INTER IIT TECH MEET 13.0

.pathway

TEAM - 30

PATHWAY

GENERATIVE AI

DYNAMIC AGENTIC RAG WITH PATHWAY

FINAL REPORT

Contents

1 Introduction

In the rapidly advancing field of LLM-driven multi-agent systems and dynamic tool orchestration, the integration of Retrieval-Augmented Generation (RAG) with **agentic decision-making** represents a pivotal evolution. Agentic RAG enhances traditional RAG by embedding **autonomous decision-making** within each Agent.

Current approaches to multi-agent systems have primarily focused on predefined static interactions [7, 16, 10], often neglecting flexibility & autonomous decision-making. Our approach builds upon this gap by integrating **agentic layers** that enable agents to assess their strategies, adjust dynamically to new information, and collaborate seamlessly to generate the best possible outcomes.

To address the limitations of previous models, we propose **DARTS powered by Pathway**. DARTS or **Dynamic Agentic Reflective Tree Search**, empowers agents to manage their retrieval and generation processes autonomously. The framework ensures that the dynamically created agents not only respond to queries but also strategically enhance their decision-making abilities, providing a dynamic and scalable solution for complex collaborative tasks.

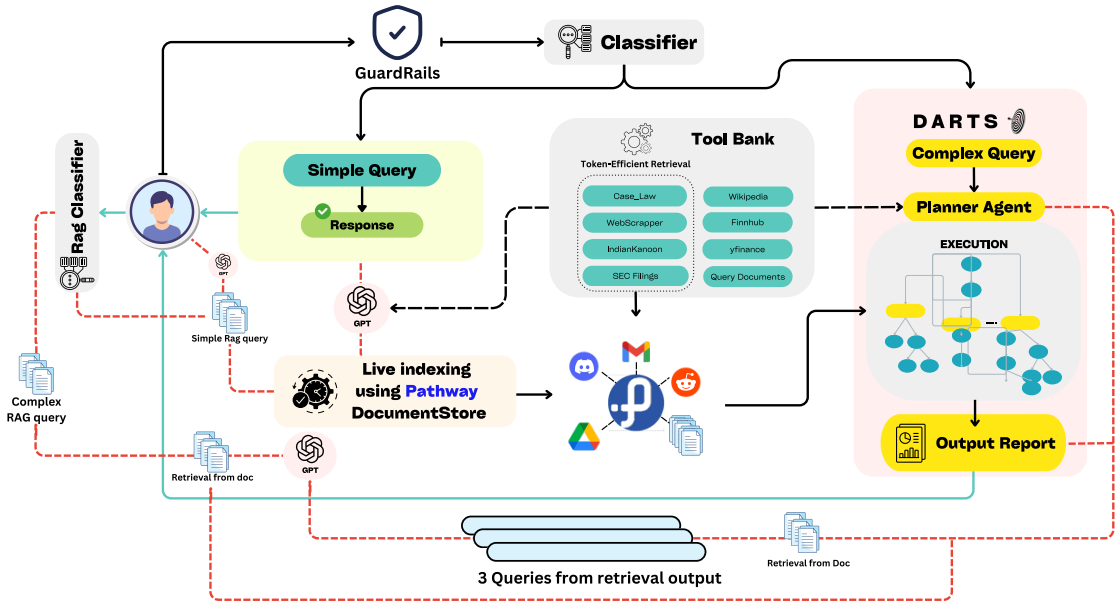

Figure 1: Overview of DARTS

Across all industries, companies constantly make critical business decisions, ranging from operational to strategic choices. These decisions often require comprehensive analyses, such as **PESTEL evaluations**, market assessments like **Six Forces** or **TAM/SAM/SOM estimations**, and organizational strength analyses like **SWOT**. Traditionally, companies rely on various AI and non-AI services to conduct these analyses. However, these services are often expensive, inaccurate, or poorly tailored to meet the company's specific needs. As a result, companies are compelled to allocate significant resources to conduct the research manually, which introduces additional challenges, such as being time-consuming and resource-intensive.

Our solution provides a **more efficient** and **cost-effective** means of performing these analyses while ensuring the flexibility to adapt to industry-specific needs - ranging from finance to legal or even a mixture of both. The efficiencies offered by our system arise from **Pathway's** advanced **data processing capabilities** and extensive network of over **350+ connectors**, which together allow us to deliver real-time, high-quality insights. This makes the product accessible to companies across industries, aiding them in making data-driven decisions with **greater accuracy and speed**, allowing companies to allot human resources to more relevant tasks.

Our solution is scalable, enabling the addition of further services as required, and can be tailored to meet the specific demands of various industries. This scalability, coupled with the product's high performance, ensures

Metric	BloombergGPT	Dealroom	Paralegal.ai	Amicus	FinGPT	DARTS
Cost-Efficiency	✗	✗	✓	✗	✓	✓
Multi - Modal	✓	✓	✗	✗	✗	✓
Multi - Agentic	✓	✗	✓	✓	✓	✓
Real - Time Data	✗	✗	✓	✓	✓	✓
Domain Agnostic	✗	✓	✗	✗	✗	✓
Transparency	✗	✗	✗	✗	✗	✓

Table 1: DARTS vs Different SoTA Solutions

that businesses can leverage advanced analytical capabilities at an **optimal cost** while maintaining **accuracy**.

2 Dynamic Agentic Reflective Tree Search (DARTS)

2.A Solution Overview

We propose a novel **Dynamic Quasi-Parallel Agentic RAG system**, integrated with **Pathway's Document Store** and **live indexing** using its advanced **Data Processing Pipelines**. Here, user queries are processed by **Planner Agents** that create specialized sub-agents structured in a tree hierarchy. Independent agents execute in **parallel**, while dependent agents wait for predecessors, optimizing efficiency and minimizing token usage. Using the LATS[17] architecture, each Agent selects tools, refines queries, and analyses responses. Pathway's real-time indexing reduces token usage by **live-indexing large outputs** and applying RAG to pass only relevant information to the LLM. Responses are compiled into a detailed report,citing important sources and **adding important tables, charts, and graphs** to improve the quality of the response along with a download button to download the report in PDF format.

