UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEGREE PROGRAM OF COMPUTER SCIENCE

CrimeLens Chat: An LLM-Powered Natural-Language Interface for Local Crime Analysis

RESEARCH WORK

In partial fulfillment for the degree of Bachelor in Computer Science

AUTHOR(S)

Jeremy Jeffrey Matos Cangalaya

Marcos Daniel Ayala Pineda

ADVISOR(S)

Cristian Lopez

Germain Garcia

Lima - Perú

2025

Contents

Pág.

ABSTRACT 1

CHAPTER I CONTEXT AND MOTIVATION **2**

1.1 Description of the Problematic Situation 3

1.2 Justification 4

1.3 Research Objectives 4

1.3.1 General Objectives 4

1.3.2 Specific Objectives 5

CHAPTER II THEORETICAL FRAMEWORK **6**

2.1 Spatio-Temporal Data 6

2.2 Large Language Models (LLMs) 6

2.2.1 LLM decoding methods 8

2.2.2 Supervised Fine-Tuning 9

2.2.3 LoRA (Low-Rank Adaptation) 9

2.2.4 Tool Integrated Reasoning 10

2.3 Prompting 11

2.3.1 Prompt Engineering 12

2.4 Metrics 13

2.4.1 Pass@K 13

2.4.2 Passˆk 13

2.4.3 Code-BLEU 14

2.5 Final considerations 15

CHAPTER III STATE OF THE ART **16**

3.1 Large-Language Models for Urban Scenarios 16

3.2 Large-Language Models for Coding-Tasks 18

3.3 Open Crime Datasets 19

3.4 Crime-Data Visualization Tools 20

3.5 Final considerations 21

CHAPTER IV METHODOLOGY **22**

4.1 Data preprocessing 22

4.1.1 Data sources 23

4.1.2 Data sources cleaning and normalization 23

4.2 Question-Answer dataset generation 24

4.2.1 Question generation 24

4.2.2 Code-answer generation 25

4.2.3 Dataset curation and splitting 26

4.3 Training and evaluation 26

4.4 Training evaluation 27

4.4.1 Training prompt structure 27

4.4.2 Model Fine-Tuning 28

4.4.3 Model Evaluation 29

4.5 Inference Workflow 29

4.6 Summary 30

CHAPTER V EXPERIMENTS AND RESULTS **31**

5.1 Hyperparameters 31

5.2 Quantitative Results 32

5.3 Qualitative Analysis 33

5.3.1 Case: Counterfactual Evaluation of Targeted Interventions in Crime Hotspots of Guangdong Province 33

5.4 Final considerations 40

CONCLUSIONS AND FUTURE WORK 42

5.5 Conclusions 42
5.6 Limitations and Future Work 42
APPENDIX 55
A.1 Dataset specification 55
A.2 Prompts 57

ABSTRACT

Large Language Models (LLMs) have demonstrated exceptional versatility across diverse domains, yet their application in crime analytics remains underexplored due to a lack of domain-specific datasets and specialized methodologies. To address this gap, we introduce ChinaCrimesQACode, a novel dataset designed to capture the intricacies of spatio-temporal crime analysis, including detailed crime incidents, street networks, and administrative boundaries with location-specific queries. Leveraging ChinaCrimesQACode, we focus on code generation from natural language questions, enabling LLMs to produce high-quality, executable Python code for geospatial crime data analysis from structured conversational inputs. Fine-tuned model and baseline models were evaluated using semantic equivalence metrics and LLM-as-a-judge. Our results demonstrate substantial improvements in generating contextually accurate analytical code, highlighting the transformative potential of tailored datasets and fine-tuning methodologies in optimizing crime intelligence workflows. This work highlights the potential of LLMs in crime analytics workflows and the essential role of domain-specific datasets in tailoring them to specialized analytical challenges.

Keywords:

Large Language Models; Crime Analytics; Geospatial Analysis; Code Generation; Conversational AI

CHAPTER I

CONTEXT AND MOTIVATION

Urban crime persists as a critical social challenge, with the *Global Organized Crime Index 2023* reporting that 83 % of the world's population now lives in countries exhibiting high levels of criminality [Global Initiative Against Transnational Organized Crime, 2023]. Such criminality is rarely uniform within a city; instead, incidents concentrate at specific micro-places and times, creating dynamic "hot" and "cool" spots that evolve over days, weeks, and seasons [García Zanabria et al., 2022]. Yet many jurisdictions still struggle with incomplete or delayed reporting, obscuring timely understanding of local risk patterns [NSSF, 2023]. Consequently, both authorities and citizens require tools that transform raw spatio-temporal crime data into actionable insights capable of guiding preventive decisions in real time.

Various visual-analytics systems have been developed to address this need [Salah and Xia, 2022; Silva et al., 2017; Zanabria et al., 2020; Garcia et al., 2021]. However, these tools often overwhelm non-experts with complex interfaces and limited guidance. Natural-language interfaces (NLIs) have emerged as a solution [Setlur et al., 2016; Narechania et al., 2021; Luo et al., 2022; Liu et al., 2021; Sah et al., 2024], implementing pipelines that translate user queries into formal representations. Yet most NLIs still struggle with ambiguity, underspecification, and complex analytical reasoning due to their reliance on template grammars or traditional semantic parsing.

Adopting an LLM-driven chat interface for spatio-temporal crime data directly addresses this gap. First, natural dialogue eliminates steep learning curves by allowing users to express information needs e.g., "Which streets saw the largest increase in robberies last weekend in this city?" without mastering dashboard controls. Second, the model can highlight relevant data used to answer questions, enabling users to validate insights and build trust in the system. This integration of conversational AI with crime intelligence seeks to democratize access to crime data, bolster data-driven policing, and ultimately enhance public safety for both residents and visitors.

1.1 Description of the Problematic Situation

In many major cities, crime is a growing concern, especially in popular tourist areas, where opportunities attract offenders targeting visitors [Tevdoradze et al., 2024; de Barros Tomé Machado, 2012]. As a result, travelers often feel discouraged from exploring freely, affecting tourism perception and local economies. But the consequences extend well beyond tourists. Residents, though familiar with their surroundings, depend on personal experience, local news, and neighborhood conversations to decide which streets or districts to avoid. This informal awareness helps them steer clear of perceived danger zones, yet it offers limited protection, meaning even locals remain vulnerable to crime within their own cities.

In response to these concerns, several visualization tools [García Zanabria et al., 2022; Salah and Xia, 2022; Silva et al., 2017; Zanabria et al., 2020; Garcia et al., 2021] have emerged to increase awareness, validate hypotheses, and support safer decision-making. However, these solutions are often difficult to use for non-expert users. They usually provide sophisticated analysis using robust feature engineering processes, restricting their accessibility to a broader audience. As a result, both citizens and authorities are left with insufficient support to anticipate risks or act proactively.

A key challenge in crime prevention is the lack of accessible tools that help both citizens and authorities interpret complex crime data. While large volumes of data are publicly available [Zhang et al., 2025b; New York City Police Department, 2025; Chicago Police Department, 2024], they're often difficult to analyze without technical skills or time [Wang and Zhang, 2020]. Crime patterns constantly shift across time and space. Some areas may be riskier at night, others during weekends or certain seasons. Without tools that clearly communicate these dynamics in real time, people struggle to make safe decisions, and authorities face obstacles in allocating resources effectively.

1.2 Justification

Large language models (LLMs) offer a human-centred interface for interacting with complex data [Yang et al., 2024; Pappula and Allam, 2023]. Unlike, web dashboards, GIS (Geographic Information System) software or specialized tools, LLM-powered chatbot lets users ask questions like "Which streets saw the largest surge in robberies in the last month?" and instantly receive concise explanations. In addition, open-source models alleviate many of the privacy concerns [Temsah et al., 2025; Ersöz et al., 2025] often associated with proprietary systems and can be more cost-effective [Liu et al., 2024].

Adopting an LLM-based conversational layer directly addresses the two principal weaknesses of current crime-analytics platforms: usability and applicability. First, a chat interface removes the steep learning curves of existing tools. Second, by supporting rapid, iterative exploration of temporal windows, geographic areas and crime types, it delivers immediate insights when users need them.

In this study, we aim to enhance crime-prevention efforts and raise awareness about the risks associated with routes used by tourists and residents by developing a chat interface. This interface will allow users to interact with crime data, presenting textual summaries, to facilitate informed decision-making. Additionally, the work establishes a foundation for future investigations into the application of LLMs to more sophisticated crime-analysis tasks.

1.3 Research Objectives

1.3.1 General Objectives

Develop an chat-based tool that leverages LLMs to assist in crime prevention efforts by enabling user-data interaction by natural language queries.

1.3.2 Specific Objectives

- Data recollection from publicly available crime datasets, and pre-processing to create a structured dataset suitable for LLM training and evaluation.
- Design and implement a chat-based interface that allows users to query crime data and receive textual responses, based on code execution.
- Create a question-answering dataset specifically tailored for crime data.
- Address privacy and security concerns by focusing on the use of open-source LLMs as alternative solutions to proprietary models.

CHAPTER II

THEORETICAL FRAMEWORK

This chapter lays the groundwork for the theories that are necessary to understand the research that is presented in this work. We look at six important areas that support our approach: (1) the basics of spatio-temporal data, (2) LLM fundamentals, (3) LLM adaptation and fine-tuning, (4) prompting techniques, (5) tool-integrated reasoning, and (6) metrics for evaluating code execution. These concepts form the conceptual basis for understanding the methodological proposal and technical implementations that will be shown in the rest of this work.

2.1 Spatio-Temporal Data

The spatio-temporal data is a type of data that contains information about the spatial and temporal dimensions of an event or phenomenon. This dual indexing enables the representation of complex relationships between spatial and temporal elements, allowing for a more comprehensive understanding of the data.

The spatio-temporal data can be formalized as a tuple $ST = \{(s_i, t_j, X_{ij} | s_i \in S, t_j \in T, X_{ij} \in \mathbb{R}^d)\}$, where S is the spatial domain, T is the temporal domain, and X_{ij} signifies the observed attributes at location s_i and time t_j.

2.2 Large Language Models (LLMs)

Large Language Models (LLMs) are built upon the transformer architecture, originally introduced by Vaswani et al. [2023]. This architecture revolutionized natural language processing (NLP) by replacing recurrent neural networks (RNNs) with self-attention mechanisms, enabling models to process entire sequences in parallel rather than sequentially.

The core components of the transformer architecture include:

- **Self-attention mechanisms**: Allow the model to assess the importance of different words within a given context. By computing weighted sums of all positions in a sequence, with weights determined by query-key interactions, the model can focus on relevant information while ignoring irrelevant parts.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q is the query matrix, K is the key matrix, V is the value matrix, and d_k is the dimension of the keys.

- **Multi-head attention**: Enables the model to attend to information from multiple representation subspaces simultaneously.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$ and W_i^Q, W_i^K, W_i^V are learned projection matrices.

- **Feed-forward neural networks**: Apply non-linear transformations to the attention outputs.

- **Layer normalization**: Helps stabilize and accelerate the training process.

$$LayerNorm(x) = \omega \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

where μ and σ^2 are the mean and variance of the input, ϵ is a small constant for numerical stability, and ω and β are learnable parameters.

- **Positional encoding**: Introduces information about the order of tokens in a sequence. It is tipically encoded using sine and cosine functions of different frequencies.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

where pos is the position and i is the dimension index.

Notable examples of LLMs include OpenAI's GPT-3 [Brown et al., 2020], Google's BERT [Devlin et al., 2018] and T5 [Raffel et al., 2023], and Meta's RoBERTa [Liu et al., 2019b]. These models have set new benchmarks across a variety of NLP tasks such as text classification, machine translation, and summarization. While all are built on the transformer architecture, they differ in their design choices and training objectives. For instance, BERT [Devlin et al., 2018], an encoder-only transformer, uses a masked language modeling objective, where random tokens in a sentence are masked and the model learns to predict them using surrounding context. In contrast, GPT-3 [Brown et al., 2020], a decoder-only transformer, follows an autoregressive training strategy, generating one token at a time based on previously generated tokens.

The theoretical foundation of LLMs lies in probabilistic language modeling. Given a sequence of tokens $x = (x_1, x_2, \ldots, x_n)$, a language model aims to estimate either the joint probability $P(x)$ or the conditional probability of the next token $P(x_n \mid x_1, \ldots, x_{n-1})$. The model is trained to assign higher probabilities to sequences that are more likely to appear in natural language, based on patterns learned from large-scale datasets. This is made possible by the transformer's use of self-attention and feed-forward layers, which together capture complex dependencies across tokens. In particular, positional encodings and multi-head attention enable LLMs to model long-range relationships, something that earlier architectures like RNNs and LSTMs struggled to achieve effectively.

