University of Calgary

Department of Electrical and Software Engineering

Title: A Multi-Agent Research Assistant (Academic MAS)

Md Tayeb Adnan

Supervisor: Dr. Behrouz Far

A report submitted in partial fulfilment of the course requirements of
the University of Calgary for the course of Agent Based Software Engineering

October 9, 2025

Abstract

Research in academia and industry often involves extensive reading, literature review, data analysis, and report writing, which can be time-consuming and repetitive. To streamline this process, we propose a Multi-Agent Research Assistant System that leverages specialized software agents to automate key research tasks. The system allows users to either submit a text query or upload a PDF paper as input. A central Orchestrator Agent detects the input type and delegates tasks to dedicated agents, including PDF parsing, literature search, summarization, knowledge base management, analysis, review, citation generation, and report writing. By integrating these agents into a cohesive workflow, the system can extract and organize relevant information, generate summaries, run experiments, and produce a polished report with citations and figures. This approach not only reduces the manual workload of researchers but also ensures consistency, accuracy, and efficiency in generating research outputs. The proposed framework demonstrates the potential of multi-agent systems to enhance academic productivity and support evidence-based research.

Contents

Chapter 1

Introduction

1.1 Background

In recent years, the exponential growth of scientific publications and research data has made it increasingly challenging for researchers to stay up to date with the latest findings and extract relevant knowledge efficiently. Conducting a comprehensive literature review, analyzing existing methodologies, and writing a well-structured report demand a significant investment of time and effort. With the advancement of Artificial Intelligence (AI) and Large Language Models (LLMs), automation in research assistance has become a promising solution to accelerate knowledge discovery.

Traditional research workflows involve multiple stages—document reading, text extraction, summarization, analysis, and citation management—each of which requires specialized expertise and manual effort. However, these tasks can be intelligently distributed among autonomous agents, each designed to perform a specific function within a collaborative ecosystem. The multi-agent system (MAS) paradigm provides a powerful framework for such distributed intelligence, enabling independent yet coordinated agents to achieve a common research goal.

1.2 Problem Statement

Despite the availability of numerous research tools, the process of understanding, analyzing, and reporting scientific information remains largely manual and fragmented. Researchers often face challenges such as:

- Extracting meaningful information from complex PDF papers.
- Searching and filtering vast literature repositories efficiently.
- Integrating information from multiple sources into coherent summaries.
- Managing citations and references accurately.
- Maintaining consistency and factual correctness throughout a report.

Existing AI-based tools typically focus on isolated functions—like summarization or citation generation—but lack integration and inter-agent coordination. Consequently, there is a pressing need for a unified, intelligent system capable of automating the entire research workflow, from literature ingestion to report generation, while ensuring accuracy, traceability, and interpretability.

1.3 Aims and objectives

The primary aim of this work is to design and develop a Multi-Agent Research Assistant System that can automate and optimize key components of the research workflow. The system seeks to assist researchers in managing and synthesizing scientific knowledge more efficiently through intelligent collaboration among agents.

The main objectives are:

1. To develop an Orchestrator Agent capable of detecting the input type (PDF or query) and coordinating the workflow.
2. To implement a PDF Parser Agent for text extraction, section segmentation, and knowledge embedding.
3. To design a Literature Search Agent that retrieves related works from online repositories such as arXiv and PubMed.
4. To create a Knowledge Base Agent that stores embeddings and provides contextual information for other agents.
5. To integrate Summarizer, Analysis, Review, and Citation Agents to enhance report quality, factual accuracy, and proper referencing.
6. To produce a final structured report in markdown or PDF format using a Writer and Final Output Agent.

Together, these objectives aim to reduce human workload, improve accuracy in knowledge synthesis, and facilitate faster and more reliable research outputs.

1.4 Solution approach

To address the challenges identified, we propose a multi-agent system architecture that employs a modular and collaborative approach. Each agent in the system is designed with a specific responsibility, ensuring scalability, flexibility, and interoperability. The proposed approach is divided into several layers and phases:

1.4.1 Input Processing Layer

- The system accepts two types of inputs: a query prompt or a PDF document.
- The Orchestrator Agent acts as the central controller that identifies the input type and triggers the appropriate downstream agents (either the PDF Parser or the Literature Search Agent).

