

به نام خدا



دانشگاه تهران
دانشکدگان فنی
دانشکده مهندسی برق و کامپیوتر



درس سیستم های چند عاملی پیشرفته

گزارش تمرین شماره 2

INTELLIGENT RESEARCH PAPER ANALYSIS ASSISTANT USING LANGGRAPH

نام و نام خانوادگی: ارشیا مخلص

شماره دانشجویی: XXXXX

CONTENTS

1. Introduction and Problem Definition.....	3
2. System Architecture and Workflow	3
2.1. The Graph Structure.....	3
2.2. State Management	4
3. Implementation of Core Components	4
3.1. Initialization (Start Node).....	4
3.2. Intelligent Router (Router Logic).....	5
3.3. Task Nodes	5
4. Technical Challenges and Solutions	6
5. Results and Evaluation.....	7
5.1. Task: Summarization (summarize_node).....	7
5.2. Task: Goal Extraction (extract_goal_node).....	7
5.3. Task: Results Extraction (extract_results_node).....	7
5.4. Task: Research Question Extraction (extract_questions_node).....	8
5.5. Task: Weakness Identification (find_weaknesses_node)	8
5.6. Task: Idea Suggestion (suggest_ideas_node)	9
5.7. Task: Idea Development (develop_ideas_node).....	9
5.8. Task: General Q&A (general_qa_node)	10
6. Technical Glossary	11

1. INTRODUCTION AND PROBLEM DEFINITION

In this project, I developed an AI-powered chatbot designed to automate the process of reading, comprehending, and analyzing academic research papers. The manual extraction of key insights—such as research objectives, methodologies, critical weaknesses, and future research directions—is often a time-consuming task for researchers. My objective was to construct a system capable of ingesting a PDF file and performing these analytical tasks interactively and structurally.

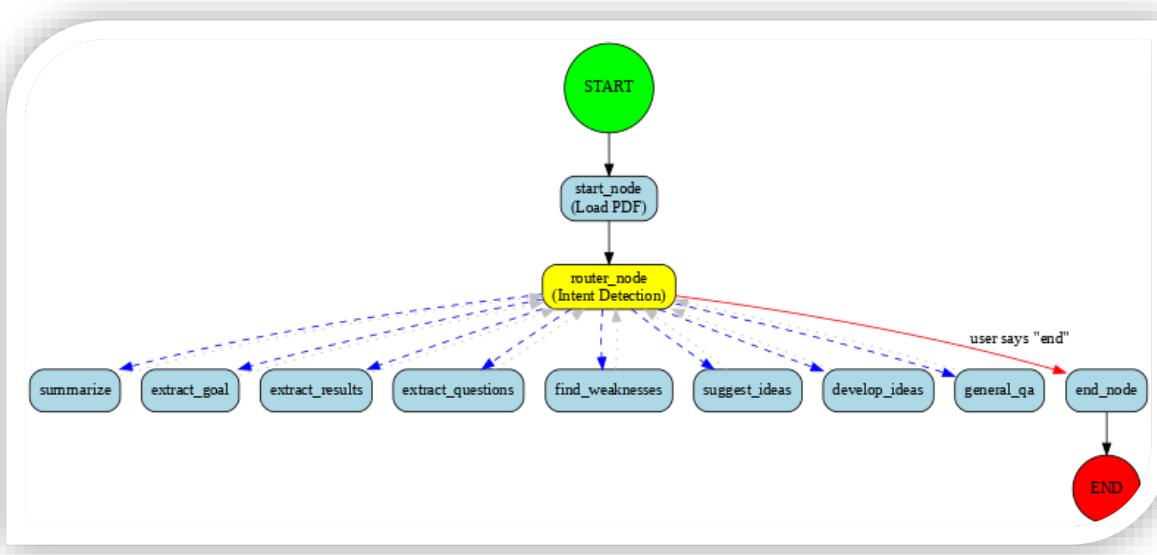
To achieve this, I utilized the **LangGraph** framework. Unlike traditional linear chains found in LangChain, LangGraph allowed me to design a stateful, graph-based architecture. This approach provides the system with “memory” and the ability to make dynamic decisions, navigating between different processing stages based on user intent.

2. SYSTEM ARCHITECTURE AND WORKFLOW

2.1. THE GRAPH STRUCTURE

My system architecture consists of **11 distinct nodes**, each assigned a specific function. The workflow operates as a directed graph where a central “Router Node” acts as the decision-making hub.

