1 Context

The vehicle-ownership rate keeps increasing since the first vehicle was invented in 1886, bringing convenience but unfortunately also an increasing number of accidents. According to the National Highway Traffic Safety Administration (NHTSA) Fact 2022, one crash occurs every five minutes on average. Among all crashes, 39,785 were fatal, meaning one life is lost every 11 minutes due to transportation accidents [2]. It is urgent to reduce accidents and analysing them through transportation accident reports is a good approach.

A transportation accident report summarises information about the accident. It usually includes both textual and visual data of infrastructure, environment and traffic conditions of the road while that traffic accident happened [2]. These information enables a comprehensive understanding of factors of accidents, moving beyond simplistic classifications of causes of accidents. They also enhance decision-making by providing actionable insights. Professionals are enables to make data-driven decisions, such as infrastructure improvements or targeted safety interventions to reduce accident frequency and severity.

By leveraging detailed data through artificial intelligence (AI) and Geographic Information System (GIS), the reports support hotspot identification, severity prediction and policy formulation, significantly reducing road deaths [4].

1.1 Current Developments in Accident Report Analysis

To analyse massive accident reports in a short time, introducing computer programmes is for more effective than reading them one by one manually.

However, when transportation accident reports are mostly written in textual form such as “A car cross the road and hit a truck head-on” which computers struggle to understand due to unstructured textural expressions like “head-on”, even though these expressions are intuitive and non-metaphorical from human perspective. Programs prefer structured data in numerical form, for example, “economic loss: 10k HKD; individuals involved: 2; cars involved: 2, etc.”. As a result, various artificial intelligences are developed to process those unstructured reports to structured data for further analysis.

Currently, accident analysis is based on Natural Language Processing (NLP). It is a branch of AI that enables computers to understand, interpret, and generate human language in a meaningful way. It encompasses techniques for tasks such as text analysis, entity recognition and relationship extraction form unstructured data. NLP is integrated with knowledge graphs (KGs) to process complex textual narratives from reports, identifying causal factors and building structured representations for safety analysis [1]. Yet, NLP highly depends on annotated data, causing inefficiency when dealing with long and ambiguous text, as well as fragmented and unstructured data, which requires knowledge graphs (KGs) for integration.

1.2 Knowledge Graphs (KGs)

Knowledge graphs (KGs) are a new research method of data mining and are still in the exploratory research. They illustrate the relationships of independent entities in the objective world, like people involved in an accident and location of the accident occur, forming a “entity-relation-entity” triple. KGs represent the triple by nodes and edges, which are entities and relationships respectively. It

organizes complex and scattered information in accident reports into a semantic network and visualises them intuitively for analysis and safety management.

1.3 Integrating Claude 3.5 and KGs for Advanced Accident Analysis

Claude, 3.5, a Large Language Model (LLM), is utilized for accident analysis in this project. It includes few-shot learning and self-judgment mechanism. The LLM employs Retrieval-Augmented Generation (RAG) that combines retrieving relevant information from a data source and generate prompts to enhance LLM's accuracy in question-answering tasks. For instance, questions such as "Is the above relationship assessment correct or incorrect?" are generated to verify its own output [3]. Comparing to NLP, this approach is expected to improve accuracy and efficiency in extracting information from textual accident reports.

KGs are constructed to provide comprehensive data supporting role in accident analysis and visualising all kinds of accident query results. The knowledge demands for traffic accident analysis are first determined form existing data sources, followed by information extraction of entities, relationship s and attributes. These elements are written into Neo4j graph database to construct KGs for traffic accidents. Based on those KGs, multidimensional visualization analysis of accident data is realized by Cypher, a query language [4].

1.4 Objectives

The primary research question is: "How effective are prompt-engineered LLMs in extracting entities and causal relations from unstructured transportation accident reports to populate a KG, compared to traditional methods?"

This project aims to explore the potential of LLMs in enhancing the efficiency and accuracy in extracting knowledge from transportation accident reports and constructing a KG for more informed decision-making in accident analysis and prevention.

To achieve the project goal, a semi-automated framework using LLM model Claude 4.5 to extract information from unstructured transportation accident reports would be developed. Afterwards, a KG would be constructed according to the structured accident data just extracted. Then, a question-answering system which acts as RAG would be created for enhancing LLM's accuracy in extracting data.

For advanced analysis, visual analysis and evaluation metrics would be process through the use of Neo4j, a graphic database, with a query language Cypher.

The detailed methodology would be introduced in Section 3.

