CompassLLM: A Multi-Agent Approach toward Geo-Spatial Reasoning for Popular Path Query

Md. Nazmul Islam Ananto[1] **Shamit Fatin**[1,2] **Mohammed Eunus Ali**[3]
Md Rizwan Parvez[4]
[1]Bangladesh University of Engineering and Technology (BUET)
[2]University of Utah [3]Monash University [4]Qatar Computing Research Institute (QCRI)
nazmulislamananto@gmail.com , shamit.f@utah.edu ,
eunus.ali@monash.edu , mparvez@hbku.edu.qa

Abstract

The popular path query—identifying the most frequented routes between locations from historical trajectory data—has important applications in urban planning, navigation optimization, and travel recommendations. While traditional algorithms and machine learning approaches have achieved success in this domain, they typically require model training, parameter tuning, and retraining when accommodating data updates. As Large Language Models (LLMs) demonstrate increasing capabilities in spatial and graph-based reasoning, there is growing interest in exploring how these models can be applied to geo-spatial problems.

We introduce **CompassLLM**, a novel multi-agent framework that intelligently leverages the reasoning capabilities of LLMs into the geo-spatial domain to solve the popular path query. **CompassLLM** employs its agents in a two-stage pipeline: the *SEARCH* stage that identifies popular paths, and a *GENERATE* stage that synthesizes novel paths in the absence of an existing one, a common problem in sparse historical data. Experiments on real and synthetic datasets show that **CompassLLM** demonstrates superior accuracy in *SEARCH* and competitive performance in *GENERATE* both while being cost-effective.

1 Introduction

Recent advancements in Large Language Models (LLMs) have demonstrated remarkable versatility, excelling not only in natural language tasks but increasingly in complex reasoning problems across diverse domains (Laskar et al., 2024). Their zero-shot and in-context learning capabilities enable adaptation to new problem spaces without system modifications, making them attractive candidates for exploring challenging computational problems. As LLMs continue to show promise in different geo-spatial reasoning tasks and map data understanding (Dihan et al., 2025; Hasan et al., 2025; Chai et al., 2023; Roberts et al., 2023; Balsebre et al., 2024; Xie et al., 2024; Deng et al., 2025), there is growing interest in understanding how these capabilities can be harnessed for navigation and route recommendation tasks.

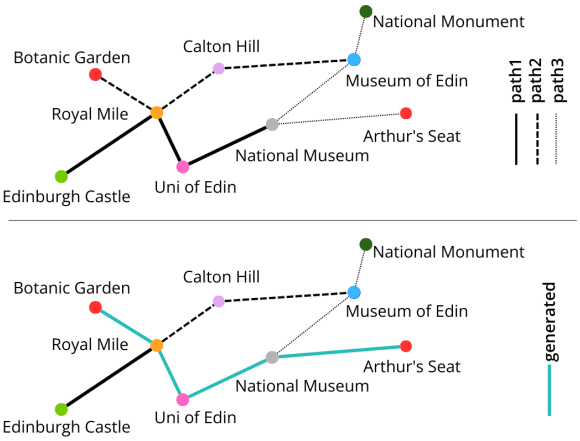

Figure 1: When *SEARCH*ing for the most popular path from **Botanic Garden** to **Arthur's Seat**, given the three trajectories (top), we can see that there is no single path spanning the entirety of these two. This incurs a *GENERATE* problem (bottom) where one must break down the trajectories into smaller segments and combine them to form a new path.

One compelling route recommendation task is the popular path query problem—mapping the most frequented routes between key locations (i.e. source-destination pair) from historical trajectory data (i.e. previously traveled paths by people). This problem has significant applications in urban planning, navigation optimization, robotics, autonomous vehicle operations, and personalized (and non personalized) travel recommendations (Zheng et al., 2011; Li et al., 2010; Yuan et al., 2011, 2010; Giannotti et al., 2007).

The problem becomes particularly interesting when considering path synthesis scenarios – cases where no direct path exists in the historical trajectory data. In sparse trajectory dataset like Figure

Findings of the Association for Computational Linguistics: ACL 2026, pages 31960–31976

1, where only 25% of all possible edges exist (only 9 edges out of $9C2 = 36$ possible edges for 9 nodes) in the dataset, generating novel path that is valid and traversable becomes a challenge. Generating valid paths is specifically crucial in this era of autonomous vehicles and other physical systems where LLMs' reasoning are being adopted (Latif, 2024; Xiao and Yamasaki, 2025; Doma et al., 2024).

Traditional algorithmic approaches, including heuristic and approximation methods (Chen et al., 2011; Luo et al., 2013; Wei et al., 2012), have achieved reasonable success in this domain. Machine learning approaches have also made progress, with earlier methods focusing on existing paths in historical data (Rashid et al., 2023; Yang and Gidofalvi, 2018; Wang et al., 2019; Tian et al., 2023; Shi et al., 2024), and more recent neural network models like NeuroMLR (Jain et al., 2021) and GEIT (Zhang et al., 2023) capable of generating paths on graph. While these approaches are effective, they typically require model training, parameter tuning, and retraining when accommodating new constraints or data updates. Due to the lack of generalization in these approaches, LLM-based methods are getting increasing attention for their adaptability. LLMs' ability to process complex, long-context inputs and reason about relationships in general makes them intriguing candidates for spatial problems.

However, applying LLMs to geo-spatial reasoning presents unique challenges. While Chain-of-Thought (CoT) prompting (Wei et al., 2022) and its extensions like Self-Consistency (Wang et al., 2023), ReAct (Yao et al., 2023b), and Reflexion (Shinn et al., 2023) have shown success in various reasoning tasks, they are not crafted for geo-spatial tasks—particularly the popular path query—where LLMs may generate invalid or untraversable paths (Aghzal et al., 2023) without proper guidance and direction. Recent efforts like LLM-A* (Meng et al., 2024a) have combined algorithmic approaches with LLMs but focus on general pathfinding rather than popular path queries. Later, multiple researches including (Li et al., 2024) and LLMAP (Yuan et al., 2025) utilizes LLM to parse user requirements and other parameters from the input. Most recently, PathGPT (Marcelyn et al., 2025) explores path recommendation with retrival augmentation to embed only relevant part of the dataset with prompts.

To explore LLMs' potential in this domain, we introduce **CompassLLM** (Coordinated Orchestration of Multi-agent Path Analysis and Spatial Synthesis), a multi-agent framework designed specifically for popular path queries. Our approach features a novel two-stage pipeline orchestrating our agents: a *SEARCH* stage that finds popular paths from historical trajectories, and a *GENERATE* stage that synthesizes new paths when direct routes don't exist. **CompassLLM** leverages LLMs' understanding and reasoning capabilities of maps/graph structures.

Our evaluation uses both real-world trajectory data from user movement tracking (Chen et al., 2016a; Lim et al., 2018) and carefully designed synthetic datasets that capture diverse spatial configurations. Results demonstrate that **CompassLLM** outperforms other methods in *SEARCH* problem and achieves competitive performance for the *GENERATE* problem with SOTA models. All while offering the advantages of training-free inference and cost-effective token usage—particularly excelling in scenarios with sparse historical data.

Our paper makes the following contributions:

- We propose **CompassLLM**, a novel multi-agent framework that explores how LLMs can be effectively applied to geo-spatial domain for popular path queries.
- We incorporate path finding and synthesis in an orchestrated two-stage pipeline to recommend both popular and valid paths even in the cases of no historical ground truth.
- We provide comprehensive experimental evaluation demonstrating **CompassLLM**'s competitive performance against existing methods across real-world and synthetic datasets, while highlighting the unique advantages of LLM-based approaches for spatial reasoning tasks.

2 Related Works

The Popular Path Query in the form of Next POI Prediction or Route Recommendation has been a central research topic, driven by the increasing availability of trajectory data and the growing demand for personalized navigation services. Existing approaches can be broadly classified into three categories: algorithmic methods, machine learning models, and, more recently, LLM-based approaches.