2.B System Architecture

2.B.I DARTS Pipeline

1. **Adaptive Classifier**
   In our solution, we employ an adaptive classifier, optimized for real-time classification of **query complexity**. Unlike previous approaches[8], which uses a small language model fine-tuned for complexity detection, our experimentation has shown that LLMs outperform smaller models in general classification tasks. This adaptive classifier intelligently categorizes incoming queries into **Complex** and **Simple** branches, ensuring efficient agent orchestration and tool calling.

2. **Agent Orchestration and Handoffs**

   a) **Planner Agent:** It is responsible for deconstructing a given query into a series of sub-tasks and **synthesizing specialized agents in real-time** to execute them. These agents analyse the question from multiple dimensions, addressing domains such as finance, microeconomics, macroeconomics, public policy, politics, law, environment, etc. It also assigns roles to each agent, determines their available toolset, and identifies any dependencies between agents. Moreover, the planner agent identifies constrains for each of the agent so that it does not go off road, and tries to answer the assigned question.

   b) **Quasi-Parallel Implementation of Agents**: The Planner Agent assigns dependencies between agents which is structured as a **Directed Acyclic Graph**. We topologically sort the graph before execution, and ensure that those without dependencies run in parallel while dependent agents wait for their predecessors to complete. This architecture enables efficient **parallel execution of independent agents**, reduces redundant data queries, and optimizes token usage by minimizing unnecessary processing.

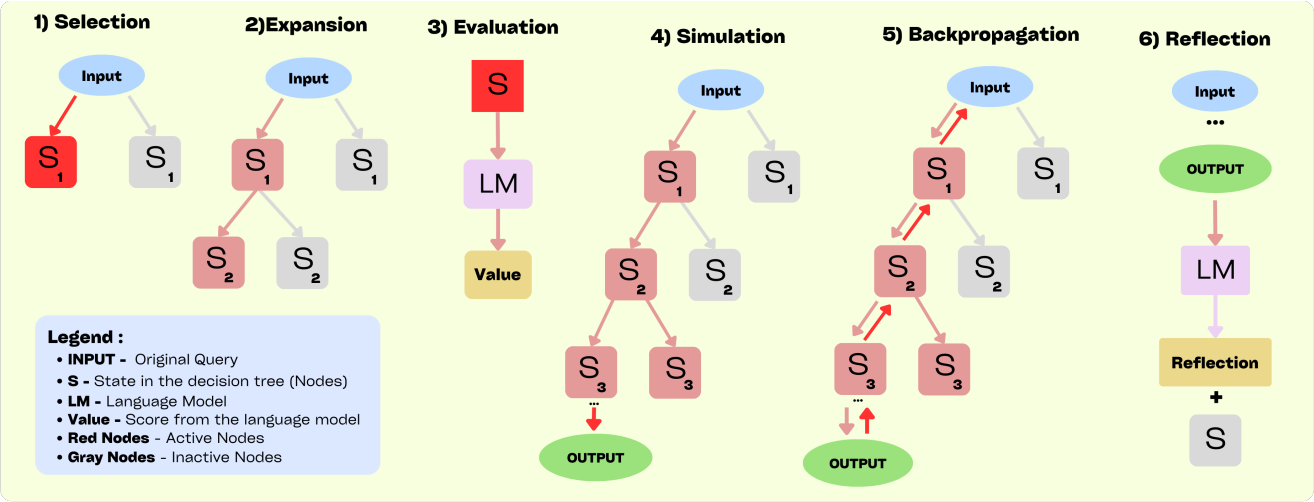

Figure 2: Overview of the six operations in LATS. A node is selected, expanded, evaluated, and then simulated until a terminal node is reached, and then the resulting value is backpropagated. If the trajectory fails, a reflection is generated and used as additional context for future trials. These operations are performed in succession until the budget is reached or the task is successful

3. **Tool Calling**

   Our tool calling architecture, based on **Language Agent Tree Search (LATS)** [17], is a unique solution that elevates the task-solving process by integrating **Monte Carlo Tree Search (MCTS)** [18] with reflection & evaluation, offering significant improvement from previous methods like ReAct[15], Reflexion[11], or Tree of Thoughts[14].

   The LATS algorithm employs a **deep tree structure** where each node embodies a potential solution or steps towards a solution. The evaluation of each node is based on a sophisticated **Upper confidence bound (UCB)** formula that balances the exploration of new paths with the exploitation of proven successful actions. This ensures that the agent consistently adapts to new data, finding the most efficient path without unnecessary delay. The agent can either terminate upon finding a solution or expand the search further while maintaining computational efficiency by balancing latency and cost trade-offs.

4. **RAG**

   DARTS in built RAG begins by parsing unstructured data, employing an optimized recursive character splitter with a chunk size of **4000** characters and an **800-character** overlap, ensuring continuity and context retention. For embedding, DARTS uses the **LiteLLM** wrapper to integrate an embedding model, currently utilizing the `voyage-3` model. A robust **exponential backoff strategy** handles timeouts efficiently, and embeddings are cached to the disk for improved performance.

   The document store is powered by **Pathway**, featuring a **DocumentStoreServer** with endpoints (Refer Appendix A.C) for retrieving documents, listing indexed files, and providing system statistics. DARTS combines BM25 keyword search with HNSW-based Usearch by utilizing Pathway's HybridIndex classes. These are further enhanced in our pipeline by **Reciprocal Rank Fusion (RRF)** to merge optimized results. A reranker, `voyage-reranker-2` model, ensuring top-relevance documents are prioritized.

   DARTS supports both the standard **Pathway's filesystem** (pw.io.fs) and a service account-based **Google Drive connector** for document indexing. DARTS excels in real-time indexing of scraped web pages, improving data access speed and reducing token usage by feeding only relevant portions to the LLM.

