Did You Enjoy the Ride: Understanding Passenger Experience via Heterogeneous Network Embedding

Carl Yang [#1], Chao Zhang [#2], Jiawei Han [#3], Xuewen Chen [*4], Jieping Ye [*5]

[#] *University of Illinois, Urbana Champaign*
201 N Goodwin Ave, Urbana, Illinois 61801, USA
[1] jiyang3@illinois.edu
[2] czhang82@illinois.edu
[3] hanj@cs.illinois.edu

[*] *Didichuxing Inc.*
8 W Dongbeiwang St, Beijing 100190, China
[4] chenxuewen@didichuxing.com
[5] yejieping@didichuxing.com

***Abstract*—Online taxicab platforms like DiDi and Uber have impacted hundreds of millions of users on their choices of traveling, but how do the users feel about these ride-sharing services, and how to improve their experience? While current ride-sharing services have collected massive travel data, it remains challenging to develop data-driven techniques for modeling and predicting user ride experience. In this work, we aim to accurately predict passenger satisfaction over their rides, and understand the key factors that lead to good/bad experiences. Based on in-depth analyses on large-scale travel data from a popular taxicab platform in China, we develop PHINE (Pattern-aware Heterogeneous Information Network Embedding) for data-driven user experience modeling. Our PHINE framework is novel in that it is composed of spatial-temporal node binding and grouping for addressing spatial-temporal data variation, and pattern preservation-based joint training for modeling the interactions among drivers, passengers, locations, and time. Extensive experiments on 12 real-world travel datasets demonstrate the effectiveness of PHINE over strong baseline methods. We have deployed PHINE in the DiDi Big Data Platform, delivering high-quality predictions for passenger satisfaction on a daily basis.**

I. Introduction

Nowadays, the way of traveling for hundreds of millions of users has been reshaped by online taxicab platforms like DiDi[1] and Uber[2][1]. Take DiDi for example, in the year of 2017, it generates over 20 million orders every day, effectively employing 20 million drivers and allocating 10% of all vehicles in China to serve over 400 million passengers, influencing about 30% of China's total population. Taxi-sharing platforms have collected an unprecedentedly large amount of travel data. Such data are not only invaluable for transportation science, but also critical to various socioeconomic tasks, such as smart city planning [2], [3], traffic congestion resolving [4], air pollution control [5], and driverless car developing [6].

However, the current research community have not paid sufficient attention to mining such travel data, and the power of such data has not been fully leveraged. How much do people enjoy traveling with the new taxi hailing platforms, and what can be done to further improve their travel experience? In this work, we aim to model drivers, passengers, locations, times and their complex interactions to accurately predict passenger experience towards their rides and understand the important factors that lead to good/bad experiences.

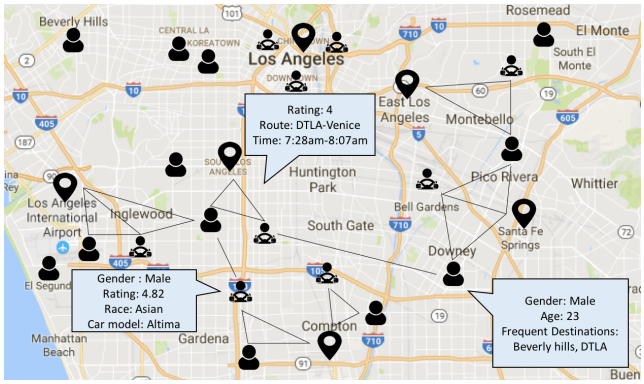


Fig. 1. An illustration of the rich travel data represented as an HIN.

Taxi-sharing data are typically structured records. Table I showcases three simplified anonymous records from a typical travel database, which include the pick-up and drop-off locations of three rides, as well as multi-dimensional attributes for the drivers and passengers. Given any one such record, we want to automatically predict what ratings the involved passengers will give. Furthermore, we aim to evaluate what components make a ride satisfactory (or not), based on which we can adjust ride dispatch strategies to improve future use experience.

