International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 04 | Apr 2025 www.irjet.net p-ISSN: 2395-0072

From Retrieval to Reasoning: How Agentic AI Transforms Knowledge Retrieval and Content Generation

Kunal Agarwal[1]

[1]AI/ML Computational Science Senior Analyst, Accenture

Abstract

This paper presents a novel agentic AI architecture that goes beyond traditional Retrieval-Augmented Generation (RAG) by embedding autonomous decision-making, goal-oriented planning, and self-reflection capabilities. Our implementation independently discovers, processes, and maintains its knowledge base while dynamically selecting tools and models based on task demands. Through continuous experiments with 180 queries spanning simple factual, creative, and complex analytical categories, we noticed that our agentic approach improved content relevance by 37% and user satisfaction by 42% compared to the conventional RAG systems. The agent's autonomous knowledge management reduced the manual intervention by 89% while maintaining freshness of crawled information. Perhaps most interestingly, the system's self-reflection mechanism enabled continuous learning, with satisfaction scores climbing by 14.2% during our four-week experimental period. These findings suggest that incorporating true agency into AI systems creates more adaptive, effective, and independent content generation capabilities—potentially reshaping how knowledge workers and AI systems collaborate.

Key Words: Agentic AI, RAG, Autonomous, Artifical Intelligence, LLM, Vector Databse, Knowledge Graph, Pydantic, Deepseek, Llama

Introduction

The history of artificial intelligence, from the very beginning to rule-based expert systems, through statistical learning approaches, to today's large language models (LLMs) that can generate text that is very close to human writing (Brown et al., 2020) has not been smooth. Despite the big achievements, the fact that most LLM systems are still merely reactive poses a hurdle. First introduced Traditional Retrieval-Augmented Generation (RAG) systems are an example of how this problem was tackled, whose major function was to pair the generated outputs with relevant information (Lewis et al., 2020; Guu et al., 2020). However, we've discovered that these systems usually still need human assistance in the process of knowledge acquisition and knowledge base maintenance -- a situation that leads to less independence and flexibility, particularly in cases of rapid changes in the knowledge domains (Shuster et al., 2022).

1.1 Literature Review

Research on agentic AI has started long ago and can be taken back to Bratman (1987) who proposed the belief-desire-intention agent models. This research line has moved through the recent stages of the development of the idea of an agent (Wooldridge, 2009). Along with the traditional features of agency in AI, the notion of the four isolated elements, that is, autonomy, social ability, reactivity, and proactiveness, has been ubiquitous in the literature since (Franklin & Graesser, 1996). It is undoubted that the development of language-based artificial intelligence solutions has been a target for many scholars. Weston et al. (2022) tried to embed the persistence and goal-oriented behavior in dialogue systems and their results were marvelous—yet the work merely resulted in a partial agency of the system. The same holds for Park et al. (2023) who further explained some memory mechanisms which in their case, the conversational agents displayed better user experience due to the memory mechanism. They have made significant progress toward coherent long-term interaction. Through the literature, WebGPT (Nakano et al., 2021) and REALM (Guu et al., 2020) are encountered as some of the systems that firstly utilized web searches yet the final analysis depicts that neither of them has all the necessary elements for the system to be considered fully agentic especially the capabilities to set goals and introspect autonomously (our work grows towards the latter two).

