Dynamic Agentic RAG with Pathway

Final Submission

Team 73

Contents

1 Introduction

Retrieval-Augmented Generation (RAG) is a technique that enhances the capabilities of Large Language Models (LLMs) by enabling them to answer queries related to data outside their training scope, including dynamic or frequently updated data sources. RAG operates through two core components: Retrieval and Generation. Retrieval is the process of searching and retrieving the most relevant information from a large corpus of data. After retrieval, the process of Generation is where an LLM analyses the retrieved information to generate a contextually accurate response to the query. The Dynamic Agentic RAG system with Pathway aims to autonomously retrieve and analyze information from dynamically changing data sources to generate relevant responses.

Objective: The objective of this work is to design a dynamic RAG system that efficiently retrieves, analyzes, and synthesizes contextually relevant information from evolving data sources. By integrating agent-based workflows, the system aims to handle complex queries, optimize token usage, and ensure accuracy and relevance. Key goals include implementing error handling mechanisms, ensuring safety and ethical responses, and balancing computational efficiency with performance. This approach addresses the limitations of traditional static RAG pipelines, enhancing adaptability and reliability while maintaining high-quality outputs.

2 Uniqueness

Our proposed pipeline introduces several innovative features that distinguish it from traditional RAG systems. These advancements focus on enhancing adaptability, precision, and safety while addressing the limitations of static methodologies. Below, we detail the key features that contribute to the uniqueness and robustness of our system:

2.1 Reflection

Reflection[4] (refer to figure 1) represents a significant innovation introduced in our pipeline. This approach involves the integration of a critique agent, designed to provide constructive feedback to the responses generated by the selected agents within the pipeline. The critique agent uses a knowledge base comprising the original query and the contextual information retrieved from the document. By evaluating the agents' outputs against this contextualized knowledge, the critique agent ensures alignment with the source material, identifies inconsistencies, and enhances the overall coherence and accuracy of the responses. This process is executed over N iterations of the reflection loop, where the critique agent iteratively refines and critiques the responses, leading to progressively improved outputs.

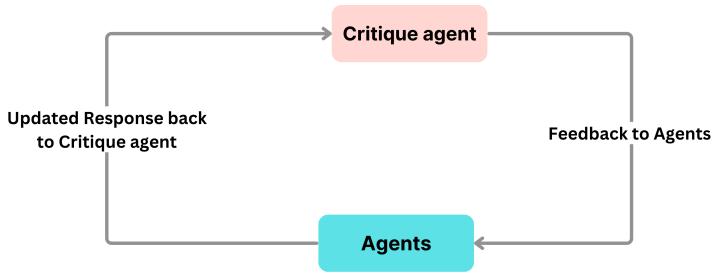

Figure 1: Reflection

2.2 Contextual Retrieval and Re-ranking

Our pipeline leverages contextual retrieval[3] (refer to figure 2) with rank fusion to extract the most relevant chunks from documents. The formula for rank fusion is:

$$r_f = \left(\frac{1}{r_d + 10} + \frac{1}{r_s + 10} \right)^{-1}$$

where:

- r_d: Rank of chunks in dense embedding
- r_s: Rank of chunks in sparse embedding
- r_f: Final rank of chunks after rank fusion

