Replacing Regex with Retrieval-Augmented Generation and Agentic AI for Tools Selection

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Engineering

in

Computer Science and Engineering

Submitted by

Nayan Mahata

Regn. No. - 1661828 of 2023 - 2024
Exam Roll No. - M4CSE25xxx
Class Roll No. - 002310502034
Session - 2023 - 2025

Under the supervision of

Prof. Debotosh Bhattacharjee

Department of Computer Science and Engineering
Jadavpur University

188, Raja S.C. Mallick Rd,
Kolkata - 700032, West Bengal, India

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University, Kolkata - 700032

CERTIFICATE OF RECOMMENDATION

This is to certify that the work embodied in this thesis entitled *"Replacing Regex with Retrieval-Augmented Generation and Agentic AI for Tool Selection"* has been satisfactorily completed by **Nayan Mahata** (Registration Number 1661828 of 2023 - 2024; Class Roll No. 002310502034; Examination Roll No. M4CSE250xx). It is a bonafide piece of work carried out under my supervision and guidance at Jadavpur University, Kolkata, for partial fulfilment of the requirements for the awarding of the **Master of Engineering in Computer Science and Engineering** degree of the Department of Computer Science and Engineering, Faculty of Engineering and Technology, Jadavpur University, during the academic year 2023 - 2025.

Prof. Debotosh Bhattacharjee
Department of Computer Science and Engineering
Jadavpur University
(Supervisor)

Forwarded By:

Prof. Nirmalya Chowdhury
HOD, Department of Computer Science and Engineering
Jadavpur University

Prof. Saroj Mandal
DEAN, Faculty of Engineering and Technology
Jadavpur University

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University, Kolkata - 700032

CERTIFICATE OF APPROVAL

This is to certify that the thesis entitled *"Replacing Regex with Retrieval-Augmented Generation and Agentic AI for Tools Selection"* is a bonafide record of work carried out by **Nayan Mahata** (Registration Number 1661828 of 2023 - 2024; Class Roll No. 002310502034; Examination Roll No. M4CSE250xx) in partial fulfilment of the requirements for the award of the degree of **Master of Engineering in Computer Science and Engineering** in the Department of Computer Science and Engineering, Jadavpur University, during the period of June 2023 to May 2025. It is understood that by this approval, the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the thesis only for the purpose of which it has been submitted.

Signature of Examiner 1
Date:

Signature of Examiner 2
Date:

Department of Computer Science and Engineering
Faculty of Engineering and Technology
Jadavpur University, Kolkata - 700032

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that the thesis entitled *"Replacing Regex with Retrieval-Augmented Generation and Agentic AI for Tools Selection"* contains literature survey and original research work by the undersigned candidate, as a part of his degree of **Master of Engineering in Computer Science and Engineering**, Jadavpur University. All the information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Nayan Mahata
Examination Roll No.: M4CSE250xx
Registration No.: 1661828 of 2023 - 2024
Thesis Title: Replacing Regex with Retrieval-Augmented Generation and Agentic AI for Tools Selection

Signature of the Candidate: _____

Acknowledgements

The success and outcome of this thesis required a lot of guidance and assistance from many people, and I am extremely lucky to come across them during the thesis. I would like to take this opportunity to thank them for their guidance and support in everything I have accomplished. I owe my deep gratitude to my supervisor and thesis guide **Prof. Debotosh Bhattacharjee**, who was always there during my whole thesis work and guided me all along till the completion of my thesis work by providing all the necessary information that would be beneficial for my research and experiments. I am extremely thankful to him for his assistance and dedicated involvement in every step of the process. His constant suggestions have been an everlasting source of motivation and inspiration to me. It has been an honour to work under a person with wisdom yet a kind, simplistic attitude.

I would also like to thank **Prof. Nirmalya Chowdhury**, Head of the department, Department of Computer Science and Engineering at Jadavpur University, for extending her help to complete this thesis.

I had the pleasure of working in the CMATER lab of Jadavpur University. I acknowledge the generous collaboration and assistance of every member of the CMATER lab. It has been a wonderful experience working with every member of this lab. I learnt many things regarding research works and was inspired to do better in my thesis work from them.

I especially like to thank my classmate and one of the members of CMATER lab, **Akash Nayak**, who helped me in many steps of my thesis work.

At the end, I like to acknowledge both of my parents for their constant support during this thesis work.

Nayan Mahata
Examination Roll No.: M4CSE250xx
Registration No.: 1661828 of 2023 - 2024
Department of Computer Science and Engineering
Jadavpur University

Abstract

Large language models (LLMs) have revolutionized the landscape of natural language processing by enabling advanced reasoning, generation, and task orchestration. While regular expressions (RegEx) remain a staple for pattern-based information extraction, they suffer from limitations in flexibility, contextual understanding, and adaptability. RegEx patterns are rigid, often break with minor format changes, and require extensive domain-specific knowledge to write and maintain—making them unsuitable for dynamic or ambiguous data environments. In contrast, Retrieval-Augmented Generation (RAG) introduces semantic reasoning and contextual adaptation by leveraging relevant knowledge from large corpora. This thesis proposes a novel framework that replaces traditional RegEx with a system built on RAG and Agentic AI. Here, autonomous agents act as orchestrators: selecting appropriate tools, retrieving relevant data, and applying tailored logic based on user intent. The system takes natural language queries, interprets them using retrieved knowledge, and dynamically composes extraction logic that is more robust and adaptable than handcrafted expressions. Our experiments demonstrate that this approach not only generalizes better than static RegEx rules but also adapts to diverse data sources and evolving information needs. This work introduces a paradigm shift in information extraction, opening pathways for intelligent, context-aware automation.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Regular Expressions (RegEx) have long served as a foundational tool in computer science for pattern recognition, string parsing, and information extraction. Their deterministic structure and mathematical precision make them effective for processing well-structured and static data patterns [**?**]. However, in dynamic and semantically rich environments—such as natural language processing (NLP), log analysis, and adaptive data mining—RegEx increasingly demonstrates critical limitations [**?**, **?**]. These patterns tend to be brittle, require frequent manual updates, and often fail to generalize across diverse or evolving input structures [**?**, **?**]. Furthermore, the development and debugging of complex RegEx patterns typically demand substantial domain expertise, rendering them less accessible to non-technical users.

Recent advancements in Artificial Intelligence, particularly in the development of large language models (LLMs), have introduced capabilities that extend beyond the constraints of traditional symbolic methods. Retrieval-Augmented Generation (RAG) [**?**] exemplifies this progress by integrating neural generative models with retrieval mechanisms to enhance factual accuracy and contextual coherence. RAG enables systems to access and synthesize information from large external corpora during inference, combining this retrieved knowledge with generative reasoning to support sophisticated language understanding, multi-step reasoning, and complex task execution.

Alongside RAG, the emergence of *Agentic AI* introduces a new layer of autonomy and decision-making into artificial intelligence systems. Agentic architectures are characterized by their ability to perform planning, select tools dynamically, utilize memory, and engage in iterative problem-solving [**?**, **?**, **?**]. These systems embody goal-directed behavior, rendering them particularly effective in tasks that demand adaptability, subtask coordination, and continuous interaction with evolving inputs. When applied to the domain of information extraction, agentic systems can autonomously select appropriate tools (e.g., search, classification, summarization, transformation) and orchestrate the

application of RAG to extract patterns and execute actions in contextually aware and dynamically adaptive ways [**?**].

Recent breakthroughs in agent-based reasoning systems have catalyzed the development of highly autonomous frameworks capable of managing complex workflows with minimal human intervention. Large Language Models (LLMs), including OpenAI's GPT-4, Qwen2.5-Omni, DeepSeek-R1, and Meta's LLaMA, have significantly advanced artificial intelligence by enabling sophisticated natural language understanding, contextual reasoning, and human-like interaction [**?**, **?**, **?**, **?**]. These capabilities have been extended into multimodal domains, facilitating tasks such as text-to-image generation, video synthesis, and cross-lingual translation. However, the static nature of pre-training poses limitations, as it can result in outdated or inaccurate outputs. This issue is mitigated by the Retrieval-Augmented Generation (RAG) paradigm [**?**], which augments generative models by retrieving up-to-date information from external sources such as APIs, databases, or the web during inference.

Building on the foundation established by Retrieval-Augmented Generation (RAG), agentic systems extend capabilities further by leveraging modular tools and orchestration protocols to exhibit reflective and adaptive behavior [**?**, **?**]. These systems are capable of generating hypotheses, planning execution sequences, interfacing with external tools, and collaborating with other agents in multi-agent environments. Such architectures have been successfully deployed across a range of domains, including research automation, clinical decision support, biomedical literature review, and scientific experiment design [**?**, **?**]. Notable frameworks such as *LitSearch*, *AgentTuning*, and *ResearchArena* illustrate that LLM-empowered agents can match or even surpass human performance in specialized cognitive tasks. Nevertheless, these advancements underscore persistent challenges related to reproducibility, reliability, and safety, particularly when deployed in high-stakes or rapidly changing contexts.

Current agentic methodologies increasingly incorporate advanced strategies such as Monte Carlo Tree Search (MCTS) [**?**, **?**, **?**], Learn-by-Interact paradigms, and reinforcement learning techniques to enhance decision-making efficiency and optimize tool utilization [**?**, **?**]. Furthermore, multi-agent architectures introduce a division of cognitive labor, enabling domain-specialized agents to collaboratively address complex tasks that surpass the capabilities of any single language model. These systems emulate the collaborative structure of human teams by integrating complementary skill sets and facilitating structured knowledge exchange [**?**, **?**]. Such collaborative intelligence extends the operational scope of agentic frameworks, particularly in domains that require distributed reasoning, parallel execution, and consensus-building.

By embedding reasoning, retrieval, action, and adaptation into a unified architecture,

Agentic AI represents more than a mere enhancement over traditional large language models (LLMs); it constitutes a foundational shift in how intelligent systems perceive, plan, and act [**?**, **?**]. As explored in this thesis, integrating these capabilities into the traditionally rigid domain of pattern matching enables the development of a new class of interpretable, dynamic, and context-aware extractive frameworks.

Historically, Regular Expressions (RegEx) have served as a cornerstone in pattern recognition, string parsing, and information extraction. Their deterministic nature and mathematical precision make them well-suited for static, well-structured data. However, in dynamic and semantically complex environments—such as natural language processing (NLP), system log analysis, and adaptive data mining—RegEx becomes increasingly inadequate. These symbolic patterns tend to be brittle, necessitate frequent manual revision, and often fail to generalize across diverse or noisy inputs. Furthermore, the construction and debugging of sophisticated RegEx patterns require considerable domain expertise, limiting accessibility for non-technical users and constraining their applicability in real-world scenarios.

Recent advancements in Artificial Intelligence, particularly in the domain of large language models (LLMs), have introduced capabilities that extend well beyond traditional symbolic approaches [**?**]. Among these innovations, Retrieval-Augmented Generation (RAG) stands out as a paradigm that integrates neural generative models with retrieval mechanisms to enhance both factual accuracy and contextual relevance. By dynamically accessing large-scale external corpora at inference time, RAG enables systems to synthesize retrieved information with generative reasoning, resulting in more informed, coherent, and context-sensitive outputs. This architecture significantly augments the natural language understanding and execution capabilities of LLMs, making them more suitable for complex, real-world applications.

