Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale Ingegneria Informatica (Computer Engineering)

A.a. 2024/2025

Graduation Session October 2025

Exploring Domain-Adapted LLMs for Crash Narrative Information Extraction

From Stated Facts to Inferred Insights: Leveraging Open-Source LLMs for Structured Information Extraction in Traffic Safety

This thesis was conducted in collaboration with Chalmers University of Technology

Supervisors:

Flavio Giobergia

Jordanka Kovaceva (Chalmers)

Candidate:

Mattia Carlino

Abstract

Free-text crash narratives recorded in real-world crash databases have been shown to play a significant role in improving traffic safety. But they remain challenging to analyze at scale due to unstructured writing, heterogeneous terminology, and uneven detail. The development of Large Language Models (LLMs) offers a promising way to automatically extract information from narratives by asking questions. However, crash narratives remain hard for LLMs to analyze because of a lack of traffic safety domain knowledge. Moreover, relying on closed-source LLMs through external APIs poses privacy risks for crash data and often underperforms due to limited traffic knowledge. Motivated by these concerns, we study whether smaller open-source LLMs can support reasoning-intensive extraction from crash narratives, targeting three challenging objectives: the travel direction of the vehicles involved in the crash, identifying the manner of collision, and classifying crash type in multi-vehicle scenarios that require accurate per-vehicle prediction. In the first phase of the experiments, we focused on extracting vehicle travel directions by comparing small LLMs with 8 billion parameters (Mistral, DeepSeek, and Qwen) under different prompting strategies against fine-tuned transformers (BERT, RoBERTa, and SciBERT) on a manually labeled subset of the Crash Investigation Sampling System (CISS) dataset. The goal was to assess whether models trained on a generic corpus could approach or surpass the performance of domain-adapted baselines. Results confirmed that fine-tuned transformers achieved the best accuracy; however, advanced prompting strategies, particularly Chain of Thought, enabled some LLMs to reach about 90% accuracy, showing that they can serve as competitive alternatives. For the second and third tasks, to bridge domain gaps, we apply Low-Rank Adaption (LoRA) fine-tuning to inject traffic-specific knowledge. Experiments on the CISS dataset show that our fine-tuned 3B models can outperform GPT-4o while requiring minimal training resources. Further analysis of LLM-annotated data shows that LLMs can both compensate for and correct limitations in manual annotations while preserving key distributional characteristics. The results indicate that advanced prompting techniques and fine-tuned open-source models prove effective in large-scale traffic safety studies.

Acknowledgements

I would like to express my sincere gratitude to my supervisor at Politecnico di Torino, Prof. Flavio Giobergia, whose guidance and valuable suggestions have been essential in shaping and strengthening this thesis.

This thesis was conducted in collaboration with Chalmers University of Technology. I am deeply thankful to my Chalmers supervisor, Prof. Jordanka Kovaceva, for her continuous supervision, valuable insights, and constructive feedback throughout the entire research process.

Finally, I gratefully acknowledge the close collaboration with Xixi Wang, with whom this thesis was jointly developed.

Table of Contents

List of Tables

List of Figures

List of Acronyms

API

Application Programming Interface

BERT

Bidirectional Encoder Representations from Transformers

CISS

Crash Investigation Sampling System

CoT

Chain of Thought

CSV

Comma-Separated Values

F1

F1-score (Harmonic Mean of Precision and Recall)

GPT

Generative Pre-trained Transformer

JSON

JavaScript Object Notation

LLM

Large Language Model

LoRA

Low-Rank Adaptation

MLM

Masked Language Model

NLP

Natural Language Processing

QA

Question Answering

RL

Reinforcement Learning

XLSX

Excel Spreadsheet File Format

Chapter 1

Introduction

1.1 Background and motivation

Today, traffic crashes remain one of the leading causes of death worldwide, with an estimated 1.19 million fatalities annually [1]. Improving road safety is therefore a critical global challenge. A key approach to addressing this issue is through analyzing real-world data recorded in crash databases. These databases contain information about the crash, people involved, injuries and crash narratives, documented by crash investigators as part of official reports. Among these records, crash narratives are especially valuable, as they include detailed crash scenarios that go beyond structured variables, such as vehicle travel directions, impact points, and other contextual factors. Despite their value, crash narratives present significant challenges for direct analysis. They are typically written in unstructured text with highly diverse styles, inconsistent terminology, and varying levels of detail. As a result, these narratives cannot be readily analyzed using conventional statistical techniques. Researchers or transport authorities have traditionally addressed this by manually preprocessing the text into structured formats such as tables for statistical analysis. This process is not only labor-intensive but also error-prone, especially when working with large datasets.

However, advances in Large Language Models (LLMs) offer a promising alternative: by efficiently extracting key information and identifying implicit crash patterns, LLMs can automate and scale up the analysis of crash reports.

While LLMs show great potential in automating information extraction, their performance in specialized domains such as traffic safety remains limited. Most publicly available LLMs are trained on general-purpose web-scale corpora, which lack the domain-specific vocabulary and knowledge found in crash reports. Consequently, these models may struggle to understand the nuances of safety-critical narratives and fail to generate accurate or reliable responses to domain-specific

queries. This limitation raises several important research questions. First, for straightforward extraction tasks where information is often explicitly stated in crash narratives, can zero-shot or prompting-based LLMs compete with or even outperform traditional approaches such as fine-tuned BERT transformers? Second, for more complex reasoning-intensive tasks, where critical information must be inferred rather than directly extracted, can fine-tuning open-source LLMs on domain-specific crash data significantly improve their ability to capture implicit details and enhance their reliability for real-world deployment?

To explore this, our study uses crash report summaries from the Crash Investigation Sampling System (CISS) dataset [2], provided by the National Highway Traffic Safety Administration (NHTSA). We evaluate whether prompting techniques, as well as fine-tuning open-source LLMs on this dataset, can improve their ability to extract crash details and pattern and enhance their reliability in accident-related information extraction tasks. We evaluate the effectiveness of prompting techniques alone for simpler tasks and of fine-tuning open-source LLMs on this dataset in improving their capacity to extract missing information and to increase reliability in accident-related information extraction tasks.

1.2 Introduction to large language models

1.2.1 Development of large language model

A Large Language Model is a type of artificial intelligence model designed to comprehend and generate human language. These models are trained on vast corpora of text data, enabling them to learn the intricate patterns and structures inherent in language. ChatGPT [3], a prominent example of an LLM, is specifically trained to facilitate engaging conversations and respond to a wide range of queries. LLMs are distinguished by their massive scale, often comprising billions of parameters, which enable them to capture complex linguistic patterns. Typically, these models are built upon deep learning architectures, particularly transformers, which empower them to achieve state-of-the-art performance across various Natural Language Processing (NLP) tasks.

1.2.2 How to train a large language model

The training process of LLMs is usually divided into two main stages: pre-training [3] and fine-tuning [4]. Each stage has different goals and methods, aiming to enable the model to have strong natural language understanding and generation capabilities. In addition, as the application of LLMs continues to expand, new training strategies such as Instruct Tuning [5], Model Distillation [6], Few-shot learning [7], reinforcement learning are also applied.

Pre-training: The goal of pre-training is to let the model learn the basics of language, such as grammar, semantics, contextual relationships, etc., from a large amount of unlabeled text data. This is a complex and resource-intensive process that usually requires a lot of computing resources, storage space, and data. The data source can be web page content, books, news articles, Wikipedia, conversation records, etc. There are usually two main ways to pre-train tasks: autoregressive Language Modeling [3]: the model predicts the next token given the context. It outputs each token in turn until the complete text is generated. Masked Language Modeling [4] involves randomly masking certain words in the input text and training the model to predict these masked words. This method is mainly used for understanding tasks.

Fine-tuning: The purpose of fine-tuning is to enable large language models to implement domain knowledge applications. First, select a model that has been pre-trained with a large amount of data, such as BERT [4], GPT-2 [8], T5 [9], etc. These models already have strong language understanding and generation capabilities. Then, as needed, prepare a small-scale, annotated dataset of domain knowledge. Then perform fine-tuning training. During the training process, the model updates its parameters through back-propagation. Usually, fine-tuning only involves the last few layers of the model [4].

Different training strategies: With the continuous expansion of LLM application areas, new training strategies such as Instruct Tuning [10], Model Distillation [6], Few-shot Learning [11], and Reinforcement Learning [12] are also being actively explored and adopted. Instruct Tuning focuses on aligning the model's behavior with human instructions by fine-tuning it on instruction-response pairs, thereby making the model more useful and controllable for practical downstream tasks. Model Distillation compresses knowledge from larger teacher models into smaller student models, significantly reducing computational costs while retaining high performance, which is crucial for deployment in resource-constrained environments. Moreover, Reinforcement Learning (e.g., RLHF) is increasingly used to optimize model outputs based on human preferences or task-specific reward signals, further enhancing alignment, safety, and task success rates in real-world applications. These training strategies not only boost the performance and usability of LLMs but also address emerging challenges related to scalability, alignment, and efficiency in industrial and academic settings.

1.3 LLMs for traffic safety

In recent years, with the successful application of LLMs in more and more fields, researchers have begun to explore their potential in the field of traffic safety [13].

Some studies [14, 15, 16, 17] have applied LLMs to traffic accident severity

prediction or risk level identification, which can be later on used for decision-making. Zhen et al. [15] fine-tuned the LLM by introducing a CoT reasoning mechanism and domain-knowledge-based prompt engineering, improving the model's accuracy and interpretability in accident severity classification. Table-to-text method [16, 17] are also widely used to effectively integrates structured data and text descriptions to identify high-risk traffic violations. These studies have demonstrated the potential of LLMs for complex tasks in the traffic safety sector, offering promising directions for future research.

Meanwhile, several recent studies have begun to explore the use of language models to address the challenge of limited or costly data annotation in traffic safety analysis. Seo et al. [18] proposed a BERT-based method that successfully extracts information on impact points and pre-collision vehicle maneuvers from unstructured traffic accident descriptions recorded by police. To further evaluate the potential of language models of different sizes in extracting unstructured traffic information, Mumtarin et al.[19] assessed ChatGPT, Bard, and GPT-4 on crash narratives, focusing on their ability to answer common safety-related questions. Across such studies, techniques such as prompt engineering, CoT reasoning, and few-shot learning are widely adopted to improve the performance. For instance, Arteaga and Park [20] applied prompt engineering to detect unreported alcohol involvement in 500 crash reports from Massachusetts, a relatively simple binary classification task. Overall, the tasks designed in these studies are relatively simple, often limited to direct fact extraction, and the resulting performance is only moderate, leaving substantial room for improvement in more challenging, inference-driven scenarios.

An important challenge is the limited adaptability of pre-trained LLMs to specialized domains, as their training corpora often lack sufficient coverage of domain-specific knowledge. Full fine-tuning can effectively bridge this gap; for example, Golshan et al. [21] fine-tuned the LLaMA 3.1 model for a QA task focused on extracting crash locations and casualty counts, achieving 97% accuracy in identifying fatalities and injuries. Jaradat et al.[22] used 26 226 traffic-related tweets from Australia to develop a multi-task learning (MTL) framework based on an LLM. With the introduction of LoRa technology [23], the cost of fine-tuning LLMs has been significantly reduced. As a result, fine-tuning has become a highly effective approach to improving the performance of LLMs in extracting traffic domain information. However, existing studies have largely focused on extracting information that is explicitly present in the text, while their effectiveness in handling tasks requiring deep reasoning remains limited and needs further improvement.

This study discusses the use of LLMs for traffic safety information extraction across tasks of varying difficulty. On the one hand, some tasks involve extracting information that is explicitly stated in the text, such as vehicle travelling direction, which mainly require accurate recognition and parsing. On the other hand, more

challenging tasks demand deeper reasoning, such as inferring the manner of collision or assigning the correct crash type to each vehicle when multiple vehicles are described within the same narrative. By comparing these different levels of information extraction, we aim to evaluate both the strengths and the limitations of LLMs in handling straightforward versus reasoning-intensive tasks.

1.4 Research Objectives and contributions

This thesis investigates the capability of LLMs to understand and extract structured information from crash narratives in the field of traffic safety. Specifically, we focus on evaluating LLMs' ability to answer domain-specific categorical questions such as:

- What is the vehicle's direction of travel?
- What is the manner of collision?
- What is the crash type of the vehicle?

The thesis uses the published dataset CISS, which contains both free-text accident summaries and corresponding structured annotations. This dual modality enables direct comparison between LLMs' predictions and ground truth labels. To comprehensively assess LLMs' performance, we compare different prompt engineering methods and fine-tuned models across multiple tasks. We examine the consistency, accuracy, and limitations of different open-source LLMs (e.g., Qwen, LLaMA) when applied to structured information extraction from crash narratives.

The intended contributions of this thesis can be outlined as follows:

Prompt engineering for crash information extraction. We propose to formulate crash classification tasks as structured question-answering problems and design a domain-specific prompting strategy. This method is intended to supports zero-shot and few-shot inference and enables the extraction of categorical information (e.g., crash type, manner of collision, direction of travel) directly from unstructured crash narratives.

Automated extraction of implicit crash information. We aim to investigate whether large language models can deduce complex crash characteristics that are not directly stated in narratives, drawing upon domain knowledge and reasoning about vehicle interaction patterns and collision dynamics. This direction seeks to reduce the necessity of manual examination of crash summaries and to facilitate automated information processing that traditionally required expert analysis.

Parameter-efficient fine-tuning of LLMs for traffic domain adaption. We plan to apply LoRA-based fine-tuning to open-source LLMs on a small annotated

crash dataset. This approach is intended to allow effective domain adaptation while significantly reducing the number of trainable parameters.

Empirical validation and performance analysis. We intend to conduct comparative evaluations to determine whether smaller fine-tuned LLMs can outperform much larger models in accuracy and macro F1-score. We also aim to analyze model overfitting behaviour and practical deployment considerations using the CISS dataset, toward developing a scalable LLM-based framework for structured information extraction in traffic safety applications.

1.5 Related Works

1.5.1 LLM applications in traffic safety analysis

The application of Large Language Models to crash narrative analysis has emerged as a promising research direction. Mumtarin et al. conducted a comparative evaluation of ChatGPT, BARD, and GPT-4 on 100 crash narratives, investigating their ability to answer safety-related questions including at-fault determination, manner of collision, and work-zone identification [19]. Their findings revealed varying performance across different types of questions, with higher similarity among LLMs for binary responses (96% and 89%) compared to complex reasoning tasks (35% for manner of collision).

Recent approaches have leveraged advanced prompting strategies for improved performance. CrashSage introduced tabular-to-text transformation strategies paired with relational data integration, demonstrating superior performance when fine-tuning LLaMA3-8B for crash severity inference [17].

Beyond classification, LLMs have also been applied to detect underreported or miscoded crash factors. Arteaga and Park introduced a large language model framework to uncover underreporting in traffic crash narratives, extending beyond specific factors to a broader range of inconsistently coded attributes [20]. This work highlights the potential of LLM-based approaches to enhance data quality by systematically detecting information that is missing or inaccurately recorded in structured crash databases.

1.5.2 Information extraction techniques and prompting strategies

Earlier studies on crash narratives used traditional machine learning classifiers, including multinomial naive Bayes, logistic regression, support vector machines, and recurrent neural networks [24]. More recently, transformer-based encoders such as BERT have been used to classify severity or contributing factors from narratives, establishing strong baselines for narrative-driven crash analysis [25].

This line of research created the groundwork for leveraging instruction-tuned LLMs, which extend narrative analysis beyond classification to more detailed information extraction.

Advanced prompting techniques have shown significant improvements over baseline approaches. Zero-shot and few-shot prompting provide direct baselines, while Chain-of-Thought prompting has demonstrated effectiveness in reasoning-intensive tasks by encouraging step-by-step problem decomposition [26]. Self-consistency techniques extend CoT prompting by generating multiple reasoning paths and selecting the most consistent answer through majority voting [27]. These methods are particularly relevant for crash reports, where reasoning often involves multiple vehicles and ambiguous references.

Parameter-efficient fine-tuning techniques, particularly Low-Rank Adaptation (LoRA), have gained prominence for domain adaptation while significantly reducing trainable parameters [23]. Extensions that target all linear layers, rather than just attention blocks, have shown improved adaptation quality and are directly applicable when adapting open-source LLMs to traffic safety domains.

1.5.3 Gap analysis and contributions

While existing work has demonstrated the potential of LLMs for crash narrative analysis, several gaps remain. Most studies have focused on closed-source models with limited evaluation of open-source alternatives that offer greater privacy protection [19]. Additionally, few works have systematically compared advanced prompting techniques and parameter-efficient fine-tuning approaches on the same dataset.

Moreover, limited research has addressed the structured extraction of directional attributes, crash configuration, and manner of collision, despite their critical importance for accurate crash reconstruction.

The present work addresses these gaps by: (1) systematically comparing open-source LLMs using controlled prompting strategies, (2) evaluating both prompting approaches and parameter-efficient fine-tuning on the same dataset and (3) providing detailed performance analysis across different crash complexity levels. This approach contributes to understanding the trade-offs between different modelling strategies for safety-critical information extraction tasks.

Chapter 2

Methods

2.1 CISS dataset

This project utilizes crash data from the Crash Investigation Sampling System (CISS), gathered by the U.S. National Highway Traffic Safety Administration (NHTSA) from 2016 to 2023 [2]. The CISS is a probability-based, nationally representative sampling technique that concentrates on police-reported collisions involving a minimum of one towed passenger vehicle, including automobiles, light trucks, SUVs, and vans.

CISS comprises traffic accident records collected from 2017 to 2023, with approximately 3,700 cases per year and collects crash data from 32 geographically diverse investigation sites (main sampling units), chosen to reflect variations in geography, population, traffic exposure, and crash characteristics. Certified crash technicians perform detailed investigations, collecting scene evidence (such as skid marks and vehicle damage), conducting vehicle inspections, interviewing occupants and witnesses, and reviewing medical records. Identifiable information, such as names, addresses, or specific crash locations, is excluded to maintain confidentiality.

The CISS database comprises about 39 relational tables[1], each describing a specific aspect of a crash, for example, `CRASH`, `VEHICLE`, `PERSON`, and `AIRBAG`, which contain structured variables such as crash severity, environmental conditions, and injury outcomes, together with the `SUMMARY` field in the `COLLISION` table that provides free-text narratives detailing the sequence of events, vehicle movements, and contextual factors, and which serves as the primary source for our NLP tasks.

This research examines the subset of CISS cases from 2017 to 2022, consisting

[1]The number and columns of tables in CISS are changing as the data-collection process is continuously updated. For instance, the dataset of year 2017 did not include the `VPICDECODE` table.

of 17,459 crash investigations. Each year comprises around 2,900 cases, with SUMMARY texts ranging from short single-sentence comments to complex multi-sentence narratives.

This dataset poses both problems and opportunities: the relational schema guarantees consistency and statistical representativeness; however, the `SUMMARY` text is unstructured but richly informative. This is particularly appropriate for investigating how LLMs (LLMs) and deep learning methodologies may derive structured crash details, such as vehicle initial movement direction, manner of collision, or crash configurations from narrative descriptions.

2.1.1 Dataset analysis

Number of vehicles per case

The number of vehicles implicated in a collision ranges significantly, indicating that the predominant majority of police-reported accidents consist of either single-vehicle occurrences or two-vehicle collisions. Table 2.1 presents the distribution identified in the dataset. As anticipated, the majority of collisions involve either a single vehicle (6,078 cases) or two cars (9,607 cases), whereas multi-vehicle collisions involving three or more vehicles are rather few. Only a small number of extreme cases, involving up to fourteen vehicles, are present in the dataset. This distribution aligns with traffic safety study findings, indicating that single and particularly two-vehicle collisions [28] compose the predominant portion of national crash statistics [29]. Although infrequent, multi-vehicle collisions remain crucial because they are linked to complex crash dynamics and more severe injuries.

Table 2.1: Distribution of crashes by number of vehicles involved.

Vehicles per crash	Number of cases
1	6078
2	9607
3	1422
4	263
5	55
6	22
7	7
8	4
14	1

Summary length statistics

The length of the `SUMMARY` narratives varies considerably, from concise one sentence descriptions to detailed multi-sentence accounts. Table 2.2 reports descriptive statistics (in characters) computed over all 17,459 summaries. The average narrative length is 356 characters, with a minimum of 60 and a maximum of 1,601 characters. This variability reflects the diversity of crash scenarios and the discretion of crash investigators in recording details.

The observed skewed distribution is typical of free-text crash reports and aligns with prior studies showing that narrative length often correlates with crash complexity: shorter notes are more common in simple rear-end or single-vehicle crashes, while longer narratives are used for multi-vehicle crashes or those requiring detailed contextual explanation. From a natural language processing perspective, this variation implies that models must handle both very short and relatively long narratives without losing contextual meaning [30, 31].

Table 2.2: Descriptive statistics of `SUMMARY` field lengths (characters).

Statistic	Value
Count	17459
Minimum	60
Maximum	1601
Mean	356.62

Implications for NLP analysis

The descriptive characteristics of the dataset have direct implications for subsequent modeling. In particular, the predominance of single and two-vehicle crashes highlights the need to design evaluation subsets that are balanced, in order to avoid biasing results toward simpler cases. At the same time, the variability in narrative length highlights the importance of identify appropriate context windows for LLMs and of designing prompts capable of capturing essential insights even from sparse descriptions. In addition, since longer narratives often involve more vehicles, they are more likely to contain ambiguities and overlapping references, highlighting the need for structured extraction methods [32].

2.2 Manual subset annotation

2.2.1 Motivation

To obtain a reliable gold standard for evaluation, we constructed a manually annotated subset of 1 000 crash narratives from CISS. Manual labeling is necessary for the first task of identifying the travel direction of the vehicles involved in the crash because these data are not available as structured fields in CISS and require contextual interpretation of the `SUMMARY` text. This annotated subset provides the ground truth against which different NLP models and LLMs are evaluated. The decision to annotate 1 000 cases balances two constraints: ensuring a sufficiently large sample to enable meaningful model evaluation and comparison, while keeping the manual effort manageable for a single annotator [33].

2.2.2 Sampling design

Reasons

Since the number of vehicles per crash varies widely (see Table 2.1), a simple random sample would have overrepresented common one- and two-vehicle cases and underrepresented rare multi-vehicle scenarios. To address this, we adopted a stratified sampling design, where cases were grouped by the number of vehicles mentioned in the narrative and sampled proportionally within each group. Stratified sampling is a widely recommended technique in traffic crash data analysis to ensure representative and unbiased samples, as supported by recent studies emphasizing careful sampling to improve model training and evaluation [33].

Threshold for rare cases

From the observed distribution (Section 2.1.1), categories with ≤ 55 cases (corresponding to crashes involving five or more vehicles) were considered *rare*. This threshold corresponds to approximately 0.5% of the dataset. All rare cases were included entirely in the sample to guarantee representation of high-order crashes. Previous research [34] has shown that, although infrequent, accidents involving multiple vehicles are often associated with more serious injuries and complex dynamics, making their inclusion methodologically important.

Proportional allocation

For the more frequent categories (one- to four-vehicle crashes), cases were sampled in proportion to their relative frequency in the dataset. Before computing the allocation, the vehicle-count distribution was verified to sum to the full dataset size

(17 459 cases), ensuring complete coverage of all records. The sampling proportion was then defined as

$$\text{proportion} = \frac{1000 - N_{\text{rare}}}{N_{\text{common}}},$$

where N_{rare} is the total number of cases in rare strata (i.e., categories with ≤ 55 cases) and N_{common} is the total number of cases in common strata. Using the observed counts ($N_{\text{tot}} = 17\,459$, $N_{\text{rare}} = 89$, $N_{\text{common}} = 17\,370$), this yields $p \approx 0.0524$ (5.24%). This proportionality ensured that the annotated subset preserved the same relative distribution of common crash types as in the full dataset.