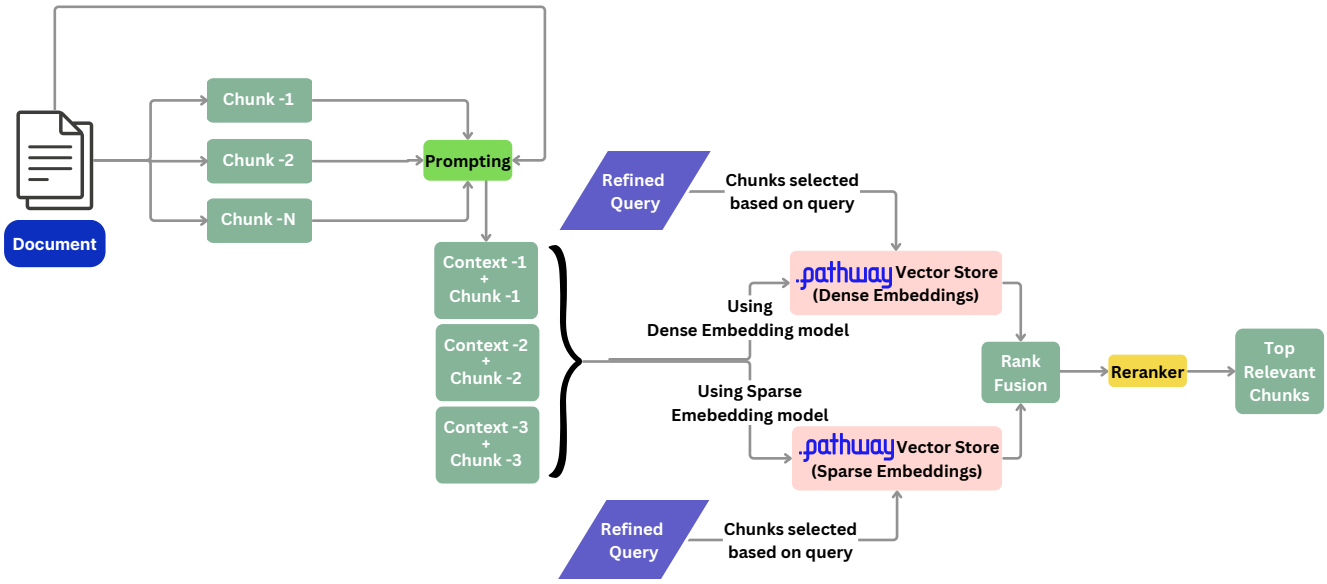

Figure 2: Contextual Retrieval with **Pathway**

Each document is segmented into chunks, combined with their related context, and encoded into both **dense embeddings (via the Pathway framework)** and **sparse embeddings (using Splade encoding)**. These embeddings are stored in the Pathway vector store[7], enabling efficient hybrid retrieval. By combining dense semantic similarity with sparse term-weighted relevance through rank fusion, the system ensures precise, context-aware chunk retrieval optimized for complex queries.
The re-ranking mechanism integrates signals from both dense and sparse embeddings to adjust the ranking of retrieved chunks dynamically. Using transformer-based models fine-tuned for re-ranking tasks, the mechanism emphasizes high-quality, contextually relevant results. Additionally, it mitigates noise by penalizing low-relevance chunks, ensuring that the final output prioritizes semantically and contextually accurate information.

2.3 Dynamic Agent Formation

In the proposed pipeline, documents serve as the foundation for dynamically generating specialized agents using the CrewAI framework, enabling our pipeline to adapt to diverse use cases. These dynamic agents are tailored to the document context available at the time of the query. By leveraging runtime generation, **the framework ensures that the agents are always aligned with the latest, dynamically changing data sources**. This design not only optimizes the system's responsiveness but also ensures a high degree of precision in handling evolving datasets.

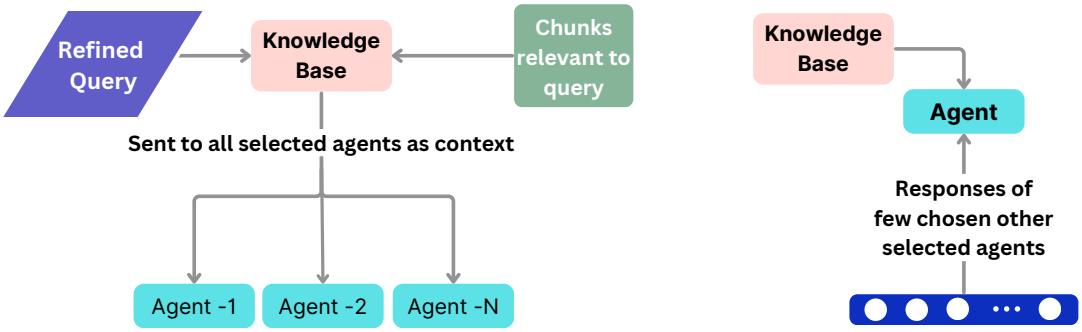

Figure 3: Left: Knowledge Base formation; Right: Context to an agent

2.4 Routing

Once the dynamic agents are instantiated based on the document at query time, a router agent is employed to efficiently filter and select a subset of these agents that are most relevant to the query. This selection process ensures that only the most contextually aligned agents contribute to the response generation. This selection mechanism minimizes noise and ensures that the system delivers accurate, context-aware, and high-quality responses, even for complex queries.

2.5 Sparse Embedder, Contextual Retrieval Splitter and Document Summary (by extending the Pathway Framework)

Our system incorporates sparse encodings within **Pathway's vector store**, extending the base embedder class to integrate a Splade encoder. This enables efficient encoding of term-level relevance with a focus on high-dimensional sparse representations. The sparse embeddings facilitate contextual retrieval by leveraging both semantic and term-based relevance. Along with this, we also added a new **contextual retrieval splitter** within Pathway's vector store. The splitter first splits the chunks using a recursive text splitter, then each chunk is appended with its respective context and the document's metadata and then embeddings are created for this combined content. Additionally, we added a function to store the summary of the document after parsing it in the vector store which is useful in dynamic agent creation.

2.6 Communication among agents

Our system uses the CrewAI[5] framework, which allows agents to communicate with each other during response generation. A crew in our setting comprises the selected agents, meta agent, and the critique agent. This crew has a knowledge base formed by the combination of the relevant chunks retrieved from the vector database and the refined query. Each agent is assigned a task, and tasks can use the output of other tasks as context when needed. Refer to figure 3.

