ProfiLLM: Utility-Aligned Agentic User Profiling for Industrial Ride-Hailing Dispatch [Scalable Data Science]

Tengfei Lyu*
HKUST(GZ)
tlyu077@connect.hkust-gz.edu.cn

Zirui Yuan*
HKUST(GZ)
zyuan779@connect.hkust-gz.edu.cn

Xu Liu
Didichuxing Co. Ltd
leoliuxu@didiglobal.com

Kai Wan
Didichuxing Co. Ltd
peterwan@didiglobal.com

Zihao Lu
Didichuxing Co. Ltd
luzihao@didiglobal.com

Li Ma
Didichuxing Co. Ltd
malimarey@didiglobal.com

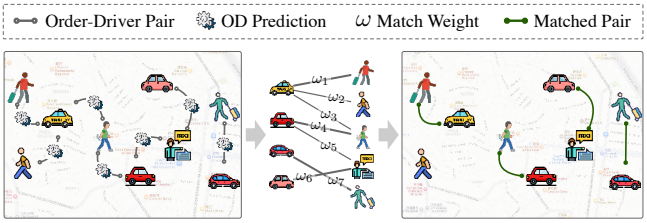
Hao Liu†
HKUST(GZ)
liuh@ust.hk

ABSTRACT

Bringing Large Language Models (LLMs) into industrial ride-hailing dispatch as semantic feature extractors over platform-scale behavioral logs is a compelling but under-explored data systems problem. Production matching pipelines remain dominated by structured numerical features, yet decisive behavioral signals (e.g., a driver's habitual aversion to certain regions) are inherently contextual and naturally expressible as LLM-generated user profiles. However, scaling such profiling to a live, millisecond-latency dispatcher faces three intertwined constraints rarely addressed together: on a platform with millions of daily orders, logs exceed any LLM's context window by orders of magnitude; most users are long-tail, with too few interactions for per-user profiling; and surface-fluent profiles do not necessarily improve downstream prediction utility. We present ProfiLLM, an agentic LLM data pipeline that operationalizes utility-aligned user profiling for production matching systems through two modules. (1) Tool-Augmented Global Knowledge Mining equips an LLM agent with 27 analytical tools to mine platform-scale data, producing reusable global knowledge, adaptive user clustering rules, and region-level supply-demand priors. (2) Utility-Aligned Profile Exploration generates multiple candidate profiles per cluster, evaluates them via a lightweight downstream utility proxy, iteratively refines the best candidates and constructs preference pairs for DPO fine-tuning. The pipeline enforces a strict offline–online contract: all LLM reasoning stays offline, while online serving reduces to a lookup of pre-computed cluster-level profile embeddings, adding sub-millisecond overhead with zero online LLM inference. Deployed on DiDi's production dispatcher, ProfiLLM achieves up to +6.14% relative AUC improvement in outcome prediction, up to +4.35% GMV gain in dispatching simulation, and consistent improvements in a 14-day online A/B test including +0.47% GMV, +0.33% Completion Rate, and −0.82% Cancel-Before-Accept rate. Code, extended experiments, additional analyses, and supplementary materials are available at our project page https://ProfiLLM.github.io.

1 INTRODUCTION

Ride-hailing services have become an essential component of modern urban transportation, fundamentally changing how people commute and travel in cities worldwide. At the core of these platforms is order dispatching, which continuously matches passenger requests with available drivers under stringent real-time latency constraints. To evaluate the quality of each potential match, the platform must anticipate a sequence of user behavioral outcomes along the order fulfillment funnel for each candidate order-driver (OD) pair, including whether the driver will *accept* the dispatched order and whether either party will *cancel* after acceptance but before trip completion. As illustrated in Figure 1, a typical production pipeline operates in three stages. (i) Predicting these per-stage outcomes for each candidate OD pair. (ii) Composing the predicted probabilities into matching weights that quantify the expected utility of each assignment. (iii) Computing a globally optimal assignment via bipartite matching, *e.g.,* Kuhn-Munkres. Among these stages, outcome prediction is the primary quality bottleneck, as prediction errors at any stage propagate directly into suboptimal matches that increase passenger waiting time, reduce driver income efficiency, and degrade platform revenue.



(a) Order-Driver Pair (b) Weight Assignment (c) Matched Results

Figure 1: Industrial order-dispatching pipeline: (a) outcome prediction for candidate order-driver (OD) pairs, (b) weight assignment, and (c) bipartite matching. Our work improves stage (a) via LLM-generated user profiles.

Current production outcome predictors mainly rely on structured numerical features (*e.g.,* distance, price), leaving implicit semantic factors largely unexploited. However, decisive signals governing acceptance and cancellations are inherently contextual, such as a driver's long-term aversion to certain areas or a passenger's strict time-sensitivity during weekday mornings. Such patterns are difficult to capture via handcrafted numerical features, but they can be naturally expressed as user profiles in language. In particular, Large Language Models (LLM) possess strong summarization and reasoning capabilities that enable them to distill complex behavioral trajectories into semantically rich contextual profiles, motivating the exploration of LLM-based user profiling for outcome prediction.

To validate this hypothesis, we conducted a pilot study on high-frequency users who have sufficient historical interactions. We

*Equal contribution. Work done during internship at Didichuxing Co. Ltd.
†Corresponding author.

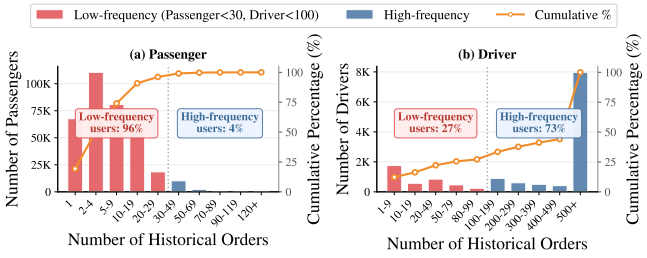


Figure 2: Distribution of historical order counts over a 38-day period in City A, revealing a severe long-tail pattern.

employed an LLM to summarize their historical order trajectories into contextual profiles and incorporated it as additional features for outcome prediction. Averaged across high-frequency users, the profile-enhanced approach achieves **3.71%** and **9.64%** relative AUC improvement for driver cancellation and passenger cancellation, respectively, over models using only structured features, indicating that LLM-generated contextual profiles capture OD-relevant signals largely invisible to traditional feature-based methods.

However, the pilot study was conducted offline and restricted to high-frequency users with rich order histories. While these offline gains are encouraging, bridging the gap to production-scale online deployment introduces several fundamental obstacles. First, the majority of users are low-frequency and lack sufficient order history for reliable LLM-based profiling. Second, without platform-level knowledge to ground the generation, the LLM lacks a consistent reference frame, and when conditioned on a single user's sparse trajectory, it produces profiles of highly variable quality. Third, even when order history is abundant, the generated profiles are not guaranteed to improve downstream prediction, as the LLM has no explicit signal to optimize for prediction utility. These observations motivate three interconnected challenges that must be addressed.

Challenge 1: Scalably mining global operational knowledge from massive historical data. Generating reliable knowledge requires understanding platform-level regularities such as temporal patterns, regional heterogeneity, and causal factors underlying order outcomes. Such global knowledge provides grounding context that individual user histories alone cannot supply, yet raw logs are massive, a small city in our deployment yields 44.3M dispatching records over a 38-day window, far exceeding any LLM's context capacity, and a full multi-city refresh scales by another order of magnitude. **Challenge 2: Adaptive user clustering under long-tail data distributions.** Even with global knowledge, per-user profiles remain infeasible for the dominant low-frequency population. Over the same 38-day window in City A, 96% of passengers appear in $\leq$ 30 orders (Figure 2); while drivers exhibit higher engagement (73% high-frequency), the passenger-side sparsity fundamentally limits user profiling since outcomes from both parties must be predicted for each candidate match. A principled clustering mechanism is required to group users by behavioral similarity so that each cluster accumulates sufficient data for reliable profile learning. **Challenge 3: Ensuring LLM-generated profiles are utility-aligned with downstream prediction.** With global knowledge and user clusters, LLMs can still produce fluent profile descriptions that fail to capture the specific factors driving behavioral decisions. As our experiments later confirm (Table 2), several strong off-the-shelf LLM backbones, degrade downstream prediction AUC by up to −7.57% when used naively, demonstrating that profile fluency is not a reliable proxy for prediction utility. A systematic mechanism is therefore needed to explore, evaluate, and refine profiles based on measurable downstream utility, while remaining compatible with strict online latency constraints.

To address these challenges, we introduce **ProfiLLM**, a practical data framework for deploying utility-aligned LLM user profiles in real-time ride-hailing matching. We devise a *Tool-Augmented Global Knowledge Mining* module in which an LLM agent, equipped with 27 analytical tools, analyzes platform-scale historical data following an `Explore` → `Deepen` → `Validate` → `Synthesize` paradigm, producing (i) actionable global knowledge (*e.g.,* , temporal patterns, weather impacts, causal relationships), (ii) adaptive user clustering with interpretable rules at appropriate granularity, and (iii) regional supply-demand priors from grid-level spatial analyses. We propose a *Utility-Aligned Profile Exploration* mechanism that, for each user cluster, generates candidates profiles conditioned on the mined global knowledge, scores them via rule-based prediction as a lightweight utility proxy, and iteratively refines the best candidates through feedback derived from prediction error analysis. The resulting utility comparisons yield preference pairs that drive DPO fine-tuning, further improving profile generation quality. The generated profiles are embedded and integrated with structured features for real-time outcome prediction. Critically, all LLM inference occurs offline; the online system operates solely on pre-computed cluster-level profile embeddings, ensuring negligible added latency compatible with production requirements. Our contributions:

- We design **ProfiLLM**, an agentic LLM data pipeline that enforces a strict offline–online contract for behavioral profiling in industrial ride-hailing: all LLM reasoning is batch-offline, while the production dispatcher serves only pre-computed cluster-level profile embeddings. To our knowledge, this is the first deployed LLM-based user profiling pipeline for a production ride-hailing dispatcher.
- We design a **Tool-Augmented Global Knowledge Mining** module that exposes 27 analytical tools as a composable operator layer, driven by an LLM agent under an Explore-Deepen-Validate-Synthesize paradigm. The agent autonomously composes operator chains to extract actionable global knowledge, adaptive user-clustering rules, and regional supply-demand priors from platform-scale logs that exceed any LLM context window.
- We propose a **Utility-Aligned Profile Exploration** mechanism that turns prediction utility into a first-class optimization signal: for each user cluster, multiple candidate profiles are generated and ranked by a lightweight LOGIC-rule proxy of downstream prediction; the resulting cross-candidate preferences drive DPO fine-tuning, aligning profile generation with the downstream matching objective.
- We **deploy ProfiLLM** on DiDi's production dispatcher. The online path adds sub-millisecond overhead with zero online LLM inference. ProfiLLM yields up to **+6.14%** outcome-prediction AUC and **+4.35%** simulator GMV; a 14-day City A A/B test confirms **+0.47%** GMV, **+0.33%** completion rate, and **−0.82%** cancel rate.

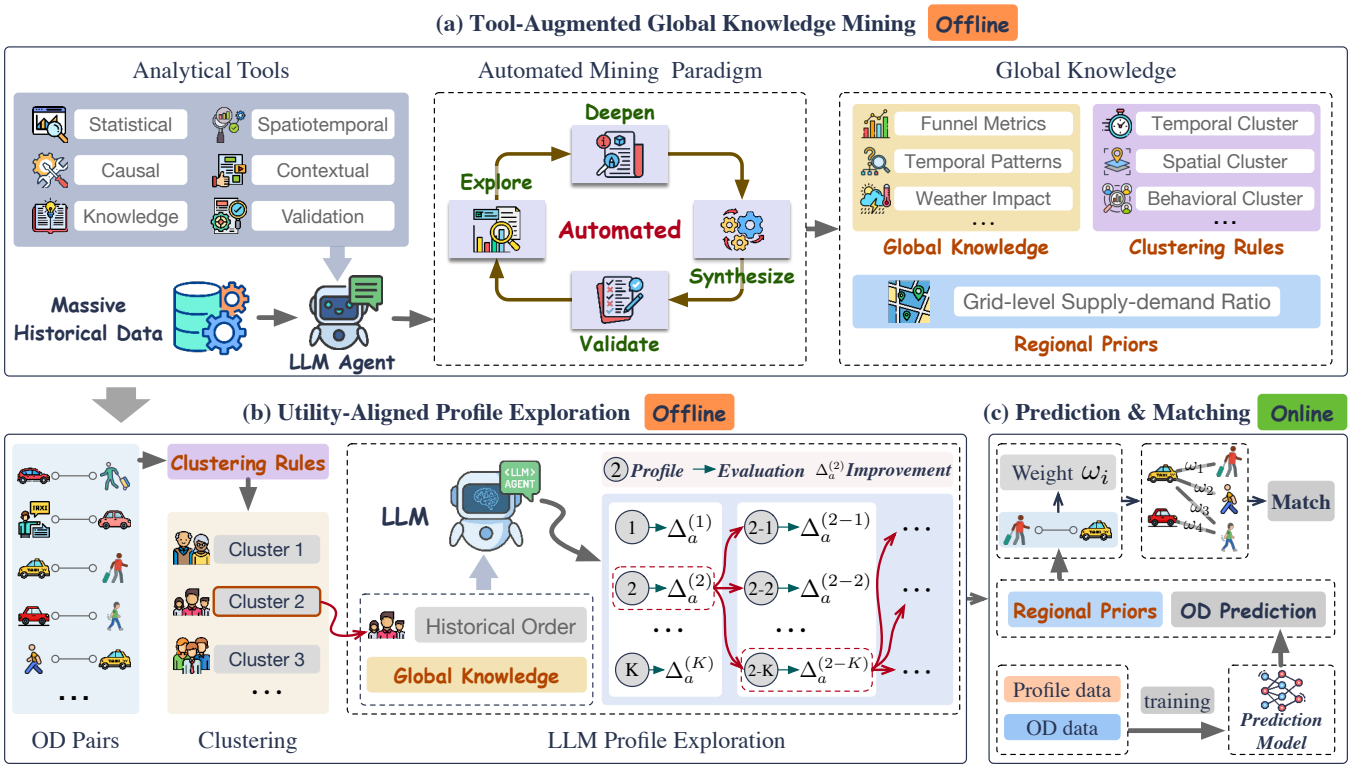


Figure 3: Overview of ProfiLLM. (a) Tool-Augmented Global Knowledge Mining and (b) Utility-Aligned Profile Exploration run offline; (c) is the online serving path.