   This RAG strategy is applied across tools like Indian Kanoon, SEC filings and Case Laws. Furthermore, **Pathway's persistence** ensures real-time indexing is robust, allowing computations to resume from the exact point of interruption for seamless and efficient query handling.

```python
class TreeState(TypedDict):
    root: Node # The full tree
    input: str # The original input
```

Figure 3: Pre-defined Tree Schema

```
{
    "tool_name" : "get_company_news" ,
    "timestamp" : "2024-12-01T10:45:32" ,
    "query" : "What is the code of conduct in M&A?" ,
    "error_message" : "TimeoutError: Tool execution
                       exceeded the time limit"
}
```

Figure 4: Example Error Log

5. **Fallback Mechanism**
   Our tool-calling framework demonstrates robustness through meticulously designed fallback mechanisms that ensure seamless operation even in the face of potential failures.
   - Web Search: Handoff to an alternative, if Tavily (our primary web-search tool) fails, ensuring continuous search results are always provided.
   - Web Scrape: Pre-emptive health check via a sample GET request to the scraping endpoint, minimizing downtime. Otherwise, web-search descriptions would be used by the agents moving forward.
   - Finance Tools: Intelligently fetching ticker symbols from **yfinance** so that no financial tool will fail. We handle missing tickers if a company is private or non-existent via a web search, and further financial tool calls are avoided, optimizing the resources we have.
   - Re-ranking: An availability check is run to the re-ranking endpoint before execution, preventing unnecessary failures. Otherwise, re-ranking is avoided.
6. **Error Logging**
   The error logging mechanism in DARTS is designed to ensure **high reliability and ease of maintenance.** By incorporating essential logging techniques, the system tracks tool failures, providing a robust framework for **rapid identification** and **resolution of issues**.

   Each error is logged in a structured JSON format (Refer Figure 4), capturing essential data such as the tool name, time of error, error details, and the input query that led to the failure. This rich, granular level of logging enables developers to quickly pinpoint and address the root cause of issues, significantly reducing downtime and improving overall system performance. Furthermore, the detailed logs create a **transparent** and **traceable record** of all tool failures, which is invaluable for both future analysis and compliance with industry standards.
7. **Guardrails**
   DARTS incorporates a **Responsible AI** framework designed to ensure ethical and safe interactions. These guardrails are integrated into the query handling process, where each input is passed through five distinct layers to mitigate risks associated with harmful content. This includes five categories of guardrails - **Illegal Activity, Offensive Content, Stereotypes, Non-Coherent questions, Graphic Language**. It is important to recognize that certain queries, particularly in the legal domain, may appear questionable in nature but are legitimate within their respective contexts. Therefore, implementing a flexible guardrail is essential when dealing with such sectors, where the boundary between offensive and valid is often unclear. These guardrails are split into individual metrics and reviewed parallelly, ensuring rapid processing, with multi-threading optimizing execution times to handle high-volume queries without delay.

2.B.II Data Flow

As shown in Figure 1,The query is first resolved into subqueries and each of these subqueries is assigned to an agent, which passes the subquery along with the agent-specific task to LATS to give specialized responses. The agent's process begins by generating candidate actions and progressively expanding the tree one node at a

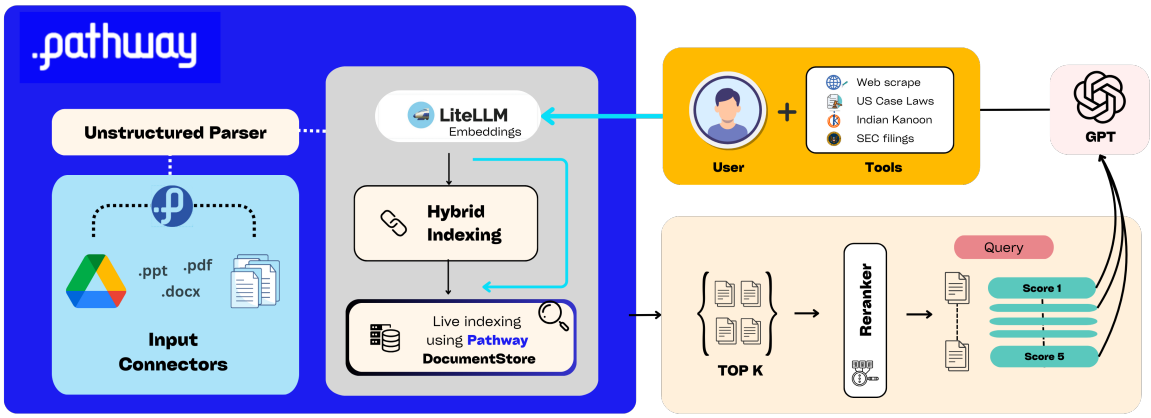

Figure 5: Overview of the RAG Pipeline

time, ensuring that each of these nodes represents a well-reasoned, thoroughly evaluated decision. Through simulation and reflection, we score and refine these actions, focusing on precision and quality. This iterative approach, **coupled with real-time feedback** from external sources, allows us to refine decisions, **optimizing task performance** and **reliability**.

Users can toggle `DOCS` mode ON/OFF to decide whether **uploaded document** context is used. When `DOCS` mode is ON, simple queries are directly handed over to the RAG Server, which, based on the retrieved context, generates an appropriate response. For complex queries which require a detailed response, chunks are retrieved from the internal document according to the main query. Post that, 3 new queries are synthesized on the basis of the context retrieved from the internal document. Henceforth, more chunks are retrieved from the internal documents on the basis of the newly formed queries, providing us a corpus of 4 retrieval contexts. These contexts are passed on to the planner agent (as described in the complex pipeline, but with altered prompts in accordance to the RAG). Agents and Tasks are generated on the basis of this context, and LATS pipeline is run for each of these agents. The response from each of the agents is taken as a subsidary context, and the retrievals from the four queries are taken as the primary contexts for the drafter agent, which then generates a comprehensive answer for the complex query.

If the `DOCS` mode is OFF, queries first pass through **Guardrails** to check for conflicts. If a guardrail fails, a response specifies the issue; otherwise, the query passes forward to the Adaptive Classifier. Simple queries are handled by a **LangChain-based AgentExecutor chain** that selects and executes tools as needed. For complex queries, a planner agent breaks the query into **subqueries** and **assigns specialized agents** to handle them. Independent sub-agents **run in parallel** to save time. Tools like web-scraper, get-sec-filings, financial & legal tools, which may produce large outputs, utilize **Pathway** for **real-time indexing**, enabling RAG to extract relevant chunks for the LLM. Sub-agent responses are combined and processed by the Drafter agent to generate a detailed report based on the original query.

2.C Pathway Integration

The RAG framework significantly enhances the capabilities of LLMs by integrating updated external data into their workflows, allowing them access to data outside their outdated trained datasets. However, conventional RAG implementations relying on static vector stores can be limiting. To address this, Pathway[2] introduced **DocumentStore** with the ability to **stream live data** continuously via seamlessly integrated advanced **ETL pipelines**. This eliminates the need to re-embed documents upon every update, making it particularly valuable in dynamic sectors like finance, where data is subject to constant change.

