Shopping Trajectory Representation Learning with Pre-training for E-commerce Customer Understanding and Recommendation

Yankai Chen*
The Chinese University of Hong Kong
Hong Kong SAR, China
ykchen@cse.cuhk.edu.hk

Quoc-Tuan Truong
Amazon
Seattle, USA
truquoc@amazon.com

Xin Shen
Amazon
Seattle, USA
xinshen@amazon.com

Jin Li
Amazon
Seattle, USA
jincli@amazon.com

Irwin King
The Chinese University of Hong Kong
Hong Kong SAR, China
king@cse.cuhk.edu.hk

ABSTRACT

Understanding customer behavior is crucial for improving service quality in large-scale E-commerce. This paper proposes C-STAR, a new framework that learns compact representations from customer shopping journeys, with good *versatility* to fuel multiple downstream customer-centric tasks. We define the notion of *shopping trajectory* that encompasses customer interactions at the level of product categories, capturing the overall flow of their browsing and purchase activities. C-STAR excels at modeling both *inter-trajectory distribution similarity*–the structural similarities between different trajectories, and *intra-trajectory semantic correlation*–the semantic relationships within individual ones. This coarse-to-fine approach ensures informative trajectory embeddings for representing customers. To enhance embedding quality, we introduce a pre-training strategy that captures two intrinsic properties within the pre-training data. Extensive evaluation on large-scale industrial and public datasets demonstrates the effectiveness of C-STAR across three diverse customer-centric tasks. These tasks empower customer profiling and recommendation services for enhancing personalized shopping experiences on our E-commerce platform.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Representation Learning; Shopping Trajectory; Customer Understanding; Shopping Intention; E-commerce Recommendation

ACM Reference Format:
Yankai Chen, Quoc-Tuan Truong, Xin Shen, Jin Li, and Irwin King. 2024. Shopping Trajectory Representation Learning with Pre-training for E-commerce Customer Understanding and Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671747

*This work is done during internship at Amazon.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671747

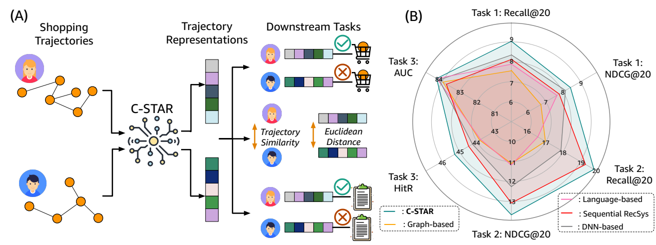

Figure 1: (A) C-STAR framework illustration; (B) performance comparison (%) with selected competitive methods.

1 INTRODUCTION

The ever-growing volume of products bombards online shoppers, making it difficult to identify items of interest. Recommender systems address this challenge by providing personalized suggestions throughout the customer shopping journey, from browsing to checkout [22, 62]. In pursuit of personalization for various scenarios, the key prong of delivering high-quality services is rooted in reliable and comprehensive customer understanding. This motivates us to effectively unveil customers' insights via mining information from their variety of historical shopping engagements.

One approach to achieving this is through customer representation learning, based on analyzing and encoding their engagement contents. However, existing methods are often tailored to one specific task, such as customer next-item recommendation [5, 24, 27, 39, 67, 73, 78, 79]. In the context of web-scale E-commerce platforms, our goal is to develop a *versatile* framework capable of addressing multiple downstream tasks related to customer profiling and services. Beyond the requisite capability for recommending items, this framework is expected to be able to effectively segment and group similar customers who are alike in their shopping behaviors. This facilitates a broader understanding of their common interests and market affinities [9, 16, 53]. Moreover, when contemplating personalization, it is essential for the framework to discern customer shopping intents, thereby enabling a more focused individual analysis [7, 18, 80]. In summary, this framework should offer efficacy and convenience by covering multiple downstream tasks, making it well-suited for real-world E-commerce scenarios.

Challenges. Achieving balanced model performance across multiple tasks poses a non-trivial challenge, as the customer representations in different scenarios own inherent diversities in learning objectives. For instance, when segmenting similar customers, the focus is primarily on capturing *customer-customer* similarity. Conversely, in tasks like item recommendation or customer shopping intent identification, customer embeddings pivot around learning *customer-item* relations or analyzing *customer-self* behaviors, respectively. Thus, we identify the critical requirement lies in **jointly** embedding *(1) accurate similarity measurement between customers* and *(2) informative content summarization in each customer engagement.* Consequently, the technical challenges are twofold:

- Customer engagements exhibit both quantitative and substantive variations. How to thoroughly reflect the similarity in customers' diverse activities within the fixed-size representations is the key question that remains to be investigated. This is particularly important for customer segmentation, where the similarity between customers is typically determined by their mutual distance in the embedding space, requiring alignment with real-world measurements. While one approach could be to aggregate all latent information, such as through concatenation or pooling of feature embeddings, this simplistic method may fail to provide a reliable embedding distance measurement with theoretical guarantees.
- Moreover, each customer's historical engagements reveal unique preferences and interests. Apart from capturing customer-wise similarity, it is imperative to preserve the semantic content of each representation to the greatest extent possible. This preserves compatibility for tasks like item recommendation or intention analysis, enabling embedding matching or classification formulation in the embedding space without exhaustive model retraining.

Approach and Contributions. In this work, we investigate the aforementioned problem and introduce a novel Customer Shopping TrAjectory Representation Learning framework (C-STAR), which is designed to be versatile for multiple downstream tasks, as depicted in Figure 1(A). C-STAR effectively encodes customer variable-size engagements into a continuous Euclidean space, facilitating efficient utilization for customer understanding and recommendation. Initially, we introduce PR-Graph, i.e., an internal knowledge base of product categories and relations that are organized in the graph format. Product interactions of each customer are then mapped into PR-Graph, creating his/her unique *shopping trajectory*. Each trajectory represents a sub-graph pattern of PR-Graph, allowing us to learn the customer trajectory representation that incorporates both structural and semantic information. The proposed model implements a representation learning paradigm enabling coarse-to-fine trajectory-wise similarity measurements as well as informative semantic enrichment in the embedding space. To enhance the embedding quality, we leverage intrinsic properties within the trajectory data structures and devise an effective pre-training strategy. As illustrated in Figure 1(B), while specialized methods demonstrate varying performances across different tasks, C-STAR consistently achieves superior and well-balanced model performances (further details in § 5). Our primary contributions can be outlined as follows:

(1) *Inter-Trajectory Distribution Similarity.* We propose to base on the assumption that elements constructing a trajectory are sampled from an underlying probability distribution reflecting customer unique preferences. By measuring distribution distance, we capture trajectory-wise similarity and incorporate this information into trajectory representations. Grounded in Optimal Transport theory, our approach offers a consistent distance measurement between realistic and embedding spaces.
(2) *Intra-Trajectory Semantic Correlation.* To capture the relational knowledge among the trajectory elements, we further propose to learn intra-trajectory semantic correlation from the structure posed by PR-Graph. This will not merely provide a *fine-grained* proximity measurement but also enrich the semantics of trajectory representations, refining its capability for shopping intent identification and item recommendation.
(3) *C-STAR Pre-training Strategy.* To improve the embedding quality, we leverage the intrinsic data properties and design two pre-training objectives. Furthermore, while the C-STAR model accommodates variable-size shopping trajectories for online inference, tensorized pre-training requires fixed-size trajectory batches. We then introduce effective data-driven sampling strategies that are independent of human prior knowledge.
(4) *Extensive Empirical Evaluation.* We systematically conduct experiments, including online A/B testing on our platform and offline evaluation on both large-scale industrial data and four public benchmarks across three tasks. Not only do we quantitatively assess our model, but we also provide case studies to broaden the understanding of our C-STAR framework.

2 RELATED WORK

Probability Distribution Distance Learning. To quantify the distance between probability distributions, one may utilize *divergences* such as Kullback–Leibler divergence [37], Jensen–Shannon divergence [15], or *metrics* like *Hellinger distance* [26]. Among these measurement tools, Wasserstein metric [29], known for its rigorous mathematical properties, has garnered attention in the machine learning community, particularly in generative modeling [1, 52, 58]. Despite its advantages, the classic Wasserstein distance suffers from high computational costs, particularly for high-dimensional distributions. In contrast to numerical optimization methods [12, 34, 54], recent studies of *Sliced-Wasserstein distance* [2, 32] has significantly reduced computational requirements. The general idea is to obtain adequate linear projections of the original distribution onto multiple one-dimensional distributions, followed by averaging the distances between these projected counterparts. This is facilitated by the closed-form solution of one-dimensional Wasserstein distance. Consequently, Sliced-Wasserstein distance has found application in various practical tasks [3, 33, 35, 42, 46], including our proposed modeling of high-dimensional trajectory distribution similarity.

E-commerce Customer Representation Learning. Representation learning for customer understanding constitutes a fundamental aspect of modern E-commerce recommender systems [59, 61]. Early methods focus on leveraging customer information such as profiles [11] or social relationships [44]. However, privacy concerns prompt the development of models that prioritize anonymity, leading to the learning particularly from item engagement sequences

[27, 78]. Another line of research centers on exploiting the inherent structure in the customer-item graph. In these models, each customer is represented by a unique and anonymous ID, and collaborative filtering signals between customers and their engaged items are captured [67]. These approaches often employ graph convolutional network (GCN) frameworks, known for their flexibility and adaptability in learning latent graph information [6, 19, 40, 45, 56, 57, 70, 77] with substantial advancements in recent years [24, 71, 73]. Different from sequential methods, these graph-based models typically require the input graph to be fixed in a customer-item adjacent matrix, which may hinder their ability to generalize to unobserved customers.

3 PROBLEM FORMULATION

PR-Graph. Our platform utilizes a product relational graph, referred to as PR-Graph, as a means to consolidate high-level product knowledge for various research and application purposes. PR-Graph is represented in the graph format as $\mathcal{G} = (\mathcal{T}, \mathcal{E}, \mathcal{V})$[1], where $\mathcal{T}$ represents all graph nodes, and $\mathcal{E} \subseteq \mathcal{T}\times\mathcal{T}$ denotes the edges connecting these nodes. Each node in $\mathcal{T}$ associated with a list of d-dimensional feature embeddings represented by $\mathcal{V} \in \mathbb{R}^{|\mathcal{T}|\times d}$. Notation explanations are in Table 1.

Table 1: Notations and meanings.