In parallel with the development of Retrieval-Augmented Generation (RAG), the emergence of Agentic AI has introduced autonomy and high-level decision-making into artificial intelligence systems [**?**, **?**]. Agentic systems are characterized by their ability to plan, select tools, utilize memory, and iteratively solve problems. These systems exhibit goal-directed behavior, enabling them to adapt to dynamic environments, decompose complex tasks into manageable subtasks, and interact continuously with evolving inputs.

When applied to the domain of information extraction, agentic architectures offer a flexible and context-sensitive alternative to rigid rule-based approaches. Such agents can autonomously select and orchestrate a suite of tools—including search engines, classifiers, summarizers, and data transformation utilities—based on the task at hand. By integrating these capabilities with RAG, agentic systems are able to perform pattern extraction and downstream tasks in a contextually informed and adaptive manner, thereby

significantly enhancing their utility in real-world applications.

1.2 Motivation

The central motivation of this thesis is to address the limitations inherent in static, rule-based pattern matching systems and to propose a robust, adaptive alternative grounded in contemporary AI paradigms. Traditional Regular Expressions (RegEx), while effective in processing well-structured data, exhibit fundamental shortcomings in interpreting user intent, adapting to structural variability, and handling noisy or ambiguous input. In practical scenarios—such as parsing unstructured log files, extracting data from heterogeneous HTML sources, or interpreting natural language commands—RegEx-based systems frequently fail unless meticulously engineered for each specific context.

By contrast, Retrieval-Augmented Generation (RAG) introduces semantic flexibility by leveraging contextually relevant external information during inference [**?**], enabling pattern interpretation that aligns with real-world variability. Complementing this, Agentic AI empowers systems with autonomous reasoning capabilities, including intelligent tool selection, dynamic task planning, and iterative problem-solving [**?**, **?**]. The convergence of RAG and Agentic AI thus offers a compelling pathway toward building intelligent, learning-based systems that not only replicate but substantially surpass the functionality of RegEx in terms of adaptability, robustness, and usability.

This research is driven by the potential to address key limitations of traditional pattern matching systems through the following objectives:

- **Democratization of pattern matching:** Enable non-technical users to define and apply extraction rules using natural language inputs, thereby lowering the barrier to entry for advanced text processing.
- **Automated adaptability:** Facilitate dynamic adaptation of extractive logic in response to evolving data formats, reducing reliance on manual rule updates.
- **Improved interpretability:** Enhance the transparency and explainability of the pattern generation process, promoting user trust and model accountability.
- **Reduced maintenance overhead:** Minimize the need for continuous manual intervention in systems reliant on complex and brittle text processing pipelines.

1.3 Research Objectives

This thesis proposes the design, implementation, and evaluation of a novel framework that replaces traditional Regular Expressions (RegEx) in pattern matching tasks with a

hybrid architecture that integrates Retrieval-Augmented Generation (RAG) and Agentic AI. The research is guided by the following specific objectives:

1. To conceptualize a system capable of interpreting natural language instructions and translating them into executable logic for structured and unstructured data extraction.

2. To develop an agent-based architecture that autonomously determines the optimal sequence and application of tools (e.g., information retrieval, summarization, syntactic parsing) based on the task context.

3. To empirically evaluate the performance of the proposed framework in comparison with traditional RegEx-based approaches across a range of real-world text processing scenarios.

4. To assess the system's adaptability, generalization capability, and robustness in handling semi-structured and noisy input data.

1.4 Research Questions

To guide the investigation and evaluation of the proposed framework, this thesis is structured around the following research questions:

1. Can a RAG and Agentic AI-based system outperform traditional Regular Expressions (RegEx) in terms of adaptability and accuracy for complex pattern extraction tasks?

2. How does agent autonomy—specifically in tool selection, planning, and execution—contribute to the overall effectiveness of text extraction?

3. What are the limitations of such a system in terms of computational resource demands, interpretability, and error handling?

4. To what extent can the proposed framework generalize across domains such as, legal documentation, and security data extraction?

This chapter has established the foundational context for the remainder of the thesis, detailing both the technical challenges and conceptual motivations underlying the proposed research. By transitioning from rigid, rule-based pattern-matching methods to intelligent, dynamic systems, this work aims to advance the frontier of AI-assisted information extraction.

Chapter 2

Literature Review

Over the past decade, the field of pattern recognition and intelligent information extraction systems has experienced substantial advancements, particularly through the integration of traditional symbolic methodologies with contemporary artificial intelligence (AI) paradigms. Regular Expressions (RegEx), long-standing tools for syntactic string pattern matching, continue to be instrumental in various text processing applications. However, their deterministic nature and structural rigidity present significant challenges, especially when applied to noisy, semi-structured, or context-sensitive data, such as that encountered in natural language processing (NLP). These limitations have spurred interest in alternative approaches that incorporate semantic understanding, adaptive reasoning, and contextual awareness.

The advent of large language models (LLMs) has profoundly transformed the landscape of computational linguistics. Seminal studies, including those by Locascio et al. [**?**] and Li et al. [**?**], have demonstrated the potential of neural architectures to derive RegEx-like patterns from natural language inputs or example data. These pioneering efforts have laid the groundwork for hybrid systems that integrate symbolic reasoning with neural computation. Within this evolving context, Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [**?**], represents a significant paradigm shift. By combining a neural text generator with an external retrieval mechanism, RAG facilitates the production of factually accurate responses and supports reasoning capabilities grounded in dynamic, real-world information sources.

Recent work by Ferrag et al. [**?**] critically examines the capabilities of Retrieval-Augmented Generation (RAG) in enabling multi-hop reasoning and orchestrating complex workflows. Their study highlights the consistent superiority of retrieval-augmented large language models over closed-book counterparts, including advanced models such as GPT-4. Using a combination of empirical benchmarks and taxonomy-driven evaluations, the authors demonstrate that integrating RAG with iterative reasoning loops significantly enhances both the accuracy and interpretability of responses across a wide range of tasks.

Notably, this body of research extends into the domain of agent-based architectures—commonly referred to as Agentic AI—which autonomously manage complex reasoning processes, tool selection, and task orchestration. These systems are characterized by goal-directed behavior, self-reflection, and iterative error correction, often coordinating actions across multiple external tools and heterogeneous knowledge sources. Prominent agentic frameworks such as AutoGPT, Reflexion, and ReAct exemplify this paradigm, achieving varying degrees of autonomy through modular planning components, persistent memory, and dynamic behavioral adaptation [?, ?, ?].

Ferrag et al. [?] further propose a comprehensive taxonomy encompassing over 60 benchmarks introduced between 2019 and 2025, aimed at systematically evaluating the performance of large language model (LLM)-based agents across a diverse set of domains. These domains include mathematical reasoning, scientific literature review, code synthesis, factual retrieval, and embodied multimodal interaction. Their work highlights the emergence of specialized frameworks—such as *LitSearch*, *AgentTuning*, and *ResearchArena*—that integrate LLMs with domain-specific toolkits tailored for academic, biomedical, and scientific research workflows.

Moreover, the study explores the role of multi-agent environments, wherein ensembles of domain-specialized agents collaborate to address complex, interdependent tasks. These architectures are inspired by the dynamics of human teams, incorporating mechanisms such as distributed memory, joint planning, and consensus-driven dialogue protocols. Such systems have been applied across a variety of domains, including multi-robot coordination, collaborative software engineering, policy simulation, and large-scale societal modeling [?, ?, ?].

Despite these notable advancements, several challenges persist. Key limitations include the reproducibility of agentic behavior, latency arising from complex tool orchestration, and the lack of universally standardized evaluation protocols, all of which constrain the scalable deployment of agent-based systems. Ferrag et al. [?] emphasize the need for modular, task-agnostic benchmarks capable of evaluating real-time reasoning capabilities, coordination of multi-tool workflows, and memory-informed planning strategies. Furthermore, the study draws attention to pressing ethical considerations—particularly those related to interpretability, fairness, and transparency—which necessitate the establishment of robust governance frameworks and alignment with evolving regulatory standards.

In conclusion, the existing body of literature offers compelling empirical and theoretical support for progressing beyond static Regular Expression (RegEx) systems. This thesis builds upon these insights by introducing a novel, context-aware framework for pattern extraction that leverages Retrieval-Augmented Generation (RAG) and Agentic

AI. The proposed approach aligns with the broader trajectory of intelligent, adaptive systems in computational linguistics and information engineering.

The intersection of traditional pattern recognition techniques—such as RegEx—and advanced AI architectures represents a pivotal shift in the field of information extraction. Although RegEx has played a foundational role, it remains constrained by its syntactic rigidity and lack of semantic comprehension. A growing body of research has highlighted its limitations in processing complex, noisy, or structurally inconsistent inputs, thereby motivating the transition toward more flexible, learning-based methodologies. By incorporating semantic reasoning, dynamic tool use, and memory-based planning, the integration of RAG and agentic frameworks promises enhanced generalization, interpretability, and task adaptability across diverse application domains.

Early advancements in natural language processing (NLP) employed conventional machine learning models primarily for pattern classification tasks. However, it was not until the emergence of large language models (LLMs) that semantic parsing at scale became practically feasible. Foundational studies by Locascio et al. [**?**] and Li et al. [**?**] introduced neural methodologies for inferring regular expressions from natural language descriptions and examples, effectively bridging the gap between symbolic and subsymbolic approaches. These contributions established a foundational link between neural representation learning and symbolic rule induction, paving the way for architectures that integrate retrieval and reasoning mechanisms—such as Retrieval-Augmented Generation (RAG)—and ultimately enabling the design of agent-based systems capable of context-sensitive pattern generation.

Building upon these developments, the study by Ferrag et al. [**?**] titled *Can Retrieval-Augmented Language Models Reason?* investigates the role of retrieval mechanisms in facilitating multi-hop reasoning for complex tasks. The authors demonstrate that even high-performing closed-book models, such as GPT-4, experience notable performance gains when augmented with classical retrieval methods like BM25. This is particularly evident in scenarios that require reasoning over factual or multi-step content. The research introduces a suite of novel benchmarks designed to evaluate this interaction, revealing that the integration of retrieval-augmented LLMs with structured reasoning chains consistently outperforms either approach in isolation. These findings reinforce the conceptual shift toward viewing retrieval not merely as a knowledge enhancer, but as a fundamental catalyst for effective reasoning in language models.

Moreover, the study contributes a comprehensive taxonomy of over 60 benchmarks developed between 2019 and 2025, aimed at evaluating the effectiveness of large language models and autonomous agents across a diverse array of domains, including mathematical reasoning, domain-specific question answering, code generation, and multimodal problem-

solving. These benchmarking frameworks extend beyond traditional language model assessment; they provide critical insights for the design and evaluation of agentic systems, particularly in contexts where tool orchestration, decision-making, and adaptive planning are central to task execution [**?**].