After applying this allocation, small discrepancies caused by integer rounding were corrected by distributing the remaining cases across the largest categories in a *round-robin* manner (i.e., assigning them one by one in a cyclic sequence across categories) until the final sample size reached exactly 1 000. This procedure ensured both the coverage of rare categories and the statistical representativeness of common ones. A fixed random seed (42) guaranteed reproducibility.

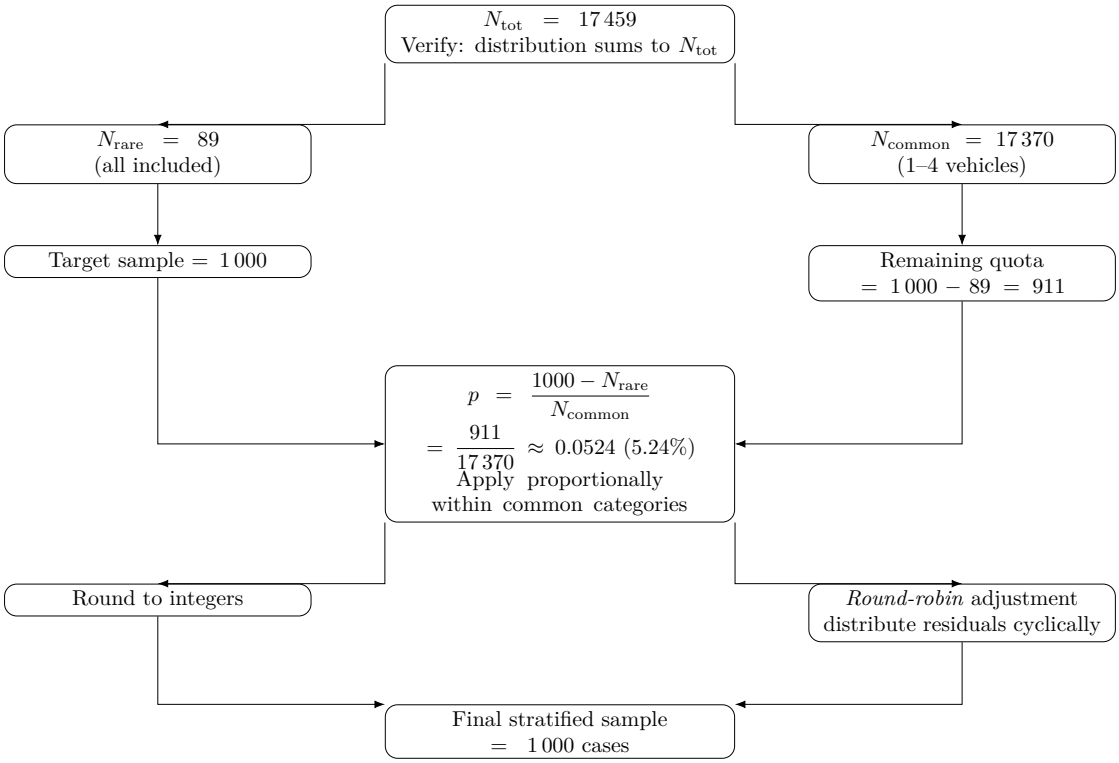


Figure 2.1: Computation of the sampling proportion and adjustment to obtain a stratified sample of 1 000 cases.

Implementation

The entire sampling pipeline was scripted in Python. A simplified excerpt of the logic is shown below:

```
# Compute proportional fraction
proportion = (1000 - sum_rare_cases) / sum_common_cases

def calc_sample(row):
    if row["cases_count"] <= rare_threshold:
        return row["cases_count"]  # include all rare cases
    else:
        return max(1, int(row["cases_count"] * proportion))

# Distribute remaining quota in round-robin among frequent categories
while difference > 0 and len(big_cats_idx) > 0:
    vc_counts.at[big_cats_idx[i], "sample_to_take"] += 1
    difference -= 1
    i = (i + 1) % len(big_cats_idx)
```

The final sample of 1,000 cases was exported in both CSV and XLSX formats, chosen for their compatibility with standard data processing libraries (e.g., `pandas` DataFrames).

2.2.3 From narrative to vehicle-count categories

Because the `SUMMARY` field is written in natural language, the number of vehicles per case was estimated using a set of robust, case-insensitive regular expression patterns. These included:

- explicit identifiers (`vehicle 1`, `vehicle #2`, `v1`, `unit 3`);
- coordinated mentions (`vehicles 1 and 2`, `v1 and v2`, or lists such as `v1, v2, v3`);
- number words (`vehicle one/two/three`, up to `ten`).

Mentions were normalized to numeric identifiers and deduplicated at the case level, yielding the variable `vehicle_count_extracted`. As a conservative fallback, if no explicit identifier was found but the narrative contained the token "vehicle," a single-vehicle count was assigned.

2.2.4 Annotation guidelines

All labels in this subset were produced *manually* by the authors, following explicit task-specific guidelines:

1. Focus exclusively on the *initial travel direction* of each vehicle.
2. Prefer explicit textual cues (e.g., "northbound") over inferred information.
3. When only orientation is described (e.g., "facing north" while parked), assign `unknown`.
4. If the description is ambiguous or inconsistent, assign `unknown`.
5. In collisions involving parked vehicles without explicit direction, assign `unknown`.

These guidelines emphasize explicitness and reproducibility, consistent with best practices in crash data annotation [35]. [2]

2.2.5 Auxiliary regex baseline

For a lightweight, interpretable point of comparison, we implemented a rule-based extractor. For each detected vehicle ID (`V1`, `V2`, ...), the algorithm searched a small token window around its mention for direction keywords (*north*, *south*, *east*, *west*, and their "-bound" variants). Bound forms were normalized to cardinal directions by stripping suffixes. The function returned a JSON-style map (e.g., `{V1: east, V2: unknown, ...}`). This baseline is not a substitute for human annotations and is expected to miss paraphrases or non-canonical phrasing. Its role was limited to serving as an auxiliary [33].

The implementation defined a set of 16 canonical direction terms, covering both cardinal points and their "-bound" variants:

```
direction_words = [
    "north", "south", "east", "west",
    "northeast", "northwest", "southeast", "southwest",
    "northbound", "southbound", "eastbound", "westbound",
    "northeastbound", "northwestbound",
    "southeastbound", "southwestbound"
]
```

[2] If multiple annotators were involved in future work, inter-annotator agreement measures (e.g., Cohen's κ) would be recommended to quantify consistency.

For each vehicle mention, the algorithm searched for a direction word within a context window of five tokens following or preceding the vehicle identifier. Matches were then normalized (e.g., "northbound" → "north"). The core matching logic can be summarized as:

```
# Search after vehicle mention
rf"\b{vehicle_id}\b((?:\W+\w+){0,5})\W+{direction_pattern}"

# Search before vehicle mention
rf"{direction_pattern}((?:\W+\w+){0,5})\W+\b{vehicle_id}\b"
```

This heuristic approach allowed attaching an automatically derived direction to each detected vehicle, which was stored in auxiliary columns of the exported subset. While not intended as a substitute for manual annotation, this baseline provides a transparent rule-based benchmark for evaluating the added value of LLMs and supervised models.

2.2.6 Reproducibility

The entire process of vehicle extraction, stratified sampling, annotation, and export was automated through reproducible scripts:

- All random operations were seeded (`random_state=42`);
- Outputs were stored in open formats (CSV, XLSX) for direct integration with analysis pipelines;
- Regex-derived directions were stored as auxiliary columns for optional comparison, but were not used as gold annotations.

2.2.7 Role of the annotated subset

The manually annotated dataset of 1 000 cases serves as the gold standard for evaluation in this work [31]. It allows for consistent comparisons across approaches: (i) Large Language Models (LLMs) under various prompting techniques and (ii) a fine-tuned BERT model. For the latter, the same 1 000 annotated cases were also used for training: the dataset was split into training, validation, and test subsets to support fine-tuning and evaluation. Since all methods are tested on the same cases, their performance can be directly compared, giving clear insights into the strengths and weaknesses of each approach.

Recent research shows that accuracy alone does not always match expert judgment [31]. Bhagat et al. found that some models with high accuracy had lower agreement with expert labels, while LLMs aligned better with expert opinions even

if their accuracy was lower. This demonstrates how important manually annotated expert datasets are as reference standards. They ensure model results capture subtle details and context that experts consider when studying crash reports.

Inter-rater reliability metrics

Metrics like Cohen's Kappa are well-known tools to measure agreement between annotators or between an annotator and a model. However, since this study used a single annotator, we did not apply these metrics.

Still, Cohen's Kappa plays an important role in assessing annotation consistency, especially when multiple annotators are involved or when checking how well models match expert views [36]. Future work could use such metrics to better measure annotation quality and model agreement, improving how NLP models for crash reports can be trusted and understood.

2.3 Approaches to vehicle direction extraction

The first objective of this work is to extract, from free-text crash narratives in the CISS dataset, (i) the set of vehicles explicitly mentioned in each case and (ii) the corresponding *initial travel direction* of each vehicle. This task is challenging because the information is not provided as structured fields in CISS but must instead be inferred from investigator-authored descriptions, which often include heterogeneous linguistic styles, varying levels of detail, and occasional ambiguity.

To address this challenge, we designed a comprehensive evaluation framework that combines *Large Language Models (LLMs)* with three fine-tuned *NLP models*. These supervised models provide a controlled point of comparison against general-purpose LLMs, allowing us to assess the trade-offs between compact, task-specific architectures and larger, instruction-tuned models.

The subsequent subsections describe the methodology in detail, covering: the selection and deployment of LLMs, inference configuration, prompt engineering strategies, robustness mechanisms for structured output, and the supervised fine-tuning of smaller NLP models for comparison. This structured approach allows us to investigate both the capabilities and the limitations of general-purpose LLMs relative to more traditional, task-specific models.

2.3.1 Overview of the LLM evaluation framework

To systematically evaluate the ability of LLMs to extract structured information from crash narratives, we developed a modular and reproducible evaluation framework. The system was implemented in Python and organized into four main components: (i) *configuration management*, to control experimental parameters

and ensure consistency across runs; (ii) *input/output utilities*, to standardize data ingestion and export; (iii) *LLM interaction handlers*, to interface with different models under controlled inference settings; and (iv) *execution orchestration*, which coordinates the end-to-end workflow and enables scalability to large datasets. This modular design facilitates maintainability, reproducibility, and transparent experimentation, aligning with established best practices for reproducible NLP research [37].

2.3.2 Model selection and justification

In designing the LLM evaluation framework, an important methodological decision concerned whether to rely on closed-source commercial models or open-source alternatives. We deliberately selected open-source models, as they provide full control over inference parameters, ensure data privacy by avoiding external API calls, support reproducibility through versioned local deployments, and reduce costs. These aspects are essential in a research setting, where consistent experimental conditions and methodological transparency must be guaranteed [38, 39].

For this task, three state-of-the-art open models were chosen to represent complementary strengths: *Mistral 7B-Instruct* [40], an instruction-tuned model optimized for efficiency and robust task adherence; *DeepSeek-R1 8B* [41], a reasoning-oriented architecture designed to handle analytical tasks with improved step-by-step consistency; and *Qwen3 8B* [42], a multilingual model with strong contextual understanding. Together, these models capture different design philosophies and allow us to evaluate the extraction task under varied inductive biases.

We restricted the selection to models in the 7–8B parameter range. This choice was motivated by methodological considerations. The extraction of vehicle mentions and initial travel directions from crash summaries constitutes a focused information retrieval problem: although the narratives are human-written, the linguistic variability is bounded and does not require the full capacity of very large models such as GPT-4 or others with 70B parameters or more. By selecting models in this intermediate range, we balance efficiency with sufficient expressive power, while also enabling a controlled comparison against smaller fine-tuned transformer baselines (BERT, RoBERTa, SciBERT). This design makes it possible to systematically assess the trade-offs between compact, task-specific architectures and larger, general-purpose LLMs.

2.3.3 Overview of selected LLMs

The three selected models represent distinct architectural philosophies and training approaches.

Mistral 7B-Instruct [40] is a 7-billion-parameter language model designed for

efficiency and strong instruction-following behavior. The suffix *Instruct* indicates that the base model has been fine-tuned with supervised and preference-based datasets to follow natural language instructions, making it better aligned with downstream tasks such as information extraction. Architecturally, Mistral introduces two key innovations: grouped-query attention (GQA), which accelerates inference by reducing redundant key-value computations, and sliding window attention (SWA), which enables handling sequences of arbitrary length with linear complexity relative to context size. These innovations allow Mistral 7B to outperform LLaMA 2 13B across all evaluated benchmarks, and even LLaMA 1 34B in reasoning, mathematics, and code generation tasks. Released under the Apache 2.0 license, Mistral represents the efficiency-oriented end of the design spectrum.

Qwen3 8B [42] introduces a unified framework that integrates both *thinking mode* (for complex, multi-step reasoning) and *non-thinking mode* (for rapid, context-driven responses), thereby eliminating the need to switch between specialized models depending on task type. A central novelty is the *thinking budget* mechanism, which adaptively allocates computational resources during inference, balancing latency and performance based on task complexity. Compared to its predecessor, Qwen3 significantly expands multilingual support, covering 119 languages (versus 29 in Qwen2), and demonstrates strong performance in both reasoning-heavy and general-purpose tasks.

DeepSeek-R1 8B [41] represents a novel approach to reasoning-focused language modeling. Unlike conventional LLMs that rely heavily on supervised fine-tuning or instruction data, DeepSeek-R1 is trained primarily through large-scale reinforcement learning (RL) without initial supervised pre-training. This paradigm allows reasoning skills to emerge naturally, yielding strong step-by-step reasoning capabilities. Empirical results reported in the release show performance comparable to OpenAI's o1-1217 on reasoning benchmarks. However, the authors highlight that the RL-based development introduces challenges such as variability in output formatting and language consistency, requiring careful handling in downstream applications.

Together, these models highlight three complementary philosophies: efficiency-focused instruction tuning (Mistral), unified multilingual reasoning with adaptive inference (Qwen3), and reinforcement learning–driven reasoning specialization (DeepSeek-R1) providing a diverse testbed for the vehicle direction extraction task.

2.3.4 Inference configuration

In this study, inference denotes the process of producing structured predictions (vehicle identifiers and initial travel directions) from unstructured crash narratives, conditioned on a fixed prompt and a controlled decoding configuration. Since generation parameters strongly affect model behavior, inference was carried out

Table 2.3: Technical specifications of selected LLMs

Specification	**Mistral 7B-Instruct**	**Qwen3 8B**	**DeepSeek-R1 8B**
Parameters	7.3B	8B	8B
Architecture	GQA + SWA	Dense/MoE hybrid	RL-optimized
Context Window	Standard	128K	Standard
Training Method	SFT + Instruct	Multi-stage + Budget	Pure RL
Reasoning Capability	General	Adaptive	Specialized
Inference Speed	Optimized	Adaptive	Standard
Multilingual	Yes	119 languages	Yes
Primary Use Case	Efficiency	Versatility	Complex reasoning

under standardized decoding settings tailored to each model while ensuring cross-model comparability. The chosen configurations are summarized in Table 2.4.

Table 2.4: Decoding parameters used during inference for each selected LLM.

Model	**Temperature**	**Top-p**	**Top-k**	**Repeat Penalty**
Mistral 7B-Instruct	0.7	Default	Default	Default
Qwen3 8B (thinking mode)	0.6	0.9	20	1.05
DeepSeek-R1 8B	0.6	Default	Default	Default

Note: In this table, *Default* refers to Ollama's standard parameters. For our experiments, when not specified, we used: $temperature = 0.3$ (standard inference) and 0.7 (self-consistency), $top\_p = 0.8$, $top\_k = 20$, $repeat\_penalty = 1.05$.

The main decoding parameters are briefly defined as follows. The **temperature** regulates sampling randomness: lower values promote deterministic, conservative outputs, whereas higher values increase diversity. The **top-k** parameter limits sampling to the k most probable tokens at each step, reducing the likelihood of generating nonsensical outputs. The **top-p** parameter (nucleus sampling) restricts sampling to the smallest set of tokens whose cumulative probability mass exceeds p, balancing diversity and coherence [43]. Finally, the **repeat penalty** controls how strongly repeated tokens are penalized: higher values discourage repetition, while lower values make the model more permissive.

All three models considered in this study are instruction-tuned variants, meaning that they have been optimized to better follow natural language prompts and produce task-oriented outputs. Instruction tuning has been shown to improve reliability in user-defined tasks, particularly in scenarios requiring structured information extraction [5]. This property is especially relevant here, as the objective is to transform free-text crash narratives into consistent JSON representations of vehicle travel directions.[3]

Overview of selected NLP models

In addition to evaluating general-purpose Large Language Models (LLMs), this work employed three transformer-based baselines that were fine-tuned specifically for the

[3]Although all models are released in an instruction-tuned form, their approaches differ: Mistral 7B-Instruct applies supervised fine-tuning and preference optimization; Qwen3 8B adopts a multi-stage framework that integrates "thinking" and "non-thinking" modes; and DeepSeek-R1 relies primarily on reinforcement learning, where instruction-following behavior emerges from the training dynamics.

vehicle direction extraction task: *BERT*, *RoBERTa*, and *SciBERT*. These models represent compact yet widely adopted architectures in natural language processing, offering strong performance in supervised settings while being computationally efficient compared to large instruction-tuned LLMs. The selection of these particular models enables a comprehensive evaluation across different pretraining paradigms and domain specializations within the transformer family.

BERT (Bidirectional Encoder Representations from Transformers) introduced a paradigm shift in language representation learning through its bidirectional training approach [4]. Unlike previous models that processed text in a left-to-right or combined left-to-right and right-to-left manner, BERT employs a masked language model (MLM) objective that enables deep bidirectional representations by randomly masking tokens and predicting them based on both left and right context. The model architecture consists of a multi-layer bidirectional Transformer encoder, with the base version (BERT-Base) containing 12 transformer blocks, 768 hidden dimensions, 12 attention heads, and approximately 110 million parameters. BERT was pretrained on BooksCorpus (800M words) and English Wikipedia (2.5B words), providing broad coverage of general English language patterns [4]. The model's bidirectional nature is particularly relevant for crash narrative analysis, where directional information may be distributed throughout a sentence and require comprehensive contextual understanding.

RoBERTa (Robustly Optimized BERT Pretraining Approach) builds upon BERT's architecture while addressing several pretraining limitations through systematic optimization [44]. Key improvements include: (i) removing the Next Sentence Prediction (NSP) task, which was found to be unnecessary and potentially harmful; (ii) training on longer sequences and larger batches; (iii) using dynamic masking rather than static masking during pretraining; and (iv) employing a substantially larger and more diverse training corpus. RoBERTa was trained on over 160GB of text data, including the original BERT corpus plus CC-NEWS, OPENWEBTEXT, and STORIES datasets, totaling approximately 160GB of uncompressed text. These optimizations resulted in consistent improvements over BERT across various downstream tasks, with particular gains in reading comprehension and natural language inference [44]. For crash narrative analysis, RoBERTa's more robust pretraining may provide better generalization to the diverse linguistic styles encountered in investigator-authored reports.

SciBERT addresses the domain gap between general language corpora and scientific text by pretraining specifically on scientific publications [45]. The model maintains the same architecture as BERT-Base but was trained from scratch on a corpus of 1.14 million full-text scientific papers from Semantic Scholar, spanning computer science and biomedical domains. This specialized pretraining resulted in a scientific vocabulary that better covers technical terminology and domain-specific linguistic patterns. Importantly, SciBERT uses a custom WordPiece vocabulary

derived from the scientific corpus rather than the general-domain vocabulary used by BERT, enabling more effective tokenization of scientific terms. Experimental results demonstrated SciBERT's superiority over BERT on various scientific text processing tasks, including named entity recognition, relation classification, and dependency parsing in biomedical and computer science texts [45]. Although crash narratives differ from typical scientific literature, they share certain characteristics with technical writing, including specialized terminology, abbreviated forms, and structured reporting patterns that may benefit from SciBERT's domain-adapted representations.

There are many reasons for including these three models. First, they provide controlled benchmarks for assessing the trade-offs between task-specific fine-tuning on limited labeled data versus zero-/few-shot prompting (Section 2.4.2) with large instruction-tuned models. This comparison is crucial for understanding when domain-specific adaptation outweighs the benefits of general-purpose reasoning capabilities. Second, their different pretraining corpora enable investigation into how domain coverage and vocabulary specialization influence extraction accuracy in crash narrative analysis. The progression from general English (BERT) to optimized general training (RoBERTa) to domain-specific scientific text (SciBERT) provides insights into the importance of training data composition for specialized tasks. Third, these models represent different points on the accuracy-efficiency trade-off curve, with their compact size (110M parameters each) offering significantly faster inference compared to billion-parameter LLMs while maintaining competitive performance in supervised settings.

By fine-tuning each model on the manually annotated subset of 1,000 crash narratives using identical training procedures, we establish controlled supervised baselines that complement the LLM evaluation. This design enables quantification of the advantages and limitations of instruction-tuned, general-purpose models compared to smaller, domain-adapted transformers, providing insights relevant to both research and practical deployment considerations in safety-critical text analysis applications.

2.4 Prompt engineering strategies

Prompt engineering represents a critical component for effectively aligning Large Language Models (LLMs) with domain-specific tasks [46]. For this study, we implemented a template-based system that enables dynamic prompt generation, ensuring that accident narratives are processed consistently within the same structural framework, taking into account the specific contextual variations of each case. The prompting strategies adopted in this work are grounded in established paradigms

documented in the literature, particularly drawing from recent comprehensive surveys that systematize the current state of prompt engineering methodologies [46]. This section details the specific approaches employed to optimize LLM performance for crash narrative analysis and structured information extraction.

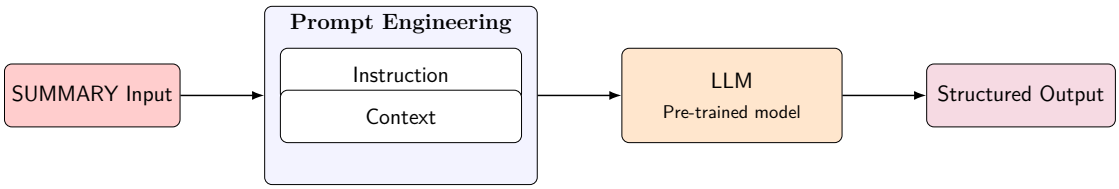


Figure 2.2: Overview of the prompt engineering pipeline. The prompt is composed of two parts: an *Instruction* (task specification) and a *Context* (e.g., few-shot demonstrations or schema hints); the LLM then produces a structured output.

2.4.1 Dynamic prompt generation

To maintain consistency, all prompts were generated from predefined templates. Each template contained placeholders for case-specific information (e.g., the crash `SUMMARY`), which were dynamically injected during runtime. This approach reduces formatting errors and ensures reproducibility across different models and prompting techniques.

2.4.2 Prompting techniques

In this study, we employed three representative prompting strategies: zero-shot, few-shot, and chain-of-thought, selected because they capture the most influential approaches reported in recent work [7, 26, 27]. In addition, we briefly explored self-consistency with one model as an extension of CoT prompting.