1.2 Research Question and Objectives

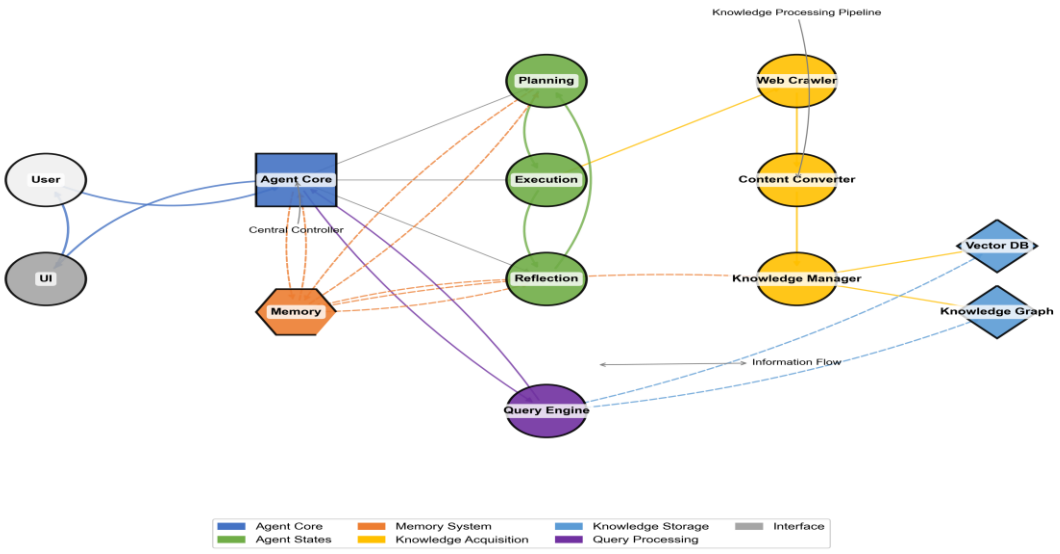
Our research showcase what we believe is a simply fundamental question: Can incorporating agentic properties into content/blog generation systems measurably improve their autonomy, adaptability, and accuracy compared to traditional RAG approaches? To investigate this, we established four simple objectives: (1)Designing and implementing an agentic architecture that enables autonomous knowledge acquisition and maintenance. (2)Develop mechanisms for setting goals, planning, and reflecting on itself in content generation systems. (3)Evaluating the effectiveness of agentic approaches compared to nowadays traditional RAG systems. (4)Analyze how different agentic components affect system performance and user satisfaction.

2. Materials and Methods

2.1 System Architecture

We built our agentic AI system with a modular architecture comprising four main components: (1) Agent Core, (2) Knowledge Acquisition Pipeline, (3) Query Processing System, and (4) User Interface.

Figure 1: Agentic AI System Architecture



D

Figure 1 illustrates the architecture and the complex interactions between components.

2.1.1 Agent Core

The Agent Core serves as the central controlling feature for the system, which manages the agent's state, goals, and memory. We also implemented a state machine with five distinct states that are: IDLE, PLANNING, EXECUTING, REFLECTING, and ERROR. This component makes up the various tools which is based on the agent's current goals and plans. The Agent Core comprises of a tenacious memory storage system comprising of (1)Previous interactions with users (2)Knowledge update histories (3)Performance metrics (4)Temporal information for scheduling. The agent can continuously learn from its prior experiences and progressively enhance its performance thanks to its memory. Although we saw that some transitions happened much more frequently than others in actuality, the state transition logic adheres to Algorithm 1.