Incorporating these insights, this thesis aligns with contemporary trajectories in large language model (LLM) research by proposing a system that unifies retrieval, reasoning, and autonomous agent planning. The resulting framework not only addresses the limitations of traditional Regular Expression (RegEx)-based approaches but also extends their applicability to complex, real-world data extraction tasks. By integrating semantic understanding, dynamic tool use, and contextual adaptation, the system represents a significant step toward more robust and intelligent information extraction pipelines.

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. [**?**], integrates pretrained transformer models with a retrieval module, enabling dynamic access to external documents at inference time. This architectural innovation markedly improves factual grounding and generalization capabilities across a range of natural language processing tasks. Expanding upon this foundation, Ferrag et al. [**?**] examine the potential of RAG to support multi-hop reasoning in complex problem domains. Their findings indicate that even state-of-the-art closed-book LLMs—such as GPT-4 and Gemini—exhibit significant performance enhancements when augmented with classical retrieval mechanisms like BM25. Through extensive benchmarking of retrieval-augmented and agentic configurations, the study reveals that the performance gains are particularly pronounced in tasks involving complex, multi-step reasoning.

The paper introduces a taxonomy of over 60 benchmarks spanning diverse domains such as academic reasoning, mathematical problem-solving, factual grounding, and multimodal task execution [**?**]. It further investigates agentic variants that integrate Retrieval-Augmented Generation (RAG) with task-oriented planning, reflective reasoning loops, and modular tool utilization. These systems are designed not only to retrieve contextually relevant knowledge but also to engage in iterative reasoning, develop multi-step solution plans, and refine their behavior through self-corrective feedback mechanisms. This agentic approach demonstrates the potential for LLM-based systems to perform complex, goal-directed tasks with increasing levels of autonomy and robustness.

This study underscores that the synergy between retrieval and generation—when orchestrated through autonomous agent architectures—constitutes a robust pathway toward the development of highly capable, intelligent systems. It further highlights the necessity of designing benchmarks and evaluation frameworks that specifically assess real-world applicability, adaptability, and reasoning efficiency in such agentic configurations. These insights directly reinforce the core objective of this thesis: to move beyond

static Regular Expression (RegEx)-based methods by introducing a dynamic, Retrieval-Augmented Generation (RAG) and Agentic AI pipeline capable of real-time semantic interpretation and information extraction.

As research in Retrieval-Augmented Generation (RAG) advanced, its limitations in addressing multi-hop and dynamic reasoning tasks became increasingly apparent. This led to the emergence of Agentic AI—a paradigm centered on autonomy, iterative planning, and sophisticated tool integration. Prominent agentic systems such as ReAct [**?**], Reflexion [**?**], and AutoGPT [**?**] exemplify cognitive characteristics including goal-directed behavior, reflective reasoning, and memory utilization. Recent contributions by Wu et al. [**?**] and Ferrag et al. [**?**] introduced frameworks in which agents function as orchestrators, dynamically selecting among various tools—such as retrievers, executors, and evaluators—while refining outputs through self-reflective feedback. Ferrag et al. [**?**] further provide a comprehensive review of such systems, emphasizing that modern agentic architectures incorporate dynamic planning, memory-based state tracking, and modular toolkits for complex task decomposition.

The authors investigate how autonomous agents interact with external environments through modular interfaces, evaluating the reliability and adaptability of these interactions across a range of tasks, including text-based research, scientific inquiry, and software engineering. The study presents a suite of prominent AI-agent frameworks developed between 2023 and 2025—such as *LitSearch*, *ResearchArena*, and *AgentTuning*—which integrate large language models (LLMs) with autonomous control modules. These systems are designed to manage complex research workflows, generate citations, and extract domain-specific information. A central finding is that such agentic frameworks demonstrate strong performance in dynamic environments where capabilities like tool selection, feedback integration, and error recovery are essential for task success [**?**, **?**].

The study also identifies several critical challenges associated with the development and deployment of autonomous agents, including the need to ensure reproducibility, reduce hallucination rates, and establish robust ethical standards. Notably, it introduces a taxonomy comprising 60 benchmarks that span a wide array of domains, such as general knowledge reasoning, mathematical problem-solving, code synthesis, multimodal analysis, and embodied AI [**?**]. This taxonomy underscores the importance of evaluating agentic reasoning across diverse operational contexts. These findings directly inform the trajectory of this thesis and further support the case for agentic orchestration as a viable replacement for rigid, rule-based systems such as Regular Expressions (RegEx), offering instead dynamic, context-sensitive AI solutions.

In parallel, agentic architectures have been applied to a range of domain-specific challenges. In the context of time series analysis, Ravuru et al. [**?**] demonstrated that

autonomous agents, when integrated with domain-specific models, significantly outperformed traditional static approaches by dynamically composing and adapting analytical workflows. In the domain of academic research, systems such as *LitSearch* and *ResearchArena* have been developed to automate cognitively intensive tasks, including literature review, information synthesis, and hypothesis testing [**?**]. These applications illustrate the potential of agentic reasoning to scale complex cognitive workflows while reducing reliance on direct human supervision. Furthermore, Ferrag et al. [**?**] highlight the utility of agentic frameworks in biomedical settings, where LLM-driven agents assist in diagnosis, patient communication, and clinical education by leveraging structured medical knowledge bases and formal clinical guidelines. Despite these advancements, significant challenges remain—particularly with respect to ethical oversight, interpretability, and domain-specific alignment of agent behavior.

The paper further highlights that large language model (LLM)-based agents are increasingly enhanced through methodologies such as *Learn-by-Interact*, *AgentGen*, and *AgentTuning* [**?**], which enable agents to generate and refine data through real-time interaction or synthetic feedback loops. Reinforcement learning techniques are often incorporated into these pipelines, allowing for more efficient training via iterative refinement and human-in-the-loop feedback. As a result, these agents are not only proficient in technical domains—such as citation generation and symbolic reasoning—but also demonstrate competence in open-ended tasks, including policy simulation, collaborative writing, and software development. These advancements underscore the centrality of tool orchestration, modular design, and iterative learning as foundational principles for constructing scalable and adaptive agentic systems.

Moreover, multi-agent environments—where domain-specialized agents collaboratively address complex tasks—have emerged as a promising strategy for scaling artificial intelligence. Drawing inspiration from human teamwork, these architectures emulate collaborative planning, role assignment, and inter-agent communication protocols. As noted by Ferrag et al. [**?**], large language model (LLM)-based multi-agent systems substantially enhance the capabilities of individual agents by enabling the division of cognitive labor and supporting collective reasoning. These agents engage in coordinated planning, structured dialogue, and consensus-building activities, effectively mirroring the cooperative dynamics of human teams and offering a scalable model for tackling interdependent subtasks in dynamic environments.

Multi-agent systems have been successfully applied across a variety of domains. In software development, individual agents specialize in coding, testing, and documentation, thereby facilitating modular and efficient workflows. In multi-robot control, these systems enable coordinated and distributed task execution, which is essential for managing complex operations. Additionally, agent-based models are utilized in social and policy

simulations to capture intricate societal interactions, enabling the analysis of negotiation processes, strategic decision-making, and collective behavior. These applications not only demonstrate the technical feasibility of agent collaboration but also underscore their potential in addressing problems that require negotiation, strategic planning, and shared memory management [?, ?, ?].

Recent frameworks have incorporated advanced training protocols that enable agents to acquire cooperative behaviors and adapt to dynamic environments. Techniques such as reinforcement learning-based fine-tuning [?], trajectory optimization [?], and meta-learning [?] have been employed to enhance multi-agent coordination and flexibility. Through these mechanisms, multi-agent systems increasingly demonstrate the capacity to address open-ended, ambiguous, and interdependent tasks across a wide range of real-world applications.

Despite the promise of Agentic AI, several challenges remain. Most notably, the reproducibility of agent behaviors, high computational costs, and the necessity for fine-grained evaluation metrics continue to be underexplored. Ferrag et al. [?] emphasize the lack of standardized tools to systematically track agent decision-making processes, interactions with external systems, and mechanisms for failure recovery. Furthermore, although reinforcement learning and memory-augmented frameworks have contributed to performance improvements, many agentic systems remain vulnerable to adversarial inputs and exhibit brittleness when confronted with novel or out-of-distribution domains.

Another pressing challenge pertains to the integration of ethical governance, interpretability, and fairness within agent decision-making processes. The absence of transparent models and explainable reasoning mechanisms undermines trust in automated system outputs, particularly in high-stakes domains such as healthcare and legal analysis [?, ?, ?]. Additionally, real-time performance continues to present a significant bottleneck, as many agentic systems experience considerable latency stemming from complex planning and retrieval operations [?, ?].

To address these challenges, ongoing efforts focus on developing modular benchmarks that evaluate distinct agentic capabilities, including real-time planning, collaborative tool use, and long-term memory retention. Proposed taxonomies categorize evaluations into areas such as factual reasoning, symbolic manipulation, multimodal inference, and embodied task execution [?, ?, ?, ?, ?]. These initiatives aim to establish standardized agent benchmarking frameworks, enabling researchers to systematically track progress and diagnose limitations in a reproducible manner.

Collectively, the literature reflects a clear trajectory: from rigid, rule-based systems toward adaptable, intelligent agents capable of interpreting context, planning actions,

and autonomously executing tasks [**?**, **?**, **?**]. This thesis builds upon this trajectory by focusing on the novel integration of Retrieval-Augmented Generation (RAG) and Agentic AI to supplant traditional Regular Expressions (RegEx) in real-world pattern extraction scenarios.

Chapter 3

Methodology

This section outlines the system architecture, tools, datasets, and evaluation protocols employed to design and assess the proposed framework that substitutes traditional Regular Expressions (RegEx) with a hybrid solution leveraging Retrieval-Augmented Generation (RAG) and Agentic AI. The methodology is structured around four core components:

3.1 Core System Architecture

The core system combines Retrieval-Augmented Generation (RAG) [**?**, **?**] with a modular agentic planner [**?**] to transform natural language instructions into actionable information extraction routines. The RAG subsystem handles knowledge grounding through a dual-phase retrieval and generation process, leveraging mechanisms explored in frameworks like Ext2Gen [**?**] and HybridRAG [**?**] to improve alignment and accuracy. Meanwhile, the agentic controller selects and sequences tools based on the semantic intent of the input, embodying goal-driven behavior, memory usage, and reflective reasoning capabilities as described in recent advances in Agentic AI [**?**] (see Figure 3.1).

3.2 Core Components of RAG

The architecture of Retrieval-Augmented Generation (RAG) systems integrates three primary components (see Figure 3.2):

- **Retrieval:** Responsible for querying external data sources such as knowledge bases, APIs, or vector databases. Advanced retrievers leverage dense vector search and transformer-based models to improve retrieval precision and semantic relevance [**?**].
- **Augmentation:** Processes retrieved data, extracting and summarizing the most relevant information to align with the query context.
- **Generation:** Combines retrieved information with the LLM's pre-trained knowledge to generate coherent, contextually appropriate responses [**?**].