Zero-shot prompting

Zero-shot prompting represents the most direct application of pretrained LLMs and has been described as a paradigm shift in how these models are leveraged [8]. In this approach, the model is given only a natural language task description without any labeled examples or task-specific fine-tuning and must rely entirely on its pre-existing knowledge acquired during large-scale pretraining. The technique gained prominence with GPT-3, which demonstrated that sufficiently large models can generalize to novel tasks through carefully crafted instructions alone [7].

In the context of crash narrative analysis, zero-shot prompting consists of directly instructing the model to identify vehicles and extract their initial travel directions

from unstructured investigator-authored text. This setting provides a baseline to assess the model's inherent ability to generalize to specialized information extraction tasks without any domain-specific demonstrations. While the approach is simple and avoids prompt-length overhead, it is also more sensitive to ambiguous phrasing compared to prompting strategies that incorporate illustrative examples.

```
You are a precise traffic accident analyst. Extract the initial
    travel directions of vehicles mentioned in the crash summary.

IMPORTANT: Only include vehicles that are explicitly mentioned in
    the text. Do NOT add vehicles that are not present.

DIRECTIONS: Use only these values:
- "north", "south", "east", "west", "northeast", "northwest", "
    southeast", "southwest", "unknown"

RULES:
- If direction is explicitly stated, use it
- If direction can be inferred from context, infer it
- If completely unclear, use "unknown"
- If only one vehicle is mentioned, output only V1
- If no vehicles are found, return empty JSON: {}
- Return ONLY valid JSON, no explanations

REQUIRED OUTPUT FORMAT:
{
  "V1": "direction"
}

OR (if multiple vehicles exist):
{
  "V1": "direction",
  "V2": "direction"
}

CRASH SUMMARY: V1 traveling west on highway departed roadway right
    , drove through ditch and contacted its frontal plane on the
    ditchs culvert.
```

Note: This template was kept consistent across models to ensure comparability. The zero-shot setting is particularly important in this study because it evaluates whether a model is capable of generalizing the extraction task without relying on contextualized examples, thus reflecting its intrinsic abilities to follow instructions and reason.

Few-shot prompting

Few-shot prompting enhances model performance by leveraging in-context learning: a small number of labeled input–output examples are embedded directly into the prompt, enabling the model to imitate patterns without parameter updates [7, 47]. This technique addresses the limitations of zero-shot prompting by offering concrete demonstrations, which previous studies show can substantially improve accuracy on complex reasoning and classification tasks.

In our implementation, representative crash summaries with annotated vehicle directions were inserted before the target case. To ensure coverage, we manually selected four examples spanning different levels of complexity: (i) a single vehicle with explicit direction, (ii) two vehicles interacting at an intersection, (iii) a case where no explicit direction was provided (annotated as `unknown`), and (iv) a multi-vehicle case mixing explicit and unknown directions.[4]

Despite its effectiveness, few-shot prompting introduces certain challenges: the inclusion of examples increases prompt length, which can consume valuable context window space; model behavior may be sensitive to the specific examples chosen; and biases toward frequently occurring patterns may persist. In our case, the relatively short length of CISS crash summaries makes this approach computationally feasible while still benefiting from the added structure that demonstrations provide.

The examples were included in the prompt exactly as shown below:

```
Example 1
Summary: "V1 was traveling in an easterly direction on a designated
north/south divided trafficway (median strip without positive barrier)
in lane 2 (of two lanes), when it departed the road on the left and was
then returned to the road from the left. Control over V1 was lost, and V1
rotated clockwise as it departed the road on the right, where it then
overturned 4 quarter turns, leading left. V1 came to rest on its wheels
facing a southwesterly direction on the roadside."
Output: {"V1": "east"}

Example 2
Summary: "V1 was traveling east approaching an intersection. V2 was
```

[4]Examples were manually curated to capture typical narrative patterns rather than chosen at random, balancing representativeness with clarity.

```
traveling south toward the same intersection. Both vehicles entered the
intersection and the front of V1 impacted the right side of V2."
Output: {"V1": "east", "V2": "south"}

Example 3
Summary: "V1 drove off the right side of the roadway and impacted 2
trees with the front plane and then a tree impacted the undercarriage."
Output: {"V1": "unknown"}

Example 4
Summary: "Vehicle #1 was traveling south. Vehicle #2 was traveling east.
Both vehicles entered an intersection and the front plane of V2 contacted
the right plane of V1. An unknown plane of either V1 or V2 contacted an
unknown plane of V3, which was traveling in an unknown direction, and
left the scene."
Output: {"V1": "south", "V2": "east", "V3": "unknown"}
```

These demonstrations were concatenated with the task instruction and followed by the new unseen summary, ensuring that the model could infer the correct JSON output format by analogy with the provided examples.

Chain-of-thought prompting

Chain-of-thought (CoT) prompting addresses the reasoning limitations of large language models by encouraging the generation of intermediate reasoning steps before producing final answers [26]. This technique facilitates coherent and step-by-step reasoning processes, guiding models through logical reasoning chains that result in more structured and thoughtful responses compared to direct answer generation. CoT prompting has demonstrated particular effectiveness in multi-step reasoning tasks and scenarios requiring disambiguation, where the explicit reasoning process helps maintain logical consistency [46]. Applied to crash narrative analysis, Chain of Thought prompts instruct the model to decompose the extraction task into sequential steps: first identifying all vehicles explicitly mentioned in the narrative, then analyzing movement patterns and directional indicators for each vehicle, and finally outputting the normalized directions in the required JSON format. This structured approach mirrors human analytical processes and ensures transparency in the model's decision-making process, which is particularly valuable

in safety-critical applications where interpretability is essential.

```
TASK:
1. Read the crash summary carefully.
2. Identify all vehicles mentioned.
3. For each vehicle, reason step by step about its initial
   movement.
4. Write down clues from the text, perform the inference, and only
    then output the final JSON.

Crash summary:
"V1 traveling west on highway departed roadway right, drove
   through ditch
and contacted its frontal plane on the ditch's culvert."

Step-by-step reasoning:
Vehicles found: V1
Direction clues: "V1 traveling west"
Inference: V1 initial direction is west

Final JSON output:
{
  "V1": "west"
}
```

Note: The general schema (allowed directions, JSON format, and rules for explicit/implicit extraction) was the same as in the zero-shot and few-shot settings. The only difference was the explicit request for step-by-step reasoning before the final structured output. The reasoning steps shown here illustrate the expected model behavior, but were not included as part of the actual prompt input.

Self-consistency

Self-consistency extends CoT prompting by leveraging multiple reasoning paths to improve the reliability and robustness of model predictions [27]. Instead of relying on a single reasoning chain, the model generates diverse outputs under stochastic decoding (e.g., higher temperature), and the most frequent final answer is selected through majority voting. This approach exploits the observation that correct solutions to reasoning tasks often emerge consistently across multiple paths, whereas incorrect reasoning tends to diverge. As a result, self-consistency has been shown to substantially improve accuracy on reasoning benchmarks compared to single-path generation [46].

In this study, self-consistency was explored only with the *Mistral 7B-Instruct* model. Multiple responses were sampled at elevated temperature, normalized into valid JSON format, and aggregated using a frequency-based consensus rule. The

final prediction corresponded to the most common structured output, with the consensus ratio serving as a confidence indicator.

In practice, we generated `SELF_CONSISTENCY_SAMPLES = 5` independent outputs at an elevated decoding temperature (`SELF_CONSISTENCY_TEMPERATURE = 0.7`). This configuration was chosen as a balance between computational efficiency and sufficient diversity of reasoning paths: five samples provided enough variation to identify stable consensus patterns while keeping runtime overhead manageable. Although applied in a limited setting, this technique proved particularly valuable for handling ambiguous crash narratives, where relying on a single reasoning chain could otherwise lead to inconsistent extractions.

Example prompt The snippet below illustrates the self-consistency prompt template used in our experiments. The task formulation explicitly requests repeated reasoning, internal verification, and strict JSON-formatted output:

```
[INST]
Extract vehicle initial travel directions from the crash summary below by carefully analyzing to ensure consistent and accurate results.

IMPORTANT: Only include vehicles explicitly mentioned in the text. DO NOT add vehicles that are not present.

TASK:
1. Read the crash summary multiple times...
...
After careful multiple reviews and consistency checks, my confident final output is:
[/INST]
```

Implementation Self-consistency was implemented in Python. Multiple outputs were sampled for each case and aggregated through a frequency-based consensus rule. A simplified excerpt is shown below:

```
def query_model_self_consistency(summary, n_samples=5):
    responses = []
    for i in range(n_samples):
        raw_response = ollama.chat(
            model=MODEL_NAME,
            messages=[{"role": "user", "content": get_prompt(summary)}],
```

```
            options={"temperature":
    SELF_CONSISTENCY_TEMPERATURE, "top_p": 0.9}
        )["message"]["content"]
        parsed = parse_json_response(raw_response)
        if parsed:
            responses.append(json.dumps(parsed, sort_keys=
    True))
     # Majority voting
     counter = Counter(responses)
     most_common, count = counter.most_common(1)[0]
     confidence = count / len(responses)
     return json.loads(most_common), confidence
```

This procedure outputs both the consensus prediction and a confidence score defined as the proportion of samples agreeing on the final result.

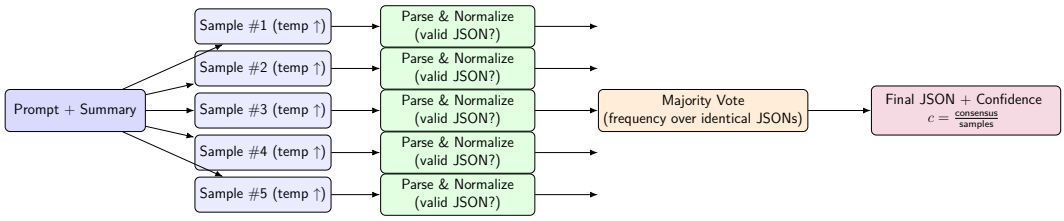


Figure 2.3: Self-consistency decoding. Five stochastic samples are parsed and normalized into JSON,
aggregated by majority vote, and returned as a final structured output with a confidence ratio.

Model-specific prompt syntax

While the logical structure of prompts was kept consistent across models, minor syntax adaptations were required to conform to model-specific instruction formats. For instance, Mistral requires prompts to be wrapped with `[INST] ... [/INST]`, DeepSeek-R1 uses the special `<think>...</think>` tokens to elicit reasoning, and Qwen3 supports a dual-mode prompting strategy ("thinking" vs. "non-thinking"). These adjustments ensured compatibility with each model's expected input format, while preserving task comparability.

2.4.3 Reasons for strategy selection

The combination of zero-shot, few-shot, and CoT covers the most prominent paradigms in prompt engineering [46]. Together, they span a spectrum from direct

task specification to structured reasoning with uncertainty reduction, providing a comprehensive evaluation of how prompting techniques affect extraction performance in safety-critical text analysis tasks since these techniques are particularly suited for crash narrative analysis, where linguistic variability, domain-specific terminology, and ambiguous descriptions require different levels of guidance and reasoning support.

2.5 LLMs' output postprocessing

2.5.1 Motivation

When prompted to generate JSON, Large Language Models (LLMs) often produce outputs that are partially malformed (e.g., unbalanced braces, duplicate keys, trailing commas) or include explanatory text. Simply discarding these cases would reduce the dataset size, introduce bias into the evaluation sample, and waste computational effort. To mitigate these issues, we implemented two complementary normalization strategies:

1. **Strategy A (valid-only)**: a conservative baseline that retains only well-formed JSON objects;

2. **Strategy B (raw-response recovery)**: a more inclusive pipeline that attempts to repair or reconstruct outputs from malformed responses.

Both tracks impose a standardized key and schema format, enabling direct comparability across models and prompting strategies.

2.5.2 Common step: ingestion and inspection

All input files were stored as JSON lists with one entry per crash case, including a unique identifier (`case_id`), the model's predicted directions, the raw textual response, and a Boolean flag (`valid`) marking whether the output could be parsed as JSON. Table 2.5 summarizes the fraction of valid versus invalid responses across models and prompting strategies, while Listings 2.5.2 and 2.5.2 illustrate typical examples of valid and invalid records. This diagnostic stage is common to both strategies and provides the baseline statistics used to guide subsequent processing.

Table 2.5: Validity of LLM outputs after initial parsing. The table reports the number of valid JSON responses retained and the resulting success rate.

Model	Prompting strategy	Valid count	Success rate
Qwen3 8B	Few-shot	1000	100.0%
Qwen3 8B	Zero-shot	999	99.9%
Qwen3 8B	Chain-of-thought	1000	100.0%
DeepSeek-R1 8B	Chain-of-thought	1000	100.0%
DeepSeek-R1 8B	Zero-shot	914	91.4%
DeepSeek-R1 8B	Few-shot	916	91.6%
Mistral 7B-Instruct	Zero-shot	996	99.6%
Mistral 7B-Instruct	Few-shot	997	99.7%
Mistral 7B-Instruct	Chain-of-thought	991	99.1%

Note. Each model was evaluated on 1,000 cases; the *valid count* indicates how many predictions could be parsed directly as JSON.

Example of a valid JSON record


```
{
  "case_id": 26987,
  "directions": {
    "V1": "north",
    "V2": "unknown"
  },
  "raw_response": " {\n    \"V1\":\"north\",\n    \"V2\":\"unknown
   \"\n    }",
  "valid": true
}
```


Example of an invalid JSON record


```
{
  "case_id": 24415,
  "directions": {},
  "raw_response": " {\n    \"V1\":\"southwest\",\n    \"V1\":\"
   unknown\" // since it rotated counterclockwise, but direction
   after rotation is unknown\n  }",
  "valid": false
}
```

2.5.3 Strategy A: valid-only records

This conservative baseline discards all invalid entries and retains only those flagged with `valid=true`. Essential fields (`case_id`, `directions`, and `raw_response`) are extracted, and vehicle identifiers are normalized to a canonical format (`V1`, `V2`, ...). This ensures consistency across models and comparability with transformer baselines.

The main advantage of this approach is transparency: the evaluation parameters reflect only well-formulated predictions without any intervention. The trade-off is a reduced sample size, as results containing even minor formatting errors are excluded.

2.5.4 Strategy B: raw-response recovery

This more inclusive approach attempts to repair malformed results, thereby increasing the number of usable cases. Recovery is performed on records marked as invalid, using a cascade of progressively more tolerant parsers:

1. try to extract the largest valid JSON block from the text;
2. fix simple format errors such as extra commas or brackets;
3. if that fails, attempt parsing with more permissive decoders (e.g., `ast.literal_eval`);
4. as a last resort, rebuild a minimal JSON object from detected key–value pairs.

JSON extraction from raw responses

```
def extract_json_from_raw_response(raw_response):
    """Recover JSON from raw LLM output."""
    if not raw_response or not isinstance(raw_response, str):
        return None

    text = raw_response.strip()

    # 1. Extract after </think> if present
    match = re.search(r"</think>\s*(.+)$", text, re.DOTALL)
    json_candidate = match.group(1).strip() if match else text

    # 2. Regex to find JSON-like blocks
    patterns = [r"\{[^{}]*(?:\{[^{}]*\}[^{}]*)*\}", r"\{[^}]+\}"]
    potential = []
    for p in patterns:
        potential.extend(re.findall(p, json_candidate, re.DOTALL))

```

```
    for js in reversed(potential):
        try:
            return json.loads(js)
        except:
            pass
        try:
            return ast.literal_eval(js)
        except:
            pass
        try:
            fixed = re.sub(r",\s*}", "}", js)
            fixed = re.sub(r",\s*]", "]", fixed)
            fixed = fixed.replace("'", '"')
            return json.loads(fixed)
        except:
            pass

    # 3. Fallback: extract key-value pairs
    matches = re.findall(r'"([^"]+)":\s*"([^"]+)"', json_candidate
    )
    if matches:
        return dict(matches)
    return None
```

Normalization of vehicle keys

Recovered objects are normalized according to the same scheme as Strategy A (e.g., #1, `vehicle 1`, or plain numbers) to a consistent format (`V1`, `V2`, ...).

```
def normalize_vehicle_keys(directions_dict):
    """Standardize vehicle identifiers to V1, V2, ..."""
    if not isinstance(directions_dict, dict):
        return directions_dict

    normalized = {}
    for key, direction in directions_dict.items():
        key_str = str(key).strip()
        patterns = [
            r"^#(\d+)$", r"^V(\d+)$",
            r"^vehicle\s*(\d+)$", r"^(\d+)$"
        ]
        number = None
        for p in patterns:
            match = re.match(p, key_str, re.IGNORECASE)
            if match:
                number = match.group(1)
                break
```

```
        if number:
            normalized[f"V{number}"] = direction
        else:
            normalized[key] = direction
    return normalized
```

2.5.5 Final datasets for evaluation

Both strategies produce standardized datasets that serve as direct input to the evaluation framework. In the *valid-only* version, malformed outputs are treated as errors, so evaluation is conducted over all 1 000 cases with invalid responses counted as incorrect predictions. In contrast, the *clean+recovered* version integrates successfully repaired outputs into a unified dataset, thereby increasing effective coverage while preserving schema consistency.

In the following chapter, we evaluate the performance of both versions of the model systematically, highlighting how the choice of post-processing strategy affects accuracy estimates and model comparisons.

2.6 Fine-tuning of transformer-based models

To provide a supervised benchmark against which to evaluate Large Language Models (LLMs), we fine-tuned three transformer-based architectures *BERT*, *RoBERTa*, and *SciBERT* on the manually annotated corpus. The models were selected to cover complementary pretraining regimes: general English corpora (BERT), an optimized large-scale variant (RoBERTa), and scientific-domain text (SciBERT). Fine-tuning adapts each backbone to the task of vehicle direction analysis, offering a point of comparison with zero-/few-shot LLM prompting.

2.6.1 Architecture and task approach

We adopted a *dual-model* design that separates incident understanding into two specialized classifiers:

1. **Vehicle Count Model**: multi-class sequence classifier predicting how many vehicles are involved in the incident summary. For the Vehicle Count Model, crash-level annotations were aggregated so that each incident corresponded to a single training example labeled with the total number of vehicles involved. To avoid sparsity from rare extreme cases, incidents with more than eight vehicles were grouped into a single category labeled “8+”.

2. **Vehicle Direction Model**: per-vehicle sequence classifier predicting the *initial* travel direction; the label set comprises nine classes: {`north`, `south`, `east`, `west`, `northeast`, `northwest`, `southeast`, `southwest`, `unknown`}.

Decomposing the problem reduces label-space complexity, allows task-specific optimization (hyperparameters, loss emphasis), and improves interpretability by decoupling crash-level and vehicle-level predictions.

2.6.2 Base models

We instantiated three pretrained backbones:

- **BERT-base-uncased** [4] (general-domain baseline).
- **RoBERTa-base** [44] (optimized training dynamics, larger corpora).
- **SciBERT (allenai/scibert_scivocab_uncased)** [45] (scientific-domain pretraining with domain-specific vocabulary).

We initially fine-tuned BERT as the baseline backbone model for vehicle direction extraction. Building on this starting point, we extended the analysis by incorporating RoBERTa and SciBERT, enabling ablation studies and comparative evaluation. This progression allowed us to test whether differences in pretraining corpora: general-domain (BERT), optimized large-scale training (RoBERTa), and scientific-domain (SciBERT) would translate into performance differences on crash narratives.

2.6.3 Dataset preparation

Data preprocessing: per-vehicle expansion

Before fine-tuning transformer baselines, the manually annotated crash dataset was transformed into a vehicle-level representation through a dedicated preprocessing pipeline. This step was essential to align the data format with sequence classification tasks, where each training instance corresponds to a single input–output pair.

Input data structure The manually annotated corpus comprised 1 000 crash cases and 2 106 vehicle-level records. Each case included a free-text `SUMMARY`, the case identifier (`CASEID`), and ground-truth annotations encoded as a dictionary mapping vehicles (e.g., `V1`, `V2`) to their initial travel directions.

Parsing and normalization Since manual annotations contained occasional inconsistencies, a fault-tolerant two-stage parsing mechanism was developed:

- **Primary parsing:** standard `ast.literal_eval()` for well-formed dictionaries.
- **Fallback parsing:** regex-based correction for missing quotation marks or minor syntax errors.
- **Normalization:** all labels converted to lowercase, with special cases (e.g., `none`) mapped to `unknown`.

Vehicle-level expansion Each crash case was expanded into multiple rows, one per vehicle–direction pair, while preserving the complete narrative. The transformation produced a dataset with the following schema:

- `case_id`: original crash identifier (traceability).
- `vehicle_summary`: full crash description (BERT input).
- `vehicle_id`: vehicle identifier (e.g., `V1`, `V2`).
- `direction_label`: normalized ground-truth direction (classification target).
- `vehicles_involved`: total vehicle count in the crash (contextual metadata).

Advantages This design offered three key benefits:

1. **Context preservation:** each vehicle record retains the full crash narrative, enabling the model to leverage inter-vehicle relationships.
2. **Balanced representation:** every vehicle contributes equally to training, regardless of case size.
3. **Robustness:** inconsistent annotations are gracefully handled, ensuring no case is discarded due to parsing errors.

Fine-tuning dataset split

Case vs.vehicle-level construction. For the *Vehicle Count Model*, records were aggregated at the incident level (one row per `CASEID`); cases with more than 8 vehicles were mapped to the class `8+`. The resulting distribution reflects real-world exposure patterns, with approximately 82.3% of cases involving 1–2 vehicles.

For the *Vehicle Direction Model*, we maintained granularity at vehicle level : each pair (`CASEID`, `vehicle_id`) produces a training instance with its corresponding direction label.

Input formatting (prompt-style templates) To make the classification objectives explicit to the backbone models, inputs were modeled minimally:

- **Count task:**

```
"How many vehicles are involved in this incident?
[SUMMARY]"
```

- **Direction task:**

```
"Direction of [vehicle_id]?  [SUMMARY]"
```

This instruction-like format preserved the entire narrative while providing clear guidance on the tasks to be performed.

Train/validation split An 85/15 stratified train–validation split was applied with a fixed random seed (42).

- **Stratification criterion:** target labels (vehicle counts for the count task; 9-way direction labels for the direction task).

- **Motivation:** to preserve that rare cases such as crashes involving more than four vehicles or uncommon direction categories were still represented in both splits, reducing the tendency of the models to overfit on the frequent single and two vehicle cases.

Tokenization The preprocessed vehicle-level data were handled using the pandas library [48] and then converted into the input format expected by the Hugging Face Transformers API [49]. All text inputs were tokenized with the pretrained tokenizer of each backbone model, using padding and truncation to fixed sequence lengths:

- Count task: max sequence length 512.

- Direction task: max sequence length 256.

2.6.4 Training configuration

Training used `PyTorch` [50] and Hugging Face Transformers [51]. Unless otherwise noted, we used AdamW with linear learning-rate scheduling and weight decay [52].

Learning rate strategy

The learning rate was selected within the standard fine-tuning range for BERT-like models (1e-5 to 5e-5) as recommended in [4]. Distinct values were adopted for the two tasks:

- **Vehicle Count Model:** 2e-5, conservative to ensure stability in the simpler 4-class problem.
- **Direction Model:** 3e-5, slightly higher to accelerate convergence in the more challenging 9-class setting.

Training duration

The Vehicle Count Model was trained for 4 epochs, while the Direction Model required 8 epochs. The latter's extended training was motivated by its higher class cardinality, semantic granularity (e.g., distinguishing "northeast" vs. "east"), and the presence of imbalanced categories (particularly the `unknown` class).