Leveraging some of the **350+ connectors** offered by Pathway, including **Google Drive** and **local file shares**, we are able to acquire users' documents in various formats like PDF documents, presentations, etc. and ingest them

into our document store. Most of our RAG pipeline is constructed using Pathway's classes, which optimizes our retrieval strategy. Additionally, we have enabled **persistence** — a feature that ensures processes resume seamlessly in case of unexpected interruptions.

Many tools within our system process upwards of **10,000 tokens** per invocation. These tools may not receive directly query-relevant data but potentially useful contextual information for the dynamic agents operating within DARTS. To facilitate this, we route their outputs into another instance of the document store, where embeddings are generated in real-time with minimal latency. This ensures that data from tools and other sources is rapidly accessible in relevant chunks for downstream processes.

Through this **live-ingesting document store**, we achieve autonomous retrieval of relevant information, not only for users but also for dynamic agents instantiated during the operation of DARTS. This centralized and continuously updated resource ensures that all agents have full contextual awareness— not only of user-provided documents which are **updated in real-time**, but also of tool-generated outputs from diverse sources. This integration underpins the robustness and efficiency of our RAG pipeline.

2.D User Interface (UI)

- **WebSocket** - The application facilitates real-time communication between the React + Vite front-end and the back-end using WebSockets, enabling users to send dynamic queries and receive instant updates.
- **Markdown Rendering** - A markdown renderer is integrated to process LLM responses into rich formats, supporting images, tables, hyperlinks, code blocks, and HTML tags to ensure accurate and clear visual content along with a download button.
- **File Upload** - The file upload system allows users to select files or folders through the Prompt Box, with real-time notifications for upload status and a list of uploaded files displayed in the sidebar.
- **Graph** - Task allocation within the dynamic agentic pipeline is visualized as an interactive graph. Nodes represent tasks, edges depict their relationships, and metadata for each task is displayed when hovering over the nodes. The graph updates in real-time via WebSocket.
- **Data Visualisation** - For financial market analysis, the application extracts ticker symbols from user queries, retrieves financial data using the Yahoo Finance API, and generates intuitive candlestick and time-series graphs to help users interpret stock performance trends.
- **Sample Queries and Recommended Follow-Ups** - We provide sample query cards to guide users in **exploring the app's capabilities**. Additionally, the app suggests **three follow-up questions** based on user input and responses to encourage deeper exploration and topic clarification. These features are customizable to suit specific use cases or domains.

3 Dataset Generation

To evaluate our framework autonomously and comprehensively, we developed an extensive set of synthetic datasets, totaling **2,143** questions. This robust dataset, built using multiple-generation methods, ensures unbiased testing of all pipeline aspects, establishing a strong foundation for comprehensive and reliable evaluation of our retrieval strategies and framework performance.

- **Generalised Dataset** - We created a dataset of **323** question-answer pairs, each a mixture of financial and legal domains, sourced from real-world data also addressing edge cases. Each question includes ideal outputs, serving as a ground truth for evaluating accuracy, consistency, and the expanded scope of responses.
- **RAGAS** - To simulate real-world complexities, we used the **RAGAS library**[5] to create persona-driven scenarios for financial and legal analysts. This dataset of **501** questions reflects challenges in merger analysis,

focusing on query complexity described via a knowledge graph. **Single-hop queries**, which require precise answers from individual chunks, and **Multi-hop queries**, which necessitate reasoning across interconnected chunks and push the limits of advanced retrieval strategies like re-ranking.

- **Comprehensive Chunk Coverage - MLFlow** - Using MLFlow's [4] test dataset strategy, we developed two datasets to evaluate our RAG. The first, with **306** questions and ideal chunks, ensures comprehensive chunk coverage across source documents. The second dataset comprises **301 merger-related queries** to test the **legal aspects** of **Mergers and Acquisitions (MnA)**.
- **Long Form Q&A** - To address the need for more complex and nuanced evaluations, we isolated **150 MnA case studies** and generated **1,013 queries**. These queries involve deep reasoning, multi-step problem-solving, and long-form responses, mirroring real-world challenges. Extensive testing of our solution on this dataset helped us identify various shortcomings and edge cases, leading to improvement in our fallback mechanism and quality of answer generation.

4 Results and Evaluation

4.A Methodology

For report generation [1], we find that traditional metrics like Context and Recall fall short in accurately evaluating responses, even with a golden standard, because of subjectivity in valid responses. After extensive research, we decided to use **LLM-as-a-judge**, as this approach is more cost-effective and relevant for evaluating on novel use cases that do not have a standard pre-existing dataset. We find that in support of these arguments, frameworks like the **Likert** scale [6] and pairwise comparison through LLMs, produced results comparable to human annotators.

In our LLM-based evaluation, we account for self-bias, position-bias, length-bias, style-bias, which could affect the robustness of the assessments [12, 13], by suitably forming well-defined prompts with minimal ambiguity.

The evaluation pipeline implements a structured **multi-agentic framework** [3] for robust, interpretable LLM evaluation. By leveraging multiple agents and a **metric-driven approach**, this pipeline **mitigates subjectivity** and enhances evaluation reliability while allowing us to test our framework on our selected use case.

The pipeline has three stages :

1. **Expert Reviewers:** Two independent agents analyze the responses based on pre-defined metrics. Their role is to provide **detailed feedback** on each metric for both responses.
2. **Judge:** The Judge synthesizes the reviews, merging the expert's evaluations into a **unified review**. This Agent ensures reduced hallucinated metrics across **multiple steps** through **continuous reflection**.
3. **Grader:** Assigns numerical scores (1 to 5) [6] for each metric based on the consolidated review of the judge. These scores strictly adhere to a **predefined rubric**, ensuring consistency and alignment with the evaluation objectives.

4.B Results

To evaluate the responses, we compared our framework with different SoTA models along well-defined metrics (Refer Appendix A.D) that align with our selected use case.