Notation	Explanation
$\mathcal{G}$=$(\mathcal{T}, \mathcal{E}, \mathcal{V})$	PR-Graph with sets of nodes, edges, and features.
$\mathcal{G}_i$=$(\mathcal{T}_i, \mathcal{E}_i, \mathcal{V}_i)$	Customer trajectory pattern.
$\mathcal{T}_i$=$[t_n^i]_{n=1}^{N_i}$	Node list of N_i trajectory elements.
$\mathcal{V}_i$=$[\boldsymbol{v}_n^i]_{n=1}^{N_i}$	Feature list associated with N_i trajectory elements.
$f_{\#}P$	Pushfoward of distribution P.
$W_p(\cdot,\cdot)$, $SW_p(\cdot,\cdot)$	p-Wasserstein distance, Sliced p-Wasserstein distance.
$F_P(\cdot)$, $F_P^{-1}(\cdot)$	Cumulative distribution function, quantile function.
$\mathbb{R}$, $\mathbb{S}$	Euclidean space and unit hypersphere.
$\boldsymbol{\theta}$	Unit vector in $\mathbb{S}$.
$g_{\boldsymbol{\theta}}(\cdot)$	Linear projection function with parameter vector $\boldsymbol{\theta}$.
P_0, P_i	Reference distribution and input distribution.
$P_0^{\boldsymbol{\theta}}$, $P_i^{\boldsymbol{\theta}}$	The slices of P_0, P_i derived by $\boldsymbol{\theta}$.
$f^*(\cdot)$	Optimal transport map between two distributions.
$\delta(\cdot)$	Dirac delta function.
$\tau(\cdot\vert\cdot)$	Ascending rank in the sorting of the given list.
$\mathcal{V}_i^{\boldsymbol{\theta}}$, $\mathcal{V}_0^{\boldsymbol{\theta}}$	Feature lists associated with distribution slices.
$\texttt{Inter-SE}(\cdot)$	Inter-Trajectory Similarity Encoder.
$\boldsymbol{E}_i$	Embedding of $\mathcal{G}_i$ with inter-trajectory information.
$Ngh(\cdot)$	Set of all neighbor nodes of the input.
$\boldsymbol{v}^{(l)}_{Ngh(t_n^i)}$	Neighborhood feature embedding of t_n^i at the l-th layer.
$\mathcal{N}_i^{(l)}$=$[\boldsymbol{v}^{(l)}_{Ngh(t_n^i)}]_{t_n^i\in\mathcal{T}_i}$	Neighborhood feature list at the l-th layer.
$\texttt{Intra-CE}(\cdot)$	Intra-Trajectory Correlation Encoder.
$\boldsymbol{E}_i^{'}$	Embedding of $\mathcal{G}_i$ with intra-trajectory information.
$\boldsymbol{E}_i^{\star}$	Ultimate trajectory representation.
$Pr(\cdot)$	Sampling probability.
r_t	Local ranking of node t in its belonging trajectory.
t^+, t^-, $\boldsymbol{v}_{t^+}$, $\boldsymbol{v}_{t^-}$	Positive and negative nodes and embeddings.
$\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}$	Margin ranking loss terms and objective function.
Δ	Set of all trainable embeddings and variables.

PR-Graph categorizes all products into approximately 15K nodes[2] within $\mathcal{T}$ and establishes 417K linkages in $\mathcal{E}$ to encapsulate strongly-correlated product relations, e.g., co-purchases.

Problem Formulation. As $\mathcal{T} = [t_n]_{n=1}^{|\mathcal{T}|}$ denotes all observed nodes in $\mathcal{G}$, for each customer i, their interactions over nodes in $\mathcal{T}$ can be snapshotted as $\mathcal{T}_i \subseteq \mathcal{T}$, i.e., $\mathcal{T}_i = [t_n^i]_{n=1}^{N_i}$ with N_i elements. Although $\mathcal{T}_i$ is 1-dimensional structure, leveraging the topological information in $\mathcal{G}$, we derive the customer's shopping trajectory $\mathcal{T}_i$ as a unique sub-graph $\mathcal{G}_i \subseteq \mathcal{G}$. Thus, our objective is to learn the trajectory representation from the knowledge contained in the 1&2-dimensional list-graph data $(\mathcal{T}_i, \mathcal{G}_i)$ such that the learned representation satisfies the following criterion simultaneously:

- **Inter-Trajectory Similarity Measurement.** This offers a macro view of trajectory representations in the latent space, enhancing customer understanding. It is particularly beneficial for applications like customer segmentation, where a holistic measurement of trajectory-wise (or customer-wise[3]) similarity is required.
- **Intra-Trajectory Content Summarization.** This provides a micro view of trajectory elements to enhance understanding of trajectory semantics. It facilitates applications such as shopping intent identification and trajectory completion by capturing the crucial correlations between elements.

4 C-STAR METHODOLOGY

Figure 2 depicts the workflow of the model. As previously mentioned, we address the first criterion, namely inter-trajectory similarity measurement, by quantifying the distance between trajectory probability distributions. Leveraging Optimal Transport theory, which offers rigorous mathematical properties [29], we begin by laying out the preliminaries and formally deriving our method.

4.1 Preliminaries: Optimal Transport

Optimal transport (OT) is the general problem of moving one distribution of mass, e.g., P, to another, e.g., Q, as efficiently as possible. Formally, given a probability distribution P, let random variable $\boldsymbol{X} \sim P$, $\boldsymbol{X} \in \mathbb{R}^d$. If $f: \mathbb{R}^d \to \mathbb{R}$, then $f_{\#}P$ is the push-forward of P, i.e., $f_{\#}P(Y) = P(\{\boldsymbol{x} : f(\boldsymbol{x}) \in Y\}) = P(f^{-1}(Y))$. The **Wasserstein distance** between P and Q, as the derived cost of their optimal transport plan, is defined with L_p transport cost [66]:

$$W_p\,(P, Q) = \left(\inf_{f\in TP(P,Q)} \int \|\boldsymbol{x} - f(\boldsymbol{x})\|^p dP(\boldsymbol{x})\right)^{\frac{1}{p}},\ p \geq 1, \quad (1)$$

where the infimum is over all possible transport plans. If a minimizer exists, denoted by f^*, it is thus the solution to the OT problem.

For *one-dimensional* distributions, there is a close-form solution to compute f^*, as $\underline{f^*(x) := F_P^{-1}(F_Q(x))}$, where F is the cumulative distribution function (CDF) associated with P. $F_P^{-1}(x)$ is the quantile function of P. For the *high-dimensional* case, due to its numerical intractability issue [32], the metric of *sliced-Wasserstein distance* has been recently investigated [2, 13, 50] and defined as:

$$SW_p\,(P, Q) = \left(\int_{\mathbb{S}^{d-1}} \left(W_p(g_{\boldsymbol{\theta}\#}P, g_{\boldsymbol{\theta}\#}Q)\right)^p d\boldsymbol{\theta}\right)^{\frac{1}{p}}. \quad (2)$$

Here $g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ and $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ is a unit vector in $\mathbb{R}^d$, and $\mathbb{S}^{d-1}$ is the unit d-dimensional hypersphere. $g_{\boldsymbol{\theta}\#}P$ is the push-forward of P with $g_{\boldsymbol{\theta}}$. This metric satisfies **positive-definiteness**, **symmetry**, and **triangle inequality** [32, 35], qualified for distance measurement in our proposed model to capture trajectory-wise similarity.

4.2 Inter-Trajectory Distribution Similarity

Following [46], we consider a list of probability measures $[P_i]_{i=1}^M$ defined in $\mathbb{R}^d$ for M observed trajectories. For each trajectory, there

[1]Data statistics are in § 5. [2]"Customer" and "trajectory" are interchangeable as each customer owns a unique trajectory.

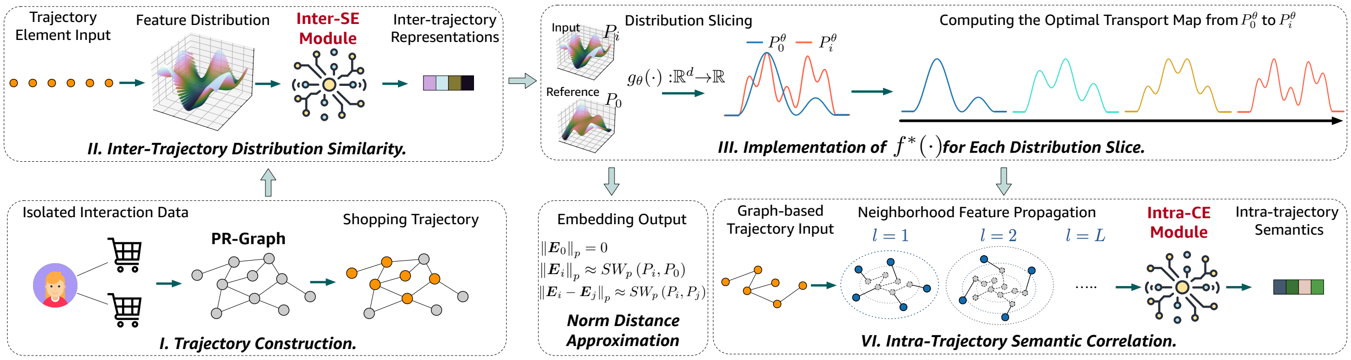

Figure 2: Illustration of C-STAR for coarse-to-fine trajectory representation learning (best view in color).

is a unique associated feature list $\mathcal{V}_i = [\boldsymbol{v}_{t_n^i} \in \mathbb{R}^d]_{n=1}^{N_i}$ with N_i elements. We assume that these feature elements are sampled from the underlying distribution P_i, and what we have snapshot is the *empirical (discrete) distribution* $\widehat{P}_i$ with its empirical CDF as:

$$F_{\widehat{P}_i}(\boldsymbol{x}) = \frac{1}{N_i}\sum_{n=1}^{N_i}\delta(\boldsymbol{x} \geq \boldsymbol{v}_{t_n^i}), \tag{3}$$

where $\delta(\cdot)$ returns 1 if the input is zero and 0 otherwise[4]. Generally, we believe empirical distributions are representative, i.e., $\widehat{P}_i \approx P_i$; thus we refer P_i to $\widehat{P}_i$ hereafter to avoid notation abuse.

To explicitly measure the trajectory-wise similarity, we propose to compare the input trajectory distribution with a certain *trainable reference* that functions as the "origin" in the trajectory embedding space. Specifically, we introduce a reference distribution P_0 with the embedding list $\mathcal{V}_0 = [\boldsymbol{v}_{t_n^0} \in \mathbb{R}^d]_{n=1}^{N}$, elements in which are the trainable embeddings. Then our target is: to get the distance between the distribution pair (P_0, P_i) to guide the learning of associated trajectory representations $(\boldsymbol{E}_0, \boldsymbol{E}_i)$ with a matched distance measurement back in the embedding space.

4.2.1 *Implementing the Optimal Transport Plan f^*.*

Noticing that f^* is crucial to determine the (shortest) distribution distance, thus, we formally introduce how we algorithmically implement it. This lays the foundation to construct the proposed trajectory representation encoder afterwards.