2 PROBLEM FORMULATION

Consider a ride-hailing platform operating in a city partitioned into a set of spatial grids $\mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{G}|}\}$. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$ denote the set of passengers and $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ denote the set of drivers. Each user $u \in \mathcal{P} \cup \mathcal{D}$ has a historical interaction sequence $\mathcal{H}_u = \{h_1^u, h_2^u, \ldots, h_{|\mathcal{H}_u|}^u\}$, where each record h_i^u contains order-level information such as timestamp, location, and trip details.

At each dispatching cycle, the system receives pending orders $\mathcal{O}$ and identifies candidate matches $\mathcal{C} \subseteq \mathcal{O} \times \mathcal{D}$. For each candidate pair $c = (o, d)$ with associated passenger p, the system predicts multiple order-level outcomes (*i.e.,* order acceptance) that are used to compute matching weights. Let $\mathbf{y}_c = (y_c^{(1)}, y_c^{(2)}, \ldots, y_c^{(L)})$ denote the outcome vector, where L is the number of prediction tasks and each $y_c^{(l)} \in \{0, 1\}$ indicates whether the l-th stage in the order fulfillment process succeeds (*e.g.,* $l = 1$ for order acceptance, $l = 2$ for order completion). The prediction model estimates: $\hat{\mathbf{y}}_c = f(\mathbf{x}_c, \mathbf{e}_p, \mathbf{e}_d)$, where $\hat{\mathbf{y}}_c = (\hat{y}_c^{(1)}, \ldots, \hat{y}_c^{(L)})$ with $\hat{y}_c^{(l)} \in [0, 1]$ being the predicted probability for outcome l, $\mathbf{x}_c$ denotes structured features (ETA, distance, price) and $\mathbf{e}_p, \mathbf{e}_d$ are profile embeddings.

These predictions are aggregated into a matching weight w_c quantifying the expected utility of each assignment. The optimal matching $\mathcal{M}^*$ that maximizes $\sum_{c \in \mathcal{M}} w_c$ is then computed via the Kuhn-Munkres algorithm, subject to one-to-one matching constraints. Our work improves outcome prediction by generating utility-aligned user profiles whose embeddings $\mathbf{e}_p$ and $\mathbf{e}_d$ complement structured features.

3 METHODOLOGY

3.1 System Overview

Production setting. ProfiLLM is designed and deployed in DiDi's production dispatching service, whose operating constraints shape every design choice below. The dispatcher operates in fixed 2-second cycles; in each cycle, it enumerates candidate order–driver (OD) pairs, predicts per-pair behavioral outcomes, composes them into matching weights, and solves the assignment via the Kuhn–Munkres (KM) algorithm under a ~200 ms end-to-end latency budget. The production predictor is a multi-task deep model producing calibrated probabilities for four behavioral events along the fulfillment funnel: driver acceptance, post-acceptance driver/passenger cancellation, and completion. Since LLM inference at per-OD-pair scale is incompatible with this budget, ProfiLLM materializes a strict offline–online decoupling as the three-layer pipeline in Figure 3.

Layer 1 — Tool-Augmented Global Knowledge Mining (offline). An LLM agent equipped with 27 analytical tools mines platform-scale historical logs to produce three reusable artifacts: global behavioral knowledge $\mathcal{K}$, an interpretable user-clustering rule set $\mathcal{A}$, and regional supply–demand priors $\mathcal{R}$. This layer runs as a batch job over the log warehouse; tools operate as composable analytical operators that the agent chains autonomously under an Explore–Deepen–Validate–Synthesize paradigm.

Layer 2 — Utility-Aligned Profile Exploration (offline). For each cluster $a \in \mathcal{A}$, the LLM samples candidate profiles conditioned on $\mathcal{K}$ and the aggregated cluster history, ranks them via a

lightweight LOGIC-rule proxy of downstream prediction utility, iteratively refines the best, and constructs preference pairs that drive a one-time DPO fine-tune of the profile generator. The resulting DPO-aligned generator emits a single profile per cluster, encoded into a d-dimensional embedding $\mathbf{e}_a$ by a frozen text encoder.

Layer 3 — Online Outcome Prediction & Matching. The latency-critical dispatcher performs only two operations per OD pair: (i) a deterministic cluster-assignment rule evaluation against $\mathcal{A}$ to look up the user's cluster, and (ii) a cached embedding fetch returning $\mathbf{e}_p$ and $\mathbf{e}_d$. These embeddings are concatenated with the existing structured-feature vector and fed to the production multi-task predictor; matching weights and KM assignment proceed unchanged. *No LLM inference occurs on this path*, and combined overhead is under 0.01 ms per pair.

The offline–online contract. Layers 1–2 emit two and only two artifacts that cross into Layer 3: the cluster-assignment rule set $\mathcal{A}$ (a few KB of interpretable Boolean rules) and the cluster-embedding table $\{\mathbf{e}_a\}$. This narrow interface is the structural reason ProfiLLM operates within DiDi's existing latency budget without modifying the downstream matching components.

3.2 Tool-Augmented Global Knowledge Mining

Directly feeding massive historical logs to LLMs is infeasible due to context length limits and computational costs. To address Challenge 1, we design a tool-augmented knowledge mining module that enables an LLM agent to systematically analyze platform-scale data through structured tool invocations.

3.2.1 Tool Design and Categorization. We design a comprehensive toolkit $\mathcal{T}$ containing 27 analytical tools organized into six categories. Each tool $t_i \in \mathcal{T}$ is defined by a tuple ($name_i$, $desc_i$, $params_i$, $func_i$), where $name_i$ is the tool identifier, $desc_i$ provides a natural language description for the LLM to understand its functionality, $params_i$ specifies the required input parameters, and $func_i$ implements the actual computation logic. The tools are designed to be composable, allowing the agent to chain multiple tools to answer complex analytical questions.

3.2.2 Explore-Deepen-Validate-Synthesize Paradigm. We structure the knowledge mining process into four phases that guide the LLM agent through systematic data exploration: **(1) Explore:** The agent invokes basic statistical tools to compute platform-level metrics (*e.g.,* completion rate, acceptance rate), examine feature distributions, and surface preliminary patterns. **(2) Deepen:** Based on exploration findings, the agent conducts focused analyses on promising directions, including temporal patterns (*e.g.,* hourly cancellation variations), spatial heterogeneity (*e.g.,* grid-level supply-demand imbalances), and user segmentation (*e.g.,* clustering by historical order patterns). **(3) Validate:** Discovered patterns are subjected to statistical hypothesis testing and causal analysis; only findings with significant effect sizes and passing appropriate tests are retained. **(4) Synthesize:** Validated findings are consolidated into three structured outputs: *Global Knowledge* $\mathcal{K}$, containing platform benchmarks, temporal regularities, weather impacts, and causal relationships; *User Clustering Rules* $\mathcal{A}$, where each cluster $a \in \mathcal{A}$ is defined by a rule-based classifier $\phi_a : \mathcal{H}_u \to \{0, 1\}$ over user history (*e.g.,* active hours, frequent regions, cancellation patterns); and *Regional Priors* $\mathcal{R}$, storing grid-level and time-slot-level supply-demand statistics $P(supply, demand \mid g, s)$ with derived metrics such as expected waiting time and fulfillment rate.

3.3 Utility-Aligned Profile Exploration

With the mined global knowledge and clustering rules, we now address Challenges 2 and 3 through a cluster-level profile exploration mechanism. The key insight is that: (1) clustering aggregates sufficient historical data for reliable profiling even for low-frequency users, and (2) profile quality should be measured by downstream prediction performance rather than surface-level fluency.

3.3.1 User Clustering. Using the clustering rules $\mathcal{A}$ from the knowledge mining module, we partition all users into $|\mathcal{A}|$ clusters. For each user $u \in \mathcal{P} \cup \mathcal{D}$, we assign them to the most appropriate cluster based on their historical behavior: $a^*(u) = \arg\max_{a \in \mathcal{A}} \phi_a(\mathcal{H}_u)$, where $\phi_a(\mathcal{H}_u) \in [0, 1]$ evaluates the compatibility between user u's history and cluster a's defining characteristics. Users within the same cluster share similar behavioral patterns, allowing us to learn a shared user profile that captures cluster-level regularities.

Let $\mathcal{U}_a = \{u : a^*(u) = a\}$ denote the set of users assigned to cluster a, and $\mathcal{H}_a = \bigcup_{u \in \mathcal{U}_a} \mathcal{H}_u$ denote the aggregated historical orders for the cluster. This aggregation ensures sufficient data volume for reliable profile learning, addressing Challenge 2.

3.3.2 Structured Profile Generation. For each cluster $a \in \mathcal{A}$, we prompt the LLM to generate K diverse candidate user profiles based on the aggregated cluster history $\mathcal{H}_a$ and global knowledge $\mathcal{K}$. Each candidate profile follows a structured three-part format:

$$\{profile_a^{(k)}\}_{k=1}^{K} = \text{LLM}(\mathcal{H}_a, \mathcal{K}, prompt_{gen}) \tag{1}$$

Each profile $profile_a^{(k)}$ consists of three components: (1) **ANALYSIS**: A detailed examination of the cluster's behavioral patterns observed in the historical data, identifying key factors that influence order outcomes. (2) **PROFILE**: A semantic description characterizing the cluster's order completion patterns, behavioral preferences, and service compatibility signals (*e.g.,* efficiency-oriented drivers who target high earnings-per-effort rides). (3) **LOGIC**: Executable decision rules derived from the analysis and profile, expressed as Boolean conditions over order features (*e.g.,* `(Price_Per_KM >= 2.5) AND (Pick_KM <= 2.0)`). The prompt $prompt_{gen}$ instructs the LLM to generate diverse profiles emphasizing different aspects (*e.g.,* temporal patterns, price sensitivity, spatial preferences) while remaining grounded in the cluster's actual order history.

3.3.3 Utility-Based Profile Evaluation. A critical design choice is how to evaluate profile quality efficiently. Rather than training full prediction models for each candidate, which would be computationally prohibitive, we leverage the LOGIC component as a lightweight proxy for utility evaluation. The intuition is that if the rules extracted from a profile accurately predict outcomes, then the profile captures meaningful behavioral patterns that are also likely to benefit the embedding-based predictor.

Specifically, for each candidate profile $profile_a^{(k)}$, we extract its LOGIC rules and apply them to the cluster's historical order-driver pairs $\{(c_i, y_i)\}_{i=1}^{n_a}$ to generate rule-based predictions $\{\hat{y}_i^{(k)}\}$:

$$\hat{y}_i^{(k)} = \texttt{evaluate}(LOGIC_a^{(k)}, c_i). \tag{2}$$

We then compute a fused prediction by blending the base model's output with the LOGIC-rule prediction. Specifically, for each order-driver pair c_i in cluster a and each prediction task l (*e.g.,* driver cancellation), the base production model produces a probability estimate $\hat{y}^{(l)}_{i,\text{base}} \in [0,1]$, while the LOGIC rules yield a binary prediction $\hat{y}^{(k)}_{i,\text{logic}} \in \{0,1\}$. The fused prediction is obtained via a convex combination controlled by a blending coefficient λ:

$$\hat{y}^{(k)}_{i,\text{fused}} = (1-\lambda)\cdot \hat{y}^{(l)}_{i,\text{base}} + \lambda \cdot \hat{y}^{(k)}_{i,\text{logic}} \tag{3}$$

where $\lambda \in [0,1]$ governs the relative influence of the LOGIC rules. The fused AUC is then computed against the ground-truth labels $\{y^{(l)}_i\}_{i=1}^{n_a}$:

$$\text{AUC}^{(k)}_a = \text{ComputeAUC}\left(\{y^{(l)}_i\}_{i=1}^{n_a},\ \{\hat{y}^{(k)}_{i,\text{fused}}\}_{i=1}^{n_a}\right). \tag{4}$$

The utility gain of profile k is measured as the AUC improvement over the base production model alone: $\Delta^{(k)}_a = \text{AUC}^{(k)}_a - \text{AUC}^{\text{base}}_a$, where $\text{AUC}^{\text{base}}_a = \text{ComputeAUC}(\{y^{(l)}_i\},\ \{\hat{y}^{(l)}_{i,\text{base}}\})$ is AUC achieved by the base model without LOGIC-rule augmentation. A positive $\Delta^{(k)}_a$ indicates that the LOGIC rules provide complementary signals that improve prediction beyond the existing production model.

This fused evaluation serves as an efficient and informative proxy for profile quality. Since the base model already captures patterns available from structured features, a positive $\Delta^{(k)}_a$ directly indicates that the LOGIC rules encode complementary behavioral signals invisible to the production predictor. Moreover, the blending mechanism ensures that only profiles contributing genuine discriminative power beyond the existing model are favored, filtering out superficially plausible descriptions that merely recapitulate what structured features already capture, thereby reliably identifying profiles most beneficial for downstream outcome prediction.

3.3.4 Iterative Profile Refinement. We adopt an iterative refinement strategy to progressively improve profile quality. Starting from the best-performing initial candidate $profile^{(k^*)}_a$ where $k^* = \arg\max_k \Delta^{(k)}_a$, we prompt the LLM to generate refined versions that address identified weaknesses:

$$profile^{(t+1)}_a = \text{LLM}(profile^{(t)}_a, \mathcal{H}_a, feedback^{(t)}, prompt_{refine}) \tag{5}$$

where $feedback^{(t)}$ summarizes the prediction errors made by the current profile's LOGIC rules (*e.g.,* false positive/negative cases with their order features). The refinement continues for T iterations or until the utility gain plateaus, yielding the selected profile $profile^*_a = \arg\max_{t\in\{1,\ldots,T\}} \Delta^{(t)}_a$.

3.3.5 DPO Fine-tuning for Profile Generation. The exploration process naturally produces preference pairs that can be leveraged for DPO fine-tuning. For each cluster a, we construct preference pairs by comparing profiles with different utility gains:

$$\mathcal{P}_a = \{(\mathcal{H}_a, profile_w, profile_l) : \Delta^{(w)}_a > \Delta^{(l)}_a + \gamma\} \tag{6}$$

where γ is a margin threshold ensuring meaningful preference differences. Aggregating across all clusters yields a preference dataset $\mathcal{P} = \bigcup_{a\in\mathcal{A}} \mathcal{P}_a$. We perform a one-time offline DPO fine-tuning of the LLM using the standard DPO objective [28] on $\mathcal{P}$. This fine-tuning aligns the LLM's profile generation capability with downstream prediction utility, improving the quality of profiles for newly defined clusters without requiring additional exploration iterations.