1.4.2 Knowledge Extraction and Representation Layer

- The PDF Parser Agent extracts text from the uploaded document, segments it into logical sections, and converts them into semantic embeddings stored in a vector database.
- Alternatively, the Literature Search Agent retrieves relevant works from academic repositories, and their metadata or text content is also embedded and stored within the Knowledge Base Agent.
- This layer creates a unified knowledge base that other agents can query to obtain contextual insights.

1.4.3 Information Processing and Analysis Layer

- The Summarizer Agent synthesizes key sections such as the introduction and methods, extracts terminologies, and identifies research gaps.
- The Analysis Agent performs additional tasks such as generating figures, running experimental analyses, or comparing methodologies from different studies.
- The Review / Critic Agent validates the generated text for coherence, factual consistency, and scientific accuracy.

1.4.4 Reporting and Output Layer

- The Citation Agent automatically generates BibTeX entries and formatted references using DOIs and metadata.
- The Writer Agent compiles the outputs from all agents—text summaries, figures, citations—into a cohesive document.
- Finally, the Final Output Agent merges all sections and exports the completed research report in the desired format (e.g., Markdown or PDF).

1.5 Organization of the report

The remainder of this report is organized as follows. Section 4 presents the overall system architecture of the proposed Multi-Agent Research Assistant, describing the interaction and coordination among agents through a structured workflow. Section 5 provides detailed descriptions of each agent, outlining their specific roles, responsibilities, and communication mechanisms. Section 6 explains the workflow of the system, illustrating how user inputs—either text queries or PDF documents—are processed step by step to produce the final output. Section 7 discusses the implementation details, including the technologies, frameworks, and models employed in developing the system. Section 8 introduces potential use cases that demonstrate how the proposed approach can automate literature review and academic writing tasks. Section 9 offers a discussion on the system's advantages, challenges, and future improvement directions. Finally, Section 10 concludes the report by summarizing the key contributions and emphasizing the system's potential to enhance research efficiency, followed by Section 11, which lists all references cited throughout the report.

Chapter 2

Methodology

2.1 Chosen methodology

Prometheus — a practical, well-documented agent methodology with clear steps for goal/role identification, architectural design, interaction protocols, and internal agent design. I'll present each Prometheus artifact adapted to your project.

2.2 System Specification — Goal identification

2.2.1 Primary goal

- G0: Automatically support and accelerate academic research workflows — from input (query or PDF) through search, summarization, analysis, and a citation-accurate draft report.

2.2.2 High-level functional goals (subgoals)

- G1: Ingest and parse user inputs (text prompts or uploaded PDFs).
- G2: Discover relevant literature (search arXiv/Semantic Scholar/CrossRef).
- G3: Extract and semantically index paper content (chunking + embeddings).
- G4: Summarize and synthesize knowledge into structured sections (intro, methods, results, gaps).
- G5: Execute light reproducibility/analysis tasks when requested (run code snippets / simulate experiments).
- G6: Generate a draft report with accurate citations (BibTeX).
- G7: Validate and critique generated content for factuality and logical coherence.
- G8: Persist knowledge for reuse (long-term memory).
- G9: Support human-in-the-loop corrections and iterative refinement.

2.3 Role Identification (Prometheus: Role Model)

The Prometheus methodology defines a *role* as a specification of a set of related tasks or responsibilities that an agent may perform within the system. Each role includes details about its goals, percepts, actions, and interactions with other agents. This section describes the major roles identified for the **Multi-Agent Research Assistant (Academic MAS)** system.

2.3.1 Overview of Roles

The system is composed of several collaborating roles, each representing a functional expertise area in the research process. The identified roles are:

1. **Manager / Orchestrator Role**
2. **PDF Parser Role**
3. **Literature Search Role**
4. **Knowledge Base Role**
5. **Summarizer Role**
6. **Analysis Role**
7. **Writer Role**
8. **Critic / Review Role**
9. **Citation Role**

2.3.2 Detailed Role Descriptions

Manager / Orchestrator Role

Goals:

- Manage user requests and initiate the workflow.
- Assign tasks to specialized agents based on input type (query or PDF).
- Ensure synchronization among all agents and handle task dependencies.