As explained in § 4.1, directly solving the high-dimensional optimal transport is extremely difficult; thus, we would first conduct the distribution slicing for one-dimensional Wasserstein distance computing. Let $g_{\boldsymbol{\theta}}(\boldsymbol{x})$ denote the linear projection function. For notation simplicity, we use $P_i^{\boldsymbol{\theta}} := g_{\boldsymbol{\theta}\#}P_i$ to denote the slice of P_i *w.r.t.* $g_{\boldsymbol{\theta}}$ (i.e., $P_i^{\boldsymbol{\theta}}$ is the push-forwarded one-dimensional distribution in $\mathbb{R}$); similarly $P_0^{\boldsymbol{\theta}} := g_{\boldsymbol{\theta}\#}P_0$. With proofs attached in Appendix A, we firstly introduce the following proposition:

Proposition 1. *f^* from reference slice $P_0^{\boldsymbol{\theta}}$ to the input distribution slice $P_i^{\boldsymbol{\theta}}$ is formulated as:*

$$f^*(x^{\boldsymbol{\theta}}|\mathcal{V}_i^{\boldsymbol{\theta}}) := F^{-1}_{P_i^{\boldsymbol{\theta}}}(F_{P_0^{\boldsymbol{\theta}}}(x^{\boldsymbol{\theta}})), \quad x^{\boldsymbol{\theta}} \in \mathcal{V}_0^{\boldsymbol{\theta}}. \tag{4}$$

To differentiate the high-dimensional input of Eqn. (3), $x^{\boldsymbol{\theta}}$ denotes the projected input of Eqn. (4) that lives in $\mathbb{R}$. For each sliced empirical distribution $P_i^{\boldsymbol{\theta}}$, the corresponding features are $\mathcal{V}_i^{\boldsymbol{\theta}} = [\boldsymbol{\theta}^\top \boldsymbol{v}_{t_n^i}]_{n=1}^{N_i}$. Similarly, the sliced reference list is $\mathcal{V}_0^{\boldsymbol{\theta}} = [\boldsymbol{\theta}^\top \boldsymbol{v}_{t_n^0}]_{n=1}^{N}$. Notice that their empirical CDFs, e.g., $F_{P_0^{\boldsymbol{\theta}}}(x^{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{n=1}^{N}\delta(x^{\boldsymbol{\theta}} - \boldsymbol{\theta}^\top \cdot \boldsymbol{v}_{t_n^0})$, is monotonically increasing. This implies that, if we know the ranking of each input $x^{\boldsymbol{\theta}}$ in the *ascending sorting* of $\mathcal{V}_0^{\boldsymbol{\theta}}$, denoted by $\tau(x^{\boldsymbol{\theta}}|\mathcal{V}_0^{\boldsymbol{\theta}})$, and $N = N_i$, the optimal transport plan f^* can be more quantitatively interpreted as follows:

Proposition 2. *If $N = N_i$, $\forall x^{\boldsymbol{\theta}} \in \mathcal{V}_0^{\boldsymbol{\theta}}$, the optimal plan in Eqn. (4) functions as the mapping between $\mathcal{V}_0^{\boldsymbol{\theta}}$ and $\mathcal{V}_i^{\boldsymbol{\theta}}$:*

$$f^*(x^{\boldsymbol{\theta}}|\mathcal{V}_i^{\boldsymbol{\theta}}) = \operatorname{argmin}_{x' \in \mathcal{V}_i^{\boldsymbol{\theta}}}\Big(\tau(x'|\mathcal{V}_i^{\boldsymbol{\theta}}) = \tau(x^{\boldsymbol{\theta}}|\mathcal{V}_0^{\boldsymbol{\theta}})\Big). \tag{5}$$

Notice that, indicator $\tau(\cdot)$ can be actually pre-processed via "`argsort`" to $\mathcal{V}_i^{\boldsymbol{\theta}}$ and "`sort`" to $\mathcal{V}_0^{\boldsymbol{\theta}}$. However, the major inadequacy is that Eqn. (5) requires $|N_i| = N$, which may not always be the case as the trajectory size varies in practice. To align the *feature list cardinalities* (i.e., for the case of $N_i \neq |N|$) but not demolish their original semantics, one neat yet effective solution is to conduct *linear interpolation*, as it is essentially a process for data continuing. Formally, we summarize our algorithmic implementation as:

For different cardinalities of feature lists $\mathcal{V}_i^{\boldsymbol{\theta}}$ and $\mathcal{V}_0^{\boldsymbol{\theta}}$, we implement $f^*(x^{\boldsymbol{\theta}}|\mathcal{V}_i^{\boldsymbol{\theta}}) = F^{-1}_{P_i^{\boldsymbol{\theta}}}(F_{P_0^{\boldsymbol{\theta}}}(x^{\boldsymbol{\theta}}))$ between $(\mathcal{V}_i^{\boldsymbol{\theta}}, \mathcal{V}_0^{\boldsymbol{\theta}})$ with the following mapping process:

$$f^*(x^{\boldsymbol{\theta}}|\mathcal{V}_i^{\boldsymbol{\theta}}) = \operatorname{argmin}_{x' \in \mathcal{V}_i^{\boldsymbol{\theta}}}\Big(\tau(x'|\mathcal{V}_i^{\boldsymbol{\theta}}) \geq \frac{N_i}{N}\cdot\tau(x^{\boldsymbol{\theta}}|\mathcal{V}_0^{\boldsymbol{\theta}})\Big). \tag{6}$$

4.2.2 *Inter-Trajectory Similarity Encoding.*

For each pair of distribution slices, e.g., $(P_0^{\boldsymbol{\theta}}, P_i^{\boldsymbol{\theta}})$, their optimal transport plan produces the shortest one-dimensional distance, i.e., $W_p(P_0^{\boldsymbol{\theta}}, P_i^{\boldsymbol{\theta}})$. According to the theory shown in Eqn. (2), the next step is to traverse all $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ for the ultimate integral between original distributions (P_0, P_i). However, this may be infeasible in practice to draw an infinite number of projections; therefore, in this work, with $\boldsymbol{\theta}_s$ denoting the s-th projection parameter uniformly sampled from $\mathbb{S}^{d-1}$, we approach this target with the Monte-Carlo approximation. Consequently, this leads to a cumulative sliced-Wasserstein distance for the original trajectory distributions:

$$SW_p(P_0, P_i) \approx \Big(\frac{1}{S}\sum_{s=1}^{S} W_p(P_0^{\boldsymbol{\theta}_s}, P_i^{\boldsymbol{\theta}_s})^p\Big)^{\frac{1}{p}}. \tag{7}$$

Based on the algorithmic implementation shown in Eqn. (6) with the associated distance regularization, we proceed to encode the

[4] $\int \delta(x)dx = 1$ for continuous inputs.

trajectory representation accordingly. Let $\Theta = \{\boldsymbol{\theta}_s\}_{s=1}^{S}$ denote the set of sampled projection parameters. Firstly, we encode the vector $\boldsymbol{O} \in \mathbb{R}^{N \cdot S}$ from the embedding reference $\mathcal{V}_0 = [\boldsymbol{v}_{t_n^0}]_{n=1}^{N}$ of P_0 as:

$$\boldsymbol{O} := \frac{1}{SN} \Big\|_{s=1}^{S} \Big\|_{n=1}^{N} \boldsymbol{\theta}_s{}^{\mathsf{T}} \boldsymbol{v}_{t_n^0}. \tag{8}$$

$\|$ denotes the concatenation operation along the innermost dimension. Given the input feature list $\mathcal{V}_i$, our `Inter-Trajectory Similarity Encoder (Inter-SE)` is formally defined as follows:

$$\texttt{Inter-SE}(\mathcal{V}_i|\Theta) := \frac{1}{SN} \Big\|_{s=1}^{S} \Big\|_{n=1}^{N} f^*(\boldsymbol{\theta}_s^{\mathsf{T}} \boldsymbol{v}_{t_n^0} | \mathcal{V}_i^{\boldsymbol{\theta}_s}) - \boldsymbol{O}, \tag{9}$$

Let $\boldsymbol{E}_i \in \mathbb{R}^{N \cdot S}$ denote the encoded representation. we have the following theorem with proof in Appendix A.

Theorem 1 (L_p **Norm Distance Approximation**). *For any two input trajectories with corresponding distributions P_i and P_j, their encoded representations $\boldsymbol{E}_i$ and $\boldsymbol{E}_j$ hold that: (1) $\|\boldsymbol{E}_i - \boldsymbol{E}_j\|_p \approx SW_p(P_i, P_j)$ and (2) $\|\boldsymbol{E}_i\|_p \approx SW_p(P_i, P_0)$.*

By setting $p = 2$, $\|\boldsymbol{E}_i - \boldsymbol{E}_j\|_2$ is exactly the Euclidean distance form that is more favorable to scenarios for recalling vectorized objects.

4.3 Intra-Trajectory Semantic Correlation

Due to the discreteness of empirical distributions, solely encoding the inter-trajectory similarity is coarse-grained, as it essentially captures the appearance disparity of trajectory features. However, the trajectory elements are further *semantically correlated*. To fuse such knowledge and enrich trajectory representations, we propose to learn the *fine-grained* trajectory-element semantic correlation. We leverage PR-Graph to have the graph-based trajectories and then employ the graph convolutional paradigm, mainly because of its powerful ability to learn high-order graph information [69]. However, vanilla designs [19, 31, 65] usually aim to encapsulate graph information into condensed outputs, which may lead to limited expressivity to measure the *correlation level*. To tackle this issue, we notice that if nodes are more correlated via the edges in PR-Graph, their local neighborhood structures tend to be more isomorphic. Based on this intuition, we introduce our *Intra-Trajectory Correlation Encoding* as follows.

4.3.1 ***Intra-Trajectory Correlation Encoding.*** For each node in trajectory $\mathcal{G}_i$, its neighbor node features can be collected as $\mathcal{N}_i = [\boldsymbol{v}_{t_\alpha^i}]_{t_\alpha^i \in Ngh(\mathcal{T}_i)}$, where $Ngh(\mathcal{T}_i)$ returns all neighbors of $\mathcal{T}_i$'s elements in PR-Graph. As shown in Figure 3(A), highly-correlated nodes should have the similar neighborhood. To explicitly embed such information, a naive solution will be to find the optimal transport map directly for $\mathcal{N}_i$, providing the correlation measurement *w.r.t.* neighbor node distributions. However, the major concern is that the size of $\mathcal{N}_i$ *exponentially* increases in the higher-order graph structure, leading to an intractable computational process.

To circumvent this issue, we develop approximation of high-order neighborhood features in layer-wise PR-Graph knowledge propagation. For each node t_n^i of $\mathcal{T}_i = [t_n^i]_{n=1}^{N_i}$, we first summarize its neighborhood feature embedding at the l-th iteration ($l > 0$):

$$\boldsymbol{v}_{Ngh(t_n^i)}^{(l)} = \sum\nolimits_{t \in Ngh(t_n^i)} \frac{w_{t,t_n^i}^{(l-1)}}{\sqrt{|Ngh(t)|+1}\sqrt{|Ngh(t_n^i)|+1}} \boldsymbol{v}_{t_n^i}^{(l-1)}. \tag{10}$$

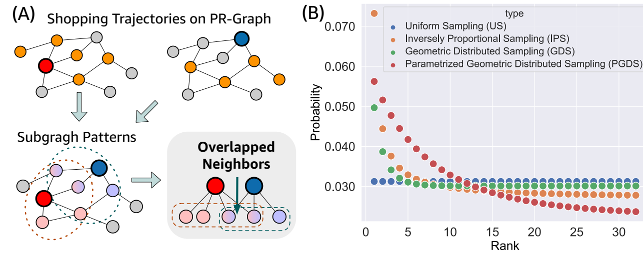

Figure 3: (A) Illustration of correlated neighborhood. (B) Probability curves of sampling strategies, where (1) sampling size $= 32$, (2) $\rho = 0.5$ for GDS, and (3) $\lambda = 10$ for PGDS.