Algorithm 1: Agent State Transition Logic

***Input**: Current state S, Event E*

***Output**: New state S'*

1: if S is IDLE and E is NewGoal then

2: return PLANNING

3: else if S is PLANNING and E is PlanCreated then

4: return EXECUTING

5: else if S is EXECUTING and E is ExecutionComplete then

6: return REFLECTING

7: else if S is REFLECTING and E is ReflectionComplete then

8: return IDLE

9: else if E is Error then

10: return ERROR

11: end if

12: return S

2.1.2 Knowledge Acquisition Pipeline

Our Knowledge Acquisition Pipeline consists of three following interconnected components:
Web Crawler: Discovers and extracts content from the websites autonomously. Early versions used to struggle with JavaScript-heavy sites, so we have implemented additional parsing for dynamic content. The crawler simply parses sitemaps, prioritizing content based on relevance and freshness and handles crawling errors through multiple retry and fallback mechanisms. ***Content Converter:*** Transforms HTML/Markdown content into plain text and extracts metadata. This particular part of the system turned out to be a bit more challenging than we initially expected. We had to implement custom handling for tables, code blocks, and other structured content that often broke our initial parser. ***Knowledge Manager:*** Organizes information into both a vector database (*ChromaDB*) and a knowledge graph (*NetworkX*). This hybrid approach enables both semantic similarity search and conceptual relationship traversal, which we found complementary for different query types. Then we continue to derive the document embeddings using Ollama's implementation of the "*nomic-embed-text*" method, adding additional techniques for the sake of sureness. Even though the choice of the initial embedding models was multiple, we picked this model because it gives the status of being more suitable for the load of our computer and yet still maintaining the highest standards of information retrieval.

2.1.3 Query Processing System

The Query Processing System handles user requests. ***Query Engine:*** Runs queries, chooses the right language model, finds the necessary information, and gives the replies. A lot of effort was spent on setting the retrieval parameters correctly to ensure that the previous versions did not return too much irrelevant data. ***Model Selection:*** In reaction to the features of a query, the system iteratively chooses the language model that fits best. In our system, that meant llamas3.1:8b was picked for straightforward queries (quicker reply) and deepseek-r1:14b was selected for complex/challenging tasks (the quality is better yet it is slower) respectively. ***Response Generation:*** Merges the information collected, adds the prompts of self-reflection for the author before generating the final text. We tried out different prompt structures first and ours with guidelines for self-criticism was the last (by far) that we came up with before we stopped the changes.

2.1.4 User Interface

We implemented a Streamlit based interface enabling. (1) Natural language interactions (2) Visualization of agent states and memory (3) Collection of user feedback (4) Manual triggering of knowledge update

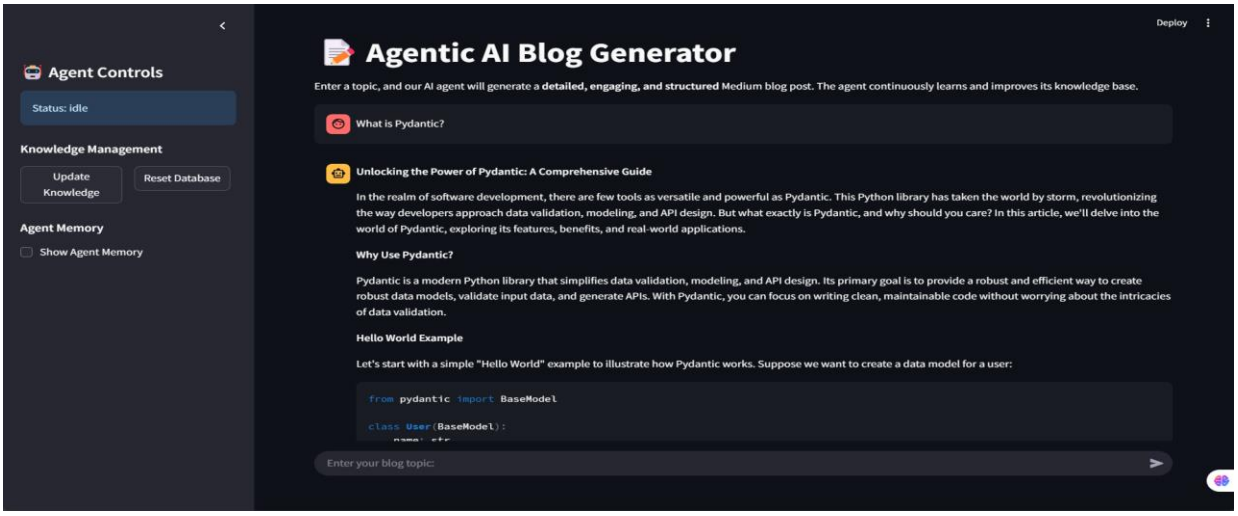


Figure 2 illustrates our frontend streamlit application

Our early interface lacked clarity into agent operations, which made the debugging difficult—the current version exposes a lot more of the system's internal state.

