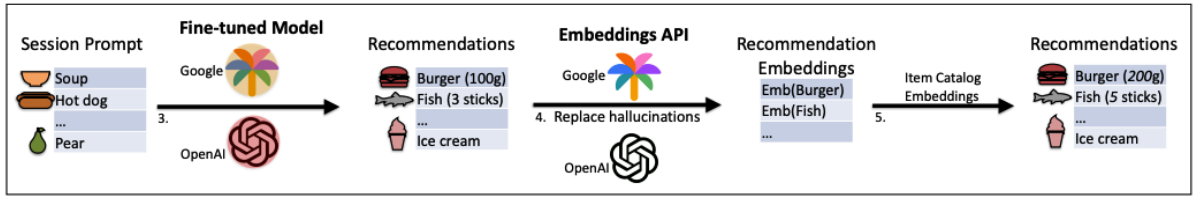
Building Efficient LLM Pipeline for Human Mobility Prediction

Chenhao Wang, Silin Zhou, Lisi Chen, Shuo Shang
University of Electronic Science and Technology of China

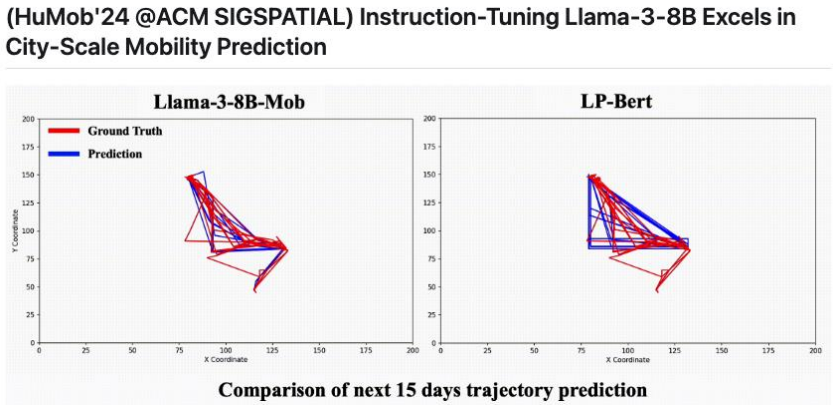
GISCUP Workshop
Nov 3rd, 2025

Motivation

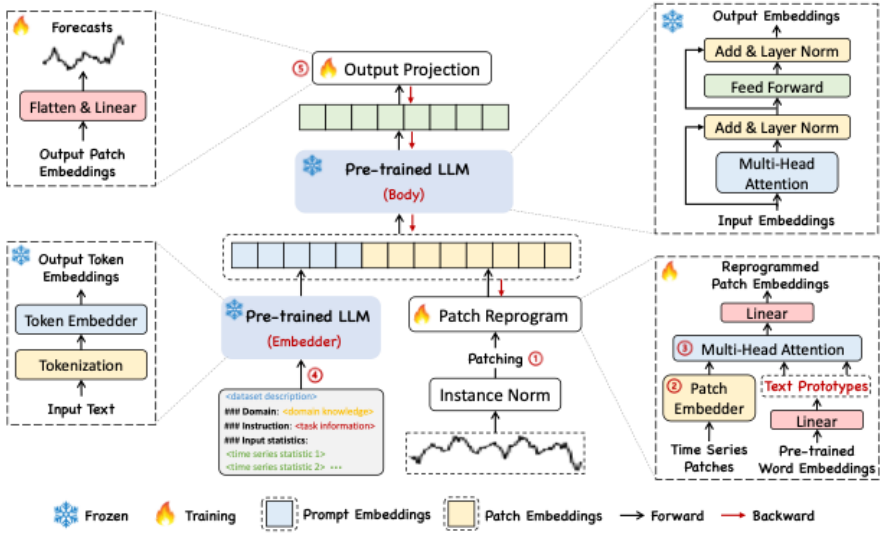
Large Language Models (LLMs) → Strong Sequence Reasoning Capabilities



Sequential Recommendation [1]



Human Mobility Prediction [3]



Time Series Forecasting [2]

[1] Improving Sequential Recommendations with LLMs. ACM Transactions on Recommender Systems, 2024.
[2] Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. ICLR, 2024.
[3] Instruction-Tuning Llama-3-8B Excels in City-Scale Mobility Prediction. HuMob, 2024.