Note that all profiles describe cluster-level behavioral patterns (*e.g.,* temporal preferences, price sensitivity, cancellation tendencies) rather than individual demographic attributes, ensuring that the profiling mechanism operates on aggregate behavioral signals without raising privacy concerns.

3.4 Online Outcome Prediction and Matching

3.4.1 Profile Embedding and Caching. Each cluster's textual PROFILE component is converted into a dense embedding via a pre-trained text encoder $\mathcal{E}$:$\mathbf{e}_a = \mathcal{E}(PROFILE^*_a) \in \mathbb{R}^d$. For each user u, their profile embedding is simply the embedding of their assigned cluster: $\mathbf{e}_u = \mathbf{e}_{a^*(u)}$. Since profiles are at the cluster level, the number of embeddings to cache is bounded by $|\mathcal{A}|$, making caching highly efficient. All cluster embeddings are pre-computed and stored in a distributed cache with sub-millisecond lookup latency.

3.4.2 Online Feature Integration and Matching. At serving time, for each candidate order-driver pair $c = (o,d)$ with passenger p, we construct the input representation by concatenating structured features with the cached profile embeddings: $\mathbf{z}_c = [\mathbf{x}_c; \mathbf{e}_p; \mathbf{e}_d; \mathbf{r}_{g,s}]$, where $\mathbf{x}_c$ denotes the original structured features, $\mathbf{e}_p$ and $\mathbf{e}_d$ are the profile embeddings retrieved from cache based on user-to-cluster assignments, and $\mathbf{r}_{g,s} \in \mathcal{R}$ is the regional supply-demand prior for grid g and time slot s. The multi-task prediction model f_θ jointly predicts order-level outcomes $\hat{\mathbf{y}}_c = f_\theta(\mathbf{z}_c)$, which are then aggregated into a matching weight w_c quantifying the expected utility of each assignment. The regional prior $\mathcal{R}$ is incorporated to adjust weights based on supply-demand conditions. Finally, the optimal matching $\mathcal{M}^*$ that maximizes $\sum_{c\in\mathcal{M}} w_c$ is computed via the Kuhn-Munkres algorithm, completing one dispatching cycle.

Since no LLM inference occurs at serving time and user-to-cluster assignment is a simple rule evaluation, the latency from profile is negligible. For new users without sufficient history, the system assigns them to a default cluster, ensuring consistent service quality.

4 EXPERIMENTS

We conduct extensive experiments on real-world industrial datasets from DiDi's ride-hailing platform to evaluate ProfiLLM. Due to space, extended analyses: the dispatching simulator design, discovered cluster archetypes, cluster-count and λ sensitivity, offline cost and complexity, the 14-day stability study, prompt templates, are deferred to our online appendix.[1]

4.1 Experimental Setup

4.1.1 Datasets. We evaluate ProfiLLM on real-world order dispatching from three Brazilian cities spanning distinct supply–demand regimes: **City A** (medium-scale, supply-constrained, highest driver utilization), **City B** (medium-scale, supply-relaxed, moderate order density), and **City C** (large-scale, high-demand, complex traffic). For each city, we use 38 days of historical data for training (including global knowledge mining and profile exploration) and

[1] https://ProfiLLM.github.io

Table 1: Relative improvement (%) over pickup-distance-based KM matching across three cities and time periods. † Time periods: Morning = 7:00–10:00, Noon = 11:00–14:00, Evening = 17:00–20:00.

City	Group	Method	Overall		Morning†		Noon†		Evening†	
			GMV	CR	GMV	CR	GMV	CR	GMV	CR
City A	*Trad.*	TVal	+2.24	+2.14	+3.86	+2.44	+1.70	+1.08	+2.06	+1.41
		GRC	+0.73	−3.42	+4.18	+0.62	+3.01	+0.22	−2.36	−3.97
	LLM	Llama-3.3-70B	+2.34	+2.76	+0.84	+1.49	+1.02	+3.00	+2.21	+7.12
		Qwen3-Next-80B	+2.41	+2.54	+1.58	+1.54	+1.80	+3.89	+2.80	+6.29
		DeepSeek-R1	+2.53	+4.57	+1.40	+2.17	+1.35	+3.62	+2.60	+7.61
		Kimi-K2	+1.96	+4.77	+2.04	+1.78	+0.93	+3.34	+2.43	+8.15
		GPT-OSS-120B	+2.44	+5.75	+0.75	+1.65	+1.95	+4.62	+0.05	+6.96
		Gemini-3-Flash	+1.41	+4.62	+0.34	+1.01	+1.57	+4.20	−0.26	+5.63
		Gemini-3-Pro	+2.95	+5.48	+3.96	+4.72	+1.74	+3.50	+1.62	+5.93
	Ours	**ProfiLLM-DPO**	**+4.02**	+6.03	+4.97	+5.14	+3.01	+0.22	+2.82	**+9.94**
		ProfiLLM	+3.52	**+7.10**	**+5.02**	**+5.67**	**+3.20**	**+4.98**	**+3.19**	+6.97
City B	*Trad.*	TVal	+1.87	+1.63	+3.12	+1.98	+1.24	+0.76	+1.72	+1.05
		GRC	+1.15	−2.18	+3.54	+0.38	+2.47	−0.15	−1.82	−3.25
	LLM	Llama-3.3-70B	+1.92	+2.31	+0.56	+1.12	+0.78	+2.54	+1.87	+6.38
		Qwen3-Next-80B	+2.08	+2.12	+1.24	+1.18	+1.45	+3.25	+2.42	+5.67
		DeepSeek-R1	+2.17	+3.89	+1.08	+1.82	+1.02	+3.08	+2.24	+6.83
		Kimi-K2	+1.63	+4.05	+1.72	+1.45	+0.68	+2.87	+2.08	+7.34
		GPT-OSS-120B	+2.06	+5.12	+0.48	+1.32	+1.62	+4.08	−0.18	+6.24
		Gemini-3-Flash	+1.08	+3.94	+0.12	+0.78	+1.24	+3.65	−0.48	+4.89
		Gemini-3-Pro	+2.51	+4.83	+3.42	+4.15	+1.38	+2.94	+1.28	+5.18
	Ours	**ProfiLLM-DPO**	**+3.58**	+5.47	+4.42	+4.68	+2.64	−0.08	+2.48	**+9.12**
		ProfiLLM	+3.14	**+6.52**	**+4.56**	**+5.24**	**+2.82**	**+4.35**	**+2.84**	+6.28
City C	*Trad.*	TVal	+2.56	+2.48	+4.24	+2.82	+1.92	+1.34	+2.38	+1.72
		GRC	+0.41	−1.87	+2.86	+0.54	+1.78	+0.08	−1.54	−2.83
	LLM	Llama-3.3-70B	+2.68	+3.12	+1.12	+1.78	+1.34	+3.42	+2.54	+7.56
		Qwen3-Next-80B	+2.75	+2.89	+1.82	+1.76	+2.04	+4.12	+3.12	+6.72
		DeepSeek-R1	+2.91	+4.93	+1.64	+2.48	+1.58	+3.94	+2.92	+7.98
		Kimi-K2	+2.24	+5.18	+2.36	+2.04	+1.12	+3.68	+2.72	+8.52
		GPT-OSS-120B	+2.79	+6.08	+0.98	+1.86	+2.18	+4.84	+0.32	+7.38
		Gemini-3-Flash	+1.72	+4.95	+0.58	+1.24	+1.82	+4.48	+0.08	+5.42
		Gemini-3-Pro	+3.28	+5.81	+4.28	+5.04	+1.98	+3.72	+1.92	+6.28
	Ours	**ProfiLLM-DPO**	**+4.35**	+6.41	+5.24	+5.48	+3.38	+0.48	+3.14	**+10.32**
		ProfiLLM	+3.87	**+7.53**	**+5.38**	**+5.92**	**+3.54**	**+5.24**	**+3.48**	+7.42

five days for testing. The City A analysis window covers 333,166 active passengers and 12,128 active drivers, with the long-tail and heterogeneity statistics applying directly to this evaluation set.

4.1.2 Evaluation Metrics. We evaluate performance at two levels. *(i) Dispatching level (realized rates).* We report **GMV** (Gross Merchandise Value) together with six realized rates computed in the dispatching simulator or the online A/B test: **CR** (Completion Rate), **DAR** (Driver Acceptance Rate), **DCR** (Driver Cancellation Rate, post-acceptance driver cancel), **PCR** (Passenger Cancellation Rate, post-acceptance passenger cancel), **CBA** (Cancel Before Accept, share of orders the passenger cancels before any driver accepts), and **BER** (Bad Experience Rate, share of completed orders with excessively long pickup distance). *(ii) Prediction level (AUC).* For each candidate OD pair, the prediction model produces a probability for each of four behavioral events. We report **AUC** for each task, denoted **Accept**, **D-Cancel**, **P-Cancel**, and **Success**, in one-to-one correspondence with DAR, DCR, PCR, and CR. The two views measure the same four events at different stages of the pipeline and can therefore move differently when a system change affects matching weights more than per-pair prediction (or vice versa). CBA and BER are pure dispatching-level rates without a direct prediction-task counterpart, since CBA occurs before any OD-pair assignment is made and BER reflects realized pickup quality after matching.

4.1.3 Baselines. We compare ProfiLLM against two categories of baselines. **Traditional dispatching methods** rely on structured features without user profiling: TVal [42] and GRC [44]. **LLM-based profiling methods** use the same clustering and knowledge mining pipeline as ProfiLLM but differ in the LLM backbone for profile generation, including Llama-3.3-70B [5], Qwen3-Next-80B [43], DeepSeek-R1 [8], Kimi-K2 [35], Gemini-3-Flash [34], Gemini-3-Pro [34], and GPT-OSS-120B. **ProfiLLM** with exploration-selected profiles but without DPO fine-tuning, and **ProfiLLM-DPO** with the full framework including DPO-aligned profile generation.

4.1.4 Implementation Details. For the global knowledge mining, we use Gemini-3-Pro as the backbone LLM agent with a temperature of 0.3 for consistent tool invocation. For profile exploration, we generate $K = 5$ initial candidate profiles per cluster and perform $T = 3$ refinement iterations. The margin threshold γ for constructing DPO preference pairs is set to 0.001 AUC improvement. We fine-tune Qwen3-8B via DPO on the collected preference pairs as the aligned profile generator. The text encoder $\mathcal{E}$ for embedding user profiles is a fine-tuned sentence transformer with output dimension $d = 768$. All offline experiments are conducted on a cluster with 8 NVIDIA L20 GPUs. Dispatching quality is evaluated with a replay-based simulator that re-runs historical orders and driver availability per city in production-identical 2-second Kuhn–Munkres cycles, sampling Accept/Cancel outcomes from the multi-task predictor. Online serving operates on pre-computed cluster embeddings stored in a distributed cache, ensuring sub-millisecond lookup latency with negligible overhead to the existing prediction pipeline.

4.2 Overall Performance

Table 1 presents the simulator-based evaluation results across three cities. From the results, we make the following observations. (1)

Table 2: Multi-task prediction AUC improvement (%) over Structured Only baseline. Higher is better. Best results are in bold.

Method	City A				City B				City C			
	Accept	D-Cancel	P-Cancel	Success	Accept	D-Cancel	P-Cancel	Success	Accept	D-Cancel	P-Cancel	Success
Llama-3.3-70B	-1.10	-0.71	+0.19	-1.14	-0.64	+0.38	-0.38	-0.45	-0.01	-0.34	+0.25	+0.01
Qwen3-Next-80B	-0.22	-0.38	+1.65	+0.02	-0.52	-0.40	-5.71	-7.57	-0.03	-0.16	+0.27	-0.06
DeepSeek-R1	+0.06	+0.23	+2.05	+0.25	+0.31	+1.85	+1.06	+0.48	+0.21	-0.13	+0.04	+0.14
Kimi-K2	-0.17	+0.82	+2.11	-0.07	-2.44	-0.44	-6.33	-1.91	+0.50	-0.11	+0.40	+0.45
Gemini-3-Flash	+0.10	+0.53	+1.83	+0.42	+0.24	+1.76	-0.11	+0.38	+0.03	-0.26	+0.37	+0.04
Gemini-3-Pro	-0.08	-0.68	+2.37	+0.56	-0.44	+0.50	+0.24	-0.31	+0.02	-0.03	+0.10	+0.05
GPT-OSS-120B	-0.02	+0.14	+1.83	+0.17	+0.11	+1.64	+0.63	+0.29	-0.09	-0.02	+0.44	-0.06
ProfiLLM-DPO	+1.51	+2.76	+6.02	+1.72	+2.25	+4.98	+5.55	+2.58	+0.65	+5.93	+5.30	+2.37
ProfiLLM	+1.56	+3.88	+6.14	+1.80	+2.26	+4.98	+6.00	+2.60	+0.84	+5.95	+5.65	+2.48

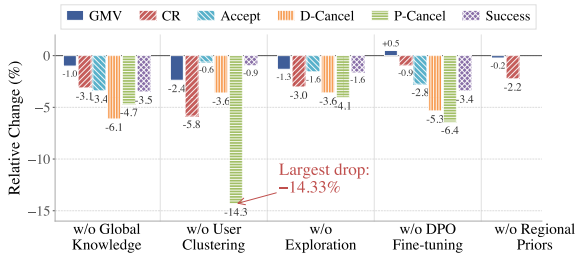


Figure 4: Ablation on City A. *CR* is the simulator's realized completion rate; *Success* is the per-OD-pair completion AUC. Regional Priors only affect matching weights, hence no prediction-AUC bar.

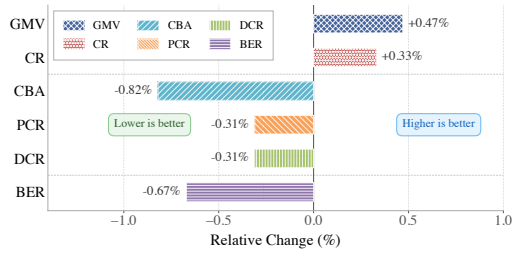


Figure 5: Online A/B over 14 days in City A: treatment-vs.-control relative change.

