AI Agent Architecture Document

Research Paper Summarizer AI Agent

Introduction

This document describes the architecture, components, and operational flow of the AI Agent developed for automated research paper retrieval and summarization. The system combines LangGraph-based orchestration, transformer-based language models, and academic APIs to generate accurate summaries of research papers based on a user-specified topic.

The architecture ensures modularity, explainability, and extensibility — key principles in modern AI agent design.

System Architecture Overview

The AI agent follows a **modular pipeline architecture** where each stage handles a specific function. The architecture consists of interconnected nodes managed through **LangGraph**, which defines transitions between tasks.

Core Components

Component	Function
User Interface (Input Layer)	Takes user input for research topic and desired number of papers.
Title Generation Node	Uses an LLM to generate topic-related academic-style paper titles.
Paper Retrieval Node	Fetches research papers from the Semantic Scholar API based on generated titles.
Summarization Node	Uses either a pre-trained or fine-tuned LLM to generate 150–200 word summaries.
Evaluation Module	Evaluates the quality of generated summaries using ROUGE and BERTScore metrics.
Fine-Tuning Module (Optional)	Fine-tunes a base model (Flan-T5) using LoRA for improved summarization quality.

Component	Function
LangGraph Orchestrator	Defines workflow, state transitions, and dependency handling between all nodes.

Interaction Flow

Step-by-Step Execution Flow

1. **User Input Stage**
    - The user provides a **research topic** and specifies how many papers to summarize (e.g., 3).
    - Input data is packaged into a structured object (PaperInfo).
2. **Title Generation Stage**
    - The system invokes the **generate_titles** node.
    - The **LLM** generates several academic-style titles to broaden the search coverage for relevant papers.
3. **Paper Retrieval Stage**
    - Each generated title is sent to the **Semantic Scholar API** to retrieve metadata (title, abstract, citation count, and URL).
    - Results are appended to the PaperInfo state.
4. **Summarization Stage**
    - The **draft_answer** node takes abstracts and passes them to the selected model (either Hugging Face endpoint or fine-tuned LoRA model).
    - The model generates structured summaries including title, citation info, and key findings.
5. **Evaluation Stage**
    - The generated summaries are evaluated using the **evaluate_summary()** function.
    - ROUGE measures word-level similarity, while BERTScore evaluates semantic similarity.
6. **Output Stage**

- The system prints and stores both the final summarized results and evaluation metrics.
- The user can optionally fine-tune the model for improved domain performance.

Flow Description

User → Title Generation → Paper Retrieval → Summarization → Evaluation → Output

The data flows linearly, but LangGraph allows **state-driven orchestration**, meaning nodes can be reused or reordered for other summarization workflows (e.g., multi-document synthesis or literature clustering).

Models Used

Model / API	Purpose	Details
openai/gpt-oss-safeguard-120b (via Hugging Face Endpoint)	Title generation and zero-shot summarization	Large transformer-based generative model capable of handling long context inputs.
google/flan-t5-base (Fine-tuned)	Scientific summarization	Base model fine-tuned on scientific_papers dataset using **LoRA** for domain-specific text summarization.
semantic-scholar/graph/v1 API	Paper retrieval	Fetches paper abstracts, titles, and citations relevant to user topic.
BERT (via bert-score package)	Evaluation	Calculates contextual similarity between generated and reference summaries.
ROUGE scorer (rouge_score library)	Evaluation	Measures lexical overlap between generated summaries and ground truth.

Design Choices and Rationale

1.LangGraph Framework

- Enables **modular and declarative orchestration** of AI tasks.
- Provides **START–END node structure** for clean, traceable pipelines.

- Simplifies management of complex task dependencies like title generation, fetching, and summarization.

2.Model Selection

- **Flan-T5 Base:**
    - Instruction-tuned and efficient for summarization.
    - Strong few-shot and zero-shot capabilities.
- **LoRA Fine-Tuning:**
    - Allows **parameter-efficient fine-tuning** by updating only low-rank layers.
    - Reduces compute cost while improving accuracy for domain-specific data (scientific texts).

3.API Integration

- **Semantic Scholar API** provides reliable access to scholarly metadata without scraping.
- Ensures reproducibility and scalability for different academic topics.

4.Evaluation Metrics

- **ROUGE** quantifies textual overlap — suitable for quick lexical comparison.
- **BERTScore** captures **semantic fidelity**, which is crucial for evaluating paraphrased or conceptually accurate summaries.

5.Modularity and Extensibility

- Each node can be swapped independently — e.g., replacing Flan-T5 with GPT-3.5, or adding a citation-analysis module.
- Future extensions can include clustering similar papers or generating literature reviews automatically

---

Interaction Diagram (Textual Description)

[User Input]

↓

[Title Generation Node] -- (LLM) --> Generates topic titles

↓

[Paper Retrieval Node] -- (API) --> Fetches papers from Semantic Scholar

↓

[Summarization Node] -- (LLM or Fine-tuned Model) --> Generates summaries

↓

[Evaluation Module] -- (ROUGE & BERTScore) --> Computes performance metrics

↓

[Output / Display Results]

The LangGraph **directed edges** define this linear workflow:
START → generate_titles → get_papers → draft_answer → END

Benefits of the Architecture

- **Explainability:** Each stage's role is explicit, enabling easier debugging and understanding.
- **Scalability:** Components can run independently or in parallel for multiple topics.
- **Reusability:** Fine-tuned models and evaluation functions can be reused across research domains.
- **Efficiency:** LoRA fine-tuning significantly reduces training cost while maintaining quality.
- **Automation:** The entire research paper summarization process — from search to summary — is handled autonomously

---

Conclusion

The designed AI agent efficiently integrates data retrieval, summarization, and evaluation into one cohesive system. By combining **LangGraph orchestration**, **transformer-based models**, and **evaluation metrics**, the agent achieves both **functional automation** and **analytical transparency**.

This architecture serves as a strong foundation for scalable research-assistant applications in academia and data-driven literature analysis.
