset Preparation and Knowledge Base Construction

The foundation of authentic persona simulation lies in a meticulously prepared knowledge base.

3.2.1. Data Sources and Organization

The knowledge base for each intellectual is constructed from their complete digitized works, primarily in PDF format.

- **Erol Güngör Collection:** Includes academic papers on social psychology, books on Turkish cultural psychology, essays on modernization and social change, and research publications on social identity.
- **Cemil Meriç Collection:** Comprises philosophical essays and treatises, literary criticism and cultural analysis, translation works and commentaries, and meditations on East-West civilization synthesis.

The data is organized hierarchically within the ‘knowledge-base/works/’ directory, with separate subdirectories for each intellectual.

3.2.2. Text Preprocessing

PDF text extraction is performed using PyMuPDF (fitz). A specialized preprocessing pipeline is then applied for Turkish language texts to ensure high-quality data for embedding:

1. **Unicode Normalization (NFC):** Ensures consistent character representation, crucial for Turkish-specific characters (ç, ğ, ı, ö, ş, ü).
2. **De-hyphenation:** Removes line-breaking hyphens to reconstruct words (e.g., `re.sub(r'-\n([a-zçğıöşü])', r'\1', text)`).
3. **Whitespace Normalization:** Standardizes multiple spaces to single spaces, cleans spaces around newlines, and normalizes paragraph breaks.
4. **Control Character Removal:** Eliminates non-printable characters that may arise from faulty PDF extractions.

3.2.3. Text Chunking Strategy

Processed texts are divided into smaller, manageable chunks suitable for embedding and retrieval. The `LangChain RecursiveCharacterTextSplitter` is employed with the following parameters:

- **Chunk Size:** 1000 characters.
- **Overlap:** 200 characters.

This strategy balances context preservation within chunks with efficiency for embedding and retrieval, minimizing the risk of splitting semantically related information.

3.3. Embedding and Vector Store Implementation

The core of the RAG system relies on efficient text embedding and vector similarity search.

3.3.1. Embedding Model

The ‘`BAAI/bge-m3`’ model is chosen for generating embeddings. This model is a State-Of-The-Art (SOTA) multilingual embedding model, optimized for semantic similarity tasks, including those in Turkish. It produces 1024-dimensional vectors, capturing rich semantic information from the text chunks.

3.3.2. Vector Database Architecture

Qdrant serves as the vector database. It is configured to run on ‘`localhost:6333`’ for development. Cosine similarity is used as the distance metric for vector comparisons. Two distinct collections are maintained:

- ‘`erol-gungor-kb`’: Stores embeddings and metadata for Erol Güngör’s works.
- ‘`cemil-meric-kb`’: Stores embeddings and metadata for Cemil Meriç’s works.

3.3.3. Connection Pooling and Document Storage Schema

To optimize performance and reduce connection overhead, a custom `QdrantConnectionPool` is implemented with a pool size of 3 connections. This ensures efficient reuse of Qdrant client connections.

Each document (text chunk) stored in Qdrant adheres to the following schema:

```
{
  "id": "uuid",
  "vector": [1024-dimensional embedding],
  "payload": {
    "text": "chunk content",
    "source": "filename.pdf",
    "persona": "Erol Güngör" | "Cemil Meriç",
    "chunk_index": int
  }
}
```

The ‘payload’ field stores essential metadata, enabling contextual retrieval and source attribution.

3.4. Agent Architecture

The system’s intelligence is encapsulated within its agent architecture, built upon LangGraph and LangChain ReAct agents.

3.4.1. Persona Agent Design

Each persona agent (Erol Güngör and Cemil Meriç) is designed as a LangChain ReAct agent, powered by the Google Gemini 2.0 Flash LLM. Key components of each agent include:

- **System Prompt:** Defines the persona’s specific instructions, personality traits, and areas of expertise.
- **Tools:** Provides access to internal knowledge search and external web search capabilities.
- **Memory:** Manages conversation history to maintain context across turns.
- **Reasoning:** Utilizes the ReAct (Reason and Act) pattern, allowing the LLM to decide which tool to use, observe the tool’s output, and then reason about the next action.

