



Traffix: The Traffic Insight Agent

Code: <https://github.com/bzgold/TRAFFIX>

Loom Video: <https://www.loom.com/share/f0df205ee7614b4c9fd32046a5904707?sid=94f7a823-0993-480d-801d-8c9ca0eca1ae>

Task 1: Defining your Problem and Audience

Problem:

Traffic analysts and transportation planners can easily observe how congestion patterns shift across corridors and time periods, but they still struggle to quickly uncover the underlying causes behind those changes.

Why this is a problem:

Traffic analysts and transportation planners need to understand not just what congestion patterns look like, but why they change over time. Although systems like RITIS and INRIX provide detailed data on travel times, bottlenecks, and incidents, identifying the underlying causes behind unusual spikes or drops still requires extensive manual work. Analysts must cross-reference multiple sources — including incident logs, news reports, event calendars, weather data, and even social media — to assemble a complete picture. This process is repetitive, time-consuming, and occurs almost daily, leaving limited time for higher-value tasks such as planning, forecasting, and improving operations.

Manual investigations are also inconsistent and prone to missing critical context. Linking current congestion patterns to historical events, long-term trends, or overlooked incidents is challenging, often leading to delayed or incomplete insights. As a result, analysts struggle to provide timely, accurate, and actionable explanations for congestion changes — insights that are essential for leadership decisions, performance reporting, and effective transportation management.

Audience and Example User Questions:

Transportation analysts, planners, and operations managers at DOTs (Department of Transportation), private sector organizations (i.e. Waze, toll road operators, Google, etc.) or city agencies — professionals who regularly work with dashboards and performance reports, but do not have the time to investigate every traffic anomaly. They frequently face pressure to explain traffic changes to leadership or stakeholders.

Example user questions:

- Why was congestion higher or lower than normal today?
- What major incidents occurred in my region this week?
- Did weather or other events affect travel time reliability today?
- Can you summarize this week's mobility highlights in a report suitable for leadership?

Task 2: Propose a Solution

Proposed Solution:

Traffic is an AI-powered storytelling assistant for transportation analytics, designed to help traffic analysts, transportation planners, and operations managers understand why congestion patterns change. Initially, Traffic integrates structured traffic data from RITIS (Regional Integrated Transportation Information System) with unstructured sources such as news articles and incident summaries, providing a comprehensive view of both daily and historical congestion causes and events. By consolidating multiple data sources, Traffic reduces the manual effort analysts spend investigating spikes or dips, enabling a faster understanding of patterns and anomalies.

The application supports multiple modes of use. Analysts can ask targeted questions, generate concise daily summaries with Quick Mode, or create in-depth research reports with Deep Mode that explore causes, contributing factors, and historical context. Traffic’s agents automatically retrieve, correlate, and synthesize information — highlighting notable patterns or events and producing evidence-based narratives that can be exported or emailed. For example, recurring congestion in a corridor might be identified and analyzed to reveal that frequent accidents are the primary cause, informing mitigation strategies such as improved signage. By automating data collection, analysis, and reporting, Traffic saves time, ensures consistency, and enables faster, more informed decision-making. Future enhancements will focus on integrating internal data sources, expanding API connections, and deepening analytical capabilities.

Technology Stack and Tools:

These tools and frameworks are used in the Traffic proof of concept (POC) and are subject to refinement or expansion in the final demo.

Layer	Tool	Purpose / Why
LLM	OpenAI GPT-4o	Provides strong reasoning and narrative capabilities, making it good for both quick summaries and deep mode research reports while being more lightweight.
Embedding Model	text-embedding-3-small	Despite being smaller than other models, text-embedding-3-small is still powerful and well-suited for this POC, providing meaningful semantic understanding of traffic incidents and news while keeping costs and speed manageable. The plan is

		to migrate to a larger embedding model in the future for a more robust MVP
Orchestration	LangGraph/LangChain	Enables multi-agent coordination between research, writing, and editing tasks. LangGraph is focused on agent orchestration and reasoning flows, while LangChain provides building blocks for connecting LLMs with tools and APIs.
Vector Database	Qdrant	High-performance, open-source vector database, ideal for local RAG pipelines created in the POC.
Monitoring	LangSmith	Tracks agent behavior, reasoning steps, and token usage, enabling performance monitoring, comparison for correctness, and evaluation of other key metrics to guide improvements.
Evaluation	RAGAS	RAGAS is used because it provides a standardized framework for evaluating retrieval-augmented generation outputs, allowing consistent measurement of faithfulness, relevance, and contextual accuracy across different queries.
User Interface	Streamlit for POC & v0/Vercel frontend for demo day	Streamlit provides a fast, interactive prototype ideal for internal testing and iteration, while v0 deployed on Vercel enables a scalable, publicly accessible interface that supports multiple report modes and broader user access.