Percepts:

- User prompts or uploaded PDFs.
- Status reports from sub-agents.

Actions:

- Create task plans and delegate to agents.
- Merge results from agents and finalize output.

Interactions:

- Communicates with all agents to distribute and collect results.

PDF Parser Role

Goals:

- Extract textual and structural information from PDF documents.
- Segment document into sections (abstract, methods, results, etc.).

Percepts:

- Uploaded research PDF.

Actions:

- Parse, clean, and convert PDF text into structured format.
- Send extracted data to the Knowledge Base.

Literature Search Role

Goals:

- Retrieve relevant research papers and data from online sources.

Percepts:

- User query or topic keywords.

Actions:

- Query APIs (e.g., arXiv, Semantic Scholar).
- Send search results to Knowledge Base.

Knowledge Base Role

Goals:

- Store, organize, and provide semantic access to research data.

Percepts:

- Processed data from PDF Parser or Literature Search agents.

Actions:

- Embed and index documents.
- Support retrieval for RAG (Retrieval-Augmented Generation) tasks.

Summarizer Role

Goals:

- Generate concise summaries of research documents or topics.

Percepts:

- Retrieved content from Knowledge Base.

Actions:

- Use LLMs to summarize documents or research trends.

Analysis Role

Goals:

- Execute computational analysis or simulations as requested.
- Support result interpretation and figure generation.

Percepts:

- Processed data or analysis queries.

Actions:

- Run code, process outputs, and provide analysis results.

Writer Role

Goals:

- Draft structured report sections (Abstract, Introduction, etc.).

Percepts:

- Summaries and analysis outputs.

Actions:

- Compose coherent and well-formatted text sections.

Critic / Review Role

Goals:

- Evaluate generated content for consistency and accuracy.

Percepts:

- Drafted text from Writer Agent.

Actions:

- Provide feedback and suggest revisions.

Citation Role

Goals:

- Manage references and citation insertion.

Percepts:

- Completed research report.

Actions:

- Insert BibTeX-formatted citations.
- Validate reference consistency.

2.3.3 Role Interaction Summary

The defined roles interact collaboratively to achieve the overall goal of automated research assistance. The Manager Agent coordinates task execution, while domain-specific agents (e.g., Summarizer, Writer, Critic) handle their specialized subtasks. The Knowledge Base ensures persistent memory and facilitates iterative refinement.

2.4 Agent System Architecture (Architectural Design)

The architectural design phase in the Prometheus methodology defines the overall structure of the multi-agent system. It identifies the agents to be implemented, their responsibilities, and the communication pathways among them. For the **Multi-Agent Research Assistant (Academic MAS)**, the system architecture follows a modular, service-oriented, and collaborative structure where agents interact through message passing to complete the research workflow.

2.4.1 Overview

The **Academic MAS** architecture is organized around a central *Manager Agent* that coordinates specialized functional agents responsible for different stages of the research process. Each agent operates autonomously within its defined scope but collaborates through a shared communication protocol and accesses a common *Knowledge Base*. This architecture promotes scalability, modularity, and fault tolerance.

2.4.2 Architectural Components

1. **Manager (Orchestrator) Agent**
   Acts as the central controller that receives user queries or PDF uploads, decomposes tasks, and assigns them to appropriate agents. It also synchronizes the workflow and merges the final results into a complete research report.

2. **PDF Parser Agent**
   Responsible for extracting structured information from uploaded research papers (e.g., abstract, methods, results). Parsed content is sent to the Knowledge Base Agent for indexing.

3. **Literature Search Agent**
   Performs online retrieval of relevant papers or related work using academic APIs (e.g., arXiv, Semantic Scholar). It expands the contextual understanding of a research topic.

4. **Knowledge Base Agent**
   Manages long-term memory and stores document embeddings for retrieval-augmented generation (RAG). Provides contextual data to other agents upon request.

5. **Summarizer Agent**
   Summarizes long documents, identifies key findings, and synthesizes literature insights to provide concise research overviews.