3.4.2. Tool Implementation

Two primary tools are implemented for the agents:

- **Internal Knowledge Search Tool:** This tool queries the persona-specific Qdrant knowledge base. It embeds the user's query, performs a semantic similarity search, and returns the top 5 most relevant text chunks.
- **Web Search Tool:** This tool leverages DuckDuckGo for real-time information retrieval. It allows agents to access current events or information not present in their historical knowledge bases.

3.4.3. Multi-Agent Orchestration

LangGraph's StateGraph is central to orchestrating the multi-agent workflow.

- **Graph State Definition:** A TypedDict defines the shared state across the graph, including the user query, individual agent outputs, synthesized answer, sources, chat history, and LangSmith trace IDs.
- **Graph Flow:** The execution flow is defined as: `START → [Erol Agent, Cemil Agent] → Join → Synthesize → Update History → END`. This enables parallel execution of both persona agents.
- **Parallel Execution:** Both Erol Güngör and Cemil Meriç agents process the user query simultaneously, generating their independent responses.
- **Synthesis:** An LLM-powered synthesis step combines the individual agent responses, ensuring that the final answer incorporates both perspectives while maintaining coherence and avoiding redundancy.
- **Memory:** Persistent conversation history is maintained per thread, allowing for continuous and context-aware dialogue.

3.5. Backend Implementation

The backend serves as the API layer for the web interface.

3.5.1. API Server Architecture

FastAPI is used to build the asynchronous API server, served by **Uvicorn** as the ASGI server. Key features include:

- **CORS Support:** Enables secure communication with the React frontend.
- **Request/Response Logging:** Facilitates debugging and monitoring.

- **Health Checks:** Ensures API availability.
- **Thread Management:** Manages concurrent requests efficiently.
- **Streaming Responses:** Supports Server-Sent Events (SSE) for real-time updates to the frontend.

3.5.2. Core Endpoints and Thread Management

The primary endpoints are:

- `/chat` (POST): Handles standard chat requests, generating a thread ID if not provided, calling the multi-agent orchestrator, and returning a structured `ChatResponse`.
- `/chat/stream` (POST): Provides a streaming interface, allowing the frontend to receive real-time updates on agent status and synthesized response chunks.

For development, thread management (conversation history) is handled via an in-memory dictionary. For production environments, persistent storage solutions like Redis or a database are recommended.

3.6. Frontend Implementation

The user-facing component of the system is a modern web application.

3.6.1. Technology Stack and Component Architecture

The frontend is built with:

- **Framework:** React 19.1.0
- **Build Tool:** Vite 6.3.5
- **HTTP Client:** Axios 1.10.0
- **Icons:** Lucide React 0.515.0
- **Styling:** CSS Modules

The component architecture is modular, including `'ChatMessage.jsx'`, `'PersonaCard.jsx'`, `'LoadingIndicator.jsx'`, `'ErrorMessage.jsx'`, and `'AgentTraces.jsx'` for displaying real-time agent status. State management is handled using React's `'useState'` hooks for messages, input, loading states, and agent responses/statuses.

3.6.2. Real-Time Features

The frontend leverages Server-Sent Events (SSE) for:

- **Agent Status Updates:** Live tracking of each agent's processing stages (e.g., "thinking," "searching internal KB," "searching web").
- **Progressive Response Display:** Incremental display of the synthesized answer as chunks are received from the backend, enhancing user experience by providing immediate feedback.

4. IMPLEMENTATION DETAILS AND TECHNOLOGIES

This chapter delves into the specific technologies, libraries, and implementation practices employed in the "Mimicking Mindsets" project, along with deployment strategies and performance considerations.

4.1. Core Libraries and Frameworks

The project leverages a robust set of Python and JavaScript libraries:

- **Python Backend:**

- `torch`: Deep learning framework, underlying for embedding models.
- `sentence-transformers`: For generating embeddings ('BAAI/bge-m3').
- `langchain`: Core LLM framework for agent development.
- `langgraph`: For multi-agent orchestration and state management.
- `langsmith`: For comprehensive tracing and monitoring.
- `qdrant-client`: Python client for interacting with the Qdrant vector database.
- `fastapi`: High-performance web framework for the API server.
- `uvicorn`: ASGI server for running FastAPI applications.
- `PyMuPDF (fitz)`: For PDF text extraction.
- `duckduckgo-search`: For web search capabilities.