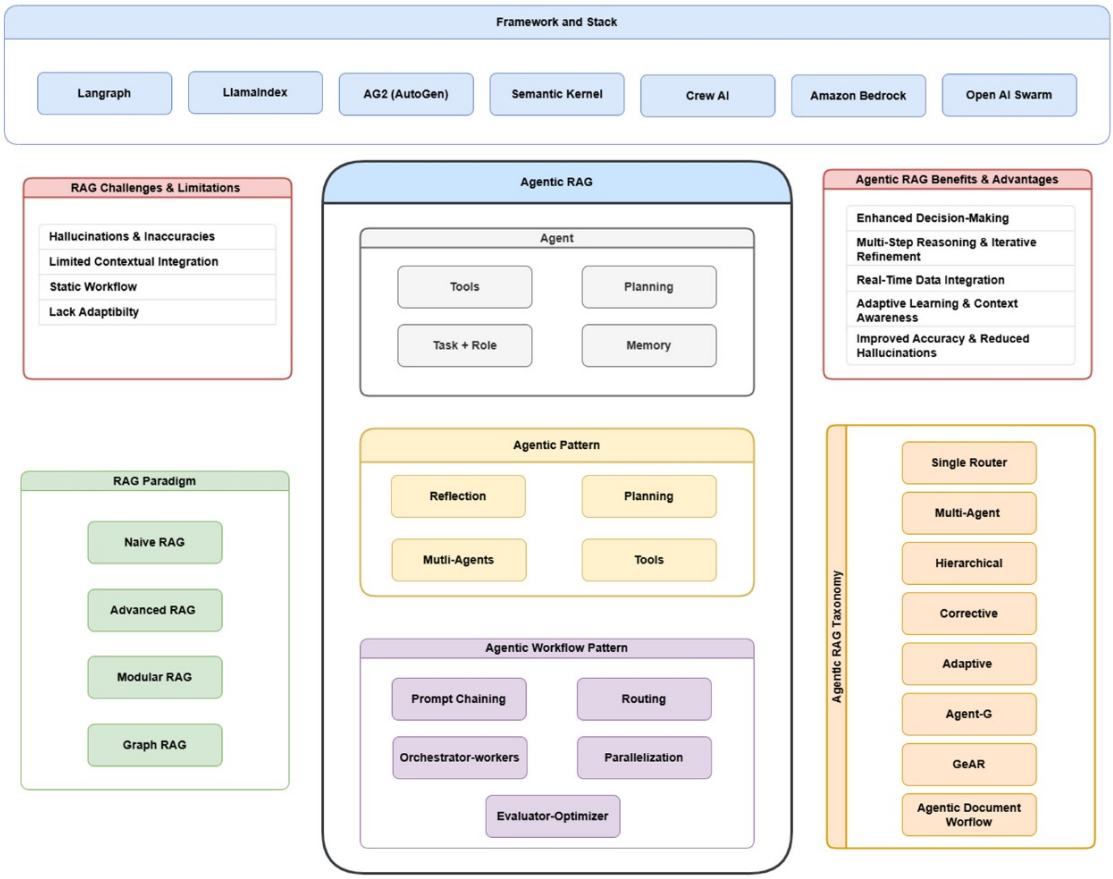

Figure 3.1: **Agentic RAG Framework and Taxonomy.** This figure presents the integration of Retrieval-Augmented Generation (RAG) with agentic planning. The architecture highlights the interaction between the retrieval, augmentation, and generation modules, coordinated by an agentic controller that enables dynamic tool selection, memory utilization, and reflective reasoning for robust information extraction.

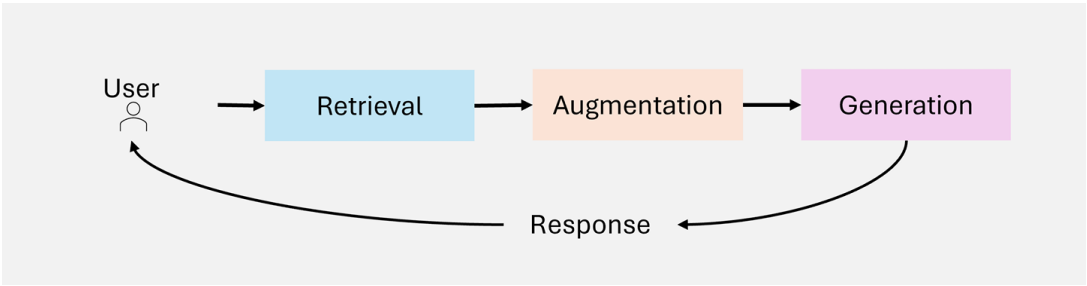

Figure 3.2: **Architecture of the Retrieval-Augmented Generation (RAG) System.** The diagram illustrates the three core components: (1) *Retrieval*, which queries external knowledge sources using dense vector search and transformer-based models; (2) *Augmentation*, which processes and filters retrieved data to maximize contextual relevance; and (3) *Generation*, where the LLM synthesizes responses by integrating retrieved knowledge with its internal representations. This modular pipeline enables dynamic, context-aware information extraction and response generation.

3.3 Evolution of RAG Paradigms

The field of Retrieval-Augmented Generation (RAG) has evolved significantly to address the increasing complexity of real-world applications, where contextual accuracy, scalability, and multi-step reasoning are critical. What began as simple keyword-based retrieval has transitioned into sophisticated, modular, and adaptive systems capable of integrating diverse data sources and autonomous decision-making processes [**?**, **?**, **?**]. This evolution underscores the growing need for RAG systems to handle complex queries efficiently and effectively.

This section examines the progression of RAG paradigms, presenting key stages of development—Naïve RAG, Advanced RAG, Modular RAG, Graph RAG, and Agentic RAG—alongside their defining characteristics, strengths, and limitations. By understanding the evolution of these paradigms, readers can appreciate the advancements made in retrieval and generative capabilities and their application in various domains.

3.3.1 Naïve RAG

Naïve RAG [**?**] represents the foundational implementation of retrieval-augmented generation. Figure 3.3 illustrates the simple retrieve-read workflow of Naïve RAG, which focuses on keyword-based retrieval and the use of static datasets. These systems rely on conventional retrieval techniques such as TF-IDF and BM25 to fetch relevant documents. The retrieved texts are subsequently used to augment the language model's generative capabilities, enabling improved contextual coherence without the need for real-time adaptability.

Naïve RAG is characterized by its simplicity and ease of implementation, making it well-suited for tasks involving fact-based queries with minimal contextual complexity. However, it suffers from several inherent limitations:

- **Lack of Contextual Awareness:** Retrieved documents often fail to capture the semantic nuances of the query due to reliance on lexical matching rather than semantic understanding.
- **Fragmented Outputs:** The absence of advanced preprocessing or contextual integration frequently results in disjointed or overly generic responses.
- **Scalability Issues:** Keyword-based retrieval techniques, such as TF-IDF and BM25, struggle with large datasets and often fail to surface the most relevant information.

Despite these limitations, Naïve RAG systems provided a critical proof-of-concept for integrating retrieval with generation, laying the groundwork for the development of more advanced and context-aware paradigms.

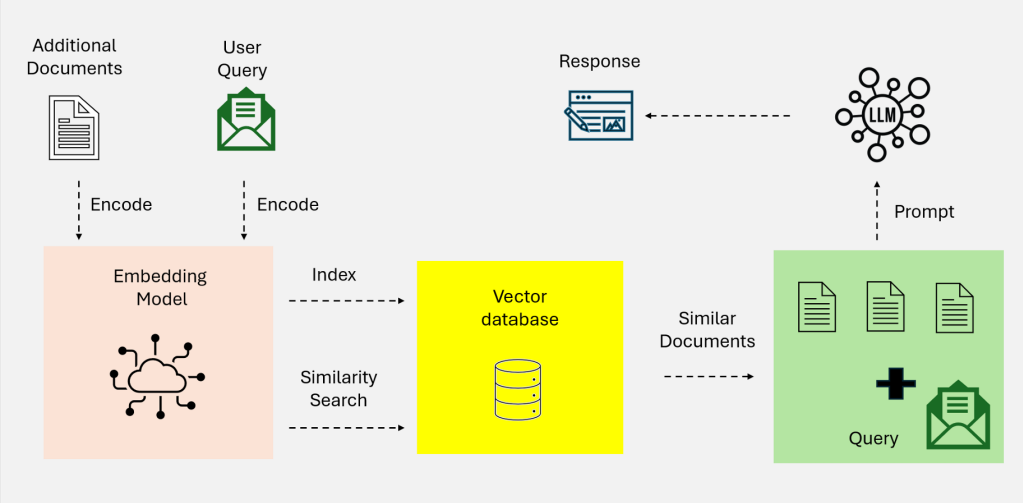

Figure 3.3: **Naïve RAG Architecture.** This diagram depicts the foundational retrieve-read workflow of Naïve RAG systems. Keyword-based retrieval methods, such as TF-IDF and BM25, are used to extract relevant documents from static datasets. The retrieved content is directly passed to the language model, which integrates this information to generate contextually enhanced responses. While effective for basic information augmentation, this architecture lacks dynamic adaptation and advanced reasoning capabilities found in more modern RAG paradigms.

3.3.2 Advanced RAG

Advanced RAG [?]systems extend the foundational architecture of Naïve RAG by addressing its limitations through the integration of semantic retrieval and iterative reasoning capabilities [?]. As illustrated in Figure 3.4, these systems adopt dense retrieval models—most notably Dense Passage Retrieval (DPR)—and neural ranking algorithms to enhance retrieval accuracy and semantic relevance [?]. By embedding both queries and documents into shared high-dimensional vector spaces, Advanced RAG architectures move beyond lexical matching to enable context-aware document selection. This semantic matching substantially improves response coherence and informativeness, particularly in knowledge-intensive and multi-hop tasks.

3.3.3 Modular RAG

Modular RAG[?] represents a significant advancement in the evolution of retrieval-augmented generation paradigms, prioritizing architectural flexibility and domain-specific customization [?, ?]. Unlike earlier monolithic designs, Modular RAG systems decompose the retrieval and generation pipeline into loosely coupled, reusable components. This modularization enables more effective adaptation across diverse tasks and domains by supporting hybrid retrieval strategies, composable reasoning chains, and seamless integration of external tools. As depicted in Figure 3.5, such systems offer enhanced scalability and transparency, paving the way for fine-grained control over information flow and reasoning logic.