ProfiLLM variants consistently outperform all baselines. Both ProfiLLM and ProfiLLM-DPO achieve the highest GMV and CR among all methods. Notably, the two variants exhibit complementary strengths: ProfiLLM-DPO achieves the best GMV (*e.g.,* +4.35% in City C), while ProfiLLM attains the best CR (*e.g.,* +7.53% in City C). This suggests that DPO fine-tuning steers profiles toward revenue-critical signals at a slight cost to CR, whereas exploration-selected profiles maintain a more balanced optimization across the fulfillment funnel. (2) LLM-based profiles substantially outperform traditional methods. TVal and GRC, which rely solely on structured numerical features, achieve moderate GMV gains but show inconsistent CR performance (*e.g.,* GRC yields negative CR in all three cities). In contrast, all LLM-based methods deliver positive improvements on both metrics, confirming that semantic user profiles capture behavioral patterns invisible to handcrafted features. (3) Utility alignment matters more than model scale. Among LLM-based methods, larger backbones generally yield better profiles, yet our DPO-aligned Qwen3-8B consistently outperforms much larger models such as Gemini-3-Pro without alignment. This demonstrates that optimizing for downstream prediction utility is more effective than scaling the LLM backbone alone.

4.3 Prediction Performance

To evaluate the quality of LLM-generated profiles for outcome prediction, we compare the AUC of multi-task prediction models trained with profiles from different methods. Table 2 reports the AUC improvement over the structured-only baseline for prediction tasks. The results reveal that naively applying LLMs for profile generation does not guarantee downstream utility. While some baseline LLMs achieve modest positive improvements, others actually degrade prediction performance: Kimi-K2 drops P-Cancel AUC by 6.33% in City B, and Qwen3-Next-80B yields a 7.57% decrease in Success. In contrast, both ProfiLLM and ProfiLLM-DPO deliver consistent positive gains across all tasks and cities, with ProfiLLM achieving up to +6.14% on P-Cancel (City A), +5.95% on D-Cancel (City C), and +2.60% on Success (City B). The largest improvements are observed for cancellation prediction, which aligns with our pilot study: cancellation behaviors depend heavily on contextual preferences such as sensitivity to pickup distance that handcrafted features struggle to capture. Finally, ProfiLLM and ProfiLLM-DPO achieve comparable prediction AUC, with ProfiLLM slightly ahead on most metrics, suggesting that iterative exploration is the primary driver of prediction gains, while DPO fine-tuning trades a small amount of per-cluster local optimality for cross-cluster generalization and substantially lower offline refresh cost.

4.4 Ablation Study

Figure 4 reports results on City A as relative change compared to the full ProfiLLM. (1) **w/o Global Knowledge:** Removing global knowledge causes substantial drops across all tasks (*e.g.,* D-Cancel −6.12%), confirming that platform-level knowledge provides essential grounding for generating predictive profiles. (2) **w/o User Clustering:** Replacing cluster-level profiles with per-user profiles yields the largest degradation (P-Cancel −14.33%), as 96% of passengers lack sufficient history for reliable individual profiling. (3) **w/o Exploration:** Skipping iterative refinement degrades both dispatching (GMV −1.32%) and prediction, confirming that the explore-evaluate-refine loop is necessary for utility alignment. (4) **w/o DPO Fine-tuning:** We replace the DPO-aligned generator with the un-finetuned base Qwen3-8B in a single-pass setting (no exploration, no alignment). Prediction AUC drops substantially (P-Cancel −6.42%), confirming that preference alignment is what makes the compact 8B generator competitive. Note that this ablation differs from the ProfiLLM variant in Table 2, which uses Gemini-3-Pro with exploration but no DPO; that path achieves comparable AUC to ProfiLLM-DPO at higher offline LLM cost. (5) **w/o Regional Priors:** Primarily impacts dispatching (CR −2.24%) without affecting prediction, consistent with priors being incorporated into matching weights rather than the prediction model.

Table 3: Offline pipeline cost breakdown in City A. DPO refresh reduces total cost by 10.6×.

Variant	LLM Calls	Wall Time	GPU-h	Cost
ProfiLLM (initial)	∼1,460	∼6.3 hrs	1.4	$54.63
ProfiLLM-DPO (refresh)	96	∼1.8 hrs	1.4	**$5.13**

4.5 Production Deployment

We deployed ProfiLLM in DiDi's production environment and report results from a 14-day A/B test in City A (extending the initial 5-day pilot to a longer window for more stable estimates). As shown in Figure 5, ProfiLLM achieves consistent improvements across every monitored realized rate (see Section 4.1.2 for definitions). Revenue and completion improve simultaneously: GMV rises by 0.47% and CR by 0.33%, confirming that the gains stem from higher matching quality rather than from trading off completion for revenue. Post-acceptance cancellations decrease on both sides of the match: PCR drops by 0.31% and DCR by 0.31%. The two non-prediction funnel rates also move in the desired direction: CBA drops by 0.82%, indicating fewer passengers walk away before a driver accepts, and BER drops by 0.67%, indicating fewer completed orders end up with poor pickup quality. The simultaneous beneficial movement across mechanistically distinct funnel stages provides converging evidence that ProfiLLM systematically improves the dispatching components it was designed to enhance. These online deltas are several times smaller than the simulation gains of Section 4.2 (*e.g.*, +0.47% vs. +4.02% GMV in City A); this gap is expected by construction, so we read the simulator for effect *sign* and method *ranking* rather than absolute magnitude (validated in our online appendix).

4.6 Production Cost and Latency

ProfiLLM's affordability at platform scale rests on a cost structure (Table 3) that shrinks at every layer. The profiling cost is first amortized by aggregation, as a single offline run yields 96 cluster profiles that cover all 348,464 City A users (348,464/96 ≈ 3,630× fewer profiles than per-user profiling). DPO then drives this offline cost down further, since after a one-time per-city training each routine refresh needs only 96 single-pass LLM calls, cutting refresh cost from $54.63 to $5.13 (**10.6**×). At serving time the marginal cost all but vanishes, as the dispatcher runs only a deterministic cluster-rule evaluation (< 0.01 ms) and a cached embedding lookup (< 0.001 ms) per OD pair, with *zero* online LLM inference, comfortably within DiDi's 200 ms budget. These compounding savings place ProfiLLM-DPO at the Pareto frontier of the cost–quality trade-off (Figure 6), where it dominates five of seven baseline LLMs.

5 RELATED WORK

Ride-hailing Order Dispatching. Order dispatching has evolved from proximity-based greedy matching [17, 36, 47, 48] to RL methods capturing long-term dynamics: demand–supply forecasting with combinatorial optimization [42], mean-field multi-agent RL [27], and cooperative Markov games [44]. Prediction-side work models driver acceptance [39] and cancellations [3]. These rely on structured numerical features, leaving semantic factors unexploited; we introduce LLM-generated profiles that capture them while preserving real-time compatibility via offline–online decoupling.

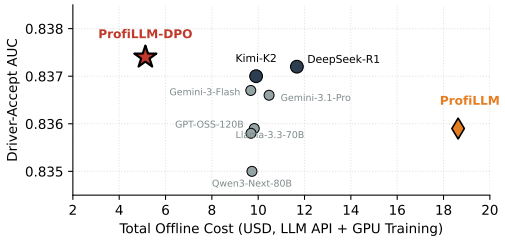


Figure 6: Cost–quality trade-off across nine LLM backbones on City A (offline cost vs. driver-accept AUC).

LLM Data Pipelines for Applied ML. Recent VLDB work operationalizes LLMs in production data systems. Closest to our setting, LEADRE [18] deploys DPO-aligned LLMs in Tencent's display-advertising pipeline, and SiriusBI [14] productionizes multi-round NL2SQL for enterprise BI. On the agentic side, DocETL [31], AutoPrep [7], SQL-Factory [19], LLM-AutoDP [12], and DBAIOps [52] rewrite or compose LLM-agent operators for document, tabular, SQL, and database tasks; KATS [40] pairs an offline LLM knowledge graph with online retrieval. Unlike these, ProfiLLM is the first to bring an agentic LLM data pipeline into a real-time ride-hailing dispatcher (2-second cycle, sub-millisecond online budget), by confining all LLM reasoning offline and serving only cluster-level profile embeddings.

LLM-based Agents and User Profiling. Tool-augmented LLM agents extend capabilities via tools (ReAct [45], Toolformer [29], and Data-Copilot [50]) and have been applied to traffic analysis [49], though mainly for general insights rather than production prediction. We instead design a systematic tool-augmented mining workflow that yields actionable knowledge directly improving prediction. Separately, user behavior modeling spans sequential models [15, 32], deep interest networks [25, 51], and recent LLM-based profiling [20, 38, 41]; in ride-hailing, prior work estimates driver value under behavioral heterogeneity [33] and models passenger demand [16]. RLHF [2, 23] aligns LLMs via reward modeling and PPO [30] but is complex to train; DPO [28] optimizes directly from preference pairs, with many extensions [1, 6, 11] and task-specific variants [4, 24, 46]. We contribute a novel application: constructing preference pairs from task rather than human judgment, aligning profile generation with prediction accuracy.

6 CONCLUSION

We presented ProfiLLM, a practical framework that bridges LLM-based semantic profiling with real-time ride-hailing order matching. ProfiLLM addresses three key challenges through two synergistic modules: *Tool-Augmented Global Knowledge Mining*, which equips an LLM agent with 27 analytical tools to extract global knowledge, clustering rules, and regional priors; and *Utility-Aligned Profile Exploration*, which iteratively refines cluster-level profiles via a lightweight prediction proxy and DPO fine-tuning. By confining all LLM inference offline and serving only pre-computed embeddings, ProfiLLM adds sub-millisecond latency. Experiments on DiDi's platform across three cities demonstrate up to +6.14% relative AUC improvement and +4.35% GMV gain in simulation, with a 14-day production A/B test confirming +0.47% GMV, +0.33% Completion Rate (CR), and −0.82% Cancel-Before-Accept rate (CBA).

REFERENCES

[1] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4447–4455.

[2] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[3] M Keith Chen and Michael Sheldon. 2016. Dynamic pricing in a labor market: Surge pricing and flexible work on the Uber platform. *Ec* 16 (2016), 455.

[4] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335* (2024).

[5] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints* (2024), arXiv–2407.

[6] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306* (2024).

[7] Meihao Fan, Ju Fan, Nan Tang, Lei Cao, Guoliang Li, and Xiaoyong Du. 2025. AutoPrep: Natural Language Question-Aware Data Preparation with a Multi-Agent Framework. *Proc. VLDB Endow.* 18, 10 (2025), 3504–3517. https://doi.org/10.14778/3748191.3748211

[8] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[9] Balázs Hidasi and Ádám Tibor Czapp. 2023. Widespread Flaws in Offline Evaluation of Recommender Systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. https://doi.org/10.1145/3604915.3608839

[10] David Holtz and Sinan Aral. 2020. Limiting Bias from Test-Control Interference in Online Marketplace Experiments. *arXiv preprint arXiv:2004.12162* (2020).

[11] Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691* (2024).

[12] Wei Huang, Anda Cheng, Yinggui Wang, Lei Wang, and Tao Wei. 2026. LLM-AutoDP: Automatic Data Processing via LLM Agents for Model Fine-tuning. *Proc. VLDB Endow.* 19, 5 (2026), 794–807. https://doi.org/10.14778/3796195.3796196

[13] Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, et al. 2025. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096* (2025).

[14] Jie Jiang, Haining Xie, Siqi Shen, Yu Shen, Zihan Zhang, Meng Lei, Yifeng Zheng, Yang Li, Chunyou Li, Danqing Huang, Yinjun Wu, Wentao Zhang, Bin Cui, and Peng Chen. 2025. SiriusBI: A Comprehensive LLM-Powered Solution for Data Analytics in Business Intelligence. *Proc. VLDB Endow.* 18, 12 (2025), 4860–4873. https://doi.org/10.14778/3750601.3750610

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[16] Jintao Ke, Feng Xiao, Hai Yang, and Jieping Ye. 2020. Learning to delay in ride-sourcing systems: A multi-agent deep reinforcement learning framework. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2280–2292.

[17] Der-Horng Lee, Hao Wang, Ruey Long Cheu, and Siew Hoon Teo. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record* 1882, 1 (2004), 193–200.

[18] Fengxin Li, Yi Li, Yue Liu, Chao Zhou, Yuan Wang, Xiaoxiang Deng, Wei Xue, Dapeng Liu, Lei Xiao, Haijie Gu, Jie Jiang, Hongyan Liu, Biao Qin, and Jun He. 2025. LEADRE: Multi-Faceted Knowledge Enhanced LLM Empowered Display Advertisement Recommender System. *Proc. VLDB Endow.* 18, 12 (2025), 4763–4776. https://doi.org/10.14778/3750601.3750602

[19] Jiahui Li, Tongwang Wu, Yuren Mao, Yunjun Gao, Yajie Feng, and Huaizhong Liu. 2025. SQL-Factory: A Multi-Agent Framework for High-Quality and Large-Scale SQL Generation. *Proc. VLDB Endow.* 19, 3 (2025), 292–305. https://doi.org/10.14778/3778092.3778093

[20] Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 452–461.

[21] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[22] Yansong Ning, Shuowei Cai, Wei Li, Jun Fang, Naiqiang Tan, Hua Chai, and Hao Liu. 2025. Dima: An llm-powered ride-hailing assistant at didi. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4728–4739.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[24] Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems* 37 (2024), 116617–116637.

[25] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2685–2692.

[26] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *The twelfth international conference on learning representations*.

[27] Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Fan Zhang, Zhe Xu, Hongtu Zhu, and Jieping Ye. 2020. Ride-hailing order dispatching at didi via reinforcement learning. *INFORMS Journal on Applied Analytics* 50, 5 (2020), 272–286.

[28] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems* 36 (2023), 53728–53741.

[29] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* 36 (2023), 68539–68551.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[31] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048. https://doi.org/10.14778/3746405.3746426

[32] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[33] Xiaocheng Tang, Fan Zhang, Zhiwei Qin, Yansheng Wang, Dingyuan Shi, Bingchen Song, Yongxin Tong, Hongtu Zhu, and Jieping Ye. 2021. Value function is all you need: A unified learning framework for ride hailing platforms. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3605–3615.