2.2.1 LLM decoding methods

Once a language model has assigned likelihoods to each possible next token, it must still choose a single path forward, a step handled by a *decoding strategy* [Shi et al., 2024]. In practice, three classical strategies are widely used:

- **Greedy strategy**: at every turn the decoder picks the candidate with the highest score. This makes generation lightning-fast and fully repeatable, but the text can feel mechanical or repetitive.
- **Beam search**: keeps the k best partial sequences (beams) at each step and expands them in parallel, usually yielding more coherent results than greedy search at the cost of extra computation.
- **Probabilistic (multinomial) sampling**: the next token is drawn at random in proportion to its probability, often after applying temperature scaling, top-k, or top-p trimming, to inject freshness and variety into the output [Qin et al., 2025].

2.2.2 Supervised Fine-Tuning

Supervised fine-tuning (SFT) of large language models (LLMs) is a technique used to adapt pre-trained models on large corpora to specific tasks, improving their performance and alignment with human preferences or task requirements [Wang et al., 2023]. This process involves training the model on a labeled dataset, where each input is paired with a corresponding output.

A common variant of SFT is instruction tuning, which trains the model on supervised datasets containing (instruction, output) pairs [Wei et al., 2022]. This method improves the model's zero-shot performance by shifting its behavior from next-token prediction to better understanding and following human directions [Zhang et al., 2024].

2.2.3 LoRA (Low-Rank Adaptation)

LoRA, introduced by Hu et al. [2021], is a parameter-eficient fine-tuning method designed to efficiently adapt large language models (LLMs) by minimizing computational overhead. Instead of updating all model parameters, which can number in the hundreds of billions for modern LLMs, LoRA introduces trainable low-rank matrices into existing weight layers, such as attention or feed-forward layers. These matrices encode task-specific adjustments while keeping the pre-trained model weights unchanged, significantly reducing the number of parameters that need to be trained.

Mathematically, LoRA approximates the weight update ΔW using a low-rank decomposition $\Delta W = AB$, where $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$, with $r \ll \min(d, k)$. This approach greatly lowers memory and computational requirements, making it particularly suitable for fine-tuning on devices with limited resources.

LoRA is highly modular and can be seamlessly integrated into transformer-based architectures without requiring structural modifications. It can also be combined with other optimization techniques, such as quantization or pruning, to further enhance efficiency. Studies have demonstrated that LoRA achieves performance comparable to or better than full fine-tuning across various tasks, while modifying less than 1.

Extensions of LoRA, such as QLoRA (Quantized LoRA) [Dettmers et al., 2023] and AdaLoRA (Adaptive LoRA) [Zhang et al., 2023], have further refined its capabilities. For instance, QLoRA enables fine-tuning on models quantized to 4 bits, facilitating efficient deployment on consumer-grade hardware.

Thanks to its efficiency and flexibility, LoRA has become a popular choice in open-source LLM projects, especially for scenarios with limited computational resources or strict privacy requirements.

2.2.4 Tool Integrated Reasoning

Tool Integrated Reasoning (TIR) represents a paradigm shift in artificial intelligence systems designed for complex mathematical problem-solving. This approach addresses the inherent limitations of traditional large language models in mathematical reasoning by seamlessly combining natural language processing with external computational tools. ToRA (Tool-integrated Reasoning Agents) introduced by [Gou et al., 2024], which demonstrates how agents can leverage both analytical reasoning capabilities and specialized computational resources such as symbolic solvers and programming libraries. This integration overcomes the computational bottlenecks that have historically constrained language models when tackling advanced mathematical problems, enabling more robust and accurate problem-solving capabilities.

The effectiveness of this approach relies heavily on sophisticated training methodologies that incorporate multi-faceted data augmentation strategies. MuMath-Code [Yin et al., 2024], which combines tool-enabled large language models with multi-perspective data augmentation techniques specifically for mathematical reasoning tasks. Their two-stage training approach demonstrates that integrating external Python interpreters with LLMs significantly enhances mathematical reasoning capabilities in open-source models. This methodology bridges two traditionally separate research directions in mathematical reasoning, showing that the synergy between tool utilization and data augmentation produces superior results compared to isolated approaches.

The advancement in this topic has been facilitated by the development of large-scale specialized datasets and computational resources. NuminaMath [Li et al., 2024b], the largest public dataset in AI4Math containing 860,000 competition-level mathematical problem-solution pairs, which won the first AIMO Progress Prize. Building on such resources, [Moshkov et al., 2025] developed state-of-the-art mathematical reasoning models using the OpenMathReasoning dataset for their AIMO-2 winning solution. Additionally, OpenCodeReasoning [Ahmad et al., 2025] contributed to advancing data distillation techniques for competitive programming, demonstrating the expansion of TIR applications beyond traditional mathematics into algorithmic problem-solving domains.

2.3 Prompting

Basically prompting is a technique used to guide the behavior of LLMs by providing them with specific instructions or context. This can be done through various methods, such as zero-shot prompting, few-shot prompting, and chain-of-thought prompting. Each method has its own advantages and disadvantages, depending on the task at hand and the desired outcome.

2.3.1 Prompt Engineering

Prompt engineering is an emerging field focused on crafting and refining prompts to make the most effective use of language models (LMs) across diverse applications and research areas. It plays a crucial role in enhancing our understanding of the strengths and limitations of LLMs. Beyond simply writing prompts, prompt engineering involves a broad set of techniques essential for building with, interacting with, and expanding the capabilities of LLMs. It also contributes to improving model safety and enables the integration of specialized knowledge and functionalities into LLM-based systems

The following methods are some of the most common techniques and used in our work:

- **Zero-shot prompting**: Involves providing the model with a task description or question without any examples. The model is expected to generate a response based solely on its pre-existing knowledge and understanding of the task.
- **Few-shot prompting**: As mentioned by Brown et al. [2020], this technique involves providing the model with a few examples of the desired output format or task. This helps the model understand the context and generate more accurate responses. A related approach is "one-shot prompting", in which the model is given only a single example.
- **Chain-of-thought prompting**: Firstly introduced by Wei et al. [2023], encourages the model to generate intermediate reasoning steps before arriving at a final answer. This approach has been shown to improve performance on complex tasks, such as arithmetic reasoning and logical inference.

2.4 Metrics

This section presents the evaluation metrics used to assess model performance in the context of code generation tasks. Each metric offers a different perspective on how accurately and effectively our models respond to user queries about crime data.

2.4.1 Pass@K

Pass@k is a metric that evaluates the probability of a model generating a correct answer within the top-k responses. It is particularly useful for assessing the performance of language models in tasks where multiple answers can be generated, such as question answering or text generation [Levi, 2024]. The metric is defined as:

$$Pass@k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i \in \hat{y}_i^{(1:k)}) \tag{II.1}$$

where N is the number of samples, y_i is the ground truth answer for sample i, and $\hat{y}_i^{(1:k)}$ is the set of the top-k generated answers for sample i. The indicator function $\mathbb{I}$ returns 1 if the ground truth answer is in the top-k responses, and 0 otherwise.

2.4.2 Passˆk

Passˆk is a metric designed to evaluate the likelihood of a model generating a correct answer across all k responses [Yao et al., 2024]. This metric is particularly useful for assessing the consistency and reliability of model performance in scenarios where multiple outputs are generated.

$$Pass^K = (\frac{c}{n})^K \tag{II.2}$$

where c is the number of correct answers in the top-k responses, and n is the total number of responses generated for each sample. This formulation allows for a more nuanced evaluation of model performance, particularly in cases where the distribution of correct answers may vary across different samples.

2.4.3 Code-BLEU

CodeBLEU is a specialized metric designed to evaluate the quality of code generation by comparing machine-generated code against one or more reference implementations. It extends the traditional BLEU metric [Papineni et al., 2002], commonly used in natural language processing for tasks such as machine translation, by incorporating programming-specific characteristics such as syntax, structure, and semantic correctness [Ren et al., 2020]. This makes CodeBLEU better aligned with the nuanced requirements of programming languages.

The metric is formulated as a weighted linear combination of four complementary components:

$$\text{CodeBLEU} = \alpha \cdot \text{BLEU} + \beta \cdot \text{BLEU}_{\text{weight}} + \gamma \cdot \text{Match}_{\text{AST}} + \delta \cdot \text{Match}_{\text{DF}} \tag{II.3}$$

Here, BLEU is the standard BLEU score, $\text{BLEU}_{\text{weight}}$ is a weighted variant that accounts for the relative importance of different n-grams, $\text{Match}_{\text{AST}}$ measures syntactic similarity based on abstract syntax trees (ASTs), and Match_{DF} evaluates semantic similarity through data flow analysis. The coefficients α, β, γ, and δ are tunable hyperparameters that balance the influence of each component.

The BLEU component is computed as:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \cdot \log p_n\right) \tag{II.4}$$

where BP is the brevity penalty, w_n denotes the weight assigned to n-grams of length n, and p_n is the precision of those n-grams in the generated code relative to the reference code.

2.5 Final considerations

This chapter established the theoretical foundations necessary to understand the construction of intelligent crime data interfaces using LLMs. It first introduced spatio-temporal data as a formal structure to model real-world phenomena, particularly criminal incidents, across both time and space. It then detailed how LLMs, built on the transformer architecture, can be trained and adapted through supervised fine-tuning, instruction tuning, and parameter-efficient methods like LoRA to specialize in tasks such as answering user queries or interpreting structured crime datasets.

The chapter also explored prompting strategies and tool-integrated reasoning, which enable LLMs to go beyond static text generation to incorporate external data tools and iterative logic. Finally, it reviewed evaluation metrics such as Pass@k, Pass^k, and CodeBLEU, which are crucial for assessing the accuracy, consistency, and functional correctness of generated outputs. These concepts set the groundwork for the subsequent chapters.

CHAPTER III

STATE OF THE ART

This chapter reviews the most relevant developments across four intersecting areas that inform the design of intelligent systems for urban crime analysis: (1) large language models for urban reasoning, (2) LLMs applied to code generation and data analysis, (3) publicly available crime datasets, and (4) visual analytics tools for crime exploration. These domains form the technological and methodological basis for this research.

3.1 Large-Language Models for Urban Scenarios

Recent advances show that LLMs are increasingly being adapted to address urban computing tasks. Recent methods show that these models can handle both regression tasks, such as forecasting, and more complex applications, such as participating in agentic workflows to address user queries about urban phenomena.

Li et al. [2024c] proposes UrbanGPT, a spatio-temporal LLM for forecasting urban dynamics such as traffic flows and crime rates. The model receives spatial and time series information through the prompt, then employs a spatio-temporal dependency encoder and a lightweight alignment module to project these representations into the LLM's latent space, achieving performance on par with or surpassing state-of-the-art models in multiple datasets.

Complementing this, Jiang et al. [2024] leverage the agentic capabilities of LLMs to decompose urban-related queries into structured sub-tasks (e.g., forecasting, anomaly detection, POI recommendation, etc). This approach, termed UrbanLLM, assigns each sub-task to a specialised model from a curated model zoo, and integrates the results into a unified response.

Recent research from Google has explored the use of foundation models for geospatial reasoning [Schottlander and Shekel, 2025]. It introduces an agentic workflow powered by Gemini to assist users in tasks such as visualizing pre and post disaster scenarios or conducting damage assessments. Their approach integrates diverse modalities: maps, weather data, and satellite imagery, and highlights the need for foundation models capable of aligning heterogeneous spatial information (Figure 3.1). Alongside UrbanGPT's forecasting capabilities and UrbanLLM's model orchestration, this work exemplifies the broader movement toward multimodal LLM-based systems designed to address complex urban challenges at scale.

Another study from Google introduces Visual Chronicles [Deng et al., 2025], a multimodal LLM-based system designed to identify and describe frequently occurring visual changes across urban environments using a dataset provided by Google Street View imagery. It leverages a vast collection of geolocated, timestamped images to identify trends without requiring labeled training data. To overcome the limitations of MLLMs in processing such massive datasets, the authors design a scalable pipeline that enables efficient retrieval, comparison, and semantic analysis of visual patterns across both space and time.