1.5 Expected Outcome

This project is expected to create a domain-specific Traffic Accident Knowledge Graph (TAKG) which is able to present connections of various elements such as people, vehicles, environmental conditions, and road information, to the causes of transportation accidents. Thus, providing a solid data foundation for further analysis, for example, visualization of relationship between elements by querying Neo4j with Cypher and plotting the returned table as a line chart.

2 Literature Review

2.1 Literature Review on "Information Extraction of Aviation Accident Causation Knowledge Graph: An LLM-Based Approach"

Chen *et al.* [1] addresses the critical challenge of extracting structured causation data form unstructured aviation accident reports to enhance safety analysis. The authors constructed a domain-specific Aviation Accident Causation Knowledge Graph using a dataset of 5,363 accident reports from the Aviation Safety Network spanning 2000 to 2023.

Their methodology called "Claude-prompt" used LLM Claude 3.5 to automate the extraction of entities and relationships. They utilize prompt engineering with few-shot learning and self-judgment mechanism. This allows the LLM to learn from a small number of examples and self-correct, thus improving the accuracy and efficiency of extraction when dealing with complex unstructured accident reports.

This project adopts the same methodological framework to analyse traffic accident reports. The study provides a validated process of using LLM for "entity-relationship-entity" triples with storing them into a graph database.

This project aims to replicate the reference's success in overcoming the limitations of traditional NLP approach by replacing Claude 3.5 with its updated version, Claude 4.5.

2.2 Literature Review on "Construction and Application of Traffic

Accident Knowledge Graph Based on LLM”

Hou *et al.* [3] offer a solid framework for this project by integrating Retrieval-Augmented Generation (RAG) with the Neo4j graph database to handle complex accident data. RAG assists LLMs by retrieving relevant information from the data source to generate context-aware prompts, which significantly enhances the accuracy of the model in question-answering tasks. This approach benefits this project as traffic reports include complex information such as numerical entities, categorical data and textual contextual details. By anchoring the LLM’s outputs in retrieved data, the system effectively manages these diverse data types while reducing the likelihood of “hallucinations” in LLMs.

The use of Neo4j and its query language, Cypher also provides a critical structure to the project goal of constructing a reliable KG. The study demonstrates the use of precise Cypher queries, such as ‘MATCH (a:Accident_Case)-[:SURROUND_BY]->(e:Environment) RETURN a,e. LIMIT 20’, to extract specific subgraphs like environmental factors surrounding an accident, which enables targeted analysis.

3 Methodology

A benchmark is set up to compare the effectiveness of using LLMs and traditional method, NLP, in extracting entities and relations from unstructured transportation accident reports. The dataset we use are restructured manually and assumed correct to be used as a ground truth benchmark. After setting up the benchmark, the unstructured dataset is analysed by Claude 4.5 to extract entities and relations. At the same time, NLP would analyse the unstructured dataset by manually

defining and detecting specific text objects using supervised machine learning model.

During structuring the dataset manually to set up benchmark, the schema of existing entities and relations are defined. For example, existing entities includes Vehicles, Person, Location, and Weather; existing relations includes: (Person) – Driving -> (Vehicle) and (Vehicle) – Crashed_At -> (Location). By setting schema constraints, LLM creating hallucinate entities and relations can be prevented.

Afterwards, extraction of the dataset using two paths, LLM and NLP, will process at the same time. For the LLM path, the unstructured reports are feed to the remote Claude 4.5 platform, through the API set up in our local Python script. 3 to 5 examples of correct extractions are included in the prompt, as known as Few-Shot Prompt. The LLM model is then asked to output JSON formatted strictly according to our Schema, such that ensure it understands the rules of extraction. Examples provide in the prompt are like "'Motor Vehicle Driver, Male, 50, Lap & Shoulder Used.' is extracted as '(50 year olds Male)-[Drive]->(Motor Vehicle)'. ", and " 'This incident occurred in Lacey, Thurston, on the 005 route increasing milepost direction at milepost 0.44.' is extracted as '(Vehicle)-[Crashed_At]->(Lacey, Thurston, 005 route)'. "

For the NLP path, a supervised machine learning model is introduced in another local Python script. First, some labelling examples are provided to train the model to recognize entities in report text as various labels, such as "Motor Vehicle = Vehicle, Driver = Role" and "Male = Gender". Then, the nodes and relations for

building KG are defined by combining labels, for example a Person node is combined from labels Gender and Age, i.e. “Male = Gender” and “Age = 50” form node (50-year-olds Male). If there is Role label, a relation is created, i.e. (Person)-[Drive]->(Vehicle).