Warmup and regularization

Linear warmup (50 steps for count; 100 steps for direction) was introduced to prevent early divergence, covering roughly 5–10% of the training schedule. A weight decay coefficient of 0.01 provided L2 regularization, improving validation performance by $\sim$3% compared to runs without regularization.

Batch size and sequence length

Empirical testing on 16GB GPUs established batch size 16 as optimal, following established best practices for transformer fine-tuning under memory constraints [4, 53]:

- Batch size 32: out-of-memory errors at 512 tokens
- Batch size 8: stable but 40% slower training with negligible performance gains
- Batch size 16: efficient GPU utilization ($\sim$14GB) with stable gradient estimation

For sequence length, we adopted differentiated configurations based on task-specific requirements. The vehicle count model uses 512 tokens to capture complete crash summaries, while the direction model employs 256 tokens, as directional cues typically appear in shorter textual spans near vehicle mentions. This differentiation balances computational efficiency with task-specific information needs.

2.6.5 Model training procedure

Both tasks were trained using the Hugging Face `Trainer` class with training arguments configured for epoch-based evaluation and model saving:

```
training_args = TrainingArguments(
    output_dir=output_dir,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    metric_for_best_model="eval_accuracy",  # or "eval_f1_macro"
   for direction
    # ... other parameters
)
```

The classification head was added to the pre-trained model through `AutoModelForSequenceClassification`, with explicit label mappings:

```
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=num_labels,
    id2label=id2label,
    label2id=label2id
)
```

Metrics

Custom metrics ensured evaluation on both overall and class-balanced performance:

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)
    return {
        "accuracy": accuracy_score(labels, preds),
        "f1_weighted": f1_score(labels, preds, average="weighted")
   ,
        "f1_macro": f1_score(labels, preds, average="macro")
    }
```

Accuracy was the selection metric for the Vehicle Count Model, while the Direction Model used macro F1-score to avoid overfitting to dominant classes.

Monitoring

Logging frequency was adjusted to dataset size: every 20 steps for the count task (850 training examples) and every 50 for the direction task (1790 examples). This enabled early anomaly detection (e.g., gradient explosion, stagnation).

Convergence dynamics

Training curves exhibited distinct behaviors across the two models:

- **Vehicle Count Model:** converged rapidly within a few epochs, consistent with the relative simplicity of the four-class prediction task.
- **Direction Model:** required longer training to achieve stable performance, reflecting the higher complexity of the nine-class classification setting.

2.6.6 Combined inference pipeline

For end-to-end analysis we implemented a sequential pipeline:

1. Predict the number of vehicles with the *Vehicle Count Model*.
2. For each predicted vehicle (if the count equals `8+`, process 8 vehicles), query the *Vehicle Direction Model* using the vehicle-aware template.
3. Aggregate predictions into a JSON object and attach classifier confidence scores for downstream use.

The modular design facilitates independent debugging and ablation studies (e.g., evaluating a single stage or swapping backbone architectures).

2.6.7 Model persistence and reproducibility

Models and tokenizers were saved via `save_pretrained()`, including weights, tokenizer configuration, and label mappings. A JSON manifest recorded training metadata (hyperparameters, seeds, metrics, and timestamps). Fixed seeds across NumPy, PyTorch, and Hugging Face Transformers ensured reproducibility of all experiments.

2.6.8 Validation

Evaluation was carried out on a held-out 15% validation split with strict case-level separation to avoid data leakage. Performance assessment relied on both aggregate metrics (accuracy, F1) and per-class reports (precision, recall, F1). This ensured that validation captured overall correctness while also highlighting class-specific behavior, particularly for minority categories.

2.6.9 Metrics

Following established best practices in NLP evaluation, we report a comprehensive set of metrics [54, 55].

Accuracy measures the overall correctness of the model by calculating the proportion of correct predictions over all predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

This metric provides a general overview of model performance but can be misleading in imbalanced datasets where one class significantly outnumbers others.

Recall (also known as sensitivity or true positive rate) quantifies the model's ability to correctly identify all positive instances within the dataset:

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

High recall indicates that the model successfully captures most of the actual positive cases, minimizing false negatives. This is particularly important in crash analysis where missing critical events (false negatives) can have serious safety implications.

F1-score represents the harmonic mean of precision and recall, providing a balanced measure that accounts for both false positives and false negatives:

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

where Precision is defined as:

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

so we can simplify F1-score as:

$$\text{F1} = \frac{2 \times \text{TP}}{2 \times \text{TP} + FP + FN}$$

The F1-score is especially valuable when dealing with imbalanced crash datasets, as it provides a single metric that balances the trade-off between precision and recall. In these formulas, TP represents true positives, TN true negatives, FP false positives, and FN false negatives.

2.6.10 Summary

The fine-tuning of *BERT*, *RoBERTa*, and *SciBERT* provided supervised baselines for the vehicle direction extraction task. These models establish a point of comparison for the subsequent evaluation of larger LLMs, highlighting methodological trade-offs between compact, task-specific transformers and general-purpose instruction-tuned models. Their inclusion ensures that performance differences can be attributed not only to model scale but also to training paradigm, thereby strengthening the comparative analysis presented later.

2.7 Fine-tuning based information extraction

LLMs are effective for text-to-table transformation, converting unstructured crash narratives into structured fields [56]. While recent traffic-safety studies can extract explicitly stated details [57], reasoning-intensive variables remain challenging due to limited crash-specific knowledge and the prohibitive cost of full-model training. We address this by adopting parameter-efficient fine-tuning with LoRA [23] on open-source LLMs using an annotated crash dataset, injecting domain knowledge at low compute/memory cost and enabling safer, more controllable deployment than closed-source APIs. Our method targets two high-value variables that support surveillance, risk assessment, and countermeasure design: (i) *manner of collision*, a reasoning task over event sequences and interacting agents; and (ii) *crash type*, a fine-grained classification with approximately 100 categories.

2.7.1 Dataset

CISS dataset includes not only SUMMARY, but also structured case coding such as MANCOLL (manner of collision) and CRASHTYPE (crash type), which is technician-determined [58]. In the CISS dataset, MANCOLL is not directly annotated but rather derived through rule-based mapping from other variables, including OBJECT CONTACTED in table EVENT as well as CRASHTYPE and TRANSPORT in table GV. The resulting classification consists of seven categories as shown in Table 2.6, one of which corresponds to unknown.

Table 2.6: Class definitions for the MANCOLL task.

Label ID	MANCOLL
0	Not Collision with Vehicle in Transport
1	Rear-End
2	Head-On
4	Angle
5	Sideswipe, Same Direction
6	Sideswipe, Opposite Direction
9	Unknown

The CRASHTYPE is a numeric value derived through a two-step process: first by selecting the Crash Category (CRASHCAT) and Crash Configuration (CRASHCONF) in the GV table (as illustrated in Figure 2.4), and then by the crash technician's assessment based on police reports, scene inspections, vehicle inspections, and interviews. This two-step procedure, from crash category to crash configuration and finally to crash type, is preferred because it provides a more structured and interpretable way to visualize crash scenarios.

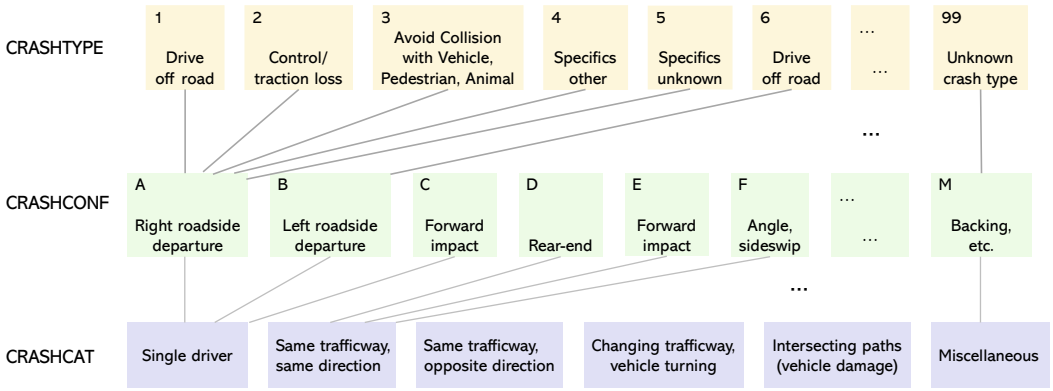


Figure 2.4: Hierarchical mapping of CRASHTYPE classification systems.

2.7.2 Task definition

To evaluate the capability of LLMs in deriving accurate information that requires deep inference directly from unstructured text, this project adopts the CISS dataset as a case study. We focus on analyzing the crash narratives (the SUMMARY column in the CRASH table) to infer structured collision-type information. Specifically, our goal is to identify the manner of collision for each crash and the crash type for all vehicles involved in a crash.

For the first task, we are interested in assigning the manner of collision based solely on the textual description of the crash (i.e., SUMMARY column in the CRASH table), without using any structured metadata. It require deep-reasoning beyond explicitly stated facts. We use the MANCOLL column in CRASH as ground truth labels.

For the second task, we want to extract the crash type for each vehicle in a crash. Unlike MANCOLL, which involves only seven classes, CRASHTYPE is considerably more challenging, encompassing nearly 100 fine-grained categories. To make this problem more tractable, we exploit the hierarchical taxonomy illustrated in Figure 2.4 and decompose the task into 13 smaller classification subtasks based on CRASHCONF. All subtasks are addressed within a single model. Since CRASHCAT and CRASHCONF classifications are relatively straightforward[5], we treat CRASHCONF as oracle knowledge and concentrate our analysis on the more difficult CRASHTYPE classification.

2.7.3 Workflow

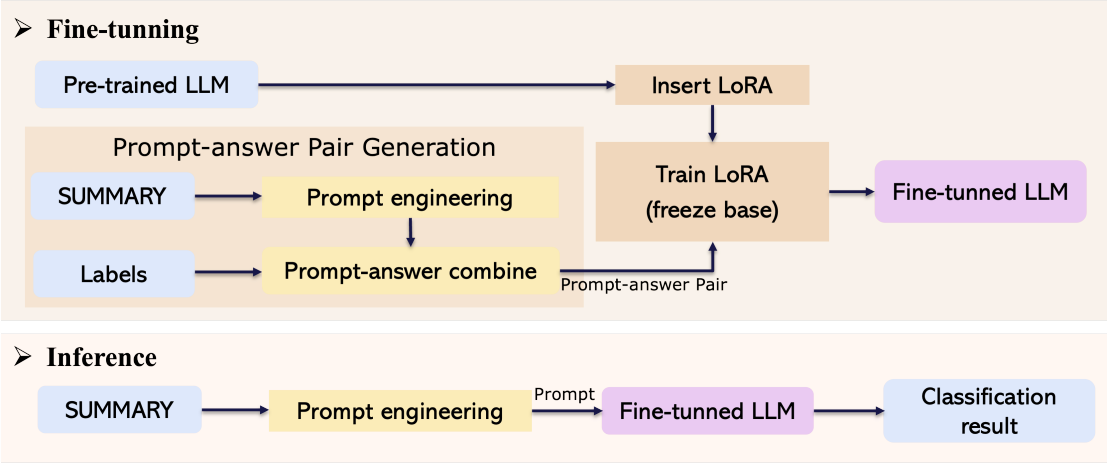


Figure 2.5: The figure illustrates the fine-tuning process.

Our overall framework is illustrated in Figure 2.5. We reformulate the information extraction task as a classification problem by prompting the LLM to output

[5]The CRASHCAT classification is very similar to MANCOLL, consisting of only six categories. After completing the first task, we found the two tasks highly overlapping and therefore did not pursue CRASHCAT further. The CRASHCONF categories are derived from CRASHCAT, with each CRASHCAT corresponding to only 1–3 CRASHCONF classes, making it a relatively simple classification task as well.

a predefined label rather than free-form text. Crash summaries and their corresponding labels in CISS are used to construct prompt–answer pairs, which serve as training samples for a LoRA adapter while keeping the base LLM frozen. During the fine-tuning stage, we employ prompt engineering methods to construct task introductions, present rules that incorporate task-specific knowledge, and provide clear instructions for the expected output. To achieve domain adaptation, we adopt parameter-efficient LoRA fine-tuning, which reduces computational overhead and accelerates training. In the inference stage, the fine-tuned model receives a narrative-based prompt and directly generates the corresponding classification result.

Prompt engineering

Prompt engineering is crucial for LLM performance, as subtle changes in wording or structure can significantly affect accuracy. In domain-specific tasks like crash narrative analysis, prompts must effectively embed domain knowledge to achieve reliable predictions. We designed a structured instruction-style prompt with clearly separated components. The prompt guides the model to understand the task, learn domain-specific category definitions, follow exception rules, and apply them to a given crash description. The full prompt can be divided into the following components:

(1) Task Introduction. This part contains expert-verified definitions for each category.

```
You are a helpful assistant that classifies vehicle collisions
into one of the following categories based on the description
provided. ...
```

(2) Clarification Rules. These clarification rules are derived from domain-specific heuristics, codified through expert consultation and data observation. Some rules were distilled from empirical insights obtained via manual inspection and a pilot study over some training examples. In addition to incorporating task-specific information into the prompts, we further enhance model performance by applying a *CoT* [59] strategy. Specifically, the prompts are designed to decompose complex reasoning tasks into step-by-step subproblems, thereby guiding the LLM to follow an explicit reasoning trajectory before arriving at the final categorical prediction.

(3) Output Instruction. Due to the generative nature and hallucination tendencies of LLMs, they may produce varied outputs even when the classification is conceptually correct. For example, in cases of angle collisions, the model may return "4", "angle", or free-form text such as "angled side impact". While semantically correct, such variations cannot be reliably processed in batch pipelines. Therefore, we explicitly instruct the model to focus solely on producing a valid class index from the predefined label space by using the prompt blew.

```
Only respond with a single number from the list above.
Do not add any explanation.
\end

\subsubsection*{Prompt in use: manner of collision}
As a concrete example, we reported the structured prompt used for the classific

\textbf{Task Introduction}
\begin{verbatim}
You are a helpful assistant that classifies vehicle collisions into one
of the following categories based on the description provided. Please
choose the most accurate collision type based on the definitions
and clarifications below:
```

Category Definitions (Expert Knowledge)

```
{{
  0: "Not Collision with Vehicle in Transport - The vehicle did not
  collide with another vehicle in motion.",
  1: "Rear-End - The front of one vehicle strikes the rear of another
  vehicle traveling in the same direction.",
  2: "Head-On - The front ends of two vehicles traveling in opposite
  directions collide.",
  4: "Angle - The front of one vehicle strikes the side of another
  at an angle (usually near intersections or crossing paths).",
  5: "Sideswipe, Same Direction - Both vehicles are moving in the same
  direction and **their sides make contact**",
  6: "Sideswipe, Opposite Direction - Both vehicles are moving in
  **opposite directions** and their **sides make contact**",
  9: "Unknown - The manner of collision cannot be determined."
}}
```

Clarification Rules (special prompting instructions)

```
Clarification:
If the collision happens at or near an intersection, classify as 4.
If it does not occur near an intersection:
and both vehicles are traveling in the same direction, classify as 5
and vehicles are traveling in opposite directions, classify as 6.
If the collision involves only one vehicle and a non-vehicle object
(e.g., animal, fence, tree), classify it as 0.
If no collision is described or it is unclear whether any impact occurred,
classify as 9.
```

```
If multiple collisions occur (e.g., chain reaction), classify based on
the **first** collision described in the summary.
```

Input Summary (classification target)

```
\"\"\"{summary}\"\"\"
```

Output Instruction (task constraint)

```
Only respond with a single number from the list above.
Do not add any explanation.
```

Prompt in use: manner of collision

As a concrete example, we reported the structured prompt used for the classification of the manner of collision. The prompt is designed according to the three components outlined above. The task introduction specifies the role of the model and provides expert-verified category definitions; the clarification rules encode domain heuristics (e.g., intersection-based disambiguation, chain reaction handling); the input summary placeholder is filled with the free-text crash description from CISS; and the output instruction forces the model to respond only with the index of the selected class. The full prompt used in this study is shown below:

Task Introduction

```
You are a helpful assistant that classifies vehicle collisions into one
of the following categories based on the description provided. Please
choose the most accurate collision type based on the definitions
and clarifications below:
```

Category Definitions (Expert Knowledge)

```
{{
  0: "Not Collision with Vehicle in Transport - The vehicle did not
  collide with another vehicle in motion.",
  1: "Rear-End - The front of one vehicle strikes the rear of another
  vehicle traveling in the same direction.",
  2: "Head-On - The front ends of two vehicles traveling in opposite
  directions collide.",
  4: "Angle - The front of one vehicle strikes the side of another
  at an angle (usually near intersections or crossing paths).",
  5: "Sideswipe, Same Direction - Both vehicles are moving in the same
  direction and **their sides make contact**",
  6: "Sideswipe, Opposite Direction - Both vehicles are moving in
  **opposite directions** and their **sides make contact**",
```

```
  9: "Unknown - The manner of collision cannot be determined."
}}
```

Clarification Rules (special prompting instructions)

```
Clarification:
If the collision happens at or near an intersection, classify as 4.
If it does not occur near an intersection:
and both vehicles are traveling in the same direction, classify as 5.
and vehicles are traveling in opposite directions, classify as 6.
If the collision involves only one vehicle and a non-vehicle object
(e.g., animal, fence, tree), classify it as 0.
If no collision is described or it is unclear whether any impact occurred,
classify as 9.
If multiple collisions occur (e.g., chain reaction), classify based on
the **first** collision described in the summary.
```

Input Summary (classification target)

```
\"\"\"{summary}\"\"\"
```

Output Instruction (task constraint)