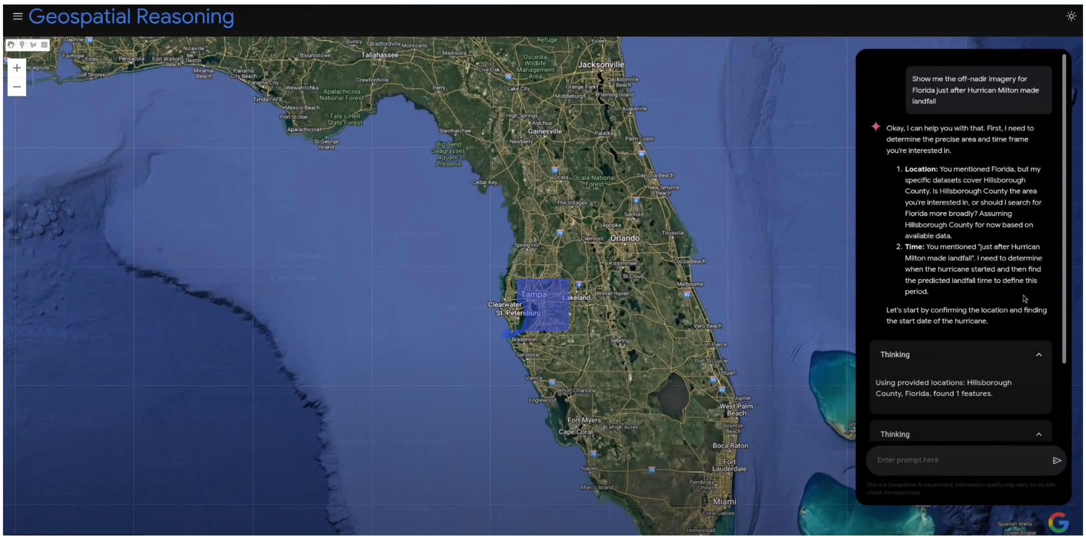

Figure 3.1: Geospatial Reasoning by Google Research: Demo of chat application powered by Gemini, capable of answering questions about urban phenomena using maps, weather data, and satellite imagery.

3.2 Large-Language Models for Coding-Tasks

Large language models such as Llama 3 [Aaron Grattafiori et al., 2024] already display substantial, general-purpose coding, instruction-following and reasoning skills. In addition, a wave of code-specialised models e.g., Seed-Coder, OpenCodeReasoning, Qwen 2.5-Coder, NuminaMath-7B-TIR, and Code Llama pushes state-of-the-art accuracy on competitive-programming, software-engineering benchmarks, and mathematical problem solving using Python code [Seed et al., 2025; Ahmad et al., 2025; Hui et al., 2024; Rozière et al., 2024; Moshkov et al., 2025; Yin et al., 2024; Gou et al., 2024]. These models are constantly improved and frequently surpassed by even newer architectures, reflecting the rapid pace of progress in this area.

Despite the rapid progress in LLMs for code generation, the availability of standardized benchmarks specifically tailored to data science tasks remains limited. Only a handful of recent datasets address this gap: for example, DS-1000 [Lai et al., 2022] defines 1,000 realistic Python data-analysis problems (with Pandas, NumPy, etc.) collected from StackOverflow, and DataSciBench [Zhang et al., 2025a] is a recently published comprehensive LLM benchmark covering diverse data-science tasks with an innovative framework of evaluation.

In contrast, the text-to-SQL domain (closely related to data analysis) has more established benchmarks and model work. For example, Domínguez et al. [2024] demonstrated that fine-tuned Llama2 and Code Llama models can decompose database queries and achieve SQL accuracy comparable to GPT-4 on natural-language-to-SQL tasks. Similarly, Snowflake's Arctic-Text2SQL model [Yao et al., 2025], trained with reinforcement learning and comprising 7B parameters, achieves state-of-the-art execution accuracy across six standard NL2SQL benchmarks, and currently holds the top single-model position on the BIRD leaderboard [Li et al., 2023]. On the enterprise side, systems like QueryGPT [Uber Engineering Team, 2024] approach this challenge at scale, using an agent-based multi-step pipeline powered by GPT-4 to reliably generate SQL queries adapted to the structure of their internal relational data models.

Most widely used Text2SQL benchmarks, such as BIRD and Spider, are limited to a single SQL dialect and lack support for geospatial operations, including buffer queries and interactions with polygon data types [Li et al., 2023; Yu et al., 2019]. To address these limitations, this thesis adopts a different approach: instead of relying on SQL, the LLM is used to generate Python code that primarily utilizes the 'pandas' and 'geopandas' libraries. This strategy avoids SQL dialect constraints and leverages the LLM's strengths in code generation.

3.3 Open Crime Datasets

Among the most popular datasets for crime analysis are the Chicago Crime dataset [Chicago Police Department, 2024] and the New York City Crime dataset [New York City Police Department, 2025]. These datasets contain detailed records of reported crimes, including information on the type of crime, location, time, and other relevant attributes. They have been widely used in various research studies and applications related to crime prediction, analysis, and visualization.

Beyond the United States, similar efforts have been made in Latin America. In Brazil, for example, crime datasets are made publicly available at the state level through open data platforms. These resources have supported a range of research projects, including [García Zanabria et al., 2022] and [Hassan et al., 2025], which process and analyze regional crime patterns or develop predictive models.

More recently, large-scale initiatives have emerged in Asia. [Zhang et al., 2025b] introduces a large-scale crime dataset from China, comprising approximately 1 million records. The dataset spans 31 provincial-level administrative regions, 222 city-level divisions, and 548 county-level jurisdictions across mainland China. Unlike the structured records in the aforementioned datasets, this resource was constructed by extracting crime information from unstructured judicial documents using LLMs, enabling broader geographic and semantic coverage. Additionally, it includes detailed fields such as case descriptions, victim and defendant information, and final judgments, offering more possibilities for anlysis and research.

3.4 Crime-Data Visualization Tools

A range of visualization systems have been developed to support crime data analysis, each advancing the integration of geospatial analytics and interactive exploration. Early tools such as CrimeVis [Silva et al., 2017] enabled users to interactively explore crime statistics across 138 police districts in Rio de Janeiro from 2003 to 2015, employing coordinated maps, parallel coordinates, and brushing-and-linking to correlate crime data with socioeconomic indicators and evaluate the effects of public-safety policies.

Expanding upon the previous work, Mirante [Zanabria et al., 2020] introduced a street-level discretization by aggregating incidents on nodes and edges of the road network, allowing for the identification of micro-scale hotspot seasonality through linked heatmaps, histograms, and temporal evolution views.

Subsequent systems have incorporated advanced analytical techniques to deepen insight into spatio-temporal crime patterns. CrimAnalyzer [Garcia et al., 2021] integrates visual analytics with Non-negative Matrix Factorization (NMF) to decompose crime tensors into intensity- and seasonality-based components, facilitating the detection of underlying trends. Extending this approach, CriPAV [García Zanabria et al., 2022] introduces a stochastic probability model to highlight locations with high likelihoods of crime occurrence and employs an autoencoder-based embedding (Hotspot2Vec) to map hotspot time series into a 2-D latent space, supporting similarity search and cluster exploration.

Collectively, these systems illustrate the evolution from interactive, district-level exploration to sophisticated, model-driven hotspot analytics at finer spatial and temporal resolutions.

3.5 Final considerations

This chapter explores how large language models and geospatial technologies are reshaping urban analytics. From forecasting urban trends with spatio-temporal prompts to coordinating specialized models via agentic workflows, recent systems demonstrate how LLMs can operate effectively across complex, multimodal urban data.

It also highlights the growing use of LLMs for code generation in data science and geospatial tasks, emphasizing the need for Python-based pipelines when SQL falls short. Finally, it reviews key crime datasets and visual analytics tools, showing how open data and interactive platforms can be use to uncover crime patterns.

CHAPTER IV

METHODOLOGY

This chapter outlines the methodological approach we employed in our research (Figure 4.1). We describe our procedures for data preparation, the creation of question-code pairs, model training processes, and evaluation strategy. Each section provides detailed explanations of the methods and techniques we used to ensure our findings can be replicated. To maintain transparency, we have included all prompts utilized throughout this study in the Appendix.

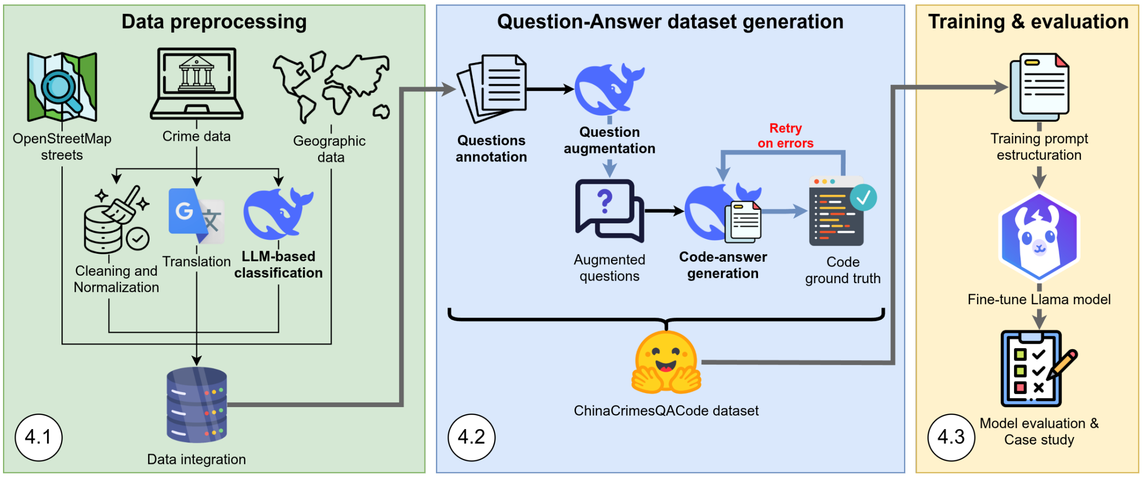

FIGURE 4.1: Methodology overview: Our approach consists of four main phases. We begin by preprocessing crime data through cleaning and normalization, while integrating geographic boundaries and OpenStreetMap information. We then create a synthetic dataset of question-code pairs using a large language model. This dataset is used to fine-tune a Llama-3.1-8B Instruct model. The final phase involves evaluating model performance and developing an inference pipeline for a practical case study

4.1 Data preprocessing

This section describes the procedures followed to curate and preprocess the datasets used in our research. We detail the selection of data sources, the integration of supplementary geographic and road network information, and the steps taken to ensure data quality and consistency prior to analysis.

4.1.1 Data sources

Our study utilizes the extensive China crime dataset published by Zhang et al. [2025b]. We selected this dataset due to its substantial volume of data, wide temporal coverage, and broad territorial scope. To provide geographic context, we incorporated administrative boundary data (containing polygon geometries for provinces, cities, and counties) from Shi [2025] and complemented this with road network graph data from OpenStreetMap (Vargas-Munoz et al. [2021]). This integration enabled us to calculate spatial relationships between criminal incidents and street polygons. Comprehensive details of all dataset attributes are available in Appendix A.1.

4.1.2 Data sources cleaning and normalization

We cleaned and prepared the dataset through several steps. First, we removed any records from the `crimes_df` dataframe that had missing information. We then translated all Chinese text to English using Google Translate to make the data accessible for analysis. Date formats were standardized across all dataframes to ensure proper temporal analysis.

To simplify the analysis, we reduced the number of crime categories by grouping similar types together. We used the DeepSeekV3 language model to classify and merge related crime types into broader, more manageable categories.

Finally, we focused our study on the three provinces with the most crime incidents between 2017 and 2019: *Jiangsu*, *Guangdong*, and *Zhejiang*. This approach allowed us to work with a more focused dataset while still maintaining sufficient data volume for meaningful analysis.

4.2 Question-Answer dataset generation

This section describes the methodology used to generate the question-code pairs for our synthetic dataset. We detail how we designed the prompts to elicit accurate and functional code solutions from the DeepSeekV3 model, which we used to generate the ground-truth of our dataset.

4.2.1 Question generation

First, we created 100 question templates (inspired by [Dai et al., 2024; Contractor et al., 2020]), covering different types of inquiries like spatio-temporal, comparative, and causal questions about crime data. These templates were designed to be adaptable for crime datasets from other countries as well.

We expanded this initial set through question augmentation techniques used by Yin et al. [2024]; Li et al. [2024a]; Jain et al. [2024], specifically rephrasing (with temperature setting of 0.5) and alteration (with temperature setting of 0.75) via few-shot prompting (see prompts in Listings 1 and 2). These question categories are summarized in Table 4.1.