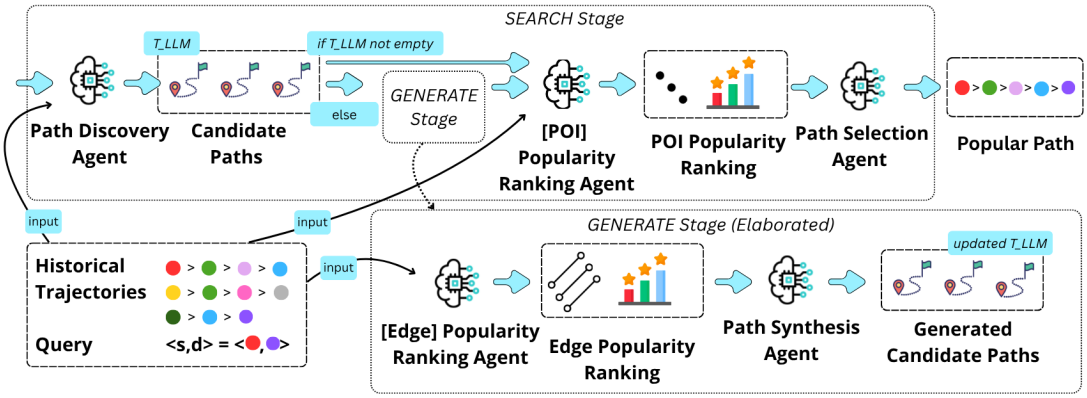

Figure 2: Overview of **CompassLLM** framework.

Algorithmic Approaches

Early research in this field predominantly employed probabilistic and heuristic methods (Banerjee et al., 2014; Chen et al., 2011; Luo et al., 2013; Wei et al., 2012; Wu et al., 2016). However, these methods often require significant manual effort to handle corner cases and when new constraints are introduced.

Machine Learning Approach

In recent years, machine learning has become the primary focus in route recommendation techniques (Wu et al., 2017; Li et al., 2020; Zheng et al., 2012). As they initially lacked path generation capabilities, deep generative models (Jain et al., 2021; Wu et al., 2017) have since addressed this limitation. However, machine learning models remain inherently dependent on the data they are trained on, requiring time-consuming retraining. In scenarios like route recommendation, where data may change frequently due to environmental, natural, or economic factors, retraining becomes a significant bottleneck.

LLM-based Approaches

Large Language Models have emerged as a revolutionary tool – techniques such as prompt engineering (Wei et al., 2023; Wang et al., 2023; Yao et al., 2023a,b; Shinn et al., 2023), fine-tuning (Himoro and Pareja-Lora, 2022), and prompt-tuning (Zhou et al., 2022) have opened new avenues for LLMs to be applied in diverse fields. In particular, LLMs have demonstrated potential in areas such as graph reasoning (Chen et al., 2024; Ye et al., 2023; Wang et al., 2024) and task planning (Song et al., 2023; Gundawar et al., 2024; Sharan et al., 2023). Nevertheless, challenges remain in ensuring that LLM-generated routes conform to real-world road networks (Kambhampati et al., 2024). Key issues include the inherent inconsistency of LLM outputs (Jang et al., 2022) and difficulties in parsing complex input structures (Tam et al., 2024). Despite these obstacles, the potential for LLMs to enhance the quality of route recommendations (Meng et al., 2024a) and incorporate user satisfaction is considerable, indicating a promising direction for research (Li et al., 2024; Yuan et al., 2025; Marcelyn et al., 2025).

3 Problem Definition

Let $\mathcal{V} = \{p_1, p_2, \ldots, p_N\}$ be a set of N POIs in a city (e.g., Edinburgh Castle, Calton Hill, National Monument). The historical trajectories, denoted as $\mathcal{T} = \{r_1, r_2, \ldots, r_M\}$, represent a collection of M routes that people have taken while visiting the city in the past. Each route $r \in \mathcal{T}$ is an ordered sequence of POIs i.e. $r = (p_1, p_2, \ldots, p_J)$, where each (p_j, p_{j+1}) is a road segment between two POIs. Moreover, $\mathcal{E}_\mathcal{T} = \{(p_j, p_{j+1}) : p_j, p_{j+1} \in r; \forall r \in \mathcal{T}\}$ denotes the set of all road segments present in $\mathcal{T}$. As the true map is not available in the dataset, we construct the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_\mathcal{T})$ from only the historical trajectories $\mathcal{T}$.

Given these historical trajectories $\mathcal{T}$ and a query $q : \langle s, d \rangle$ as input, our aim is to find the single most popular path $\mathcal{R} = (q_1, q_2, \ldots, q_K)$. This path should satisfy the conditions that $q_1 = s$, $q_K = d$ and $\forall k : q_k \in \mathcal{V}$.

The ground truth trajectories $\mathcal{T}_{GT} = \{r_1\prime, r_2\prime, \ldots\}$ for a given query $q : \langle s, d \rangle$ is the list of unique paths or sub-paths, $r\prime$, extracted from each path $r \in \mathcal{T}$ such that $r_\prime = (s, \ldots, d)$. The popularity of each path r' is defined as the cumulative popularity of its POIs: $\text{Popularity}(r') = \sum_{k=1}^{K} pop(q_k)$

Thus, the most popular path—defined here as the path maximizing cumulative POI popularity

among candidates in $\mathcal{T}_{GT}(s,d)$—is obtained by:

$$\mathcal{R} = \arg \max_{r' \in \mathcal{T}_{\mathcal{G}\mathcal{T}}(s,d)} \sum_{q_k \in r'} pop(q_k)$$

This POI-additive formulation is distinct from exact route-frequency maximization as our objective ranks candidate paths (T_{LLM} in Figure 2, elaborated in Section 4) by the cumulative popularity of their constituent POIs, following prior literature (Rashid et al., 2023).

On the other hand, if $\mathcal{T}_{GT} = \emptyset$, there is no path satisfying the given query, let alone a popular one. In such cases, our goal is to generate a popular path which is traversable in the graph $\mathcal{G}$. We consider a route $\mathcal{R}$ to be fully traversable if its constituent edges belong to $\mathcal{E}_{\mathcal{T}}$.

4 Methodology

We propose a multi-agent framework (Islam et al., 2024; Hong et al., 2023) for popular path queries that orchestrates specialized agents across a two-stage pipeline: *SEARCH* and *GENERATE*. Our approach coordinates four key agents—*Path Discovery*, *Popularity Ranking*, *Path Synthesis*, and *Path Selection*—each designed to handle distinct aspects of the popular path finding process.

Figure 2 shows an overview of the framework. The orchestration starts with the *SEARCH* stage. The *Path Discovery Agent* first retrieves candidate paths from historical trajectories that satisfy the query $\langle s, d \rangle$. When candidate paths are available, the framework remains in the *SEARCH* stage and proceeds directly to path evaluation. However, when no suitable paths exist in the historical data, the framework enters the *GENERATE* stage.

In the *GENERATE* stage, the *Popularity Ranking Agent* operates in edge mode, computing popularity scores for individual edges based on historical traversal patterns. These rankings guide the *Path Synthesis Agent* in constructing new candidate paths that connect the source and destination, and these candidates are then returned to the *SEARCH* stage. Thus, the *GENERATE* stage is conditionally executed within the *SEARCH* stage, and our orchestration begins and ends with *SEARCH*.

Upon returning to (or remaining in) the *SEARCH* stage with candidate paths—whether discovered or generated—the *Popularity Ranking Agent* is invoked in POI mode to rank locations by their popularity. Finally, the *Path Selection Agent* evaluates all candidates using these POI rankings and returns the top-ranked path as the *Popular Path*.

This orchestration enhances the overall spatial reasoning capability of the framework by leveraging each agent's specialization while maintaining flexibility through conditional stage transitions. In the following subsections, we discuss the agents, their responsibilities, and interactions.