```
Only respond with a single number from the list above.
Do not add any explanation.
```

Model and training

We evaluate several open-source large language models, including variants of LLaMA and Qwen, using instruction-style prompts. All models are integrated via the HuggingFace Transformers interface.

For model adaptation, we employ LoRA as our fine-tuning framework to efficiently specialize LLMs for the classification task.

Table 2.7: LLM Adaptation and Inference Configuration for Crash Classification

Component	Configuration
Base Models	Qwen-7B (training + inference), LLaMA-70B (inference only)
Adaptation Method	LoRA (Low-Rank Adaptation)
Training Framework	PEFT (Parameter-Efficient Fine-Tuning)
Backend Library	Hugging Face Transformers
Training Resource	Qwen-7B: 1×A100 GPU
Inference Resource	Qwen-7B: 1×A100 GPU; LLaMA-70B: 4×A100 GPUs

We follow a standard few-shot setting, using a small set of labeled examples to guide the model via in-context learning or LoRA-based fine-tuning.

LoRA adapter

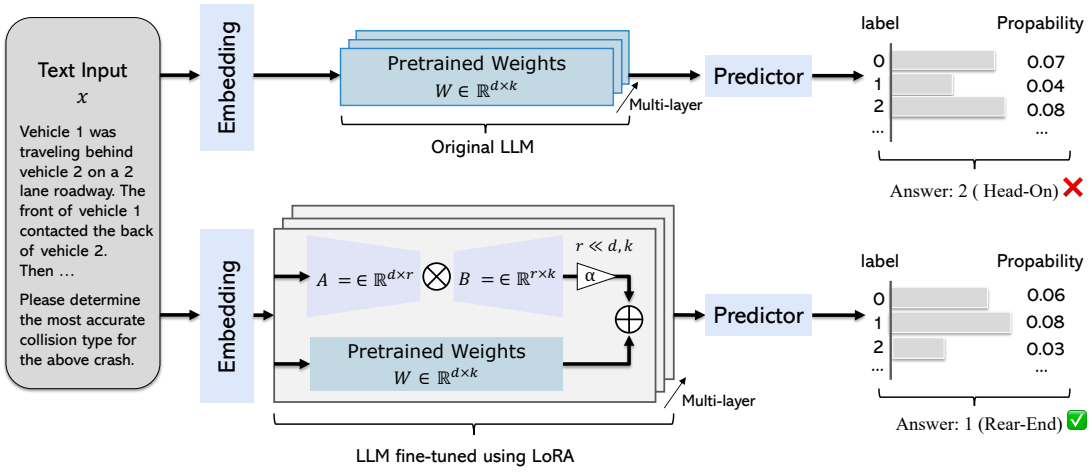


Figure 2.6: Prediction processes of the original LLM (top) and the LoRA fine-tuned LLM (bottom). For the same input, the original model fails to produce the correct answer, whereas the fine-tuned model adapts the LLM's original weights **W** by adding the product of low-rank matrices **A** and **B**, which alters the label prediction probabilities and enables the correct output.

The foundation of modern LLMs is the Transformer architecture, whose core component is the self-attention mechanism [60]. By scaling up this architecture to billions of parameters and training on massive text corpora, LLMs acquire broad linguistic and world knowledge. However, traffic-domain narratives (e.g., crash reports, incident logs, work-zone descriptions) are underrepresented in such corpora, so out-of-the-box LLMs struggle to capture domain-specific entities, relations, and causal patterns critical for crash narrative analysis. To bridge this gap efficiently, we adopt parameter-efficient fine-tuning by keeping the pretrained backbone frozen and learning lightweight adapters specialized for traffic narratives.

Self-Attention Primer (Where Adaptation Can Act) Given a token sequence represented by hidden states $X \in \mathbb{R}^{T\times d}$, a single attention head computes

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d\times k}$. The attention output is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{k}}\right)V, \tag{2.1}$$

This decomposition exposes three linear projections, W_Q, W_K, W_V, as natural adaptation points. For traffic narratives, adjusting them helps the model align with transportation-specific cues, and emphasize values relevant to causality.

LoRA for parameter-efficient domain adaptation Low-Rank Adaptation (LoRA) [23] inserts trainable low-rank matrices into selected frozen projections so that only a small number of parameters are updated. A comparison between the original LLM and the LoRA fine-tuned LLM is illustrated in Figure 2.6. For a target weight $W \in \mathbb{R}^{d\times k}$ (e.g., one of W_Q, W_K or W_V), LoRA learns a low-rank update

$$\Delta W = \frac{\alpha}{r} AB, \qquad A \in \mathbb{R}^{d\times r},\ B \in \mathbb{R}^{r\times k},\ r \ll \min(d, k),$$

and uses the adapted weight

$$W' = W + \Delta W \tag{2.2}$$

During fine-tuning, the large pretrained W is *frozen* and only A, B are trained. This reduces trainable parameters from $d \times k$ to $(d + k)r$ and lowers memory cost, while preserving general linguistic competence. Crucially, because attention is factorized into Q, K, V, placing LoRA on W_Q/W_K directly modulates the *attention weights* in Equation (2.1), enabling the model to learn traffic-specific matching patterns; placing LoRA on W_V refines *content aggregation and readout*, improving how incident attributes and causal chains are represented downstream. In our implementation, we apply LoRA to all the query, value projection layers, and key projection layers (W_Q, W_K and W_V) of the transformer blocks, which are empirically found to be effective for language modeling tasks [61].

Loss function In our classification task, each crash description is assigned a label corresponding to one of the predefined classes. To simplify the output, we encode all class labels as single-token numeric identifiers using a dictionary mapping, as shown in Table 2.6. Consequently, the model is trained to generate a single token representing the class ID for each input.

We adopt the standard cross-entropy loss for training. Cross-entropy is well-suited for this setting because it directly measures the divergence between the predicted probability distribution over the vocabulary and the one-hot target distribution [62], thereby encouraging the model to assign maximal probability to the correct class token. Although only a single token is expected as output, the

model still computes the loss over the entire vocabulary V, which is consistent with the standard token-level formulation of autoregressive language models.

Formally, let x denote the input sequence (i.e., the crash summary), and let $y \in \{0, 1, ..., k\}$ be the target class index, where k is the total number of classes. At the final decoding step, the LLM outputs a vocabulary-sized logit vector $\mathbf{z} \in \mathbb{R}^{|V|}$. The probability of generating the target class token is computed via the softmax function:

$$P(y \mid x) = \frac{\exp(z_y)}{\sum_{i=1}^{|V|} \exp(z_i)} \tag{2.3}$$

The training objective is to minimize the negative log-likelihood of the correct class token, which corresponds to the standard cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\log P(y \mid x) = -\log \left(\frac{\exp(z_y)}{\sum_{i=1}^{|V|} \exp(z_i)} \right) \tag{2.4}$$

Chapter 3

Results

In this section, we present the results of applying our models to crash narratives from the CISS dataset. The problem was categorized into three task types according to whether the target information is explicitly stated in the text and whether it involves single or multiple entities. Specifically:

- Section 3.1 reports on **travel direction extraction**, where summaries may contain multiple vehicles and the relevant information is explicitly expressed.
- Section 3.2 focuses on **manner of collision classification**, which involves a single entity per summary and relies primarily on implicit cues.
- Section 3.3 addresses **crash type classification**, which combines both challenges: multiple entities per summary and implicit target information.

3.1 Travelling direction

3.1.1 Evaluation methodology

The entire manually annotated subset of 1,000 crash instances with corresponding ground-truth labels was used to evaluate all LLMs (see Section 2.2). In contrast, the transformer baselines (BERT, RoBERTa and SciBERT) could not be directly evaluated on all 1,000 cases because they were tuned on the same annotated set. Rather, they were compared with LLMs on the annotated validation split created for the fine-tuning. Lastly, the transformer baselines were further applied to the entire CISS dataset of 17.459 instances, which does not have full manual labeling, in order to evaluate generalization at scale.

Evaluation therefore covers three complementary perspectives:

1. LLMs on annotated data: performance measured on the full set of 1,000 manually labeled cases.

2. LLMs vs. transformers: comparison limited to the validation split of the annotated subset.

3. Transformers on entire dataset: implementation of BERT models on the entire CISS dataset (17,459 without ground-truth labels).

As for evaluation metrics, we rely on the standard definitions of accuracy, precision, recall, and F1-score, as detailed in Section 2.6.9.

3.1.2 LLMs evaluation

Settings

Evaluation is divide in two different versions, mirroring the post–processing strategies in Section 2.5:

- Valid-only: only valid JSON predictions are used, and invalid outputs are counted as errors on the complete set of 1,000 cases.

- Valid and recovered: in addition to valid predictions, we include results successfully recovered from malformed responses, increasing coverage.

For both configurations, we report metrics with two levels of detail: *per vehicle* (percentage of vehicles with correctly predicted direction) and *per case* (a case is correct only if all vehicles are correct).

Valid-only results

Per-vehicle metrics. Table 3.1 reports total accuracy (correct over all vehicles), valid-only accuracy (excluding invalid/missing), and response rate (vehicles with a valid prediction).

Table 3.1: LLM performance in the valid-only setting at the per-vehicle level.

Model–Prompt	Total accuracy	Valid-only accuracy	Response rate
Mistral 7B — Zero-shot	0.680	0.714	0.953
Mistral 7B — Few-shot	0.680	0.709	0.960
Mistral 7B — CoT	0.789	0.832	0.948
DeepSeek-R1 8B — Zero-shot	0.811	0.910	0.892
DeepSeek-R1 8B — Few-shot	0.772	0.898	0.859
DeepSeek-R1 8B — CoT	0.877	0.903	0.972
Qwen3 8B — Zero-shot	0.887	0.894	0.992
Qwen3 8B — Few-shot	0.856	0.877	0.976
Qwen3 8B — CoT	0.915	0.921	0.993

Per-case metric Table 3.2 reports case accuracy. A case is counted as correct only if *all* vehicles are predicted correctly.

Table 3.2: LLM performance in the valid-only setting at the per-case level.

Model–Prompt	Case accuracy	Correct cases
Mistral 7B — Zero-shot	0.576	576 / 1000
Mistral 7B — Few-shot	0.601	601 / 1000
Mistral 7B — CoT	0.743	743 / 1000
DeepSeek-R1 8B — Zero-shot	0.777	777 / 1000
DeepSeek-R1 8B — Few-shot	0.742	742 / 1000
DeepSeek-R1 8B — CoT	0.846	846 / 1000
Qwen3 8B — Zero-shot	0.844	844 / 1000
Qwen3 8B — Few-shot	0.829	829 / 1000
Qwen3 8B — CoT	0.895	895 / 1000

Accuracy vs. response rate Figure 3.1 illustrates how the accuracy of a vehicle and the response rate are related for each model and prompt configuration. The graph shows that there is a trade-off: models that are stricter about generating valid JSON may be more accurate but cover fewer cases, while models that are more consistent in their responses may be less accurate. For instance, DeepSeek-R1 (few-shot) demonstrates notable accuracy with valid outputs but exhibits a lower response rate. Conversely, Qwen3 8B (CoT) achieves both elevated accuracy and nearly full coverage, signifying enhanced robustness. This implies that evaluation should take into account both dimensions, as a model with high accuracy but

frequent invalid outputs is less practical in application.

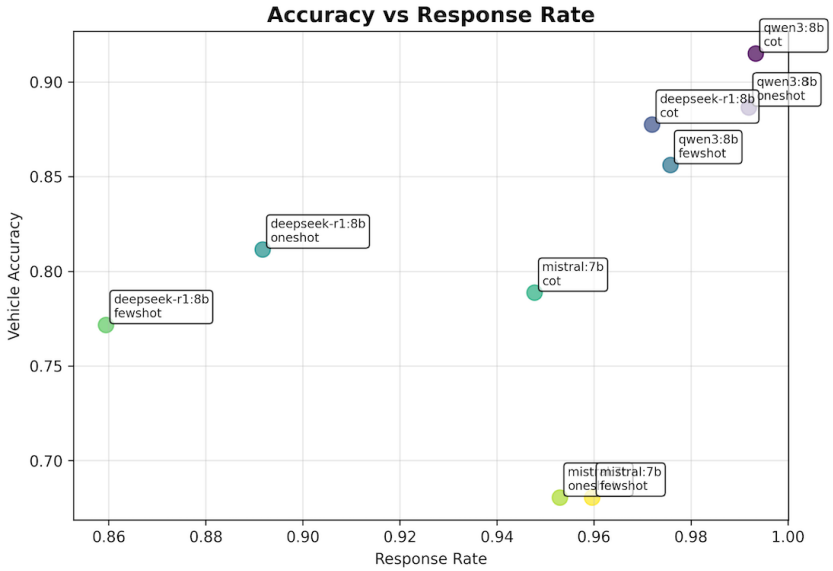


Figure 3.1: Valid-only setting: vehicle-level accuracy versus response rate across models and prompting strategies.

Interpretation of valid-only results The valid-only version represents a stringent evaluation criterion, wherein incorrect JSON outputs are rejected and classified as errors. In this context, three distinct patterns become evident.

Initially, prompting techniques significantly influence performance. In all models, CoT prompting consistently increases both per-vehicle and per-case accuracy when compared to zero-shot and few-shot prompting. Mistral 7B demonstrates an improvement from 0.576–0.601 case accuracy under zero-/few-shot prompting to 0.743 when applying CoT, signifying that sequential reasoning mitigates the misinterpretation of narrative crash descriptions.

The architecture of the model affects its robustness. Both DeepSeek-R1 8B and Qwen3 8B attain optimal results, with the latter demonstrating a vehicle accuracy of up to 0.915 and a case accuracy of 0.895, alongside a response rate of 0.99.

Third, a trade-off exists between precision and coverage. As shown in Figure 3.1, models exhibiting lower response rates (e.g., DeepSeek-R1 few-shot) attain elevated valid-only accuracy when outputs are generated; however, total accuracy is compromised due to the prevalence of invalid outputs.

Valid and recovered results

For completeness, we present the aggregate (per-vehicle / per-case) metrics for the *valid and recovered* version. This method demonstrates performance after repair

and incorporates recoverable faulty outputs, thus enhancing effective coverage and preserving schema consistency.

Per-vehicle metrics Table 3.3 presents the per-vehicle outcomes in the valid+recovered methodology. In the valid-only context, we provide (i) total accuracy across all vehicles, (ii) valid-only accuracy limited to records with applicable predictions, and (iii) response rate, which denotes the proportion of vehicles for whom a prediction was retrievable.

Table 3.3: LLM performance in the valid+recovered approach at the *per-vehicle* level.

Model–Prompt	Total Accuracy	Valid-only Accuracy	Response Rate
Mistral 7B — Zero-shot	0.682	0.714	0.955
Mistral 7B — Few-shot	0.681	0.708	0.961
Mistral 7B — CoT	0.793	0.829	0.957
DeepSeek-R1 8B — Zero-shot	0.881	0.914	0.964
DeepSeek-R1 8B — Few-shot	0.839	0.896	0.936
DeepSeek-R1 8B — CoT	0.877	0.903	0.972
Qwen3 8B — Zero-shot	0.887	0.894	0.992
Qwen3 8B — Few-shot	0.856	0.877	0.976
Qwen3 8B — CoT	0.915	0.921	0.993

Per-case metric Table 3.4 presents the case-level accuracy within the valid+recovered context. A case is considered correct only if every vehicle is predicted accurately, incorporating repaired predictions when recovery has been successful.

3.1.3 LLMs vs. fine-tuned transformers

Context

This section compares the large language models with the fine-tuned transformer baselines (BERT, RoBERTa, SciBERT; refer to Methods, Section 2.6) using the manually annotated subset. The evaluation was conducted solely on the validation split due to a limitation in our setup, as a portion of the 1,000 manually labeled cases had already been utilized for training and testing during fine-tuning. All fine-tuned models utilized a consistent train/validation/test partition, thereby ensuring a fair comparison.

Table 3.4: LLM performance in the valid and recovered setting at the per-case level.

Model–Prompt	**Case Accuracy**	**Correct Cases**
Mistral 7B — Zero-shot	0.579	579 / 1000
Mistral 7B — Few-shot	0.602	602 / 1000
Mistral 7B — CoT	0.744	744 / 1000
DeepSeek-R1 8B — Zero-shot	0.856	856 / 1000
DeepSeek-R1 8B — Few-shot	0.809	809 / 1000
DeepSeek-R1 8B — CoT	0.846	846 / 1000
Qwen3 8B — Zero-shot	0.844	844 / 1000
Qwen3 8B — Few-shot	0.829	829 / 1000
Qwen3 8B — CoT	0.895	895 / 1000

Fine-tuned transformers are expected to demonstrate superior performance within their respective domains due to optimization on task-specific data. Large language models can achieve comparable results even in the absence of supervised training. This contrast underscores the trade-off between domain-specific models and those with enhanced generalization capabilities.

BERT-family models setup

We initially imported the CSV files produced by the BERT models into dataframes for preliminary analysis. Each output file initially comprised one row per case, including a JSON-formatted field containing all vehicle-level predictions. This structure was then modified to ensure that each vehicle and its associated expected direction were represented as independent rows. Subsequent to this transformation, we validated that all models included predictions for the identical 388 unique case identifiers (those related to the validation set). This reorganization generated a uniform, vehicle-level dataset for all models, facilitating further comparisons and metric computations.

Table 3.5: Validation split over three BERT-family models. The table shows the number of cases, JSON-parsed vehicle-level predictions, and vehicle distribution per case.

Model	Cases	Vehicle-level records	Vehicle distribution
BERT	388	990	1v: 88, 2v: 192, 3v: 34, 5v: 47, 6v: 17, 7v: 6, 8v: 3, 14v: 1
RoBERTa	388	955	1v: 88, 2v: 192, 3v: 29, 5v: 78, 7v: 1
SciBERT	388	943	1v: 88, 2v: 192, 3v: 35, 5v: 72, 7v: 1

The consolidated dataframe, resulting from the alignment of model predictions, had a total of 1005 vehicle-level records, providing a uniform baseline for later comparisons.

LLMs setup

In the LLMs dataframe, we included exclusively the outputs derived from the processed raw JSON results, as delineated in Section 2.5.4. This guaranteed that all predictions maintained to a uniform format, facilitating direct comparison with the outputs of the BERT family.

Data merging and preprocessing

To facilitate a direct comparison between the BERT family models and LLMs, the two prediction dataframes were consolidated into a single unified dataset. Each row represents a vehicle within a case, incorporating both the BERT-based prediction and the LLM prediction for the vehicle's driving direction. This was executed using an inner join on `case_id` and `vehicle_id`, as illustrated below:

```
# Merge on common cases and vehicles
merged_df = pd.merge(
    bert_dataframe,
    llm_dataframe,
    on=["case_id", "vehicle_id"],
    how="inner",
)
```

Overall comparison of models

Table 3.6 indicates that the fine-tuned transformers established themselves as highly effective supervised baselines, achieving accuracies ranging from 95% to 96%.RoBERTa achieved the highest performance at 95.9%, closely followed by SciBERT at 95.8% and BERT at 94.9%. In LLMs, performance varied significantly based on both the model employed and the prompting approach adopted.The optimal configuration is Qwen3 8B utilizing CoT prompting (90.0% accuracy, 99.4% coverage), succeeded by the DeepSeek-R1 configurations (88-90%).Mistral 7B demonstrated accuracies between 67% and 80%, depending on the prompt chosen.

Two observations can be made:

- **Prompting strategy matters** Chain-of-thought prompting showed greater results across all LLMs, surpassing both zero-shot and few-shot techniques. Specifically, few-shot prompting yielded inferior accuracy compared to zero-shot prompting, both in this evaluation and in the prior assessment of the 1,000 manually labeled examples (Section 3.1.2). This outcome is likely associated with the selection of examples: if the chosen "shots" are unrepresentative, overly complicated, or add ambiguity, they may confuse the model instead of providing helpful guidance.

- **LLMs remain competitive** Despite the absence of fine-tuning, the highest-performing LLMs neared the efficacy of transformers, with Qwen3 8B (CoT) achieving 90.0%. The comparison exclusively included small LLMs, around 8 billion parameters. Larger variants of the same families (e.g., 30B, 70B, or

higher) are available and may possibly produce even more significant results. This highlights the efficacy of comprehensive pre-training for domain transfer, especially in specialized activities such as incident narrative analysis.

Table 3.6: Accuracy and coverage for fine-tuned transformers and LLM variants.

Model–Prompt	Accuracy	Coverage
BERT	94.9%	95.0%
RoBERTa	95.9%	94.8%
SciBERT	95.8%	95.2%
Mistral 7B—Zero-shot	69.9%	93.1%
Mistral 7B—Few-shot	67.1%	94.6%
Mistral 7B—CoT	80.3%	93.8%
DeepSeek-R1 8B—Zero-shot	89.9%	94.7%
DeepSeek-R1 8B—Few-shot	87.8%	91.3%
DeepSeek-R1 8B—CoT	88.6%	96.1%
Qwen3 8B—Zero-shot	89.6%	98.3%
Qwen3 8B—Few-shot	85.7%	97.1%
Qwen3 8B—CoT	90.0%	99.4%

Table 3.7: F1-scores, precision, and recall (macro-averaged and weighted) for fine-tuned transformers and LLM variants.

Model–Prompt	Macro F1	Weighted F1	Macro Precision	Macro Recall
BERT	89.2%	94.7%	92.6%	87.2%
RoBERTa	89.0%	95.9%	91.5%	87.0%
SciBERT	88.6%	95.7%	91.3%	86.5%
Mistral 7B—Zero-shot	14.8%	76.5%	15.4%	15.5%
Mistral 7B—Few-shot	34.9%	74.9%	38.3%	36.8%
Mistral 7B—CoT	15.4%	85.3%	16.2%	15.1%
DeepSeek-R1 8B—Zero-shot	52.3%	92.6%	53.9%	53.8%
DeepSeek-R1 8B—Few-shot	84.3%	91.0%	88.1%	86.2%
DeepSeek-R1 8B—CoT	77.1%	91.5%	79.8%	79.3%
Qwen3 8B—Zero-shot	80.1%	91.7%	80.7%	83.5%
Qwen3 8B—Few-shot	84.9%	89.2%	89.1%	86.8%
Qwen3 8B—CoT	78.9%	92.3%	81.1%	81.0%

Impact of prompting techniques

The efficacy of LLMs varies according to prompting methodologies. Figure 3.2 provides a performance heatmap that groups techniques and models together. This

shows that some techniques work better than others in certain situations. As previously stated, few-shot prompting achieved suboptimal results compared to analogous zero-shot configurations across the majority of models.

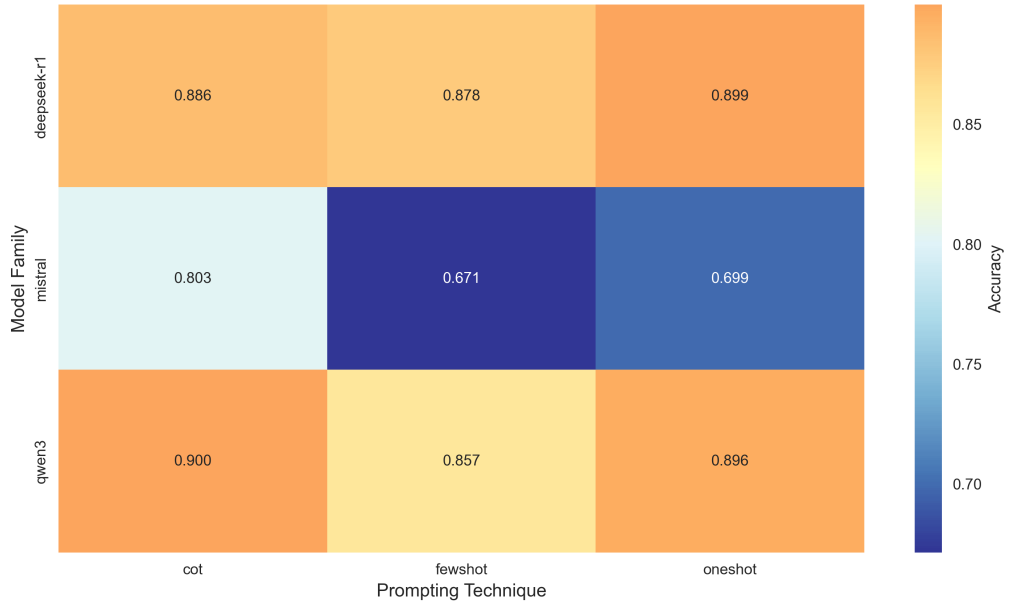


Figure 3.2: LLM performance by prompting strategy across models.

Specific behavior based on direction

Figure 3.3 illustrates that basic cardinal directions are typically easier to identify, whereas inter-cardinal classes and the category labeled “unknown“ present more difficulties across various systems. This aligns with prior findings (Section 3.1.2) and indicates that ambiguous or inadequately specified narratives are the primary source of disagreement. Cases involving parked vehicles may have introduced additional inconsistencies. In certain cases, the parking direction was explicitly indicated; however, these cases were still labeled as “unknown” due to the vehicle’s stationary status.

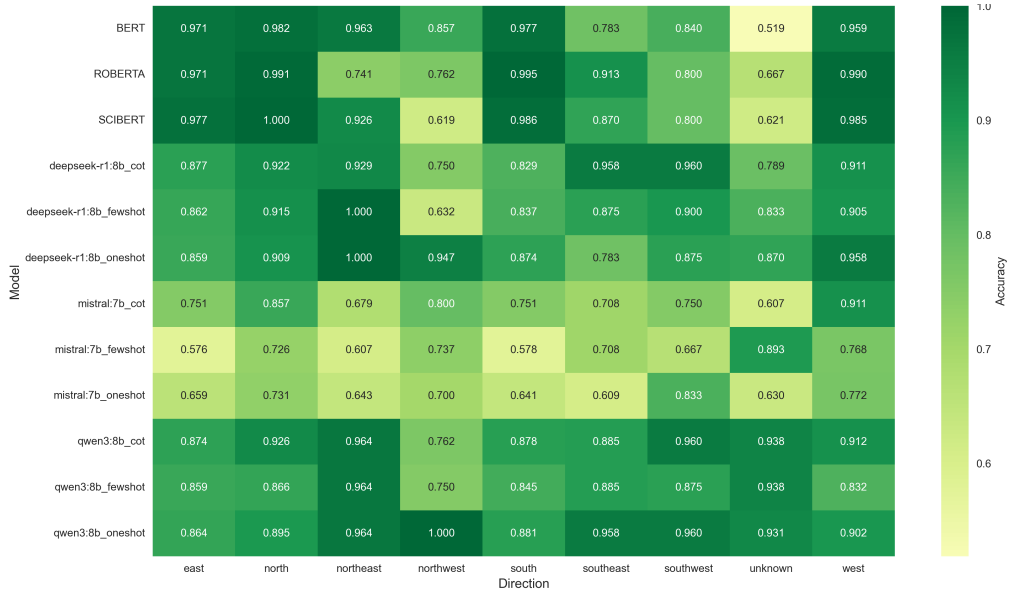


Figure 3.3: Per-direction accuracy heatmap across models.

Statistical significance

Figure 3.4 presents the outcomes of McNemar's test, which evaluated Qwen3 8B (CoT) against the three fine-tuned BERT models. The left panel illustrates the differences in accuracy, which are slight and consistently negative. This indicates that Qwen3 8B (CoT) underperformed BERT, RoBERTa, and SciBERT by approximately 1–5 percentage points. The right panel displays the p-values corresponding to the logarithmic scale. All comparisons fall below the significance threshold ($p > 0.05$), indicating that these differences lack statistical significance, although being observed in all three comparisons [63]. This indicates that while fine-tuned transformers provide numerically better results, the observed differences cannot be conclusively distinguished from random variation attributable to our sample size. The lack of statistical significance, combined with moderate effect sizes, indicates that both techniques, specialised fine-tuning and general-purpose prompting, produce comparable performance in vehicle direction extraction. This finding aligns with recent studies suggesting that the choice between fine-tuning and prompting often depends on pragmatic considerations, such as computational resources and deployment constraints, rather than substantial performance differences [64].

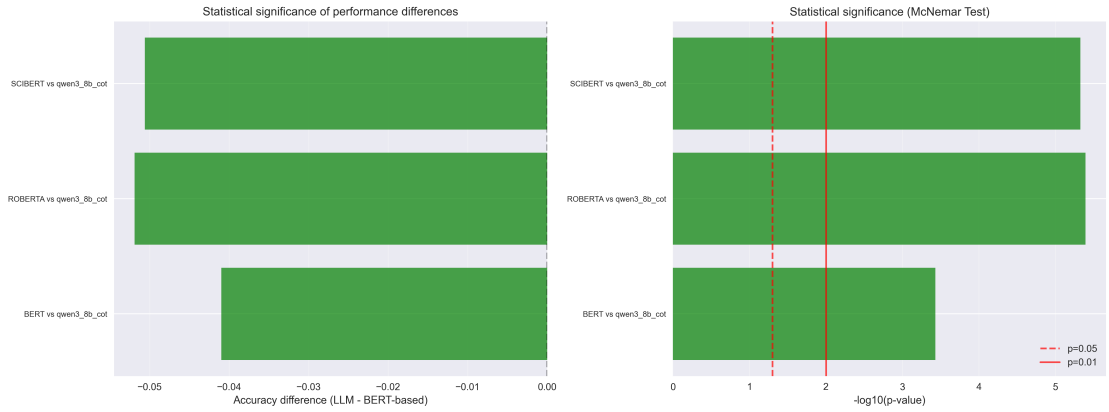


Figure 3.4: Statistical comparison of Qwen3 8B (CoT) against fine-tuned BERT models.

3.1.4 Analysis of BERT-family models on the full dataset

Dataset overview

This part of the evaluation covers the entirety of the 17.459 crash cases within the CISS dataset, surpassing the manually annotated subset.Here, the focus is on assessing the stability and consistency of model predictions at scale, in contrast to the earlier evaluations that relied on ground-truth labels.

The three transformer-based models that were previously introduced (BERT, RoBERTa, and SciBERT) are examined in terms of their agreement patterns, prediction distributions, and confidence levels. In total, the merged dataset comprises 29.962 vehicle-level predictions, reflecting cases with one or more vehicles described in the narrative.[1]

Data merging and preprocessing

The outcomes from the three models were consolidated into a single data set by merging on `case_id` and `vehicle_id`. This stage involved a function that consolidates the results of BERT, RoBERTa, and SciBERT into a singular dataframe. During the pre-processing phase, nested JSON fields were analyzed, and the standardization of vehicle identifiers was performed to ensure that results from various models could be compared under identical circumstances.

[1]The total number of predictions exceeds the number of crashes because each case may involve multiple vehicles, and the three models sometimes differed in the number of vehicles extracted from the same narrative.

Model agreement analysis

The degree of agreement across the three models was quantified using the function `analyze_bert_model_agreement()`. Results revealed a high level of consistency:

- **Complete agreement**: in 93.5% of cases, all three models produced identical directional predictions.
- **Partial agreement**: in 6.1% of cases, two models agreed while the third differed.
- **No agreement**: only 0.4% of cases showed three different predictions.

Pairwise evaluations validated this robustness: the concordance rate among model pairs varied from 94.6% to 96.8%, indicating significant alignment despite the lack of complete agreement. The results indicate that, despite being trained on different datasets, BERT, RoBERTa, and SciBERT typically converge on similar decision models when addressing this particular domain task.

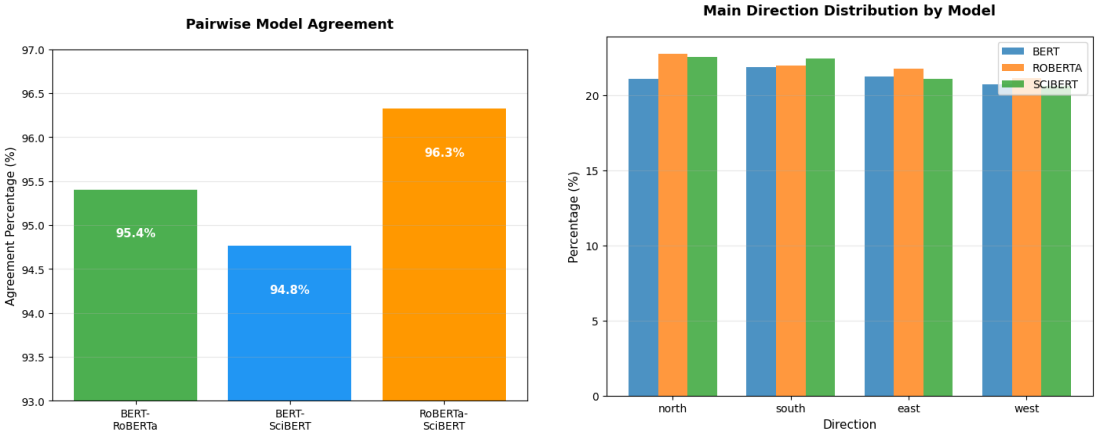
Prediction distribution

The distribution of predictions across the three models indicates that the majority of instances were allocated to the four primary directions: north, south, east, and west.The intermediate directions (northeast, northwest, southeast, and southwest) were less prevalent; however, they manifested with very similar rates across all models.The prediction for the label `unknown` was limited, consistently remaining below 15%. This suggests that the models often attempted to provide a directional prediction instead of leaving the question unresolved. The three models demonstrated quite analogous patterns in their predictions.

Visual summary To complete this analysis, Figure 3.5a shows the pairwise agreement rates between the models. The Figure 3.5b shows the distribution of predicted directions for the main directions. Together, these visualizations reinforce the observation that the three models not only agree in most cases, but also maintain stable prediction patterns across the entire set of directions.

High-confidence predictions We set a confidence threshold of 0.8 to identify any predictions with confidence scores over this value, thus improving our understanding of the model's reliability. Out of 28,004 cases where all three models agreed, 24,159 (86.3%) were categorized inside this high-confidence group. This indicates a significant correlation between agreement and reliability.

This evaluation demonstrates that the fine-tuned BERT-family models exhibit outstanding consistency when utilized on extensive crash narratives, achieving total



(a) Pairwise agreements **(b)** Prediction distributions

Figure 3.5: Visualization of BERT, RoBERTa, and SciBERT results on the full dataset.

agreement in 93.5% of cases. This level of agreement is noteworthy given that the models were trained separately, and it can be considered a form of indirect validation in the absence of full ground truth labels.

The strong agreement suggests that the directions described in crash reports follow linguistic patterns that can be reliably learned, regardless of the specific model variant. RoBERTa was the most confident and reliable of the group.

3.2 Manner of collision

3.2.1 Experimental setting

Backbones and baselines We evaluated multiple LLMs on the classification task under different fine-tuning configurations. The backbone models include several open-source families: LLaMA3 series [65] (LLaMA3.2-1B, LLaMA3.2-3B, LLaMA3.1-8B, LLaMA3.3-70B), Qwen series [66] (Qwen2.5-7B-Instruct), and Mistral series (Mistral-7B-Instruct-v0.3 [40]). We also evaluated the closed-source GPT-4o [67]. Parameter size and deployment resources of these models are shown in table 3.8[2]. To ensure a fair comparison, all experiments were conducted on a single NVIDIA A100 GPU, except for LLaMA3-70B, which required 4 A100 GPUs due to its larger size.

Table 3.8: Overview of backbone models: parameter size and deployment resources.

Model	Parameters	GPU Memory Required
Open-source models		
LLaMA3.2-1B	≈ 1.23 B	≈ 3 GB
LLaMA3.2-3B	≈ 3.21 B	≈ 6 GB
Qwen2.5-7B-Instruct	≈ 7 B	≈ 14 GB
Mistral-7B-Instruct-v0.3	≈ 7.3 B	≈ 14 GB
LLaMA3.1-8B	≈ 8 B	≈ 15 GB
LLaMA3.3-70B-Instruct	≈ 70 B	≈ 131 GB
Closed-source models		
GPT-4o	—	—

Dataset To assess the generalizability of our approach to future police reports, model training was performed on crash data of year 2020, consisting of 300 annotated

[2]Model sizes and deployment requirements are referenced from the official NVIDIA NIM documentation: `https://docs.nvidia.com/nim/large-language-models/latest/introduction.html`.

examples for prompt design and adjustment and about 3,000 examples for fine-tuning the LLM with LoRA. We evaluated on 2,000 test cases from 2021 to measure inference performance.

Hyper-Parameters Inference was conducted with a fixed temperature of 0.2 to ensure output stability, which is important in industrial application scenarios. Moreover, through multiple pilot experiments we observed that a temperature of 0.2 achieved the best trade-off between accuracy and robustness for the tasks considered in this study. Fine-tuning was performed with LoRA-based supervised fine-tuning for up to four checkpoints, using the same configurations to ensure fair comparisons. Following [68], we set $r = 8$ and $\text{LoRA}_\alpha = 16$, which balances training efficiency and performance for the classification task, providing sufficient task-specific adaptation without excessive computational cost. The original LLM's weights remained frozen during training.

3.2.2 Experimental results

To evaluate both the overall correctness of predictions and the balance of performance across different classes, we assessed model performance using accuracy and macro F1-score. In addition, deploying, training, and running inference with LLMs require substantial computational resources. To better understand the trade-offs, we compare models of different sizes and from different providers, reporting both training and inference times. These results are then analyzed in conjunction with accuracy and macro F1-score to provide a comprehensive evaluation.

During manual inspection, we found that *Unknown* (label ID: 9) in the original dataset could reasonably be reassigned to other valid categories. Therefore, in addition to reporting results on the original dataset, we also present evaluations with this label *Unknown* excluded to provide a more precise assessment of model performance.

Main results

The overall results of MANCOLL classification are summarized in Table 3.9. Fine-tuning brings significant improvements to closed-source models. For example, the 3B model improves from an initial accuracy of 50.2% to over 95.1% after fine-tuning. At the same time, the performance of fine-tuned LLaMA3-3B model is comparable to that of 7B and 8B models and outperforms GPT-4o, despite being significantly smaller in model size and computational requirements. This indicate that a 3B model is sufficient to address this task with similar performance.

Training efficiency is also notable: convergence is achieved in less than 1,700 steps, requiring only 890 seconds for the 3B model and 1,615 seconds for the 8B model. latency also remains low, with all models below 8B requiring less than

Table 3.9: Accuracy and Macro F1 scores of various LLMs on the Manner of Collision (MANCOLL) classification task.

Backbones	Tr_step	Tr_time (s)	Inf_time (ms)	Accuracy(%)		Macro F1	
				Raw	-Unknown	Raw	-Unknown
Open source models							
LLaMA3-1b	Original		25.6	1.6	0.0	0.005	0.000
	417	112.9	20.1	45.4	46.2	0.151	0.178
	834	226.7	21.1	81.9	83.3	0.359	0.423
	1251	339.8	20.4	85.8	87.2	0.407	0.479
	1668	452.4	20.0	**87.5**	**88.9**	**0.491**	**0.496**
LLaMA3-3b	Original		33.8	50.2	23.3	0.184	0.124
	417	222.1	33.5	92.8	94.4	0.690	0.815
	834	443.4	35.3	94.0	95.4	0.754	0.745
	1251	668.9	33.7	95.0	96.4	0.762	0.753
	1668	890.5	33.3	**95.1**	**96.4**	**0.779**	**0.753**
Qwen2.5-7B	Original		48.2	76.2	77.0	0.562	0.559
	417	396.2	49.7	92.7	94.1	0.680	0.679
	834	795.3	49.9	94.0	95.5	0.745	0.746
	1251	1192.7	49.5	**94.4**	**95.7**	**0.774**	**0.755**
	1668	1583.9	49.7	94.4	95.7	0.772	0.753
Mistral-7b	Original		49.7	81.7	83.1	0.553	0.561
	417	377.63	48.8	90.1	91.5	0.680	0.679
	834	756.12	49.8	92.4	93.3	0.765	0.729
	1251	1149.84	49.5	91.2	92.2	0.740	0.701
	1668	1512.86	49.7	**94.4**	**95.7**	**0.772**	**0.753**
LLaMA3-8b	Original		54.9	34.3	34.9	0.214	0.251
	417	406.6	56.8	94.4	96.0	0.749	0.886
	834	803.6	56.8	95.2	96.5	0.803	0.773
	1251	1209.8	56.6	95.9	97.1	0.833	0.789
	1668	1615.4	57.1	**96.1**	**97.1**	**0.848**	**0.788**
LLaMA3-70b	Original		508.3	91.4	92.7	0.692	0.704
Closed source models							
GPT-4o				90.9	92.3	0.674	0.805

60 ms per instance, and the 3B model achieving as little as 33 ms. The fast and training speeds are largely attributed by our problem formulation, where the task is cast as a classification problem and the LLM only needs to generate a single token as output.

Another important observation arises from the presence of an *Unknown* class in the raw dataset, representing unknown categories. Throughout our sample analysis, we found that many instances labeled as *Unknown* could in fact be reasonably mapped to known categories, and the employed LLMs were often able to classify them correctly. As shown in Table 3.9, removing the *Unknown* class yields a noticeable improvement in both accuracy and Marco F1 score, confirming the existence of potential noise in the original labels.

Overall, these findings demonstrate that with only a small number of fine-tuning steps, even relatively small-scale LLMs can be effectively adapted for the manner

of collision classification task.

Consistency analysis

To evaluate the stability of the models, we designed a set of consistency experiments, which include self-consistency, measuring the stability of repeated outputs from the same model, and cross-model consistency, measuring the agreement between outputs produced by different models.

Results are presented in Figure 3.6. The seven smaller subplots show the self-consistency results. The horizontal and vertical axes of each sub-figure represent predictions from different runs of the same model, while the values in the grid cells indicate the corresponding consistency scores. As observed, with the exception of LLaMA3-1B, most models achieve very high self-consistency, indicating that their outputs are stable across repeated predictions. The relatively low performance of LLaMA3-1B also explains its weaker consistency.

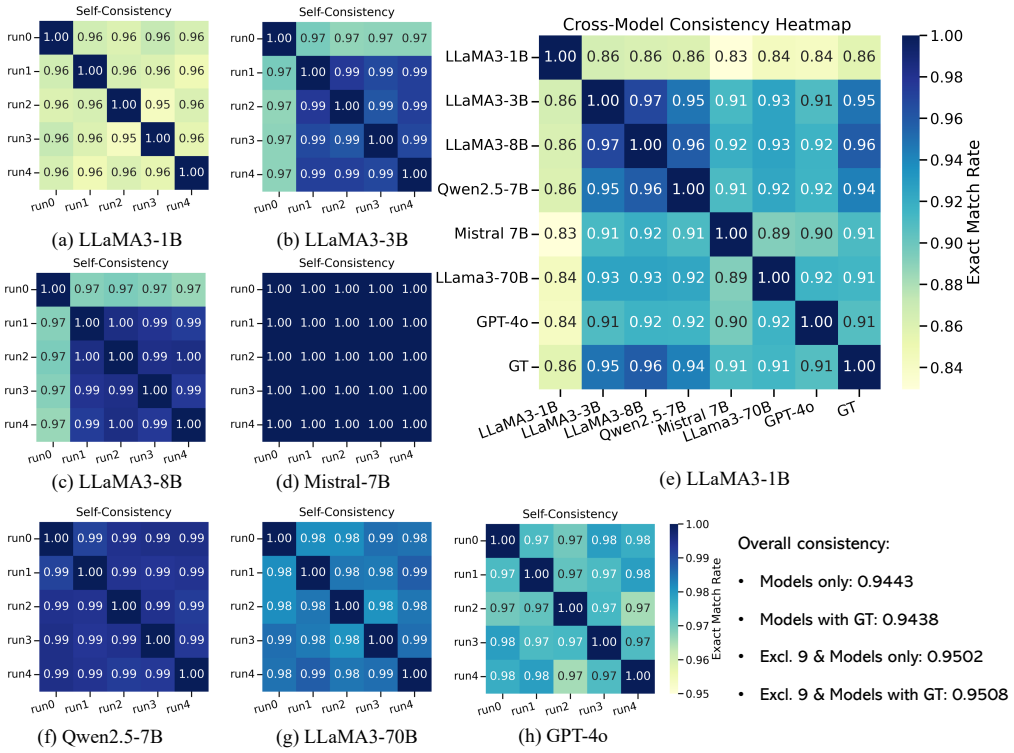


Figure 3.6: Heatmap results of consistency in the MANCOLL classification task. Each of the seven small heatmaps (left) illustrates the self-consistency of an individual model, while the large heatmap (top-right) shows the cross-model consistency among them and ground truth labels (GT) in CISS.

The larger plot in the upper-right corner illustrates the cross-model consistency results. Here, we find that consistency scores between most model pairs exceed 0.9, with the notable exception of LLaMA3-1B, whose weaker performance leads to lower agreement with other models. Furthermore, when comparing model predictions against the ground truth labels, fine-tuned models including LLaMA3-3B, LLaMA3-8B, and Qwen2.5-7B achieve significantly higher consistency than non fine-tuned GPT-4o and LLaMA3-70B. This finding is consistent with the improvements reported earlier in Table 3.9.

Overall, although cross-model consistency scores are slightly lower than those of self-consistency, they remain consistently high across models. These results demonstrate the robustness of our fine-tuning approach.

Data analysis

In the original database, the proportion of samples labeled as Unknown is about 1.6%. Previous experiments demonstrated that LLMs achieve strong overall performance on the MANCOLL classification task, with some models reaching up to 97% accuracy. Notably, the models were also able to reassign many of the samples originally labeled as Unknown into more specific categories. To further investigate this behavior, we conducted a detailed analysis focusing on the predictions for the Unknown subset.

As shown in Figure 3.7, categories 2 (Head-On) and 6 (Sideswipe, Opposite Direction) consistently appear with very low proportions across all models. This is partly because these categories are rare in the original ground-truth annotations, and also because narrative descriptions often provide limited clues for such cases. In contrast, models (c)–(f) maintain a relatively stable distribution pattern, showing that LLMs not only resolve the ambiguity in Unknown cases but also exhibit strong cross-model consistency, which enhances the reliability of the results.

3.3 Crash type

Compared with manner of collision, the extraction of crash type extraction is more challenging. There are 97 categories in total, and due to the hierarchical structure 2.4, the set of candidate CRASHTYPE is determined by the associated CRASHCONF. This decomposes the task into 13 smaller classification subtasks (one for each CRASHCONF), with the largest subtask containing up to 14 classes. Another source of difficulty is that a single crash may involve multiple vehicles. When classifying the CRASHTYPE for one vehicle, the narrative often contains descriptions of other vehicles, which can interfere with or complicate the judgment for the target vehicle.

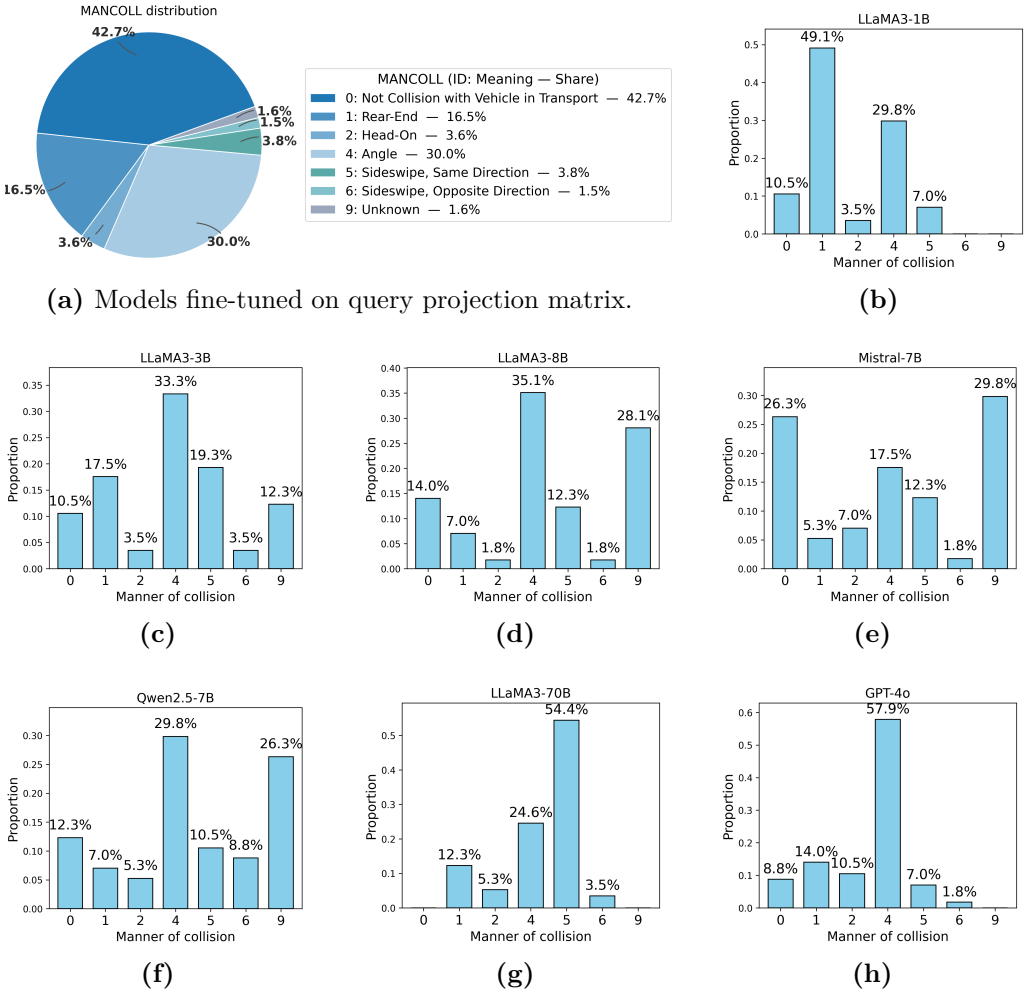


Figure 3.7: Distributional analysis of MANCOLL cases originally labeled as Unknown. (a) The original distribution of MANCOLL in database, where Unknown accounts for about 1.6%. (b–h) LLM-based reclassification results across different models. The models not only resolve many of the originally Unknown cases into specific categories, but also show consistent distribution patterns.

Considering above, it remains challenging for LLMs even though all are within the traffic safety domain. To better understand how model performance can be improved when encountering such complex task, we further experiment with different LoRA projection configurations and examine their impact on fine-tuning results. As described earlier in the database section, CRASHCONF classification is relatively easy to achieve high accuracy. Therefore, in this part we treat CRASHCONF as oracle knowledge and focus only on the classification of CRASHTYPE.

Main results on crash type

In reality, one crash may involve multiple vehicles and other vehicle descriptions can affect the prediction for the target vehicle. To account for this factor, we evaluate model performance by analyzing classification accuracy under different vehicle-count settings, grouping crashes into four categories: 1, 2, 3, and more than 3 vehicles.

Table 3.10 reports the performance of different LLMs on CRASHTYPE classification under varying numbers of vehicles per crash. The results show that this is a highly challenging task: without fine-tuning, most models achieve very low accuracy, typically in the range of 1%∼25%. However, after parameter-efficient fine-tuning, all models see substantial improvements. In particular, LLaMA3-3B, LLaMA3-8B, and Qwen2.5-7B consistently reach close to $\sim 80\%$ accuracy across different vehicle settings. Even the smallest 1B model, once adapted to the traffic safety domain, can surpass GPT-4o and the much larger LLaMA3-70B. Moreover, these smaller models achieve such performance with significantly lower training costs (less than 1 GPU hour) and inference requirements, striking an effective balance between efficiency and accuracy.

