INTER IIT
TECH MEET 13.0

HIGH PREP

.pathway

DYNAMIC AGENTIC RAG WITH PATHWAY

Background of Industry

As Gartner notes, Generative AI is rapidly transitioning from a niche technology to a core component of enterprise infrastructures. The year 2024 marks the full-scale adoption of Generative AI across industries, as organizations shift from pilot projects to production-ready implementations and RAG is at the center of it.

Pathway is uniquely positioned to support this transformation, offering easy-to-use tools for managing and deploying GenAI applications in real time, making it a key player in the enterprise AI and RAG landscape.

What is RAG: RAG stands for Retrieval-Augmented Generation. It extends the capabilities of LLMs like ChatGPT by retrieving relevant information from a large corpus of data and feeding it into the model, allowing it to answer questions with more context and accuracy.

For example: If you ask a model to answer a question from a recent research paper, it may not have that information within its training set. But using RAG, it can first retrieve the most relevant information from the research paper, and then generate an answer with this additional/retrieved knowledge. Here are a few resources on RAG that will help you learn more: RAG Introductory Blog and MultiModal RAG Introductory Blog

What is Agentic RAG: Agentic RAG enhances standard RAG systems by introducing autonomy and proactivity. The "agent" (your AI application) is empowered to make decisions on how to retrieve information, manage the retrieved data, and apply intelligent strategies, resulting in a more self-directed and optimized RAG pipeline.

Introduction

About the Company: Pathway Technology Inc. (pathway.com) is the maker of the world's fastest global data processing engine (GitHub). With offices in the US, France, and Poland, their ~25-member team has deep expertise from top AI labs like Microsoft Research, Google Brain, and ETH Zurich. Many of their members have worked at Google and hold degrees from prestigious institutions like École Polytechnique, UC Berkeley, CNRS, and HEC Paris—one even earned a PhD at just 20. their CTO has co-authored notable works with AI pioneers Geoffrey Hinton and Yoshua Bengio. Their leadership includes the co-founder of Spoj.com (one of the earliest CP platforms with over 1M developers) and NK.pl (Poland's first social media with 13.5M+ users), and advisors from current/previous leadership of OpenAI, SAP, and DHL.
Join their open-source community by starring the GitHub repositories below.
https://github.com/pathwaycom/llm-app
https://github.com/pathwaycom/pathway

About Pathway Framework
Pathway is a Python data processing framework designed for analytics and AI pipelines over data streams. It is the ideal solution for real-time processing use cases such as streaming ETL or Retrieval Augmented Generation (RAG) pipelines for unstructured, changing data.

Key Components of this definition:
Python Framework: Written in Rust 🦀 for speed and efficiency, Pathway is usable via Python, making it powerful yet simple to use with just Python know-how.

Data Processing: Pathway excels at processing large-scale, real-time data and is recognized as the world's fastest data processing engine. As a developer, you can use it for tasks like performing JOINS on incoming data streams (real-time data flow) or updating vector/hybrid indexes in real time. These are just simple examples—its potential goes much further.

AI Pipelines Over Data Streams: Pathway helps AI systems learn from real-time data streams, enabling applications like sentiment analysis, anomaly detection, and RAG pipelines that automatically adapt to incoming data.

Problem Statement

At Pathway, we offer several app templates with end-to-end executable code in Dockerized environments. However, for this challenge, we expect participants to explore and build solutions using their own expertise, rather than relying on a pre-defined template.

In this context, participants are tasked with building agents that can:

- Autonomously retrieve information or determine the appropriate course of action.
- Optimize token usage for more efficient processing.
- Ensure the accuracy and contextual relevance of the generated output.

Now, let's dive into the problem statement for the challenge.

Objective of the Problem Statement:
Create an Agentic RAG system using Pathway that autonomously retrieves, analyzes, and synthesizes information from multiple data sources. The system should dynamically decide the best approach for handling complex queries, utilizing techniques like corrective RAG and multi-agent collaboration to provide accurate responses.

Questions may be simple questions that are answered on the documents, or they may require pulling a variety of paragraphs from the documents, reasoning about them, and composing a final response. You are free to implement the agents in any framework or plain Python.

For its simplicity, OpenAI's Swarm library can also be explored. Important points are features, accuracy and robustness of the agent.

Key Features:

- **Autonomy in decision-making:** The agent decides how to retrieve and analyze information based on the query's complexity and the available data.
- **Flexibility in design**: Participants are free to choose how to implement the system, using multi-agent collaboration, corrective RAG, human-in-the-loop RAG, or other methodologies as needed.
- **Dynamic RAG with Pathway** (see definition): The index of choice should be either Pathway VectorStore or DocumentStore.
- **There should be a UI component:** UI is not crucial and it will not be prioritized against the agent implementation during the evaluation; however, it is nice to have visual elements. It may be a Streamlit/Gradio app or a more specialized web application. We have provided some examples below. Optionally, the UI can be enhanced to add transparency into the agent's responses such as the reasoning process, allowing users to understand and trust the agent's actions
- **Make it a responsible agent:** If possible, implement guardrails to ensure responsible AI practices, preventing the agent from producing harmful or unintended outputs.
- **Level 2:** Resilience to Error Handling. Simulate external API failures and instruct the agent to switch to alternatives. For instance, if the Google Search API fails, the agent should switch to Bing Search API or manual scraping tools. Test and demonstrate this by providing the agent with multiple ways to solve tasks, simulating callback failures, and giving the agent space to resolve them. Focus is not just on error handling in the application but managing failures in the callbacks.