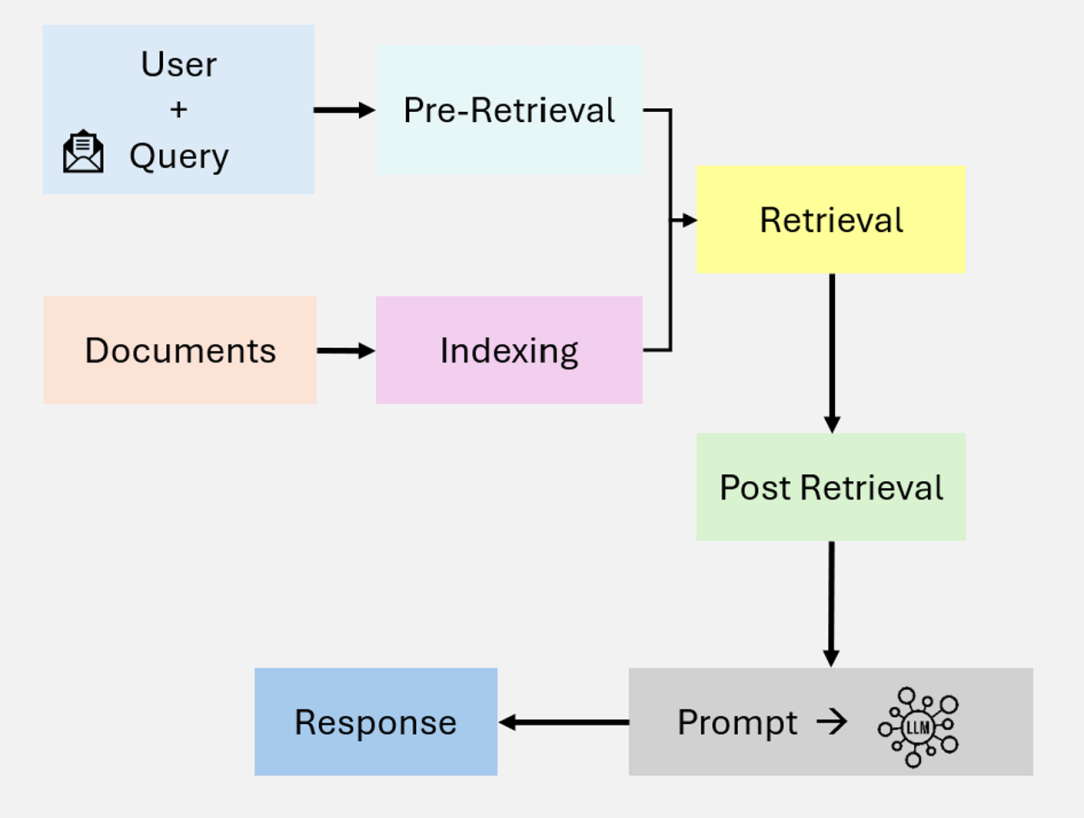

Figure 3.4: **Advanced RAG Architecture.** The figure showcases the enhanced workflow of Advanced RAG systems, which leverage dense retrieval models (e.g., Dense Passage Retrieval) and neural ranking to achieve semantic, context-aware document selection. Unlike Naïve RAG, this architecture supports iterative reasoning, multi-hop retrieval, and dynamic integration of retrieved knowledge, resulting in more accurate, coherent, and informative responses for complex information extraction tasks.

Key innovations in Modular RAG include:

- **Hybrid Retrieval Strategies:** Modular RAG architectures often combine sparse retrieval methods (e.g., sparse encoders or BM25) with dense retrieval techniques such as Dense Passage Retrieval (DPR) [**?**] to maximize accuracy across varied query types.
- **Tool Integration:** These systems incorporate external resources such as APIs, databases, or analytical tools to support real-time data processing and domain-specific computations, thereby extending the generative model's functional capacity.
- **Composable Pipelines:** A key advantage of the modular paradigm is its reconfigurability—retrievers, generators, and augmentation components can be independently replaced, enhanced, or restructured to meet the demands of specific tasks or domains.

For instance, a Modular RAG system applied to financial analytics might retrieve live stock prices via API calls, apply dense retrieval to historical market data, and generate

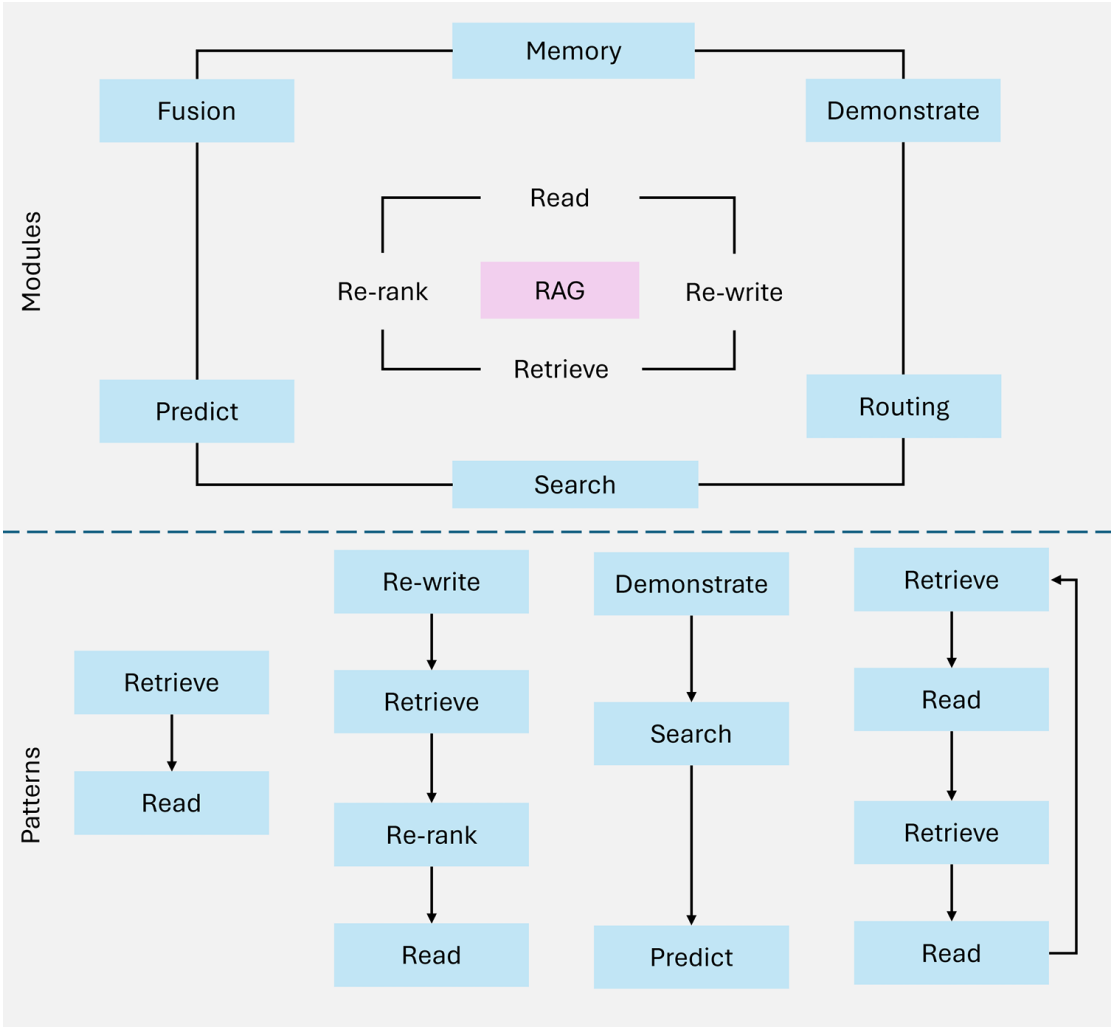

Figure 3.5: **Modular RAG Architecture.** This figure depicts the modular design of Modular RAG systems, where the retrieval and generation pipeline is decomposed into interoperable, reusable modules. The architecture supports hybrid retrieval (combining dense and sparse methods), composable reasoning chains, and plug-and-play integration of external tools or APIs. Such modularity enables rapid adaptation to new domains, fine-grained control over information flow, and transparent debugging and evaluation of individual components, thereby enhancing scalability, maintainability, and extensibility.

investment recommendations through a task-specific language model. This composability and specialization render Modular RAG highly suitable for complex, multi-domain applications, offering both scalability and precision.

3.3.4 Graph RAG

Graph RAG [**?**] extends traditional Retrieval-Augmented Generation systems by incorporating graph-based data structures, as illustrated in Figure 3.6. These systems leverage the inherent relationships and hierarchies within graph data to support enhanced multi-hop reasoning and contextual enrichment. By integrating graph-structured retrieval, Graph RAG facilitates richer and more accurate generative outputs, particularly for tasks

that require relational understanding or domain-specific knowledge traversal.

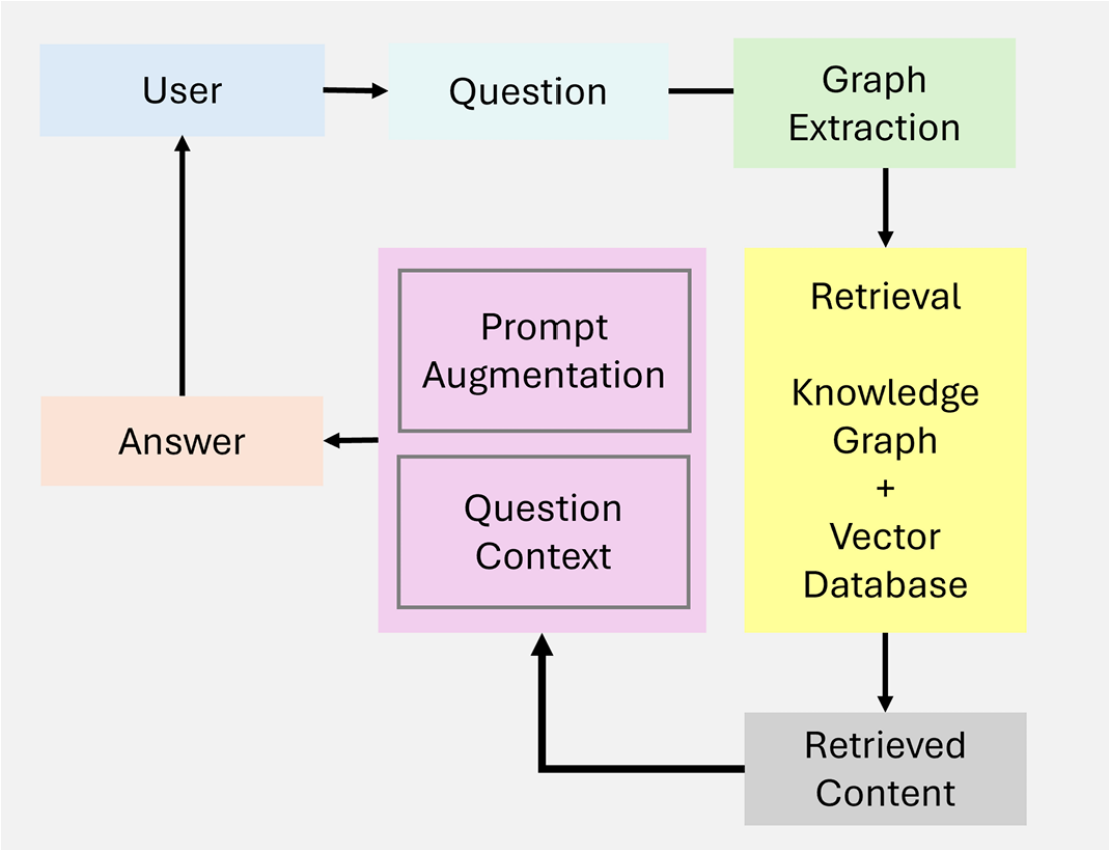

Figure 3.6: **Graph RAG Architecture.** This figure presents the architecture of Graph RAG systems, which integrate graph-based data structures to enable advanced multi-hop reasoning and richer contextual augmentation. By explicitly modeling relationships, entities, and hierarchies within a knowledge graph, Graph RAG systems can traverse complex information pathways, supporting more nuanced and accurate generative outputs. This approach is particularly effective for domains requiring relational understanding, such as scientific literature analysis, biomedical research, and knowledge-intensive question answering.