Agents and Reasoning:

Traffic leverages multiple specialized agents to automate the collection, analysis, synthesis, and evaluation of traffic data. Agentic reasoning allows each agent to make independent, context-aware decisions within its domain, coordinate with other agents, and adapt its actions based on query complexity and available data. This enables the system to handle simple daily summaries, complex, deep-dive research reports efficiently, and manual PDF review.

Agent	Role	Description
Supervisor Agent	Orchestrator	Analyzes user queries, determines complexity, assigns tasks, and routes work between Research, Writer, Editor, and Evaluator agents.
Research Agent	Analyst	Gathers and synthesizes structured and unstructured data from RITIS and news sources, identifying key incidents and their contextual causes. Future enhancements will incorporate additional data sources, including detailed weather information and social media.
Writer Agent	Storyteller	Converts research data into coherent, actionable narratives based on; generates concise daily summaries or detailed research reports.
Editor Agent	Copy & Context Checker	Refines content by ensuring factual accuracy, consistent terminology, clear readability, and an appropriate tone for a corporate or public setting.
Evaluator Agent	QA	Uses RAGAS-style heuristics to evaluate and improve pipeline quality, applied selectively during initial analyses to provide transparency.

Task 3: Dealing with Data

Data Sources:

For this POC, the following data sources and APIs were used for the workflow.

- **RITIS Events API (structured data)** – Provides real-time and historical traffic incidents, work zones, and events. Used to identify congestion patterns and root causes. To manage API rate limits, data is precompiled and stored in an in-memory CSV for efficient access.
- **Tavily Search API (unstructured data)** – Aggregates local news, weather updates, roadwork and transportation information to contextualize traffic events.
- **OpenAI API (LLM and embeddings)** - Generates reports, summaries, and narratives. Embeddings enable semantic search and retrieval-augmented generation.
- **LangSmith (monitoring)** - tracks agent performance, reasoning steps, and token usage to debug and optimize pipeline.
- **Cohere API (advanced retrieval)** - Provides enhanced semantic search capabilities and reranking for more accurate and context-aware information retrieval

Chunking Strategy:

Since this is a POC, this baseline method was chosen to split documents into semantic chunks of 600 characters with a 100-character overlap. It's a straightforward approach that works well for the types of documents being processed, providing enough context for meaningful analysis while keeping retrieval efficient. The overlap helps maintain continuity across chunk boundaries, and structured RITIS event data is grouped by region and time periods to preserve situational context and relationships between related incidents. This setup allows both structured and unstructured data—from traffic incidents to news articles—to be analyzed effectively, enabling coherent, context-aware retrieval and reporting. While simple, it provides a solid starting point for testing and can be adjusted or refined as the system evolves and more data sources are incorporated.

Task 4: Building an End-to-End Agentic RAG Prototype

For the POC, Traffix is a locally hosted multi-agent RAG (Retrieval-Augmented Generation) system that leverages LangGraph for orchestration, Qdrant for semantic vector storage, and Streamlit for user interaction.


The system supports three main use cases:

- Quick Summary Mode – generates concise, one-paragraph daily digests suitable for leadership or rapid updates.
- Deep Research Mode – produces comprehensive reports analyzing causes, contributing factors, and historical trends behind congestion events.
- PDF Reader Mode – allows users to upload documents and ask questions; responses are strictly derived from the uploaded text.

Task 5: Creating a Golden Test Data Set

Created a ground truth test set using a report from the Federal Highway Administration (“Traffic Congestion and Reliability: Trends and Advanced Strategies for Congestion Mitigation”) to evaluate the retriever in the multi-agent workflow. The report examines the causes, measurement, and consequences of traffic congestion and unreliable travel, and explores strategies for mitigating their impacts. It highlights how congestion affects travel times, economic efficiency, safety, and quality of life, while outlining methods to monitor, manage, and improve highway system performance. Example questions are provided below and have also been saved in a CSV for future reference.

3. What role does the FHWA play in monitoring urban congestion and its impacts on transportation?
 4. How do traffic incidents contribute to traffic congestion and what strategies can be implemented to mitigate their effects on urban mobility?
 5. What role do traffic incidents play in contributing to traffic congestion and how can effective monitoring help mitigate these issues?