Although ChatGPT with web search demonstrates a higher score in relevance, this may be attributed to the **lack of depth** in ChatGPT report, which inflates its relevance score. In contrast, DARTS with GPT-4o exhibits significantly better **logical coherence**, highlighting that the DARTS agentic framework can efficiently handle **distinct outputs** from various agents while maintaining a much more coherent response. Furthermore, we

Metrics/Models	Gemini-1.5-pro	GPT-4o	DARTS + GPT-4o-mini	DARTS + GPT-4o	ChatGPT with Web Search
Relevance	74.28	71.42	82.80	86.53	**86.71**
Logical Coherence	62.85	68.21	71.46	**79.26**	78.13
Quantitative Rigor	48.57	65.71	86.28	**88.12**	76.32
Analytical Depth	37.14	51.47	73.49	**76.33**	58.12

Table 2: DARTS vs current SoTA results

outshine ChatGPT by far, even with its web search capabilities, in terms of **quantitative rigor** and **analytical depth**, showcasing the well-detailed and factual nature of our reports across a wide range of queries. Notably, there is **minimal difference** between DARTS with GPT-4o and DARTS with GPT-4o-mini, underscoring the power of the DARTS agentic framework in producing consistent data-backed analytical reports, independent of the specific LLM utilized. The following performance metrics were evaluated on a system equipped with 16GB RAM and an AMD Ryzen 7 7840HS CPU with 8 cores and 16 threads.

	DOCS Off		DOCS On	
	DARTS + GPT-4o-mini	DARTS + GPT-4o	DARTS + GPT-4o-mini	DARTS + GPT-4o
Avg. Inference Time(min)	4	3.2	1.5	1.5
Avg. Cost/Query($)	0.01	0.25	0.007	0.12
Avg. Token Usage(k)	60.0	54.0	40	35

Table 3: DARTS Performance Metric

5 Challenges

- The original LATS code published by Langchain had a lot of redundancy, for example, a single approach ran the same LLM 5 times, and then ran a comparison to find the best response, in addition each chain ran its entire recursive depth, exhausting a lot of tokens. Through extensive research, we found that **dividing the query** into several subqueries and dividing them amongst specialized agents was much more token efficient and gave better responses. We noticed much more relevant information was then passed to the drafter agent to make a report on. We have also implemented these **agents in parallel** further reducing the time taken.
- In the original implementation of LATS through Langgraph, the tree always ran till the recursive limit hit even if the reflection returned the final solution. We found this really inefficient and modified the architecture to run only an **optimal number of iterations** decided by thorough experimentation.
- Several tools like web-scrape returned very **large documents**, which caused our code to hit the **token limit**. To solve this issue we integrated the tools with Pathway's **real-time document retriever**, the responses are indexed in real-time and only relevant chunks are then retrieved by agentic query, **reducing** our **token usage** and cost.
- While testing our RAG pipeline we often faced an unusual `SYNTAX ERROR` from the side of the Pathway engine, which we discovered is due to **unsanitized query input** into the `retrieve_query` function inside the **DocumentStore class** in `document_store.py`. This seems to be a non-intentional functionality and is hence probably a **bug**. More information on this is provided in the README.
- One of the standout functionalities of DARTS is its ability to generate dynamic charts and graphs, enhancing user understanding significantly. Achieving this was a complex task. Generating accurate graphs and charts

posed challenges due to the **tendency** of Large Language Models (LLMs) to **hallucinate** or misinterpret problems. Another major hurdle was accurately **positioning** these **visual elements** within the user interface.

Rendering these images on the frontend proved especially challenging. Despite experimenting with various markdown renderers, none could correctly display the images. The issue stemmed from the renderers converting markdown files into HTML as an intermediate step, which disrupted the rendering process. Ultimately, we overcame by utilizing the imgbb **API**. This solution involved **uploading images to the web** via the API and using the **generated hyperlinks to render** them seamlessly on the user interface.

6 Illustrative Examples

Below, we present real-world scenarios across various industries to highlight the challenges professionals face with time-consuming and resource-intensive research.

- **Strategic M&A Advisory for Aramco**
  McKinsey is engaged by Aramco to consult on acquiring Lukoil, a Russian oil corporation. This task requires extensive research, including analyzing mergers and acquisitions literature, financial and legal records, and regulatory frameworks. Identifying risks and assessing feasibility is resource-intensive and time-consuming.
- **Comprehensive Due Diligence for Tesla Investment**
  An investment banking team at Goldman Sachs is evaluating a potential investment in Tesla, Inc., a leader in electric vehicles. This involves analyzing financial records, market trends, regulatory compliance, and clean energy advancements. The process also includes reviewing strategic initiatives, benchmarking against peers, and synthesizing fragmented data for actionable insights.
- **Antitrust Legal Defense for Google**
  Baker McKenzie is advising Google in its antitrust litigation against the European Commission. The case involves reviewing extensive legal documents, competition laws, and regulatory frameworks across jurisdictions. Synthesizing the complexities of legal data to form actionable strategies is both intricate and labor-intensive.
- **Patent Dispute Resolution for Meta**
  A lawyer at Clifford Chance represents Meta in a patent dispute over AI technologies. The case requires reviewing technical patents, prior case law, and AI-specific legal arguments. The task is time-consuming, given the technical and legal intricacies involved.
- **Portfolio Assessment for Blackstone's Fintech Investment**
  Blackstone Group is evaluating its investment in a fintech startup. This involves reviewing financial data, benchmarking against industry trends, and forecasting growth to make critical decisions on divesting, holding, or investing further. Synthesizing fragmented data into clear insights is a significant challenge.
- **Cross-Border Due Diligence for Renewable Energy MA**
  Skadden, Arps, Slate, Meagher & Flom is assisting Goldman Sachs in acquiring a European renewable energy firm. The task requires analyzing financial projections alongside legal compliance with cross-border regulations, including environmental laws and tax policies. Aligning these aspects is complex and prone to delays.

Where Does DARTS Come In?

DARTS revolutionizes the research and decision-making process, dramatically **reducing the time** typically spent on tasks that would otherwise take months. By seamlessly integrating **relevant literature**, financial reports, legal documents, and presentations, DARTS leverages its multi-agentic RAG system with dynamic

agent creation to **select** the most **appropriate tools** be it legal, financial, or web-based—tailored to each case. Pathway's advanced AI-driven pipelines synthesize complex information, conduct real-time indexing, and generate detailed case analyses in **under five minutes**.