[34] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[35] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534* (2025).

[36] Jing-Peng Wang, Hai Wang, Peng Liu, and Hai-Jun Huang. 2025. Order dispatching strategy and pricing scheme in ride-sourcing markets with consideration of service cancellation. *Transportation Research Part B: Methodological* 199 (2025), 103266.

[37] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (2024), 186345.

[38] Lu Wang, Di Zhang, Fangkai Yang, Pu Zhao, Jianfeng Liu, Yuefeng Zhan, Hao Sun, Qingwei Lin, Weiwei Deng, Dongmei Zhang, et al. 2025. Lettingo: Explore user profile generation for recommendation system. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 2985–2995.

[39] Yansheng Wang, Yongxin Tong, Cheng Long, Pan Xu, Ke Xu, and Weifeng Lv. 2019. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 1478–1489.

[40] Zixin Wei, Yucan Guo, Jinyang Li, Xiaolin Han, Xiaolong Jin, and Chenhao Ma. 2026. Revisiting Task-Oriented Dataset Search in the Era of Large Language Models: Challenges, Benchmark, and Solution. *Proc. VLDB Endow.* 19, 5 (2026), 973–986. https://doi.org/10.14778/3796195.3796209

[41] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 12–22.

[42] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 905–913.

[43] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388* (2025).

[44] Zhaoxing Yang, Haiming Jin, Guiyun Fan, Min Lu, Yiran Liu, Xinlang Yue, Hao Pan, Zhe Xu, Guobin Wu, Qun Li, et al. 2024. Rethinking order dispatching in online ride-hailing platforms. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3863–3873.

[45] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

[46] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

[47] Xinlang Yue, Yiran Liu, Fangzhou Shi, Sihong Luo, Chen Zhong, Min Lu, and Zhe Xu. 2024. An End-to-End Reinforcement Learning Based Approach for Micro-View Order-Dispatching in Ride-Hailing. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 5054–5061.

[48] Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. 2017. A taxi order dispatch model based on combinatorial optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2151–2159.

[49] Siyao Zhang, Daocheng Fu, Wenzhe Liang, Zhao Zhang, Bin Yu, Pinlong Cai, and Baozhen Yao. 2024. Trafficgpt: Viewing, processing and interacting with traffic foundation models. *Transport Policy* 150 (2024), 95–105.

[50] Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2023. Data-copilot: Bridging billions of data and humans with autonomous workflow. *arXiv preprint arXiv:2306.07209* (2023).

[51] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1059–1068.

[52] Wei Zhou, Peng Sun, Xuanhe Zhou, Qianglei Zang, Ji Xu, Tieying Zhang, Guoliang Li, and Fan Wu. 2026. DBAIOps: A Reasoning LLM-Enhanced Database Operation and Maintenance System using Knowledge Graphs. *Proc. VLDB Endow.* 19, 6 (2026), 1319–1331. https://doi.org/10.14778/3797919.3797937

A EMPIRICAL MOTIVATION: USER BEHAVIORAL HETEROGENEITY

This appendix supplements Section 1 by quantifying *how much* of order-outcome variance is governed by stable user-level traits that are invisible to structured per-order features, motivating profiling as a complementary information source. All analyses use 38 days of City A production logs (44.3M dispatching records, 333,166 passengers, 12,128 active drivers).

Structured features leave large unexplained behavioral variance. We partition grabbed orders into 100 buckets defined by (fee quintile × ETA quintile × time-of-day), so orders within a bucket share nearly identical structured features. Within these matched buckets the PCR still varies substantially across users: the average within-bucket standard deviation is **24.9%**, and the P90–P10 spread reaches **37.1 percentage points**. A variance-component (ICC) analysis on the 31,160 passengers with ≥ 20 orders attributes **15.8%** of PCR variance to passenger identity alone: a stable, systematic signal that per-order features cannot expose.

Driver and passenger decile gaps under matched contexts. Figure 7 reports complementary cuts of the data. Panel (a) compares the most- vs. least-cancellation-prone passengers among those with ≥ 5 grabbed orders: despite virtually identical average order fee (19.8 vs. 19.0) and ETA (293s vs. 292s), the former's PCR reaches 32.6% while the latter's is 0.0%, a 32.6 p.p. gap. Panel (b) plots DAR percentiles across the 12,128 active drivers; the P90/P10 ratio is **8.2**× with $\sigma = 12.2\%$, far beyond what order-level features can account for. Panel (c) shows that driver cancellation behavior is similarly bimodal: top-decile DCR (38.1%) is **10.8**× the bottom decile (3.5%) under matched order conditions. Panel (d) quantifies the same effect at dispatching-round granularity: in 16.3% of 11.8M dispatching rounds (1.93M rounds) the same broadcasted order receives mixed outcomes from different drivers (some accept, others reject), and an extreme observed case had 18 of 19 drivers reject a single order with fee 17 and ETA 384s. These patterns are stable individual traits that profiling is designed to capture.

Heterogeneity is largely orthogonal to order-level features. Beyond decile-level summaries, we observe systematic individual preferences such as ETA-sensitivity heterogeneity (9.0% of 42,865 multi-ETA passengers exhibit >30 p.p. PCR swings across ETA bins; one passenger with 50 orders has 0% PCR at ETA 4–6 min but 100% at ETA >8 min) and driver price-tier selectivity (11.3% of 10,687 multi-tier drivers show ≥20 p.p. DAR gap between high- and low-price tiers, with a smaller 7.1% group exhibiting the opposite preference). The pilot study in Section 1 confirms that LLM-generated profiles convert these heterogeneity signals into 3.71% and 9.64% relative AUC gains on driver and passenger cancellation respectively, on top of a mature structured-feature predictor. Together, these analyses establish that user-side variance is large, systematic, and complementary to per-order features, precisely the gap ProfiLLM is designed to close.

Long-tail prevalence amplifies the need for clustering. Of the 333,166 unique passengers in the analysis window, **44.9%** appear in ≤ 3 orders and **59.3%** in ≤ 5 orders (consistent with Figure 2). For these users no individual-level behavioral signal can be reliably estimated,

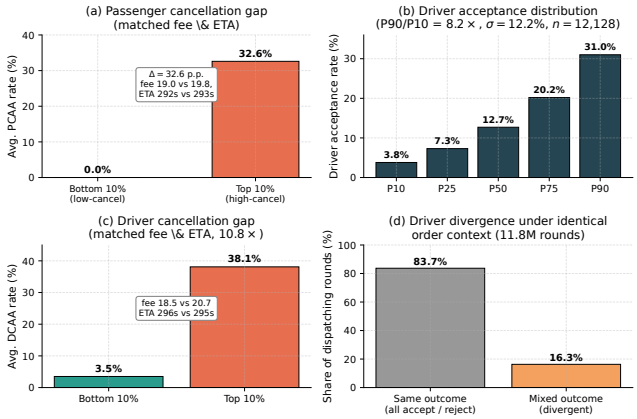


Figure 7: Behavioral heterogeneity invisible to structured features. (a) Passenger PCR decile gap under matched fee/ETA. (b) Driver DAR percentiles. (c) Driver DCR decile gap under matched conditions. (d) Share of dispatching rounds in which the same broadcasted order receives divergent accept/reject decisions from different drivers.

motivating ProfiLLM's adaptive cluster-level profiling that transfers knowledge from data-rich groups to data-sparse individuals.

B REPRESENTATIVE CASE STUDIES

To illustrate the kind of behavioral heterogeneity that ProfiLLM's profile embeddings are designed to capture, we walk through two representative cases drawn from City A production logs.

Driver case: divergent acceptance under nearly identical order context. In a single dispatching round, order `2209******7356` (fee \$17.0, ETA 384 s, evening hour 17, origin/destination both in the city center) was broadcast to **19 drivers**. Only **one driver accepted**; the other **18 rejected**. The accepting driver had a baseline DAR of 0.089 and historical completion rate of 0.289; a representative rejecting driver had DAR 0.114 and completion rate 0.222, so a predictor relying purely on order-side and aggregate driver-side structured statistics would have ranked the rejecting driver *higher*. Across the full log, **16.3%** of dispatching rounds (1.93 M of 11.8 M batches) exhibit such mixed-outcome broadcasts where the same order receives divergent accept/reject decisions from different drivers under near-identical order-side context (Appendix A). ProfiLLM's cluster-level driver profile encodes precisely this kind of identity-driven pattern, e.g., the accepting driver's cluster is characterized by a willingness to take evening city-center orders with mid-tier fares, while the rejecting drivers' clusters are not.

Passenger case: ETA-sensitivity stratification under matched fare. The most cancellation-prone passengers in City A (18,394 passengers, average PCR 32.6%) book at average fare \$19.8 and ETA 293 s; a comparison group of 97,579 passengers exhibits PCR 0.0% under almost identical conditions (fare \$19.0, ETA 292 s). A finer-grained example: passenger `8796******8501` (50 grabbed orders) cancels **0%** of orders with ETA in 4–6 min but **100%** of orders with ETA > 8 min, while the platform average rises only from 6.9% to 11.3% across

the same ETA band. This per-passenger ETA-tolerance threshold is invisible to structured features that report only the absolute ETA value; ProfiLLM's cluster-level passenger profile materializes such latent tolerance patterns as part of the cluster's PROFILE narrative (e.g., "commute-hour passengers who tolerate ≤ 6 min wait but defect at longer ETAs"), feeding the prediction model a discriminative signal the structured-feature baseline cannot express.

C BACKGROUND

Tool-Augmented LLM Agents. Recent advances have demonstrated that LLMs can effectively leverage external tools to accomplish complex tasks beyond their inherent capabilities [13, 22, 26, 37, 45]. A tool-augmented LLM agent operates by iteratively generating reasoning traces and invoking tools based on intermediate observations. Formally, given an input query q and a tool set $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$, the agent produces a trajectory $\tau = \{(a_i, r_i)\}_{i=1}^{L}$, where a_i denotes an action (either reasoning or tool invocation) and r_i denotes the corresponding observation or tool result. This paradigm enables LLMs to analyze data at scales far exceeding their context window limitations.

Direct Preference Optimization (DPO). DPO [28] provides an efficient approach to align LLM outputs with human preferences without explicit reward modeling. Given preference pairs (x, y_w, y_l) where y_w is preferred over y_l for input x, DPO directly optimizes the policy π_θ via:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(x,y_w,y_l)}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{7}$$

where π_{ref} is a reference policy (typically the supervised fine-tuned model), β controls the deviation from the reference, and $\sigma(\cdot)$ is the sigmoid function. In our context, we extend DPO to align profile generation with downstream prediction utility.

D ANALYTICAL TOOL DETAILS

Table 4 presents the complete categorization of the 27 analytical tools used in the Tool-Augmented Global Knowledge Mining module. These tools are organized into six categories based on their analytical functionality: (1) **Statistical** tools for computing aggregate statistics, comparing segments, clustering users, and identifying important features; (2) **Causal** tools for discovering causal relationships, performing counterfactual analysis, and conducting multi-stage iterative mining with uncertainty quantification; (3) **Knowledge** tools for extracting global causal rules, generating classification benchmarks, and constructing profile knowledge bases; (4) **Validation** tools for verifying discovered patterns and cross-validating conclusions; (5) **Spatiotemporal** tools for detecting peak periods, analyzing day-of-week and hourly patterns, identifying spatial hotspots, and examining origin-destination flows; and (6) **Contextual** tools for analyzing supply-demand balance, wait time factors, matching efficiency, anomalies, special periods, and weather impacts. All tools are designed to accept structured parameters and return interpretable results that the LLM agent can reason over, enabling composable tool chains for complex analytical queries.

Algorithm 1 Tool-Augmented Global Knowledge Mining

Require: Historical data $\mathcal{H}$, Tool set $\mathcal{T}$, LLM agent $\mathcal{M}$
Ensure: Global knowledge $\mathcal{K}$, Clustering rules $\mathcal{A}$, Regional priors $\mathcal{R}$

// Phase 1: Explore
$findings_1 \leftarrow \emptyset$
for $t \in \mathcal{T}_{basic}$ **do**
  $result \leftarrow t$.execute($\mathcal{H}$)
  $findings_1 \leftarrow findings_1 \cup \mathcal{M}$.interpret($result$)
end for
// Phase 2: Deepen
$directions \leftarrow \mathcal{M}$.identify_directions($findings_1$)
$findings_2 \leftarrow \emptyset$
for $dir \in directions$ **do**
  $tools \leftarrow \mathcal{M}$.select_tools($dir, \mathcal{T}$)
  $result \leftarrow$ ExecuteToolChain($tools, \mathcal{H}$)
  $findings_2 \leftarrow findings_2 \cup \mathcal{M}$.analyze($result$)
end for
// Phase 3: Validate
$candidates \leftarrow \mathcal{M}$.extract_hypotheses($findings_1 \cup findings_2$)
$validated \leftarrow \emptyset$
for $hyp \in candidates$ **do**
  $pval, eff \leftarrow$ `validate_hypothesis`($hyp, \mathcal{H}$)
  if $pval < \alpha$ **and** $|eff| > \epsilon$ **then**
    $validated \leftarrow validated \cup \{(hyp, eff)\}$
  end if
end for
// Phase 4: Synthesize
$\mathcal{K} \leftarrow \mathcal{M}$.synthesize_knowledge($validated$)
$\mathcal{A} \leftarrow \mathcal{M}$.generate_clustering_rules($findings_2$)
$\mathcal{R} \leftarrow$ `compute_regional_priors`($\mathcal{H}, \mathcal{G}$)
return $\mathcal{K}, \mathcal{A}, \mathcal{R}$

E ALGORITHM PSEUDOCODE

This section provides the detailed pseudocode for the two core procedures in ProfiLLM.

Algorithm 1 formalizes the Tool-Augmented Global Knowledge Mining workflow described in Section 3.2. The procedure follows the four-phase Explore-Deepen-Validate-Synthesize paradigm. In the Explore phase, the agent invokes basic statistical tools to obtain an initial understanding of the data landscape. The Deepen phase identifies promising analytical directions from preliminary findings and applies targeted tool chains for focused investigation. The Validate phase subjects each discovered pattern to statistical hypothesis testing, retaining only findings with p-value below threshold α and effect size exceeding ϵ. Finally, the Synthesize phase consolidates validated findings into three structured outputs: global knowledge $\mathcal{K}$, user clustering rules $\mathcal{A}$, and regional supply-demand priors $\mathcal{R}$.