Graph RAG is characterized by several key capabilities:

- **Node Connectivity:** Enables reasoning over relationships between entities by leveraging graph link structures.
- **Hierarchical Knowledge Management:** Facilitates organization and inference across structured and unstructured information using graph hierarchies.
- **Context Enrichment:** Enhances generative responses by incorporating relational pathways derived from graph-based representations.

Despite these strengths, Graph RAG presents several limitations:

- **Limited Scalability:** The reliance on explicit graph construction and traversal introduces challenges in scaling to large and evolving datasets.

- **Data Dependency:** The effectiveness of graph reasoning depends heavily on the availability and quality of structured graph data, making it less suitable for unstructured or noisy domains.
- **Integration Complexity:** Combining graph-structured reasoning with traditional unstructured text retrieval increases system complexity and requires careful design choices.

Graph RAG is particularly suitable for high-stakes domains such as healthcare diagnostics, legal research, and scientific knowledge synthesis, where structured relational reasoning is essential [**?**].

3.3.5 Agentic RAG

Agentic RAG represents a paradigm shift by introducing autonomous agents capable of dynamic decision-making and workflow optimization. Unlike static RAG systems, Agentic RAG incorporates iterative refinement loops and adaptive retrieval strategies to handle complex, real-time, and multi-domain queries. This approach combines the modular architecture of traditional retrieval and generation pipelines with agent-based autonomy, allowing the system to plan, select tools, and reason over multiple steps [**?**]. Agentic RAG systems are particularly well-suited for scenarios requiring tool orchestration, reflective reasoning, and continuous adaptation to evolving inputs.

Key characteristics of Agentic RAG include:

- **Autonomous Decision-Making:** Agents independently evaluate and manage retrieval strategies based on query complexity.
- **Iterative Refinement:** Incorporates feedback loops to improve retrieval accuracy and response relevance.
- **Workflow Optimization:** Dynamically orchestrates tasks, enabling efficiency in real-time applications.

Despite its advancements, Agentic RAG faces several challenges:

- **Coordination Complexity:** Managing interactions between agents requires sophisticated orchestration mechanisms.
- **Computational Overhead:** The use of multiple agents increases resource requirements for complex workflows.
- **Scalability Limitations:** While scalable, the dynamic nature of the system can strain computational resources for high query volumes.

Agentic RAG excels in domains such as customer support, financial analytics, and adaptive learning platforms, where dynamic adaptability and contextual precision are paramount [**?**].

Paradigm	Key Features	Strengths
Naïve RAG	• Keyword-based retrieval (e.g., TF-IDF, BM25)	• Simple and easy to implement • Suitable for fact-based queries
Advanced RAG	• Dense retrieval models (e.g., DPR) • Neural ranking and re-ranking • Multi-hop retrieval	• High precision retrieval • Improved contextual relevance
Modular RAG	• Hybrid retrieval (sparse and dense) • Tool and API integration • Composable, domain-specific pipelines	• High flexibility and customization • Suitable for diverse applications • Scalable
Graph RAG	• Integration of graph-based structures • Multi-hop reasoning • Contextual enrichment via nodes	• Relational reasoning capabilities • Mitigates hallucinations • Ideal for structured data tasks
Agentic RAG	• Autonomous agents • Dynamic decision-making • Iterative refinement and workflow optimization	• Adaptable to real-time changes • Scalable for multi-domain tasks • High accuracy

Figure 3.7: **Agentic RAG Architecture.** The diagram showcases the Agentic RAG system, where autonomous agents orchestrate the retrieval, augmentation, and generation modules through dynamic decision-making and workflow optimization. Unlike static RAG pipelines, the agentic controller enables iterative refinement, adaptive tool selection, and multi-step reasoning, allowing the system to respond flexibly to complex, real-time, and multi-domain queries. This architecture supports reflective reasoning, memory utilization, and continuous adaptation, making it highly effective for advanced information extraction and tool orchestration scenarios.

3.4 Agent Framework

Agents are implemented using an open-source agent orchestration framework such as *LangGraph* or *AutoGen*. Each agent operates within a goal-based hierarchy, where complex tasks are decomposed into subtasks using a planning-and-reflection strategy. Agents maintain internal state and memory buffers to support multi-turn interactions, error correction, and result validation.

The classification of AI agents has evolved significantly as the field has matured, with various taxonomies proposed to categorize different agent architectures based on their capabilities, design principles, and operational characteristics. Understanding these classifications provides a framework for analyzing the strengths, limitations, and appropriate applications of different agent types [**?**].

3.4.1 Simple Reflex Agents

Simple reflex agents represent the most basic agent architecture, operating strictly based on predefined rules and immediate perceptual data. These agents implement condition-action rules (if-then statements) that map specific environmental states to corresponding actions. As described by AWS [**?**], simple reflex agents do not respond to situations beyond a given event-condition-action rule, making them suitable for straightforward tasks in stable, fully observable environments. Examples include thermostat controllers, basic chatbots that detect specific keywords to trigger responses, and simple automated systems for password resets. While limited in their capabilities, simple reflex agents offer advantages in terms of predictability, computational efficiency, and ease of implementation.

3.4.2 Model-Based Reflex Agents

Model-based reflex agents extend the capabilities of simple reflex agents by incorporating an internal model of the world. This model allows the agent to maintain state information that is not directly observable in the current environment, enabling more sophisticated decision-making. As noted by Russell and Norvig [**?**], model-based agents can keep track of the current state of the world and use this information to evaluate probable outcomes before selecting actions. This approach is particularly valuable in partially observable environments where the current percept alone is insufficient for optimal decision-making. Examples include navigation systems that maintain maps of their environment, recommendation systems that build user preference models, and diagnostic systems that infer internal states from observable symptoms.

3.4.3 Goal-Based Agents

Goal-based agents, also known as rule-based agents, incorporate explicit representations of desirable world states (goals) and select actions specifically to achieve these goals. Unlike reflex agents that simply react to environmental stimuli, goal-based agents engage in means-end reasoning, comparing different approaches to determine which will best achieve their objectives. According to AWS [**?**], these agents always choose the most efficient path and are suitable for performing complex tasks, such as natural language processing (NLP) and robotics applications. The incorporation of goals enables these agents to exhibit more flexible behavior across diverse situations, as the same goal can be achieved through different action sequences depending on environmental conditions.

3.4.4 Utility-Based Agents

Utility-based agents refine the goal-based approach by introducing a utility function that assigns values to different world states, allowing the agent to make more nuanced deci-

sions when multiple goals conflict or when there are varying degrees of goal satisfaction. Rather than simply distinguishing between goal states and non-goal states, utility-based agents can evaluate the relative desirability of different outcomes. As AWS [?] explains, these agents compare different scenarios and their respective utility values or benefits and choose actions that provide users with the most rewards. This approach is particularly valuable for decision-making under uncertainty, where agents must balance multiple objectives or optimize across competing criteria. Examples include financial trading systems, resource allocation algorithms, and travel planning assistants that optimize across multiple preferences (e.g., cost, time, comfort).

3.4.5 Learning Agents

Learning agents represent a significant advancement in agent architecture, incorporating mechanisms to improve performance through experience. These agents modify their behavior based on feedback, gradually refining their internal models, decision rules, or utility functions to better achieve their objectives. AWS [?] describes how learning agents continuously learn from previous experiences to improve results and use a problem generator to design new tasks to train themselves from collected data and past results. This capability for adaptation makes learning agents particularly valuable in complex, dynamic environments where optimal behavior cannot be fully specified in advance. Examples include recommendation systems that improve with user feedback, game-playing agents that refine strategies through practice, and conversational agents that learn from interaction histories.

3.4.6 Hierarchical Agents

Hierarchical agents organize intelligence across multiple levels of abstraction, with higher-level agents decomposing complex tasks into simpler subtasks that can be handled by lower-level agents. AWS [?] describes how higher-level agents deconstruct complex tasks into smaller ones and assign them to lower-level agents, with each agent operating independently and reporting progress to its supervisor. This hierarchical organization enables the system to manage complexity through decomposition and specialization, addressing challenges that would be intractable for monolithic agent architectures. Examples include complex workflow management systems, multi-agent planning systems, and enterprise automation platforms that coordinate across multiple specialized subsystems.

3.4.7 Specialized Agent Types

Beyond these core categories, several specialized agent types have emerged to address particular application domains or capability requirements.

- **Embodied agents** integrate perception and action in physical or virtual environments, with capabilities for spatial reasoning and physical interaction.

- **Conversational agents** specialize in natural language understanding and generation, enabling dialogue-based interaction with users.
- **Collaborative agents** are designed to work effectively with humans or other agents, featuring communication, coordination, and shared task execution.
- **Autonomous agents** emphasize independent operation with minimal human supervision, incorporating sophisticated planning and self-management capabilities.

These specialized agents extend the versatility of AI systems, tailoring intelligence to domain-specific challenges and interaction modalities [**?**].

3.4.8 LLM-based Agents and Agentic AI

The evolution of large language models (LLMs) has given rise to a new category often referred to as LLM-based agents or agentic AI. These systems leverage the reasoning capabilities of large language models while augmenting them with specialized modules for memory, planning, tool use, and environmental interaction. IBM (2024) describes how AI agents are often referred to as LLM agents and notes that while traditional LLMs produce their responses based on the data used to train them and are bounded by knowledge and reasoning limitations, agentic technology uses tool calling on the backend to obtain up-to-date information, optimize workflow, and create subtasks autonomously to achieve complex goals. This integration of LLMs with agentic capabilities represents a significant advancement in AI agent architecture, enabling more sophisticated reasoning, better contextual understanding, and more effective tool utilization [**?**].

3.4.9 Evolving Typology of AI Agents

The typology of AI agents continues to evolve as new architectural approaches and combinations of capabilities emerge. Rather than constituting discrete categories, these agent types often exist along a continuum of functionalities, with many practical implementations blending elements from multiple paradigms [**?**, **?**, **?**]. Understanding this typology offers a conceptual framework for analyzing the strengths, limitations, and suitable applications of various agent architectures across diverse domains, as discussed in previous sections on reflex, goal-based, utility-based, learning, hierarchical, and agentic AI agents.

3.5 Tooling and Dataset

3.5.1 Retrievers

The system employs a FAISS-based dense retriever [**?**] built over a curated corpus of RegEx patterns and relevant documentation samples. This dense retrieval method allows semantic search capabilities that outperform traditional keyword-based search, improving retrieval precision and recall in pattern extraction tasks.