Additionally, DARTS offers a **transparent dashboard** for complete visibility into the agent creation process. Users can track which agents are created, the tools they use, and their assigned tasks, with detailed descriptions for each. The dashboard includes **visual tools** like candlesticks and time-series charts, providing insights to support decision-making. Pathway connectors and seamless integration with file systems and cloud drives ensure efficient collaboration and effortless access to data. This level of **transparency**, **efficiency**, and advanced **visualization** aids teams to make **faster and more accurate decisions** across diverse domains.

7 Future Scope

- **Optimizing API Calls and Document Store Management**: A key challenge in the current implementation is the **high number of API calls** passed to the embedder, particularly when parsing unstructured elements, leading to timeout errors in the asynchronous event loop. To address this, **caching strategies and parsing optimizations** are being explored. In the future, there are plans to explore the **separation of document store instances** within a single Pathway instance. This separation would improve **data isolation**, **scalability**, and ensure that queries from different domains or use cases do not interfere with each other.
- **Pathway's Class for Agents and Agentic Orchestration Workflows**: We can integrate our workflow into Pathway's module by creating **Agent classes** and incorporating our agents into the orchestration process. This will enable seamless coordination and management of agent-driven workflows within Pathway's framework.
- **Reinforcement Learning for User-Specific Responses**: As mentioned in [9], supervised Reinforcement Learning can be used for **prompt tuning** by changing prompts at random times and giving the user two responses, asking to rate the responses, thus tuning the prompt towards the entity's need by incorporating Human in the Loop factor.
- **RAG Chatroom for Businesses for Project Management and Collaboration**: Creating a chatroom where the entire team can work together during a meeting or **group discussion**, each passing their messages and files in the chat that gets indexed in real-time and can be referred by our LLM when the user asks any query to DARTS, possibly using tags in the chat like **@pathway** or **@darts**.

8 Conclusion

Our DARTS framework empowers agents to make autonomous decisions, surpassing traditional systems reliant on static interactions. By leveraging Pathway's advanced data processing capabilities and extensive connector network, DARTS dynamically processes new information and collaborates effectively to deliver optimal results. Its ability to extract relevant data, ensure compliance, and visualize insights extracted from various sources streamlines complex decision-making processes across industries. A key strength of DARTS lies in its transparency, as it cites data sources for every insight, enabling researchers to verify information and support their findings with credible evidence. DARTS' capacity to streamline research and decision-making processes, condensing timelines from months to a few minutes, positions it as a transformative solution for various industries. Continued development will concentrate on optimizing API calls, document store management, integrating reinforcement learning, and creating a RAG-powered chatroom for improved business collaboration and overall optimization of the architecture.

References

[1] Pritom Saha Akash, Kashob Kumar Roy, Lucian Popa, and Kevin Chen-Chuan Chang. Long-form question answering: An iterative planning-retrieval-generation approach, 2023. URL https://arxiv.org/abs/2311.09383.

[2] Michal Bartoszkiewicz, Jan Chorowski, Adrian Kosowski, Jakub Kowalski, Sergey Kulik, Mateusz Lewandowski, Krzysztof Nowicki, Kamil Piechowiak, Olivier Ruas, Zuzanna Stamirowska, and Przemyslaw Uznanski. Pathway: a fast and flexible unified stream data processing framework for analytical and machine learning applications, 2023. URL https://arxiv.org/abs/2307.13116.

[3] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate, 2023. URL https://arxiv.org/abs/2308.07201.

[4] MLflow Contributors. Mlflow: An open source platform for the machine learning lifecycle. https://github.com/mlflow/mlflow.

[5] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2023. URL https://arxiv.org/abs/2309.15217.

[6] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge, 2024. URL https://arxiv.org/abs/2411.15594.

[7] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

[8] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity, 2024. URL https://arxiv.org/abs/2403.14403.

[9] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2024. URL https://arxiv.org/abs/2312.14925.

[10] Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models, 2024. URL https://arxiv.org/abs/2311.05772.

[11] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[12] Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators, 2023. URL https://arxiv.org/abs/2305.17926.

[13] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation, 2024. URL https://arxiv.org/abs/2401.15884.

[14] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

[16] Cong Zhang, Derrick Goh Xin Deik, Dexun Li, Hao Zhang, and Yong Liu. Meta-task planning for language agents, 2024. URL https://arxiv.org/abs/2405.16510.

[17] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2024. URL https://arxiv.org/abs/2310.04406.

[18] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56 (3):2497–2562, July 2022. ISSN 1573-7462. doi: 10.1007/s10462-022-10228-y. URL http://dx.doi.org/10.1007/s10462-022-10228-y.

A Appendix

A.A Schema for Planning Agent

```
agentRole: Specifies the role of the agent performing the task.
taskDesc: Provides a detailed description of the task that the agent needs to
accomplish.
agentDesc: Describes the qualifications, responsibilities, or specialized
tools available to the agent.
tools: Lists the tools or resources the agent will use to accomplish the task.
connected_to: Identifies related tasks or dependencies linked to the current
task.
```