$w_{t,t_n^i}^{(l-1)} \in \mathbb{R}$ and $\boldsymbol{v}_{t_n^i}^{(0)}$ is initialized by $\boldsymbol{v}_{t_n^i}$ in $\mathcal{V}_i$. We then re-define $\mathcal{N}_i$ as the layer-wise *neighborhood feature list* as follows:

$$\mathcal{N}_i^{(l)} = \Big[\boldsymbol{v}_{Ngh(t_n^i)}^{(l)}\Big]_{t_n^i \in \mathcal{T}_i}. \tag{11}$$

Based on these propagated neighborhood features, we have our `Intra-Trajectory Correlation Encoder (Intra-CE)` as:

$$\texttt{Intra-CE}(\mathcal{V}_i|\Theta) := \sum\nolimits_{l=1}^{L} \alpha_l \Big(\frac{1}{SN} \Big\|_{s=1}^{S} \Big\|_{n=1}^{N} f^*(\boldsymbol{\theta}_s^{\mathsf{T}} \boldsymbol{v}_{t_n^0} | \mathcal{N}_i^{(l)}) - \boldsymbol{O} \Big). \tag{12}$$

α_l is the l-th coefficient and, for brevity, we set it as $\alpha_l = 1/L$. Notice that node embeddings are usually updated with neighborhood information iteratively, i.e., $\boldsymbol{v}_{t_n^i}^{(l)} = \mathrm{AGG}(\boldsymbol{v}_{Ngh(t_n^i)}^{(l)}, \boldsymbol{v}_{t_n^i}^{(l-1)})$. $\mathrm{AGG}(\cdot)$ abstracts different aggregators [19, 31, 65, 69] and we directly follow classic GCN [31]. With $\boldsymbol{E}_i$ and $\boldsymbol{E}_i'$ denoting outputs from `Inter-SE` and `Intra-CE`, we complete the trajectory representation as: $\boldsymbol{E}_i^\star = [\boldsymbol{E}_i, \boldsymbol{E}_i'] \in \mathbb{R}^{2NS}$. As shown in § 5.3, both modules make pragmatic contributions to model performance over all evaluation tasks.

4.4 C-STAR Pre-training Strategy

4.4.1 ***Pre-training Objectives.*** Pre-training is an effective strategy to enhance model performance [41, 55]. We design following pre-training objectives by capturing two data properties:

- *Inter-trajectory Element Overlaps.* Given two trajectories, the more overlapping trajectory elements they share, the more similar they tend to be with common historical preferences. Therefore, for each trajectory, e.g., $\mathcal{T}_i$, we explicitly rank its similar counterparts based on the number of overlapping elements and use it as the supervision signal. Specifically, let Ω_i denote such ranking list associated with $\mathcal{T}_i$; we retrieve the trajectory pair $(\mathcal{T}_j, \mathcal{T}_k) \in \Omega_i$ such that $\mathcal{T}_i$ shares more overlapping elements with $\mathcal{T}_j$ than $\mathcal{T}_k$, i.e., $|\mathcal{T}_i \cap \mathcal{T}_j| > |\mathcal{T}_i \cap \mathcal{T}_k|$. The loss term is defined as follows:

$$\mathcal{L}_1 = \sum_{i=1}^{M} \sum_{(\mathcal{G}_j, \mathcal{G}_k) \in \Omega_i}^{H} \max\big(0, \|\boldsymbol{E}_i^\star - \boldsymbol{E}_j^\star\|_2 - \|\boldsymbol{E}_i^\star - \boldsymbol{E}_k^\star\|_2 + margin\big), \tag{13}$$

  where $\boldsymbol{E}_j^\star$, $\boldsymbol{E}_k^\star$ are the encoded trajectory embeddings of $\mathcal{T}_j$, $\mathcal{T}_k$.
- *Intra-trajectory Contextual Relations.* Another aspect to improve the trajectory embedding quality is to capture the trajectory-element relationship. We randomly pick the positive nodes, i.e., appeared elements in the given trajectory, and then maximize the matching scores between trajectory embedding and the chosen element embeddings. This aims to ensure that these matching scores surpass the scores computed from other negative nodes

that not appear in the trajectory. Our second loss term is:

$$\mathcal{L}_2 = \sum_{i=1}^{M} \sum_{t^+ \in \mathcal{T}_i, t^- \notin \mathcal{T}_i}^{H} \max\left(0, \boldsymbol{E}_i^{\star} \cdot \boldsymbol{v}_{t^+} - \boldsymbol{E}_i^{\star} \cdot \boldsymbol{v}_{t^-} + margin\right), \quad (14)$$

where $\boldsymbol{v}_{t^+}$, $\boldsymbol{v}_{t^-}$ denotes the embeddings of nodes t^+ and t^- that are relevant and irrelevant to $\mathcal{T}_i$, respectively.

Our complete pre-training objective function is formally defined:

$$\mathcal{L} = \mathcal{L}_1 + \mu \mathcal{L}_2 + \mu' \|\Delta\|_2^2. \quad (15)$$

μ, μ' are hyper-parameters and $\|\Delta\|_2^2$ L2-regularizes all trainable embeddings and variables to avoid over-fitting.

4.4.2 ***Sampling Strategy for Efficiency.*** Although C-STAR takes *variable-size* trajectory inputs for representation encoding, it is however more efficient and common to use fixed-size tensors, i.e., batches of trajectory feature lists, for pre-training. As the basic requirement is: sampled trajectories should be representative and informative, in this paper, we implement the *importance-indicator* by using *frequency* to rank all observed trajectory elements. Then, for an element $t \in \mathcal{T}$, based on $t's$ global frequency rank, its local rank r_t in its belonging trajectory can be subsequently obtained. Let $Pr(t)$ denote the derived sampling probability, we provide the following sampling strategies with illustrative curves in Figure 3(B):

(1) *Uniform Sampling (US).* The most common manner is to conduct uniform sampling without any probability bias.
(2) *Inversely Proportional Sampling (IPS).* This is based on the assumption that frequently-appeared elements should be assigned with higher probability. We implement it via an inversely proportional function with normalization:

$$Pr(t) = \frac{\exp(r_t^{-1})}{\sum_{t' \in \mathcal{T}} \exp(r_{t'}^{-1})}. \quad (16)$$

(3) *Geometric Distributed Sampling (GDS).* Given a hyper-parameter ρ, the probability can be assigned with:

$$Pr(t) = \frac{\exp(\rho(1-\rho)^{r_t})}{\sum_{t' \in \mathcal{T}} \exp(\rho(1-\rho)^{r_{t'}})} \text{ and } \rho \in (0,1). \quad (17)$$

(4) *Parametrized Geometric Distributed Sampling (PGDS).* A further modification is to control the curve sharpness:

$$Pr(t) = \frac{\exp(e^{-r_t/\lambda})}{\sum_{t' \in \mathcal{T}} \exp(e^{-r_{t'}/\lambda})} \text{ and } \lambda \in \mathbb{R}^+. \quad (18)$$

With the balanced performance over evaluation tasks shown in § 5.5, we finally adopt the geometric distributed sampling strategy. So far, we have explained all the technical parts of C-STAR. We attach the algorithm pseudo-codes in Algorithm 1.

4.5 Complexity Analysis

We provide complexity analysis in Table 2. (1) M and $\overline{N_i}$ are the trajectory number and the trajectory length after the sampling for tensorization. B and E are the batch size and epoch number. In each epoch, it takes $O(\overline{N_i} \log \overline{N_i})$ to implement f^* for each sliced distribution pair. Thus our `Inter-SE` takes $O(\frac{SEM\overline{N_i}\log\overline{N_i}}{B})$. (2) The graph normalization for weighting in Eqn. (10) can be pre-processed within $O(2|\mathcal{E}|)$, where $|\mathcal{E}|$ is the edge number of PR-Graph. (3) For the graph convolutions to extract PR-Graph knowledge into Eqn. (11), it takes $O(\frac{2LE|\mathcal{E}|^2}{B})$ complexity in total. (4) Based on aggregated embeddings, our `Intra-CE` module thus takes $O(\frac{SLEM\overline{N_i}\log\overline{N_i}}{B})$ complexity. (5) Lastly, the loss computation is $O(4EMH)$ where H is a small constant in Eqn's. (13) and (14) that $H \ll E$ and M.

Algorithm 1: C-STAR Learning Algorithm.

Input: Trajectories $\{\mathcal{G}_i\}_{i=1}^M$ with corresponding feature lists $\{\mathcal{V}_i\}_{i=1}^M$; variables Θ, Δ, S, N, μ, μ' L, ρ, $\cdots$

while not converge **do**
  for each trajectory $\mathcal{G}_i \in \{\mathcal{G}_i\}_{i=1}^M$ **do**
    $\mathcal{G}_i \leftarrow$ Sampled trajectory of $\mathcal{G}_i$;
    $\mathcal{V}_i \leftarrow$ Updated feature list associated with $\mathcal{G}_i$;
    $\boldsymbol{E}_i \leftarrow$ `Inter-SE`$(\mathcal{V}_i|\Theta)$ **for** $l \in \{1, 2, \cdots, L\}$ **do**
      $\{\mathcal{N}_i^{(l)}\} \leftarrow$ l-th neighborhood feature list
    $\boldsymbol{E}_i' \leftarrow$ Encode with `Intra-CE`$(\mathcal{V}_i|\Theta)$ and $\{\mathcal{N}_i^{(l)}\}_{l=1}^L$
    $\boldsymbol{E}_i^{\star} \leftarrow [\boldsymbol{E}_i, \boldsymbol{E}_i']$;
    $(\mathcal{T}_j, \mathcal{T}_k) \leftarrow$ Retrieve trajectories from Ω_i;
    $\boldsymbol{E}_j^{\star}, \boldsymbol{E}_k^{\star} \leftarrow$ Trajectory representations of $\mathcal{T}_j$ and $\mathcal{T}_k$;
    $t^+, t^- \leftarrow$ Retrieve relevant and irrelevant nodes;
    $\boldsymbol{v}_{t^+}, \boldsymbol{v}_{t^-} \leftarrow$ Get embeddings for t^+, t^-;
    $\mathcal{L}_1, \mathcal{L}_2 \leftarrow$ compute the loss terms;
  $\mathcal{L} \leftarrow$ Optimize C-STAR with regularization;
return Pre-trained model C-STAR.

Table 2: Training time complexity.

`Inter-SE`	Graph Norm.	Graph Conv.	`Intra-CE`	Pre-training
$O(\frac{SEM\overline{N_i}\log\overline{N_i}}{B})$	$O(2\|\mathcal{E}\|)$	$O(\frac{2LE\|\mathcal{E}\|^2}{B})$	$O(\frac{SLEM\overline{N_i}\log\overline{N_i}}{B})$	$O(4EMH)$

5 EXPERIMENTAL EVALUATION

In this section, we discuss our empirical model evaluation on large-scale industrial data with the following research questions. Due to the page limit, we report the experimental results of four public benchmarks in Appendix.