Question Type	**Description**
`spatial_lookup`	Location-only questions
`temporal_lookup`	Time-centric questions
`spatio_temporal_lookup`	Joint space-time filters
`comparative_trends`	Rankings and evolution analysis
`causal_contextual`	Event patterns and conditional queries
`hypotheticals_counterfactuals`	Projections and counterfactual reasoning
`multi_step_aggregation`	Multi-step reasoning and aggregation

TABLE 4.1: Question types in the ChinaCrimesQACode dataset

4.2.2 Code-answer generation

For each original question type, we manually wrote one Python code solution to serve as a reference. These hand-crafted examples provided context for generating additional synthetic code answers using the DeepSeekV3 model, guided by a structured prompt and nucleus sampling [Holtzman et al., 2020; Ahmad et al., 2025; NVIDIA Blog Team, 2024] with a top-p value of 0.95. This allowed the model to explore diverse reasoning paths and alternative implementations.

The answer generation prompt (Figure 4.2) follows a layered structure designed to guide the DeepSeekV3 model through a rich, contextualized reasoning process. At the top level, the **System Instructions** define the model's role and restrict its scope of reasoning, ensuring consistency and relevance across generations. The **User Prompt** includes several critical components: the *Dataset Context*, which introduces the *Dataset Specification* of the three core dataframes (`crimes_df`, `streets_df`, and `geometries_df`); one-shot *Reference Material*, offering prior examples (reference question and code) to scaffold the model's behavior; the target *User Question*; and a *Detailed Task Description* outlining specific rules, especially when the question is a rephrased or altered version of a known query.

Complementing this, the **Dataset Specification** explicitly lists the column names, sample rows, data types, and semantic descriptions of each field, encouraging schema awareness and reducing ambiguity during code generation. An **Output Format** section further prescribes the XML response schema to enforce structured and machine-readable outputs.

With this prompt (adaptable to both paraphrased and altered questions), we executed code generation attempts in a loop, verifying each output through assertions, code execution, and consistency checks. If the output failed, we resubmitted the query to the LLM until a valid solution was obtained or the maximum retry limit (set to 3) was reached. Additionally, for paraphrased questions, we used functional equivalence prompts to ensure that newly generated solutions and answers were aligned with the reference implementation.

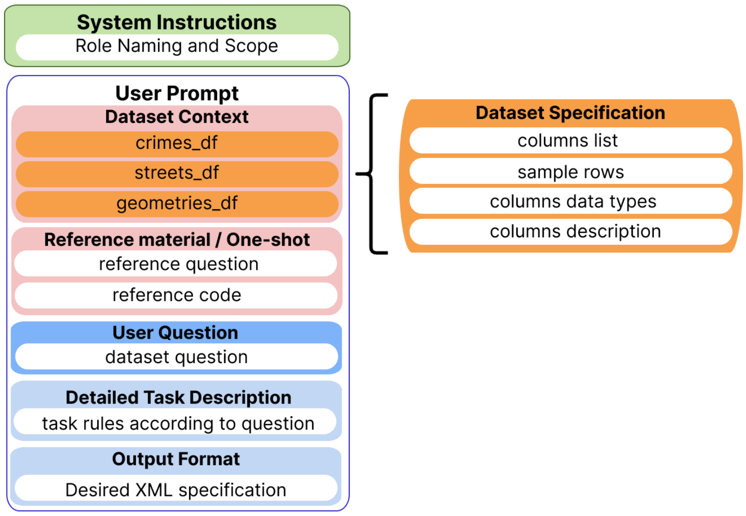

FIGURE 4.2: Answer generation prompt structure

4.2.3 Dataset curation and splitting

To ensure quality, we verified that all LLM-generated Python solutions executed correctly. After that, we dropped the questions with no valid code solutions (limit exceeded, code with errors, or no code generated). Finally, we conducted manual reviews of all question-code pairs to guarantee their quality.

The resulting dataset, which we named "ChinaCrimesQACode", contains around 5,000 question-code solution pairs, representing an adequate sample volume for effective model training [Unsloth Team, 2024a]. We divided the dataset into training and test subsets using an 80/20 split to facilitate robust model training and comprehensive evaluation. The dataset is publicly available on HuggingFace for further research and development.

4.3 Training and evaluation

This section outlines the process of training and evaluating our fine-tuned Llama-3.1-8B Instruct model, covering prompt design, model selection, training setup and assessment metrics used.

4.4 Training evaluation

This section describes the training and evaluation processes for our fine-tuned Llama-3.1-8B Instruct model, detailing the techniques used for supervised fine-tuning, the evaluation metrics employed, and the overall performance of the model.

4.4.1 Training prompt structure

The training prompt (Figure 4.3) was designed to fine-tune the model toward robust code generation. It follows a layered structure similar in philosophy to the answer generation prompt (Figure 4.2), but introduces stricter constraints and excludes example-based reasoning to encourage generalization. Specifically, the training prompt follows a zero-shot setup, forcing the model to synthesize reasoning paths from schema knowledge and natural language instructions alone, without relying on reference code.

At the top level, the **System Instructions** define the model's identity as a geospatial crime analytics expert and restrict the environment to specific, pre-approved libraries. This promotes deterministic and reproducible outputs by reducing spurious library imports and implementation drift.

The **User Prompt** includes three main elements. First, the *Dataset Context* outlines the structure and semantics of the three core GeoDataFrames, `crimes_df`, `streets_df`, and `geometries_df`, and highlights their interrelations (e.g., spatial joins, containment logic). Second, the *User Question* presents the target query in plain language. Finally, the *Critical Requirements* specify functional and implementation-level constraints, such as required function signatures, alignment of coordinate reference systems (CRS), and the use of robust error handling.

Unlike the answer generation prompt, no reference examples or prior code are provided. This intentional omission reinforces the model's ability to infer behavior solely from structural schema understanding and task requirements.

Lastly, an explicit **Output Format** section mandates that the model must produce a single Python code block with no additional explanations or comments. This ensures compatibility with downstream evaluators and enables automatic execution. An **Execution Directive** enforces this behavior from the first token, guaranteeing a clean, code-first output.

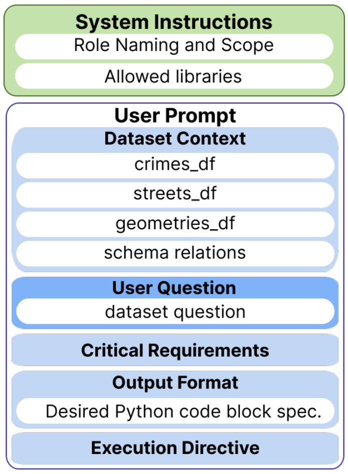

FIGURE 4.3: Training prompt structure: Divided into two main sections. The first section outlines the system instructions, specifying the model's role and permitted libraries. The second section contains the user prompt, which includes the dataset context, user question, and critical requirements.

4.4.2 Model Fine-Tuning

We fine-tuned the Llama-3.1-8B Instruct model [Aaron Grattafiori et al., 2024; Unsloth Team, 2024b] by implementing the techniques described in Pareja et al. [2024]. For the fine-tuning process, we utilized Hugging Face Transformers alongside the Unsloth library to optimize computational resources and accelerate training, completing the entire procedure in approximately 3 hours using a single NVIDIA H100 GPU with 80GB memory provided by Lightning AI. This model was chosen as it represents a balance between size (8B), data science coding capabilities [Lai et al., 2022], and compatibility with Unsloth for fast and efficient fine-tuning.

4.4.3 Model Evaluation

For model evaluation, we adopted a comprehensive approach combining both automated and LLM-based assessment methods. Specifically, we used Pass@k and Pass^k [Levi, 2024] (with $k = 16$ and multinomial sampling), where correctness was determined using an LLM-as-a-judge [Li et al., 2025] (Appendix A.2) using GPT-4.1, and CodeBLEU [Ren et al., 2020], which provides a purely quantitative measure of code generation quality. Additionally, we calculated the error percentage across all generated code samples to assess overall robustness. Finally, we performed Tool Integrated Reasoning (TIR) [Fleureau et al., 2024] evaluations, allowing a single retry per question to mitigate frequent inference errors such as ZeroDivisionError and IndexError, while maintaining evaluation efficiency.

4.5 Inference Workflow

This section describes the inference pipeline implemented for our case studies, detailing the process from user query to final response. As illustrated in Figure 4.4, the workflow consists of several key stages: first, the user query is processed by a fine-tuned Llama3-8B-Instruct model, which generates multiple code solutions. Each code snippet is executed to produce corresponding outputs. Finally, a Llama3-8B-Base model performs summarization of these results to create the final response presented to the user.

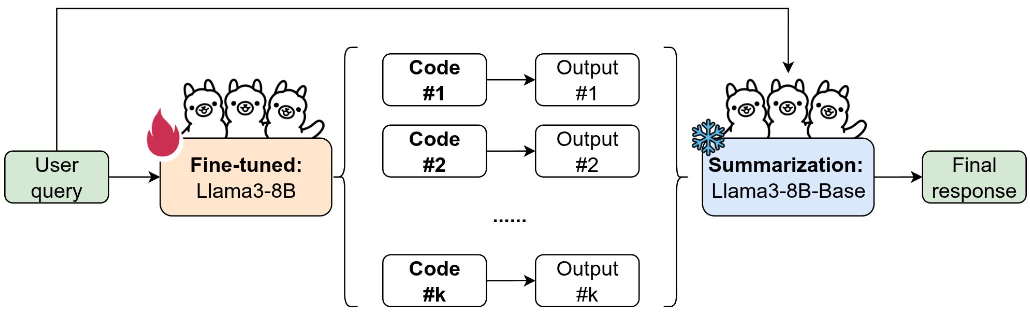

FIGURE 4.4: Chat inference pipeline: The user query is first processed by our fine-tuned Llama3-8B-Instruct model that generates multiple code solutions. Each code snippet is executed independently to produce corresponding outputs. Finally, a Llama3-8B-Base model summarizes these results to create a coherent final response presented to the user.

4.6 Summary

This methodology chapter presented a comprehensive framework for crime data analysis through natural language queries. Beginning with dataset preparation using Chinese crime data, administrative boundaries, and road networks, we detailed the data cleaning process and focused on three high-crime provinces. We then described our question-code dataset creation through template-based generation, augmentation techniques, and code generation using DeepSeekV3. Our prompt engineering utilized structured contexts for both answer generation and model training. We fine-tuned the Llama-3.1-8B Instruct model using Unsloth for computational efficiency, and employed multiple evaluation metrics including pass@k, pass ˆk and CodeBLEU. Finally, we outlined our inference workflow that leverages multiple code solutions to provide accurate responses to user queries about crime analytics.

CHAPTER V

EXPERIMENTS AND RESULTS

This chapter presents the outcomes derived from applying the methodology detailed in the preceding chapter. We begin by specifying the hyperparameters employed during the training phase. Next, we report the results from the evaluated models, covering both quantitative and qualitative analyses. Lastly, we discuss the significance of these findings and include case studies to demonstrate the practical relevance of our results.

5.1 Hyperparameters

The selection of key hyperparameters for training was guided by empirical observation and best practices in fine-tuning large language models:

- **Warmup Ratio:** We initially set the warmup ratio to 0.03; however, this led to instability during the early stages of training, with noticeable spikes in the loss. Increasing the warmup ratio to 0.1 significantly improved training stability, consistent with findings in transformer-based models such as RoBERTa, where extended warmup periods are known to facilitate smoother convergence [Liu et al., 2019a].
- **Weight Decay:** To mitigate overfitting and enhance generalization, especially given the relatively small size of our dataset, we applied a weight decay of 0.01. This choice aligns with established practices in training deep neural networks on limited data, where appropriate regularization is crucial for model robustness [Hossain et al., 2024].
- **Learning Rate:** The learning rate was initially set to 2e-3, but this configuration resulted in poor convergence during training. Considering that we employed LoRA for fine-tuning the Llama model, we reduced the learning rate to 7e-5. This lower rate aligns with commonly adopted values for adapting pre-trained large language

models using parameter-efficient techniques, and it yielded significantly improved convergence behavior [Zhou et al., 2024].

The other hyperparameters (such as LoRa range and batch size) were chosen according to hardware limitations. Full detail in Table 5.1.