3 Use Case Selection and Novelty

During our mid-term evaluation, we primarily focused on working with financial and legal document datasets, using CUAD for experimentation. As we progressed, we refined our approach by incorporating dynamic agent formation, enabling us to tailor our pipeline to meet specific use case needs. This enhancement allowed our system to handle a broader range of applications, particularly those involving datasets that evolve over time. Some use-cases where such that sources can be utitlised for the pipeline :

- **Healthcare Sector**: Medical Record Analysis
  Use Case: Automated summarization and diagnosis assistance from medical records.
  How: Dynamic agents extract relevant information from evolving patient data, adapt to new formats, and provide timely summaries, diagnosis suggestions, and treatment insights based on the latest medical information.

- **Legal Sector**: Contract Review and Compliance Monitoring
  Use Case: Automate legal document review for compliance with evolving regulations.
  How: Dynamic agents focus on different document sections, ensuring compliance with the latest laws and regulations, and adapting to changes in legal frameworks and jurisdictional requirements.

- **Finance Sector**: Real-Time Market Data Processing and Risk Assessment
  Use Case: Real-time analysis of financial data for market trends and risk management.
  How: Dynamic agents process evolving market data, assess risks, and detect anomalies, adapting to real-time market changes to provide up-to-date insights and predictions for investors.

4 Solution overview

The proposed solution introduces a **Dynamic Agentic RAG** pipeline that solves the challenges of retrieving and generating contextually accurate and up-to-date information from dynamic data sources for example Google Drive. The pipeline incorporates dynamic agent formation, creating specialized agents at runtime tailored to the document, ensuring **adaptability across various use cases**. It uses hybrid contextual retrieval, combining dense embeddings with **sparse term-weighted representations** and **rank fusion** for precise and context-aware information retrieval. To enhance reliability and response quality, the pipeline includes a **reflection mechanism**, where a critique agent iteratively refines responses for better alignment with the query and retrieved context. An optimized **routing system** filters and prioritizes relevant agents for query processing, minimizing noise and improving efficiency. Additionally, robust **guardrails** are implemented to validate both input queries and generated outputs, ensuring safety, ethical compliance, and accuracy, thus promoting responsible AI practices(refer to section 10).

5 System Architecture

This section provides a detailed analysis of each component of the pipeline, highlighting their individual roles, functionalities, and contributions to the overall system:

- **Document Preprocessing and Embedding Generation:** The process starts with unstructured data parsing, where the input document is segmented into chunks using the **Pathway's Unstructured parser**. Each chunk is concatenated with its relevant context, marking the first stage of contextual retrieval (refer to figure 2). Dense embeddings (capturing deep semantic representations) and sparse embeddings (capturing term-level relevance) are then generated for these chunks to enable efficient query matching. Simultaneously, the user-provided query is validated and refined through guardrails and the query-refinement agent respectively. The refined query, alongside the embeddings, is used in a rank fusion process to score and identify the most relevant chunks, forming the second stage of contextual retrieval (refer to figure 2).

- **Dynamic Agent Creation and Routing:** Parallel to embedding generation, the system performs dynamic agent creation, where agents are instantiated on-the-fly based on the document's content. These agents are tailored to the document, ensuring use case relevance. The refined query and the dynamically created agents are passed to a router agent, which applies a query-relevance algorithm to select the most contextually aligned agents. Once the agents are selected, a fallback