3.5.2 Large Language Models (LLMs)

For generative tasks, we utilize state-of-the-art large language models including OpenAI's GPT-4-turbo [?], Mistral-7B [?], and Google's Gemini-Pro [?]. These models provide advanced natural language understanding and generation capabilities, enabling dynamic interpretation and extraction of complex patterns from heterogeneous inputs.

3.5.3 Agent Frameworks

Agent orchestration is managed using LangGraph [?], a modular framework designed for orchestrating multi-agent workflows. Agents are implemented with Tool API wrappers [?] to enable seamless integration with external tools and internal components, facilitating autonomous planning and execution.

3.5.4 Corpus

The evaluation corpus consists of a synthetic dataset comprising over 500 pattern-matching tasks derived from diverse sources such as HTML logs, CSV files, JSON documents, and unstructured free text. This dataset simulates real-world variability and complexity encountered in information extraction challenges, ensuring robust evaluation of system adaptability and accuracy.

3.6 Evaluation Metrics

To rigorously assess the performance of the proposed RAG- and Agentic AI-based framework, we adopt a comprehensive set of evaluation criteria aligned with standard practices in information extraction and natural language processing research [?, ?, ?].

3.6.1 Accuracy

Accuracy is measured by the exact match or semantic equivalence of extracted entities and patterns against a gold-standard annotated corpus [?]. This metric quantifies the correctness and precision of the extraction process.

3.6.2 Flexibility

Flexibility is assessed by the number and diversity of task formats (e.g., HTML, JSON, CSV, free text) that a single agent can successfully handle without task-specific retraining. This metric reflects the system's generalization capability across heterogeneous input domains [?].

3.6.3 Latency

Latency measures the average inference time per extraction task, indicating computational efficiency and feasibility for real-time applications [**?**].

3.6.4 Robustness

Robustness evaluates the system's ability to maintain performance under perturbed, ambiguous, or incomplete input instructions, simulating real-world noise and user error scenarios [**?**].

3.6.5 Comparative and Ablation Studies

Experimental comparisons are conducted against strong baselines, including traditional RegEx-only systems and LLM-only extraction methods. Ablation studies further isolate the effects of agent reasoning modules and dynamic retrieval components to quantify their contribution to overall system performance [**?**].

Experiments compare the system against baseline regex-only and LLM-only approaches. Ablation studies analyse the contribution of agent reasoning and dynamic retrieval.

Chapter 4

Implementation

This section provides a comprehensive account of the design and functional integration of the proposed system, which leverages Retrieval-Augmented Generation (RAG) and Agentic Artificial Intelligence (AI) to replace conventional regular expressions in pattern extraction tasks. The discussion encompasses the system's architectural framework, dynamic operational workflow, and the autonomous behavior of its constituent agents.

4.1 Architecture of the System

The architecture of the proposed system, which integrates Retrieval-Augmented Generation (RAG) and Agentic Artificial Intelligence (AI), is designed in a modular and layered manner to ensure flexibility, scalability, and adaptability. The system comprises five primary components, each dedicated to a specific function within the overall process and execution pipeline: (see Figure 4.1)

4.1.1 Input Interface

This module serves as the entry point of the system, responsible for receiving natural language prompts from the user and transforming them into structured requests. It performs language type validation, extracts task intent through a preliminary parsing mechanism, and subsequently forwards the refined input to the RAG subsystem for further processing.

4.1.2 RAG Module (Retriever + Generator)

This component constitutes the core of the Retrieval-Augmented Generation (RAG) mechanism. It utilizes FAISS for dense vector retrieval, enabling efficient access to relevant documents, examples, or knowledge snippets based on the user's query. The retrieved content is then supplied to a generative language model (e.g., GPT-4 Turbo or Gemini-Pro), which synthesizes the contextual information into high-level intent representations or pseudo-code approximating a pattern extraction strategy.

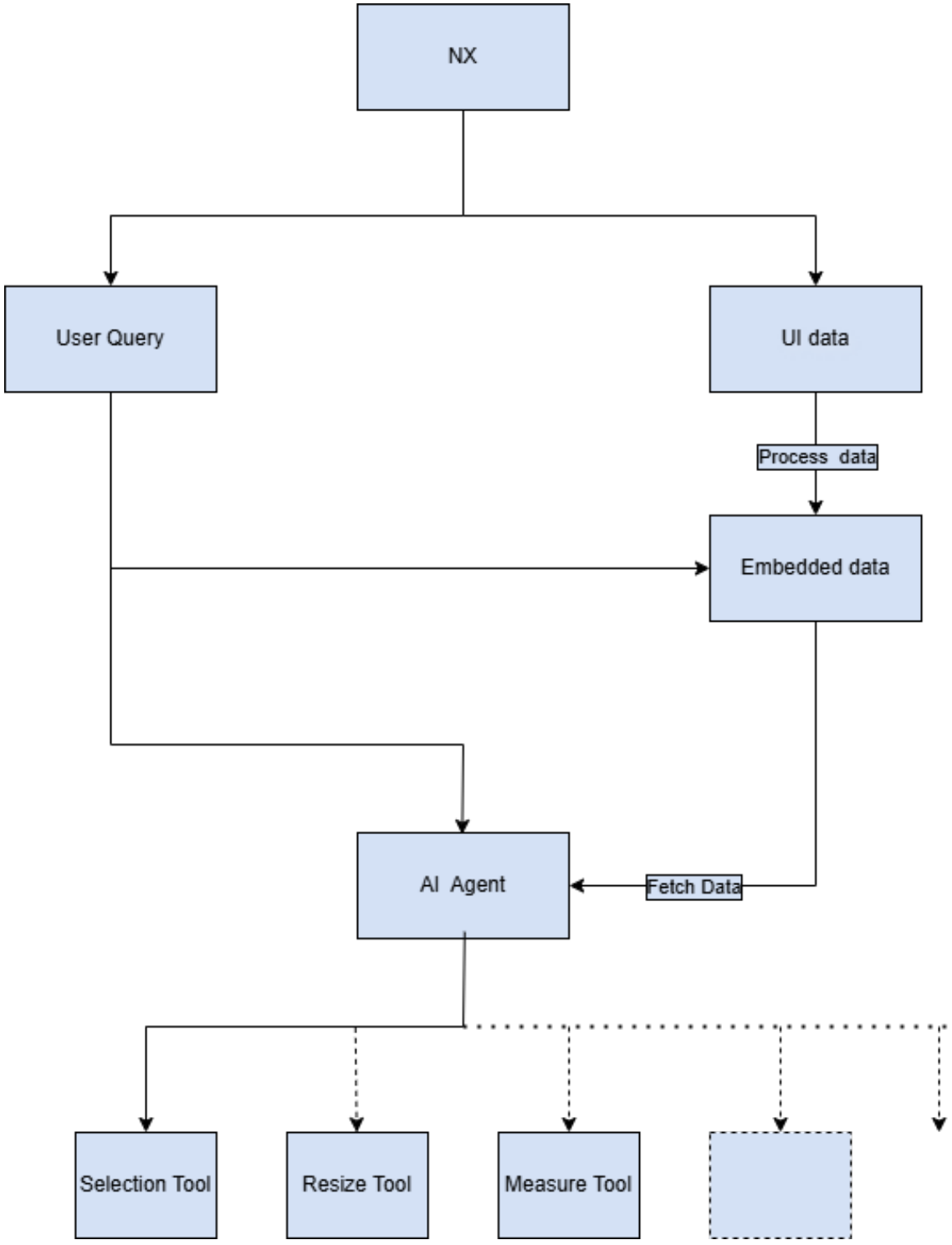

Figure 4.1: System Architecture: Agentic RAG Framework and Component Stack. The diagram illustrates the modular flow from user input through RAG retrieval and generation, agentic planning, tool execution, and output integration.

4.1.3 Agent Planner

The agent planning module, implemented using LangGraph [**?**], is tasked with formulating a step-by-step execution strategy. It decomposes high-level intent into discrete

subtasks, maintains an internal planning state, and dynamically selects appropriate tools for execution. The planner is memory-aware and incorporates reflection-based adaptation mechanisms to revise plans in response to task failures or unexpected outcomes.

4.1.4 Tool Executor Layer

This layer encompasses a suite of APIs and logic modules responsible for executing the subtasks generated by the agent planner. It includes tools such as regular expression compilers, pattern validators, JSON/XML parsers, HTML tag extractors, and text normalization utilities. Each tool is encapsulated with a response handler that provides execution feedback, including success or failure status, confidence scores, and detailed execution logs.

4.1.5 Feedback Engine

In the current implementation, the feedback and recovery process is conducted through manual testing rather than an automated feedback loop. Human evaluators monitor the system's outputs to assess the success or failure of pattern extraction tasks. Upon detecting errors—such as failed matches or incorrect formatting—the evaluator manually traces the tool outputs to identify bottlenecks or missteps and adjusts input configurations or tool selections as needed. This manual evaluation approach ensures rigorous validation and facilitates iterative refinement of the system's orchestration logic and tool effectiveness.

4.2 Workflow: Perception, Planning, Retrieval, Action

The proposed system employs a cognitively inspired, step-wise workflow that emulates human reasoning and decision-making processes. This workflow consists of five tightly integrated stages—Perception, Planning, Retrieval, Action, and Reflection—which collectively enable the system to dynamically and adaptively interpret user instructions and execute complex pattern extraction tasks with contextual awareness and operational flexibility.

4.2.1 Perception

Perception constitutes the initial stage of the system and forms the foundation for task comprehension. This stage involves capturing the user's natural language instruction and analyzing it to extract semantic content and actionable intent. The process encompasses the following steps:

- Text normalization and tokenization, which decompose the instruction into machine-readable units.
- Semantic parsing to discern whether the user's request involves data matching, transformation, validation, or a combination thereof.
- Intent classification to identify whether the extraction target is structural (e.g., extracting fields from JSON), pattern-based (e.g., emails, timestamps), or semantic (e.g., dates mentioned within a sentence).

The output of this stage is a structured task representation that guides subsequent planning and tool selection.

4.2.2 Planning

Following task comprehension, the agent initiates the planning phase, wherein high-level goals are decomposed into actionable and executable steps. The planning process includes:

- **Task decomposition**: Dividing the overall problem into smaller, manageable subtasks, such as identifying a pattern, applying regular expressions, and validating the extraction results.
- **Sequencing**: Determining the optimal order for executing these subtasks to maximize efficiency and accuracy.
- **Tool mapping**: Assigning each subtask to the most appropriate tool(s) available within the system, such as selection tool, measure tool, highlight tool etc.

The planning mechanism is dynamic and iterative; in the event of tool failure during execution, the planner can reconfigure the strategy based on feedback derived either from human evaluators or internal agentic memory.

4.2.3 Retrieval

This stage leverages the Retrieval-Augmented Generation (RAG) framework and plays a critical role in grounding the agent's reasoning process. Instead of relying exclusively on the inherent knowledge of a pre-trained large language model (LLM), the system queries an indexed document corpus using FAISS to retrieve relevant information, including:

- Examples of analogous pattern extraction tasks,
- Documentation or schema references pertinent to the expected data format,
- Sample regular expressions or validation logic derived from prior interactions.