Motivation

Challenges of Using LLMs for Human Mobility Prediction

Representation & Personalization

- Reformulate mobility trajectories into **textual sequences** that preserve **spatio-temporal dependencies** and **user-specific priors** for accurate prediction.

Low Fine-tuning Efficiency

- Fine-tuning LLMs on **large-scale mobility data** is **computation-intensive** and often takes days or weeks.

Long trajectories → Token & Memory Limits

- Some users have over **2,000 records** in 60 days (GISCUP 2025), easily **exceeding context length** and **GPU memory capacity**.

Goal

Build an **efficient LLM pipeline** for human mobility prediction that:

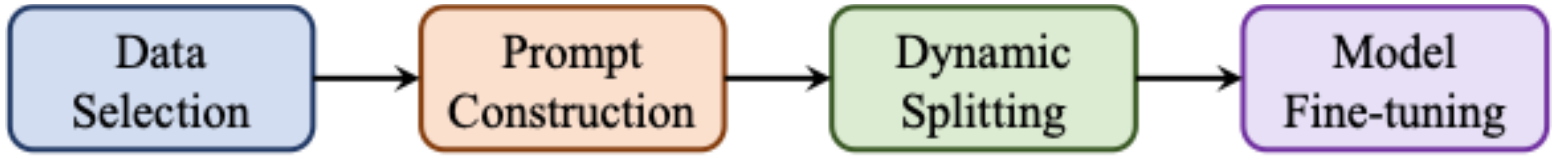
- Combine **user profiles** and **spatio-temporal features** in prompts
- Achieve strong performance with **limited training data**
- Handle long trajectories efficiently under **limited context length**

We propose **ELP-Mob (Efficient LLM Pipeline for Human Mobility Prediction)**

Overview

Four-Stage Pipeline Design

- **Data Selection:** Identify and retain the most informative users
- **Prompt Construction:** Convert mobility data into instruction-style text
- **Dynamic Splitting:** Control token length for long historical trajectories
- **Model Fine-Tuning:** Fine-Tuning LLM with LoRA



Data Selection Strategy

- **Not all users contribute to model learning**
- Compute **spatial overlap score** between training and test users:

$$\text{Score}(u_{\text{train}}) = \frac{1}{|\mathcal{U}^{\text{test}}|} \sum_{u_{\text{test}} \in \mathcal{U}^{\text{test}}} \frac{|G(u_{\text{train}}) \cap G(u_{\text{test}})|}{|G(u_{\text{test}})|},$$

$G(u)$ is the set of visited grid cells of user u

- Select **top-K users** with the highest scores for fine-tuning

Reduce irrelevant information and data redundancy

Prompt Construction

Instruction Block

Defines model role, environment, trajectory format, prediction task, and output format

Input Block

Includes user profile, historical trajectory, and future time slots

Output Block

Provides predicted locations matching the input format

Instruction

[Role]
You are a human mobility prediction assistant.
[Environment]
The city is divided into a 200 × 200 grid, each cell representing a 500m × 500m area. The top-left corner is (1,1); the bottom-right is (200,200). Time is discretized into 30-minute intervals, giving 48 time slots per day.
[Trajectory Format]
Each record is formatted as: <day_id> <timeslot_id> <x> <y>. For example: `12 16 103 88` means on day 12, at time slot 16 (7:30am–8:00am), the person was at cell (103,88).
[Task]
You will receive: 1. [User Profile]: Top-K most frequently visited locations with their proportions. 2. [Trajectory History]: Known locations from day 1 to day 60. 3. [Future Time Slots]: From day 61 to day 75, with missing locations (represented as `999 999`).
Your task: Predict the missing locations for all [Future Time Slots], leveraging both the [User Profile] and the [Trajectory History].
[Output]
Return a list of records `<day_id> <timeslot_id> <x> <y>`. Predictions must correspond exactly to the missing entries in [Future Time Slots]. Maintain the same order as they appear in [Future Time Slots].