Comparing outputs of both path with the benchmark, 3 parameters which are true positives, false positives and false negative, can be found. True positives (TP) mean the number of triples the model extracted that exactly match the benchmark. False positives (FP) mean the number of triples the model extracted that do not exist in the benchmark. False negatives (FN) mean the number of triples in the benchmark that the model missed completely. They are used in calculation of evaluation metrics: precision, recall and F1-score. The evaluation metrics are introduced detailly in section 3.4.

Furthermore, triples extracted from the LLM path are exported and stored in graph database Neo4j AuraDB. Triples in query patterns are transformed into KG by the query language provided by Neo4j, Cypher. As a result, visualization and reasoning of accidents is achieved. The graph database Neo4j Aura DB is introduced detailly in section 3.1.2.

3.1 Development environment

The development environment of this project includes three main components: program code written in Python to interact with LLMs, graphic database Neo4j AuraDB to store entities and relationships for building a KG, and a LLM to analyse traffic accident reports.

3.1.1 Program Language: Python

Python is an interpreted and interactive programming language. It supports multiple paradigms, such as scripts for data processing, classes for abstractions and functional utilities for text processing [5].

Python 3.13.5 is used as the program language in this project since it is capable for processing traffic accident reports quickly through using its standard library, which covers tasks for text handling and networking for calling LLM APIs. Combined with third-party libraries, likes Neo4j driver, the pipeline of the project would be easily built.

3.1.1.1 Python Libraries

Pandas and numpy are introduced for data loading, processing and managing.

In order to connect with the Neo4j database, connectors such as py2neo or neo4j driver is introduced to the code.

Httpx and python-dotenv are installed for calling Claude 4.5 API and storing API key respectively.

Since RAG is used for retrieving the output of LLMs, transformers and json are necessary for creating vector embeddings to search data in KG.

Seqeval, scikit-learn and matplotlib are also installed for calculating evaluation metrics and comparing their results from path LLM and NLP.

3.1.2 Graph Database Neo4j AuraDB

Neo4j AuraDB is a fully managed cloud service for the Neo4j graph database [6]. As a result, its cloud instance can be connected in our Python scripts and push the program output directly to the cloud without local installation, saving storage area for our computer especially while the size dataset is huge.

Also, Neo4j AuraDB is highly optimized for vector search and graph retrieval [6]. Once the KG is constructed, our program can quickly query the cloud graph database to find relevant data according to user inputs. Therefore, it enhances the effectiveness of RAG by feeding precise and fact-based content to the LLM, further improve the accuracy of LLM in analysing traffic accident report.

3.1.2.1 Query Language: Cypher

Cypher is Neo4j's declarative and GQL conformant query language. It provides an intuitive and visual way of matching patterns and relationships in a format that similar to natural language we use in daily life [7].

Graphs built by Cypher is composed of nodes and relationships, also with assigned properties. Nodes are expressed in format of '(node)', while relationships are represented by '(node1) - [relationship] -> (node2)' [7]. The arrow in relationships indicate the direction between nodes.

With this query language, KGs in both simple and complex pattern are easily to construct using entities and relationships extracted from data.

3.1.3 Large Language Model: Claude 4.5

Claude 4.5 is the LLM used in this project because it preforms better in coding-heavy environments while present strong structured reasoning ability comparing other LLM models. Its few-shot learning ability enables users to specify the structure of output, such as forcing it to express its output in '(node1) -[relationship] -> (node2)' with providing just a few examples, reducing errors and improve output data quality.

3.1.3.1 API

Application Programming Interface (API) is a set of definitions and protocols that allow a software to communicate with another. Setting up an API in our local Python script allows content in traffic accident reports to be sent to a remote Claude 4.5 platform. The remote platform would be the Perplexity education version that includes tokens quota equivalent to $5.00 USD.

The following API key is stored in a ".env" configuration file for setting up local environment variable.

'PERPLEXITY_API_KEY = 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'

The environment variable with the API key is then loaded into the Python script:

```
'import os
from dotenv import load_dotenv
from perplexity import Perplexity

load_dotenv()
```

client = Perplexity() # Uses PERPLEXITY_API_KEY from .env file’

The interaction between local Python script and the remote Claude 4.5 is hence achieved by:

‘# This sends a "Request" to the API
response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a traffic safety expert."},
        {"role": "user", "content": "Extract the main cause of the accident: 'Car A ran a red light.'"}
    ]
)

Print the output
print(response.choices[0].message.content)’

3.2 Raw Data

The raw crash data used in this project were obtained from the HSIS and Google Maps. It includes 16,188 traffic accidents data from Washington State and 42,715 data from Illinois State [8]. Around 200 data from Washington and 200 data from Illinois are chose randomly as our candidate pool in this project for a manageable workload.