 Saving questions to CSV...
 Saved to: traffic_synthetic_questions.csv

	user_input	reference_contexts	reference	synthesizer_name
0	What are the primary causes of congestion on t...	[Traffic Congestion and Reliability: Trends, \...	Research identifies seven primary sources of c...	single_hop_specific_query_synthesizer
1	What are the impacts of work zones on traffic ...	[speeds and increasing delays. These disruptio...	Work zones are construction activities that re...	single_hop_specific_query_synthesizer
2	What role does the FHWA play in monitoring urb...	[Travel time is the fundamental metric for con...	The FHWA aggregates speed and volume data from...	single_hop_specific_query_synthesizer
3	How do traffic incidents contribute to traffic...	[<1-hop>\n\nefficiently not only restores traf...	Traffic incidents, such as crashes, breakdowns...	multi_hop_specific_query_synthesizer
4	What role do traffic incidents play in contrib...	[<1-hop>\n\nefficiently not only restores traf...	Traffic incidents, such as crashes and breakdo...	multi_hop_specific_query_synthesizer
5	What are the primary causes of traffic congest...	[<1-hop>\n\nefficiently not only restores traf...	Traffic congestion is primarily caused by a co...	multi_hop_specific_query_synthesizer


Assessment Results:

In the multi-agent workflow, a naive retrieval was used as a baseline approach with basic semantic chunking. This makes sense because it provides a simple, predictable reference point to evaluate the workflow's performance. Using this straightforward method allows the system to test functionality with minimal processing, and the results can later be compared to more advanced retrieval strategies to measure improvements in accuracy and efficiency.

Metrics used to examine the pipeline through RAGAS were as follows:

- Context precision - Measures how much of the retrieved context is relevant (quality of retrieved info).
- Context Recall – How much relevant information from the source documents was retrieved.
- Faithfulness – How much of the generated answer is supported by the retrieved context.
- Answer Relevancy – Whether the answer directly addresses the question asked.
- Context Entity Recall – The fraction of key entities from the source documents that appear in the answer or are retrieved.

Outputted results of Naïve RAG:

Retriever	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Context Entity Recall
Naïve	1.000 	0.925	0.87	0.954	0.368

Conclusion of Performance:

For the Naïve retriever, the results indicate strong precision and reliability but moderate coverage of key information. With a context precision of 1.000, all retrieved information was relevant, showing high quality retrieval. Context recall of 0.925 means most relevant information was captured, though not everything. Faithfulness at 0.954 shows that the generated answers were highly supported by the retrieved context. Answer relevancy of 0.870 indicates the answers generally addressed the questions well, and context entity recall of 0.368 reveals that only about a third of key entities were included, suggesting some gaps in detail coverage. Overall, the Naïve retriever is accurate and reliable but misses some specific entities.

Task 6: The Benefits of Advanced Retrieval

The RAG system initially uses a naïve embedding-based retriever, which serves as a baseline to capture semantic similarity between document chunks and user queries. To improve retrieval performance, the following techniques were tested in the same environment:

- **BM25 Retriever** – A keyword-based retriever that excels at finding exact matches and relevant terms. BM25 complements semantic embeddings when precise text matching is important (key terms).
- **Multi-Query Retriever** – Generates multiple reformulations of a user query, covering different ways a question might be asked. This approach improves recall for complex queries across the document set and helps with varied userbase.
- **Parent Document Retriever** – Links smaller text chunks to their parent documents, enabling context-aware retrieval that preserves the overall structure of longer documents. This is could particularly be useful for analyzing incident reports or detailed requirements.
- **Contextual Compression Retriever (Cohere Rerank)** – Ranks retrieved chunks based on relevance to the query, improving precision by filtering out less useful content and reducing noise in the answers.
- **Ensemble Retriever** – Combines multiple retrievers (e.g., BM25, naïve embeddings, and multi-query) using weighted voting, balancing speed, recall, and relevance for more robust retrieval performance.

Each method provides a unique perspective on document similarity and relevance, allowing the system to flexibly adapt to different types of analysis or reporting needs. Following section shows outcomes of testing the different techniques.