- **RQ1**: How does our proposed C-STAR perform in both online and offline evaluation?
- **RQ2**: How does different proposed modules contribute to C-STAR performance?
- **RQ3**: Can we provide empirical studies to broaden the understanding of C-STAR?
- **RQ4**: How do different settings influence the model performance?

5.1 A/B Testing Results (RQ1.A)

We have conducted an A/B testing with randomized live customer sessions, in both desktop and mobile platforms. For the control group, we show recommendations from previously deployed system; while for the treatment group, we show product recommendations based on C-STAR. The experiments had been running for two weeks, where we observed positive results with +1.24% improvement in product sales, +1.03% improvement in profit gain, and all the results are statistical significance with p-value ≤ 0.05.

5.2 Offline Evaluation (RQ1.B)

5.2.1 ***Downstream Tasks.*** We first introduce three important customer-centric tasks for our E-commerce platform.

- *Task 1: Customer Segmentation.* The fundamental property required by customer segmentation is customer-wise similarity measurement. Given query customers, the model seeks to retrieve their most similar customers, based on their learned trajectory embeddings.

Table 3: The statistics of three datasets.

	Task 1	Task 2	Task 3
# Pre-training trajectories		100,000	
# Avg. trajectory length		27.52	
# PR-Graph nodes		14,695	
# PR-Graph edges		416,610	
# PR-Graph density		0.0386	
# Fine-tuning trajectories	30,000	30,000	30,000
# Avg. trajectory length	28.45	26.96	27.47
# Evaluation trajectories	1,000,000	5,000,000	5,000,000
#Avg. trajectory length	29.22	26.52	27.52

- *Task 2: Shopping Trajectory Completion.* Assuming customers' shopping journeys have not yet finished, this task aims to complete customers' trajectories by recommending relevant yet unexplored elements.
- *Task 3: Shopping Intent Identification.* Predicting customers' shopping intentions gives rise to several scenarios such as *complementary recommendation.* For example, "*ankle braces*" are the complements of "*basketball shoes*", as they share the same intent, i.e., "playing basketball". This task is formulated as matching to labels that are the list of pre-defined shopping intents.

5.2.2 ***Datasets.*** We use customer engagements for the period of 28 days whereby the data are fully anonymized. To prevent data leakage risk, we process them separately for different tasks with their statistics reported in Table 3. For three downstream tasks, we collect the datasets for fine-tuning and evaluation as follows. For Task 1 of Customer Segmentation, about 1M data are collected following rule-based semantic similarity. Specifically, for each given customer trajectory, its similar trajectories are ranked by their common associated shopping intents, which are independently provided by our internal labeling mechanism. We then use these ranking lists for model fine-tuning and further evaluation. For Task 2 of Shopping Trajectory Completion, we collect around 5M new shopping trajectories and randomly hide 20% percent of trajectory elements. For Task 3 of Shopping Intent Identification, different from the 100K pre-training trajectories that are collected from the *click* data, we merge 5M additional trajectories from the *purchase* data. This is based on the intuition that, customers normally need to gather enough information before making a desirable deal. We then match the learned trajectory embeddings with the labeled intents for identification.

5.2.3 ***Metrics.*** For Task 1, we treat it as the Top-K ranking task towards candidates of similar/relevant customers. We directly use Eqn.(13) for fine-tuning via replacing Ω_i by the ground-truth ranking list. For Tasks 2 and 3, similarly, we replace the matching scores in Eqn. (14), i.e., $\boldsymbol{E}_i^{\star} \cdot \boldsymbol{v}_{t^+}$, by the scores between encoded trajectory embedding and unexplored product categories or shopping intents. Thus, we adopt Recall@K and NDCG@K as the evaluation metrics for Task 2. For Task 3, we use hit ratio (denoted as HitR) and AUC to evaluate the model classification capability.

5.2.4 ***Baselines.*** We include (1) shallow neural models (TPooling and MLP); (2) conventional GCN models (GCN$^+$ [31], GAT$^+$ [65], GraphSage$^+$ [19]); (3) language-based models (LSTM [28], Transformer [64], Graph Transformer [14]); (4) neural recommender models (DIN [79], DIEN [78], SURGE [4]) and (5) deep learning models (DeepSets [74], Set Transformer [38], PSWE [46]). We abbreviate model names for brevity, e.g., GS$^+$ for GraphSage$^+$. Detailed model descriptions are introduced in Appendix B.1.

We exclude early collaborative-filtering-based methods [25, 36, 51] and recent GCN-based recommender models[24, 67]. The reason is that these methods are **transductive** only for observed customers. Note that for the large-scale setting at E-commerce platforms, we therefore require a method with a good capability of conducting **inductive** inference.

5.2.5 ***Runtime Environment Settings.*** All codes are based on Python 3.8 and PyTorch 1.14.0. The experiments are run on a Linux machine with 4 NVIDIA 3090 GPUs. For all the baselines, we follow the officially reported hyper-parameter settings or apply a grid search in case lack recommended settings. The embedding dimension is searched in {32, 64, 128, 256}. The learning rate is tuned within $\{10^{-4}, 10^{-3}, 10^{-2}\}$. Coefficients μ and μ' are tuned among $\{0.1, 0.5, 1\}$ and $\{10^{-5}, 10^{-4}, 10^{-3}\}$, respectively. We initialize and optimize all models with default normal initializer and Adam optimizer [30].

5.2.6 ***Overall Performance.*** We report the average results based on a five-fold evaluation in Table 4, where the bold and the underlined represent the best- and second-best performing cases. We color cells when Wilcoxon signed-rank tests indicate the improvements are statistically significant with at least 95% confidence level.

- *Task 1: Customer Segmentation.* As reported in Table 4, (1) assembling with TPooling or MLP, graph-based implementations (i.e., GCN$^+$, GAT$^+$, and GraphSage$^+$) perform better than these two vanilla baselines, indicating that graph convolutional operations can effectively extract knowledge from PR-Graph to boost model performance. (2) Language models and neural recommender models generally underperform state-of-the-art deep learning models (i.e., DeepSets, Set Transformer, and PSWE) for the customer segmentation task. One explanation is that these deep methods organize the trajectory into a set structure, which can well capture their collective information and thus improve their trajectory-wise similarity measurement. (3) Our C-STAR consistently outperforms the second-best model by 4.74%~8.14% and 2.53%~7.88% *w.r.t.* Recall@K and NDCG@K. Moreover, the Wilcoxon significance tests indicate these improvements are all statistically significant with over 95% confidence level.
- *Task 2: Shopping Trajectory Completion.* From Table 4, we observe: (1) different from the under-performing situation in Task 1, language models generally work better in Task 2, compared to some recent deep learning models. The main reason is that, they can well capture the semantic relations between a "trajectory" and its "elements", similar to the case between a "sentence" and its "words". (2) Specialized recommender models present the best performance among all baselines; meanwhile, our proposed model C-STAR further achieves at least 0.62% and 0.85% of statistically significant improvements over Recall and NDCG metrics.
- *Task 3: Shopping Intent Identification.* For this task, we pre-train the model with *browse* data and use *purchase* data for fine-tuning and evaluation. Our C-STAR model presents competitive performance with positive improvement over two metrics. The fact that these two parts of data are independently collected from different data sources, basically substantiates our intuition in which customer shopping intentions are correlated during browsing and

Table 4: Evaluation results of three downstream tasks.

	Task 1								Task 2								Task 3	
	Top-5 (%)		Top-20 (%)		Top-50 (%)		Top-100 (%)		Top-5 (%)		Top-20 (%)		Top-50 (%)		Top-100 (%)		(%)	
Metric	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	HitR	AUC
TPooling	0.74	0.93	2.82	1.67	6.10	2.56	10.74	3.29	5.97	7.31	13.71	9.62	22.43	12.36	31.37	14.70	40.26	82.85
MLP	0.65	0.78	2.57	1.45	5.56	2.27	9.42	3.02	5.84	7.11	13.56	9.37	22.28	12.14	30.96	14.44	40.62	82.90
GCN+	2.21	4.49	7.48	6.38	12.36	7.83	15.29	9.29	6.74	8.35	15.25	10.79	25.27	13.59	35.69	15.98	43.39	83.81
GAT+	2.44	4.72	7.57	6.71	12.91	8.31	16.22	9.78	6.99	8.40	16.97	11.10	26.16	14.23	36.21	16.76	43.84	83.96
GS+	2.30	4.56	7.51	6.52	12.68	7.96	15.75	9.74	6.73	8.48	16.48	11.20	26.11	14.22	36.16	16.74	42.81	83.43
LSTM	2.87	4.95	7.53	6.98	13.21	9.67	18.28	11.53	6.85	8.99	17.24	11.78	27.74	15.06	37.79	17.81	40.28	83.43
TSFM	3.08	6.51	7.72	7.52	13.50	9.98	20.25	12.49	7.26	9.11	17.47	11.85	27.66	14.98	37.97	18.15	39.66	82.77
G-TSFM	3.10	6.39	7.89	7.64	13.56	10.04	20.19	12.53	6.81	8.69	16.53	11.18	26.52	14.28	36.84	17.11	43.78	84.29
DIN	2.92	6.62	7.93	7.90	13.67	10.31	20.62	12.68	7.52	9.13	19.05	12.77	28.84	15.83	38.08	18.21	43.45	83.52
DIEN	2.85	6.31	7.87	7.82	13.84	10.40	20.76	12.85	7.88	9.21	19.18	12.95	29.04	16.24	38.73	18.60	43.36	83.41
SURGE	2.96	6.57	8.13	7.77	13.96	10.43	20.74	12.96	7.93	9.23	19.29	13.08	29.13	16.31	38.67	18.72	44.47	83.76
DeepSets	3.13	6.62	7.83	7.59	13.95	10.27	20.67	12.74	6.29	7.60	15.01	10.21	24.96	13.32	35.12	15.97	35.13	82.24
S-TSFM	2.98	6.54	7.84	7.63	14.11	10.34	20.81	12.96	6.42	7.84	14.39	10.20	23.92	12.86	33.17	15.33	38.86	83.28
PSWE	3.18	6.85	8.35	8.13	14.59	10.83	21.73	13.44	7.87	9.27	17.87	12.26	28.09	15.23	38.05	18.13	44.25	84.17
C-STAR	**3.41**	**7.39**	**9.03**	**8.42**	**15.63**	**11.20**	**22.76**	**13.78**	**8.28**	**9.64**	**19.78**	**13.67**	**29.61**	**16.79**	**38.97**	**18.88**	**45.28**	**84.32**
Gain	7.23%	7.88%	8.14%	3.57%	7.13%	3.42%	4.74%	2.53%	4.41%	3.99%	2.54%	4.51%	1.65%	2.94%	0.62%	0.85%	1.82%	0.04%

Table 5: Results of ablation study.