6. **Analysis Agent**
   Executes computational experiments, data analysis, or simulations as required by the research process. Supports figure generation and result interpretation.

7. **Writer Agent**
   Generates well-structured textual content such as abstracts, introductions, methodology, and conclusions using the data provided by other agents.

8. **Critic / Review Agent**
   Evaluates generated content for coherence, correctness, and completeness. It also checks logical consistency and provides improvement suggestions to the Writer Agent.

9. **Citation Agent**
   Manages the insertion and formatting of references and citations, ensuring academic consistency and compliance with citation styles.

2.4.3 Communication Mechanism

Agents communicate using a *message-passing paradigm* where information is exchanged asynchronously. The communication follows a publish-subscribe model mediated by the Manager Agent. Each agent publishes task results and subscribes to messages relevant to its function. For example:

- The Manager Agent sends a "*summarize_document*" message to the Summarizer Agent.
- The Summarizer Agent replies with "*summary_ready*".
- The Writer Agent subscribes to "*summary_ready*" events to compose the initial draft.

2.4.4 System Topology

The system adopts a hybrid *hub-and-spoke* topology:

- The Manager Agent acts as the central hub.
- Specialized agents function as spokes performing domain-specific tasks.
- The Knowledge Base Agent serves as a shared memory accessible to all agents.

2.4.5 Architectural Advantages

- **Scalability:** New agents can be added (e.g., Data Visualization Agent) without disrupting existing functionality.
- **Modularity:** Each agent is autonomous and encapsulates its functionality.
- **Collaboration:** Agents cooperate dynamically to achieve shared system goals.
- **Reusability:** Agents can be reused across different academic or research workflows.

2.4.6 Architectural Diagram

The following diagram illustrates the high-level communication structure and relationships between agents in the Academic MAS system.