The data presents traffic accidents with a basic categorizing on crash date, time and location details, weather conditions. It also records the information of

vehicles involved, number of vehicles and pedestrians involved, direction of vehicle travel and any manoeuvres [8].

3.3 Evaluation Metrics

The outputs of the two paths are compared with the benchmark using precision, recall and F1-score.

Precision represents the accuracy of what was found in both benchmark and outputs. A higher precision means more accurate in identifying and extracting entities and relations.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall represents the coverage of the truth comparing outputs with the benchmark. A higher recall means more actual existing entities and relations are found by the model.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score balances precision and recall. A higher F1-score means a better performance in relation extraction.

$$F1 = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

4 Progress

4.1 Project Overview

The final year project is scheduled to be run from September 2025 to June 2026. Thus far, three months into the project, the overall framework and methodology

have been configured. The following lays out the progress of the project.

4.2 LLM Selection and Evaluation

During initial technical discussions in September 2025, our team evaluated finetuned AI-based approach and pre-trained AI-based approach for analysing transportation accident reports and building KGs. Finetuned AI-based approaches require our team to train an LLM to analyse unstructured crash events. It requires devices with high computing power to complete fine-tuning process. Pre-trained AI-based approach uses pre-trained LLMs that are capable of processing semi-structured data with few-shot learning.

This project employs pre-trained AI-based approach in considering the hardware limitations and workload of each team members. The manual structured data will serve as a benchmark that assumed correct, and LLM and NLP extraction results will be compared against this benchmark to evaluate the effectiveness of two paths.

Our team evaluated five LLMs based on reasoning capability, processing speed and few-shot learning requirements. Performance comparison between Claude 4.5, CPT-5.1, Gemini 3 Pro, Kimi K2 Thinking, and Sonar (Llama-3.170B) are taken by Datacamp.com and summarised in table 4.1.

Model	Reasoning Score	Key Strengths
Claude 4.5	84	Coding reasoning, structured outputs [10]
GPT-5.1	94	Complex synthesis, context tracking [11]
Gemini 3 Pro	86	Multimodal, nuanced

		analysis [12]
Kimi K2 Thinking	95	Research-grade, agentic orchestration [13]
Sonar (Llama-3.1 70B)	79	Factual logic, open-source deployment [14]

Table 4.1 Performance of 5 LLMs

The reasoning score of an LLM is tested by Humanity's Last Exam that tests the reasoning ability of the LLM using expert-level and closed-ended academic questions across multiple disciplines [9].

By comparing abilities of five LLMs, Claude 4.5 prioritizes coding and software development with careful reasoning, which is the best match of the project.

4.3 Data Availability

From the start of the project till November 2025, our team has identified several semi-structured and unstructured data sources. Hong Kong Traffic Injury Collision Database provides Hong Kong accident statistics yet lacks detailed textual descriptions. Additional outreach to Transport Department and Police Force are required for full accident narratives [15]. Considering the scale of this project and the feasibility of getting access for full accident reports, this data source is not chosen. Washington State Accident Data Repository provides semi-structured accident reports that objects related such as vehicles and environments are already categorized [16]. Although it lacks comprehensive and unstructured accident textual reports, considering the workload of team members and hardware limitations. It is chosen as our database.

4.4 API installation

After firming the LLM and data source, the LLM API access is set up in November 2025. Yet not all of the team members successfully integrate and run the API in their local development environment. It may be due to the differences of environment variables set in our local terminals. The process of setting and testing the API is still on-going.

4.5 Upcoming Steps and Timeline

Based on the progress made through November 2025, the project will follow this projected timeline for the remaining period.

From December 2025 to January 2026, all team members should have stable access to Claude 4.5 API and resolve local development environment issues. Also, the manual extraction for the data should started for preparing the benchmark.

From February 2026 to March 2026, the entities and relations extraction of LLM with few-shot prompting and traditional NLP should start. Then, the extracted entities and relations should be mapped into KG using graph database Neo4j AuraDB.

In April 2026, the accuracy and effectiveness of the LLM approach and NLP approach should be evaluated by comparing their KGs against the manual structured data benchmark using metrics precision, recall and F1-score.

From May 2026 to June 2026, the practical implications of traffic accident reports analysis using LLM and NLP should be concluded and finalized in the final

report.