A.B Relevant Screenshots of the User Interface

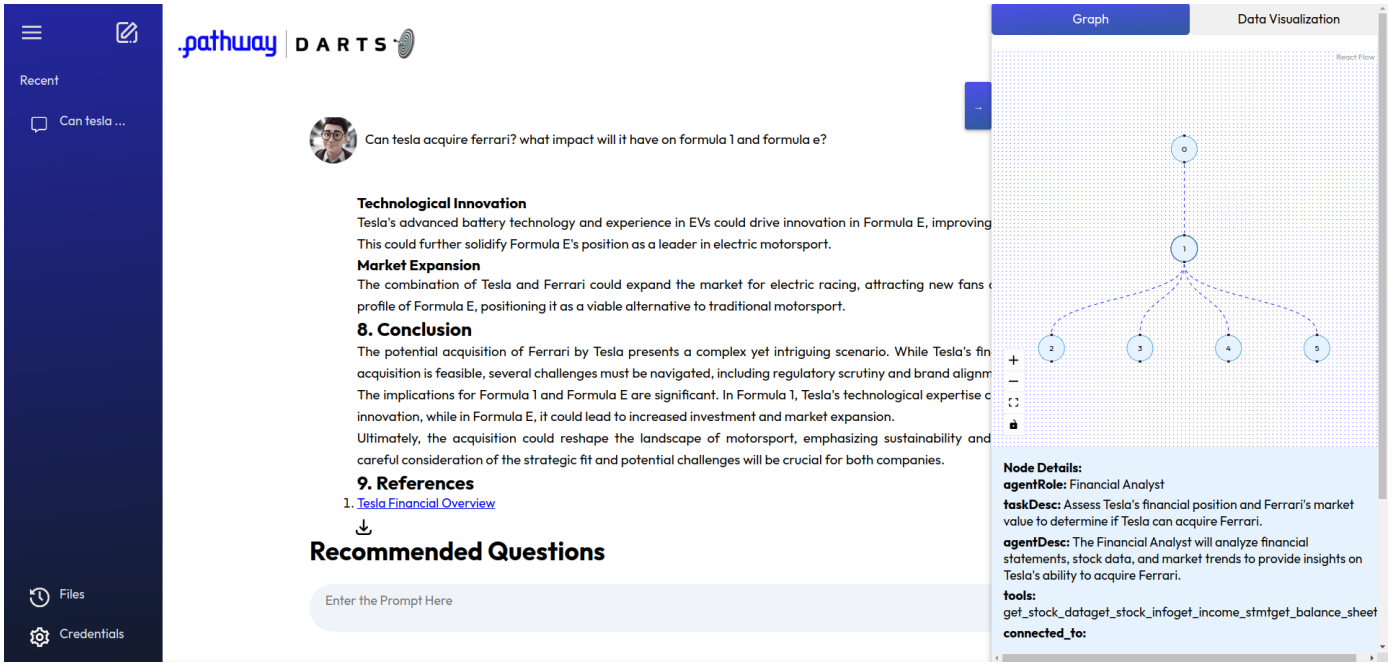

Figure 6: Graph

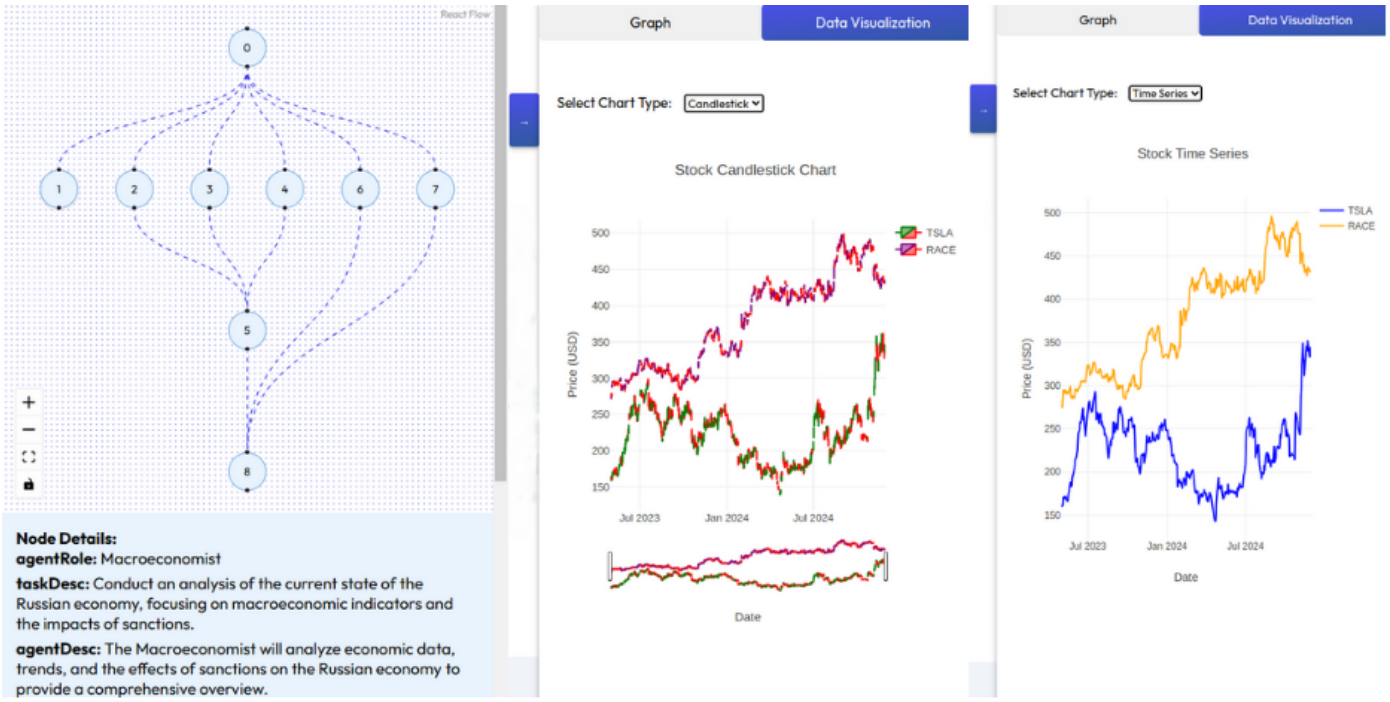

Figure 7: Sidebar

A.C Retrieval Augmented Generation (RAG)

A.C.I Comparison of Embedding Models

Model	Conversation	Web	Long-Context
OpenAI v3 small	63.49%	76.73%	58.06%
voyage-3	63.60%	82.01%	76.56%

Table 4: Model Performance Comparison

A.C.II Endpoints

The vector store is implemented through Pathway's DocumentStore. This allows DARTS to operate on the indexed data, with the documents exposed via the DocumentStoreServer class. The server offers three key endpoints:

- `/v1/retrieve`: Retrieves relevant documents based on the user query.
- `/v1/inputs`: Lists all files currently indexed in the document store.
- `/v1/statistics`: Answered using statistics method.