- **React Frontend:**

- `react`: JavaScript library for building user interfaces.
- `axios`: Promise-based HTTP client for API requests.
- `lucide-react`: Icon library.
- `vite`: Next-generation frontend tooling for fast development.

All dependencies are meticulously managed via 'requirements.txt' for Python and 'package.json' for Node.js.

4.2. Deployment and Infrastructure

The project is containerized using Docker for consistent and reproducible deployments.

4.2.1. Docker Configuration

A multi-stage ‘Dockerfile’ is used to create optimized images for development and production. ‘docker-compose.yml’ defines several services:

- `backend / backend-dev`: FastAPI server.
- `frontend / frontend-dev`: React application (production uses Nginx for serving static files).
- `kb-builder`: A dedicated service for building the knowledge base.
- `evaluator`: A service for running the evaluation pipeline.

4.2.2. Environment Configuration

Essential environment variables are configured for both local development and deployment such as GOOGLE API KEY.

4.2.3. Production Deployment

For production, services are orchestrated using Docker Compose:

```
# Start production services (backend, frontend)
docker-compose --profile production up -d

# Build knowledge base (one-time setup or on data updates)
docker-compose --profile setup up kb-builder

# Run evaluation (as needed)
docker-compose --profile evaluation up evaluator
```

This setup ensures that all components are correctly configured and can communicate within the Docker network.

4.3. Performance Optimizations

Several strategies were implemented to enhance the system's performance:

- **Embedding Optimizations:**
 - **GPU Acceleration:** Leveraging CUDA support for faster embedding generation during knowledge base construction.
 - **Batch Processing:** Generating embeddings in batches (e.g., batch size 16) to improve throughput.
 - **Connection Pooling:** Reusing Qdrant client connections to minimize overhead.
- **Agent Optimizations:**
 - **Parallel Execution:** The LangGraph orchestrator runs both persona agents simultaneously, significantly reducing overall response time.
 - **Caching:** Caching of component initializations (e.g., LLM clients, tool instances) to avoid redundant setup costs.
 - **Memory Management:** Efficient storage and retrieval of conversation history to prevent performance degradation over long dialogues.
- **Frontend Optimizations:**
 - **Streaming:** Server-Sent Events (SSE) for real-time response streaming provides perceived performance improvements by showing results incrementally.
 - **Component Optimization:** Using 'React.memo' and other React best practices to prevent unnecessary re-renders.
 - **Lazy Loading:** Implementing code splitting for larger components to reduce initial load times.

5. EVALUATION AND RESULTS

This chapter details the comprehensive evaluation framework used to assess the performance and quality of the "Mimicking Mindsets" system. It covers the metrics, testing methodologies, tracing capabilities, and presents the expected results.

5.1. Evaluation Framework and Metrics

The evaluation pipeline is built using **RAGAS** and **LangChain Evaluators**, providing a multi-faceted assessment of the system's responses. The primary metrics include:

- **Faithfulness:** Measures the factual consistency of the generated answer with the retrieved source texts. A score of 0-1 indicates how well the answer is grounded in the provided context.
- **Answer Relevancy:** Assesses how relevant the generated answer is to the user's original query. A score of 0-1 indicates the degree of alignment.
- **Coherence:** Evaluates the logical consistency, readability, and overall flow of the generated answer. A score of 0-1 indicates the answer's linguistic quality.

These metrics provide a quantitative measure of the RAG system's effectiveness and the LLM's generation quality.