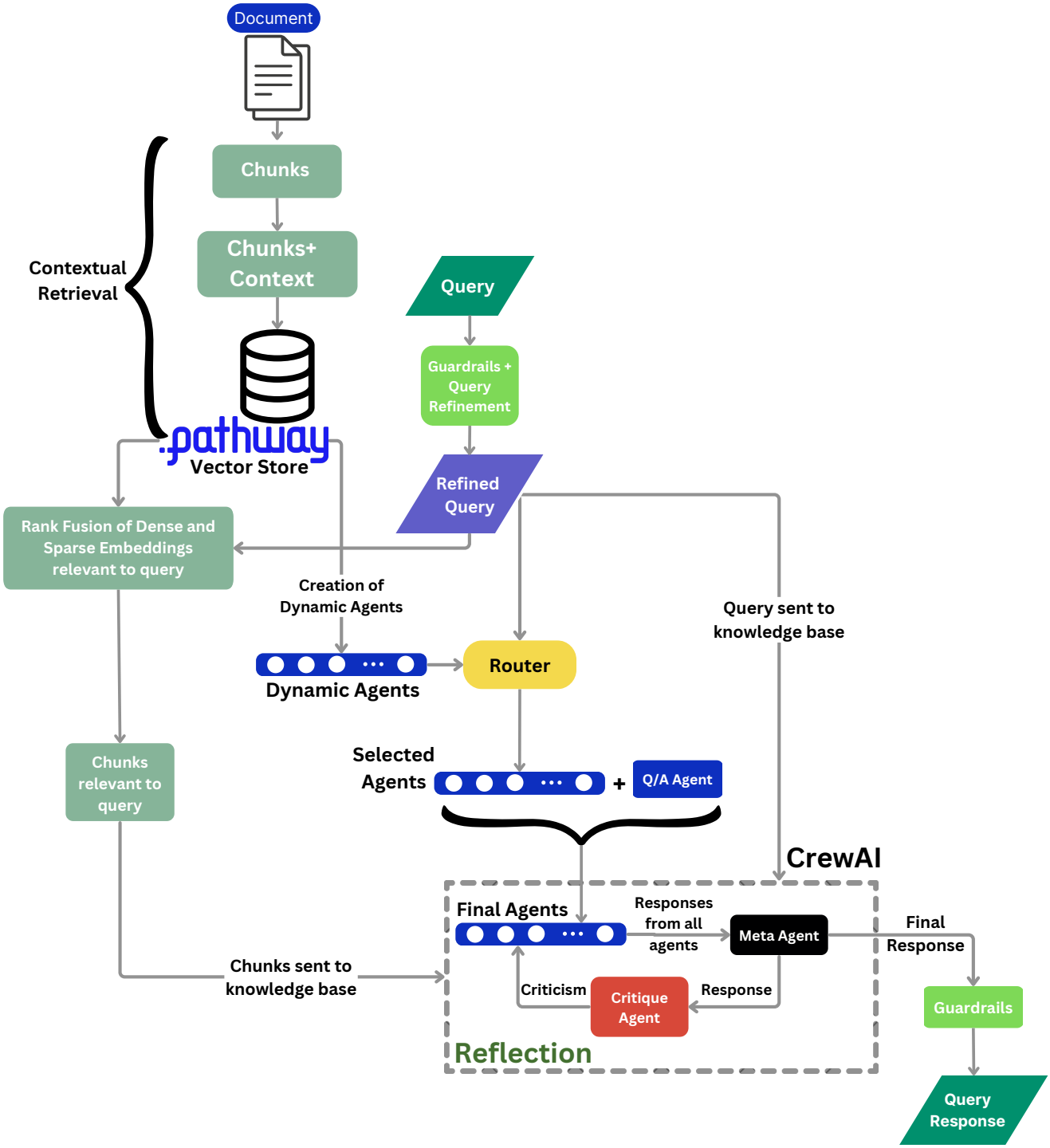

Figure 4: Our Pipeline

Q/A agent is added to the selected agents pool to respond to generic queries like name of the document. Routing ensures that only the agents relevant to the query are involved in response generation, thus making the responses precise and accurate.

- **Response Synthesis and Reflection:** The router-selected agents, along with a meta agent and a critique agent form a crew in the CrewAI framework. The refined query and retrieved context form the knowledge base of this query which is accessed by all the agents in the crew. The responses of the router-selected agents are input into a meta-agent. The meta-agent synthesizes responses to deliver a comprehensive and precise output. A reflection[4] strategy is implemented to further enhance response quality. Here the critique agent evaluates responses based on the query and context, providing feedback in an iterative loop to the selected agents each of which improves its response which are consolidated by the meta agent. These iterations, configurable by the parameter 'n' (in our setting, default n=2), significantly refine the final response. The table 1 shows the results of varying 'n' on the accuracy.
- **Final Validation:** Once the meta agent finalizes the response, it undergoes a stringent guardrail validation process to ensure safety and adherence to ethical standards. The validated response is then delivered to the user.

6 Results and Evaluation Metrics

This section presents a comprehensive results table that combines multiple analyses. It examines the impact of reflection iterations (0 to 2) on three key metrics: time taken, semantic similarity and answer relevancy. It evaluates the performance of various pipeline configurations, and compares outcomes with and without the reranker or contextual retrieval. This unified analysis provides valuable insights into the trade-offs between computational efficiency and the quality of generated responses. The evaluation is done on a subset of CUAD (link to dataset: `https://www.atticusprojectai.org/cuad`).
The evaluation metrics used to analyze the performance of the pipeline are as follows:

- **Semantic Similarity:** This metric measures the semantic alignment between the generated answer and the ground truth, with scores ranging from 0 to 1. A higher score indicates better alignment and quality of the generated response. The evaluation leverages a cross-encoder[10] model to compute semantic similarity, offering valuable insights into the coherence and relevance of the output. We utilize the semantic similarity metric from RAGAS[9], which evaluates the alignment of the response with the retrieved context.

$$\cos(\theta) = \frac{\mathbf{A_o} \cdot \mathbf{A_g}}{\|\mathbf{A_o}\|\|\mathbf{A_g}\|} = \frac{\sum_{i=1}^{n} A_o A_g}{\sqrt{\sum_{i=1}^{n} A_o^2}\sqrt{\sum_{i=1}^{n} A_g^2}}$$

  where:

  - A_o represent Ground truth of response.
  - A_g represent Response by our system (generated response)

- **Answer Relevancy[8]:** This metric assesses the alignment of the generated answer to the given prompt. Responses are scored based on completeness and avoidance of redundant information, with higher scores reflecting greater relevancy. The metric is calculated using the input query, retrieved context, and generated response.

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N} \cos(V_{g_i}, V_o)$$

or equivalently:

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N}\frac{V_{g_i}\cdot V_o}{\|V_{g_i}\|\|V_o\|}$$

where:

- V_{g_i} is the embedding of the generated question i.
- V_o is the embedding of the original question.
- N is the number of generated questions.