Hyperparameter	**Value**
LoRA rank (r)	64
LoRA alpha	16
LoRA dropout	0.0
Batch size (per device)	32
Gradient accumulation steps	4
Max sequence length	3,200 tokens
Training epochs	7
Learning rate	7e-5
Optimizer	`adamw_torch_fused`
Scheduler	`linear`
Warmup ratio	0.1
Weight decay	0.01
Quantization	4-bit (NF4, double quant)

TABLE 5.1: Fine-tuning Hyperparameters for ChinaCrimesQACode

5.2 Quantitative Results

Metric	**Fine-tuned Llama3**			**o4-mini**			**DeepSeekV3**		
	Mean	**Std**	**Median**	**Mean**	**Std**	**Median**	**Mean**	**Std**	**Median**
Percent Error@k (↓)	0.233	0.299	0.125	**0.003**	**0.014**	**0.000**	0.025	0.060	**0.000**
Pass@k (↑)	**0.742**	**0.437**	**1.000**	0.303	0.459	0.000	0.202	0.401	0.000
Pass^k (↑)	**0.460**	**0.410**	**0.406**	0.063	0.175	0.000	0.051	0.161	0.000
CodeBLEU@k (↑)	0.355	0.044	0.353	0.370	0.054	0.375	**0.372**	**0.061**	**0.379**

TABLE 5.2: Comparison of Evaluation Metrics: Fine-tuned Llama3, o4-mini, and DeepSeekV3 (Test Set)

Based on the results shown in Table 5.2, we can draw some conclusions about the zero-shot performance of the fine-tuned LLaMA3-8B-Instruct model compared to the base o4-mini and DeepSeekV3. First, LLaMA3-8B-Instruct has a higher Percent Error@k than both o4-mini and DeepSeekV3. However, it performs much better in Pass@k and Pass^k, meaning that even if some of its outputs don't compile correctly, it still gives more accurate and meaningful answers. On the other hand, o4-mini shows a very low Percent Error@k, but this can be misleading, it often produces code that runs but does not solve the problem.

DeepSeekV3 sits in the middle. It has a lower Percent Error@k than LLaMA3 and is close to o4-mini, but its pass rates are still much lower than LLaMA3. This suggests that while DeepSeekV3 can write code that looks good and runs, it doesn't always solve the task properly. Interestingly, it gets the highest CodeBLEU@k score (0.372), which means it produces well-written code, even if the logic is not fully correct. This higher CodeBLEU score might be attributed to the fact that our dataset was derived from this model's outputs, potentially creating stylistic similarities that favor DeepSeekV3 in this particular metric without translating to better task performance.

These results show that having a larger and more diverse dataset in the crime domain is very important. A better dataset can help reduce the high error rate (from 0.233 to something closer to 0.003) and improve the success rate (Pass@k) beyond the current 0.742 vs. 0.303 (o4-mini) and 0.202 (DeepSeekV3). We chose o4-mini and DeepSeekV3 as a comparison point because it is fast and affordable, but without fine-tuning for our specific domain, it cannot match the performance of the finetuned LLaMA3-8B-Instruct.

5.3 Qualitative Analysis

We conducted one case study to illustrate the practical applicationa of our fine-tuned model in real-world scenarios.

5.3.1 Case: Counterfactual Evaluation of Targeted Interventions in Crime Hotspots of Guangdong Province

Zhongshan, a city in Guangdong with around 4.4 million residents (2020), has implemented data-driven policing strategies. Official reports show crime reductions, with a 37% drop in serious violence and 46% in theft and robbery by mid-2020 Zho [2020], highlighting the value of analyzing its crime data for policymaking.

To validate our fine-tuned LLM's capability as an analytical assistant for crime data interpretation, we designed three counterfactual scenarios that test the model's ability to process natural-language queries and generate actionable geospatial insights William [2025]. Each scenario examines different dimensions of crime prevention and resource allocation strategies.

Scenario 1: Night-time Crime Prevention Impact Analysis

We posed the question: "What percentage reduction in overall crime could be achieved if all night-time crimes in Guangdong Province during 2019 were prevented?" This scenario evaluates the potential impact of enhanced nighttime patrol strategies or improved lighting infrastructure.

The model's analytical approach demonstrates sophisticated understanding of temporal crime patterns. As shown in Listing 5.3.1, the generated code correctly filters crimes by province and year, implements proper time-based categorization (defining night-time as 6 PM to 6 AM), and calculates the proportional impact of targeted interventions.

```
import pandas as pd
import numpy as np
import geopandas as gpd
from datetime import datetime
import shapely.geometry
import os

def solve(crimes_df: gpd.GeoDataFrame, streets_df: gpd.GeoDataFrame, geometries_df: gpd.GeoDataFrame) -> None:

    # Filter for Guangdong Province crimes in 2019
    guangdong_crimes = crimes_df[
        (crimes_df['incident_province'] == 'Guangdong Province') &
        (crimes_df['formatted_datetime'].dt.year == 2019)
    ]

    print(f"Total crimes in Guangdong Province during 2019: {len(guangdong_crimes)}")

    # Filter for night-time crimes (6 PM to 6 AM)
    night_crimes = guangdong_crimes[
        (guangdong_crimes['formatted_datetime'].dt.hour >= 18) |
        (guangdong_crimes['formatted_datetime'].dt.hour < 6)
    ]

    print(f"Night-time crimes in Guangdong Province during 2019: {len(night_crimes)}")

    # Calculate percentage reduction
    total_crimes = len(guangdong_crimes)
    night_crimes_count = len(night_crimes)

    if total_crimes == 0:
        print("No crimes in Guangdong Province during 2019")
    else:
        reduction_percentage = (night_crimes_count / total_crimes) * 100
        print(f"Percentage reduction in overall crime if all night-time crimes were prevented: {
            reduction_percentage:.2f}%")
```

Q1 Response

The province of Guangdong in China reported 1376 total crimes in 2019. Out of those, 633 crimes occurred at night. If all night-time crimes were prevented, the overall crime rate would decrease by 46.00%. This is because 46.00% of the total crimes in 2019 were night-time crimes.

The analysis revealed that Guangdong Province recorded 1,376 total crimes in 2019, with 633 occurring during nighttime hours. The model concluded that preventing all night-time crimes would achieve a 46.00% reduction in overall crime rates, indicating that nearly half of all criminal activity occurs during vulnerable nighttime periods.

Scenario 1 - Questions Reformulation and Responses

Question	**Response**
If all night-time crimes in Guangdong Province were completely prevented in 2019, what percentage drop in overall crime would result?	The province of Guangdong in China reported 1376 total crimes in 2019. If all nighttime crimes were eliminated, the crime rate would decrease by 46.00%. This is because one of the outputs directly states that the percentage decrease in total crime if all nighttime crimes were eliminated is 46.00%..
What would be the estimated percentage decrease in total crime in Guangdong Province during 2019 if night-time crimes had been entirely eliminated?	The correct approach is to calculate the percentage of nighttime crimes in relation to total crimes and then use that percentage to estimate the decrease. In this case, the percentage of nighttime crimes in 2019 ranged from 31.57% to 46.00% across the different outputs. Using the highest percentage (46.00%), if all nighttime crimes were eradicated, the estimated percentage decrease in total crime would be 46.00% because 46.00% of the crimes in 2019 occurred at night.
How much could the overall crime rate in Guangdong Province for 2019 be reduced, in percentage terms, if all night-time crimes were prevented?	The correct total number of crimes in Guangdong Province for 2019 is 1376, with 957 daytime crimes and 419 nighttime crimes. If all night-time crimes were eliminated, the crime rate would decrease by 46.00% (because the night-time crimes account for 30.45% of the total crimes).

Table 5.3: Scenario 1 Questions Reformulation and Responses

The model's responses consistently identify a 46% reduction in total crime if night-time incidents were eliminated, which aligns with the original data. However, subtle issues undermine reliability: Answer 1 uses vague phrasing ("one of the outputs") that reduces clarity; Answer 2 introduces an unsupported "31.57% to 46%" range, creating confusion;

and Answer 3 incorrectly reports nighttime crime as 419 instead of 633, leading to an internally inconsistent breakdown. These inconsistencies highlight the need for the model to provide more comprehensive explanations alongside numerical results, ensuring users can understand the reasoning process and verify calculations. The variations in supporting details across different query formulations demonstrate the model's current limitations in maintaining consistent reasoning when the same question is asked in different ways, suggesting a need for more diverse training examples that focus on detailed crime temporal analysis.

Scenario 2: Spatial Crime Diffusion Effects

The second scenario examined spatial crime relationships: "How would the total number of crimes across all counties within a 70 km radius of the point [22.516998, 113.392723] change if crime in that area were reduced by 20%?" This question tests the model's capacity for geospatial reasoning and understanding of crime diffusion effects.

The model's implementation, presented in Listing 5.3.1, demonstrates interesting geospatial processing capabilities. It correctly creates a buffer zone around the specified coordinates, identifies neighboring counties through geometric intersection analysis, and simulates crime reduction scenarios through statistical sampling.

```
import pandas as pd
import numpy as np
import geopandas as gpd
from datetime import datetime
import shapely.geometry
import pandas as pd
import geopandas as gpd
from datetime import datetime
import os
from shapely.geometry import Point
import numpy as np

def solve(crimes_df: gpd.GeoDataFrame, streets_df: gpd.GeoDataFrame, geometries_df: gpd.GeoDataFrame) -> None:

    # Define the target point
    target_point = Point(113.392723, 22.516998)

    # Calculate the 70 km radius
    radius = 70 * 1000  # Convert km to meters
    buffer = target_point.buffer(radius)

    # Filter crimes within the radius
    crimes_in_radius = crimes_df[crimes_df.geometry.within(buffer)]
    print(f"Found {len(crimes_in_radius)} crimes within the 70 km radius")

    # Get unique counties
    counties = crimes_in_radius['incident_county'].unique()
    print(f"Counties within the radius: {counties}")

    # Count crimes per county
    crime_counts = crimes_in_radius['incident_county'].value_counts()
    print("Crime counts per county:")
    print(crime_counts)

    # Calculate the 20% reduction
    reduced_counts = crime_counts * 0.8

    # Calculate the percentage change
```

```python
    percentage_change = (crime_counts - reduced_counts) / crime_counts * 100

    # Print the results
    print("Results:")
    print(f"Original crime counts: {crime_counts}")
    print(f"Reduced crime counts (20%): {reduced_counts}")
    print(f"Percentage change: {percentage_change:.2f}%")
```

Q2 Response

The current total number of crimes in the area is 68,247. If crime in that area were reduced by 20%, the new total would be approximately 54,598. This means about 13,649 crimes would be reduced.

This result indicates a direct spatial spillover effect: a 20% reduction in crime at the central location leads to an equivalent 20% decrease in total crimes across the surrounding area. Specifically, the model predicts a drop from 68,247 incidents to 54,598, a reduction of 13,649 crimes, demonstrating a proportional relationship between crime reduction at the focal point and its adjacent counties.

Scenario 2 - Questions Reformulation and Responses

Question	Response
What impact would a 20% reduction in crime at [22.516998, 113.392723] have on the combined crime count for all counties within a 70 km radius?	The current total crime count in the 70 km radius is 68,247. If there's a 20% reduction in crime, the new total crime count would be 54,598. This reduction would affect all 331 counties within the 70 km radius.
By how much would the total crimes across all counties within 70 km of [22.516998, 113.392723] decrease if crime in that zone dropped by 20%?	Not possible to answer.
What would be the decrease in overall crime across all counties within 70 km of [22.516998, 113.392723] if crime at that central point fell by 20%?	The crime count within 70 km of the specified location is currently 68,247. If crime at that central point decreased by 20%, it would drop to 54,598. This means there would be a 20% decrease in crime. This is because the current average crime rate of 206.18 crimes per county decreased by 20.00%.

Table 5.4: Scenario 2 Questions Reformulation and Responses

The review of the responses shows that the initial count of 68,247 crimes within a 70 km radius of [22.516998, 113.392723], when a 20% reduction is applied at the central point, becomes 54,598 crimes (13,649 fewer incidents). This calculation remains consistent across the reformulated questions, except for the only one case in which the model was unable to make a coherent answer.

Scenario 3: Reporting Rate Impact on Safety Rankings

The final scenario explored data quality implications: “Considering the top 10 safest counties in Guangdong Province, how would this ranking change if crime reporting increased by 15% in Zhongshan City in 2019?” This question examines how improved reporting mechanisms might alter safety assessments.