Algorithm 2 details the DPO-Aligned Profile Exploration procedure described in Section 3.3. Given a user cluster a with its aggregated history and the global knowledge base, the algorithm first generates K diverse candidate profiles and evaluates each via the LOGIC-rule-based utility proxy (Eq. 4). The best-performing candidate then undergoes iterative refinement for T rounds: at each iteration, prediction errors are analyzed to produce targeted feedback, and the LLM generates an improved profile conditioned on this feedback. Throughout the process, all candidate profiles are compared pairwise to construct preference pairs with a margin threshold γ, which are subsequently used for DPO fine-tuning to

Table 4: Categorization of analytical tools in ProfiLLM.

Category	#	Tool	Description
Statistical	4	`AggregateStats`	Calculate aggregate statistics by dimension
		`CompareSegments`	Compare segments on specific metrics
		`UserClustering`	K-Means clustering on user behavior
		`FeatureImportance`	Analyze key features for target metric
Causal	5	`CausalDiscovery`	Discover causal relationships
		`CounterfactualAnalysis`	Perform "what if" analysis
		`ChainOfMining`	Multi-stage iterative analysis
		`UncertaintyAwareMining`	Provide confidence intervals
		`ContrastiveAnalysis`	Compare similar groups for differences
Knowledge	3	`GlobalCausalRules`	Discover global causal rules
		`GlobalBenchmarks`	Generate benchmarks for classification
		`ProfileKnowledgeBase`	Generate profile usage guide
Validation	2	`ValidationDiscovery`	Validate discovered patterns
		`ConclusionValidation`	Cross-validate conclusions
Spatio-temporal	6	`DetectPeakPeriods`	Detect peak periods for metrics
		`DayOfWeekPattern`	Analyze weekday vs weekend patterns
		`HourlyPattern`	Analyze 24-hour detailed patterns
		`SpatialHotspot`	Identify spatial distribution hotspots
		`ODFlowAnalysis`	Analyze origin-destination flow patterns
		`RegionCharacteristics`	Analyze regional characteristic profiles
Contextual	7	`SupplyDemandAnalysis`	Analyze supply-demand balance
		`WaitTimeFactors`	Analyze factors affecting wait time
		`MatchingEfficiency`	Analyze order matching efficiency
		`DetectAnomalies`	Detect anomalies in metrics
		`SpecialPeriodAnalysis`	Analyze holiday/event patterns
		`WeatherFactorAnalysis`	Analyze weather impact on metrics
		`WeatherScenario`	Analyze specific weather scenarios

align the LLM's generation capability with downstream prediction utility.

Algorithm 2 Utility-Aligned Profile Exploration

Require: Cluster a, Aggregated history $\mathcal{H}_a$, Global knowledge $\mathcal{K}$, LLM $\mathcal{M}$
Ensure: Optimal profile $profile_a^*$, Preference pairs $\mathcal{P}_a$

// Initial candidate generation
$\{profile_a^{(k)}\}_{k=1}^{K} \leftarrow \mathcal{M}.\text{generate}(\mathcal{H}_a, \mathcal{K}, K)$
// Evaluate initial candidates via LOGIC rules
for $k = 1$ **to** K **do**
  $LOGIC_a^{(k)} \leftarrow \text{extract\_logic}(profile_a^{(k)})$
  $\Delta_a^{(k)} \leftarrow \text{EvaluateUtility}(LOGIC_a^{(k)}, \mathcal{H}_a)$ ▷ Eq. 4
end for
$k^* \leftarrow \arg\max_k \Delta_a^{(k)}$
$profile_a^{best} \leftarrow profile_a^{(k^*)}$; $\Delta_a^{best} \leftarrow \Delta_a^{(k^*)}$
// Iterative refinement
for $t = 1$ **to** T **do**
  $feedback \leftarrow \text{AnalyzeErrors}(LOGIC_a^{best}, \mathcal{H}_a)$
  $\{profile_a^{(t,k)}\}_{k=1}^{K} \leftarrow \mathcal{M}.\text{refine}(profile_a^{best}, feedback, K)$
  for $k = 1$ **to** K **do**
    $\Delta_a^{(t,k)} \leftarrow \text{EvaluateUtility}(profile_a^{(t,k)}, \mathcal{H}_a)$
  end for
  $k^\dagger \leftarrow \arg\max_k \Delta_a^{(t,k)}$
  if $\Delta_a^{(t,k^\dagger)} > \Delta_a^{best}$ **then**
    $profile_a^{best} \leftarrow profile_a^{(t,k^\dagger)}$; $\Delta_a^{best} \leftarrow \Delta_a^{(t,k^\dagger)}$
  end if
end for
$profile_a^* \leftarrow profile_a^{best}$
// Construct preference pairs for DPO
$\mathcal{P}_a \leftarrow \{(\mathcal{H}_a, profile_w, profile_l) : \Delta_w > \Delta_l + \gamma\}$
return $profile_a^*$, $\mathcal{P}_a$

F DISCOVERED CLUSTER ARCHETYPES

ProfiLLM's clustering rules are *not* manually specified. They are produced by the LLM agent during the Synthesize phase of Tool-Augmented Global Knowledge Mining (Algorithm 1), which interprets cluster centroids and their z-score deviations from the population mean to assign meaningful archetype labels. Drivers and passengers are clustered separately into $A = A_D \cup A_P$, where the agent settled on $|A_D| = 28$ driver clusters and $|A_P| = 49$ passenger clusters in our main experiments. Clustering uses dozens of behavioral features, including order volume and frequency, acceptance/grab rate, cancellation rate and patterns, completion rate, average fee and price sensitivity, active-time distribution, trip distance and duration, spatial activity patterns, and derived ratios (e.g., cancel-to-complete ratio, peak-hour share).

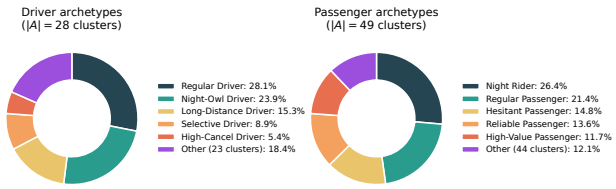
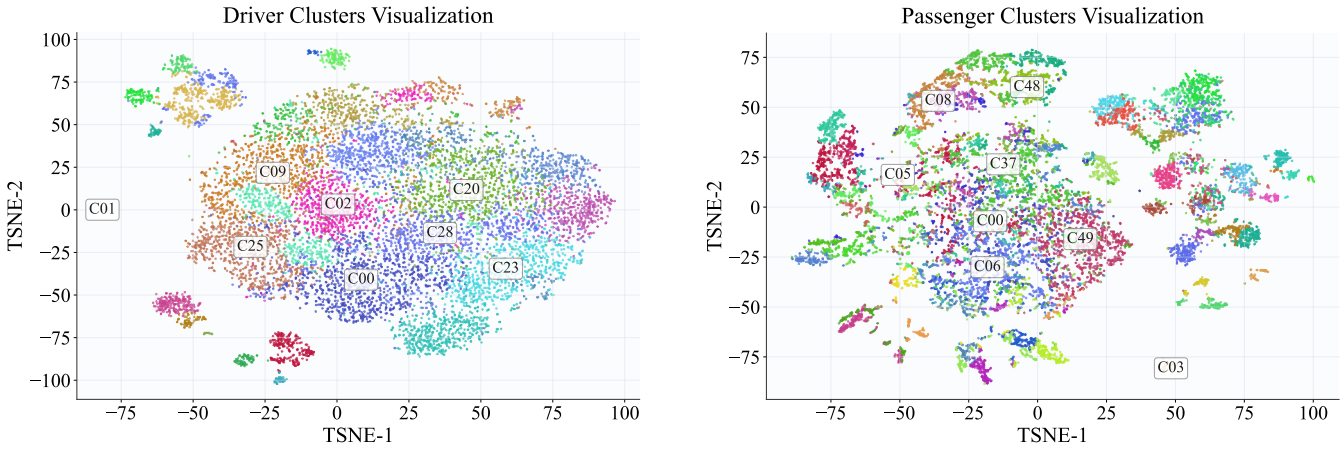


Figure 8: Population share of the top-5 LLM-discovered archetypes in City A for drivers ($|A_D| = 28$) and passengers ($|A_P| = 49$), with the remaining clusters aggregated for clarity.

Figure 8 reports the population share of the top-5 archetypes for each role, and Table 5 lists their defining characteristics. The dominant driver groups (Regular, Night-Owl, Long-Distance, Selective, High-Cancel) together cover 81.6% of the active driver fleet, while the corresponding top-5 passenger groups cover 87.9% of the passenger population. Tail clusters retain meaningful distinctions (e.g., airport-specialist drivers, time-sensitive commuter passengers) that the LLM agent labels using domain-grounded heuristics from the mined global knowledge.

F.1 User Cluster Embedding Visualization

To qualitatively assess whether the clustering rules $\mathcal{A}$ yield behaviorally separable groups, we visualize user-level embedding distributions via t-SNE [21]. For each user $u \in \mathcal{P} \cup \mathcal{D}$, we extract a behavioral feature vector from $\mathcal{H}_u$ and assign them to cluster $a^*(u)$ via rules $\mathcal{A}$. Before dimensionality reduction, we apply core filtering (retaining the nearest 80% of points to each cluster centroid) and stratified sampling (capping at 15,000 points per role). The filtered embeddings are standardized, reduced to 50 dimensions via PCA,



(a) Driver clusters

(b) Passenger clusters

Figure 9: t-SNE visualization of user cluster embeddings in City A, showing (a) driver clusters and (b) passenger clusters. Each point represents a user colored by cluster assignment; the top-8 largest clusters are labeled at their median positions.

Table 5: Representative archetypes discovered by the LLM agent in City A. Listed signatures are cluster-centroid characteristic values (salient deviations from the population mean) used by the agent for labeling, not realized post-acceptance PCR/DCR rates.

Archetype	Share	Key behavioral signatures
Driver		
Regular Driver	28.1%	Moderate activity, morning-dominant hours
Night-Owl Driver	23.9%	Avg. hour 16.1, high evening concentration
Long-Distance Driver	15.3%	Avg. fee 20.8, avg. trip 7.7 km
Selective Driver	8.9%	Low volume but 44.6% grab rate
High-Cancellation Driver	5.4%	Cancel rate 84.5%, low completion
Passenger		
Night Rider	26.4%	Avg. hour 16.8, evening-dominant
Regular Passenger	21.4%	Moderate volume, morning-focused
Hesitant Passenger	14.8%	Cancel rate 69.6%, low reliability
Reliable Passenger	13.6%	Low volume, completion rate 46.9%
High-Value Passenger	11.7%	Avg. fare 34.0, avg. trip 14.1 km

and projected to 2D with t-SNE (perplexity = 35, PCA initialization, seed = 42).

Figure 9 presents the resulting scatter plots for City A, with the top-8 largest clusters annotated at their median positions. The driver embedding space (Figure 9 (a)) exhibits well-separated clusters corresponding to discovered archetypes, including *Regular Drivers* (C00), *Night Owls* (C01), and *Long-Distance Drivers* (C02). The passenger embedding space (Figure 9 (b)) similarly reveals distinct groups such as *Night Riders*, *Regular Passengers*, and *Hesitant Passengers* with elevated cancellation tendencies, though with more overlap reflecting higher behavioral diversity on the passenger side. The clear visual separation confirms that the LLM-agent-derived clustering rules partition users into behaviorally coherent groups, supporting cluster-level profiles as effective proxies for individual user behavior in downstream outcome prediction.

G DISPATCHING SIMULATOR ARCHITECTURE

We summarize the design choices that make our simulator a faithful and reproducible offline counterpart to production dispatcher.

Replay-based discrete-event environment. The simulator replays five full days of historical order arrivals and driver availability per city, rather than generating synthetic demand. All orders arrive at their actual timestamps with real origin, destination, dynamic pricing (including surge multipliers), and over 30 contextual features. Driver positions are initialized from actual GPS trajectory logs, and each driver's online-duration distribution is reconstructed from historical records. The geographic space uses the same production grid system, preserving real-world spatiotemporal distributions of demand, supply, and traffic.

Production-identical dispatching cadence and matching. The simulator operates in 2-second dispatching cycles. At each cycle, candidate OD pairs are enumerated within a 1,500-meter pickup radius; the production STR (Spatio-Temporal Revenue) formula scores each pair using Accept/P-Cancel/D-Cancel predictions, pickup cost, and order value with production-calibrated weights; and the optimal bipartite assignment is solved via a Lagrangian-relaxation variant of Kuhn–Munkres, identical to the algorithm deployed in production.

Production routing API integration. The simulator queries DiDi's production routing service via Thrift RPC to obtain live-traffic-aware ETA and pickup distance (with 25th/75th-percentile confidence intervals), eliminating a major source of bias that approximate offline routing would introduce.

Three-stage stochastic outcome simulation. Rather than sampling a single completion probability, the simulator implements a three-stage sequential decision process: (i) Accept sampling (driver acceptance); (ii) conditional D-Cancel sampling; (iii) P-Cancel sampling. An order completes *only if* the driver accepts and neither party

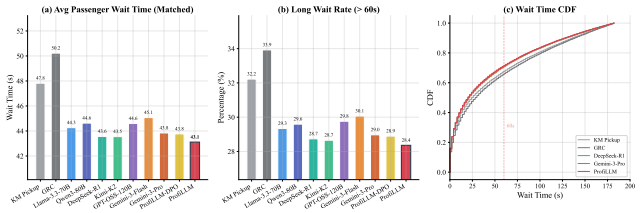


Figure 10: Passenger wait time analysis in the dispatching simulator. (a) Average wait time for matched orders. (b) Proportion of orders with wait time exceeding 60 seconds. (c) Cumulative distribution function (CDF) of wait times for representative strategies.