The user experience modeling problem for taxi-sharing data is not a trivial problem. There are two major challenges that we need to address:

Challenge 1: Alleviating Data Sparsity. While there are large amounts of travel data, considering each entity (*e.g.*, location, driver, passenger) as independent could lead to severe data sparsity, because many entities are involved in a limited

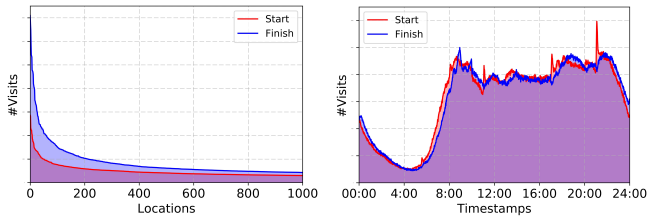
[1]https://www.xiaojukeji.com/
[2]https://www.uber.com/

driver			passenger			location		time	
avg_star	*orders*	*car_level*	*avg_star*	*cancels*	*complaints*	*start*	*finish*	*start*	*finish*
4.89	620	500	5	3	0	116.3104, 39.9258	116.3105, 39.9392	2017-06-01 17:45:38	2017-06-01 17:57:03
5	24	700	4.43	2	1	116.0799, 39.7122	116.1603, 39.7567	2017-06-01 11:15:25	2017-06-01 11:40:22
4.62	5856	500	0	0	0	116.4552, 39.9338	116.3976, 39.9699	2017-06-01 21:22:51	2017-06-01 21:41:38

TABLE I
THREE TYPICAL ANONYMOUS TRAVEL RECORDS INCLUDING MULTI-DIMENSIONAL INFORMATION OF SPECIFIC RIDES.

number of rides. For example, Figure 2 shows the frequencies of locations and timestamps recorded at the starting points and finishing points of rides. The frequency distributions of visited locations clearly suffers from the long-tail problem, and the frequency of visits clearly changes along time. Therefore, to predict the quality of rides, we should consider not only the involved entities, but also the semantic similarities among them. *E.g.*, nearby locations in geographical distance may share similar traffic conditions at particular times, and passengers with similar personalities might evaluate situations in similar styles. Moreover, while user experience in terms of explicit ratings reflects how satisfactory the user is, the rating data are quite sparse – it is observed that no more than 20% rides receive explicit ratings from passengers.

Challenge 2: Integrating Heterogeneous Knowledge. The interactions among drivers, passengers, locations, and time spots are highly complex and heterogeneous. It is challenging to jointly consider such heterogeneous interactions for user experience prediction. Furthermore, we normally have prior knowledge about what entities should be similar (*e.g.*, nearby locations, passengers sharing similar trajectories, *etc.*). It is important yet nontrivial to incorporate such prior knowledge into the modeling process.

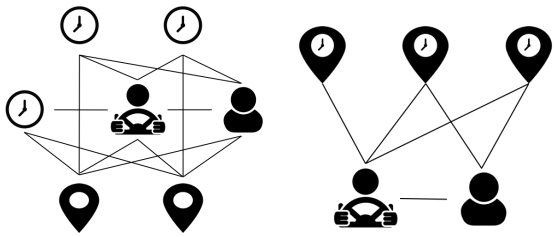


(a) #Visits vs. Locations (b) #Visits vs. Timestamps

Fig. 2. **Motivating spatiotemporal analysis of rides.**

Approach: PHINE (*Pattern-aware Heterogeneous Information Network Embedding*). To address the above challenges, we propose PHINE, which is a neural framework for user experience prediction. PHINE uses heterogeneous information networks (HINs) to encode rich context information and performs pattern-aware embedding to deliver accurate passenger satisfaction prediction and insights into good/bad rides. As the travel data are highly structured, it is natural to use HIN [7] as the data model for such data. For example, given the records in Table I, Figure 1 illustrates an HIN that encodes the entities and the interactions among them.

Now given a ride represented by a subset in a rich HIN as illustrated in Figure 3 (a), we want to automatically predict what ratings the involved passengers will give. For this task, we adopt a neural network embedding framework, which has been shown advantageous in modeling structured data for various tasks [8], [9]. However, to address the aforementioned challenges, the network embedding process in PHINE has two unique characteristics.



(a) HIN with location and times (b) Spatial-temporal binding

Fig. 3. **Rides modeled as subsets of a transportation HIN.**

First, *spatial-temporal spot binding and grouping.* A ride can be naturally modeled as an HIN such as the one illustrated in Figure 3 (a). However, locations usually have changing functions at different times and thus different visitor behaviors and traffic conditions. For example, a busy shopping area during the daytime might become a quiet block with few visitors at night, and a cozy bar during the afternoon may turn into a center of nightlife activities after dark. Therefore, instead of modeling location and time as separate node types, we *bind location and time into spatiotemporal nodes* like the one in Figure 3 (b). Such a binding can effectively model the dynamics and traffic conditions of the same location at different times. Moreover, to address the spatiotemporal sparsity and variation, we further develop a grid-based grouping method, such that the spatiotemporal units that are close to each other are grouped together, and rich data around typical spots can be leveraged to profile the properties of nearby ones.