Task 7: Assessing Performance

Multiple retriever comparison to baseline Naive, using text-embedding-3-small

Retriever	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Context Entity Recall
Naïve	1.000 ✓	0.925	0.87	0.954	0.368
BM25	1.000 ✓	0.8	0.877 ✓	0.933	0.29
Multi-Query	1.000 ✓	0.917	0.87	0.976	0.425 ✓
Parent Document	1.000 ✓	0.825	0.87	0.874	0.295
Contextual Compression	1.000 ✓	0.892	0.87	0.945	0.314
Ensemble	1.000 ✓	0.950 ✓	0.869	0.980 ✓	0.335

	Retriever	Latency (s)	Questions
1	BM25	110.129075	10
0	Naive	122.944868	10
3	Parent Document	124.400105	10
4	Contextual Compression	128.307050	10
2	Multi-Query	149.312663	10
5	Ensemble	173.492774	10

Multiple retriever comparison to baseline Naive, using text-embedding-3-large

Retriever	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Context Entity Recall	Latency (s)
Naïve	1.000 ✓	0.975	0.852	0.964	0.485	155.6
BM25	1.000 ✓	0.93	0.863 ✓	0.901	0.223	123.2 ✓
Multi-Query	1.000 ✓	1.000 ✓	0.853	0.953	0.501 ✓	176.8
Parent Document	1.000 ✓	0.95	0.858	0.948	0.156	106.9
Contextual Compression	1.000 ✓	0.975	0.855	0.992 ✓	0.503 ✓	139.3
Ensemble	1.000 ✓	0.95	0.855	0.964	0.408	179.9

Results:

In the first table using the small embedding model, the Naive retriever served as the baseline with perfect context precision (1.000), moderate context recall (0.925), and moderate entity recall (0.368). Compared to this baseline, BM25 was faster (110.1s vs. 122.9s) and slightly improved answer relevancy (0.877 vs. 0.87) but at the cost of lower context recall (0.8) and entity coverage (0.29). Multi-Query and Ensemble offered higher context recall (0.917 and 0.950, respectively) and higher faithfulness (0.976 and 0.980), with Multi-Query also improving entity recall (0.425), though both had higher latency (149.3s and 173.5s). Contextual Compression increased faithfulness (0.945) with modest gains in context recall (0.892), while Parent Document lagged in recall (0.825) and entity coverage (0.295), showing that more complex strategies trade speed for information completeness.

In the second table using the large embedding model, overall context recall, and entity recall improved across most retrievers. The Naive baseline itself reached 0.975 context recall and 0.485 entity recall. Multi-Query achieved perfect context recall (1.000), and highest entity recall (0.501), while Contextual Compression also improved entity recall to 0.503 and faithfulness to 0.992. BM25's recall improved to 0.93 from 0.80, though its entity coverage remained low (0.223), enforcing the speed-over-depth trade-off. Ensemble maintained high context recall (0.95) and faithfulness (0.964), though latency increased slightly to 179.9s. Comparing small and large embeddings, large embeddings consistently enhanced context recalls and entity coverage, particularly for Multi-Query and Contextual Compression, at the cost of higher latency for the more complex retrievers.

LangSmith Trace Example: Demonstrates the workflow in action, showing how data moves throughout the system while the RAGAS evaluation is taking place.

Future Modifications

In the second half of the course, I plan to make improvements to my application to enhance usability, functionality, and overall user experience:

1. **Expand Data Sources:** I want to integrate more sources of information, such as social media feeds, public datasets, and additional domain-specific reports, to provide more comprehensive insights and richer context for users.
2. **Front-End Enhancements:** I aim to create a cleaner, more intuitive front-end, moving from a prototype interface (Streamlit) to something that feels production-ready (v0). This includes better layout, responsive design, and clearer navigation for all features.
3. **Improved Transparency & Reasoning:** I plan to implement more robust chain-of-thought reasoning and explainability features, like confidence, so users can see how conclusions are reached. This will build trust in the outputs and help users understand the system's logic.
4. **Document Comparison & Analysis:** I want to add the ability to compare multiple documents or PDFs side-by-side, highlighting differences, trends, and key takeaways. For example, a user could load a requirements document and quickly see if another document meets those criteria for a specific incident. This feature will make research and analysis faster, more efficient, and easier to understand on a more ad-hoc use.
5. **Interactive Report Chatbot:** I want to create a chatbot interface that allows users to ask questions and interact with reports generated by the system. This conversational layer will make the application more engaging and practical.
6. **Optimization & Efficiency:** I plan to optimize the application to run faster and more efficiently, by trying to reduce token usage in LLM interactions and improving backend processing times.

7. **Customizable Outputs:** I want to allow users to customize report formats, summaries, and visualizations to better fit their specific needs. For example, having it from the viewpoint of a leader verse an analyst.
8. **RAG:** Use different retriever methods based on performance scores and the specific use case.
9. **Embeddings:** Switch to a larger embedding model where it makes sense to improve semantic accuracy.