HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Civil and Environmental Engineering
CIVL 4910/4920 Civil and Environmental Engineering Final Year Project/Thesis
2025/26

Interim Report

Title of FYP: Construction and Application of Transportation Accident Knowledge Graph Based on Large Language Models (LLMs)

Name of Supervisor: Prof. Sisi Jian

Student Name: CHAH, Hoi Yan

Student Number: 20952833

Teammates: LO, Yat Long Timothy (20966949), WANG, Chi Yuen (20963533)

Date of Submission: December 29, 2025, 12:00am

Content

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, resulting in one death every 11 minutes due to transportation accidents [1]. Therefore, it is urgent to reduce accidents and analysing them through transportation accident reports is a good approach.

A transportation accident report summarises information about an accident. It usually includes both textual and visual information on the road's infrastructure, environment and traffic conditions at the time of the accident [1]. This information helps us understand factors of accidents comprehensively, moving beyond basic labels for why accidents happen. They also enhance decision-making by providing actionable insights. Professionals are enabled to make data-driven decisions, such as improving infrastructure or launching specific safety measures to reduce accident frequency and severity.

By using artificial intelligence (AI) and Geographic Information System (GIS) to analyse detailed data, the reports support hotspot identification, severity prediction and policy formulation, reducing road deaths effectively [2].

1.1 Current Developments in Accident Report Analysis

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

when transportation accident reports are mostly written in textual form such as “A car crosses the road and hits a truck head-on” which computers struggle to understand due to unstructured textual 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 involves techniques for tasks such as text analysis, entity recognition and relationship extraction from unstructured data. NLP is integrated with knowledge graphs (KGs) to process complex written descriptions from reports. It identifies contributing factors and builds structured frameworks for safety analysis [3]. Yet, NLP highly depends on annotated data, causing inefficiency in dealing with long and ambiguous text, along with unstructured data. Thus, knowledge graphs (KGs) are a must for data 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, such as people involved in an accident and location of the accident occur, forming an “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 4.5 and KGs for Advanced Accident Analysis

Claude 4.5, 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 pulling in relevant information from a data source and generating prompts to enhance LLM's precision in answering question. For instance, it creates questions like "Is the above relationship assessment correct or incorrect?" to verify its own outputs [4]. Compared 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 support in accident analysis and visualise all types of query results related to crashes. The knowledge demands for traffic accident analysis are first determined from existing data sources. Then, information including entities, relationships and attributes is extracted. These components are written into Neo4j graph database to form KGs for traffic accidents. Based on those KGs, multidimensional visual analysis of accident data is accomplished using Cypher, a query language [2].

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 final year 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 framework using LLM model Claude 4.5 to extract information from semi-structured transportation accident reports would be developed. Afterwards, a KG would be constructed according to the extracted entities and relations. 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 processed through the use of Neo4j, a graphic database with Cypher.

The detailed methodology is introduced in Section 3.

1.5 Expected Outcome

This project expects the effectiveness of analysing traffic accident reports of LLM approach should be higher than the traditional NLP approach. As the wordings of each text report are not the same, there is no established standard of wording. It is predicted that the NLP method may appear keyword omissions, resulting in classification errors.

This project also expects to create a traffic accident Knowledge Graph 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 relationships 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.* [3] address the critical challenge of extracting structured causation data from 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 [3]. 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.* [4] 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 [4].

3 Methodology

A benchmark is set up to compare the effectiveness of using LLMs and the

traditional method, NLP, in extracting entities and relations from semi-structured transportation accident reports. Our dataset is first analysed using NLP by manually defining and detecting specific text objects across a whole crash report. The outputs are assumed correct to be used as a benchmark. After setting up the benchmark, the dataset is analysed by Claude 4.5 to extract entities and relations.

The schema of existing entities and relations is first defined before extractions. For instance, existing entities include “Vehicles”, “Person”, “Location”, and “Weather”, while existing relations include “(Person) – Driving -> (Vehicle)” and “(Vehicle) – Crashed_At -> (Location)”. By setting schema constraints, not only can text objects be clearly defined and categorized, but LLMs creating hallucinated entities and relations can also be prevented.

Afterwards, the extraction using NLP is started for setting up the benchmark. Specific text objects in reports are recognized and categorized to various entities labels. For example, text “Motor Vehicle” is categorized as label “Vehicle”, text “Driver” is categorized as label “Role” and text “Male” is categorized as label “Gender”. Then, the nodes and relations for building KG are defined by combining labels, such as a “(Person)” node is combined from labels “Gender” and “Age”, i.e. “Male = Gender” and “Age = 50” form node “(50-year-old Male)”. If there is a “Role” label, a relation is created, i.e. “(Person)-[Drive]->(Vehicle)”. The resulting entities and relations serve as the benchmark which contribute to further comparison with outputs using LLM.

The extraction using LLM is then processed. The semi-structured reports are fed to the remote Claude 4.5, through the API set up in our local Python scripts. 3 to

5 examples of correct extractions are included in the prompt, also known as a Few-Shot Prompt. The LLM model is then asked to output JSON formatted strictly according to our Schema, such that it ensures it understands the rules of extraction. Examples provided in the prompt are like " 'Motor Vehicle Driver, Male, 50, Lap & Shoulder Used.' is extracted as '(50-year-old 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)'. "

Comparing outputs of LLM approach with the benchmark, 3 parameters which are true positives, false positives and false negatives, 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 in detail in section 3.3.