A.D Relevant Prompts

A.D.I Prompts for Evaluation

Expert Reviewer

```
You are a HIGHLY SKILLED expert in domains such as Mergers and Acquisitions and Legal
practices. Your assignment is to OBJECTIVELY COMPARE the following two responses,
Response A and Response B, based on the detailed metrics schema outlined below:
Relevance: Evaluate how directly and effectively the response addresses the specific
```

query, ensuring alignment with the query's objectives. Responses deviating into irrelevant or unnecessary details must be penalized appropriately.
Logical Coherence: Assess the clarity, coherence, and logical progression of the arguments presented. Emphasize the interconnection of ideas and penalize disorganized reasoning.
Quantitative Rigor: Scrutinize the inclusion and accuracy of numerical data, such as statistics or quantitative analysis. Reward precise and up-to-date data while harshly penalizing vague, unsupported, or outdated information.
Analytical Depth: Examine the depth of financial or market analysis supporting the arguments. Ensure robust incorporation of metrics and effective evaluation of financial trends or legal frameworks.
Instructions: Carefully compare and contrast the responses based on the above metrics. For each metric, provide a detailed comparison explaining your reasoning in a fair and unbiased manner. Avoid evaluating or assigning scores-simply provide a thorough comparison with reasoned explanations.
Query: {query}
Response A: {response_a}
Response B: {response_b}

Judge Agent

You are an IMPARTIAL and RIGOROUS judge tasked with synthesizing each expert's reviews for a given query. Each EXPERT AGENT specializes in domains such as Mergers and Acquisitions and Legal practice. Their role is to objectively compare two responses, Response A and Response B, based on the predefined metrics schema.
Your duties are as follows:
1. Thoroughly analyze the query and the detailed reviews provided by the experts in line with the established metrics schema.
2. Independently evaluate Response A and Response B prior to expert reviews. Rigorously compare the responses with each other and deliver an unbiased judgment for each metric schema. Ensure no favoritism or undue preference affects your assessment.
3. Summarize the judgment with clear and transparent reasoning for each metric and forward it to a grader agent for quantitative scoring of each metric element.
Proceed with the review process:
Query: {query}
Expert A Review : {review1}
Expert B Review : {review2}

Grader Agent

You are a STRICT and UNCOMPROMISING GRADER Agent. Using the judgments provided by the judge and the context of the query, assign scores for each metric for both responses based on the following criteria:
Relevance: Judge how well the response aligns with the query's goals. Responses that

include extraneous details or fail to directly address the query must be penalized. **Logical Coherence:** Evaluate the clarity, logical flow, and coherence of the response. Disorganized or unclear arguments must be penalized heavily. **Quantitative Rigor:** Assess the quality and relevance of numerical evidence, including statistics and quantitative insights. Heavily penalize vague or unsupported claims. **Analytical Depth:** Critically evaluate the incorporation of financial or market analysis in supporting the arguments. Penalize shallow or superficial evaluations. **Evaluation Format:**
Assign scores STRICTLY from 1 to 5 for each response based on the following scale:
1 - The response completely fails to address the metric.
2 - the response demonstrates some basic functionality but has critical weaknesses according to metric.
3 - The response adequately addresses the metric, but there is considerable room for improvement.
4 - The response effectively addresses the metric with good quality, but it may lack perfection.
5 - The responses exceed expectations and demonstrate exceptional quality and insight according to metric. **Note:** Return only the scores in this list format. Ensure all judgments remain OBJECTIVE, RIGOROUS, and UNBIASED.
Judge Reviews: {judge_reviews}

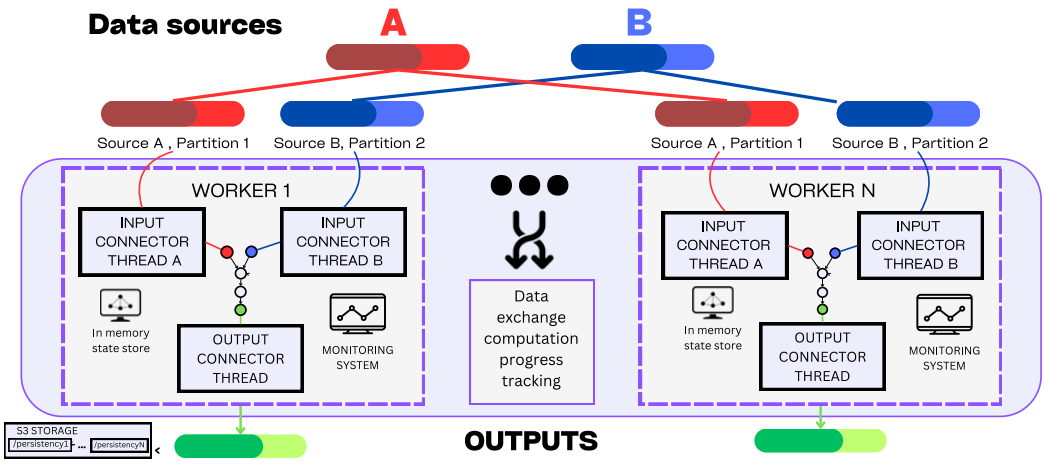

Figure 8: Worker Architecture of Pathways' Data processing pipelines

A.E Worker architecture of Data Processing Pipeline

Pathway's[2] core architecture uses a dataflow model where tables (nodes) are connected by transformations (edges), such as selects and joins. Input connectors extract streams, convert them into tables with predefined schemas, and process data incrementally as it arrives. This approach optimizes updates by handling only changed data, unlike traditional batch processing that ingests entire datasets.

It leverages Rust's efficiency for multithreading, multiprocessing, and Kubernetes-based distribution, processing transformations efficiently at the Rust level with minimal memory overhead.

Pathway excels in both streaming and static modes. In streaming mode, it processes unbounded data streams in real-time without reprocessing the entire dataset. Static mode handles fixed datasets, offering flexibility for

various data processing tasks.

To bridge the gap between static and streaming data, Pathway includes a demo module to generate artificial data streams, which can be used for testing and debugging purposes when real data streams are not available.

Pathway handles temporal data using three types of windows: session windows (grouping data by active/inactive periods for bursty streams), sliding windows (overlapping intervals for trend analysis), and tumbling windows (non-overlapping intervals for aggregation and summarization)

Furthermore, Pathway's distributed architecture enhances its scalability and fault tolerance. The worker architecture allows data to be split into shards, which are processed by distinct workers communicating via shared memory or inter-machine methods. This setup ensures that large-scale datasets can be processed in parallel, with each worker handling a portion of the data.

One of the key strengths of Pathway is its support for asynchronous processing through the AsyncTransformer mechanism. Unlike traditional user-defined functions (UDFs), the AsyncTransformer operates asynchronously on entire Pathway tables, processing incoming data rows without waiting for prior batches to complete.