4.6 Forecast

The project proceeds with a clear trajectory towards the August 2026 completion date, and the team remains assured in its capacity to execute the remaining phases of extraction and evaluation. In the rapidly evolving domain of intelligent transportation systems, where LLM like Claude 4.5 are redefining not only how we process unstructured text but also how we semantics and structure complex accident causality, the technological landscape shifts quickly and it is vital that we remain methodologically agile to integrate the latest advancements. This is particularly relevant in our upcoming phase of KG construction, where we must rigorously parse the Washington State accident narratives and validate extracted entities against our manual benchmarks to ensure the integrity of critical safety data. We anticipate technical hurdles in the comparative analysis of NLP and pre-trained LLM approaches, but as we transition from framework configuration to full-scale implementation, we are fully equipped to resolve these complexities and deliver a robust safety analysis framework.

5 References

SafeTraffic/data/IL/train/inj.csv at main · Puw242/SafeTraffic · GitHub

[1] L. Chen, J. Xu, T. Wu, and J. Liu, “Information Extraction of Aviation Accident Causation Knowledge Graph: An LLM-Based Approach,” *Electronics*, vol. 13, no. 19, p. 3936, Oct. 2024, doi: https://doi.org/10.3390/electronics13193936.

[2] Z. Fan *et al.*, “Learning Traffic Crashes as Language: Datasets, Benchmarks, and What-if Causal Analyses,” *arXiv.org*, 2024. https://arxiv.org/abs/2406.10789 (accessed Dec. 26, 2025).

[3] Hou, Y., Shao, Y., Han, Z., and Ye, Z., “Construction and Application of Traffic Accident Knowledge Graph Based on LLM,” SAE Technical Paper 2025-01-7139, 2025, doi:10.4271/2025-01-7139.

[4] L. Zhang et al., “Analysis of Traffic Accident Based on Knowledge Graph,” Journal of Advanced Transportation, vol. 2022, pp. 1–16, Aug. 2022, doi: https://doi.org/10.1155/2022/3915467.

[5] Python, “General Python FAQ — Python 3.8.1 documentation,” Python.org, 2020. https://docs.python.org/3/faq/general.html#what-is-python

[6] “Aura,” Graph Database & Analytics, Aug. 19, 2024. https://neo4j.com/product/auradb/

[7] neo4j, “Graph database concepts - Getting Started,” Neo4j Graph Data Platform, 2024, doi: https://doi.org/10423062553/neo4j-social-share-21.

[8] Y. Zhao, P. Wang, Y. Zhao, H. Du, and H. F. Yang, “SafeTraffic Copilot: adapting large language models for trustworthy traffic safety assessments and decision interventions,” Nature Communications, vol. 16, no. 1, Oct. 2025, doi: https://doi.org/10.1038/s41467-025-64574-w.

[9] M. Pedigo, “What is Humanity’s Last Exam? The AI Benchmark For Expert-Level Reasoning,” Datacamp.com, Dec. 18, 2025. https://www.datacamp.com/blog/what-is-humanitys-last-exam-ai-benchmark (accessed Dec. 26, 2025).

[10] M. Crabtree, “Claude Sonnet 4.5: Tests, Features, Access, Benchmarks, and More,” Datacamp.com, Sep. 30, 2025. https://www.datacamp.com/blog/claude-sonnet-4-5 (accessed Dec. 26, 2025).

[11] J. Waples and M. Crabtree, “GPT-5.1: Two Models, Automatic Routing, Adaptive Reasoning, and More,” Datacamp.com, Nov. 13, 2025. https://www.datacamp.com/blog/gpt-5-1 (accessed Dec. 26, 2025).

[12] M. Crabtree and J. Waples, “Gemini 3: Google’s Most Powerful LLM,” Datacamp.com, Nov. 19, 2025. https://www.datacamp.com/blog/gemini-3 (accessed Dec. 26, 2025).

[13] Bex Tuychiev, “Kimi K2 Thinking: Open-Source LLM Guide, Benchmarks, and Tools,” Datacamp.com, Nov. 10, 2025. https://www.datacamp.com/tutorial/kimi-k2-thinking-guide (accessed Dec. 26, 2025).

[14] R. Ong, “Reflection Llama-3.1 70B: Testing & Summary of What We Know,” Datacamp.com, Sep. 08, 2024. https://www.datacamp.com/tutorial/reflection-llama-3-1-70b (accessed Dec. 26, 2025).

[15] H. Kong, “香港車禍傷亡資料庫 | Hong Kong Traffic Injury Collision Database | Kenneth Wong - Cities and Maps,” Kenneth Wong - Cities and Maps, Nov. 2022. https://mappyurbanist.com/project/trafficcollision/ (accessed Dec. 26, 2025).

[16] Puw242, “GitHub - Puw242/SafeTraffic,” GitHub, Aug. 18, 2025. https://github.com/Puw242/SafeTraffic (accessed Dec. 26, 2025).