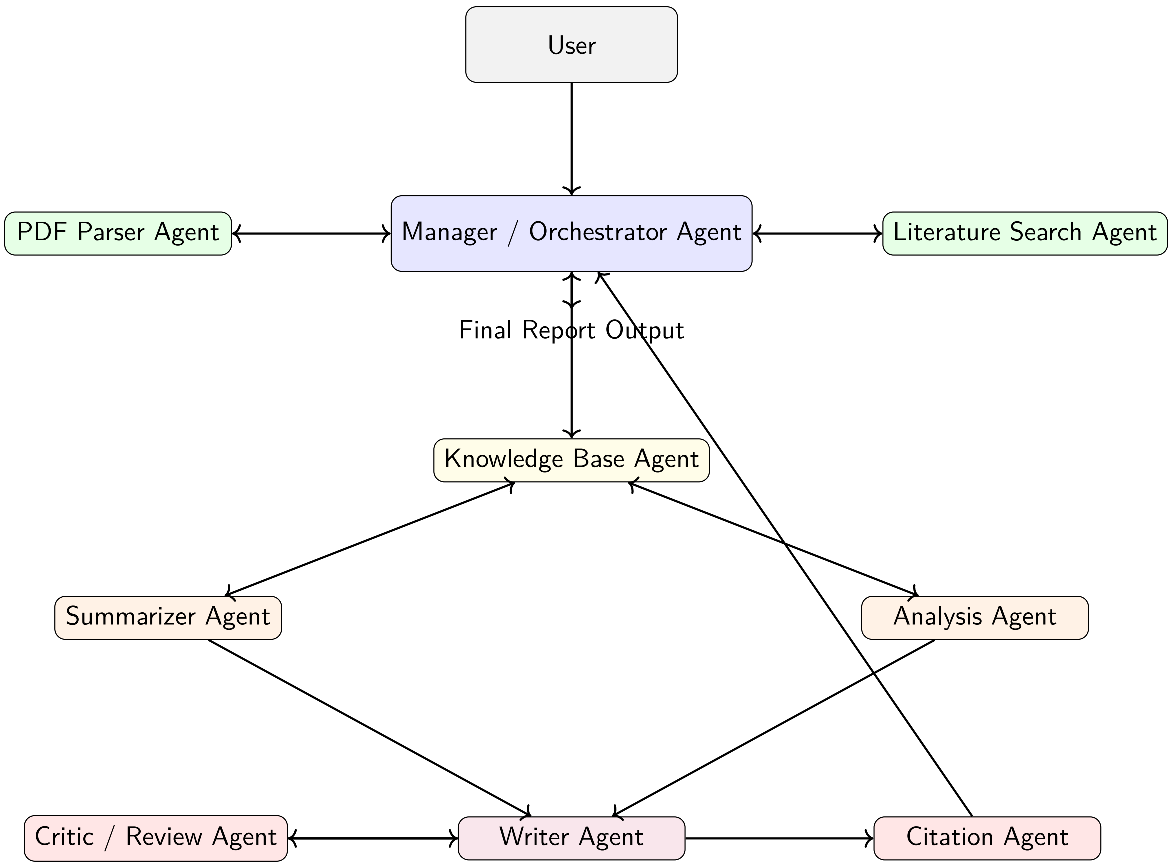

Figure 2.1: Agent System Architecture of the Academic MAS

2.5 Agent Descriptions (Prometheus: Agent/Capability Model)

For each agent, we describe their responsibilities, inputs/outputs, preconditions/postconditions, and interactions.

2.5.1 Manager / Orchestrator Agent

Purpose: Central planner and task dispatcher.
Responsibilities:

- Parse user input and create workflow plan.
- Dispatch tasks to appropriate agents.
- Collect and merge results.
- Handle errors and retries.
- Maintain session state.

Inputs: User prompt or uploaded file metadata.
Outputs: Final report artifact, status messages.
Interactions: Calls PDF Parser, Searcher, Retriever, Summarizer, Writer, Critic, Citation, and Memory Agents.
Failure Handling: If an agent fails, retries with exponential backoff or prompts human review.

2.5.2 PDF Parser Agent

Purpose: Extract text, metadata, figures, and sections from uploaded PDFs.
Core Tools: PyMuPDF, pdfplumber, unstructured.
Inputs: PDF file path.
Outputs: Document object:
`{title, authors, sections:[{heading, text, start, end}], figures[], doi?}`
Postcondition: Document chunks are stored (or returned) and queued to the Retriever Agent for embedding.

2.5.3 Literature Search Agent

Purpose: Discover related literature for a query.
APIs: arXiv, Semantic Scholar, CrossRef, SerpAPI.
Inputs: Query keywords and constraints (date range, domain).
Outputs: Ranked list of paper metadata and links.
Notes: Should de-duplicate overlapping results with existing knowledge base.

2.5.4 Retriever / Knowledge Base Agent

Purpose: Chunk documents, produce embeddings, and handle similarity search.
Inputs: Document objects from the PDF Parser or Search results.
Outputs: Embedding IDs and retrieval results for RAG queries.
Storage: Vector database with metadata linking to document offsets.
APIs: Embedding model calls and vector database SDK.

2.5.5 Summarizer Agent

Purpose: Create multi-level summaries (chunk $\rightarrow$ section $\rightarrow$ paper $\rightarrow$ topic).
Inputs: Retrieved chunks or documents.
Outputs: Structured summary objects:
`{paper_id, abstract, methods, dataset, metrics, limitations, claims[]}` plus provenance pointers (chunk IDs and offsets).
Strategy: Use hierarchical summarization and LLM prompts for extractive + abstractive blends.

2.5.6 Analysis / Execution Agent

Purpose: Reproduce small experiments or run supplied scripts reliably.
Capabilities:

- Create execution environment (Conda/Docker).
- Run Jupyter notebooks or scripts via nbclient.
- Report outputs, capture logs, and save artifacts.

Inputs: Code snippets or experiment recipes with data access info.
Outputs: Results (plots, metrics), container logs, success/failure flags.

2.5.7 Writer Agent

Purpose: Compose draft document sections using summaries and analysis outputs.
Inputs: Structured summaries, analysis outputs, and user style preferences.
Outputs: Markdown/LaTeX sections with in-text citation placeholders (e.g., [cite:paper_id]).
Features: Can request Critic Agent for iterative refinement.