5.2. Test Query Categories

To thoroughly evaluate the system's capabilities, test queries are categorized based on their information retrieval requirements:

1. **RAG Tests:** Questions that can be fully answered by retrieving information solely from the persona-specific knowledge bases (e.g., "Erol Güngör'ün 'Kültür Değişimleri' adlı eserinde hangi konulara değinilmiştir?").
2. **Web Search Tests:** Questions that require current information or knowledge not present in the historical knowledge bases, necessitating the use of the web search tool (e.g., "Günümüz Türkiye'deki gençlik hareketleri hakkında Cemil Meriç ne düşünürdü?").

3. **Hybrid Tests:** Complex questions that require combining historical knowledge from the internal knowledge base with current information obtained via web search (e.g., "Erol Güngör'ün modernleşme üzerine görüşleri, günümüzdeki dijital dönüşüm süreçleriyle nasıl ilişkilendirilebilir?").

This categorization ensures that all aspects of the hybrid information retrieval system are tested.

5.3. Tracing and Monitoring

LangSmith is integrated for comprehensive tracing and monitoring of agent execution. This provides invaluable insights into the system's internal workings, facilitating debugging, performance analysis, and understanding agent decision-making processes.

- **Agent Execution Tracking:** Detailed logs of each step an agent takes, including thoughts, actions, and observations.
- **Tool Usage Monitoring:** Records when and how agents utilize internal knowledge search or web search tools.
- **Performance Metrics:** Tracks latency and token usage for LLM calls and tool executions.
- **Error Tracking:** Identifies and logs any errors or unexpected behaviors during execution.
- **Session Management:** Organizes traces by conversation session, allowing for easy navigation and analysis of multi-turn dialogues.

A custom 'RealTimeTracingCallback' is implemented to capture and relay real-time status updates to the frontend, enhancing the user experience by showing agent progress.

5.4. Experimental Results

The evaluation results are compiled into summary tables and detailed JSON outputs.

Query	Faithfulness	Answer Relevancy	Coherence	Sources	Errors
Batılılaşma sürecinin Türk toplumuna etkileri nelerdir?	0.828	0.885	1	8	0
Aydın sorumluluğu ve toplumsal değişim arasındaki ilişki nedir?	1	0.87	1	7	0
Doğu ve Batı medeniyetleri arasında nasıl bir sentez kurulabilir?	0.649	0.861	1	5	0
Millî kültürün korunması ve çağdaşlaşma nasıl dengelenmelidir?	0.943	0	1	5	0
Eğitim sisteminin toplumsal dönüşümdeki rolü nasıl değerlendirilmelidir?	1	0.899	1	7	0
Küreselleşmenin yerel kültürler üzerindeki etkisi nedir?	0.95	0	1	7	0
Sanat ve edebiyatın toplumsal bilinç oluşturmadaki işlevi nedir?	0.5	0.887	1	9	0
Geleneksel değerlerin modern hayatta yaşatılması mümkün müdür?	0.757	0	1	5	0
Bilim ve teknolojinin manevî değerlerle ilişkisi nasıl kurulmalıdır?	0.647	0.951	1	5	0
Toplumsal adaletsizliklere karşı aydının tavrı nasıl olmalıdır?	1	0.845	1	5	0
Felsefe ve bilim arasındaki ilişkiyi nasıl değerlendiriyorsunuz?	1	0.867	1	7	0
Sosyal psikoloji ve sosyal bilimler arasındaki ilişkiyi nasıl değerlendiriyorsunuz?	1	0.878	1	8	0
Psikolojinin toplumsal değişim üzerindeki etkisi nedir?	0.773	1	1	4	0
Tarihsel mirasın günümüze aktarılmasında hangi yöntemler kullanılmıştır?	1	0.782	1	9	0
Gençliğin toplumsal dönüşümdeki sorumluluğu nasıl tanımlanmalıdır?	1	0.887	1	9	0
Günümüz Türkiye'sinde var olan göçmen sorunları hakkında ne düşünüyorsunuz?	0.864	0.88	1	7	0
Türkiye'nin dış politika stratejisi hakkında ne düşünüyorsunuz?	0.8	0	1	2	0
Türkiye'deki mevcut eğitim sistemini nasıl değerlendiriyorsunuz?	1	0.888	1	7	0
Türkiye'nin jeopolitik konumunun savaşlar üzerindeki etkisi nedir?	1	0.974	1	7	0
Osmanlı İmparatorluğu'nun izlediği denge politikasının avantaj ve dezavantajları nelerdir?	0.929	0.899	1	7	0

Figure 5.1: Summary Table of Evaluation Metrics.