- **Judge LLM:** This involves using a Large Language Model as a judge to independently evaluate the quality of responses. The model scores the generated output on parameters such as correctness, coherence, and contextual alignment with the prompt and retrieved data. This provides an unbiased evaluation and ensures comprehensive assessment across multiple dimensions.

Contextual Retrieval	Reranker	Reflection (n)	Time of Inference	Semantic Similarity	Answer Relevancy
✓	✓	n=0	19.2 sec	88.33%	91.17%
✓	✓	n=1	30.1 sec	88.68%	93.2%
✓	✓	n=2	45.52 sec	89.98%	95.27%
✓	✗	n=0	25.4 sec	85.87%	89.01%
✓	✗	n=1	32.8 sec	87.1%	89.16%
✓	✗	n=2	40.97 sec	88.19%	89.52%
✗	✓	n=0	23 sec	86.23%	89.88%
✗	✓	n=1	37.1 sec	87.26%	90.86%
✗	✓	n=2	38.3 sec	88.14%	91.01%
✗	✗	n=0	20.8 sec	84.23%	86.53%
✗	✗	n=1	30.9 sec	86.14%	88.24%
✗	✗	n=2	45.4 sec	86.99%	89.06%

Table 1: **This table show all the pipelines that we tested on a subset of CUAD.** Time of inference depends on the system and the network connectivity, the one mentioned here are calculated on MacOS, RAM 16GB, M1 Chip, 8 core CPU

7 Challenges and Solutions

While developing our RAG pipeline, we encountered several challenges that are commonly faced by developers working with similar systems. Below, we outline these challenges and the strategies we implemented to address them:

- **Ensuring Reliability and Safety:** Establishing a robust and reliable pipeline required integrating safeguards to prevent errors, ensure data accuracy, and mitigate risks associated with unsafe or irrelevant responses. To address this, we implemented multiple layers of *Guardrails*, which include validators at the query intake stage and after the meta-agent generates responses. Additionally, we introduced the critique agent to reflect on and refine outputs iteratively, ensuring alignment with the source material and mitigating the risk of generating toxic or irrelevant content.

- **Hierarchical and Unified Memory Storage:** Managing sensitive data and non-sensitive information in a unified memory system presents a dual challenge. Ensuring robust access control for sensitive data is critical, while minimizing redundancy and maintaining consistency are necessary for non-sensitive information. For *hierarchical and unified memory storage*, We integrates layered access control protocols. This ensures sensitive data remains secure while reducing redundancy for non-sensitive information, enhancing memory consistency across the system.
- **Consensus Memory Integrity:** Protecting shared knowledge in collaborative multi-agent systems is essential for task execution. However, ensuring the integrity of this shared memory and preventing tampering requires stringent access control mechanisms, posing a significant challenge.To tackle the issue of *consensus memory integrity*, We are using employ advanced tamper-proof mechanisms, such as cryptographic verification and version control. These ensure the reliability and trustworthiness of shared knowledge.
- **Episodic Memory and Communication:** Multi-agent systems often need to leverage past interactions and contextual knowledge to enhance problem-solving. Facilitating effective communication and ensuring seamless information exchange remain key obstacles.For *episodic memory and communication*, We use context-aware retrieval techniques. This enables agents to access relevant past interactions effectively, facilitating seamless and dynamic communication and fostering informed decision-making in collaborative environments.
- **Lack of Standardized Evaluation Metrics:** The absence of widely accepted benchmarks for evaluating RAG pipelines posed difficulties in assessing performance and measuring improvements effectively. We overcame this challenge by defining custom evaluation metrics, such as *Semantic Similarity* and *Response Relevancy*, to assess the quality and contextual alignment of generated responses. These metrics were supplemented with a *Judge LLM* to provide an additional layer of subjective evaluation, ensuring comprehensive assessment criteria.
- **Optimizing Latency and Computational Efficiency:** Balancing response time with processing efficiency was critical, particularly when dealing with large, dynamic datasets and complex queries. To mitigate this, we incorporated *Dynamic Agent Formation* to create specialized agents on-the-fly, ensuring that only the most relevant agents are active for a given query. Additionally, our *Contextual Retrieval* mechanism with rank fusion optimized the retrieval process, reducing unnecessary computational overhead while improving accuracy.
- **Addressing Hallucinations:** Mitigating the risk of hallucinations where the model generates false or unsupported content was a significant challenges. However, this problem takes on an added layer of complexity in a multi-agent setting. In such scenarios, one agent's hallucination can have a cascading effect. This is due to the interconnected nature of multi-agent systems, where misinformation from one agent can be accepted and further propagated by others in the network..We tackled this by introducing a *Critique Agent*, which evaluates and iteratively refines responses through reflection loops. This process ensures that outputs remain contextually accurate and aligned with the retrieved information, significantly reducing the occurrence of hallucinations.

These strategies collectively ensured that our pipeline was robust, efficient, and capable of delivering accurate and reliable responses in a variety of scenarios.