4.1 Path Discovery Agent

The *Path Discovery Agent* serves as the entry point, utilizes LLM to identify trajectory segments where the source appears before the destination, extracting candidate paths $\mathcal{T}_{LLM}$ without heavy computations or external retrieval models. We instruct the LLM to first identify occurrences of source s and destination d POIs within historical data $\mathcal{T}$, then extract complete path segments where s precedes d. In cases of loops, the agent considers both the shorter and the longer segments. The agent either provides *Candidate Paths* for evaluation (remaining in *SEARCH* stage) or returns an empty set $\mathcal{T}_{LLM} = \emptyset$ that triggers the *GENERATE* stage.

4.2 Popularity Ranking Agent

The *Popularity Ranking Agent* quantifies spatial entity importance through frequency analysis, operating in two distinct modes: *Edge ranking mode* and *POI ranking mode*. In *Edge ranking mode*, activated conditionally within the *GENERATE* stage when no historical paths exist, we prompt the LLM to identify and rank edges from historical trajectories in descending order of occurrence frequency. These edge rankings guide the *Path Synthesis Agent* in constructing novel paths. In *POI ranking mode*, which is always executed within the *SEARCH* stage—either directly after path discovery or following the *GENERATE* stage execution—we prompt the LLM to rank all unique POIs in descending order of occurrence frequency.

4.3 Path Synthesis Agent

The *Path Synthesis Agent* generates novel traversable paths when discovery fails, operating exclusively within the *GENERATE* stage. This agent receives edge rankings from the *Popularity Ranking Agent* and constructs valid paths by connecting popular edges, analogous to humans planning routes through unfamiliar areas by combining known road segments. The prompt emphasizes using only edges present in edge popularity ranking; this encodes validity in the *GENERATE* stage through an observed-edge constraint, eliminating

Type	Method	DisHolly 13 POIs	Epcot 17 POIs	CaliAdv 25 POIs	Edin 28 POIs	Toro 29 POIs	Disland 31 POIs	Melb 88 POIs	Average -
ML/DL based	Markov	0.61	0.59	0.53	0.59	0.60	0.50	0.51	0.56
	NASR	0.63	0.60	0.54	0.58	0.58	0.50	0.53	0.57
	DeepAltTrip-LSTM	0.62	0.59	0.53	0.58	0.59	0.52	0.52	0.56
	DeepAltTrip-Samp	0.62	0.58	0.53	0.58	0.59	0.51	0.51	0.56
	NMLR	0.70	0.69	0.55	0.62	0.69	0.65	0.60	0.65
LLM based (Llama 3.1 8b)	Direct	0.50	0.46	0.47	0.45	0.45	0.43	0.40	0.45
	CoT	0.59	0.53	0.56	0.54	0.57	0.48	0.47	0.53
	SC	0.57	0.55	0.54	0.56	0.54	0.50	0.45	0.53
	APE	0.45	0.40	0.42	0.41	0.38	0.34	0.32	0.39
	ReAct	0.68	0.64	0.66	0.70	0.59	0.47	0.43	0.60
	Reflexion	0.69	0.64	0.66	0.70	0.59	0.48	0.44	0.60
	LLM A*	0.58	0.53	0.55	0.60	0.52	0.42	0.40	0.51
	PathGPT	0.68	0.61	0.57	0.74	0.76	0.57	0.63	0.65
	CompassLLM	0.75	0.69	0.71	0.76	0.62	0.49	0.44	0.64
LLM based (GPT 4o)	Direct	0.56	0.51	0.52	0.50	0.50	0.48	0.45	0.50
	CoT	0.66	0.57	0.61	0.59	0.62	0.51	0.52	0.58
	SC	0.64	0.59	0.58	0.61	0.59	0.53	0.49	0.58
	APE	0.79	0.71	**0.75**	0.81	0.65	0.50	0.45	0.67
	ReAct	0.74	0.70	0.73	0.74	0.64	0.49	0.45	0.64
	Reflexion	0.74	0.70	0.73	0.74	0.64	0.51	0.46	0.65
	LLM A*	0.61	0.56	0.57	0.62	0.54	0.45	0.42	0.54
	PathGPT	0.54	0.43	0.34	0.54	0.58	0.24	0.44	0.44
	CompassLLM	0.80	0.73	0.74	0.81	0.65	0.50	0.45	0.67
LLM based (GPT o3 mini)	Direct	0.74	0.73	0.59	0.57	0.68	0.67	0.65	0.66
	CoT	0.82	0.73	0.71	0.75	0.84	0.67	0.78	0.76
	SC	0.74	0.76	0.57	0.68	0.84	0.64	0.80	0.72
	APE	0.81	0.74	0.71	0.80	0.89	0.64	0.81	0.77
	ReAct	0.75	0.73	0.61	0.72	0.87	0.63	0.77	0.73
	Reflexion	0.75	0.75	0.65	0.77	0.81	0.52	0.77	0.72
	LLM A*	0.75	0.73	0.72	0.67	0.87	**0.72**	0.77	0.77
	PathGPT	0.74	0.61	0.69	0.68	0.76	0.47	0.59	0.65
	CompassLLM	**0.84**	**0.79**	0.73	**0.82**	**0.90**	0.71	**0.82**	**0.80**

Table 1: $F1$ scores comparison across datasets. For each dataset, best scores are bolded and second bests are underlined.

hallucination of non-existent connections—a critical requirement for safety-critical navigation systems. The resulting *Generated Candidate Paths* update $\mathcal{T}_{LLM}$ and are returned to the *SEARCH* stage for evaluation.

4.4 Path Selection Agent

The *Path Selection Agent* performs the final path evaluation within the *SEARCH* stage. It receives candidate paths $\mathcal{T}_{LLM}$ (either discovered or synthesized) along with POI rankings, then prompts the LLM to score each path by aggregating the popularity of its constituent POIs. The agent selects the highest-scoring path $\mathcal{R}$ as the final *Popular Path*.

Appendix A contains the algorithm of the entire pipeline and Appendix E provides the detailed prompts for each agent.

5 Experiments

5.1 Dataset

Real-world Data: We used seven Real-world datasets (Rashid et al., 2023) across two domains: geo-tagged Flickr traces from Edinburgh (**Edin**), Toronto (**Toro**), and Melbourne (**Melb**) (Chen et al., 2016a), and trip data from four theme parks, Disney's Hollywood Studios (**DisHolly**), Epcot Theme Park (**Epcot**), Disney California Adventure (**CaliAdv**) & Disneyland (**Disland**) (Lim et al., 2018). While all the datasets were used for *SEARCH* problem, only **Edin**, **Toro** & **Melb** were used for the *GENERATE* problem as other datasets didn't have any POI pair that doesn't have any ground truth, thus unsuitable for *GENERATE* problem evaluation.

Synthetic Data: We generated synthetic datasets to introduce spatial diversity, addressing the edge

Type	Method	Edin 28 POIs	Toro 29 POIs	Melb 88 POIs	Avg -
DL	NMLR	0.89	**0.98**	0.95	0.94
LLM based (Llama 3.1 8b)	Direct	0.12	0.22	0.17	0.17
	CoT	0.21	0.23	0.19	0.21
	SC	0.19	0.22	0.21	0.21
	APE	0.53	0.57	0.42	0.51
	ReAct	0.56	0.51	0.63	0.57
	Reflexion	0.71	0.77	0.78	0.75
	LLM A*	0.78	0.82	0.49	0.70
	PathGPT	0.60	0.53	0.17	0.43
	CompassLLM	0.72	0.78	0.61	0.70
LLM based (GPT 4o)	Direct	0.30	0.21	0.33	0.28
	CoT	0.48	0.60	0.20	0.43
	SC	0.50	0.60	0.22	0.44
	APE	0.86	0.89	0.94	0.90
	ReAct	0.82	0.85	0.86	0.84
	Reflexion	0.83	0.86	0.88	0.86
	LLM A*	0.84	0.82	0.90	0.85
	PathGPT	0.83	0.78	0.43	0.68
	CompassLLM	0.84	0.91	**0.96**	0.90
LLM based (GPT o3 mini)	Direct	0.54	0.51	0.53	0.53
	CoT	0.56	0.51	0.56	0.54
	SC	0.83	0.56	0.53	0.64
	APE	0.80	0.71	0.88	0.80
	ReAct	0.69	0.79	0.81	0.76
	Reflexion	0.54	0.63	0.91	0.69
	LLM A*	0.56	0.53	0.90	0.66
	PathGPT	0.89	0.86	0.71	0.82
	CompassLLM	**0.95**	0.94	0.95	**0.95**

Table 2: $Traversability$ scores across datasets. For each dataset, best scores are bolded and second bests are underlined.

count and edge distribution in real data. Using the reverse-delete algorithm, we controlled edge density while a Steiner tree approximation (Robins and Zelikovsky, 2000) emphasized key "highway" routes. Users followed these highways 90% of the time, deviating only when shorter paths were available, closely mirroring real-world movement patterns. Appendix B provides the entire algorithm and validation for synthetic data generation in detail.