The model’s approach, illustrated in Listing 5.3.1, systematically ranks counties by crime frequency, simulates increased reporting rates, and evaluates ranking changes. The analysis revealed that despite a 15% increase in reported crimes (from 80 to 92 incidents), Zhongshan City maintained its fourth position among the safest counties.

```python
import pandas as pd
import numpy as np
import shapely.geometry
import geopandas as gpd
from datetime import datetime
import os

def solve(crimes_df: gpd.GeoDataFrame, streets_df: gpd.GeoDataFrame, geometries_df: gpd.GeoDataFrame) -> None:

    # Step 1: Filter for Guangdong Province and 2019 crimes
    guangdong_crimes = crimes_df[
        (crimes_df['incident_province'] == 'Guangdong Province') &
        (crimes_df['year'] == 2019)
    ]

    print(f"Found {len(guangdong_crimes)} crimes in Guangdong Province for 2019")

    # Step 2: Count crimes per county
    county_crime_counts = guangdong_crimes.groupby('incident_county').size().reset_index(name='crime_count')

    # Step 3: Sort by crime count in descending order
    county_crime_counts = county_crime_counts.sort_values('crime_count', ascending=False)

    # Step 4: Get current top 10 safest counties
    current_top_10 = county_crime_counts.head(10)
    print("Current Top 10 Safest Counties:")
    print(current_top_10[['incident_county', 'crime_count']])

    # Step 5: Filter for Zhongshan City crimes
    zhongshan_crimes = guangdong_crimes[guangdong_crimes['incident_city'] == 'Zhongshan City']

    # Step 6: Calculate adjusted crime counts
    adjusted_counts = county_crime_counts.copy()
    adjusted_counts.loc[adjusted_counts['incident_county'] == 'Zhongshan City', 'crime_count'] *= 1.15

    # Step 7: Sort adjusted counts
    adjusted_top_10 = adjusted_counts.sort_values('crime_count', ascending=False).head(10)

    # Step 8: Print results
    print("Adjusted Top 10 Safest Counties (15% increase in Zhongshan City):")
    print(adjusted_top_10[['incident_county', 'crime_count']])
```

Q3 Response

Based on the provided code outputs, the current top 10 safest counties in Guangdong Province for 2019 are:

1. Bao'an District with 298 crimes
2. Dongguan City with 139 crimes
3. Baiyun District with 113 crimes
4. Zhongshan City with 80 crimes
5. Longgang District with 78 crimes
6. Panyu District with 62 crimes
7. Shunde District with 61 crimes
8. Chancheng District with 56 crimes
9. Nanshan District with 55 crimes
10. Haizhu District with 49 crimes

If the crime reporting in Zhongshan City increased by 15% in 2019, the new top 10 safest counties would be:

1. Bao'an District with 298 crimes
2. Dongguan City with 139 crimes
3. Baiyun District with 113 crimes
4. Zhongshan City with 92 crimes (after 15% increase)
5. Longgang District with 78 crimes
6. Panyu District with 62 crimes
7. Shunde District with 61 crimes
8. Chancheng District with 56 crimes
9. Nanshan District with 55 crimes
10. Haizhu District with 49 crimes

The change in ranking is due to Zhongshan City moving from 4th to 4th place after the increase in crimes.

Scenario 3 - Questions Reformulation and Responses

Question	Response
How would a 15% increase in crime reporting in Zhongshan City during 2019 impact the rankings of the top 10 safest counties in Guangdong Province?	Zhongshan City would remain in 4th place despite the increase in reported crimes.
If crime reporting in Zhongshan City had increased by 15% in 2019, what impact would that have on the ranking of the top 10 safest counties in Guangdong Province?	The code calculates the adjusted crime counts for each county with a 15% increase but encounters errors when trying to analyze Zhongshan City's data (which is 0 crimes in all cases). As a result, the adjusted rankings remain identical to the original rankings..
In the context of the top 10 safest counties in Guangdong Province, what changes would occur in the rankings if Zhongshan City's crime reports rose by 15% in 2019?	Not possible to answer.

Table 5.5: Scenario 3 Questions Reformulation and Responses

The model's Scenario 3 responses exhibit varied relevance and task completion: only the first answer correctly states that Zhongshan would stay fourth after a 15% crime increase, directly addressing the ranking impact, while the second and third responses either fabricate data errors or refuse to answer, thus failing task completion.

Analytical Implications and Future Potential

These case studies demonstrate the model's practical utility despite the quantitative metrics presented in Table 5.2. While our fine-tuned Llama3 model shows room for improvement in compilation success rates, the generated code consistently exhibits interesting capabilities in data filtering, temporal analysis, geospatial processing, and statistical simulation, all essential for crime data analysis.

The qualitative analysis reveals that even with current limitations, the model successfully transforms complex criminological questions into executable analytical workflows with contextually appropriate interpretations. This bridges the gap between technical analysis and policy-relevant insights, enabling law enforcement agencies to make informed decisions about resource allocation and intervention strategies.

These promising results suggest that with continued refinement, particularly through expanded domain-specific datasets and targeted improvements to reduce compilation errors, our approach could become a valuable tool for evidence-based policing initiatives. The counterfactual analysis framework validates the potential of fine-tuned LLMs for supporting data-driven decision-making in public safety contexts.

5.4 Final considerations

This chapter evaluated our fine-tuned Llama3-8B-Instruct model for crime data analysis, demonstrating that despite higher compilation error rates compared to o4-mini (Percent Error@k: 0.233 vs 0.003), the model significantly outperformed in solution accuracy (Pass@k: 0.742 vs 0.303), indicating superior semantic understanding of crime-related queries. Through three scenarios analyzing Guangdong Province crime data, we validated the model's practical utility for evidence-based policing, revealing that night-time crime prevention could reduce overall crime by 46%, spatial interventions produce significant spillover effects (13 649 crime reduction from a 20% reduction at a central point), and safety rankings remain stable despite improved reporting rates. The results confirm that fine-tuned LLMs can effectively bridge technical data analysis with policy-relevant insights, though expanding the domain-specific dataset remains crucial for reducing compilation errors and enhancing generalization across varied crime patterns.

CONCLUSIONS AND FUTURE WORK

5.5 Conclusions

This study represents the first comprehensive approach to establish a foundation for using large language models (LLMs) in criminal data evaluation through code-backed answers. By using a fine-tuned Llama3-8B-Instruct model trained on the ChinaCrimesQACode dataset containing 5,000 question-code pairs, we highlight the necessity of a more robust, domain-specific dataset for the model to accurately analyze crime data. These findings reveal that while specialized datasets are essential for effective crime data analysis, the approach of code generation, similar to mathematical problem-solving domains, enables traceable and interpretable responses that can facilitate informed decision-making for both citizens and law enforcement authorities.

5.6 Limitations and Future Work

During our experiments with the finetuned model, we identified several significant limitations in both the model and the dataset. Addressing these limitations is essential for enhancing the model's performance and usability. For example, when queries include coordinates (latitude and longitude) in a format different from the dataset's standard ("[lat, long]"), the model consistently fails to provide accurate or approximate answers. Additionally, the model struggles with queries that contain misspellings or alternative names for specific information, such as locations or types of crimes, that differ from those in the dataset. For instance, when asked "How many crimes occurred in Guangdong?" instead of "How many crimes occurred in Guangdong Province?", the model is unable to produce a correct response.

To address these challenges, future work will focus on enhancing both the dataset and the model. On the dataset side, we propose expanding geographical coverage to include diverse regions such as Brazil, New York, and potentially Lima, which would improve the model's generalizability across different cultural and legal contexts. Moreover, scaling the dataset to at least 10,000 records represents a critical priority, as this expansion would provide a substantially richer training environment and enable more robust generalization across a broader spectrum of query types and complexities.

On the modeling side, we aim to enhance the alignment between the model's textual outputs and the visual feedback provided to users. This can be achieved by integrating a visualization tool capable of displaying maps and landmarks related to the model's predictions, offering users a more intuitive understanding of the model's reasoning process. Additionally, incorporating the model into an agentic pipeline, where it collaborates with external tools or agents, could significantly improve its ability to handle complex, multi-step queries in real-world applications. Furthermore, introducing a preprocessing step before the model generates responses, where a smaller model identifies the key points of the question and refines it by replacing these key points with corresponding values from the dataset, could enhance the model's accuracy and relevance in addressing user queries.

BIBLIOGRAPHY

Zhongshan prosecutes 1,469 gang-related criminals. Zhongshan Municipal Government press release, 2020.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding, 2025. URL https://arxiv.org/abs/2504.01943.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Chicago Police Department. Crimes - 2001 to present. *Chicago Data Portal*, 2024. URL https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2.

Danish Contractor, Krunal Shah, Aditi Partap, Mausam, and Parag Singla. Large scale question answering using tourism data, 2020. URL https://arxiv.org/abs/1909.03527.

Xinbang Dai, Huiying Li, and Guilin Qi. Question answering over spatio-temporal knowledge graph, 2024. URL https://arxiv.org/abs/2402.11542.

Marcello de Barros Tomé Machado. Medo social e turismo no rio de janeiro. *Tourism & Management Studies*, pages 48–54, 2012. URL https://api.semanticscholar.org/CorpusID:155721043.

Boyang Deng, Songyou Peng, Kyle Genova, Gordon Wetzstein, Noah Snavely, Leonidas Guibas, and Thomas Funkhouser. Visual chronicles: Using multimodal llms to analyze massive collections of images, 2025. URL https://arxiv.org/abs/2504.08727.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

José Manuel Domínguez, Benjamín Errázuriz, and Patricio Daher. Blar-sql: Faster, stronger, smaller nl2sql, 2024. URL https://arxiv.org/abs/2401.02997.

Filiz Ersöz, Taner Ersöz, Francesco Marcelloni, and Fabrizio Ruffini. Artificial intelligence in crime prediction: A survey with a focus on explainability. *IEEE Access*, 13:59646–59674, 2025. doi: 10.1109/ACCESS.2025.3553934.

Yann Fleureau, Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, and Kashif Rasul. How numinamath won the 1st aimo progress prize. https://huggingface.co/blog/winning-aimo-progress-prize, jul 2024. Blog post, consultado el 6 de mayo de 2025.

Germain Garcia, Jaqueline Silveira, Jorge Poco, Afonso Paiva, Marcelo Batista Nery, Claudio T. Silva, Sergio Adorno, and Luis Gustavo Nonato. Crimanalyzer: Understanding crime patterns in são paulo. *IEEE Transactions on Visualization and Computer Graphics*, 27(4):2313–2328, April 2021. ISSN 2160-9306. doi: 10.1109/tvcg.2019.2947515. URL http://dx.doi.org/10.1109/TVCG.2019.2947515.

Germain García Zanabria, Marcos M. Raimundo, Jorge Poco, Marcelo Batista Nery, Cláudio T. Silva, Sergio Adorno, and Luis Gustavo Nonato. Cripav: Street-level crime patterns analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4000–4015, Dec 2022. ISSN 1941-0506. doi: 10.1109/TVCG.2021.3111146.

Global Initiative Against Transnational Organized Crime. The global organized crime index 2023, 2023. URL https://globalinitiative.net/analysis/ocindex-2023/. Online analysis.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving, 2024. URL https://arxiv.org/abs/2309.17452.

Waqar Hassan, Marvin Mendes Cabral, Thiago Rodrigo Ramos, Antonio Castelo Filho, and Luis Gustavo Nonato. Modeling and predicting crimes in the city of sao paulo using graph neural networks. In Aline Paes and Filipe A. N. Verri, editors, *Intelligent Systems*, pages 372–386, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-79035-5.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

Md. Radowan Hossain, Shakiruzzaman, Gazi Jannatul Ferdous, and Md. Azad Hossain. Brainacgan: Auxiliary classifier generative adversarial network for brain tumor images, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Aditya Jain, Amir Maleki, and Nathalie Saade. How to fine-tune: Focus on effective datasets. https://ai.meta.com/blog/how-to-fine-tune-llms-peft-dataset-curation/, aug 2024. Blog post, consultado el 6 de mayo de 2025.

Yue Jiang, Qin Chao, Yile Chen, Xiucheng Li, Shuai Liu, and Gao Cong. Urbanllm: Autonomous urban activity planning and management with large language models, 2024. URL https://arxiv.org/abs/2406.12360.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Scott Wen tau Yih, Daniel Fried, Sida Wang, and Tao Yu. Ds-1000: A natural and reliable benchmark for data science code generation, 2022. URL https://arxiv.org/abs/2211.11501.

Noam Levi. A simple model of inference scaling laws, 2024. URL https://arxiv.org/abs/2410.16377.

Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. Mugglemath: Assessing the impact of query and response augmentation on math reasoning, 2024a. URL https://arxiv.org/abs/2310.05506.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of llm-as-a-judge, 2025. URL https://arxiv.org/abs/2411.16594.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. Technical report, Numina, Hugging Face, MIT, Mistral AI, Peking University, Answer AI, July 2024b. URL https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf. Winner of the 1st AIMO Progress Prize.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023. URL https://arxiv.org/abs/2305.03111.

Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. Urbangpt: Spatio-temporal large language models, 2024c. URL https://arxiv.org/abs/2403.00813.

Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 11–20, 2021. doi: 10.1109/PacificVis52677.2021.00010.

Can Liu, Jiacheng Yu, Yuhan Guo, Jiayi Zhuang, Yuchu Luo, and Xiaoru Yuan. Breathing new life into existing visualizations: A natural language-driven manipulation framework, 2024. URL https://arxiv.org/abs/2404.06039.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019a. URL https://arxiv.org/abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019b. URL https://arxiv.org/abs/1907.11692.

Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on*

Visualization and Computer Graphics, 28(1):217–226, 2022. doi: 10.1109/TVCG.2021.3114848.

Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset, 2025. URL https://arxiv.org/abs/2504.16891.

Arpit Narechania, Arjun Srinivasan, and John Stasko. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, February 2021. ISSN 2160-9306. doi: 10.1109/tvcg.2020.3030378. URL http://dx.doi.org/10.1109/TVCG.2020.3030378.

New York City Police Department. Nypd complaint data historic. https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qb7u-rbmr, 2025.

NSSF. America has a crime reporting problem, August 2023. URL https://www.nssf.org/articles/america-has-a-crime-reporting-problem/. Online article.

NVIDIA Blog Team. How ai reasoning is being tested with international math olympiad problems, 2024. URL https://blogs.nvidia.com/blog/reasoning-ai-math-olympiad/. Accessed: 2025-04-20.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Sharmila Reddy Pappula and Sathwik Rao Allam. Llms for conversational ai: Enhancing chatbots and virtual assistants. *International Journal of Research Publication and Reviews*, 2023. URL https://api.semanticscholar.org/CorpusID:266461220.

Aldo Pareja, Nikhil Shivakumar Nayak, Hao Wang, Krishnateja Killamsetty, Shivchander Sudalairaj, Wenlong Zhao, Seungwook Han, Abhishek Bhandwaldar, Guangxuan Xu, Kai Xu, Ligong Han, Luke Inglis, and Akash Srivastava. Unveiling the secret recipe: A guide for supervised fine-tuning small llms, 2024. URL https://arxiv.org/abs/2412.13337.

Zongyue Qin, Zifan He, Neha Prakriya, Jason Cong, and Yizhou Sun. Dynamic-width speculative beam decoding for efficient llm inference, 2025. URL https://arxiv.org/abs/2409.16560.

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. URL https://arxiv.org/abs/1910.10683.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. Codebleu: a method for automatic evaluation of code synthesis, 2020. URL https://arxiv.org/abs/2009.10297.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024. URL https://arxiv.org/abs/2308.12950.

Subham Sah, Rishab Mitra, Arpit Narechania, Alex Endert, John Stasko, and Wenwen Dou. Generating analytic specifications for data visualization from natural language queries using large language models, 2024. URL https://arxiv.org/abs/2408.13391.

Mokhtar Mansour Salah and KeWen Xia. Big crime data analytics and visualization. *Proceedings of the 2022 6th International Conference on Compute and Data Analysis*, 2022. URL https://api.semanticscholar.org/CorpusID:248990234.

David Schottlander and Tomer Shekel. Geospatial reasoning: Unlocking insights with generative ai and multiple foundation models. Google Research Blog, April 2025. Available at: https://research.google/blog/geospatial-reasoning-unlocking-insights-with-generative-ai-and-multiple-foundation-mode (Accessed: 2025-04-18).

ByteDance Seed, Yuyu Zhang, Jing Su, Yifan Sun, Chenguang Xi, Xia Xiao, Shen Zheng, Anxiang Zhang, Kaibo Liu, Daoguang Zan, Tao Sun, Jinhua Zhu, Shulin Xin, Dong Huang, Yetao Bai, Lixin Dong, Chao Li, Jianchong Chen, Hanzhi Zhou, Yifan Huang, Guanghan Ning, Xierui Song, Jiaze Chen, Siyao Liu, Kai Shen, Liang Xiang, and Yonghui Wu. Seed-coder: Let the code model curate data for itself, 2025. URL https://arxiv.org/abs/2506.03524.

Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 365–377, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341899. doi: 10.1145/2984511.2984588. URL https://doi.org/10.1145/2984511.2984588.

Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. A thorough examination of decoding methods in the era of llms, 2024. URL https://arxiv.org/abs/2402.06925.

Eric Shi. geojson-map-china: Geojson data of china's administrative divisions. https://github.com/longwosion/geojson-map-china, 2025. URL https://github.com/longwosion/geojson-map-china. Accessed: 2025-06-14.

Luiz José Schirmer Silva, Sonia Fiol-González, Cassio F. P. Almeida, Simone Diniz Junqueira Barbosa, and Hélio Côrtes Vieira Lopes. Crimevis: An interactive visualization system for analyzing crime data in the state of rio de janeiro. In *International Conference on Enterprise Information Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:46819581.

A. Temsah, K. Alhasan, I. Altamimi, A. Jamal, and A. Al-Eyadhy. Deepseek in healthcare: Revealing opportunities and steering challenges of a new open-source artificial intelligence frontier. *Cureus*, 17(2):e341667, 2025. doi: 10.7759/cureus.341667. URL https://www.cureus.com/articles/341667-deepseek-in-healthcare-revealing-opportunities-and-steering-challenges-of-a-new-pdf.

Sopiko Tevdoradze Sopiko Tevdoradze, Zurab Mushkudiani Zurab Mushkudiani, and Nugzar Tevdoradze Nugzar Tevdoradze. The effect of criminal activity on tourism. *The New Economist*, 2024. URL https://api.semanticscholar.org/CorpusID:273525446.

Uber Engineering Team. Querygpt: Natural language to sql using generative ai. *Uber Blog*, September 2024. URL https://www.uber.com/blog/query-gpt/. Accessed: June 24, 2025.

Unsloth Team. Datasets guide - unsloth documentation, 2024a. URL https://docs.unsloth.ai/basics/datasets-guide#how-big-should-my-dataset-be. Accessed: 2025-04-25.

Unsloth Team. What model should i use? - unsloth documentation, 2024b. URL https://docs.unsloth.ai/get-started/beginner-start-here/what-model-should-i-use. Accessed: 2025-04-25.

John E. Vargas-Munoz, Shivangi Srivastava, Devis Tuia, and Alexandre X. Falcão. Openstreetmap: Challenges and opportunities in machine learning and remote sensing. *IEEE Geoscience and Remote Sensing Magazine*, 9(1):184–199, 2021. doi: 10.1109/MGRS.2020.2994107.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey, 2023. URL https://arxiv.org/abs/2307.12966.

Zengli Wang and Hong Zhang. Construction, detection, and interpretation of crime patterns over space and time. *ISPRS International Journal of Geo-Information*, 9(6), 2020. ISSN 2220-9964. doi: 10.3390/ijgi9060339. URL https://www.mdpi.com/2220-9964/9/6/339.

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. URL https://arxiv.org/abs/2109.01652.

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. URL https://arxiv.org/abs/2201.11903.

Ijibadejo Oluwasegun William. Geo-spatial analysis with large language model. *International Journal of Advanced Natural Sciences and Engineering Researches*, 9(1):1–9, Jan. 2025. URL https://as-proceeding.com/index.php/ijanser/article/view/2409.

Diyi Yang, Sherry Tongshuang Wu, and Marti A. Hearst. Human-ai interaction in the age of llms. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 5: Tutorial Abstracts)*, 2024. URL https://api.semanticscholar.org/CorpusID:270514463.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Zhewei Yao, Guoheng Sun, Lukasz Borchmann, Zheyu Shen, Minghang Deng, Bohan Zhai, Hao Zhang, Ang Li, and Yuxiong He. Arctic-text2sql-r1: Simple rewards, strong reasoning in text-to-sql, 2025. URL https://arxiv.org/abs/2505.20315.

Shuo Yin, Weihao You, Zhilong Ji, Guoqiang Zhong, and Jinfeng Bai. Mumath-code: Combining tool-use large language models with multi-perspective data augmentation for mathematical reasoning, 2024. URL https://arxiv.org/abs/2405.07551.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019. URL https://arxiv.org/abs/1809.08887.

Germain Garcia Zanabria, Erick Gomez Nieto, Jaqueline Silveira, Jorge Poco, Marcelo Batista Nery, Sergio Adorno, and Luis Gustavo Nonato. Mirante: A visualization tool for analyzing urban crimes. *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 148–155, 2020. URL https://api.semanticscholar.org/CorpusID:227221852.

Dan Zhang, Sining Zhoubian, Min Cai, Fengzu Li, Lekang Yang, Wei Wang, Tianjiao Dong, Ziniu Hu, Jie Tang, and Yisong Yue. Datascibench: An llm agent benchmark for data science, 2025a. URL https://arxiv.org/abs/2502.13897.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.10512.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2024. URL https://arxiv.org/abs/2308.10792.

Yan Zhang, Mei-Po Kwan, and Libo Fang. An llm driven dataset on the spatiotemporal distributions of street and neighborhood crime in china. *Scientific Data*, 12(1):467, mar 2025b. ISSN 2052-4463. doi: 10.1038/s41597-025-04757-8. URL https://doi.org/10.1038/s41597-025-04757-8.

Changhai Zhou, Shiyang Zhang, Yuhua Zhou, Zekai Liu, and Shichao Weng. Automixq: Self-adjusting quantization for high performance memory-efficient fine-tuning, 2024. URL https://arxiv.org/abs/2411.13814.

APPENDIX

A.1 Dataset specification

The data consists of three interconnected geographical datasets that contain crime incidents, street networks, and administrative boundaries. These tables specify the structure, data types, and relationships between the datasets that form the foundation of the spatial and temporal analyses conducted throughout this research.

Column	**Type**	**Description**
case_type	string	Category of the crime incident (e.g., theft, assault).
latitude	float	Latitude of the incident location.
longitude	float	Longitude of the incident location.
incident_location	string	Textual description of the incident location.
incident_province	string	Province where the incident occurred.
incident_city	string	City where the incident occurred.
incident_county	string	County or district of the incident.
formatted_datetime	datetime	Standardized timestamp of the incident.
street_name	string	Name of the street where the incident occurred.
geometry	geospatial point	Geographic point of the incident.

TABLE 0.6: Dataset Description: `crimes_df` – Crime Records

Column	**Type**	**Description**
street_name	string	Official name of the street (can be linked to `crimes_df.street_name`).
geometry	line polygon	Geometric representation of the street.
incident_province	string	Province where the street is located.

TABLE 0.7: Dataset Description: `streets_df` – Street Network

Column	**Type**	**Description**
name	string	Name of the administrative region.
geometry	polygon	Boundary geometry of the region.
geom_type	string	Administrative level (e.g., province, city, county).

Table 0.8: Dataset Description: `geometries_df` – Administrative Boundaries

A.2 Prompts

The prompts shown below were carefully designed to guide the AI systems in performing various tasks, from question augmentation to code generation and evaluation. By providing these prompts in full detail, we enable others to replicate our methods and verify our findings. Additionally, these prompts serve as a reference for those seeking to build upon or extend our approach in future studies.

```
You are an AI tasked with producing {k} distinct and imaginative rewordings of a given {content_type}. Each version should blend direct questions with indirect, conversational phrasings that convey the same intentsome clearly interrogative, others more casually woven into dialogue. Vary structure and diction markedly while preserving the original meaning. The fewer paraphrases requested, the more distinct and creative each one should be, while still maintaining the original intent of the question. Respond as a JSON object containing a ’paraphrases’ key whose value is a list of exactly {k} rewritten items. Reorder arguments when necessary, since the prompts are template-generated. Keep every paraphrase brief, as if an average user were chatting with a bot. Any terms wrapped in <> must remain unchanged. The question is about a dataset of crime incidents covering the period between 2017 and 2019. As the question is generated by a template, it may contain errors when sampling the template. In that case, please fix the errors in the question and then generate the paraphrases.
```