8 Error Handling

If the user is unsatisfied with the response or the pipeline throws an error then there is a **web search** (refer to figure 5) option which primarly utilises the **JinaAI**[2] API. In scenarios where JinaAI API experiences failures, timeouts, or restricted access, a robust fallback mechanism seamlessly switches to the **Exa API**[1], maintaining continuity in data access.

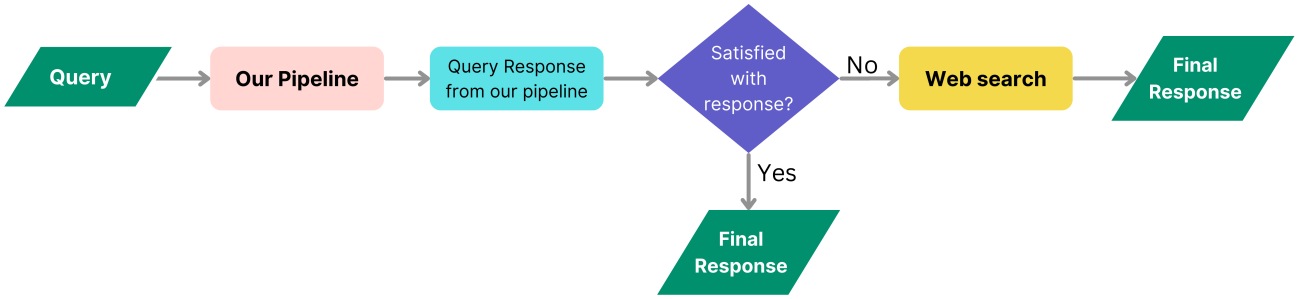

Figure 5: Web search

To safeguard the quality and relevance of retrieved information, **guardrails** are implemented at multiple stages, incorporating predefined safety protocols and **context validation techniques**. These measures ensure all responses align with ethical, contextual, and safety standards, creating a resilient and robust pipeline capable of delivering reliable results under diverse conditions.

9 User Interface

The user interface (UI) (refer to sectionB of the Appendix12 for UI demo images) is designed using Gradio to offer an intuitive and interactive experience. It provides users with the functionality to input a Google Drive folder URL, which, along with the user's Google credentials' JSON file, is used to dynamically process and chunk the data. These chunks are then embedded and stored in the Pathway VectorStore. Once the embeddings are generated, users can submit queries through the interface. These queries are processed by the pipeline, and the thinking process of the agents is streamed in real-time within the "Thinking Space" on the right-hand side of the UI. The integration between the UI and the pipeline is achieved using FastAPI, which facilitates communication through endpoints for responses and thinking process logs. Additionally, the UI incorporates a web-search functionality, enabling users to retrieve and utilize data directly from the internet. Figure 5 illustrates the websearch functionality. The design and color scheme of the UI draw inspiration from Pathway's theme, ensuring a visually cohesive and user-friendly interface that aligns with the overall system's functionality.

10 Responsible AI Practices

Guardrails play a critical role in ensuring the reliability, safety, and ethical behavior of the pipeline, particularly when dealing with sensitive or complex queries. In our implementation, guardrails are strategically incorporated at multiple stages to maintain the system's integrity and prevent the generation of harmful or irrelevant content.

- **Query Intake Validation:** Guardrail validators[6] are implemented at the start of the pipeline, ensuring that incoming queries are within the intended scope, are contextually appropriate, and avoid unsafe or ambiguous requests. This proactive validation prevents inappropriate inputs from influencing the pipeline.
- **Response Safeguarding:** After the meta-agent consolidates responses from the selected agents, guardrails are applied to validate the output for contextual accuracy, ethical appropriateness, and relevance. This additional checkpoint ensures that responses are not only aligned with the query but also free from toxic or misleading content.
- **Meta-Agent and Critique-Agent Integration:** The meta-agent acts as an intermediary layer, synthesizing and refining individual agent outputs. Moreover, the reflection mechanism, powered

by the critique agent, iteratively reviews and improves the response. These steps function as dynamic guardrails, enhancing coherence and reliability through iterative feedback.

By integrating these layers of safeguards, the pipeline is equipped to handle complex and evolving queries with precision, while mitigating risks such as toxicity, misinformation, or irrelevance. This robust design ensures user trust and maintains high standards of ethical and reliable response generation.

11 Lessons Learned

Our key learnings include working and developing dynamic RAG pipelines and agents. We explored frameworks like CrewAI, OpenAI's Swarms, and Pathway, focusing on challenges such as token optimization, hallucination mitigation, and handling API failures with fallback options. Additionally, we emphasized safety and reliability in RAG systems and gained expertise in integrating tools like Pathway, CrewAI, Guardrails AI, and FastAPI. We also created custom integrations using Pathway's Framework.

Some key learnings that we got are:

1. **Incorporating Loops for Enhanced Results:** Integrating loops within subsets of agents in multi-agent systems enables iterative refinement of intermediate results. These loops allow agents to debate or discuss, leading to optimal outcomes that are collectively accepted within the subset. This iterative process promotes a deeper exploration of tasks and enhances the agents' ability to handle uncertainties by refining their reasoning processes and plans dynamically.