	Task 1		Task 2		Task 3	
	Recall	NDCG	Recall	NDCG	HitR	AUC
w/o PT	14.94(-34.4%)	9.56(-30.6%)	28.45(-27.0%)	13.34(-29.3%)	38.43(-15.1%)	79.30(-6.0%)
w/o KE	21.21(-6.8%)	12.73(-7.6%)	33.14(-15.0%)	16.24(-14.0%)	43.58(-3.8%)	83.84(-0.6%)
w/o `Inter-SE`	17.53(-23.0%)	10.45(-24.2%)	33.85(-13.1%)	16.45(-12.9%)	42.94(-5.2%)	83.20(-1.3%)
w/o `Intra-CE`	20.42(-10.3%)	12.04(-12.6%)	35.63(-8.6%)	17.23(-8.7%)	43.69(-3.5%)	83.76(-0.7%)
Best	**22.76**	**13.78**	**38.97**	**18.88**	**45.28**	**84.32**

purchasing; more importantly, this provides the functionality of knowledge transfer from rich click data to the other, as purchase data is usually sparse and hard to directly mine knowledge from.

5.3 Ablation Study (RQ2)

We systematically study the effectiveness of each proposed module:

(1) *Trajectory Representation Learning with Pre-training.* Let the variant w/o PT skips the pre-training stage and starts from initializing trainable embeddings with normal initializer. As shown in Table 5, w/o PT exhibits a conspicuous performance decay across three tasks. This demonstrates the effect of our designs to improve the embedding quality by explicitly capturing the trajectory overlapping elements and trajectory-element relations as the supervision signals within the pre-training data.

(2) *PR-Graph Necessity for Knowledge Extraction.* Variant w/o KE omits the graph convolutions in PR-Graph by removing $\boldsymbol{E}_i'$ in $\boldsymbol{E}_i^\star = [\boldsymbol{E}_i, \boldsymbol{E}_i']$ and expanding the dimensionality of $\boldsymbol{E}_i$ to $2NS$ for fair comparison. As shown in Table 5, the performance decay of w/o KE not only indicates the informativeness of PR-Graph in organizing multiple product-to-product relations at the category-level, but also the usefulness of graph convolutions for knowledge extraction.

(3) *Impact of* `Inter-SE` *Module.* Variant w/o `Inter-SE` replaces the algorithmic implementation of Eqn. (9) by a two-layer of MLP to encode trajectory representations. The performance gaps of three tasks between w/o `Inter-SE` and C-STAR prove the effectiveness of our proposed solution, in which we convert the measurement of trajectory-wise similarity to the *distribution distance* with the Optimal Transport methodology.

(4) *Impact of* `Intra-CE` *Module.* We directly disable the `Intra-CE` module in Eqn. (12) and average the aggregated neighborhood embeddings in Eqn. (11) for $\boldsymbol{E}_i'$ in $\boldsymbol{E}_i^\star = [\boldsymbol{E}_i, \boldsymbol{E}_i']$. The notable performance degradation validates our early assumption that the conventional graph convolutions do not have a certain mechanism to maintain both the similarity measurement and semantic enrichment, which may lead to the disruption of learning balanced representations, especially for Tasks 1 and 2.

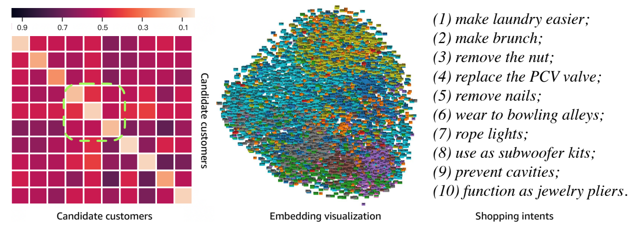

Figure 4: Illustration of case studies (best view in color).

Table 6: Trajectories with the same elements highlighted.

Customer ID	Shopping Trajectory (with anonymized index)
Query Customer	10041, 10048, 9706, 9881, 9965, 10133, 6487
Customer 1	6600, **9706**, **9965**, **10133**, 9887, 13402
Customer 2	3443, 5259, **9965**, 10635, 2365, 7609

5.4 Case Study (RQ3)

(1) *Embedding Distance for Similarity Measurement.* C-STAR uses the Euclidean distance for ranking similar customers. To visualize this process, we randomly select 10 trajectories and present their mutual distances from their fine-tuned embeddings. As shown in the left-most heatmap of Figure 4, with lighter-colored cells representing smaller distances after fine-tuning, the green block indicates these customers are likely similar in real-world space. In addition, given a query customer, we retrieve his/her two most similar customers (ranked by their distances) in Table 6. We observe that, similarity measurement using Euclidean distances with fine-tuned embeddings presents reasonable alignment in terms of overlapping trajectory structures, providing the explainability for our pre-training design.

(2) *Embedding Visualization with Shopping Intents.* We randomly pick 10 intents and use them to retrieve 10,000 customer trajectories that predict these 10 intents as their respective major

Table 7: Results of different module designs.

Type	Task 1 Recall	Task 1 NDCG	Task 2 Recall	Task 2 NDCG	Task 3 AUC	Task 3 HitR
US	22.68	13.73	38.43	18.35	44.95	83.95
IPS	22.57	13.64	38.14	17.95	**45.40**	84.12
GDS	22.76	13.78	**38.97**	**18.88**	45.28	**84.32**
PGDS	**22.95**	**13.83**	38.59	18.35	45.12	84.19
Non-attention-based	22.76	13.78	38.97	18.88	**45.28**	**84.32**
	Average pre-training time cost/Epoch: 233 s					
Attention-based	**23.13**	**13.94**	**39.69**	**19.20**	45.22	84.28
	Average pre-training time cost/Epoch: 815 s					

intents (i.e., with the highest prediction score from Task 3). We project the learned embeddings into a 2D-space with t-SNE [63]. Each projected vector is discriminatively colored with his/her major shopping intent. As shown in the right-hand-side two figures of Figure 4, these clustered trajectories generally indicate the same major intents, demonstrating that the trajectory embeddings are indeed effective for intent identification, as they are geometrically close in the embedding space.

5.5 Analysis on Model Settings (RQ4)

Sampling Strategy for Effective Training. We report the Top-100 results of all sampling strategies in Table 7. As we can observe, although these strategies produce different final performances over different tasks, the geometric distributed sampling strategy generally presents a balanced performance across all tasks.

Selection of Graph Convolution Paradigm. We experiment with attention-based graph convolutions [65] on PR-Graph. The results are generally better than the non-attention-based version; however, we notice both the computation time cost and memory footprint of attention-based version are much larger, e.g., nearly four times slower in pre-training. Therefore, in C-STAR, we employ the non-attention-based implementation. We leave the several advanced efficiency optimizations [8, 20, 47–49, 72, 76] for future exploration.

Settings of N and S. As demonstrated in Figure 5 for Task 1, compared to N, altering S is more influential to the performance, as increasing S produces a more accurate cumulative approximation. However, this also increases the computational cost. Thus, (32, 64) for (N, S) setting is the balanced spot with positive momentum that presents a practical trade-off between the model performance and resource consumption.

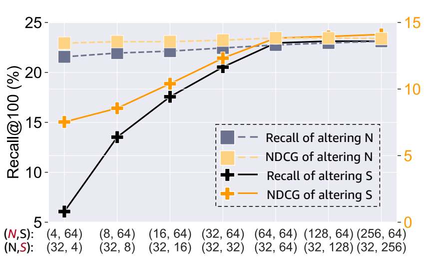

Figure 5: Altering (N, S).

6 CONCLUSION AND FUTURE WORK

We present C-STAR, a novel framework for learning informative representations of customer shopping trajectories. The proposed methodology jointly models two critical aspects: inter-trajectory distribution similarity and intra-trajectory semantic correlation. This coarse-to-fine learning paradigm enriches trajectory representations by leveraging both global feature distribution knowledge and local product relationships within individual trajectories. Furthermore, we specifically design two pre-training objectives to enhance the quality of the learned embeddings. The empirical results on both industrial and public datasets not only illustrate the usefulness of learned embeddings over multiple customer-centric tasks, but also justify the effectiveness of all proposed modules.

As for future work, we plan to investigate two major directions. Firstly, we aim to enhance our model by incorporating *multimodal information* [39, 43, 60] to enrich the object features, where the difficulty is to propose effective knowlegde fusion methodology. Secondly, considering that trajectory data undergoes continuous evolution, it is worth exploring streaming methods through *Continual Learning* [21, 75] instead of re-training the model. This approach enables us to efficiently capture emerging patterns while retaining the knowledge acquired by the initial model, which proves particularly efficacious in large-scale settings.

7 ACKNOWLEDGMENTS

The research presented in this paper was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF 2151185).

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In ICML. PMLR, 214–223.
[2] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and radon wasserstein barycenters of measures. JMIV 51, 1 (2015), 22–45.
[3] Mathieu Carriere, Marco Cuturi, and Steve Oudot. 2017. Sliced Wasserstein kernel for persistence diagrams. In ICML. PMLR, 664–673.
[4] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential Recommendation with Graph Neural Networks. In SIGIR. 378–387.
[5] Xiong-Hui Chen, Bowei He, Yang Yu, Qingyang Li, Zhiwei Qin, Wenjie Shang, Jieping Ye, and Chen Ma. 2023. Sim2rec: A simulator-based decision-making approach to optimize real-world long-term user engagement in sequential recommender systems. In ICDE. IEEE, 3389–3402.
[6] Yankai Chen, Yixiang Fang, Qiongyan Wang, Xin Cao, and Irwin King. 2024. Deep Structural Knowledge Exploitation and Synergy for Estimating Node Importance Value on Heterogeneous Information Networks. In AAAI. 8302–8310.
[7] Yankai Chen, Tuan Truong, Xin Shen, Ming Wang, Jin Li, Jim Chan, and Irwin King. 2023. Topological representation learning for e-commerce shopping behaviors. (2023).
[8] Yankai Chen, Yifei Zhang, Huifeng Guo, Ruiming Tang, and Irwin King. 2022. An Effective Post-training Embedding Binarization Approach for Fast Online Top-K Passage Matching. In AACL. 102–108.
[9] Yankai Chen, Yifei Zhang, Menglin Yang, Zixing Song, Chen Ma, and Irwin King. 2023. WSFE: Wasserstein Sub-graph Feature Encoder for Effective User Segmentation in Collaborative Filtering. In SIGIR. 2521–2525.
[10] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In SIGKDD. 1082–1090.
[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In RecSys. 191–198.
[12] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS 26 (2013).
[13] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander G Schwing. 2019. Max-sliced wasserstein distance and its use for gans. In CVPR. 10648–10656.
[14] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699 (2020).
[15] Dominik Maria Endres and Johannes E Schindelin. 2003. A new metric for probability distributions. IEEE TIT 49, 7 (2003), 1858–1860.
[16] Farzad Eskandanian, Bamshad Mobasher, and Robin D Burke. 2016. User Segmentation for Controlling Recommendation Diversity.. In RecSys.
[17] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. 2015. Learning image and user features for recommendation in social networks. In ICCV.
[18] Miguel Alves Gomes, Richard Meyes, Philipp Meisen, and Tobias Meisen. 2022. Will This Online Shopping Session Succeed? Predicting Customer's Purchase Intention Using Embeddings. In CIKM. 2873–2882.