2.5.8 Critic / Reviewer Agent

Purpose: Fact-check claims, verify citations mapping, and flag inconsistencies.
Inputs: Draft text and provenance pointers.
Outputs: List of issues (hallucination flags, unsupported claims) with priority levels.
Method: Runs RAG lookups to find supporting chunks and uses focused LLM prompts to validate claims.

2.5.9 Citation Agent

Purpose: Resolve DOIs, format references, and insert BibTeX and in-text citations.
Inputs: Metadata IDs, DOIs, and placeholder citations generated by Writer Agent.
Outputs: `references.bib`, resolved citation blocks, and clickable links.
APIs: CrossRef, Zotero, Semantic Scholar.

2.5.10 Memory Agent

Purpose: Persist session artifacts and long-term knowledge.
Inputs/Outputs: Store and retrieve summaries, user preferences, and embedding references.
Retention Policy: Configurable (e.g., per project).

2.6 Agent Internal Architecture (Detailed Design — Prometheus Internal View)

We follow a standard internal architecture template for each agent: **Perception → Deliberation/Reasoning (Planner) → Action → Memory/Knowledge → Communication**. Below, we provide a detailed internal architecture for the Manager / Orchestrator agent and variations for other agents.

2.6.1 Manager / Orchestrator — Internal Architecture

Components

Input Parser (Perception) Parses user prompt, detects file uploads, and extracts user constraints (e.g., deadline, depth).

Task Planner / Scheduler (Deliberation) Converts goals into tasks/subtasks with dependencies (DAG), prioritizes tasks, and sets timeouts.

Policy Engine (Decision) Applies rules to select agents (e.g., if file uploaded → PDF Parser first). Capable of adaptive decisions based on agent feedback.

Task Dispatcher (Action) Issues Task messages via message queue or direct API calls.

Coordinator Memory Session state store: {tasks[], results[], retries[], provenance[]}.

Result Merger Merges agent outputs into a unified report structure.

Error Manager Handles failures, retrials, and escalation to human-in-the-loop.

Communicator Standard message adaptor; marshals messages to/from agents.

Plan Library (Examples)

Plan: IngestPDF
1. Save file
2. PDFParser.parse
3. Retriever.ingestChunks
4. Summarizer.summarizePaper
5. Return paper_summary

Plan: LiteratureSurvey
1. Searcher.query
2. Retriever.fetchTopN
3. Summarizer.multiSummarize
4. Writer.seedDraft
5. Critic.review
6. Iterate until quality threshold

Data Structures

Task {id, type, inputs[], status, created_at, owner_agent, deps[]}

Provenance {paper_id, chunk_id, offset_start, offset_end}

2.6.2 PDF Parser — Internal Architecture

Document Loader Parse byte stream to raw text and images.

Sectionalizer Identify headings and boundaries (heuristics + NLP).

Figure Extractor Save images, OCR captions if needed.

Chunker Split into chunks of N tokens with overlap.

Metadata Extractor Detect DOI, title, authors (regex + CrossRef lookup).

Output Formatter & Uploader Write Document object and push to Retriever with metadata.

2.6.3 Retriever / KB — Internal Architecture

Embedding Adaptor Calls embedding model (batched).

Indexer Writes vectors to Vector DB with metadata.

Search API kNN search and hybrid scoring (embedding + BM25).

Cache & TTL Cache recent queries.

2.6.4 Summarizer — Internal Architecture

Chunk Summarizer Summarization prompts for each chunk.

Section Aggregator Combine chunk summaries into section summaries with provenance mapping.

Claim Extractor Extract claims/statements and link to chunks.

Confidence Scorer Estimate summary confidence (LLM self-assess + token overlap heuristics).

2.6.5 Analysis Agent — Internal Architecture

Job Runner Create isolated container, load environment, execute code.

Resource Manager Limit CPU/RAM, enforce timeouts.

Artifact Collector Pick up plots, logs, outputs.

Result Serializer Produce standard result object.

2.6.6 Writer & Critic — Internal Architecture

Writer Template engine, prompt templates per section, insertion points for citations (placeholders).

Critic Targeted RAG queries per claim, scoring heuristics, prioritized issue list.

References