Live Social Analyst

The "Live AI" Engine for Real-Time Global Pulse

Pathway Datathon Submission

Generated on: 2026-01-20 19:07

Live Social Analyst - Project Documentation

1. Documentation & Architecture

High-Level Architecture:
Live Social Analyst is a real-time Retrieval-Augmented Generation (RAG) system powered by Pathway. It ingests live data from thousands of sources, processes it incrementally, and enables an LLM to answer questions about events happening right now.

Core Components:
1. Inputs:
   - 34 OPML Files (1000+ RSS Feeds)
   - NewsData.io API
   - GNews API & HackerNews
   - Twitter/X API (Ready)

2. Pathway ETL Engine:
   - Ingests streams in real-time
   - Deduplicates identical stories
   - Normalizes text and metadata
   - Managing consistency and out-of-order data

3. Storage & Retrieval:
   - Hybrid Vector Store: In-memory live buffer for 0-latency access
   - SQLite Archive: For historical persistence and complex filtering
   - RAG Pipeline: Finds relevant context based on user queries

4. AI & Inference:
   - LLM: Gemini 1.5 Flash (Primary) / Groq Llama 3 (Fallback)
   - Generates answers using the retrieved real-time context

Deep Dive: The Life of a Data Point (Micro-Architecture):
1. T+0ms: News is published (RSS/API).
2. T+10ms: Pathway Connector (`pw.io`) detects the change and appends a row to the input Table.
3. T+15ms: The Streaming Engine propagates the update.
   - Deduplication Check: Global state checks if this URL hash was seen recently.
   - Normalization: HTML stripping and text cleaning (stateless transform).
4. T+100ms: Embedding: Using `pw.xpacks.llm`, the text is vectorized.
5. T+120ms: Indexing: The vector is upserted into the KNN index.
6. T+200ms: Ready for Query: The data is now live and retrievable by the RAG system.

2. Pathway Integration: Connectors & xPack

1. Live Data Ingestion with Pathway Connectors (pw.io):
   We utilize Pathway's real-time connector ecosystem to handle diverse data streams:
   - Custom Python Connectors (pw.io.python.read): We implemented a custom OPML connector that acts as an infinite stream of XML data, normalizing it on-the-fly.
   - API Connectors: Custom class connectors for NewsData.io and Twitter/X treating APIs as infinite tables.

2. Streaming Transformations (pw.temporal & pw.state):
All feature engineering happens in streaming mode:
- Incremental Deduplication: Using pw.io.deduplicate(pathway.table) to merge identical stories from different feeds.
- Sliding Windows: using pw.temporal.sliding to group high-velocity news events by 5-minute intervals to detect "breaking" clusters.

3. LLM Integration (pw.xpacks.llm):
We leverage Pathway's LLM xPack to orchestrate the RAG pipeline:
- pw.xpacks.llm.embedders: Live embedding of incoming text stream.
- pw.xpacks.llm.retrievers: The vector index is updated incrementally, allowing the LLM to query data milliseconds after ingestion.

3. Scalability, Observability & Extension

Scalability:
- Rust-Based Engine: Pathway handles high-throughput streams (thousands of events/sec) efficiently on a single node.
- Distributed Ready: Although running locally for the demo, Pathway pipelines can scale to clusters using Docker/Kubernetes without code changes.

Observability:
- Real-time logs track ingestion rates and LLM latency.
- "Global Pulse" UI visualizes the ingestion heartbeat.

Extension to Other Domains:
- Finance: Monitor stock tickers and news for algorithmic trading signals.
- Supply Chain: Track shipping disruptions and weather events in real-time.
- Cybersecurity: Analyze log streams for anomaly detection.

4. Example Scenario: "The Market Crash"

Scenario:
A major tech company announces a surprise acquisition at 10:00 AM.

1. T+0s (Announcement): The press release hits a news wire.
2. T+2s (Ingestion): Our OPML ingestor detects the new item.
3. T+3s (Processing): Pathway normalizes the text and updates the vector index.
4. T+5s (User Query): User asks "What just happened with [Company]?"
5. T+6s (Answer): The system retrieves the article ingested 3 seconds ago and generates: "Breaking: [Company] just announced an acquisition of..."

Contrast with Traditional RAG:
A standard RAG system would wait for the nightly scrape (12-24 hours late), answering "I have no recent news on this."

Live Social Analyst - Project Documentation

5. Demonstration Pipeline

The running demo illustrates:
1. "Global Pulse" Sidebar: Visualizing the raw stream of incoming news (updated every 5s via /pulse endpoint).
2. "Fetch Live" Button: Manually triggering a high-speed "Burst Mode" ingestion.
3. Chat Interface: Asking questions like "What is the latest on Apple?" and getting answers citing articles from 5 minutes ago.
4. Source Attribution: Every answer cites the specific RSS feed and timestamp, proving freshness.

6. Testing & Evaluation

Core validations performed:
1. Freshness Test: Verified that /pulse endpoint returns articles with timestamps < 5 minutes old.
2. Duplication Test: Confirmed that identical headlines from different sources are deduplicated.
3. Latency Test: Measured end-to-end time from "Ingest" to "Queryable" is < 2 seconds.
4. Load Test: Successfully handled ingestion of 1000+ feeds in "Burst Mode" without crashing.

7. Team Presentation Summary

Concept:
"Live Social Analyst" - A move from Static AI to Live AI.

Architecture:
A seamless pipeline: RSS/API -> Pathway Engine -> RAG -> Gemini.

Key Challenges:
- Ingesting noisy RSS feeds (solved with cleaning logic).
- Handling "Time": RSS feeds have inherent delays; we built a hybrid sorting mechanism to prioritize true publication time.
- Deduplication: Preventing the same AP/Reuters story appearing 50 times (solved with Pathway).

Why It Matters:
In a 24/7 world, static knowledge is a failure. This architecture proves that "Live AI" is accessible, scalable, and essential for decision-making.