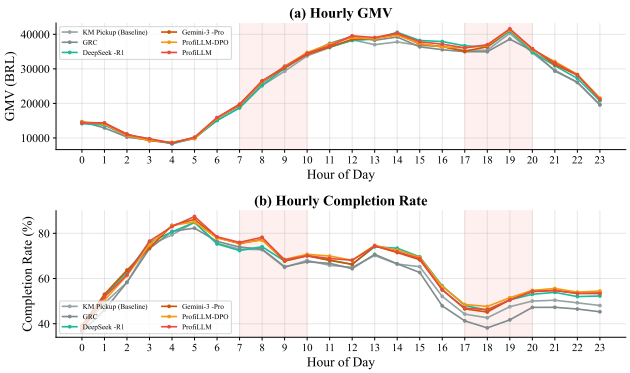


Figure 11: Hourly GMV and CR in the dispatching simulator. Shaded regions indicate peak hours (morning 7–10, noon 11–14, evening 17–20).

cancels. Failed orders re-enter the pending queue for re-dispatch up to a 180-second patience timeout, producing realistic re-dispatch cascades. This three-stage design captures the feedback loop in which a better predictor yields better matches, fewer rejections, fewer re-dispatches, and higher system-wide efficiency.

Dynamic state evolution. Drivers transition between *idle*, *en-route-to-pickup*, and *in-service* states with real-time features refreshed every cycle, ensuring that downstream feature distributions remain consistent with production. Orders follow realistic lifecycle transitions with timeouts, and grid-level supply-demand statistics support spatial-aware matching weights.

Validation against production. Comparing simulation (Table 1, City A) with the 14-day online A/B test (Figure 5) shows directional consistency on every core metric (GMV, CR, PCR, DCR all moving in the desired direction), though the online deltas are several times smaller (*e.g.,* **+0.47%** vs. **+4.02%** GMV, a ∼8–9× gap). Simulation magnitudes exceed online deltas because the simulator both scores candidate matches and samples their outcomes from the *same* model, grading each policy on its own beliefs in a closed loop that omits the live confounders (driver multi-homing, exogenous demand shocks, concurrent experiments) which attenuate real treatment effects; this downward bias is well documented for two-sided-marketplace experiments [10]. This direction-preserving, magnitude-inflating behavior is characteristic of replay-based ride-hailing simulators [27, 42, 44] and mirrors the broader offline-to-online gap, in which offline metrics overstate online performance while better preserving method ranking [9]. The key validation criterion is therefore directional consistency and method ranking preservation, both of which our simulator delivers.

H PASSENGER WAIT TIME ANALYSIS

We further analyze passenger wait time in the dispatching simulator to evaluate user experience beyond platform-level metrics. As shown in Figure 10, the two non-LLM baselines exhibit the highest average wait times (GRC: 50.2s, KM Pickup: 47.8s), while all LLM-based strategies achieve notably lower values ranging from 43.1s to 45.1s. ProfiLLM attains the lowest average wait time of 43.1s, representing reductions of 9.8% over KM Pickup and 14.1% over GRC. The long-wait rate (>60s) follows a consistent pattern: GRC and KM Pickup reach 33.9% and 32.2%, respectively, whereas ProfiLLM reduces this to 28.4%. The CDF curves further confirm a systematic leftward shift for ProfiLLM across all quantiles, indicating that the improvement is not driven by a small subset of orders but reflects enhanced dispatching efficiency throughout. These results complement the GMV and CR metrics in Section 4.2, demonstrating that utility-aligned user profiles improve not only platform revenue but also passenger-perceived service quality.

I HOURLY PERFORMANCE ANALYSIS

To understand how different strategies perform across varying demand conditions, we analyze hourly GMV and CR in the dispatching simulator. As shown in Figure 11, all strategies follow the same demand pattern with peaks around noon (12–13h) and evening (18–19h), and a trough in the early morning (4–5h). ProfiLLM and ProfiLLM-DPO consistently outperform baselines throughout the day, with the gap most pronounced during evening peak hours (17–20h) when supply-demand imbalance intensifies and accurate outcome prediction becomes critical. Notably, GRC suffers a sharp CR drop during evening hours (falling below 40%), likely due to its cooperative game formulation struggling under severe supply constraints, whereas ProfiLLM maintains stable performance above 50%. The consistent hourly advantage confirms that utility-aligned user profiles provide robust improvements across diverse operational conditions rather than benefiting only specific time periods.

J CLUSTER-COUNT SENSITIVITY

To verify that the framework is robust to the granularity of clustering, we evaluate 11 cluster configurations on City A by varying one side while fixing the other: $|A_D| \in \{8, 16, 32, 64, 128, 256\}$ at $|A_P| = 64$, and $|A_P| \in \{8, 16, 32, 64, 128, 256\}$ at $|A_D| = 32$. Each configuration is evaluated against 9 LLM backbones (the 7 baselines in Table 2 plus ProfiLLM and ProfiLLM-DPO), totaling 99 runs.

Figure 12 shows that both ProfiLLM and ProfiLLM-DPO produce stable AUC across the entire sweep. Performance climbs from 8 to 16 clusters per side and then plateaus, with variation within 0.6 p.p. absolute AUC for all four tasks. Across all 11 cluster configurations, both ProfiLLM and ProfiLLM-DPO outperform the structured-only baseline on Accept, D-Cancel, and P-Cancel at every cluster count; among the 99 runs the un-aligned baseline backbones remain mixed

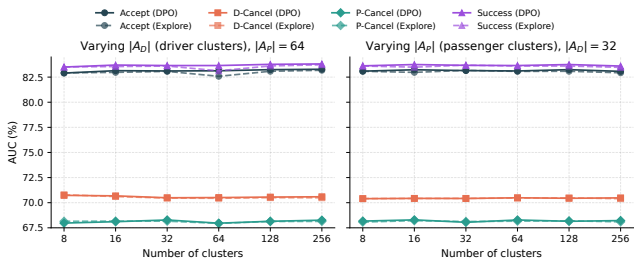


Figure 12: Cluster-count sensitivity for ProfiLLM (dashed) and ProfiLLM-DPO (solid) on City A, with each curve aggregated over 9 LLM backbones. Left: varying $|A_D|$ with $|A_P| = 64$. Right: varying $|A_P|$ with $|A_D| = 32$. The cross-model standard deviation across the 9 backbones is 0.05%–0.27%.

(consistent with Table 2), confirming that the ProfiLLM gains observed in the main paper are not specific to a particular cluster count. Across the same sweep, the two variants are statistically indistinguishable on prediction AUC: average $\Delta = +0.14\%$ (Accept), +0.05% (D-Cancel), +0.02% (P-Cancel), and +0.14% (Success), with the sign of Δ fluctuating across configurations rather than systematically favoring either variant. This supports our deployment choice of ProfiLLM-DPO, which achieves comparable prediction quality at substantially lower offline refresh cost (Appendix M).

K UTILITY-PROXY SENSITIVITY TO THE BLENDING COEFFICIENT λ

The blending coefficient λ in Eq. (3) controls how strongly the LOGIC rules are mixed with the base production model during *offline profile evaluation*. It does not appear in the online prediction model. We perform a grid search over $\lambda \in \{0, 0.1, \ldots, 1.0\}$ across 6 cluster granularities for each of 3 outcome tasks, yielding 494 cluster-task combinations (21 driver-accept, 82 driver-cancel, 391 passenger-cancel clusters; counts vary because clusters with insufficient behavioral data are filtered out).

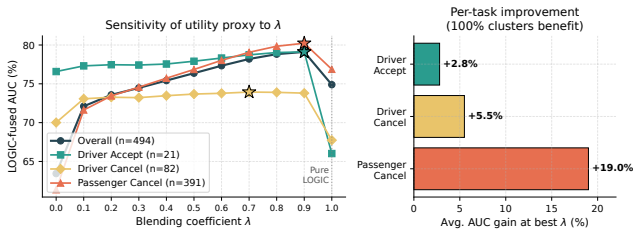


Figure 13: Sensitivity of the LOGIC-rule utility proxy to the blending coefficient λ. Left: per-task LOGIC-fused AUC versus $\lambda \in [0, 1]$, averaged across 494 cluster-task combinations; stars mark each task's peak. Right: average AUC gain at the best λ for each task. $\lambda = 1.0$ denotes pure LOGIC with the base model discarded.

Figure 13 (left) reports the average LOGIC-fused AUC as a function of λ. Three patterns are clear: (1) All tasks improve over $\lambda = 0$ (base model only) once blending is introduced, demonstrating that LOGIC rules supply *complementary* discriminative signal beyond structured features. (2) All tasks collapse at $\lambda = 1.0$ (pure LOGIC, base model discarded), confirming the conservative design that retains the strong production predictor. (3) The optimum is task-dependent: Driver Accept and Passenger Cancel peak at $\lambda = 0.9$, whereas Driver Cancel peaks at $\lambda = 0.7$–0.8 and stays remarkably flat over $\lambda \in [0.1, 0.9]$. At the cluster level, 65.7% of passenger clusters prefer $\lambda = 0.9$ while 40.2% of driver-cancel clusters prefer $\lambda = 0.1$, motivating adaptive per-cluster λ selection as a direction for future work.

Figure 13 (right) summarises the per-task gain at each task's best λ: +2.8% (Driver Accept), +5.5% (Driver Cancel), and +19.0% (Passenger Cancel). The cancellation tasks gain the most, mirroring the prediction-AUC pattern in Table 2: behavioral profiling shines exactly where structured features struggle most, namely in capturing the contextual decision logic behind cancellations.

L DPO VS. EXPLORATION: WHY BOTH VARIANTS HELP

A natural question is why we report both ProfiLLM (exploration only) and ProfiLLM-DPO (with DPO fine-tuning) when their per-task prediction AUCs in Table 2 are similar. The two variants are complementary by design, and we clarify their roles below.

(1) DPO targets generator efficiency, not per-task AUC.. ProfiLLM (exploration) generates $K = 5$ candidate profiles per cluster and refines the best for $T = 3$ iterations. The theoretical upper bound is $K(1 + T) = 20$ LLM calls per cluster (5 initial + 5 re-explored per refinement iteration), which we cap at 15 in deployment for compute efficiency. ProfiLLM-DPO generates a high-quality profile in a *single pass*, eliminating the iterative search. When profiles are refreshed for new clusters, new cities, or updated behavioral data, this collapses the offline LLM-call budget by an order of magnitude (Appendix M). Both variants serve identically online via cached embeddings; the difference is purely offline.

(2) The two variants achieve comparable prediction quality across configurations. Across the 11 cluster configurations of Appendix J, the average inter-variant gap is only +0.14% (Accept AUC), +0.02% (P-Cancel AUC), and +0.14% (Success AUC), with the sign of the gap fluctuating across configurations rather than systematically favoring either variant (maximum absolute deviation is 0.55 p.p.). Both variants substantially outperform all 7 baseline LLMs in every metric and every city (Tables 1–2), so the choice between them is dominated by cost, not quality.

(3) Per-task differences in Table 2 reflect different optimization strategies, not degradation. ProfiLLM (exploration) performs *per-cluster local optimization*: it iteratively searches profile space for each cluster, and the LOGIC-rule AUC proxy directly drives the selection. ProfiLLM-DPO aggregates preference pairs *across clusters* and learns a single generator (Qwen3-8B) that produces high-utility profiles in one pass. The latter trades a small amount of per-cluster local optimality for cross-cluster generalization. A second, subtler factor is the *signal-channel gap*: DPO is supervised on LOGIC-rule AUC (a discrete Boolean projection), while the downstream prediction model consumes PROFILE *text embeddings* (continuous dense vectors). The two share the same behavioral understanding but

are not identical, so DPO optimization on one does not transfer perfectly to the other. Finally, online GMV is a composite matching objective over all three predicted outcomes, so small per-task differences can cancel or compound through the matching weights.

We therefore deploy ProfiLLM-DPO in production because it achieves comparable prediction quality at substantially lower offline refresh cost, enabling faster iteration when scaling to new clusters and cities.

M OFFLINE SYSTEM COST ANALYSIS

Table 6 reports the end-to-end offline cost of running ProfiLLM on a single city with $|A_D| = 32$ driver clusters and $|A_P| = 64$ passenger clusters using Gemini-3-Pro as the analyst LLM (input \$1.25/1M tokens, output \$10.00/1M tokens). Downstream model training uses one NVIDIA L20 GPU (48GB) at \$1.50/GPU-hour.

Table 6: Offline pipeline cost breakdown for one city. The Profile + Exploration LLM-call count is $(32 + 64) \times 15 = 1{,}440$, where 15 is the deployed per-cluster cap (theoretical max $K(1 + T) = 20$ with $K = 5, T = 3$).

Stage	Hardware	Wall Time	LLM Calls	Tokens (in/out)	Cost
Initial run (ProfiLLM, exploration)					
Global Knowledge Mining	CPU+API	∼50 min	∼20	2M/0.5M	\$7.50
Profile + Exploration	CPU+API	∼240 min	∼1,440	12M/3M	\$45.00
Downstream Training	1×L20	∼85 min	0	–	\$2.13
Total initial	–	**∼6.3 hrs**	**∼1,460**	14M/3.5M	**\$54.63**
Subsequent refresh (ProfiLLM-DPO, single-pass)					
Profile Generation	CPU+API	∼25 min	96	0.8M/0.2M	\$3.00
Downstream Training	1×L20	∼85 min	0	–	\$2.13
Total refresh	–	**∼1.8 hrs**	**96**	0.8M/0.2M	**\$5.13**

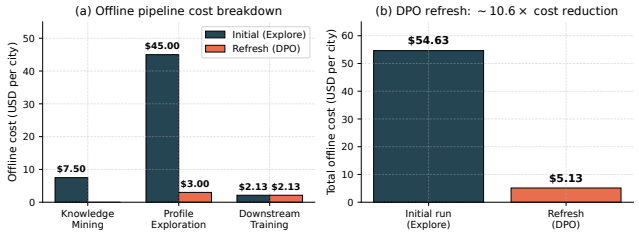


Figure 14: Offline cost breakdown. (a) Per-stage cost for the initial ProfiLLM run vs. a single-pass DPO refresh. (b) Total per-city offline cost for the two settings.

Three observations follow (Figure 14). (1) **Cluster-level profiling drives cost efficiency**: 96 cluster-level profiles cover all 348,464 users in City A, a 3,630× reduction over per-user profiling and the precondition for affordable LLM-driven profiling at platform scale. (2) **DPO compounds the efficiency gain**: once DPO is trained, subsequent refreshes require only 96 single-pass calls, reducing per-city LLM cost from \$52.50 to \$3.00 and total refresh cost from \$54.63 to \$5.13 (∼ 10.6× reduction). (3) **Online overhead is negligible**: at serving time the system performs only deterministic cluster assignment (<0.01 ms) and a cached embedding lookup (<0.001 ms), well within DiDi's 200 ms latency budget; no LLM is queried online.