Furthermore, triples extracted from NLP and LLM are exported and stored in graph database Neo4j AuraDB. Triples in query patterns are transformed into KGs by the query language provided by Neo4j, Cypher. As a result, visualization and reasoning of accidents are achieved. The graph database Neo4j Aura DB is introduced in detail 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, graph database Neo4j AuraDB to store entities and relationships for building a KG, and an 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 of 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 the 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” are 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 LLM and NLP, a.k.a. the benchmark.

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 are 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 indicates the direction between nodes.

With this query language, KGs in both simple and complex patterns are easy 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 performs better in coding-heavy environments while presents strong structured reasoning ability compared 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)" by 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 allows software to communicate with another. Setting up an API in our local Python scripts allows content in traffic accident reports to be sent to a remote Claude 4.5 platform. The remote platform would be the Perplexity educational version that includes token 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 Python scripts:

"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 scripts and the remote Claude 4.5 is hence achieved by:

```
“# This sends a "Request" to the API
response = client.chat.completions.create(
    model="Claude 4.5",
    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 stored in GitHub and obtained from the HSIS and Google Maps. It includes 16,188 traffic accidents data records from Washington State and 42,715 data from Illinois State [8]. 500 to 1000 data from

the two states are chosen 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 LLM approach 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 NLP extraction results would serve as a benchmark that assumed correct, and LLM extraction results would be compared against the benchmark to evaluate its effectiveness.

Our team evaluated five LLMs based on reasoning capability, processing speed and few-shot learning requirements. Performance comparison between Claude 4.5, GPT-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 [9]
GPT-5.1	94	Complex synthesis, context tracking [10]
Gemini 3 Pro	86	Multimodal, nuanced analysis [11]
Kimi K2 Thinking	95	Research-grade, agentic orchestration [12]
Sonar (Llama-3.170B)	79	Factual logic, open-source deployment [13]

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 [14].

By comparing abilities of five LLMs, Claude 4.5 prioritizes coding and software development with careful reasoning, which is the best match for 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. One of them is Hong Kong Traffic Injury Collision Database. It provides Hong Kong accident statistics yet lacks detailed textual descriptions. Additional outreach to the Transport Department and Police Force is 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. Another one of them is the Washington and Illinois state accident data repository that is stored on GitHub. It provides semi-structured

accident reports in which related objects such as vehicles and environments are already categorized [16]. Although it doesn't provide unstructured accident textual reports, considering the workload of team members and hardware limitations, it is chosen as our database.

4.4 API installation

After confirming 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. The traditional NLP approach should start for setting up the benchmark.

From February 2026 to March 2026, the extraction of entities and relations using the LLM with few-shot prompting should be processed. 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 should be

evaluated by comparing its KG against the benchmark using metrics precision, recall and F1-score.

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

4.6 Forecast

The project proceeds with a clear trajectory towards the June 2026 completion date, and our team remains assured in its capacity to execute the remaining phases of extraction and evaluation. In the rapidly evolving domain of intelligent transportation systems, not only how we process unstructured text, but also how we handle 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 accident narratives and validate extracted entities against our benchmark to ensure the integrity of critical safety data. We anticipate technical hurdles in the comparative analysis of NLP and 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

[1] Z. Fan *et al.*, “Learning Traffic Crashes as Language: Datasets, Benchmarks, and What-if Causal Analyses,” *arXiv (Cornell University)*, Jun. 2024, doi: https://doi.org/10.48550/arxiv.2406.10789.

[2] 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.

[3] 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.

[4] Y. Hou, Y. Shao, Z. Han, and Z. Ye, “Construction and Application of Traffic Accident Knowledge Graph Based on LLM,” *SAE Technical Paper Series*, vol. 1, Feb. 2025, doi: https://doi.org/10.4271/2025-01-7139.

[5] Python Software Foundation, “General Python FAQ web page (What is Python?),” Python 3.14 documentation. https://docs.python.org/3/faq/general.html#what-is-python (accessed Dec. 26, 2025).

[6] Neo4j AuraDB Introduction, “Neo4j Aura DB: Fully Managed Graph Database,” Neo4j. https://neo4j.com/product/auradb/ (accessed Dec. 26, 2025).

[7] Graph Database Concepts, “Graph database concepts - Getting Started,” Neo4j Graph Data Platform. https://neo4j.com/docs/getting-started/appendix/graphdb-concepts/ (accessed Dec. 26, 2025).

[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. 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).

[10] 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).

[11] 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).

[12] B. 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).

[13] 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).

[14] 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).

[15] *香港車禍傷亡資料庫 | Hong Kong Traffic Injury Collision Database | Kenneth Wong - Cities and Maps*, Hong Kong Transport Department, Road Traffic Accident Statistics, Nov. 2022. [Online]. Available: https://mappyurbanist.com/project/trafficcollision/

[16] *GitHub - Puw242/SafeTraffic*, U.S. Department of Transportation, Federal Highway Administration, Highway Safety Information System (HSIS), Google Maps Static API Documentation, Aug. 18, 2025. https://github.com/Puw242/SafeTraffic (accessed Dec. 26, 2025).