Input

[User Profile]
TOPK=5: (112,86)@12.02%,(86,41)@11.24%,(111,86)@5.04%,(87,40)@3.1%,(87,41)@2.33%
[Trajectory History]
1 35 106 69\n1 36 107 70\n1 38 112 86\n ... \n60 40 106 55\n60 41 106 56\n60 42 112 86
[Future Time Slots]
61 15 999 999\n61 16 999 999\n61 17 999 999\n ... \n75 31 999 999\n75 35 999 999\n75 36 999 999

Output

[Prediction]
61 15 102 73\n61 16 90 56\n61 17 86 42\n ... \n75 31 127 79\n75 35 124 86\n75 36 112 86

Provides LLM with personalized and structured context

Dynamic Splitting Strategy

- **Dynamic splitting** sequences while preserving temporal coverage
- Enables **parallel processing** and **token-length control**
- Reorganizes prediction results in **temporal order** during inference

Algorithm 1: Dynamic Splitting Strategy

Input: Historical trajectory S^H, future sequence S^F, cutoff length $L_{\max}$, maximum partitions $N_{\max}$

Output: Set of valid prompts $\mathcal{P}$

for $n \leftarrow 1$ **to** $N_{\max}$ **do**
  Partition S^H and S^F into n interleaved subsets $\{S_i^H\}, \{S_i^F\}$ by interval sampling;
  $\mathcal{P} \leftarrow \{(S_i^H, S_i^F)\}_{i=1}^{n}$;
  if $\max_i length(P_i) \leq L_{\max}$ **then**
    return $\mathcal{P}$;
return $\emptyset$;

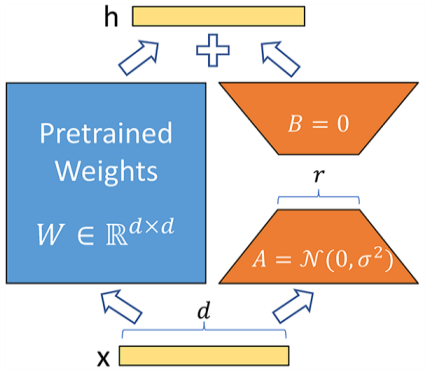
Preserves sequential accuracy while improving token efficiency

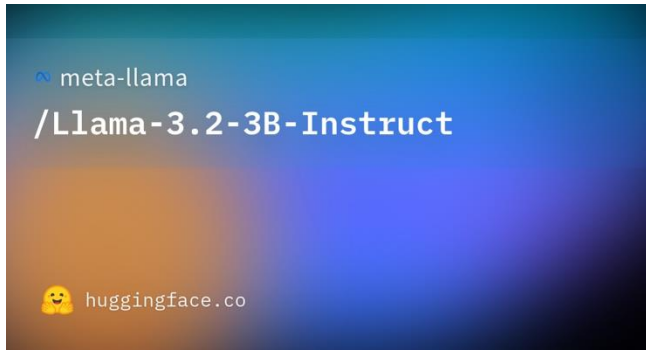
Model Fine-Tuning

Finetune Strategy: Use **LoRA**[1] for parameter-efficient fine-tuning

Backbone: **Llama-3.2-3B-Instruct**[2] (any open-source LLM can be applied)

Framework: **LLaMA-Factory**[3]





LLaMA-Factory
Easy and Efficient LLM Fine-Tuning

[1]Lora: Low-rank adaptation of large language models. ICLR, 2022
[2]https://github.com/hiyouga/LLaMA-Factory
[3]https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Experiments