5.2 Baselines & Implementation

We evaluated **CompassLLM** against traditional machine learning, deep learning, and LLM-based models. The **Markov** model (Chen et al., 2016b) predicts user movements with Markov chains, while **NASR** (Wang et al., 2019) employs neural architecture search for urban navigation. **DeepAltTrip** (Rashid et al., 2023) generates diverse alternative routes, and **NMLR** (Jain et al., 2021) creates paths for large-scale traffic networks using Graph Neural Network. Among LLM techniques, we benchmarked **Direct** (Zero Shot), **CoT** (Chain of Thought) (Wei et al., 2023) which breaks tasks into sequential steps, **SC** (Self-Consistency) (Wang et al., 2023) which selects the most consistent reasoning paths, and **APE** (Zhou et al., 2022) which automates task-specific prompt generation. **ReAct** (Yao et al., 2023b) integrates reasoning with action, **Reflexion** (Shinn et al., 2023) enhances performance through iterative feedback, and **LLM A*** (Meng et al., 2024b) adapts A* search for improved spatial reasoning. Finally, **PathGPT** (Marcelyn et al., 2025) implements RAG for retrieving relevant historical data to be sent with zero-shot prompting.

For each method, hyperparameters like *temperature* were set and prompts were crafted following the respective guidelines of the technique. We recorded the number of tokens used in the prompt as well as the tokens generated in the response. For LLM-based methods, we used both an open source (Llama 3.1 8B) and two closed-source models (GPT 4o & GPT o3 mini) as the underlying language model. Additionally, a medium *reasoning effort* was used for the reasoning model, GPT o3 mini. For **CompassLLM**, we maintained a lower *temperature* across all experiments and models as it shows better results [Appendix D.1].

Classical graph-search algorithms would be NP-hard for our objective (Casel and Schmid, 2023; Martens and Popp, 2022), as maximizing POI-additive popularity over simple paths between fixed $\langle s, d \rangle$ reduces to a maximum-weight simple path problem. Similarly, popularity-weighted graph-search variants rely on heuristic design choices (edge weights, normalization, length-bias control) rather than serving as exact solvers for the popular-path objective. For these reasons, we do not include them as baselines.

5.3 Evaluation Metrics

We used two key metrics: $F1$ for *SEARCH* problems (Rashid et al., 2023) and $Traversability$ for *GENERATE* problems.

The $F1$, a harmonic mean of precision and recall, evaluates the accuracy of the recommended paths by comparing them to the ground truth popular paths in the historical trajectories in $\mathcal{T}$.

Let $\mathcal{R} = (q_1, q_2, \ldots, q_K)$ be a recommended path. Also, let $\mathcal{V}_r$ and $\mathcal{V}_\mathcal{R}$ denote the sets of POIs in the ground-truth popular path and the recommended path, respectively. The precision(P), recall (R), and $F1$ of the recommended itinerary $\mathcal{R}$

Metric	Method	DisHolly	Epcot	CaliAdv	Edin	Toro	Disland	Melb	Average
Token Count	APE	5.4k	8.7k	11.6k	17.7k	25.5k	32.5k	67.7k	24.2k
	LLM-A*	2.8k	4.3k	**5.6k**	**8.5k**	**12.0k**	**15.3k**	**31.6k**	**11.4k**
	Reflexion	10.1k	15.6k	20.5k	31.5k	46.2k	59.7k	99.9k	40.5k
	CompassLLM	**2.6k**	**4.2k**	5.7k	8.6k	12.4k	15.8k	33.0k	11.8k

Table 3: Cost comparison among different LLM-based approaches. Best scores are bolded and second bests are underlined.

is calculated as:

$$P = \frac{|\mathcal{V}_r \cap \mathcal{V}_\mathcal{R}|}{|\mathcal{V}_\mathcal{R}|}; R = \frac{|\mathcal{V}_r \cap \mathcal{V}_\mathcal{R}|}{|\mathcal{V}_r|}; F1 = \frac{2 * P * R}{P + R}$$

The $Traversability$ metric is an improved version of $Reachability$ from **NMLR** (Jain et al., 2021). While $Reachability$, being a binary metric, only checks if the last POI in the recommended path is the query destination or not, $Traversability$ measures the model's ability to generate paths that connect source and destination points using consecutive road segments that actually exist in historical routes. For a recommended route $\mathcal{R} = (q_1, q_2, \ldots, q_K)$ and the set of all road segments $\mathcal{E}_\mathcal{T}$, it is mathematically defined as,

$$Traversability = \frac{|\{(q_k, q_{k+1}) : (q_k, q_{k+1}) \in \mathcal{E}_T, k \in [1, K-1]\}|}{K-1}$$

A higher $Traversability$ score indicates a higher degree of edge validity, particularly crucial for safety-critical systems.

6 Results

Performance Analysis

According to established literature in this domain (Jain et al., 2021; Rashid et al., 2023), models aim for even slight improvement in accuracy. In Table 1 we can see that LLM-based methods consistently outperform traditional ML/DL-based methods in the *SEARCH* phase. **APE** performs well on GPT o3 mini & GPT 4o but struggles with Llama 3.1 8b due to its smaller model size; as this method is feedback based, performance in each stage is accumulated. **LLM-A*** struggles with smaller model as well. On the other hand, **ReAct** and **Reflexion** can perform consistently with smaller models too. But both **APE** and **LLM-A*** outperform these approaches with GPT o3 mini, an actual reasoning model. Notably, **CompassLLM** outperforms other methods in almost all model-specific implementations, achieving high-quality results with consistency. As GPT o3 mini is a reasoning model, it has an overall better performance over GPT 4o and Llama 3.1 8b.

In table 2, only **NMLR** is used among ML/DL-based methods, as others failed to *GENERATE* paths without ground truth. It is worth noticing that **NMLR** is specifically built to synthesize valid paths-thus score high on metrics like $Traversability$ with low score on $F1$. However, this comes at the cost of requiring significant retraining on new data, limiting their adaptability in dynamic environments. **PathGPT** performs well in small datasets (Edin) but struggle more than other methods as the dataset grows large (Melb). In contrast, **CompassLLM** maintains competitive $Traversability$ performance with **NMLR** while outperforming others.

This establishes **CompassLLM** as an obvious choice for popular path queries offering (1) Significant practical benefits of LLMs, requiring fewer resources and retraining in dynamic datasets, (2) Outperforming others in *SEARCH* and matching SOTA in *GENERATE*, better of both worlds. **CompassLLM** is most beneficial under sparsity and missing-route scenarios; simpler methods may be competitive in dense regimes. Additional evaluations, including statistical significance and variance analysis, are presented in Appendix D.

Cost Analysis

A key advantage of **CompassLLM** is its ability to balance performance with computational efficiency, as demonstrated in Table 3 (keeping only the best performing methods from Table 1). **APE**, while achieving strong results, incurs significantly higher computational costs due to its reliance on extensive prompt engineering. For instance, in dataset **Melb** with the largest number of POIs, **APE**'s token count reaches 67.7 thousand. Similarly, **Reflexion** also uses a lot of tokens and costs more than the other two. On the other hand, **LLM-A*** is the most cost-effective method but sometimes struggles with performance as seen in table 1 & 2, making it unsuitable for high-quality path generation.