Here, the agentic approach comes into play as the system doesn't just retrieve documents—it analyzes and decides which approach is the most reliable, demonstrating intelligent decision-making beyond basic retrieval.

Note: On Windows, you must use Pathway with Docker. We've provided a basic resource in additional links if you're new to containerization or Docker.

Extra/Bonus

Participants can further explore building the entire RAG pipeline within Pathway or enhancing one of its existing classes. Leveraging Pathway's dynamic data-handling capabilities will be key to success in this bonus challenge.

Final Deliverables

- Solutions must be presented on a GitHub repository with a clear README, documentation to replicate the solution, the architecture used, etc.
- The README must include a working demo with a video.
- A final presentation of the project is also required.

Report Required (Mid-term + End-term)

- Mid-term: Report should include the problem statement, approach considered, and the novelty area identified.
- End-term: The final report should cover the final deliverables, including the solution, implementation, lessons learned, and improvements.

Evaluation Criteria

Midterm report (research + experiments) – 10%

- Clarity and depth of research and experiments
- Understanding and analysis reflected in the report

Novelty of use-case – 20%

- Originality and innovation
- Requires high preparation vs. available examples
- Improvements over existing solutions

Technical Implementation – 35%

- Efficiency of RAG solution and retrieval logic
- Effective use of Pathway's dynamic data handling
- Cost and efficiency metrics of the agentic RAG system
- Resilience to Failures: Ability to manage service failures by switching to alternative services and handling callback failures

Solution utility & simplicity of architecture – 20%

- Robustness of the architecture and components
- User experience and UI (if applicable)
- Integration with data sources via Pathway

End-term report – 15%

- Completeness of research and results
- Documentation of the technical solution
- Lessons learned and implementation improvements
- Quality of presentation

References

Below are some resources to help you get started:

How to deploy agents with Pathway?
Here you will see how you can build custom endpoints using Pathway RAG classes. There are two ways to serve agents: using the serve_callable API (which is easier to manage and recommended) or with an external web server like FastAPI.
If you prefer, you can start with an external web server and move the endpoint to Pathway later.

What type of agents should I use?
You are free to use any agent pattern or a mix of different patterns. We've shared resources from LangGraph and LangChain below. If you're using LangGraph or LangChain, make sure to use Pathway's Vectorstore through LangChain and Pathway for deploying your RAG application to the web server.
The same applies to LlamaIndex, where you should use Pathway Reader and Pathway Retriever. You can also explore or get inspiration from recent research, such as CRAG (https://arxiv.org/abs/2401.15884)

Additional Links and Resources:

- What is RAG: Beginner Blog on Pathway | Video by IBM
- What is Agentic RAG: CodeBasics Tutorial | Tutorial by IBM
- Pathway Developer Documentation: Link to Pathway Developer Docs
- Pathway App Templates
- Adaptive RAG to reduce costs without comprising accuracy in RAG
- Basic RAG with Open AI models (will be soon available on pathway.com as a part of Introductory bootcamp releasing on 22nd October. Make sure you stay tuned to register for it when it opens up)
- Basic RAG with Gemini models
- UI component for RAG App example, using Streamlit: RAG with Google Drive
- Sample Project READMEs (good to have) for Reference
  - https://github.com/pathwaycom/pathway/blob/main/examples/notebooks/showcases/mistral_adaptive_rag_question_answering.ipynb
  - https://github.com/leabuende/mike-llm-slack-plugin/
  - https://github.com/abdul756/AURA
  - https://github.com/AnavAgrawal/AlgoAce
  - https://github.com/Paulescu/virtual-assistant-llm
- https://pathway.com/developers/api-docs/pathway-xpacks-llm/question_answering#pathway.xpacks.llm.question_answering.AdaptiveRAGQuestionAnswerer.serve_callable
- https://python.langchain.com/docs/integrations/vectorstores/pathway/
- https://pathway.com/developers/templates/langchain-integration
- https://docs.llamaindex.ai/en/stable/examples/workflow/corrective_rag_pack/
- https://blog.langchain.dev/agentic-rag-with-langgraph/
- https://github.com/langchain-ai/langgraph/blob/main/examples/rag/langgraph_agentic_rag.ipynb
- https://github.com/langchain-ai/langgraph/blob/main/examples/rag/langgraph_crag.ipynb

- Leverage Gen AI wisely. If you see difficult-to-comprehend error messages, the least you should do is ask the query on Bard / Bing AI search, etc.
- For faster resolution, it is best to identify where the gap is and ask relevant doubts on the Inter IIT Discord Server.