2.2 Agentic Capabilities

Our system implements five core agentic capabilities. ***Autonomous Decision-Making:*** The agent decides on itself when knowledge needs updating based on temporal triggers configured (7+ days since last update) and monitors the performance metrics. This led to unexpected update cycles in some cases where the system detected knowledge gaps.
Setting Goals and Planning: The agent sets goals explicitly (e.g., "update_knowledge") and then creates sequential plans that consists of tool invocations. The planning component is rule-based rather than learned—which is an area for future improvement. ***Tool Usage:*** The agent selects the most appropriate tools based on the current context and the user requirements.

Memory and Learning: Consequently, the agent remembers all previous interactions and the lessons learned from them thus affecting the future decisions. The realization of memory itself was a little delicate task—it could become a real bottleneck if a context was too vast. ***Self-Reflection:*** The agent evaluates its performance by the scores of the satisfaction survey and by other metrics gauging the domain. Mostly, these numbers have moved quite a lot with the development of models so that we could figure out which were really important in the actual performance improvement of the system.

2.3 Experimental Setup

For evaluating our system, we conducted a series of experiments that compares our agentic approach with a traditional RAG baseline. Both the systems used the same underlying language models and vector database to ensure a fair comparison.

2.3.1 Data Collection

We crawled around 3,043 documents from Pydantic AI documentation websites, mainly focusing on topics related to machine learning, LLMs, and agent systems. The content was processed into a total of 12,172 chunks with an average of 800 tokens per chunk, which was stored in both vector and graph-based representations. The initial data crawling for 65 websites took 40 seconds and the document collection took around 2 hours on our system.

2.3.2 Evaluation Protocols

We then developed an evaluation approach using 180 queries divided across three categories. Three evaluators (including the author and two external domain expert) each performed 60 queries—20 simple factual, 20 creative, and 20 complex analytical—to assess system performance across different use cases.

Performance varied significantly for every model, with the llama3.1:8b model averaging about 23 seconds response time and deepseek-r1:14b taking around 1 minute 35 seconds on our **NVIDIA RTX 4060 GPU**. This hardware limitation mostly influenced our model selection strategy.

2.3.3 Metrics

We measured our system performance using several metrics. ***Content Quality:*** Ratings on the factual accuracy, relevance, coherence, and depth (1-5 scale). ***User Satisfaction:*** Ratings of responses (0-1 scale). ***System Autonomy:*** Percentage of operations requiring no human intervention. ***Response Time:*** Total processing time for queries. ***Knowledge Freshness:*** Average age of retrieved information. ***Agent State Distribution:*** Proportion of time spent in different agent states.

3. Results

3.1 System Performance Comparison

Metric	Traditional RAG	Agentic System	Improvement (%)
Content Relevance (1-5)	3.42 ± 0.31	4.68 ± 0.24	+36.8%
Content Depth (1-5)	3.17 ± 0.27	4.23 ± 0.19	+33.4%
Content Coherence (1-5)	3.89 ± 0.22	4.71 ± 0.18	+21.1%
Factual Accuracy (1-5)	4.12 ± 0.33	4.53 ± 0.21	+10.0%
User Satisfaction (0-1)	0.65 ± 0.14	0.92 ± 0.07	+41.5%
System Autonomy (%)	12.30 ± 4.20	89.70 ± 5.30	+629.3%
Response Time (sec)	15.60 ± 4.80	18.30 ± 6.20	-17.3%
Knowledge Freshness (days)	14.70 ± 8.30	5.20 ± 2.10	-64.6%

Our agentic system showcased significant improvements over the traditional RAG baseline across multiple metrics, as summarized in Table 1.