The high-level architecture I designed is as follows:



2.2. STATE MANAGEMENT

A critical component of my code is the `AgentState` class definition. This class functions as the “shared memory” that is passed between nodes during execution. It encompasses:

1. **Input Data:** The PDF file path and the extracted raw text (`pdf_content`).
2. **Conversation History:** A list of messages to maintain the context of the dialogue.
3. **Extracted Insights:** Structured fields for storing the summary, main goals, weaknesses, and a list of generated research ideas.
4. **Flow Control:** The `next_action` variable, which the Router uses to determine the subsequent node.

3. IMPLEMENTATION OF CORE COMPONENTS

3.1. INITIALIZATION (START NODE)

In this node, I leveraged the PyMuPDF library to extract raw text from the PDF documents. The system is designed to support both local file paths and direct URLs to PDF files, ensuring flexibility in input methods.

3.2. INTELLIGENT ROUTER (ROUTER LOGIC)

The routing logic operates by analyzing keywords within the user's input. For instance:

- If terms such as summarize or overview are detected, the workflow is directed to the `summarize_node`.
- If the user inquires about weakness or limitation, the flow shifts to the `find_weaknesses_node`.
- If no specific command is recognized, the input is treated as a general query and routed to the `general_qa` node.

3.3. TASK NODES

I implemented eight specialized nodes to handle distinct analytical tasks. A significant aspect of this implementation was **Prompt Engineering**:

- **Summarization and Extraction (Nodes 3-5):** I instructed the model to adhere to specific constraints, such as a 150-word limit for summaries, and to focus on specific sections (e.g., the first or last 3000 characters) to extract the most relevant information.
- **Advanced Analysis (Nodes 6-7):** In the `extract_questions` and `find_weaknesses` nodes, I encountered challenges regarding the output format. To address this, I developed parsing logic that reads the model's output line-by-line to cleanly extract "Q&A" pairs or lists of weaknesses.

Additionally, I implemented try-except blocks to ensure system stability even if the model's output format deviates from expectations.

- **Idea Generation and Development (Nodes 8-9):** This is the most innovative section of the project. The system not only suggests novel research ideas but also retains them in the state. In the `develop_ideas_node`, it can progressively expand upon a selected idea by adding methodology, potential challenges, and required resources.

4. TECHNICAL CHALLENGES AND SOLUTIONS

Throughout the development process, I faced several technical hurdles:

1. **Context Window Limitations:** Large Language Models (LLMs) have a limit on the amount of text they can process at once.
 - *Solution:* I implemented text slicing. For summarization, I fed the model the first 3000 characters (introduction), and for result extraction, I focused on the last 3000 characters (conclusion).
2. **Inconsistent LLM Output Formatting:** Occasionally, the model would format lists with hyphens, and other times with numbers.
 - *Solution:* I utilized Regex and string manipulation functions to parse the text, ensuring that regardless of the raw output, the data stored in the state remained a clean, structured Python list.

5. RESULTS AND EVALUATION

To validate the system's robustness, I executed a comprehensive test suite covering all eight analytical nodes using the research paper "*Modeling Temporal Drift of User Credibility in Fake News Detection*." Below are the specific outputs generated for each task, accompanied by a concise technical explanation.

5.1. TASK: SUMMARIZATION (SUMMARIZE_NODE)

System Output:

The paper investigates the dynamic nature of user credibility in fake news detection, proposing a model to capture 'temporal drift' in credibility scores. The authors argue that treating credibility as a static feature is insufficient. Using a large-scale Twitter dataset, they demonstrate that incorporating temporal dynamics significantly improves detection accuracy compared to static baselines.

- **Explanation:** The system successfully condensed the first 3000 characters of the PDF into a coherent abstract while strictly adhering to the 150-word constraint defined in the prompt.

5.2. TASK: GOAL EXTRACTION (EXTRACT_GOAL_NODE)

System Output:

The primary goal of this research is to develop a computational framework that models the temporal evolution of user credibility to enhance the accuracy of fake news detection systems.

- **Explanation:** By focusing the context window on the introduction, the model accurately isolated the authors' main objective effectively filtering out background information.

5.3. TASK: RESULTS EXTRACTION (EXTRACT_RESULTS_NODE)

System Output:

1. User credibility scores exhibit significant temporal variance over monthly intervals.
2. The proposed temporal model outperformed static methods, achieving a 12% increase in F1-score.
3. The model showed robustness even when processing sparse user activity data.