When analyzing results across different vehicle counts, we observe that collisions involving exactly two vehicles yield the lowest accuracy for all models. This is expected. Single-vehicle cases are the easiest, as the crash narrative only describes one vehicle, no textual interference from other vehicles. Multi-vehicle crashes involving three or more vehicles often correspond to chain collisions (e.g., multi-car rear-end crashes), where multiple vehicles share the same crash type. This homogeneity reduces ambiguity and makes classification more straightforward, despite the larger number of vehicles. The most challenging setting is crashes involving exactly two vehicles. In these cases, the narrative usually interweaves descriptions of both vehicles, and their crash types are often different. Consequently, two-vehicle crashes represent the most difficult scenario, demanding stronger reasoning and disambiguation capabilities from the models.

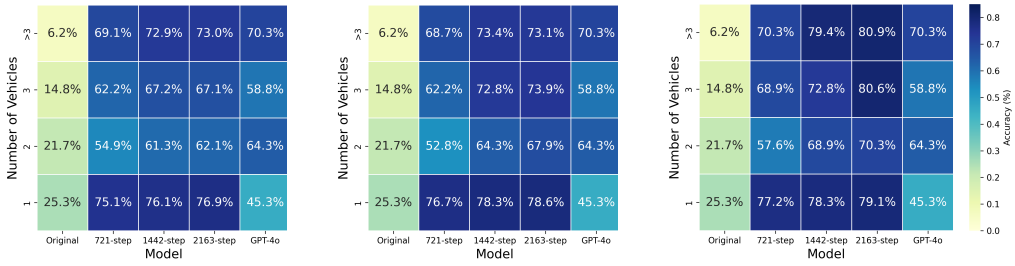
Effect of LoRA Training Strategies

Given the complexity of the *Crush Type* task, we further explore how the choice of LoRA-adapted parameters impacts performance. Specifically, we vary which attention projection matrices are trainable under LoRA, selecting from the query, key, and value projections $W_Q, W_K, W_V \in \mathbb{R}^{d\times k}$. We consider three configurations: (i) LoRA on W_Q only; (ii) LoRA on W_Q and W_V; and (iii) LoRA on all three, W_Q, W_K, and W_V. For each setting, we measure the model's self-consistency, and report results in Figure 3.8.

Results show that tuning more projection matrices consistently improves accuracy, with self-consistency increasing monotonically as more attention matrices

Table 3.10: Performance of different LLMs for CRASHTYPE classification under different settings. The best results are highlighted in bold.

Backbones	Tr_step	Tr_time (s)	No. Veh/Crash			
			=1	=2	=3	>3
Open source models						
LLaMA3-1b		Original	7.1	3.6	3.7	1.3
	721	198.4	49.1	26.7	43.3	45.1
	1442	397.2	43.7	42.7	64.1	75.2
	2163	594.7	52.6	49.4	64.5	73.4
	2884	780.3	**62.2**	**51.4**	**64.7**	**75.4**
LLaMA3-3b		Original	50.2	23.3	18.4	12.4
	721	430.1	73.9	53.7	67.1	77.4
	1442	842,9	74.0	69.5	78.9	81.9
	2163	1249.8	73.6	71.5	82.6	83.9
	2884	1683.7	**76.0**	**73.7**	**81.8**	**83.6**
Qwen2.5-7B		Original	25.3	21.7	14.8	6.2
	721	659.3	77.2	57.6	68.9	70.3
	1442	1308.5	78.3	68.9	72.8	79.4
	2163	1965.2	**79.1**	**70.3**	**80.6**	**80.9**
	2884	2605.5	77.2	72.9	80.1	81.5
LLaMA3-8b		Original	40.4	18.9	19.0	15.1
	721	694.3	73.9	53.7	67.1	75.2
	1442	1387.5	76.6	77.0	83.3	83.6
	2163	2088.5	**77.1**	**77.3**	**82.0**	**84.8**
	2884	2780.7	77.6	77.3	81.9	81.2
LLaMA3-70b		Original	72.7	41.8	44.3	55.0
Closed source models						
GPT-4o			45.3	64.3	58.8	70.3



(a) Models fine-tuned on query projection matrix.

(b) Models fine-tuned on query and value projection matrices

(c) Models fine-tuned on query, key and value projection matrices.

Figure 3.8: Accuracy (%) across different numbers of vehicles for various fine-tuning strategies. (a) Fine-tuning only the query projection, (b) fine-tuning both query and value projections, and (c) fine-tuning query, key, and value projections.

are adapted: updating W_Q alone yields the weakest performance; adapting both W_Q and W_V performs better; and jointly adapting W_Q, W_K, and W_V achieves the best results. We attribute this to the task's requirement for token interactions: restricting adaptation to queries constrains the model's ability to reshape attention patterns; co-adapting keys improves query–key alignment for retrieval, while adapting values allows the model to better propagate matched evidence through the representation. In short, enabling LoRA on multiple attention projections provides the necessary degrees of freedom to fit the crash type task more effectively.

Consistency analysis

By statistics, crashes involving one or two vehicles account for nearly the same amount, and together they represent over 93% of all cases. Since such crashes are also of primary interest to researchers, our consistency analysis focuses on these two settings.

For single-vehicle setting, as shown in Figure 3.9, we observe that LLaMA3-3B, LLaMA3-8B and Qwen2.5-7B show near-perfect run-to-run agreement (~ 0.98), while smaller models LLaMA3-1B maintain a consistency (~ 0.87). This indicates that fine-tuned models yield stable outputs across different inference runs on crash type classification. The agreements between LLaMA3-3B, LLaMA3-8B and Qwen2.5-7B are also high (~ 0.91), suggesting that different LLMs converge to similar predictions despite variations in size and architecture. When compared with CISS ground-truth labels (GT), the average consistency is slightly lower (~ 0.78), reflecting the possible presence of annotation noise and labeling errors in the dataset, which LLMs may help to mitigate.

For the two-vehicle setting, as shown in Figure 3.10, we observe a similar trend as in the single-vehicle case: larger models such as LLaMA3-3B, LLaMA3-8B, and Qwen2.5-7B maintain high run-to-run agreement (≥ 0.97), while smaller models like LLaMA3-1B achieve a reasonable level of consistency (~ 0.90). However, cross-model agreements among LLaMA3-3B, LLaMA3-8B, and Qwen2.5-7B drop noticeably (to around 0.79), indicating that when multiple vehicles are involved, textual descriptions of other vehicles interfere with the prediction of the current vehicle, thus reducing model accuracy. Compared with CISS ground-truth labels (GT), the average consistency (~ 0.78) is also lower than in the single-vehicle setting, reflecting the added complexity and ambiguity of multi-vehicle crash narratives.

Data analysis

To further verify that the labels generated by the LLM do not substantially alter the intrinsic characteristics of the data, we compared some statistics between LLM-generated labels and ground-truth. Here we primarily focus on accidents involving one or two vehicles, which together account for approximately 93% of

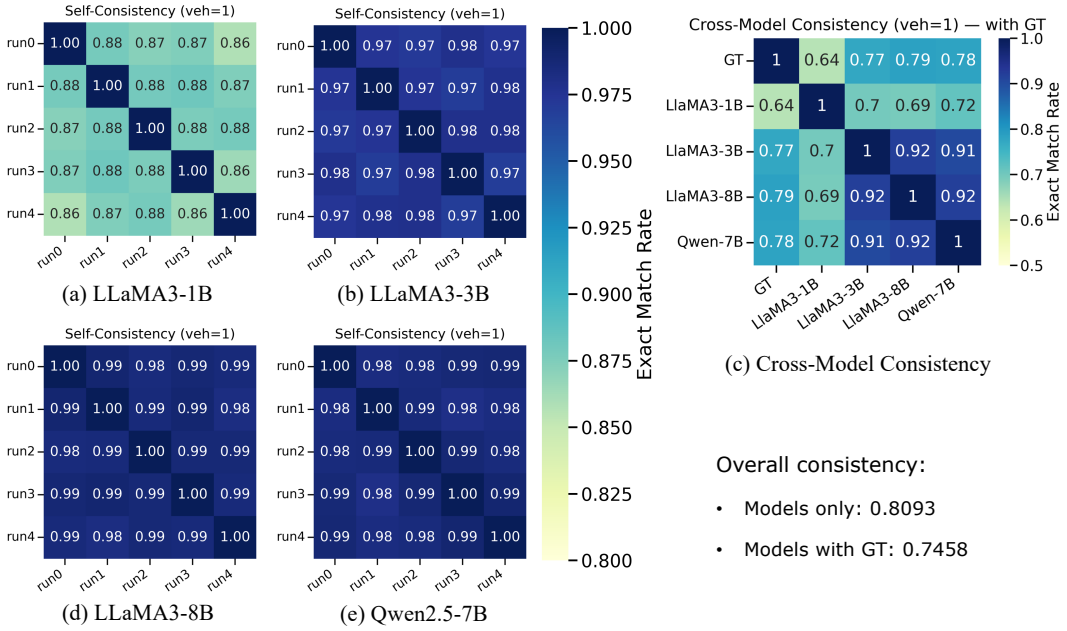


Figure 3.9: Single-vehicle setting: self-consistency of four models (left) and their cross-model consistency (right).

the dataset. For single-vehicle crashes, we conduct distributional analysis of crash types, while for two-vehicle crashes, we perform correlation analysis to assess the association between the crash types of the two vehicles. **Single-vehicle setting.** To characterize single-vehicle crashes, we analyze the distribution of their crash types. Specifically, we compute the frequency distribution of the most common crash categories (1-16), and the overall Divergence of the distribution. Since some crash types in the distribution may have zero or near-zero frequencies, we adopt Jensen–Shannon (JS) divergence to measure the difference between the two distributions, as it is symmetric, bounded, and robust to zero-probability events, thereby avoiding the divergence issues inherent in Kullback–Leibler divergence.

Table 3.11: Examples of single-vehicle CRASHTYPE categories related to road departure

Code	Description	Departure Direction
1	Roadside departure under a controlled situation.	Right
2	Roadside departure because of lost traction or control.	Right
6	Roadside departure under a controlled situation.	Left
7	Roadside departure because of lost traction or control.	Left

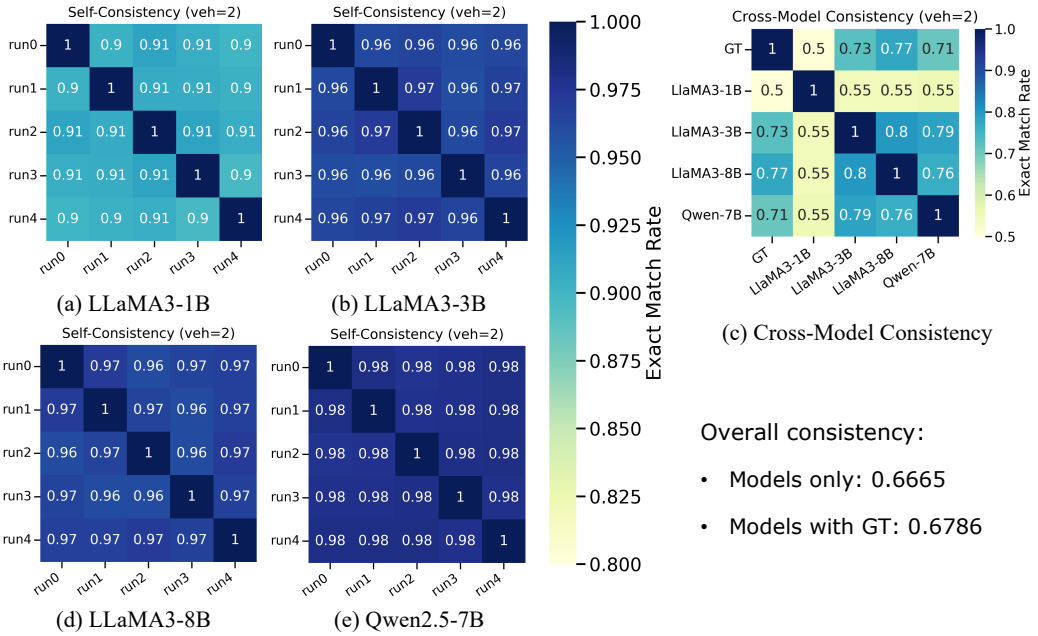
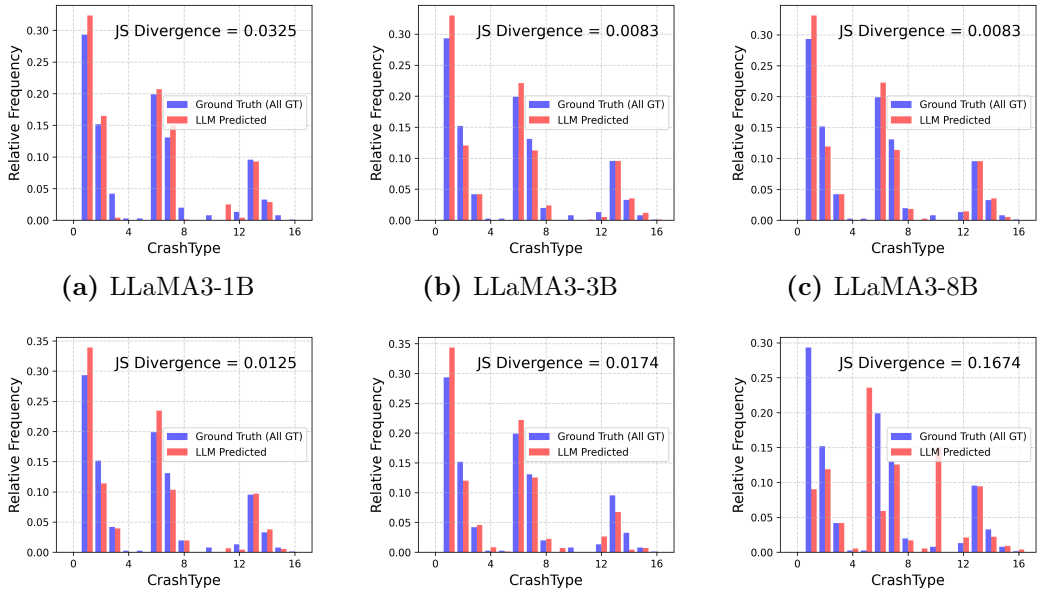


Figure 3.10: Two-vehicle setting: self-consistency of four models (left) and their cross-model consistency (right).

Figure 3.11 presents the predicted distribution of single-vehicle CRASHTYPE across different models, where only the first 16 crash categories are included since single-vehicle data is restricted to this subset. Overall, the fine-tuned models successfully preserve the distributional characteristics of the ground-truth data: the relative frequency and ranking of each class are largely consistent with the ground truth labels. Moreover, except for the 1B model, all fine-tuned models achieve closer alignment with the ground-truth distribution compared to much larger non-fine-tuned models such as LLaMA3-70B and GPT-4o. Besides, noticeable discrepancies remain for certain categories, particularly CRASHTYPE 1, 2, 6, and 7[3]. Through manual inspection of the CISS dataset, we found that part of this mismatch arises from inconsistencies in the recorded ground truth in CISS, where some cases were not assigned correctly.

Two-vehicle setting. The crash types of vehicles in a single accident are naturally related to each other. To examine the extent to which the crash types of two involved vehicles are related, we quantify this relationship by computing correlation coefficients. We employ Kendall's Tau as the measure of association. Since the crash type values are discrete identifiers rather than continuous quantities,

[3]The detailed definitions of these crash types are provided in Table 3.11



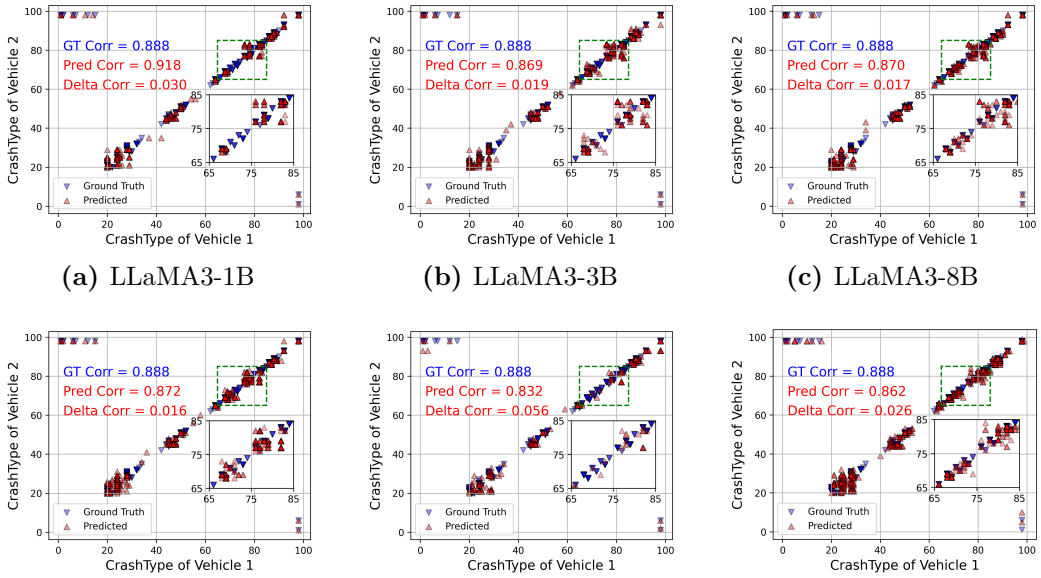
(a) LLaMA3-1B **(b)** LLaMA3-3B **(c)** LLaMA3-8B

(d) Qwen2.5-7B **(e)** LLaMA3-70B **(f)** GPT-4o

Figure 3.11: Comparison of CRASHTYPE distributions between the CISS ground truth (blue) and LLM-predicted results (red) for crashes involving a single vehicle. Each subplot corresponds to a different model, with the JS divergence reported as a measure of distributional similarity. Overall, the fine-tuned LLMs preserve the ground-truth distribution more closely than GPT-4o.

Kendall's Tau is more appropriate than *Pearson* or *Spearman* correlation [69]. It is important to note that in this task, a higher correlation is not necessarily better; instead, what matters is whether the correlation obtained from the predicted labels is closely aligned with the ground-truth correlation.

Figure 3.12 illustrates the correlation between the crash types of two vehicles involved in the same crash. Fine-tuned models generally achieve correlations very close to the ground truth (GT Corr = 0.888), and we observe that models with higher overall classification accuracy tend to produce predicted correlations that are closer to the ground-truth correlation. Importantly, across all fine-tuned models, the predicted correlations remain very close to the ground-truth value, with only minor deviations. This indicates that the fine-tuned models not only achieve high prediction accuracy at the individual crash type level but also successfully preserve the intrinsic inter-vehicle dependency patterns present in the CISS dataset.



(a) LLaMA3-1B **(b)** LLaMA3-3B **(c)** LLaMA3-8B

(d) Qwen2.5-7B **(e)** LLaMA3-70B **(f)** GPT-4o

Figure 3.12: Correlation analysis of CRASHTYPE combinations between two vehicles in the same crash. Blue markers represent CISS ground truth (GT) and red markers represent LLM predictions, with Spearman correlation coefficients reported for both.

Chapter 4

Discussion

Through the analysis of the results from the various questions, we identify several findings.

4.1 Vehicle travel direction

The initial research question focused on the ability of various models to accurately determine the direction of travel of vehicles involved in an collision. Comparative analysis revealed distinct patterns between optimised transformers and LLMs when tested with different prompting strategies.

4.1.1 LLM prompting strategies

A significant and rather surprising finding was the lower performance of few-shot prompting relative to zero-shot across all evaluated LLMs. This is compatible with recent research that shows how important it is to choose the right examples for in-context learning: inadequately chosen examples can even make performance worse than zero-shot baselines [70, 71]. Therefore, it is correct to attribute the decline in accuracy not to intrinsic constraints of few-shot prompting, but to the insufficient representativeness or clarity of the selected shots. This indicates that more methodical strategies for example selection may enhance the results. In opposite, CoT prompts significantly improved accuracy across all large language models. By breaking down the task into sequential steps, initially detecting cars, followed by analyzing directional clues, CoT prompts offered planned instructions that proved particularly effective in complex narratives involving multiple vehicles and interactions. This confirms the growing evidence that explicit reasoning strategies substantially improve information retrieval tasks that require multi-step inferences [26].

To further investigate these effects, we compared performance under two complementary evaluation methods. The "valid-only" version establishes a conservative lower limit by categorizing any formatting problems as unsuccessful predictions, thus underscoring the importance of precise compliance with output accuracy. Conversely, the "valid+recovered" methodology estimates an upper limit by include predictions that can be corrected and restored following error correction, hence representing the model's prospective efficacy in real-world scenarios involving post-processing.

This comparison yields two essential observations. Initially, prompting tactics, especially CoT prompting, consistently enhance performance in both evaluation methods, highlighting their efficacy in guiding large language models through intricate narrative reasoning. The models exhibit significant differences in their ability to adapt to formatting errors: Qwen3 8B sustains high performance regardless of error correction, whereas DeepSeek-R1 demonstrates substantial enhancements when recovered outputs are incorporated (an increase from 0.77 to 0.88), and Mistral 7B reveals persistent limitations in semantic accuracy despite recovery efforts.

From an application point of view, these findings underscore the necessity to assess LLM models for both accuracy and comprehensiveness. High accuracy is insufficient if accurate predictions encompass only a limited range of situations, whereas comprehensive coverage with decreased semantic precision is also unsuitable for reliable implementation. By examining both valid-only and valid+recovered versions, we enhance our comprehension of each model's strengths and weaknesses, facilitating a more equal and informative comparison of LLM models.

4.1.2 LLMs vs. fine-tuned transformers

Fine-tuned BERT transformers achieved superior overall accuracies, ranging between 94 and 96%, establishing them as the top-performing approach for this classification task. However, LLMs demonstrated remarkably competitive performance, with the best configuration (Qwen3 8B with CoT prompting) reaching approximately 90% accuracy. This result is particularly noteworthy considering that it was achieved using a relatively compact 8B-parameter model with advanced prompting techniques alone, without any task-specific fine-tuning.

Statistical analysis 3.1.3 revealed that the performance differences between the two approaches were not statistically significant, suggesting comparable effectiveness for practical applications. This finding aligns with recent studies indicating that the gap between task-specific fine-tuning and sophisticated prompting strategies has narrowed considerably as LLM pretraining and prompting methodologies have evolved [64].

The solid performance of the 8B LLM indicates that larger models or additional

optimization techniques, such fine-tuning, ensemble methods, or more sophisticated prompting strategies may effectively match or even surpass the performance gap with fine-tuned transformers. LLMs provide notable benefits regarding generalization capabilities, as they do not require retraining to adjust to new scenarios and exhibit enhanced flexibility in situations where labeled data is limited. [7].

Fine-tuned models work best when the target data is very similar to the training distributions. This makes inference predictable, reduces the amount of computing needed, and gives deterministic outputs, which play a key role in applications where reliability and safety are critical [72]. Conversely, LLMs provide exceptional adaptability and simple implementation, making them especially beneficial for dynamic environments and exploratory research contexts.

4.1.3 Consensus analysis of BERT-family on the full dataset

The models in the BERT family showed exceptional internal consistency, with 93.5% of cases showing complete agreement between the three variants. This demonstrates that the way traffic accident reports are written follows consistent linguistic patterns that different systems are able to capture reliably. RoBERTa, in particular, showed a slightly higher level of reliability. A group of 24,159 predictions was identified in which all three models agreed and the confidence level was above 0.8. Approximately 80% of all cases of agreement fall within this high-confidence consensus set and constitute a particularly valuable resource. It can be used as pseudo-ground truth to train more models, to help design future NLP pipelines, or as a reference dataset for active learning [73, 74]. The models were able to reliably extract meaningful directional patterns from the narratives, as evidenced by the size and reliability of the results.

Together, these findings indicate that the extraction of travel direction is a feasible and reliable task for both fine-tuned transformers and properly prompted LLMs. Fine-tuned models are the most stable choice for domain-specific deployment; however, LLMs utilizing advanced prompting techniques, especially CoT, demonstrate competitive performance and enhanced adaptability. Hybrid approaches, including ensemble methods that combine the determinism of fine-tuned models with the flexibility of large language models, may enhance robustness by utilizing the complementary strengths of both paradigms.

4.2 Crash manner of collision

The second research question centered on the ability of different models to accurately identify the manner of collision from textual crash descriptions, a task that requires reasoning beyond surface-level cues. Comparative analysis highlighted clear

differences between fine-tuned models of varying sizes and families and large-scale general-purpose LLMs.

4.2.1 Fine-tuned LLMs vs. large-scale baselines

Accuracy analysis results highlight the practicality of fine-tuning small-scale LLMs for manner of collision detection. Fine-tuning not only yields substantial gains in accuracy, like LLaMA3-1B and LLaMA3-3B to a performance level comparable to or exceeding much larger models such as GPT-4o, but also achieves this with modest training time and low inference cost. This proves that effective solutions can be obtained without using resource-intensive large-scale LLMs. Furthermore, the improvement observed when removing the *Unknown* category underscores the presence of noise in the dataset and suggests that fine-tuned models are often capable of resolving ambiguities overlooked in the original labels. Together, these findings emphasize the efficiency and robustness of fine-tuned small-sized models, offering a cost-effective alternative for large-scale deployment in safety-critical applications.

4.2.2 Consistency analysis

The consistency experiments further demonstrate the robustness of the fine-tuning approach. All models achieve very high self-consistency, confirming that their predictions remain stable across repeated runs. Cross-model consistency also remains strong, with most model pairs exceeding 0.90, indicating broad agreement across fine-tuned models. Importantly, overall consistency improves slightly when ground truth is excluded and increases further when the *Unknown* class is removed, suggesting that part of the observed disagreement originates from noisy or ambiguous labels. These findings highlight not only the stability of fine-tuned models but also their potential to surpass the quality of existing labels in noisy real-world datasets.

4.2.3 Generated data analysis

Distribution of reclassified Unknown MANCOLL cases demonstrates that LLMs can successfully classify crashes originally labeled as *Unknown* in the CISS dataset to be specific ones. Moreover, fine-tuned models often converged on a high consistent class assignment for cases labeled as *Unknown* in CISS, implying that these models may have recovered correct answers overlooked in the original annotation. Thus, fine-tuned LLMs can complement human annotation and address existing deficiencies in the dataset caused by limited knowledge.

4.3 Vehicle crash type

The third research question focused on the ability of different models to accurately classify the crash type from textual crash descriptions involving nearly one hundred categories. This task is considerably more challenging, as information about multiple vehicles is often intertwined within the same crash narrative, which can mislead LLMs in their judgment. Comparative analysis revealed that while large-scale general-purpose LLMs struggle with such complexity, fine-tuned models demonstrate improved robustness in disentangling vehicle-specific details and assigning the correct crash type.

4.3.1 Fine-tuned LLMs vs. large-scale baselines

The results on crash type classification show that fine-tuning is essential for this challenging task. Without adaptation, LLMs perform poorly, often near random accuracy, but parameter-efficient fine-tuning enables strong improvements across all settings. Notably, mid-sized fine-tuned models such as LLaMA3-3B, LLaMA3-8B, and Qwen2.5-7B reach around 80% accuracy, outperforming much larger-scale models like GPT-4o and LLaMA3-70B. This demonstrates that smaller models, when adapted to the traffic safety domain, can deliver competitive accuracy at a fraction of the training and inference cost.

Our ablation on LoRA training strategies further explains these gains. We find that adapting a larger set of attention projection matrices leads to higher self-consistency and better overall performance. This suggests that the task requires richer interactions between tokens than what query adaptation alone can provide. At the same time, the analysis of different vehicle settings highlights two-vehicle crashes as the most difficult scenario, since narratives mix information about both vehicles with distinct outcomes. In contrast, single-vehicle crashes are easier, and multi-vehicle chain collisions often share the same crash type, reducing ambiguity. These observations point to the need for fine-tuned models with enhanced reasoning and disambiguation abilities to handle the hardest cases.

4.3.2 Consistency analysis