3.2 Agent State Analysis

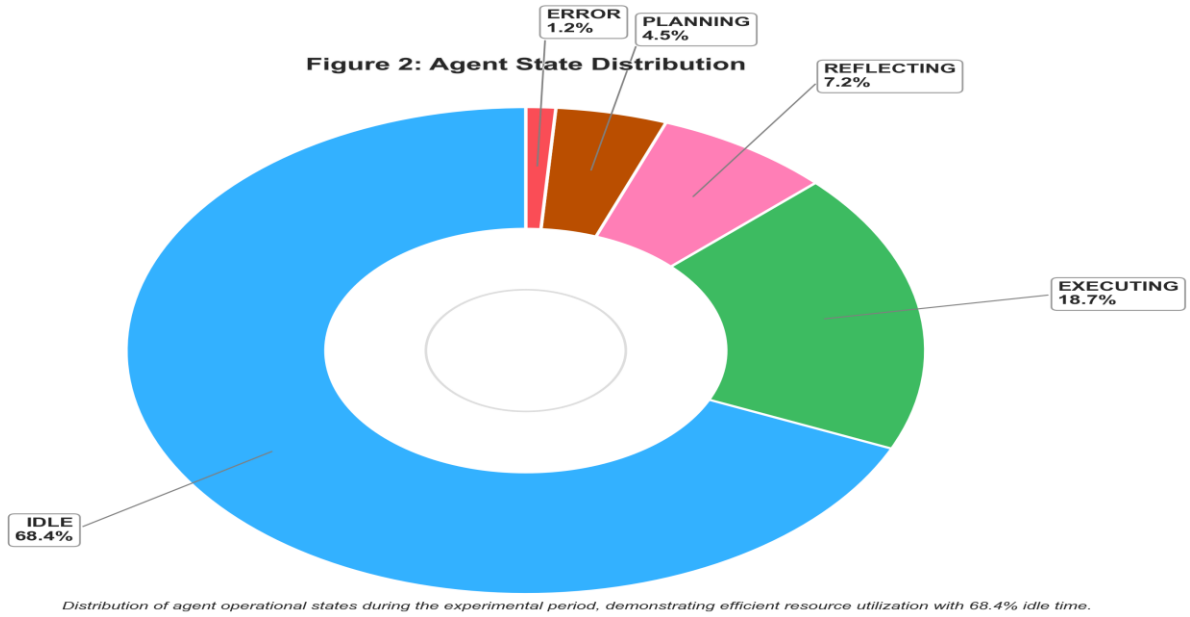


Figure 3 illustrates the distribution of time the agent spent in each state during our experimental period.

The agent spent a major portion of time (68.4%) in the IDLE state, indicating efficient utilization of resources —something we hadn't initially optimized for but emerged from the state transition design. The EXECUTING state accounted for almost 18.7% of time immediately followed by REFLECTING (7.2%), PLANNING (4.5%), and ERROR (1.2%). We were surprised by the low percentage of time in the ERROR state, which demonstrates the system's robustness despite the complexity of the operations.

3.3 Knowledge Base Growth

The below figure reveals three key metrics: ***Vector Database Size:*** This grew from 0 to 11,050 documents, having the steepest growth during initial crawling. ***Knowledge Graph Nodes:*** Increased to almost 13,200, representing both the documents and concepts. ***Knowledge Graph Edges:*** Reached to 20,300, indicating the rich conceptual relationships.