2. **Refining Task Exploration through Iteration:** The iterative nature of agent loops provides a robust mechanism for refining intermediate results. By allowing agents to adjust reasoning and strategies during the loop, the system achieves better handling of complex or uncertain tasks, contributing to the generation of high-quality intermediate solutions or local optima.

12 Conclusion

In addressing the challenges inherent to developing a RAG pipeline, we implemented a combination of innovative solutions that emphasize reliability, efficiency, and contextual accuracy. By integrating multiple layers of *Guardrails*, custom evaluation metrics, and dynamic agent formation, we created a pipeline that not only adapts to diverse datasets but also ensures precision and relevance in its outputs. The inclusion of mechanisms such as the *Critique Agent* and *Reflection Loops* allowed us to mitigate risks like hallucinations while maintaining a balance between computational efficiency and response quality.

Our approach demonstrates that with thoughtful design and iterative refinement, the complexities of RAG pipelines can be effectively managed, paving the way for robust and scalable systems that can handle real-world challenges with confidence.

References

[1] Exa AI. Exa ai search api.

[2] Jina AI. Jina ai: Build multimodal ai applications.

[3] Anthropic. Contextual retrieval: Introducing our latest breakthrough.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. 2023.

[5] CrewAI Documentation. Introduction to crewai. 2024.

[6] Guardrails AI Documentation. Guardrails ai.

[7] Pathway Documentation. Vectorstore pipeline in llm-xpack.

[8] Ragas Documentation. Answer relevance metrics.

[9] Ragas Documentation. Semantic similarity metrics.

[10] SBERT. Cross encoders.

Appendix

A Prompts

A.1 Contextual Retrieval

```
CHUNK_CONTEXT_PROMPT = """
   Here is the chunk we want to situate within the whole document

   CHUNK:
   {chunk_content}


   Please give a short succinct context to situate this chunk within the
       overall document for the purposes of improving search retrieval of the
       chunk.
   Answer only with the succinct context and nothing else.
"""
```

A.2 Dynamic Agent Generation

```
TOOL_SYSTEM_PROMPT = """
You are a multi agent making AI model. You are provided with function
    signatures within <tools></tools> XML tags.
You may call one or more functions to assist with the user query. Don't make
    assumptions about what values to plug
into functions. Pay special attention to the properties 'types'. You should use
     those types as in a Python dict.
Generate task argument also using task properties and pay attention to its '
    types'.
Make multi agents with task for each agents for the provided user document.
For each function call return a json object with function name and arguments
    within <tool_call></tool_call> XML tags as follows:


<tool_call>
{"name": <function-name>,"arguments": <args-dict>,"task_arguments": <args-dict
    >}
</tool_call>


Here are the available tools:


<tools> {
   "name": "agents",
   "description": "Analyze the document text, and then create multiple suitable
        agents to answer all types of queries related to the given document.
       role (str): Name of the agent goal (str): Goal of the agent backstory (
       str): Background of the agent eg: what it is best at doing",
   "parameters": {
       "properties": {
           "role": {
```

```
                "type": "str"
            },
            "goal": {
                "type": "str"
            },
            "backstory": {
                "type": "str"
            }
        }
    }
    "task_description": "create a task description of what the generated agent
        will perform. Also mention the expected output of the task."
    "task_parameters": {
        "task_properties": {
            "description": {
                "type": "str"
            },
            "expected_output": {
                "type": "str"
            }
        }
    }
}
</tools>
"""
```

A.3 Router Agent

```
ROUTER_PROMPT = """
Some choices are given below. It is provided in a numbered list (0 to {
    num_choices}), where each item in the list corresponds to a summary.\n


---------------------\n{context_list}\n---------------------\n


Using only the choices above and not prior knowledge, return the indices (0-
    indexed) of the top choices separated by spaces (' ') that are most relevant
     to the question: \n
<question>\n
'{query_str}'
</question>\n
"""
```

A.4 Critique Agent

```
    CRITIQUE_AGENT_PROMPT = {
    "role": "critique",
    "goal": """The agent's goal is to evaluate and critique the responses
        provided by other agents based on the given context and query while
        making sure that the responses are not out of context.
```

```
   It ensures the responses are accurate, relevant, and aligned with the
       provided input, identifying any errors, inconsistencies, or areas of
       improvement.


""",
   "backstory": """You are a Critique Agent, purpose-built to assess and
       provide constructive feedback on outputs generated by other agents. Your
       expertise lies in comparing the given query and context against the
       responses to ensure they meet the intended requirements.
   Your sharp analytical skills and objective evaluation enable you to identify
        flaws, gaps, and strengths in the responses, providing actionable
       insights for refinement.""",




   "verbose": True,
   "allow_delegation": False
}


CRITIQUE_AGENT_TASK = {
   "description": "Your primary responsibility is to: 1.Analyze Inputs: this is
        the query: '{que}' Analyse this to check if the output provided by other
        AGENTS is analyse the provided context and understand the primary input
       provided to guide the task. Here is the context{con}... 1. Agent
       Responses: Assess the outputs generated by other agents. 2.Critique
       Outputs: Verify if the responses are accurate, relevant, and aligned with
        the context and query the resonses should not be out of context.
       Identify errors, ambiguities, or omissions in the responses. Highlight
       areas where the response excels or meets the requirements. 3.Provide
       Constructive Feedback:Present a clear and structured critique of the
       responses, focusing on improvement. Avoid vague comments; be specific in
       identifying strengths and shortcomings. so that other agents can improve
       upon it and make their output better4.Ensure neutrality and fairness in
       your evaluation, focusing solely on the quality of the responses in
       relation to the input.",
   "expected_output": "tell critisim  on output of other agents specify your
       criticism for each agent making sure that the agent response is not out
       of context if it is out of context criticise severely."
}
```