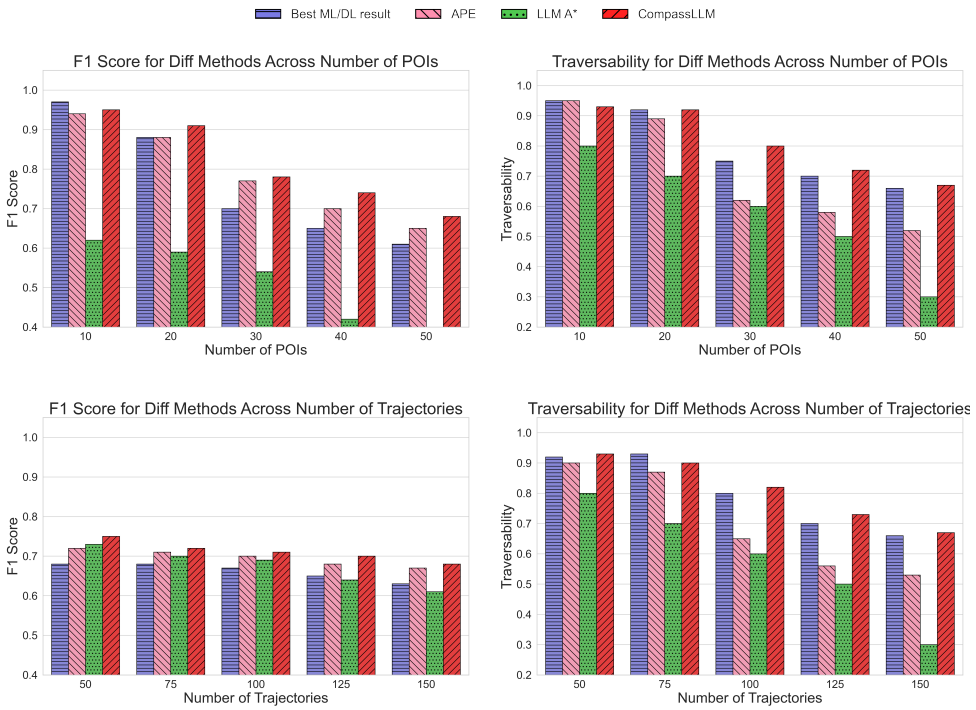

Figure 3: Comparison among approaches on synthetic data.

The efficiency of **CompassLLM** becomes even more pronounced in datasets with smaller POIs, where it incurs costs similar to **LLM-A*** while outperforming it in reasoning for popular path.

Scalability Studies

We evaluate scalability using synthetic datasets with varying complexity: 10-50 POIs and 50-150 trajectories. Figure 3 compares **CompassLLM** against the best-performing ML/DL baseline and competitive LLM-based methods from table 1, 2 & 3. A general downward trend is observed as either of #POI or #trajectories increases. Higher #POI implies sparse data and increased #trajectories denote a larger dataset.

As #POIs increases (top row), $F1$ scores decline across all methods, reflecting the increased difficulty of sparse networks. **CompassLLM** maintains superior $F1$ performance, outperforming **Best ML/DL** by **14%** at 40 POIs and by **11.5%** at 50 POIs. More notably, **CompassLLM** achieves **29%** higher $Traversability$ than **APE** at 30 to 50 POIs, while **LLM-A*** degrades significantly in both metrics.

With increasing #trajectories (bottom row), **CompassLLM** maintains a consistent **6-10%** $F1$ advantage over **Best ML/DL** baselines across all settings. For $Traversability$, **CompassLLM** sustains **+26%** higher performance than **APE** in larger data (100-150 trajectories), while remaining competitive with ML/DL methods (within 1.5%). We can see that, varying #trajectories with #POI kept constant (bottom-left graph) has little effect on $F1$ score (compared to other graphs) as frequency of all the candidate paths are increased proportionally by repeating major patterns.

CompassLLM's superior scalability stems from its specialized agent architecture. The *Discovery Agent* efficiently filters historical data before ranking, reducing context size for downstream agents, compared to methods that process all trajectories simultaneously. When paths must be generated, the *Synthesis Agent* constrains LLM outputs to edges present in historical data only, eliminating hallucinated connections that plague simple prompting approaches. This explicit validation maintains high $Traversability$ even as network sparsity increases. Meanwhile, the *Popularity Ranking Agent*'s frequency-based scoring ensures that **CompassLLM** identifies truly popular paths and obtains high $F1$. In contrast, **LLM-A*** degrades rapidly because it relies on heuristic search without explicit edge constraints or popularity goal. **APE**'s iterative prompt engineering accumulates errors across feedback rounds in sparse networks,

while **CompassLLM**'s modular design isolates and handles each sub-problem (discovery, ranking, synthesis, selection) independently.

7 Conclusion & Future Works

In this work, we presented **CompassLLM**, a framework utilizing multiple agents for spatial reasoning to solve the Popular Path Problem. Our experiments show that **CompassLLM** is cost-effective, highly competitive in generating and often outperforming in finding paths, not to mention particularly effective under sparse data conditions. However, our results remain experimental, as **CompassLLM** may generate suboptimal or invalid paths in certain cases, and the inherent stochasticity of LLMs poses challenges in consistency. Our work bolsters LLMs' success in geospatial reasoning, inaugurates agentic approach, and could be extended in many directions. Future work can focus on incorporating more contextual data (e.g., time, user preference, and environmental factors), and enhancing models' reliability through improved prompt engineering and dynamic memory management to handle larger datasets effectively.

Limitations

While **CompassLLM** demonstrates promising results in popular path discovery and synthesis, several limitations warrant consideration:

Large-scale Data Handling: CompassLLM faces challenges when processing extensive datasets. The current implementation struggles with input sizes exceeding 128,000 tokens, due to the context window limitations of existing LLMs, which limits its applicability to very large or complex spatial networks. Moreover, according to Table 7 in Appendix, no further compression of **CompassLLM** prompts are feasible.

Incompatibility with Internal Prompt Compression: Many LLMs employ internal prompt compression techniques to manage large inputs efficiently. However, **CompassLLM**'s performance relies on the full, uncompressed prompt structure. Consequently, models that automatically compress prompts may not be compatible with our approach, potentially limiting the range of applicable LLMs.

Prompt and Reasoning Chain Optimization: While our current prompting strategy yields effective results, there may exist more optimized prompt structures or reasoning chains that could further enhance **CompassLLM**'s performance. The current implementation, while effective, may not represent the absolute optimal prompt design for all scenarios.

References

Mohamed Aghzal, Erion Plaku, and Ziyu Yao. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *arXiv preprint arXiv:2310.03249*.

Pasquale Balsebre, Weiming Huang, and Gao Cong. 2024. Lamp: A language model on the map. *Preprint*, arXiv:2403.09059.

Prithu Banerjee, Sayan Ranu, and Sriram Raghavan. 2014. Inferring uncertain trajectories from partial observations. In *ICDM*, pages 30–39.

Katrin Casel and Markus L. Schmid. 2023. Fine-Grained Complexity of Regular Path Queries. *Logical Methods in Computer Science*, Volume 19, Issue 4:8625. ArXiv:2101.01945 [cs].

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *Preprint*, arXiv:2310.05845.

Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016a. Learning points and routes to recommend trajectories. In *CIKM 2016 - Proceedings of the 2016 ACM Conference on Information and Knowledge Management*, International Conference on Information and Knowledge Management, Proceedings, pages 2227–2232. Association for Computing Machinery. Publisher Copyright: © 2016 ACM.; 25th ACM International Conference on Information and Knowledge Management, CIKM 2016 ; Conference date: 24-10-2016 Through 28-10-2016.

Dawei Chen, Cheng Soon Ong, and Lexing Xie. 2016b. Learning points and routes to recommend trajectories. In *The Conference on Information and Knowledge Management (CIKM)*, pages 2227–2232.