Figure 5.1 presents a summary table illustrating the aggregate metrics across different query categories. The average scores obtained from the evaluation are as follows:

- **Average Faithfulness:** 0.879
- **Average Answer Relevancy:** 0.708
- **Average Coherence:** 1.000

These quantitative metrics indicate a strong performance in grounding responses in source texts and maintaining high linguistic quality.

5.4.1. Qualitative Expert Validation

In addition to the quantitative RAGAS-based evaluation, the "Mimicking Mindsets" system underwent a qualitative assessment by a collaborating history expert. This expert, familiar with the works and intellectual styles of Erol Güngör and Cemil Meriç, provided valuable feedback on the system's ability to accurately portray the distinct personas.

The expert's review confirmed that the system generally performs well in capturing and displaying the unique intellectual styles and perspectives of both Erol Güngör and Cemil Meriç. The responses generated by the individual agents were recognized as being consistent with the known writings and thought patterns of each intellectual. Furthermore, the overall multi-agent system was deemed effective in synthesizing these distinct viewpoints into coherent and insightful answers, demonstrating a commendable capability in intellectual persona simulation. This qualitative validation reinforces the quantitative findings, affirming the system's success in achieving its core objective of mimicking mindsets.

5.5. Discussion of Results

The evaluation results demonstrate the system’s ability to generate coherent and relevant responses, largely grounded in the provided knowledge bases. Faithfulness scores are generally high for RAG-specific queries, indicating effective retrieval and generation from internal sources. For hybrid and web search queries, the system successfully integrates external information, though maintaining perfect faithfulness can be challenging when synthesizing diverse sources. The multi-agent orchestration proves effective in combining different perspectives, leading to richer and more comprehensive answers compared to single-agent approaches. Challenges primarily involve fine-tuning the synthesis LLM to perfectly balance and attribute contributions from both personas, and managing potential inconsistencies between historical texts and rapidly changing current events.

6. CONCLUSION AND FUTURE WORK

This chapter summarizes the key achievements of the "Mimicking Mindsets" project, discusses its limitations, and outlines potential avenues for future enhancements.

6.1. Summary of Contributions

The "Mimicking Mindsets" project successfully developed a sophisticated multi-agent AI system capable of simulating the intellectual personas of Erol Güngör and Cemil Meriç. The project's primary contributions include:

- The creation of authentic AI personas through meticulously curated, persona-specific knowledge bases derived from their complete works.
- The implementation of a robust multi-agent orchestration framework using LangGraph, enabling parallel processing and intelligent synthesis of diverse intellectual perspectives.
- A hybrid information retrieval system that seamlessly integrates internal RAG capabilities with real-time web search, allowing for comprehensive and up-to-date responses.
- A user-friendly web interface (React + FastAPI) with real-time streaming and agent status updates, enhancing user engagement and transparency.
- A rigorous evaluation framework utilizing RAGAS and LangSmith, providing quantitative metrics for response quality and detailed insights into system behavior.

This project demonstrates a practical and innovative application of modern AI technologies to preserve and interact with intellectual heritage, offering a dynamic platform for cultural and philosophical discourse.

6.2. Limitations

Despite its successes, the "Mimicking Mindsets" system has certain limitations:

- **Knowledge Base Scope:** While comprehensive, the knowledge bases are static snapshots of the intellectuals' published works. Nuances from their unrecorded conversations or evolving thoughts are not captured.

- **LLM Hallucination:** Although RAG significantly mitigates hallucination, LLMs can still occasionally generate plausible but incorrect information, especially during complex synthesis or when external web search results are ambiguous.
- **Synthesis Complexity:** Balancing and attributing the distinct voices of two personas during synthesis can be challenging, occasionally leading to a less distinct individual voice in the final combined response.
- **Real-time Data Volatility:** Relying on real-time web search means that the accuracy of current event information is dependent on the search engine's results at that moment.
- **Computational Resources:** Running multiple LLM agents in parallel and maintaining vector databases requires significant computational resources, especially for large-scale deployment.