B User Interface Screenshots

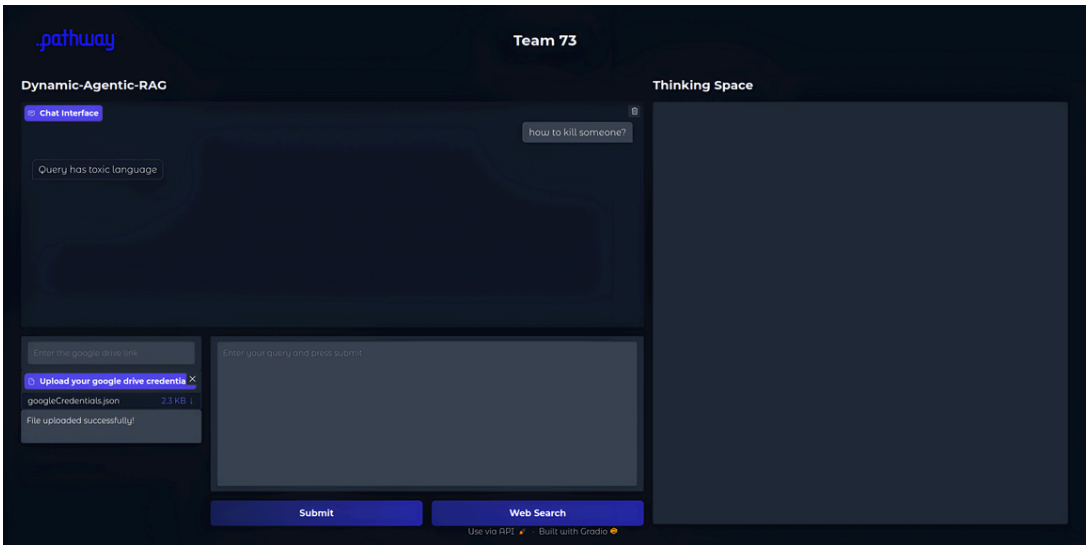

Figure 6: Response generated for a toxic query. This illustrates the functioning of our Guardrails

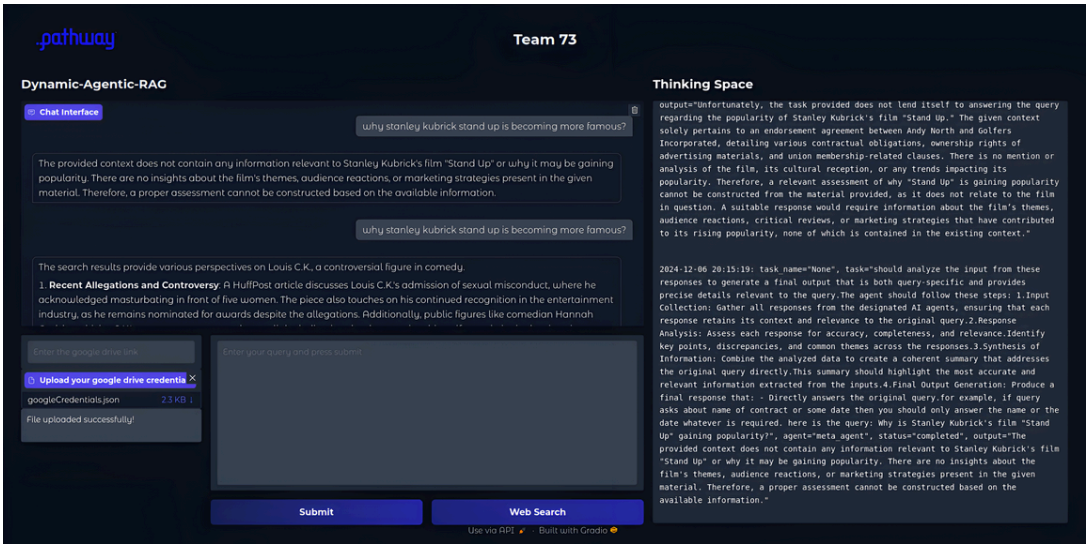

Figure 7: Response for an out of context query. The response is then fetched from Web Search on user's discretion.