Z. Chen, H. T. Shen, and X. Zhou. 2011. Discovering popular routes from trajectories. In *ICDE*, pages 900–911.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and 1 others. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Hourui Deng, Hongjie Zhang, Jie Ou, and Chaosheng Feng. 2025. Can llm be a good path planner based on prompt engineering? mitigating the hallucination for path planning. In *International Conference on Intelligent Computing*, pages 3–15. Springer.

Mahir Labib Dihan, MD Tanvir Hassan, MD TANVIR PARVEZ, Md Hasebul Hasan, Md Almash Alam, Muhammad Aamir Cheema, Mohammed Eunus Ali, and Md Rizwan Parvez. 2025. Mapeval: A map-based evaluation of geo-spatial reasoning in foundation models. In *Forty-second International Conference on Machine Learning*.

Pranav Doma, Aliasghar Arab, and Xuesu Xiao. 2024. Llm-enhanced path planning: Safe and efficient autonomous navigation with instructional inputs. *Preprint*, arXiv:2412.02655.

Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. Trajectory pattern mining. In *KDD*, pages 330–339.

Zhouhong Gu, Haoning Ye, Xingzhou Chen, Zeyang Zhou, Hongwei Feng, and Yanghua Xiao. 2025. Structext-eval: Evaluating large language model's reasoning ability in structure-rich text. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 223–244.

Atharva Gundawar, Mudit Verma, Lin Guan, Karthik Valmeekam, Siddhant Bhambri, and Subbarao Kambhampati. 2024. Robust planning with llm-modulo framework: Case study in travel planning. *arXiv preprint arXiv:2405.20625*.

Md Hasebul Hasan, Mahir Labib Dihan, Mohammed Eunus Ali, and Md Rizwan Parvez. 2025. Mapagent: A hierarchical agent for geospatial reasoning with dynamic map tool integration. *Preprint*, arXiv:2509.05933.

Marcelo Yuji Himoro and Antonio Pareja-Lora. 2022. Preliminary results on the evaluation of computational tools for the analysis of Quechua and Aymara. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5450–5459, Marseille, France. European Language Resources Association.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapcoder: Multi-agent code generation for competitive problem solving. *arXiv preprint arXiv:2405.11403*.

Jayant Jain, Vrittika Bagadia, Sahil Manchanda, and Sayan Ranu. 2021. Neuromlr: Robust & reliable route recommendation on road networks. *Advances in Neural Information Processing Systems*, 34:22070–22082.

Myeongjun Jang, Deuk Sin Kwon, and Thomas Lukasiewicz. 2022. BECEL: Benchmark for consistency evaluation of language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3680–3696, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Kaya Stechly, Mudit Verma, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks. *arXiv preprint arXiv:2402.01817*.

Joseph B Kruskal. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, and 1 others. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. *arXiv preprint arXiv:2407.04069*.

Ehsan Latif. 2024. 3p-llm: Probabilistic path planning using large language model for autonomous robot navigation. *Preprint*, arXiv:2403.18778.

Bohang Li, Kai Zhang, Yiping Sun, and Jianke Zou. 2024. Research on travel route planning optimization based on large language model. In *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, pages 352–357.

Xiucheng Li, Gao Cong, and Yun Cheng. 2020. Spatial transition learning on road networks with deep probabilistic models. In *ICDE*, pages 349–360.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. 2010. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734.

Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2018. Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowl. Inf. Syst.*, 54(2):375–406.

Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M Ni. 2013. Finding time period-based most frequent path in big trajectory data. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, pages 713–724.

Steeve Cuthbert Marcelyn, Yucen Gao, Yuzhe Zhang, Xiaofeng Gao, and Guihai Chen. 2025. Pathgpt: Leveraging large language models for personalized route generation. *arXiv preprint arXiv:2504.05846*.

Wim Martens and Tina Popp. 2022. The Complexity of Regular Trail and Simple Path Queries on Undirected Graphs. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '22, pages 165–174, New York, NY, USA. Association for Computing Machinery.

Silin Meng, Yiwei Wang, Cheng-Fu Yang, Nanyun Peng, and Kai-Wei Chang. 2024a. Llm-a*: Large language model enhanced incremental heuristic search on path planning. *arXiv preprint arXiv:2407.02511*.

Silin Meng, Yiwei Wang, Cheng-Fu Yang, Nanyun Peng, and Kai-Wei Chang. 2024b. Llm-a*: Large language model enhanced incremental heuristic search on path planning. *Preprint*, arXiv:2407.02511.

Syed Md Mukit Rashid, Mohammed Eunus Ali, and Muhammad Aamir Cheema. 2023. Deepalttrip: Top-k alternative itineraries for trip recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9433–9447.

Jonathan Roberts, Timo Lüddecke, Sowmen Das, Kai Han, and Samuel Albanie. 2023. Gpt4geo: How a language model sees the world's geography. *Preprint*, arXiv:2306.00020.

Gabriel Robins and Alexander Zelikovsky. 2000. Improved steiner tree approximation in graphs. In *SODA*, pages 770–779.

SP Sharan, Francesco Pittaluga, Manmohan Chandraker, and 1 others. 2023. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*.

Dingyuan Shi, Yongxin Tong, Zimu Zhou, Ke Xu, Zheng Wang, and Jieping Ye. 2024. GRAPH-CONSTRAINED DIFFUSION FOR END-TO-END PATH PLANNING. In *The Twelfth International Conference on Learning Representations*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*.

Wei Tian, Jieming Shi, Siqiang Luo, Hui Li, Xike Xie, and Yuanhang Zou. 2023. Effective and efficient route planning using historical trajectories on road networks. *Proceedings of the VLDB Endowment*, 16(10):2512–2524.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36.

Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. 2019. Empowering a* search algorithms with neural networks for personalized route recommendation. In *KDD*, pages 539–547.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *SIGKDD*, pages 195–203.

Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. In *IJCAI*, pages 3083–3090.

Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. 2016. Probabilistic robust route recovery with spatio-temporal dynamics. In *KDD*, pages 1915–1924.

Ling Xiao and Toshihiko Yamasaki. 2025. Llm-advisor: An llm benchmark for cost-efficient path planning across multiple terrains. *Preprint*, arXiv:2503.01236.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *Preprint*, arXiv:2402.01622.

Can Yang and Gyozo Gidofalvi. 2018. Fast map matching, an algorithm integrating hidden markov model with precomputation. *International Journal of Geographical Information Science*, 32(3):547–570.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*.

Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324.

Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pages 99–108.

Liangqi Yuan, Dong-Jun Han, Christopher G. Brinton, and Sabine Brunswicker. 2025. Llmap: Llm-assisted multi-objective route planning with user preferences. *Preprint*, arXiv:2509.12273.

Pujun Zhang, Shan Liu, Jia Shi, Liying Chen, Shuiping Chen, Jiuchong Gao, and Hai Jiang. 2023. Route planning using divide-and-conquer: A gat enhanced insertion transformer approach. *Transportation Research Part E: Logistics and Transportation Review*, 176:103176.

K. Zheng, Y. Zheng, X. Xie, and X. Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE*, pages 1144–1155.

Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A COMPASSLLM Algorithm

The full **COMPASSLLM** algorithm is outlined in Algorithm 1.

Algorithm 1: COMPASSLLM

Input: $\mathcal{T}$ is a set of historical trajectories, s is the start point, d is the destination
Output: Popular path $\mathcal{R}$ from s to d

$\mathcal{T}_{LLM} \leftarrow$ PathDiscoveryAgent$(\mathcal{T}, s, d)$;

`// No Candidates - GENERATE Block`
if $\mathcal{T}_{LLM} = \emptyset$ **then**
  EdgeRanks $\leftarrow$ PopularityRankingAgent$(\mathcal{T}, mode = edge)$;
  `// Update Candidates`
  $\mathcal{T}_{LLM} \leftarrow$ PathSynthesisAgent$(s, d,$ EdgeRanks$)$

POIRanks $\leftarrow$ PopularityRankingAgent$(\mathcal{T}, mode = poi)$;
rankedPaths $\leftarrow$ PathSelectionAgent$(\mathcal{T}_{LLM},$ POIRanks$)$;

`// Return best path`
$\mathcal{R} \leftarrow$ rankedPaths[0];
return $\mathcal{R}$;

B Synthetic Data Generation

B.1 Synthetic Data Generation Pipeline

Our synthetic trajectory generation follows a four-step pipeline illustrated in Figure 4, designed to create realistic spatial networks that mirror the structural properties of real-world transportation systems.