The 14-day A/B improvement of +0.47% GMV on a platform processing millions of daily orders translates into revenue gains that exceed the offline cost by several orders of magnitude, even before considering the platform-side savings from reduced cancellations and bad-experience rates.

N COMPLEXITY ANALYSIS

We analyze the computational complexity of ProfiLLM along offline, online, and storage dimensions, and compare against per-user profiling. Notation follows Section 2: $M = |\mathcal{P}|$ passengers and $N = |\mathcal{D}|$ drivers; $\mathcal{H} = \bigcup_u \mathcal{H}_u$ aggregated history with $|\mathcal{H}| = \sum_u |\mathcal{H}_u|$; $\mathcal{A} = \mathcal{A}_D \cup \mathcal{A}_P$ cluster set; K=5 candidates per generation, T=3 refinement iterations; d=768 embedding dimension; and $|C|$ candidate OD pairs per dispatching cycle.

N.1 Offline Complexity

(O1) Knowledge Mining (Algorithm 1). Each tool performs at most a single pass over $\mathcal{H}$ at $O(|\mathcal{H}|)$ cost. The Explore phase invokes the basic tool set $\mathcal{T}_{basic} \subset \mathcal{T}$; Deepen and Validate may then invoke any tool in $\mathcal{T}$ via targeted chains, with each tool used at most once across the workflow (so the total tool-invocation count is $O(|\mathcal{T}|)$). The aggregate tool-execution cost is therefore $O(|\mathcal{T}| \cdot |\mathcal{H}|)$. The LLM agent issues a constant number of reasoning calls (≈20 in deployment, Table 6), independent of $|\mathcal{H}|$. This stage is linear in data and constant in LLM calls.

(O2) Cluster Assignment. For each user u, we evaluate $|\mathcal{A}|$ membership rules over $\mathcal{H}_u$, costing $O(|\mathcal{A}| \cdot |\mathcal{H}_u|)$. Summed over all users this is $O(|\mathcal{A}| \cdot |\mathcal{H}|)$ and is embarrassingly parallel.

(O3) Profile Exploration (Algorithm 2). Per cluster a with aggregated history $\mathcal{H}_a$ (where $\sum_a |\mathcal{H}_a| \le |\mathcal{H}|$), the initial generation issues K LLM calls, and each of the T refinement iterations regenerates K candidates conditioned on prediction-error feedback, contributing KT additional calls; every generated candidate's LOGIC rule is then evaluated over $\mathcal{H}_a$ at $O(|\mathcal{H}_a|)$ cost. Aggregating over clusters:

$$\text{LLM calls} = O\big(|\mathcal{A}| \cdot K(1+T)\big), \qquad \text{LOGIC eval} = O\big(K(1+T) \cdot |\mathcal{H}|\big).$$

The theoretical upper bound is $K(1 + T) = 20$ calls per cluster; in deployment we cap total calls at 15 per cluster, which corresponds to early-terminating refinement after at most two iterations of the K-candidate regeneration, yielding $|\mathcal{A}| \cdot 15 = 1{,}440$ calls per city for $|\mathcal{A}| = 96$ (Table 6). This is the LLM-call-dominated stage, but the cost is amortized across all users in each cluster.

(O4) DPO Fine-tuning. Preference-pair construction over the $K(1 + T)$ profiles per cluster is at most $O\big(|\mathcal{A}| \cdot K^2(1 + T)^2\big)$. The DPO training cost is the standard LLM fine-tuning loop, $O(E \cdot |\mathcal{P}_{pref}| \cdot L \cdot c_{LLM})$, where E is the number of epochs, L is profile token length, and c_{LLM} denotes the per-token forward+backward FLOPs of the base model. In our deployment this takes ≈85 minutes on one NVIDIA L20 GPU.

(O5) Embedding Precomputation. A single encoder forward per cluster profile: $O(|\mathcal{A}| \cdot L \cdot c_{enc})$. Empirically negligible (seconds for $|\mathcal{A}|$=96).

N.2 Online Complexity (Per Dispatching Cycle)

(N1) Per OD-pair scoring. Passenger and driver cluster IDs are pre-assigned offline and refreshed at user registration; serving requires only two $O(1)$ cache lookups returning $\mathbf{e}_p, \mathbf{e}_d \in \mathbb{R}^d$, an $O(d)$ feature concatenation, and a constant-cost prediction-network forward $O(c_f)$.

(N2) Per-cycle cost. For $|C|$ candidate OD pairs the per-cycle cost is $O\big(|C| \cdot (d + c_f)\big) + O_{\mathrm{KM}}(|C|)$, where O_{KM} denotes the production Lagrangian-relaxation Kuhn–Munkres matching cost (identical to the structured-only baseline; see Appendix G). Profile features add only the $O(|C| \cdot d)$ feature-concat term, which empirically contributes well under 1 ms per cycle (Appendix M).

(N3) Cold-start. Users without sufficient history are mapped to a default cluster at registration ($O(1)$); no additional online cost.

N.3 Storage Complexity

ProfiLLM's serving state comprises three components. The cluster embeddings occupy $|\mathcal{A}| \cdot d \cdot 4\,\mathrm{B} = 96 \times 768 \times 4 \approx 295\,\mathrm{KB}$; the user-to-cluster table stores a 32-bit cluster ID per user at 4 B each, $\approx 1.4\,\mathrm{MB}$ for the 348,464 users in City A; and the LOGIC rules and PROFILE text amount to $|\mathcal{A}|$ short strings, $\approx$ 50 KB. The active footprint is therefore a few MB per city, nearly three orders of magnitude smaller than caching a per-user d-dimensional embedding ($\approx$1 GB for City A).

N.4 Comparison with Per-User Profiling

Cluster-level profiling reduces both LLM-call count and embedding storage by a factor of $(M + N)/|\mathcal{A}|$. For City A:

$$\frac{|\mathcal{P} \cup \mathcal{D}|}{|\mathcal{A}|} = \frac{348{,}464}{96} \approx 3{,}630\times .$$

(The deployment registry contains 348,464 users; Appendix A reports 345,294 users active within a narrower 38-day analysis window, hence the small discrepancy.) This is the structural source of ProfiLLM's offline cost efficiency, and Appendix J confirms that this reduction does not sacrifice prediction quality once $|\mathcal{A}| > 16$.

N.5 Summary

Table 7 consolidates the analysis. Offline complexity is linear in data ($|\mathcal{H}|$) for tool execution and cluster assignment, and the LLM-call count is linear in the *cluster* count $|\mathcal{A}|$ rather than the *user* count $(M + N)$. Online complexity is dominated by the existing bipartite-matching solver; profile features add only $O(|C| \cdot d)$ FLOPs and two $O(1)$ cache lookups per OD pair. Storage for LLM-introduced artifacts is sub-MB.

Stage	Time	Notes
Knowledge Mining (O1)	$O(\lvert\mathcal{T}\rvert \cdot \lvert\mathcal{H}\rvert)$	$+\,O(1)$ LLM calls
Cluster Assignment (O2)	$O(\lvert\mathcal{A}\rvert \cdot \lvert\mathcal{H}\rvert)$	parallel
Profile Exploration (O3)	$O(K(1+T) \cdot \lvert\mathcal{H}\rvert)$	$+\,O(\lvert\mathcal{A}\rvert \cdot K(1+T))$ LLM calls
DPO Training (O4)	$O(E \cdot \lvert\mathcal{P}_{pref}\rvert \cdot L)$	one-time
Embedding (O5)	$O(\lvert\mathcal{A}\rvert \cdot L)$	seconds
Online per OD pair (N1)	$O(d + c_f)$	two cache lookups
Online per cycle (N2)	$O(\lvert C\rvert \cdot d) + O_{\mathrm{KM}}$	KM dominates
Embeddings storage	$O(\lvert\mathcal{A}\rvert \cdot d)$	$\approx$ 295 KB
User-cluster table	$O(M + N)$	$\approx$ 1.4 MB (City A)

Table 7: Complexity summary. $|\mathcal{H}|$: total history records; $|\mathcal{A}|$: cluster count; K=5, T=3; $|C|$: candidate OD pairs per cycle; d=768; c_f: constant prediction-network forward cost.

O EXTENDED 14-DAY A/B TEST: LONG-TERM STABILITY

The 14-day deployment in Section 4.5 extends the initial 5-day pilot to a longer window for more stable estimates. We summarize the comparison and broader generalization evidence here.

Stability over time. GMV improvement *grew* from +0.36% (5 days) to **+0.47%** (14 days) and CR rose from +0.16% to +0.33%, while every cancellation and bad-experience metric remained directionally negative across the full window (CBA, PCR, DCR, BER). The fact that effect sizes did not decay, and in several cases grew, over the extended observation period supports the interpretation that ProfiLLM produces durable matching-quality gains rather than transient effects.

Joint movement across funnel stages. Every monitored realized rate moves in the beneficial direction across mechanistically distinct stages of the fulfillment funnel: revenue/completion (GMV, CR), pre-acceptance attrition (CBA), post-acceptance cancellation (PCR, DCR), and completed-order experience (BER). Each stage reflects a different behavioral mechanism, so the joint consistency provides converging evidence that the improvement is systematic. A model that improves only one stage would be expected to show mixed signs elsewhere; the uniform direction here is consistent with profiling improving the underlying outcome prediction that feeds every stage.

Generalization beyond a single city. While the A/B was conducted in City A, the offline simulator evaluation in Table 1 covers three cities with distinct supply-demand regimes (City A supply-constrained, City B supply-relaxed, City C large-scale high-demand) and uses the production routing API for realistic ETA. ProfiLLM dominates baselines across all three cities and all time-of-day buckets (Figure 11), supporting that the online gains would carry over. Broader online rollout across additional cities is in progress and will be reported in a follow-up.

P PRIVACY AND FAIRNESS CONSIDERATIONS

Because user profiles directly influence which drivers receive which orders, we discuss the privacy and fairness implications of ProfiLLM explicitly.

Privacy by architectural design. Two safeguards limit exposure of individual data. (1) *Cluster-level abstraction.* The LLM never sees a single user's identifiable trajectory in isolation. It processes only cluster-pooled, re-sampled order records together with summary statistics aggregated over all members of a cluster (e.g., "this cluster of drivers cancels orders with pickup distance > 5 km during evening hours at rate r"). An individual user's data only contributes to a cluster's aggregate and cannot be reconstructed from the profile description. (2) *Offline-only LLM inference.* Every LLM call happens in the offline knowledge-mining and profile-exploration stages.

At serving time the online system performs only a deterministic cluster-assignment rule evaluation and a cache lookup for precomputed embeddings (Section 3.4); no user data is transmitted to any LLM API during real-time dispatching.

Driver earning equity. If certain driver clusters are profiled as "likely to cancel long-pickup orders," the system may avoid such assignments, improving platform efficiency but potentially affecting those drivers' earning opportunities. This is, however, the behavior any accurate prediction model would produce, whether using profiles or structured features; ProfiLLM merely makes the prediction signal more explicit and *auditable*. Platform operators can inspect the textual PROFILE of each cluster (Section 3.3.2) and verify it does not encode undesirable biases, a transparency advantage over opaque deep-feature interactions.

Passenger service equity. Infrequent users, including the 96% long-tail passengers in Figure 2, are still assigned to behavioral clusters via $a^*(u)$ and receive shared cluster-level profiles rather than being excluded from profiling (Section 3.4.2); only genuine cold-start users with no usable history fall back to a default cluster. They therefore receive at least baseline matching quality, closing the cold-start gap that per-user profiling methods would face.

Behavioral vs. protected attributes. The clustering rules use only behavioral features (order patterns, cancellation history, temporal activity, spatial preferences), never protected attributes. We acknowledge that behavioral features may correlate with such attributes (commute-hour patterns with occupation, frequent regions with socioeconomic status). The cluster-level design supports a tractable auditing path: outcome distributions (driver income, passenger wait time, allocation rates) can be measured across clusters and clusters with statistically anomalous treatment flagged. We are integrating such auditing into our deployment pipeline as an ongoing direction.

Q PROMPT TEMPLATE

We present the abstracted prompt template in Table 8.

Table 8: Prompt templates used for profile exploration. Placeholders denote injected inputs; the concrete feature legend and data tables are omitted.

Template	Abstracted Prompt
Driver—Draft	[Role] Expert analyst (data science + behavioral economics) for ride-hailing driver behavior. [Task] Infer a stable driver persona and decision logic. [Inputs] Summary: {SUMMARY_STATS}; Grouped recent records: {RECENT_GROUPED_RECORDS}. [Guidelines] Records may be re-sampled; focus on feature differences across outcomes; use only features in the provided legend. [Reasoning] (1) rejection patterns (ignored) (2) regret patterns (post-accept cancellations) (3) persona + generalizable rules. [Output] XML only: <ANALYSIS>, <PROFILE>, <LOGIC_ACCEPT> (1-line Python), <LOGIC_CANCEL> (1-line Python).
Driver—Improve	[Inputs] {SUMMARY_STATS}, {RECENT_GROUPED_RECORDS}, plus previous response {PREVIOUS_RESPONSE} and feedback {FEEDBACK}. [Task] Improve profile + logic to increase validation performance; change only what is justified by patterns and feedback. [Output] A complete, self-contained updated response (not a patch), in the same XML-only format.
Passenger—Draft	[Role] Expert analyst for ride-hailing passenger post-match behavior. [Task] Infer patience and post-match cancellation triggers. [Inputs] Summary: {SUMMARY_STATS}; Grouped recent records (completed vs cancelled-after-match): {RECENT_GROUPED_RECORDS}. [Guidelines] Compare feature differences across outcomes; use only features in the provided legend (high-level: price/trip, ETA/waiting, context such as time and weather). [Reasoning] (1) time vs money (2) sunk cost (3) context modifiers (4) persona synthesis. [Output] XML only: <ANALYSIS>, <PROFILE>, <LOGIC_CANCEL> (1-line Python).
Passenger—Improve	[Inputs] {SUMMARY_STATS}, {RECENT_GROUPED_RECORDS}, plus previous response {PREVIOUS_RESPONSE} and feedback {FEEDBACK}. [Task] Improve profile + logic with minimal, data-justified changes. [Output] A complete, self-contained updated response (not a patch), in the same XML-only format.