Original Datasets

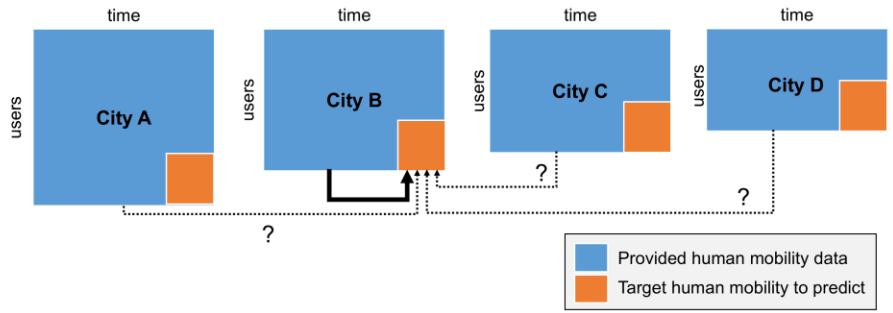


Figure from https://sigspatial2025.sigspatial.org/giscup/dataset.html

- Cities A, B, C, and D contain **150,000**, **30,000**, **25,000**, and **20,000** individuals, respectively
- Among them, **3,000** individuals in each city are used for mobility prediction

Preprocessed Datasets

- **Training Set: 15,000 users** in each dataset are selected using the **data selection strategy**
- **Validation Set: 500 users** randomly sampled from each dataset

Table 1: Dataset Statistics.

Dataset	A	B	C	D
# Users	150,000	30,000	25,000	20,000
# Train users	15,000	15,000	15,000	15,000
# Validate users	500	500	500	500
# Test users	3,000	3,000	3,000	3,000
# Train prompts	40,889	37,670	27,645	28,619
# Validate prompts	1,153	1,196	795	818
# Test prompts	6,589	7,149	4,341	4,597

Experiments

Effectiveness

Table 2: Performance comparison.

Methods	A	B	C	D	Mean
Per-User Mode	0.07984	0.08116	0.10789	0.10296	0.09296
Bigram Model	0.04687	0.05492	0.06249	0.06212	0.05660
Bigram Model (top-p 0.7)	0.09384	0.09232	0.07039	0.07997	0.08413
ELP-Mob	**0.13719**	**0.13286**	**0.16767**	**0.16237**	**0.15002**

- Baseline results from https://sigspatial2025.sigspatial.org/giscup/problem.html

ELP-Mob consistently outperforms all official baselines across datasets A–D

Efficiency

Table 3: Efficiency comparison on city A.

Methods	GPU Mem. (Training)	Training Time	Inference Time
Llama-3-8B-Mob	23.93 GiB	74.68 h (estimated)	10.84 s/user
ELP-Mob	14.81 GiB	6.27 h	2.52 s/user

- Llama-3-8B-Mob is a leading method in GISCUP 2024
- Code available at https://github.com/TANGHULU6/Llama3-8B-Mob
- We use the same number of training users for a fair comparison

ELP-Mob lowers GPU memory consumption and reduces training and inference time

Experiments

Experimental Environment

- Experiments conducted on a server with 4× NVIDIA GeForce RTX 4090 GPUs
- For 3B LLM backbones, 16 GB GPU memory is sufficient to reproduce all results
- For 8B LLM backbones, 24 GB GPU memory is sufficient

Hyperparameters

Fine-tuning	Cutoff Length	4096
	LoRA Rank	16
	Per-device Train Batch Size	1
	Gradient Accumulation Step	8
	Learning Rate	0.0005
	Scheduler	cosine
	Warmup Ratio	0.05
	Epoch	3.0
Inference	Per-device Eval Batch Size	4
	Temperature	0.1
	Top p	1
	Top k	200

Thank you!

Code: https://github.com/chwang0721/ELP-Mob

Contact: chenhao.wang@std.uestc.edu.cn