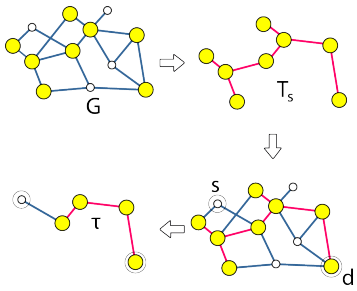

Figure 4: Synthetic Data Generation Process

Step 1: Graph Construction and Pivotal Node Identification. We begin by constructing a base graph G using the reverse-delete algorithm (Kruskal, 1956) to generate a random connected network with controlled edge density. Within this network, we strategically identify pivotal nodes V_s (depicted as yellow nodes) representing high-importance locations such as popular destinations, transportation hubs, or landmarks. Regular nodes (white nodes) serve as intermediate waypoints, creating a realistic spatial hierarchy.

Step 2: Steiner Tree Construction. Using approximation algorithms, we construct a Steiner tree T_s that efficiently connects all pivotal nodes V_s. This tree structure, highlighted with pink/red edges, forms the backbone of our synthetic network and represents the primary "highway" system—the most efficient pathways between important locations that real-world travelers would naturally prefer.

Path Discovery	Popularity Ranking	Path Synthesis	Path Selection	$F1$	$Traversability$	Token Usage	Remarks
✓	✓	✓	✓	**0.80**	**0.95**	**11.8k**	Best performing
×	✓	✓	✓	0.63	0.91	15.8k (increased)	Always triggers GENERATE even when historical paths exist; slow and expensive; may generate worse paths than discovered ones; **shows necessity of conditional two-stage workflow**
✓	×	✓	✓	0.48	0.90	10.2k	Path Selection Agent has no popularity guidance for ranking candidates; essentially random/heuristic-based selection; **shows Popularity Ranking drives the system to Search for Popular paths effectively**
✓	✓	×	✓	0.80	0.42	9.5k	Only impacts GENERATE stage; greedy approach (pick most popular outgoing edge) creates locally optimal but globally invalid paths; can get stuck with no path to destination; **demonstrates need for global path-planning**
✓	✓	✓	×	0.77	0.95	10.6k	Simple arithmetic scoring captures core popularity logic; **shows LLM selection adds modest value; programmatic approach is competitive**

Table 4: Ablation Study Results

Experiment	CompassLLM (GENERATE)	Zero Shot	CoT	Self-consistency	ReAct	Reflexion	LLM A*	APE
path with [[X]] POIs	8	0	5	6	7	8	7	1
path that goes through [[Y]] POI	6	0	4	4	7	8	6	0
path that avoids [[Z]] POI	8	0	6	6	9	9	8	1

Table 5: Comparison of constraint satisfaction across LLM-based approaches

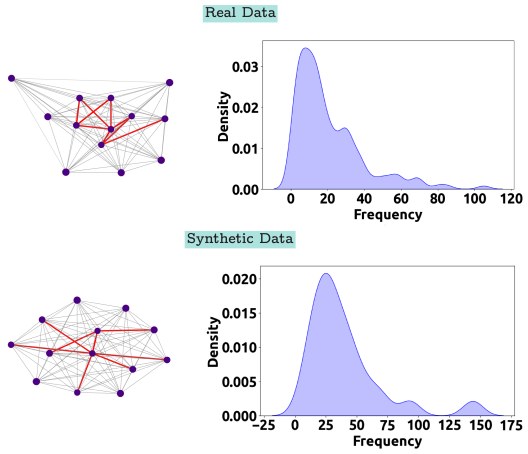

Figure 5: Comparison of Real and Synthetic Data

Location	Nodes	Density	Routes	Result	Actual
Epcot	17	0.2	1248	212	207
Disneyland	31	0.4	2792	681	618
DisHolly	13	0.4	901	107	134

Table 6: Comparison of synthetic and real-world data across multiple locations. The table presents key characteristics of the generated spatial datasets, including the number of nodes, spatial density, and the number of unique routes. The "Result" column denotes the number of unique source-destination pairs generated by our synthetic trajectory model, while the "Actual" column provides the corresponding real-world counts.

Step 3: Source-Destination Pair Selection. We systematically select source node s and destination node d pairs from the network, ensuring diverse spatial configurations. This selection process considers both the connectivity patterns established by the Steiner tree and the spatial distribution of pivotal nodes to generate realistic travel scenarios.

Step 4: Trajectory Synthesis. We generate synthetic trajectories τ from source s to destination d using a hybrid approach: trajectories preferentially utilize Steiner tree edges (the highway network) for efficiency, while incorporating direct connections when they provide superior routes. This balances realistic movement behavior with network structure constraints.

Dataset	Metric	COMPASSLLM GENERATE	Selective Context (10%)	Selective Context (20%)
Epcot	Reduction $Traversability$	0 0.84	16% 0.15	24% 0
CaliAdv	Reduction $Traversability$	0 0.91	14% 0.12	22% 0
Disland	Reduction $Traversability$	0 0.96	13% 0.10	20% 0.1
Average	Reduction $Traversability$	0 0.80	13.75% 0.14	21% 0.05

Table 7: Comparison of Reduction and $Traversability$ across various datasets and techniques.

B.2 Validation Against Real-World Data

Figure 5 demonstrates that our synthetic data generation successfully captures the essential characteristics of real-world spatial patterns. The comparison reveals striking similarities between synthetic and real trajectory distributions:

- **Network Structure:** Both synthetic and real networks exhibit similar connectivity patterns, with certain paths (highlighted in red) experiencing significantly higher traffic than others (shown in grey).
- **Frequency Distributions:** The density plots show comparable frequency distributions between synthetic and real data, with both exhibiting right-skewed distributions characteristic of real-world movement patterns where a few popular routes dominate usage.

Table 6 provides quantitative validation of our synthetic data generation, comparing the count of unique source-destination ground truth pairs between synthetic and real datasets. The "Result" column indicates generated unique pairs, while "Actual" represents corresponding real-world counts, demonstrating our method's ability to produce realistic trajectory volumes.

This systematic validation confirms that our synthetic trajectories exhibit authentic spatial characteristics while maintaining the structural properties necessary for effective LLM evaluation in spatial reasoning tasks. The density plots (where the area under each curve equals one) reveal that our synthetic data successfully replicates the underlying statistical patterns of real-world spatial movement, providing a reliable foundation for comprehensive algorithm evaluation.

This methodology enables us to generate synthetic trajectories that closely model actual spatial data, providing a nuanced representation of spatial interactions for comprehensive evaluation of LLM performance in spatial data analysis.

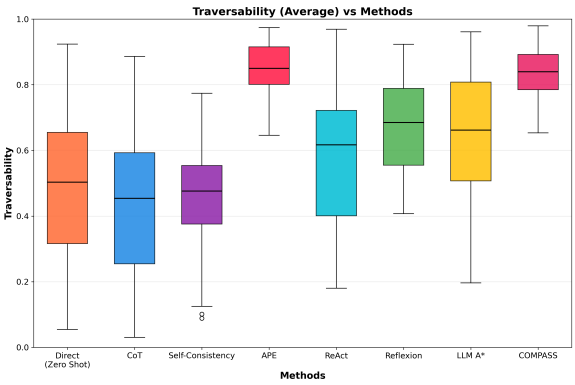

Figure 6: Consistency Across Prompting Techniques: This box plot illustrates the variation in $Traversability$ scores across different prompting techniques. Lower variance is preferable, as it indicates more consistent results.