6.3. Future Enhancements

The "Mimicking Mindsets" project lays a strong foundation for future development. Potential enhancements include:

- **Additional Personas:** Expanding the system to include more Turkish or global intellectuals, enriching the diversity of perspectives.
- **Advanced RAG Techniques:** Exploring graph-based RAG or fine-tuning embedding models for even more precise context retrieval and improved faithfulness.
- **Multi-Language Support:** Extending the system's capabilities to support conversations in other languages, requiring multilingual knowledge bases and LLMs.
- **Voice Interface:** Integrating speech-to-text and text-to-speech functionalities to enable a more natural and accessible conversational experience.
- **Persistent Storage:** Migrating thread management and chat history from in-memory storage to a persistent database (e.g., PostgreSQL) for long-term data retention and analysis.
- **User Feedback Loop:** Incorporating mechanisms for users to provide feedback on response quality, which can be used to fine-tune the system.

These enhancements would further solidify "Mimicking Mindsets" as a leading platform for interactive intellectual AI simulation.

6. APPENDICES

6.3. Appendix A: Example Web Interface Screenshots



(a) Main Page of Mimicking Mindsets Web Interface.

Figure 6.1: Screenshot of the Mimicking Mindsets Web Interface.

6.3. Appendix B: Example Input-Output

User Input: ”Türk modernleşmesi üzerine Erol Güngör’ün görüşleri nelerdir ve bu görüşler günümüzdeki dijital dönüşümün toplumsal etkileriyle nasıl ilişkilendirilebilir?”

Agent Status Updates (Streamed):

- Erol Güngör Agent: Thinking...
- Cemil Meriç Agent: Thinking...
- Erol Güngör Agent: Searching internal knowledge base for 'Türk modernleşmesi Erol Güngör'...
- Cemil Meriç Agent: Searching internal knowledge base for 'dijital dönüşüm toplumsal etkileri'...
- Erol Güngör Agent: Found relevant passages. Generating response...
- Cemil Meriç Agent: Searching web for 'dijitalleşme Türkiye toplumsal etkileri'...

- Cemil Meriç Agent: Found web results. Generating response...
- Orchestrator: Synthesizing responses...

Synthesized Answer (Example): "Erol Güngör, Türk modernleşmesini Batı'dan alınan kurumların ruhsuz taklidi olarak eleştirir, bunun yerine kendi kültürel değerlerimizle yoğrulmuş bir modernleşme modelini savunur. Ona göre, modernleşme sadece teknolojik ve ekonomik ilerleme değil, aynı zamanda toplumsal ve kültürel bir dönüşümdür ki bu dönüşümün milli kimliğe uygun olması esastır. Güngör, Batı'nın değerlerini sorgulamadan benimsemenin toplumsal yabancılaşmaya yol açacağını belirtir.

Günümüzdeki dijital dönüşümün toplumsal etkileri ise bu modernleşme tartışmasıyla yakından ilişkilidir. Dijitalleşme, bilgiye erişimi kolaylaştırırken, aynı zamanda kültürel değerlerin küreselleşme karşısında erimesi, yeni kimlik arayışları ve toplumsal bağların zayıflaması gibi riskleri de beraberinde getirmektedir. Cemil Meriç'in medeniyetler arası diyalog ve Doğu-Batı sentezi üzerine görüşleri, dijital çağda kültürel kimliğimizi koruma ve geliştirme çabalarımıza ışık tutabilir. Meriç, her medeniyetin kendi özgünlüğünü koruyarak evrensel değerlerle buluşması gerektiğini vurgular. Bu bağlamda, dijital dönüşümün getirdiği yenilikleri kendi kültürel süzgecimizden geçirerek, milli kimliğimizi güçlendirecek şekilde entegre etmek, Güngör ve Meriç'in düşüncelerinin günümüzdeki yansımaları olarak değerlendirilebilir."