Our consistency analysis shows that fine-tuned models achieve near-perfect run-to-run stability and strong cross-model agreement on single-vehicle crashes. The lower agreement with ground truth suggests that some labels may be noisy rather than model errors. In contrast, two-vehicle crashes introduce greater complexity: although self-consistency remains high, cross-model agreement drops noticeably. For two-vehicle crashes, models remain stable but agree less with one another, likely because descriptions of multiple vehicles make the task harder. This proves

that while fine-tuned models are reliable in simpler cases, multi-vehicle crashes are more challenging and may need better data or new modelling approaches.

4.3.3 Generated data analysis

CRASHTYPE distributions of CISS annotations vs. LLM predictions in the single-vehicle setting reveal that several mismatches are due to errors in the recorded ground truth, where some cases were incorrectly assigned. In these instances, LLM predictions were found to be correct, a fact further confirmed through manual inspection. Thus, fine-tuned LLMs can identify and correct errors in the existing labels of crash data (even though such errors are relatively rare in the original dataset).

In the two-vehicle setting, the comparison between CISS annotations and LLM predictions shows that fine-tuned LLMs preserve the relational characteristics of crash data without introducing distributional bias, underscoring their reliability for crash data annotation.

4.4 Limitations

This study has several limitations that impacted both its scope and findings. The most significant limitation is the time constraint, which forced us to focus on a limited number of research questions. As a result, the analysis focused on three key aspects of accident reconstruction: the direction of travel of the vehicles, the manner of collision and the type of accident. These are essential aspects for road safety, but many other factors present in accident descriptions, such as driver behaviour, environmental conditions,or the consequences of injuries were omitted, reducing the overall scope of the study.

Another limitation derives from the size of the manually annotated. The relatively small number of annotated cases may have affected the robustness of the models training and evaluation. The annotation process itself may introduce human bias.

Finally, the pipeline was only tested on a single English-language dataset, which limits its generalisability. It is unclear how well the approach can be transferred to other datasets written in different languages or following different reporting styles.

This work should be considered a first step. Future research should extend the scope by using larger annotated datasets, testing the approach in multilingual or cross-dataset contexts, comparing it with additional methods and evaluating its stability over time. These extensions would strengthen the reliability and applicability of the pipeline for large-scale road safety analysis.

Chapter 5

Conclusion

This thesis aimed to examine the ability of large language models in understanding and extracting structured information from crash narratives within the domain of traffic safety. We specifically aimed to see how well LLMs could answer domain-specific categorical questions about the direction of travel of a vehicle, the type of collision, and the vehicle crash type from unstructured text narratives.

Regarding the task of determining the vehicle's direction of travel, where essential information is frequently contained within the narrative, we demonstrated that small-size LLMs, when prompted by carefully designed prompting strategies, can achieve remarkable results. Specifically, techniques such as CoT reasoning allowed small models to achieve accuracy levels comparable to those of fine-tuned BERT-family transformers, showing that domain-specific fine-tuning is not necessarily required for every task.

Our findings demonstrate the advantages of parameter-efficient fine-tuning methods for identifying the manner of collision and crash type. Small-scale fine-tuned models consistently outperformed larger general-purpose LLMs, achieving accuracy levels comparable to or even exceeding models such as GPT-4o. The consistency study validated their reliability and demonstrated that fine-tuned models can exceed the quality of existing labels, effectively addressing ambiguities in the original annotations.

The classification task for crash types represented the most challenging scenario and highlighted the clear need for domain adaptation. In the absence of fine-tuning, large language models exhibited suboptimal performance on this complex task. Parameter-efficient fine-tuning provided significant enhancements, allowing models to achieve around 80% accuracy while preserving computational efficiency.

This thesis demonstrates that open-source LLMs, when combined with effective prompting and parameter-efficient fine-tuning, can consistently extract both explicit and implicit crash aspects from unstructured narratives, demonstrating that these methodologies may be applied to extensive datasets where labeled data is missing,

enhance human annotations, and eventually facilitate progress in traffic safety research.

Chapter 6

Future work

Generalization capability analysis This study focused on extracting travel direction, manner of collision, and crash type from crash narratives in the CISS dataset. However, other attributes, such as impact points, roadway characteristics, and driver behavior, are also crucial for understanding collision patterns and improving traffic safety. In future work, our approach can be extended to cover these additional dimensions of crash. In addition, while our experiments were conducted on the CISS dataset, the proposed method can be generalized to other traffic accident datasets from different countries, as well as alternative sources such as tweets on social media platforms. Moreover, the current method still requires a certain amount of annotated data for fine-tuning, which raises the question of how well the model can generalize beyond the training domain. A particularly important direction is to investigate the performance of models trained on one dataset and applied to another, in order to better understand their robustness and cross-dataset transferability.

Causal sequence analysis of crash narratives In this study, we assumed that the description of a crash follows a time order. In reality, multi-vehicle collisions are often triggered by a minor initial crash, which subsequently leads to a chain of collisions, and the technicians need to include all of these crash segments into the report. However, when they compile accident reports based on collected information, the narrative style may vary. For instance, some may emphasize what they consider the most important aspect of the accident at the beginning, and only afterwards describe other details. Such variations in narrative structure can mislead LLMs when identifying the first harmful event of the accident, particularly for smaller-scale models with limited reasoning ability. We argue that enhancing LLMs' capability to analyze the causality is crucial not only for this project but also for advancing research on LLMs more broadly.

Reasoning-oriented LLMs Another promising direction is to explore the use of reasoning-oriented LLMs. In our experiments, we employed instruction-tuned

models and prompted them to directly output results. Although we used CoT prompting, the models did not explicitly generate intermediate reasoning steps, nor did we adopt reasoning-specific models. But prior studies have shown that encouraging LLMs to articulate their reasoning process can significantly improve prediction accuracy. In addition, during fine-tuning, it is possible to introduce guidance tokens that help the model learn to structure its reasoning. However, these approaches may also increase the computational cost, which presents a trade-off between performance and efficiency that warrants further investigation.

Author's note on the use of AI tools

In the preparation of this thesis project, Large Language Models (LLMs) such as ChatGPT and Claude were used as supportive tools. The English text was originally drafted in either Italian or informal English and subsequently revised and refined with the assistance of these models to achieve a more formal academic style.

LLMs were also employed during the development phase of the project code. Their contribution included improving the clarity and readability of plots, suggesting code optimizations and assisting in code refactoring to ensure efficiency and consistency.

All conceptual contributions, experimental design and analysis remain the responsibility of the author. The use of LLMs was limited to language refinement and technical assistance, without replacing the author's critical reasoning or original contributions.

Bibliography

[1] World Health Organization. *Global Status Report on Road Safety 2023*. ISBN 978-92-4-008651-7 (electronic version). Geneva, 2023. URL: https://www.who.int/publications/i/item/9789240086517 (cit. on p. 1).

[2] National Highway Traffic Safety Administration. *Crash Investigation Sampling System (CISS)*. Accessed: 2025-07-18. 2023. URL: https://www.nhtsa.gov/crash-data-systems/crash-investigation-sampling-system (cit. on pp. 2, 8).

[3] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. «Improving language understanding by generative pre-training». In: (2018) (cit. on pp. 2, 3).

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186 (cit. on pp. 2, 3, 21, 35, 38).

[5] Long Ouyang et al. «Training language models to follow instructions with human feedback». In: *Advances in neural information processing systems* 35 (2022), pp. 27730–27744 (cit. on pp. 2, 20).

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. «Distilling the knowledge in a neural network». In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on pp. 2, 3).

[7] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165 (cit. on pp. 2, 23, 25, 81).

[8] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. «Language models are unsupervised multitask learners». In: *OpenAI blog* 1.8 (2019), p. 9 (cit. on pp. 3, 23).

[9] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023. arXiv: 1910.10683 [cs.LG]. URL: https://arxiv.org/abs/1910.10683 (cit. on p. 3).

[10] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. *Finetuned Language Models Are Zero-Shot Learners*. 2022. arXiv: 2109.01652 [cs.CL]. URL: https://arxiv.org/abs/2109.01652 (cit. on p. 3).

[11] Archit Parnami and Minwoo Lee. *Learning from Few Examples: A Summary of Approaches to Few-Shot Learning*. 2022. arXiv: 2203.04291 [cs.LG]. URL: https://arxiv.org/abs/2203.04291 (cit. on p. 3).

[12] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. *Deep reinforcement learning from human preferences*. 2023. arXiv: 1706.03741 [stat.ML]. URL: https://arxiv.org/abs/1706.03741 (cit. on p. 3).

[13] Kun Xie, Di Yang, Kaan Ozbay, and Hong Yang. «Use of real-world connected vehicle data in identifying high-risk locations based on a new surrogate safety measure». In: *Accident Analysis & Prevention* 125 (2019), pp. 311–319 (cit. on p. 3).

[14] Yang Zhao, Pu Wang, Yibo Zhao, Hongru Du, and Hao Frank Yang. «Towards Reliable and Interpretable Traffic Crash Pattern Prediction and Safety Interventions Using Customized Large Language Models». In: *arXiv preprint arXiv:2505.12545* (2025) (cit. on p. 3).

[15] Hao Zhen, Yucheng Shi, Yongcan Huang, Jidong J Yang, and Ninghao Liu. «Leveraging large language models with chain-of-thought and prompt engineering for traffic crash severity analysis and inference». In: *Computers* 13.9 (2024), p. 232 (cit. on pp. 3, 4).

[16] Zihao Li, Chaolun Ma, Yang Zhou, Dominique Lord, and Yunlong Zhang. «Leveraging Textual Description and Structured Data for Estimating Crash Risks of Traffic Violation: A Multimodal Learning Approach». In: *IEEE Transactions on Intelligent Transportation Systems* (2025) (cit. on pp. 3, 4).

[17] Hao Zhen and Jidong J Yang. «CrashSage: A Large Language Model-Centered Framework for Contextual and Interpretable Traffic Crash Analysis». In: *arXiv preprint arXiv:2505.07853* (2025) (cit. on pp. 3, 4, 6).

[18] Younghoon Seo, Jihyeok Park, Gyungtaek Oh, Hyungjoo Kim, Jia Hu, and Jaehyun So. «Text classification modeling approach on imbalanced-unstructured traffic accident descriptions data». In: *IEEE Open Journal of Intelligent Transportation Systems* 4 (2023), pp. 955–965 (cit. on p. 4).

[19] Maroa Mumtarin, Md Samiullah Chowdhury, and Jonathan Wood. «Large Language Models in Analyzing Crash Narratives–A Comparative Study of ChatGPT, BARD and GPT-4». In: *arXiv preprint arXiv:2308.13563* (2023) (cit. on pp. 4, 6, 7).

[20] Cristian Arteaga and JeeWoong Park. «A large language model framework to uncover underreporting in traffic crashes». In: *Journal of Safety Research* 92 (2025), pp. 1–13 (cit. on pp. 4, 6).

[21] Reza Golshan Khavas and Mohammadreza Nosrati. «Extracting Traffic Crash Information from Social Media: An Llm-Based Approach». In: *Available at SSRN 5379743* (2025) (cit. on p. 4).

[22] Shadi Jaradat, Richi Nayak, Alexander Paz, Huthaifa I Ashqar, and Mohammad Elhenawy. «Multitask learning for crash analysis: A Fine-Tuned LLM framework using twitter data». In: *Smart Cities* 7.5 (2024), pp. 2422–2465 (cit. on p. 4).

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. «Lora: Low-rank adaptation of large language models.» In: *ICLR* 1.2 (2022), p. 3 (cit. on pp. 4, 7, 42, 50).

[24] Md Abu Sayed, Rohit Kate, and D M Anisuzzaman. *Text Mining Crash Narrative Draft Final Report*. Technical Report. `https://uwm.edu/ipit/wp-content/uploads/sites/570/2023/05/Text-Mining-Crash-Narrative-Draft-Final-Report-11-2021.pdf`. Milwaukee, WI, USA: University of Wisconsin-Milwaukee, Institute for Physical Infrastructure and Transportation (IPIT), 2021 (cit. on p. 6).

[25] Amir Hossein Oliaee, Subasish Das, Jinli Liu, and M Ashifur Rahman. «Using Bidirectional Encoder Representations from Transformers (BERT) to classify traffic crash severity types». In: *Natural language processing journal* 3 (2023), p. 100007 (cit. on p. 6).

[26] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: `2201.11903 [cs.CL]`. URL: `https://arxiv.org/abs/2201.11903` (cit. on pp. 7, 23, 26, 79).

[27] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. arXiv: `2203.11171 [cs.CL]`. URL: `https://arxiv.org/abs/2203.11171` (cit. on pp. 7, 23, 27).

[28] Hao Gong, Guangnan Zhang, Zhixiong Guo, Yu Sun, Haibo Huang, and Zhen Ling. «Two-vehicle driver-injury severity: A multivariate random parameters logit approach». In: *Analytic Methods in Accident Research* 33 (2022), p. 100190. DOI: `10.1016/j.amar.2021.100190` (cit. on p. 9).

[29] National Highway Traffic Safety Administration. *Traffic Safety Facts 2022 Data*. Tech. rep. U.S. Department of Transportation, 2023. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813667 (cit. on p. 9).

[30] Sudesh Bhagat, Ibne Farabi Shihab, and Jonathan Wood. *Identification of Potentially Misclassified Crash Narratives using Machine Learning (ML) and Deep Learning (DL)*. 2025. arXiv: 2507.03066 [cs.CL]. URL: https://arxiv.org/abs/2507.03066 (cit. on p. 10).

[31] Sudesh Ramesh Bhagat, Ibne Farabi Shihab, and Anuj Sharma. *Accuracy is Not Agreement: Expert-Aligned Evaluation of Crash Narrative Classification Models*. 2025. arXiv: 2504.13068 [cs.CL]. URL: https://arxiv.org/abs/2504.13068 (cit. on pp. 10, 15).

[32] Abdoulaye E. Kitali and Srinivas S. Pulugurtha. «Exploring the Need to Model Two- and Multiple-Vehicle Crashes Separately». In: *Transportation Research Record* (2022). DOI: 10.1177/03611981211037882 (cit. on p. 10).

[33] S. Wang et al. «Improving Crash Data Quality with Large Language Models». In: *arXiv preprint arXiv:2508.04399* (2025) (cit. on pp. 11, 14).

[34] K Digges and G Bahouth. «Frequency of injuries in multiple impact crashes». In: *Annual Proceedings/Association for the Advancement of Automotive Medicine*. Vol. 47. 2003, p. 417 (cit. on p. 11).

[35] Qinghua Zeng et al. «Using natural language processing for extracting crash information». In: *Transportation Research Record* 2591 (2016), pp. 116–125 (cit. on p. 14).

[36] Jacob Cohen. «A Coefficient of Agreement for Nominal Scales». In: *Educational and Psychological Measurement* 20.1 (1960), pp. 37–46. DOI: 10.1177/001316446002000104. URL: https://doi.org/10.1177/001316446002000104 (cit. on p. 16).

[37] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Hugo Larochelle. *Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program)*. 2020. arXiv: 2003.12206 [cs.LG]. URL: https://arxiv.org/abs/2003.12206 (cit. on p. 17).

[38] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971 (cit. on p. 17).

[39] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2025. arXiv: 2303.18223 [cs.CL]. URL: https://arxiv.org/abs/2303.18223 (cit. on p. 17).

[40] Albert Q. Jiang et al. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825 (cit. on pp. 17, 66).

[41] DeepSeek-AI. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948 (cit. on pp. 17, 18).

[42] Qwen Team. *Qwen3 Technical Report*. 2025. arXiv: 2505.09388 [cs.CL]. URL: https://arxiv.org/abs/2505.09388 (cit. on pp. 17, 18).

[43] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL]. URL: https://arxiv.org/abs/1904.09751 (cit. on p. 20).

[44] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692 (cit. on pp. 21, 35).

[45] Iz Beltagy, Kyle Lo, and Arman Cohan. *SciBERT: A Pretrained Language Model for Scientific Text*. 2019. arXiv: 1903.10676 [cs.CL]. URL: https://arxiv.org/abs/1903.10676 (cit. on pp. 21, 22, 35).

[46] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. «A systematic survey of prompt engineering in large language models: Techniques and applications». In: *arXiv preprint arXiv:2402.07927* (2024) (cit. on pp. 22, 23, 26, 27, 29).

[47] Qingxiu Dong et al. *A Survey on In-context Learning*. 2024. arXiv: 2301.00234 [cs.CL]. URL: https://arxiv.org/abs/2301.00234 (cit. on p. 25).

[48] Wes McKinney. «Data Structures for Statistical Computing in Python». In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a (cit. on p. 37).

[49] Hugging Face. *Hugging Face: The AI community building the future*. https://huggingface.com. 2025 (cit. on p. 37).

[50] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG]. URL: https://arxiv.org/abs/1912.01703 (cit. on p. 37).

[51] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771 (cit. on p. 37).

[52] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101 (cit. on p. 37).

[53] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. *A Primer in BERTology: What we know about how BERT works*. 2020. arXiv: 2002.12327 [cs.CL]. URL: https://arxiv.org/abs/2002.12327 (cit. on p. 38).

[54] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG]. URL: https://arxiv.org/abs/2010.16061 (cit. on p. 41).

[55] Marina Sokolova and Guy Lapalme. «A systematic analysis of performance measures for classification tasks». In: *Information processing & management* 45.4 (2009), pp. 427–437 (cit. on p. 41).

[56] Sarthak Harne, Arvind Agarwal, et al. «LLM driven Text-to-Table Generation through Sub-Tasks Guidance and Iterative Refinement». In: *arXiv preprint arXiv:2508.08653* (2025) (cit. on p. 42).

[57] Wenlu Du, Ankan Dash, Jing Li, Hua Wei, and Guiling Wang. «Safety in traffic management systems: A comprehensive survey». In: *Designs* 7.4 (2023), p. 100 (cit. on p. 42).

[58] National Highway Traffic Safety Administration et al. «NHTSA Field Crash Investigation 2023 Coding and Editing Manual». In: *Publication DOT HS* 813 (2025), p. 614 (cit. on p. 42).

[59] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. «Chain-of-thought prompting elicits reasoning in large language models». In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837 (cit. on p. 45).

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 49).

[61] Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. «A survey on lora of large language models». In: *Frontiers of Computer Science* 19.7 (2025), p. 197605 (cit. on p. 50).

[62] Zhilu Zhang and Mert Sabuncu. «Generalized cross entropy loss for training deep neural networks with noisy labels». In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 50).

[63] Quinn McNemar. «Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages». In: *Psychometrika* 12.2 (1947), pp. 153–157. DOI: 10.1007/BF02295996 (cit. on p. 62).

[64] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. «Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing». In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: 10.1145/3560815. URL: https://doi.org/10.1145/3560815 (cit. on pp. 62, 80).

[65] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783 (cit. on p. 66).

[66] Jinze Bai et al. *Qwen Technical Report*. 2023. arXiv: 2309.16609 [cs.CL]. URL: https://arxiv.org/abs/2309.16609 (cit. on p. 66).

[67] OpenAI et al. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: https://arxiv.org/abs/2410.21276 (cit. on p. 66).

[68] Sebastian Raschka. «Practical tips for finetuning llms using lora (low-rank adaptation)». In: *Ahead of AI (Nov. 2023). url: https://sebastianraschka. substack. com/p/practical-tips-for-finetuning-llms* (2024) (cit. on p. 67).

[69] Stephan Arndt, Carolyn Turvey, and Nancy C Andreasen. «Correlating and predicting psychiatric symptom ratings: Spearmans r versus Kendalls tau correlation». In: *Journal of psychiatric research* 33.2 (1999), pp. 97–104 (cit. on p. 77).

[70] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. *Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?* 2022. arXiv: 2202.12837 [cs.CL]. URL: https://arxiv.org/abs/2202.12837 (cit. on p. 79).

[71] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. *What Makes Good In-Context Examples for GPT-3?* 2021. arXiv: 2101.06804 [cs.CL]. URL: https://arxiv.org/abs/2101.06804 (cit. on p. 79).

[72] XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. «Pre-trained models for natural language processing: A survey». In: *Science China Technological Sciences* 63.10 (Sept. 2020), pp. 1872–1897. ISSN: 1869-1900. DOI: 10.1007/s11431-020-1647-3. URL: http://dx.doi.org/10.1007/s11431-020-1647-3 (cit. on p. 81).

[73] Dong-Hyun Lee. «Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks». In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013) (cit. on p. 81).

[74] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009. URL: http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf (cit. on p. 81).