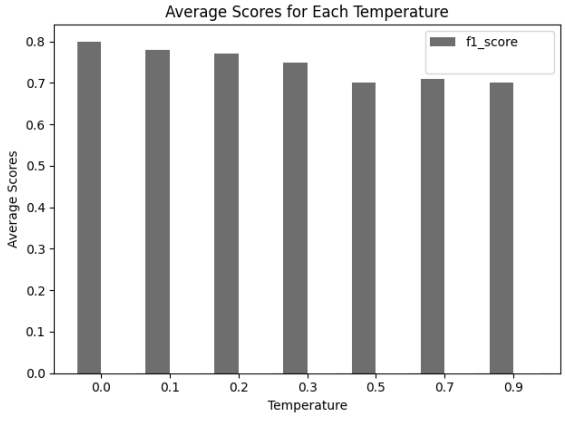

Figure 7: Average F1 Score across all dataset against varying temperature

C Ablation

An ablation study with dropping one agent at a time and replaced by algorithmic counterparts is presented at Table 4. Experiments are conducted with GPT o3 mini.

D Additional Results & Trends

D.1 $F1$ with varying Temperature

Fig 7 shows that LLMs generate better result with lower temperatures as a higher temperature can lead to hallucinating and a lower temperature gives out thoughtful answer. This led us to run all our experiments with temperature zero.

D.2 Statistical Significance

Across the 7 *SEARCH* datasets, a one-sided Wilcoxon signed-rank test confirms that **CompassLLM** (GPT o3 mini) significantly outperforms every LLM and ML/DL baseline ($p < 0.05$ for all; $p < 0.01$ for all except LLM-A*). With GPT 4o, **CompassLLM** significantly outperforms most baselines but achieves parity with APE (mean difference +0.003, $p = 0.38$), consistent with our observation that gains are smaller in denser regimes. For *GENERATE*, having only three datasets limits statistical power, so we refer to the variance analysis in Appendix D.3 and not significance testing.

D.3 Variance of $Traversability$

A higher $traversability$ is always preferable but consistency is also necessary. Fig 6 shows that APE and **CompassLLM** are the only high performing and low variant ones.

D.4 $Traversability$ with Reduction

Selective Context (Li et al., 2023) is a method that enhances the inference efficiency of LLMs by identifying and pruning redundancy in the input context to make the input more compact. Table 7 shows the comparison between **CompassLLM**, Selective Context 10% and Selective Context 20% in average reduction and $traversability$ across various datasets. This proves that **CompassLLM** prompts are already compact enough themselves and no level of reduction can improve/maintain the same performance. It validates the low-cost of **CompassLLM**.

D.5 Constraint Handling

We have experimented some graph prior constraints on the LLM based approaches - 10 experiment for each case. Table 5 shows a comparison of how many of the tasks each approach could solve properly.

E CompassLLM Example Prompts

As an adaptive framework, CompassLLM uses only the source-destination pair and historical paths as structured input (Gu et al., 2025). While POI ID or Name could be used interchangeably in our solution (`25->16 or EdinCastle->Mound`), doing so made little difference in our experiments. This indicates that no inherent knowledge in the LLMs about these locations was particularly useful in the process; only the orchestration itself drove the LLMs to achieve such results. Here are some example prompts for each agent.

Path Discovery Agent Prompt

```
You are given a list of historical trajectories and a source-destination pair. Your task is to extract all candidate paths that connect the source to the destination.

Historical Trajectories:
Calton Hill -> National Monument -> Arthur's Seat;
Edinburgh Castle -> Royal Mile -> National Museum of Scotland -> University of Edinburgh -> Calton Hill -> Royal Botanic Garden -> Edinburgh Zoo -> Greyfriars Kirkyard -> Camera Obscura -> Scottish Parliament -> Nelson Monument -> Dynamic Earth;
Princes Street Gardens -> Palace of Holyroodhouse -> National Museum of Scotland -> Scott Monument -> St Giles' Cathedral -> Dynamic Earth;
Edinburgh Castle -> Palace of Holyroodhouse -> St Giles' Cathedral -> Greyfriars Kirkyard -> Dynamic Earth;
. . .
Royal Mile -> Arthur's Seat -> Nelson Monument;
Arthur's Seat -> National Museum of Scotland -> Edinburgh Zoo;
Edinburgh Castle -> National Museum of Scotland -> Edinburgh Zoo;

Source: Royal Botanic Garden
Destination: Arthur's Seat

———

Important: Your response must follow the following JSON format:

{
"identification_process": "Explain how we can identify existing paths from historical data
```

that contain both source and destination.",
"candidate_paths": "Retrieve all possible routes connecting the source and destination from the historical data."
}

Popularity Ranking Agent Prompt

You are given historical trajectory data. Your task is to analyze and rank all {POIs | POI Pairs (Edges)} based on their popularity.

Historical Trajectories:
Calton Hill -> National Monument -> Arthur's Seat;
Edinburgh Castle -> Royal Mile -> National Museum of Scotland -> University of Edinburgh -> Calton Hill -> Royal Botanic Garden -> Edinburgh Zoo -> Greyfriars Kirkyard -> Camera Obscura -> Scottish Parliament -> Nelson Monument -> Dynamic Earth;
Princes Street Gardens -> Palace of Holyroodhouse -> National Museum of Scotland -> Scott Monument -> St Giles' Cathedral -> Dynamic Earth;
Edinburgh Castle -> Palace of Holyroodhouse -> St Giles' Cathedral -> Greyfriars Kirkyard -> Dynamic Earth;
. . .
Royal Mile -> Arthur's Seat -> Nelson Monument;
Arthur's Seat -> National Museum of Scotland -> Edinburgh Zoo;
Edinburgh Castle -> National Museum of Scotland -> Edinburgh Zoo;

———

Important: Your response must follow the following JSON format:

For POI Mode:

{
"calculation_method": "Step by step explain how the popularity of various points of interest (POIs) can be analyzed using historical trajectory data.",
"ranking_analysis": "Analyze the ranking the POIs based on their frequency of visits or interactions in the dataset and provide the ranking.",
"poi_rank": "Give out the POIs in the ascending order of rank."
}

For Edge Mode:

{
"extracted_edges": "Extract edges that can be found from the trajectory data.",
"analysis_method": "How to analyze edge frequency and find the most popular edges.",
"edge_rank": "Give out the edges in (POI1, POI2) format in descending order of their popularity."
}

Path Synthesis Agent Prompt

You are given edge popularity rankings. For a source-destination pair with no existing path in historical data, generate candidate paths.

Source: Royal Botanic Garden
Destination: Arthur's Seat

Edge Popularity Ranking: (Dynamic Earth, Scott Monument), (Calton Hill, Royal Botanic Garden), (Royal Mile, National Museum of Scotland), (Scott Monument, Calton Hill), (Edinburgh Castle, Royal Mile), (National Museum of Scotland, University of Edinburgh), (Calton Hill, National Monument), (National Monument, Arthur's Seat)...

———

Important: Your response must follow the following JSON format:

{
"generation_strategy": "Explain the strategy for combining popular edges to create realistic paths.",
"generated_paths": "Generate possible path candidates using edge popularity rankings."
}

Path Selection Agent Prompt

You are given candidate paths extracted from historical data and POI popularity rankings. Your task is to rank these paths based on their popularity.

Candidate Paths:
Royal Botanic Garden -> Calton Hill -> National Monument -> Arthur's Seat;
Royal Botanic Garden -> Edinburgh Zoo -> National Museum of Scotland -> University of Edinburgh -> Royal Mile -> Arthur's Seat;
Royal Botanic Garden -> Princes Street Gardens -> Palace of Holyroodhouse -> Arthur's Seat;

POI Popularity Ranking: National Museum of Scotland, Scott Monument, Edinburgh Zoo, St Giles' Cathedral, Royal Mile, Greyfriars Kirkyard, Royal Botanic Garden, Calton Hill, Arthur's Seat, Palace of Holyroodhouse, Edinburgh Castle, Camera Obscura, University of Edinburgh, National Monument

———

Important: Your response must follow the following JSON format:

{
"evaluation_method": "Think step by step on

```
how to evaluate paths using POI popularity
rankings and retrieve the best path.",
"ranked_paths": "Rank all candidate paths
based on the popularity of POIs they
traverse."
}
```