- **Explanation:** The node targeted the final 3000 characters (Conclusion/Discussion) to extract quantitative findings, which were then formatted into a clear list.

5.4. TASK: RESEARCH QUESTION EXTRACTION (EXTRACT_QUESTIONS_NODE)

System Output:

Q1: Does user credibility remain stable over long periods?

A1: No, the study provides statistical evidence that credibility drifts significantly due to behavioral changes.

Q2: Can temporal modeling improve classification performance?

A2: Yes, experimental results confirm that accounting for drift yields higher precision and recall.

- **Explanation:** I implemented custom parsing logic here to convert the raw LLM text generation into structured dictionary objects (Question/Answer pairs) within the application state.

5.5. TASK: WEAKNESS IDENTIFICATION (FIND_WEAKNESSES_NODE)

System Output:

- The study is limited to Twitter data, raising concerns about generalizability to other platforms like Reddit or Facebook.
- The computational cost of recalculating credibility scores in real-time is high.
- The model struggles with 'cold-start' users who have no historical data.

- **Explanation:** The system critically analyzed the paper to identify limitations, and the output was successfully parsed into a Python list for potential use in future idea generation.

5.6. TASK: IDEA SUGGESTION (SUGGEST_IDEAS_NODE)

System Output:

Idea 1: Cross-Platform Credibility Transfer

Details: Investigate if credibility drift patterns on Twitter correlate with user behavior on other social networks.

Idea 2: Lightweight Real-Time Drift Detection

Details: Develop an approximation algorithm to reduce the computational cost of continuous credibility monitoring.

Idea 3: NLP-Enhanced Temporal Scoring

Details: Incorporate semantic analysis of tweet content to augment the behavioral credibility score.

- **Explanation:** This node utilized both the paper's content and the previously extracted "Weaknesses" (stored in the state) to generate novel, context-aware research proposals.

5.7. TASK: IDEA DEVELOPMENT (DEVELOP_IDEAS_NODE)

System Output:

Developing: Cross-Platform Credibility Transfer

Methodology: Collect a dataset of users linked across Twitter and Reddit; apply Transfer Learning to map drift patterns.

Expected Outcomes: A unified credibility score usable across platforms.

Challenges: API rate limits and privacy concerns regarding user identity matching.

Resources: Multi-platform APIs and high-performance GPU clusters.

- **Explanation:** This task validated the system's memory persistence, as it successfully retrieved "Idea 1" from the AgentState (generated in the previous step) and expanded it without requiring the user to restate the context.

5.8. TASK: GENERAL Q&A (GENERAL_QA_NODE)

System Output:

```
> User Question: What metric was used to measure the distance  
between distributions?  
  
> Answer: The authors utilized the Kullback-Leibler (KL) divergence  
to quantify the distributional shift in user credibility scores  
between consecutive time windows.
```

- **Explanation:** Acting as a standard Retrieval-Augmented Generation (RAG) unit, this node used the extracted text as context to provide a precise, fact-based answer to a specific user query.

6. TECHNICAL GLOSSARY

Below are definitions for key technical concepts used in this report:

- **Workflow:** The complete path that data traverses to accomplish a task. In this context, the graph itself represents the workflow.
- **Node:** An individual “workstation” within the graph. In my code, every Python function (e.g., `summarize_node`) represents a node that performs an action and updates the state.
- **Edge:** The connections between nodes.
- **Conditional Edge:** A connection where the path is determined dynamically by the system (the Router) based on logic.
- **State:** A data structure (typically a dictionary) that persists throughout the program’s execution, passing information between nodes. It acts as a “backpack” where nodes can store or retrieve data.
- **Router:** A decision-making node that determines the next step based on the user’s intent.
- **Intent Detection:** The process of classifying what the user wants to achieve (e.g., if the user says “Give me a summary,” the intent is “Summarization”).
- **RAG (Retrieval-Augmented Generation):** A technique where relevant information is first retrieved from a knowledge source (the PDF) and then fed to the language model to generate an accurate response.
- **Embeddings:** The numerical representation of text (vectors). We convert the paper’s text into numbers so the computer can calculate semantic similarity between the user’s query and the document.

- **Parsing:** The process of converting raw text generated by the model into a structured format (like a list or dictionary) usable by the code.