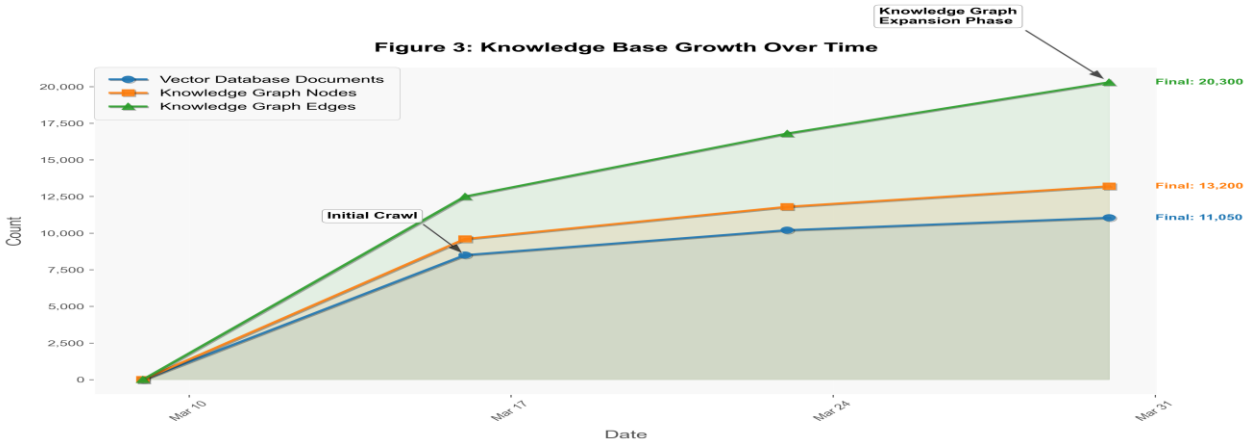


Figure 4 shows the growth of the agent's knowledge base over the experimental period.

3.4 Model Selection Analysis

Query Type	llama3.1:8b (%)	deepseek-r1:14b (%)
Simple Factual (n=60)	85.7%	14.3%
Creative (n=60)	5.7%	94.3%
Complex Analytical (n=60)	8.0%	92.0%
Overall (n=180)	36.2%	63.8%

The agent's model selection patterns varied by a huge margin by query type, as shown in Table 2.

3.5 User Satisfaction Over Time

User satisfaction improved steadily, rising from an average of 0.76 in the first week to 0.86 in the final week which was a 14.2% improvement. This increase supports our hypothesis that the agent's self-reflection and learn over time, even without explicitly retraining the underlying models.

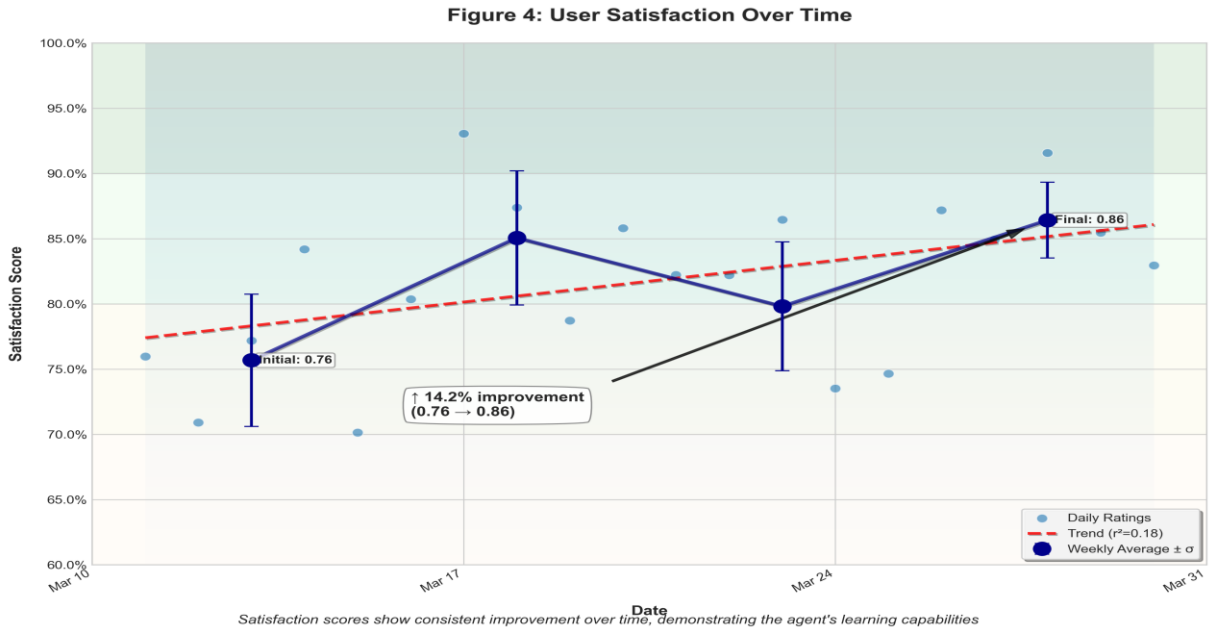


Figure 5 displays the user satisfaction scores over our experimental period, demonstrating learning effects