[19] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In NeurIPS. 1025–1035.
[20] Bowei He, Xu He, Renrui Zhang, Yingxue Zhang, Ruiming Tang, and Chen Ma. 2023. Dynamic Embedding Size Search with Minimum Regret for Streaming Recommender System. In CIKM. 741–750.
[21] Bowei He, Xu He, Yingxue Zhang, Ruiming Tang, and Chen Ma. 2023. Dynamically expandable graph convolution for streaming recommendation. In Proceedings of the ACM Web Conference 2023. 1457–1467.
[22] Bowei He, Yunpeng Weng, Xing Tang, Ziqiang Cui, Zexu Sun, Liang Chen, Xiuqiang He, and Chen Ma. 2024. Rankability-enhanced Revenue Uplift Modeling Framework for Online Marketing. arXiv preprint arXiv:2405.15301 (2024).
[23] Ruining He and Julian McAuley. 2016. Modeling the visual evolution of fashion trends with one-class collaborative filtering. In WWW. 507–517.
[24] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In SIGIR. 639–648.
[25] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In WWW. 173–182.
[26] Ernst Hellinger. 1909. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. Journal für die reine und angewandte Mathematik 1909, 136 (1909), 210–271.
[27] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In ICLR.
[28] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
[29] Leonid V Kantorovich. 1960. Mathematical methods of organizing and planning production. Management science 6, 4 (1960), 366–422.
[30] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In ICLR.
[31] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.
[32] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. 2019. Generalized sliced wasserstein distances. NeurIPS 32 (2019).
[33] Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. 2018. Sliced Wasserstein auto-encoders. In ICLR.
[34] Soheil Kolouri, Akif B Tosun, John A Ozolek, and Gustavo K Rohde. 2016. A continuous linear optimal transport approach for pattern analysis in image datasets. Pattern recognition 51 (2016), 453–462.
[35] Soheil Kolouri, Yang Zou, and Gustavo K Rohde. 2016. Sliced Wasserstein kernels for probability distributions. In CVPR. 5258–5267.
[36] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8 (2009), 30–37.
[37] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The annals of mathematical statistics 22, 1 (1951), 79–86.
[38] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In ICML. PMLR, 3744–3753.
[39] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In SIGKDD. 1258–1267.
[40] Langzhang Liang, Xiangjing Hu, Zenglin Xu, Zixing Song, and Irwin King. 2023. Predicting Global Label Relationship Matrix for Graph Neural Networks under Heterophily. In NeurIPS.
[41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692 (2019).
[42] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus, and Fabian-Robert Stöter. 2019. Sliced-Wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In ICML. PMLR, 4104–4113.
[43] Sichun Luo, Bowei He, Haohan Zhao, Yinya Huang, Aojun Zhou, Zongpeng Li, Yuanzhang Xiao, Mingjie Zhan, and Linqi Song. 2023. RecRanker: Instruction Tuning Large Language Model as Ranker for Top-k Recommendation. arXiv preprint arXiv:2312.16018 (2023).
[44] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In WSDM. 287–296.
[45] Ziqiao Meng, Peilin Zhao, Yang Yu, and Irwin King. 2023. Doubly stochastic graph-based non-autoregressive reaction prediction. In IJCAI. 4064–4072.
[46] Navid Naderializadeh, Joseph F Comer, Reed Andrews, Heiko Hoffmann, and Soheil Kolouri. 2021. Pooling by sliced-Wasserstein embedding. NeurIPS 34 (2021), 3389–3400.
[47] Zexuan Qiu, Jiahong Liu, Yankai Chen, and Irwin King. 2024. HiHPQ: Hierarchical Hyperbolic Product Quantization for Unsupervised Image Retrieval. In AAAI. 4614–4622.
[48] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. 2021. Unsupervised Hashing with Contrastive Information Bottleneck. In IJCAI.
[49] Zexuan Qiu, Qinliang Su, Jianxing Yu, and Shijing Si. 2022. Efficient Document Retrieval by End-to-End Refining and Quantizing BERT Embedding with Contrastive Product Quantization. In EMNLP. 853–863.
[50] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2011. Wasserstein barycenter and its application to texture mixing. In SSVM. Springer, 435–446.
[51] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. arXiv (2012).
[52] Xin Shen, Kyungdon Joo, and Jean Oh. 2023. FishRecGAN: An End to End GAN Based Network for Fisheye Rectification and Calibration. Adv. Artif. Intell. Mach. Learn. 3, 2 (2023), 1180–1197.
[53] Xin Shen, Yan Zhao, Sujan Perera, Yujia Liu, Jinyun Yan, and Mitchell Goodman. 2023. Learning Personalized Page Content Ranking Using Customer Representation. arXiv preprint arXiv:2305.05267 (2023).
[54] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. ACM Transactions on Graphics (ToG) 34, 4 (2015), 1–11.
[55] Zixing Song, Ziqiao Meng, and Irwin King. 2024. A Diffusion-Based Pre-training Framework for Crystal Property Prediction. In AAAI, Vol. 38. 8993–9001.
[56] Zixing Song, Yifei Zhang, and Irwin King. 2023. Optimal Block-wise Asymmetric Graph Construction for Graph-based Semi-supervised Learning. In NeurPS.
[57] Zixing Song, Yuji Zhang, and Irwin King. 2023. Towards fair financial services for all: A temporal GNN approach for individual fairness on transaction networks. In CIKM. 2331–2341.
[58] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2018. Wasserstein auto-encoders. ICLR (2018).
[59] Quoc-Tuan Truong and Hady Lauw. 2019. Multimodal review generation for recommender systems. In The World Wide Web Conference. 1864–1874.
[60] Quoc-Tuan Truong, Aghiles Salah, and Hady Lauw. 2021. Multi-modal recommender systems: Hands-on exploration. In RecSys. 834–837.
[61] Quoc-Tuan Truong, Aghiles Salah, and Hady W Lauw. 2021. Bilateral variational autoencoder for collaborative filtering. In WSDM. 292–300.
[62] Quoc-Tuan Truong, Tong Zhao, Changhe Yuan, Jin Li, Jim Chan, Soo-Min Pantel, and Hady W Lauw. 2022. AmpSum: Adaptive Multiple-Product Summarization towards Improving Recommendation Captions. In WebConf. 2978–2988.
[63] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. JMLR 9, 11 (2008).
[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. NeurIPS 30 (2017).
[65] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. ICLR.
[66] Cédric Villani. 2009. Optimal transport: old and new. Vol. 338. Springer.
[67] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In SIGIR. 165–174.
[68] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In SIGIR. 1001–1010.
[69] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? ICLR (2019).
[70] Menglin Yang, Zhihao Li, Min Zhou, Jiahong Liu, and Irwin King. 2022. Hicf: Hyperbolic informative collaborative filtering. In SIGKDD. 2212–2221.
[71] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. 2022. HRCF: Enhancing collaborative filtering via hyperbolic geometric regularization. In Proceedings of the ACM Web Conference 2022. 2462–2471.
[72] Menglin Yang, Min Zhou, Rex Ying, Yankai Chen, and Irwin King. 2023. Hyperbolic Representation Learning: Revisiting and Advancing. ICML (2023).
[73] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In SIGKDD. 974–983.
[74] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. NeurIPS 30 (2017).
[75] Xinni Zhang, Yankai Chen, Chenhao Ma, Yixiang Fang, and Irwin King. 2024. Influential Exemplar Replay for Incremental Learning in Recommender Systems. In AAAI, Vol. 38. 9368–9376.
[76] Yifei Zhang, Yankai Chen, Zixing Song, and Irwin King. 2023. Contrastive Cross-scale Graph Knowledge Synergy. In SIGKDD.
[77] Yifei Zhang, Hao Zhu, Yankai Chen, Zixing Song, Piotr Koniusz, Irwin King, et al. 2023. Mitigating the popularity bias of graph collaborative filtering: A dimensional collapse perspective. NeurIPS 36 (2023), 67533–67550.
[78] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In AAAI. 5941–5948.
[79] 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 KDD. 1059–1068.
[80] Yuchen Zhuang, Xin Shen, Yan Zhao, Chaosheng Dong, Ming Wang, Jin Li, and Chao Zhang. 2023. G-STO: Sequential main shopping intention detection via graph-regularized stochastic transformer. In CIKM. 3677–3687.
[81] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In WWW. 22–32.

A THEORETICAL PROOFS

Proposition 1. *f^* from reference slice P_0^{θ} to the input distribution slice P_i^{θ} is formulated as:*

$$f^*(x^{\theta}|\mathcal{V}_i^{\theta}) := F_{P_i^{\theta}}^{-1}\big(F_{P_0^{\theta}}(x^{\theta})\big), \quad x^{\theta} \in \mathcal{V}_0^{\theta}. \tag{1}$$

Proof. The transport cost, i.e., its derived distance, as one candidate transport map for Eqn. (1), can be computed as:

$$\begin{aligned} \text{Distance} &= \Big(\int_{\mathbb{R}} \|x^{\theta} - f^*(x^{\theta}|\mathcal{V}_i^{\theta})\|^p dP_0^{\theta}(x^{\theta})\Big)^{\frac{1}{p}} \\ &= \Big(\int_{\mathbb{R}} \|x^{\theta} - F_{P_i^{\theta}}^{-1}\big(F_{P_0^{\theta}}(x^{\theta})\big)\|^p dP_0^{\theta}(x^{\theta})\Big)^{\frac{1}{p}} \\ &= \Big(\int_0^1 \|F_{P_0^{\theta}}^{-1}(r) - F_{P_i^{\theta}}^{-1}(r)\|^p dr\Big)^{\frac{1}{p}} = \underline{W_p(P_0^{\theta}, P_i^{\theta})}. \end{aligned} \tag{2}$$

This proves to be the optimal distance for these two slices. □

Proposition 2. *If $N = N_i$, $\forall x^{\theta} \in \mathcal{V}_0^{\theta}$, the optimal plan in Eqn. (4) functions as the mapping between $\mathcal{V}_0^{\theta}$ and $\mathcal{V}_i^{\theta}$:*

$$f^*(x^{\theta}|\mathcal{V}_i^{\theta}) = \operatorname{argmin}_{x' \in \mathcal{V}_i^{\theta}} \Big(\tau(x'|\mathcal{V}_i^{\theta}) = \tau(x^{\theta}|\mathcal{V}_0^{\theta})\Big). \tag{3}$$

Proof. Based on their empirical distributions:

$$\begin{aligned} F_{P_0^{\theta}}(x^{\theta}) &= \frac{1}{N}\sum_{n=1}^{N} \delta(x^{\theta} - \theta^{\top} \cdot v_{t_n^0}) \\ F_{P_i^{\theta}}(x^{\theta}) &= \frac{1}{N_i}\sum_{n=1}^{N_i} \delta(x^{\theta} - \theta^{\top} \cdot v_{t_n^i}), \end{aligned} \tag{4}$$