Listing 1: Prompt for generating reworded question paraphrases

```
You are an expert crime-data instructor
I will give you a style analytical question about crime incidents in a certain country (spatiotemporal dataset with counts, rates, places, dates, etc.) covering the period between 2017 and 2019.
Your task is to create five ({k}) altered versions of that question, each applying at least one of the transformation strategies listed below (you may combine several in the same question).
Use clear, natural language; keep every new question answerable from the same dataset.  As the question is generated by a template, it may contain errors when sampling the template. In that case, please fix the errors in the question and then generate the paraphrases. Altered questions must be able to be answered with Python code (pandas, geopandas, numpy, shapely, etc).

**Any terms wrapped in <> must remain unchanged.**

Transformation strategies:

1. Change specific numbers / dates
    - Modify counts, percentages, years, or timewindows while staying realistic.
2. Introduce ratios or percentages
    - Convert absolute figures into growthrates, percentage drops, etc.
3. Add a conditional statement
    - Include an if clause that alters the scenario (e.g., If night-shift patrols double ..., Suppose ...).
4. Add temporal ranges like weekly, monthly, quarterly, weekends, weekdays, date_ranges, etc.
    - Specify the time frame for the question (e.g., "What was the crime rate in <city> during weekends in 2018?").
5. Question inversion
    - Rephrase the question to ask about the opposite (e.g., Instead of "What is the most common crime?", ask "What is the least common crime?").

Return the result as a JSON object with a single key "altered_questions" whose value is a list of the {k} strings. Additionally, tag each altered question in one of the following categories: ’spatial_lookup’, ’temporal_lookup’, ’spatio_temporal_lookup’, ’comparative_trends’, ’causal_contextual’, ’hypothetical_counterfactuals’, ’multi_step_aggregation’

{{
```

```
  "altered_questions": [
    {{
        "alt_q": "altered question 1",
        "modification_explanation": "explanation of the modification",
        "alt_q_new_category": "altered question category 1"

    }},  // Q1
    {{
        "alt_q": "altered question 2",
        "modification_explanation": "explanation of the modification",
        "alt_q_new_category": "altered question category 2"
    }},  // Q2
    ...,
    {{
        "alt_q": "altered question {k}",
        "modification_explanation": "explanation of the modification",
        "alt_q_new_category": "altered question category {k}"
    }}   // Q_{k}
  ]
}}
```

Listing 2: Prompt for generating altered questions

```
You are an expert AI judge. Your sole task is to decide whether TWO answers convey the  same information **with respect to the
      QUESTION**.
Treat them as equivalent and answer \yes\ if **every fact needed to fully answer the question is present (explicitly or implicitly)
      in both answers**, even when:
   wording, synonyms, order, formatting or level of detail differ
   one answer adds extra context that does NOT contradict the other
   numbers are rounded or expressed in different but compatible units
Answer \no\ only when at least ONE of these is true:
   a factual statement appears in one answer and is contradicted or missing in the other
   the two answers would lead a careful reader to different conclusions about the question
Evaluation procedure (think step-by-step):
  1. Restate the key requirements implied by the question.
  2. Extract the atomic facts from Output 1 and Output 2.
  3. Check mutual entailment: does each set of facts satisfy the other w.r.t. the question?
  4. Decide \yes\ or \no\.
Return **only** a JSON object with keys:
   \cot\: a brief explanation of your comparison (1-3 sentences).
   \answer\: \yes\ or \no\
```

Listing 3: System prompt for output evaluation

```
QUESTION:{question}
=== Output 1 ===
{ans}
=== Output 2 ===
{gen}
Are the two outputs semantically equivalent answers to the question?
Respond now.
```

Listing 4: User prompt for output evaluation

```
You are an expert **Python** datascience assistant.

----------------------
REFERENCE DATASETS
----------------------

### Crime dataset Overview

This is a dataset that contains information about various crimes, including their locations (street, county, city, province) and
      timestamps. The dataset is structured as a `GeoDataFrame`, which allows for spatial operations and analysis. Represented as
      `crimes_df` GeoDataFrame variable in `solve` function. The crimes have "lat lon" coordinates and are assigned to the nearest
      street.

- **Schema Columns:**
  {SCHEMA_COLS}

- **Sample Row:**
  {SCHEMA_VALUES}

- **Column Data Types:**
  {SCHEMA_TYPES}

- **Dataset dictionary:**
  {SCHEMA_DICT}

---

### Streets dataset Overview

This dataset didn't contain crime information is a `GeoDataFrame` that contains streets and their geometries. It is used to analyze
      the spatial relationship between streets and crime locations. Represented as `streets_df` GeoDataFrame variable in `solve`
      function.

- **Schema Columns:**
  {STREET_SCHEMA_COLS}

- **Sample Row:**
  {STREET_SCHEMA_VALUES}

- **Column Data Types:**
  {STREET_SCHEMA_TYPES}

- **Dataset dictionary:**
  {STREET_SCHEMA_DICT}

---

### Geometries dataset Overview (provinces, cities and counties)

This dataset contains polygons representing the geometries of provinces, cities, and counties. It is used to analyze the spatial
      relationship between crime locations and administrative boundaries. Represented as `geometries_df` GeoDataFrame variable in
      `solve` function.

- **Schema Columns:**
  {GEO_SCHEMA_COLS}

- **Sample Row:**
```

~~~~
  {GEO_SCHEMA_VALUES}

- **Column Data Types:**
  {GEO_SCHEMA_TYPES}



----------------------
REFERENCE MATERIAL
----------------------
 **Reference question:** {reference_question}



 **Reference code:**

```python
{reference_code}
```

 **My question:** {my_question}

----------------------
YOUR TASK
----------------------

// Depending on the type of question, different rules apply: if its a rephrased version of the original, one set of rules is used; if
      its a modified or altered version, a different set of rules comes into play.

----------------------
OUTPUT FORMAT
----------------------
<think>
</think>

<tool_call>
</tool_call>
~~~~

Listing 5: Prompt for synthetic code generation

```
You are a highly skilled programming assistant specialized in **geospatial data analysis** and **Python development**.
Your role is to write Python code that analyzes geospatial data to answer user questions. You have in-depth knowledge of:

* Geospatial libraries like GeoPandas and Shapely for spatial data manipulation
* Crime data exploration and recognizing geographic patterns
* Spatial relationships, geometric operations, and administrative boundary joins
* Temporal analysis (e.g. moving averages, seasonal trends, time-series decomposition)
* Writing clean, efficient, and well-documented Python code with proper error handling

You have access to **three** interconnected GeoDataFrames: `crimes_df`, `streets_df`, and `geometries_df`. These contain crime
      incidents, street networks, and administrative boundaries, respectively.

## Dataset Descriptions

### `crimes_df`  Crime Records (Points)

Each row is a crime incident with location and time information:

* **case_type** (object): Category of the crime incident (e.g., theft, assault)
```

```
* **latitude** (float64): Latitude of the incident location
* **longitude** (float64): Longitude of the incident location
* **incident_location** (object): Detailed textual description of the location
* **incident_province** (object): Province where the incident occurred (administrative level)
* **incident_city** (object): City where the incident occurred
* **incident_county** (object): County where the incident occurred
* **formatted_datetime** (datetime64[ns]): Timestamp of the incident (standardized format)
* **street_name** (object): Name of the street where the incident occurred
* **geometry** (geometry, Point): Geographic point location of the incident (Shapely Point)

### ‘streets_df‘  Street Network (Lines)

Street geometries and their associated province:

* **street_name** (object): Official name of the street (this can link to ‘crimes_df.street_name‘)
* **geometry** (geometry, LineString): Geometric representation of the street (Shapely LineString)
* **incident_province** (object): Province in which the street lies (matches with administrative regions)

### ‘geometries_df‘  Administrative Boundaries (Polygons)

Polygon boundaries for regions at different administrative levels:

* **name** (object): Name of the administrative region
* **geometry** (geometry, Polygon/MultiPolygon): The boundary geometry of the region
* **geom_type** (object): The administrative level of the region (e.g., "province", "city", "county")

**Relationships:** The data is related by location. For example, ‘crimes_df‘ points lie on ‘streets_df‘ geometries (matching by ‘street_name‘), and each record is located within one of the polygons in ‘geometries_df‘ (matching by name and ‘geom_type‘ like province/city/county).

## INSTRUCTION

Using the above datasets, **write Python code** to answer the following question: **"{question}"**. Your solution should analyze the GeoDataFrames to produce the answer, utilizing spatial and temporal logic as needed.

## CRITICAL REQUIREMENTS

* **Function Definition:** Implement your solution inside a function ‘def solve(crimes_df: gpd.GeoDataFrame, streets_df: gpd.GeoDataFrame, geometries_df: gpd.GeoDataFrame) -> None:‘. This function will be the entry point for execution.
* **Imports and Setup:** Begin with all necessary ‘import‘ statements. Only the following libraries are allowed for use: **pandas**, **numpy**, **geopandas**, **shapely**, and **datetime**. **Do not import or use any other libraries outside of these.** Include any setup like CRS alignment if needed.
* **Data Integrity Checks:** Before analysis, perform sanity checks (e.g., handle null or missing geometries, ensure all GeoDataFrames use the same Coordinate Reference System (CRS) to avoid mismatches, and confirm that merges/joins will not fail on empty results). If there are CRS differences, fix them by re-projecting to a common CRS.
* **Question Parsing:** If the question contains specific names, locations, or placeholders (for example, a city name or ‘<CITY>‘ in angle brackets), **extract those entities** and use them to filter or query the data appropriately. This means your code should dynamically handle any ‘<...>‘ placeholders by treating them as query parameters.
* **Spatial Analysis:** Utilize geospatial operations to answer location-based parts of the question. This might include spatial joins (e.g., assigning crimes to a region polygon or matching crimes to street segments), geometric calculations (distances, containment, intersections), or aggregating data by regions. Use GeoPandas/Shapely functions to accomplish these tasks.
* **Temporal Analysis:** If the question involves time (dates, seasons, trends), incorporate time-based analysis. For example, you may need to filter by date range, compute moving averages over time, or examine seasonal trends in incident counts. Use pandas’ datetime utilities (and other allowed libraries if necessary) for time-series calculations. **Do not use any libraries outside the allowed set.**
* **Comments and Clarity:** Explain each major step of your solution with **clear, concise English comments**. This includes describing any filtering, grouping, spatial joins, calculations, or anomaly handling in the code. The goal is to make the code easy to follow for others.
```

```
* **Error Handling:** Include try/except blocks or conditional checks to handle potential errors gracefully. For instance, handle the
      case where the question’s criteria yield no results (perhaps by printing a message like "No data found for X"). Also guard
      against indexing errors or key errors if an expected column or name is not found. **Pay special attention to ZeroDivisionError
      when performing calculations (such as averages, ratios, or percentages) and IndexError when accessing elements from
      potentially empty lists or DataFrames. Always check if denominators are zero before division and verify list/DataFrame lengths
      before indexing.**
* **No File Output:** Do not save any files or create any file outputs. All results and analysis should be displayed using **print
      statements only**.
* **Final Output:** The function should **print** the answer (whether its a number, text, or JSON) as the final result of the
      analysis. Do not ‘return‘ anything from ‘solve()‘ (it should return None). Make sure the printed answer directly addresses the
      question asked.

## OUTPUT FORMAT

Your response **must** be a single Python code block that meets these criteria:

* The python codeblock must always start with ‘python and finish with ‘
* Begins with all necessary ‘import‘ statements and any initial setup (no need for extraneous comments before imports).
* Contains the full ‘solve()‘ function implementation, including the signature provided and ending with the ‘return None‘.
* **No additional text or explanation outside the code**  *only the code block* should be present in the output. Do not include
      example usages, ‘if __name__ == "__main__":‘ guards, or any printouts except for the final answer.

## EXECUTION DIRECTIVE

**Begin coding immediately** following the above requirements. **Do not provide any explanation or conversation** in your answer
      only the Python code within the single code block. The users question will be inserted where ‘"{question}"‘ is, and your job
      is to output the code solving that specific query. Remain focused on that task and ensure your code runs without errors.

**Now, write the Python code solution below:**

‘‘‘python
import pandas as pd
import numpy as np
import geopandas as gpd
from datetime import datetime
import shapely.geometry
# Allowed libraries: pandas, numpy, geopandas, shapely, datetime (use only these).

def solve(crimes_df: gpd.GeoDataFrame, streets_df: gpd.GeoDataFrame, geometries_df: gpd.GeoDataFrame) -> None:
    # Complete this function to answer the question using the three GeoDataFrames
    # 1. Parse and identify any specific entities or locations from the question
    # 2. Perform spatial joins or filtering on crimes_df, streets_df, geometries_df as needed
    # 3. Perform temporal analysis if time-based trends are relevant to the question
    # 4. Aggregate or compute the result based on the question (e.g., count of crimes, trend over time, etc.)
    # 5. Handle edge cases (no data, null values, etc.) and ensure CRS consistency for spatial operations
    # 6. Print out the final answer clearly. Make sure it’s understandable and directly answers the question.

    # --- Your code implementation begins here ---



    # --- Your code implementation ends here ---

    # Print the final result (already done within the steps above as needed)
    return None
‘‘‘
```

Listing 6: Training prompt for code generation