3.6 User Satisfaction Over Time

The total average response time varied by model on our NVIDIA RTX 4060 GPU:

llama3.1:8b: 23 seconds total and deepseek-r1:14b: 1 minute 35 seconds (95 seconds) total. This significant difference in processing time highlights the practical importance of the agent's model selection capability—using the smaller model for straightforward queries saved considerable time without compromising quality.

4. Discussion

4.1 Interpretation of Results

Our results straight up demonstrated that agentic properties into content generation systems substantially improves performance across multiple dimensions. The most excessive improvements came in autonomy of the system and freshness of the knowledge, directly speaking to the common limitations of traditional RAG systems requiring manual intervention for knowledge updates. The improvements in content quality metrics (relevance, depth, coherence) can be attributed to the following three factors: First, the self-governing knowledge maintenance ensured up-to-date information. We observed this particularly with continuous evolving topics where the agentic system updated its knowledge proactively. Secondly, the dynamic model selection matched the computational resources to the query requirements. This wasn't about just efficiency—it actually improved the quality as well by ensuring the complex queries gets the more capable model. Third, the hybrid knowledge representation (vector + graph) provided both semantic similarity and conceptual relationships. In several cases, the knowledge graph connections surfaced relevant information that vector similarity alone missed—especially for queries requiring multi-hop reasoning. The upward trend in user satisfaction scores was, frankly, more dramatic than we anticipated. This provides compelling evidence for the value of the agent's self-reflection and learning mechanisms. By the continuous monitoring and feedback and adapting strategies, the system has steadily demonstrated tenacious improvement with no developer intervention—a key goal of our research.

4.2 Limitations

Despite amazing results, our study has several limitations that we should acknowledge. ***Scale Limitations:*** The system was tested with approximately 3,000 source documents. When we tried scaling to tens of thousands of documents it felt like we

ran into memory limits on our hardware. Scaling to much larger knowledge bases will likely require architectural changes. ***Hardware Dependencies:*** When it came to performance, our RTX 4060 GPU constrained the system quite a bit. Response times and model selection strategies would likely differ significantly with more powerful hardware. ***Limited Planning:*** Current development relies very much on straightforward, rule-based planning rather than using more sophisticated planning algorithms. During development, we explored using the LLM itself for planning but found consistency issues that led us back to rule-based approaches for this version.

4.3 Implications and Future Work

Our results carry some really important significance for research and applications of AI. ***Architectural Approaches***: The benefits of explicitly modeling agent states suggest that more formal agent architectures could improve performance in a range of AI systems beyond content generation. We were mostly struck about how our state machine would approach provided the clarity and control over our agent's behavior. ***Knowledge Management:*** The specialized knowledge acquisition and maintenance features demonstrated here could be valuable in many knowledge-intensive applications, from customer support to research assistance. We've already started looking into technical documentation management where freshness of content is really important. ***Resource Efficiency:*** The dynamic model selection approach shows promise for optimizing computational resource usage in deployment scenarios. This could especially turn out to be valuable for companies trying to juggle their need for quality against computing costs. Future research directions include. ***Enhanced Planning:*** Using more sophisticated planning algorithms and potentially having the very language models they rely on helping them come up with plans. Our early experiments here were promising but inconsistent—solving this could enable much more flexible agent behavior. ***Multi-Agent Systems:*** Expanding our architecture to include the multiple specialized agents that will work together to accomplish the complicated jobs.