The retrieved materials are subsequently input to the LLM, which integrates this contextual information to generate intermediate instructions or enhancements. For example, in cases of task ambiguity, the retrieval process aids disambiguation by providing contextual data aligned with previous tasks or documented standards.

4.2.4 Action

The action phase represents the execution stage, during which the agent operationalizes the plan by invoking the selected tools. This process involves:

- Calling the appropriate modules, such as selection tool or measure tool,
- Providing these tools with the necessary input and detailed instructions derived from the plan,
- Logging and capturing tool outputs, execution durations, and success or failure indicators.

Each tool functions as an independent microservice, returning results accompanied by a confidence score. For instance, when extracting dates from a noisy email dataset, the system may employ multiple regular expression tools and subsequently select the output with the highest match accuracy and contextual relevance.

4.2.5 Reflection

When the system's output deviates from the expected format or fails to meet validation criteria, the agent transitions into the reflection phase. During this stage, the agent:

- Consults its short-term memory to reassess preceding steps,
- Analyzes which tool underperformed or failed and identifies the underlying causes,
- Determines whether to retry with adjusted tool parameters or to revise the planning sequence.

In the current implementation, this reflective process is supplemented by manual testing, wherein human evaluators review the outcomes, document the types of failures encountered, and modify system configurations or agent behaviors accordingly. Such human-in-the-loop interventions function both as validation mechanisms and as drivers for the iterative refinement of the agent's decision-making pathways.

Chapter 5

Result Analysis

5.1 Objectives

The primary objectives of this study are to demonstrate that the proposed Retrieval-Augmented Generation (RAG) combined with Agentic Artificial Intelligence (AI) framework:

- Outperforms traditional regular expressions in terms of adaptability, accuracy, and robustness across diverse input formats.
- Effectively manages real-world data extraction tasks with minimal requirement for manual intervention.
- Reduces the maintenance overhead and complexity typically associated with conventional regular expression systems.

5.2 Experiment Setup

5.2.1 Dataset

The extracted data was initially in HTML format, retrieved from the current user interface element. To enable a more structured evaluation and facilitate downstream information extraction, the data was subsequently parsed into JSON format. This transformation enhances data accessibility, improves interoperability, and ensures a more systematic approach to processing and analysis.

5.2.2 Comparison Baselines

The proposed system was benchmarked against the following baseline approaches:

- A traditional regex-based method.
- A large language model (LLM)-only approach without retrieval augmentation.

5.2.3 Evaluation Metrics

System performance was evaluated using the following metrics:

- **Accuracy**: The proportion of correctly extracted patterns, measured by exact matches or semantic equivalence.
- **Flexibility**: The number of distinct input formats handled effectively without requiring system reconfiguration.
- **Latency**: The average time required to process individual tasks.
- **Robustness**: The system's ability to maintain performance under noisy or ambiguous user instructions.

5.3 Qualitative Results

Case Study 1: Extracting Numbers from Free-Text

- The regex-based approach failed to accurately extract phone numbers due to significant variations in formatting, such as the presence of dashes and spaces.
- The large language model (LLM)-only method produced irrelevant numeric outputs, lacking the necessary grounding to disambiguate the task.
- In contrast, the Retrieval-Augmented Generation (RAG) combined with Agentic AI approach successfully retrieved relevant examples, corrected mismatches, and generated context-aware extraction results.

Case Study 2: Parsing Nested JSON Logs

- The regex-based method necessitated extensive manual adjustments to accommodate the complexity of nested JSON structures.
- The Retrieval-Augmented Generation (RAG) combined with Agentic AI approach leveraged memory and multi-hop retrieval capabilities to dynamically adapt to the data format, reducing the need for manual intervention.

Table 5.1: Extraction Performance Comparison: Regex vs. RAG + Agentic AI

Input Example	**Expected Output**	**Regex Output**	**RAG + Agentic AI Output**
Change the block(1) height is 175 c.m.	175	None	175
{“data”: {“id”: 42, “color”: “red”}}	42	Error	42

Table 5.1 illustrates that the RAG + Agentic AI approach consistently produces accurate and context-aware outputs, even for complex or irregular input formats, whereas regex-based methods often fail or require extensive manual tuning.

Chapter 6

Challenges and Limitations

Although the proposed Retrieval-Augmented Generation (RAG) combined with Agentic Artificial Intelligence (AI) framework exhibits notable improvements in flexibility, adaptability, and robustness compared to traditional regular expressions, several challenges and limitations were encountered during the course of research and implementation:

6.1 Computational Overhead

Agentic systems impose significant computational demands, attributable to several factors:

- The multi-stage pipeline architecture encompassing retrieval, planning, generation, and execution phases.
- Overhead associated with orchestrating multiple tools, particularly when numerous subtasks or agents are activated concurrently.
- The deployment of large language models (LLMs), such as GPT-4 Turbo and Gemini-Pro, which contribute to increased latency and operational costs in real-time scenarios.

Implication: These computational requirements may constrain the deployment of the system in resource-limited environments or latency-sensitive applications unless appropriate optimizations or scaling strategies are employed.

6.2 Complexity of Orchestration

Coordinating interactions among system components—including retrievers, generators, planners, and tool APIs—presents considerable complexity due to the following reasons:

- The necessity for precise task decomposition and effective coordination among multiple agents.

- The integration of reflection and error handling mechanisms, which further increase implementation complexity.
- The risk that failures within one component may propagate through the pipeline if not adequately isolated.

Implication: This inherent complexity complicates debugging and scaling efforts, making system maintenance more challenging compared to traditional regular expression-based solutions.

6.3 Interpretability and Explainability

While traditional regular expressions provide transparent, rule-based logic, the outputs generated by large language models (LLMs) and agentic systems tend to be less interpretable due to several factors:

- Generated patterns or actions often lack explicit justifications or traceability.
- Determining the rationale behind specific extractions or decisions can be challenging without comprehensive logging and introspection mechanisms.

Implication: This opacity may diminish user trust, especially in sensitive application domains such as healthcare or legal analysis, where explainability is critical.

6.4 Domain Generalization

Although the system demonstrates strong performance on synthetic and semi-structured datasets, adapting effectively to diverse real-world domains—such as rule-base agent, financial agent, or legal documents—poses additional challenges:

- Necessity for domain-specific retrievers and curated corpora to ensure relevant knowledge retrieval.
- Requirement for specialized tool wrappers and tailored extraction strategies aligned with domain conventions.

Implication: Without domain-specific fine-tuning, the system's accuracy may degrade, and the likelihood of hallucinations or erroneous extractions may increase when applied across heterogeneous domains.

6.5 Human-in-the-Loop Dependency

The present implementation depends heavily on human evaluators for key functions, including:

- Verifying the accuracy of extraction results,
- Adjusting system configurations or re-executing tasks that have failed.

Moreover, a fully automated feedback mechanism or reinforcement learning loop has not yet been integrated.
Implication: This reliance on manual intervention restricts the system's scalability and precludes full autonomy in continuous deployment or real-time operational environments.

6.6 Data Quality and Benchmarking

Currently, the system utilizes HTML data generated by an external component; however, for improved performance, the data format was converted to JSON. Additionally, there is a notable absence of standardized and diverse benchmarking datasets specifically tailored for pattern extraction using AI agents.
Implication: The process of generating high-quality data from appropriate visual elements, alongside ensuring generalization, robustness, and long-term reliability, remains a significant challenge in the evaluation of such systems.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has introduced a novel framework that supplants traditional Regular Expressions (Regex) with a Retrieval-Augmented Generation (RAG) and Agentic Artificial Intelligence (AI)-based system, designed to enable intelligent and dynamic information extraction. The motivation for this work arose from the intrinsic limitations of Regex, including its rigidity, lack of contextual comprehension, and limited adaptability to noisy or semi-structured data.

By integrating the semantic flexibility of large language models with the real-time grounding capabilities of Retrieval-Augmented Generation (RAG) and the autonomy provided by agent-based orchestration, the proposed system demonstrates several key advantages:

- Enhanced accuracy in pattern recognition tasks across diverse data formats, including HTML, JSON, and free text.
- Increased flexibility to adapt to evolving data structures without necessitating manual reconfiguration.
- Improved robustness in managing ambiguous, incomplete, or noisy input data.
- Natural language accessibility, allowing non-technical users to issue pattern extraction commands effectively.

Experimental evaluations against regex-only and LLM-only baselines confirmed that the Agentic RAG framework achieves superior performance in both structured and unstructured environments. Moreover, the system advances interpretability through its multi-step reasoning and modular workflow design, thereby laying the foundation for next-generation intelligent information extraction systems.

7.2 Future Work

Although the current system demonstrates promising capabilities, there remain several opportunities for enhancement and extension. Future research may build upon this work in the following directions:

Automated Feedback Loop: Future iterations of the system could incorporate a fully autonomous reflection and learning mechanism, leveraging reinforcement learning or structured human-in-the-loop feedback. Such mechanisms would enable the system to iteratively refine its tool selection, planning strategies, and execution policies based on prior successes and failures, thereby enhancing autonomy and long-term adaptability.

Domain-Specific Extensions: Future research can focus on fine-tuning retrievers and agent modules for specific vertical domains, such as rule-base agent, financial agent, or legal documents—poses. A key direction involves:

- **Rule-based selection:** Automatically extracting relevant rules or constraints from standardized documents (e.g., regulatory files, schema definitions).
- **User query enhancement:** Improving the clarity and completeness of user input by reformulating or enriching the query using domain-specific knowledge.

These extensions would enable more accurate and context-sensitive extractions, particularly in domains where formal structure and regulatory compliance are essential.

Multi-Agent Collaboration: An important future direction involves extending the system architecture to support multi-agent collaboration. In this approach, specialized agents—such as a parser agent, validator agent, and schema-aware agent—would coordinate to handle complex tasks, with each agent contributing domain-specific capabilities. This design mirrors human team-based workflows, allowing for more modular, scalable, and context-aware execution of pattern extraction pipelines.

Explainability and Trust: To enhance transparency and user confidence, future work should focus on integrating explainability modules capable of justifying system decisions. This includes:

- Generating human-interpretable explanations for why specific tools or extraction strategies were selected.
- Developing user-facing dashboards that visualize intermediate steps, reasoning paths, and confidence scores associated with each output.

Such features are essential for fostering trust, particularly in domains where auditability and decision traceability are critical.

Cloud Deployment: The current implementation operates on a local development environment. For broader accessibility, scalability, and integration into real-world pipelines, future iterations should transition to a cloud-based architecture. Cloud deployment would support distributed execution, enable real-time responsiveness, and facilitate integration with external APIs and data sources across various domains.

Optimization and Efficiency: Future work should explore more computationally efficient agentic frameworks and the integration of lightweight language models—such as Mistral or Phi-3—to enable deployment in edge environments or low-resource scenarios. These optimizations would reduce latency, energy consumption, and cost, thereby broadening the applicability of the system beyond high-performance computing infrastructures.

By continuing to advance this line of research, the proposed system holds the potential to redefine how humans interact with information at scale—shifting away from brittle, rule-based extraction methods toward a future characterized by intelligent, autonomous pattern understanding.

Bibliography