It is straightforward to find that they are monotonically increasing. If $N = N_i$, we can firstly modify the original form of the optimal transport map $F_{P_i^{\theta}}^{-1}\big(F_{P_0^{\theta}}(x^{\theta})\big)$ to:

$$f^*(x^{\theta}|\mathcal{V}_i^{\theta}) = \operatorname{argmin}_{x' \in \mathcal{V}_i^{\theta}} \big(F_{P_i^{\theta}}(x') = r\big) \text{ where } r = F_{P_0^{\theta}}(x^{\theta}). \tag{5}$$

Let $\tau(x^{\theta}|\mathcal{V}_0^{\theta})$ denote the ranking of each input x^{θ} in the *ascending sorting* of $\mathcal{V}_0^{\theta}$, and then we can quantitatively replace the term $F_{P_i^{\theta}}(\cdot)$ and completes the proof.

$$f^*(x^{\theta}|\mathcal{V}_i^{\theta}) = \operatorname{argmin}_{x' \in \mathcal{V}_i^{\theta}} \Big(\tau(x'|\mathcal{V}_i^{\theta}) = \tau(x^{\theta}|\mathcal{V}_0^{\theta})\Big) \text{ if } N = N_i. \tag{6}$$

□

Theorem 1 (L_p **Norm Distance Approximation**). *For any two input trajectories with corresponding distributions P_i and P_j, their encoded representations E_i and E_j hold that: (1) $\|E_i - E_j\|_p \approx SW_p(P_i, P_j)$ and (2) $\|E_i\|_p \approx SW_p(P_i, P_0)$.*

Proof. (1) For $\|E_i - E_j\|_p$, we have:

$$\begin{aligned} &\|E_i - E_j\|_p \\ &= \Big\| \big\|_{s=1}^{S} (E_i^{\theta_s} - E_j^{\theta_s}) \Big\|_p \\ &= \Big\| \sum_{s=1}^{S} (E_i^{\theta_s} - E_j^{\theta_s})^p \Big\|_p \\ &= \Big\| \frac{1}{S}\sum_{s=1}^{S} \frac{1}{N}\sum_{n=1}^{N} \Big(f^*(\theta_s^{\top} v_n^0|\mathcal{V}_i^{\theta_s}) - f^*(\theta_s^{\top} v_n^0|\mathcal{V}_j^{\theta_s})\Big)^p \Big\|_p \end{aligned} \tag{7}$$

where its continuous form is:

$$\begin{aligned} &\approx \Big\| \int_{\mathbb{S}^{d-1}} \int_{\mathbb{R}} \Big(F_{P_i^{\theta}}^{-1}\big(F_{P_0^{\theta}}(t)\big) - F_{P_j^{\theta}}^{-1}\big(F_{P_0^{\theta}}(t)\big)\Big)^p dP_0^{\theta}(t) d\theta \Big\|_p \\ &= \Big\| \int_{\mathbb{S}^{d-1}} \int_0^1 \big(F_{P_i^{\theta}}^{-1}(r) - F_{P_j^{\theta}}^{-1}(r)\big)^p dr d\theta \Big\|_p \\ &= \Big\| \int_{\mathbb{S}^{d-1}} W_p(P_i^{\theta}, P_j^{\theta})^p d\theta \Big\|_p \\ &= SW_p(P_i, P_j). \end{aligned} \tag{8}$$

(2) For the reference P_0, its encoded representation is:

$$\begin{aligned} E_0 := &\frac{1}{SN} \Big\|_{s=1}^{S} \Big\|_{n=1}^{N} \Big(f^*(\theta_s^{\top} v_n^0|\mathcal{V}_0^{\theta_s})\Big) - O \\ &= \frac{1}{SN} \Big\|_{s=1}^{S} \Big\|_{n=1}^{N} (\theta_s^{\top} v_n^0) - O = \mathbf{0}. \end{aligned} \tag{9}$$

Thus we have:

$$\|E_i\|_p = \|E_i - E_0\|_p \approx SW_p(P_i, P_0), \tag{10}$$

which completes the proof. □

B EXPERIMENTAL SETUPS

B.1 Baseline Descriptions

- **TPooling** is a straightforward implementation that aggregates all element embeddings of each customer trajectory. The pooling strategy could be *mean*, *max*, or *min*. We report the best performance of these strategies and denote it as TPooling, if no confusion is caused.
- **MLP** is a fundamental neural network that first concatenates trajectory element embeddings as input and passes them through one hidden layer and finally arrives at the output layer.
- **GCN** [31] is one of the classic graph convolutional networks. We implement it on PR-Graph to gather information and further aggregate trajectory-level embeddings via TPooling (i.e., **GCN+TPooling**) or MLP (i.e., **GCN+MLP**). We use the notation **GCN$^+$** to denote the one with better metrics (e.g., on Recall@K and NDCG@K in ranking tasks).
- **GAT** [65] is the representative graph-based model with the attention mechanism. Similarly, we implement it for PR-Graph information propagation and summarize trajectory embeddings with two variant models (i.e., **GAT+TPooling** and **GAT+MLP**). Similarly, **GAT$^+$** denotes the better variant.
- **GraphSage** [19] is the graph convolutional network with the inductive learning setting. Similarly, we have two implementations with TPooling and MLP, and **GraphSage$^+$** is the better one.
- **LSTM** [28] is a competitive baseline that directly takes trajectory data as sequential input to learn the holistic target representation.
- **Transformer** [64] is another strong baseline with the self-attention mechanism. In our implementation, we input each customer trajectory as a language sentence to learn its embedding.
- **Graph Transformer** [14] is one of state-of-the-art Transformer-based model that deploys on the graph data. We implement it on PR-Graphand trajectory data to jointly learn the representations.
- **DIN** [79] is a classical recommender model that attentively learns customer embeddings by aggregating the history interaction within the trajectories.

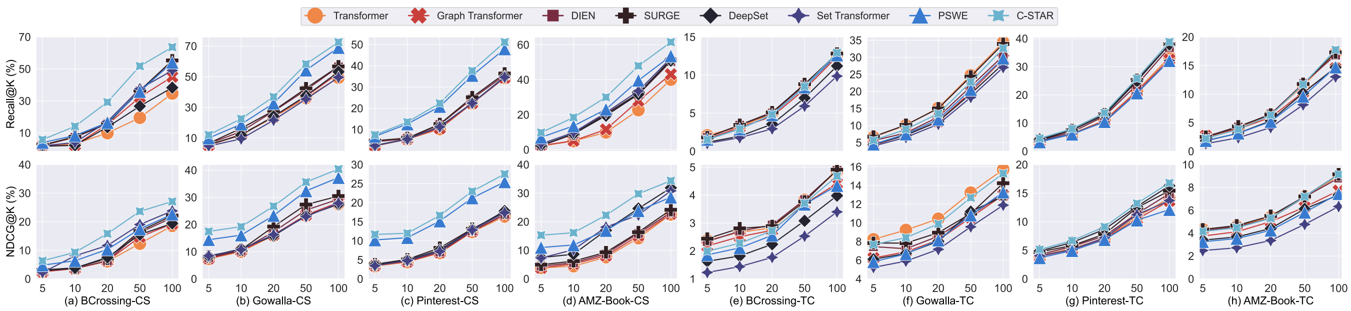

Figure 1: Recall@K and NDCG@K are respectively reported in the first and second row (best view in color).

Table 1: Public dataset statistics.

	BCrossing	Gowalla	Pinterest	AMZ-Book
#User trajectories	16,411	29,858	55,186	52,643
# Items	36,143	40,919	9,855	91,576
#Avg. trajectory length	35.711	49.272	26.516	56.686

- **DIEN** [78] is a representative recommender model that encodes customer representations by a two-layer GRU for sequential recommendation.
- **SURGE** [4] is a state-of-the-art recommender model that utilizes GCN learning ability to capture item-item relations for next-item sequential recommendation.
- **DeepSets** [74] is an exemplary deep learning model that is originally proposed to learn representations for "*compound objects*" such as point clouds. In our experiments, it takes all trajectory units as a set and learns the unified representation while maintaining the intrinsic semantics of trajectory elements.
- **Set Transformer** [38] is a state-of-the-art model that integrates self-attention mechanism [64] while lowering computation complexity. We adapt it to encode the trajectory embeddings.
- **PSWE** [46] is the latest state-of-the-art method that subsumes the learning process under the Wasserstein metric framework. In this work, we reproduce it to learn the trajectory embeddings.

C SUPPLEMENTARY EXPERIMENTS

C.1 Evaluation on Public Benchmarks

Public Datasets. We collect four public datasets that are widely evaluated for E-commerce recommendation [10, 17, 24, 67, 68, 81]. For these datasets, we synthesize their own "PR-Graph" by *creating edges if items are co-purchased by over 20 different customers.* Dataset statistics are reported in Table 1 with descriptions as follows.

- **BCrossing**[5] [81] is a public dataset of book ratings in Book-Crossing Community. To guarantee the dataset quality, we filter out readers and books with less than five interactions and then merge each reader's rated books into a unique trajectory.
- **Gowalla**[6] [67] is the check-in dataset [10] from Gowalla, where users share their locations by check-in. We directly use the dataset split by [67] where users and items are selected to have at least then interactions. We integrate each customer's check-in locations into his/her trajectory.
- **Pinterest**[7] [17] is an implicit feedback dataset for image recommendation [17]. Each user is associated with his/her own trajectory towards 9,855 different images.
- **AMZ-Book**[8] is the book review dataset between readers and book trajectories, organized from the book collection of Amazon-review [23]. We directly use the existing data split from [67], where each reader and book have at least ten interactions.

Evaluation Results. For public datasets, we proceed to the tasks of *customer segmentation* (CS) and *shopping trajectory completion* (TC) for illustration. We select language models (Transfomer and Graph Transformer), specialized recommender models (DIEN and SURGE), and deep learning models (DeepSet, Set Transformer, and PSWE) that show good performance in previous sections as competing methods. We have two major observations as follows:

- For the task of customer segmentation, due to their absence of ground-truth similar customers, we construct the similar customer data based on the number of overlapping trajectory elements and thus skip the pre-training phase. We split the data with 8:2 ratio for training and evaluation. As shown in Figure 1(a)-(d), C-STAR presents consistent performance superiority on these public datasets. Compared to other methods, this indicates the effectiveness of our proposed mechanism in capturing trajectory structural similarity.
- For the second task, similar to the previous one, we directly train the model with 80% of all trajectories without pre-training. From Figure 1(e)-(h), C-STAR performs competitively among all comparative models, except on Gowalla dataset where it is the second best. It reassures the effectiveness of our C-STAR framework for the task of shopping trajectory completion on public datasets.

[5]https://www.kaggle.com/datasets/ruchi798/bookcrossing-dataset
[6]https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/gowalla
[7]https://sites.google.com/site/xueatalphabeta/dataset-1/pinterest_iccv
[8]https://github.com/gusye1234/LightGCN-PyTorch/tree/master/data/amazon-book