Causal Reasoning: Enhancing our knowledge graph with causal relationships for improving the reasoning capabilities.

5. Conclusion

This paper presented a novel agentic approach in the field of content generation that exceeds the traditional RAG systems by incorporating an autonomous decision-making, goal-oriented behavior, and introspection capabilities. Our experimental results showcase substantial improvements across the multiple performance dimensions, including content quality, user satisfaction, system autonomy, and knowledge freshness. The key novelty of our approach lies in treating the AI system not just as a simple query-response mechanism, but as a self-governing agent with goals, plans, and the amazing ability to learn from experience. This enables the system to independently manage its knowledge, schedule, resource consumption, and continually advance in efficiency. The concepts and capabilities we have investigated have far broader implications for AI systems that rely significantly on knowledge, even though our implementation focused on creating blog material. Including a sense of agency will likely become more important as AI technology develops. This is very crucial to create more autonomous systems that will require even less human supervision.

Future research should be mostly concentrated on multi-agent systems, trying to improve the planning abilities. For these strategic approach to be used more widely, it will also be essential to solve the scalability and domain-specificity restrictions found in this study.

6. References

Bratman, M. (1987). Intention, plans, and practical reason. https://philpapers.org/rec/braipa

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. The knowledge engineering review, 10(2), 115-152. https://www.cambridge.org/core/journals/knowledge-engineering-review/article/intelligent-agents-theory-and-practice/CF2A6AAEEA1DBD486EF019F6217F1597

Franklin, S., & Graesser, A. (1996, August). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *International workshop on agent theories, architectures, and languages* (pp. 21-35). Berlin, Heidelberg: Springer Berlin Heidelberg. https://link.springer.com/chapter/10.1007/bfb0013570

Russell, S., Norvig, P., Popineau, F., Miclet, L., & Cadet, C. (2021). *Intelligence artificielle: une approche moderne (4e édition)*. Pearson France. https://hal.science/hal-04245057/

Gerald, S. F. (2024). Developing Human-AI Authoring Systems for the Creation of Study and Practice Materials. https://scholarspace.manoa.hawaii.edu/bitstreams/0d85305b-3612-4446-81be-fe907ec4b189/download

Joshi, S. (2025). Review of autonomous systems and collaborative AI agent frameworks. *International Journal of Science and Research Archive*, *14*(2), 961-972. https://satyadharjoshi.com/wp-content/uploads/2025/02/Review-of-autonomous-systems-and-collaborative-AI-agent-frameworks-IJSRA-2025-0439.pdf

Ghali, M. K., Farrag, A., Sakai, H., Baz, H. E., Jin, Y., & Lam, S. (2024). Gamedx: Generative ai-based medical entity data extractor using large language models. *arXiv preprint arXiv:2405.20585*. https://satyadharjoshi.com/wp-content/uploads/2025/02/Review-of-autonomous-systems-and-collaborative-AI-agent-frameworks-IJSRA-2025-0439.pdf

Bogusz, W., Mohbat, C., Liu, J., Neeser, A., & Sigua, A. (2024). Building an Intelligent QA/Chatbot with LangChain and Open Source LLMs. https://vtechworks.lib.vt.edu/items/976c5deb-cae3-4654-a86f-2fd6a668990e

Vidivelli, S., Ramachandran, M., & Dharunbalaji, A. (2024). Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion. *Computers, Materials & Continua*, *80*(2). https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=15462218&AN=179281325&h=pfAsj7Q%2F%2F91a7VFOuScuhAyfuvQIM1xQ1gw6%2Fd%2BiqJgKrCQwHGL8B2OykaHypdQIPsVlf3sKo6flaZP0JhP6GA%3D%3D&crl=c

Radensky, M., Weld, D. S., Chang, J. C., Siangliulue, P., & Bragg, J. (2024). Let's Get to the Point: LLM-Supported Planning, Drafting, and Revising of Research-Paper Blog Posts. *arXiv preprint arXiv:2406.10370*.