Second, *pattern preservation-based joint embedding.* Recent successful network representation learning methods usually attempt to capture the interactions among nodes through path sampling [8], [11], [12], [13] or neighborhood preserving [9], [14], [15], [16]. However, existing network embedding techniques are not designed to leverage any particular types of interactions. Specifically, sampling random paths or preserving local neighborhoods do not necessarily lead to meaningful objectives in an HIN like our transportation network shown in Figure 1, where the interactions actually follow regular semantic patterns and we have useful prior knowledge about which ones might be useful. To accurately predict passenger experience, we design an end-to-end joint training neural framework composed of two objectives – one is a supervised

loss for passenger rating prediction, and the other is an unsupervised loss for semantic pattern preservation. We further develop a pattern sampling process based on the idea of path sampling [8], which effectively utilizes available rating data while naturally incorporating various prior knowledge about useful HIN semantic patterns.

The main contributions of this work are summarized as follows:

1) We conduct various data analysis on real travel data to provide insights into the passenger experience prediction problem and motivate proper modeling techniques.
2) We develop a novel HIN embedding algorithm, which utilizes spatial-temporal spot binding and grouping for alleviating data sparsity, and meanwhile effectively leverages prior knowledge about useful semantic patterns.
3) We deploy the framework in DiDi Big Data Center and perform extensive analyses on 12 travel datasets to deliver high-quality passenger experience predictions with interpretable insights.

The rest of this paper is organized as follows. In Section 2, we introduce some preliminaries about the problem. Section 3 will present our PHINE neural framework in details. We will cover our extensive experimental results on real travel datasets in Section 4. Related works are discussed in Section 5 and a quick summery is provided in Section 6.

II. Preliminaries

In this section, we introduce some key concepts in the context of our framework, followed by a formal description of our passenger experience learning problem.

A. Key Concept Definition

1) Heterogeneous Information Networks: Due to advanced data warehousing techniques [17] and the much effort put into knowledge acquisition [18], [19], lots of data nowadays are stored in structures. They are usually of multiple types, interconnected, forming complex, heterogeneous information networks [7], [20]. An information network represents an abstract of the real world, focusing on the entities and the interactions among them. Formally, information networks and HIN are defined as follows.

Definition II.1. Information Network [7]. An information network is defined as a directed graph $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ with an entity type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$. Each entity $v \in \mathcal{V}$ belongs to one particular entity type in the entity type set $\mathcal{A} : \phi(v) \in \mathcal{A}$, and each link $e \in \mathcal{E}$ belongs to a particular relation type in the relation type set $\mathcal{R} : \psi(e) \in \mathcal{R}$. If two links belong to the same relation type, the two links share the same starting entity type as well as the ending entity type.

Definition II.2. Heterogeneous Information Network (HIN) [7]. The information network is called an HIN if the types of entities $|\mathcal{A}| > 1$ or the types of relations $|\mathcal{R}| > 1$.

Our travel data naturally reside in an HIN $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{A} = \{\mathcal{D}, \mathcal{P}, \mathcal{L}, \mathcal{T}\}$, corresponding to entity types of driver, passenger, location and timestamp, respectively.

2) Spatial and Temporal Hotspot: Unlike drivers and passengers that are natural basic units for embedding, the space and time are continuous and it is infeasible to embed every location and timestamp. To address the spatiotemporal continuities, variations and data sparsity, [10] proposes to group locations and timestamps separately based on spatial and temporal hotspot detection. They define hotspot based on kernel density maximization as follows.

Definition II.3. Spatial and Temporal Hotspot [10]. Given a transportation HIN $\mathcal{N}$, let $\mathcal{L}(\mathcal{V})$ be the collection of locations in $\mathcal{N}$, a spatial hotspot is a local maximum of the kernel density function estimated from $\mathcal{L}(\mathcal{V})$. Similarly, let $\mathcal{T}(\mathcal{V})$ be the collection of timestamps in $\mathcal{N}$, a temporal hotspot is a local maximum of the kernel density in $\mathcal{T}(\mathcal{V})$.

As we want to carefully capture the different visitor distributions and traffic conditions of locations during different times of the day, we apply novel spatial-temporal spot binding. Moreover, since we need to consider both bursting locations/times as well as vacant or remote ones, we apply grid-based grouping instead of hotspot-based grouping.

3) Semantic Pattern: Unlike homogeneous networks where entities and links are of the same type, links in an HIN may carry rather different semantic meanings across entities of different types. Therefore, embedding entity proximities based on sampling random paths [8] or preserving local neighborhoods [9] is often meaningless. In this work, we propose to capture the semantic proximities among entities based on sampling semantic patterns, which are defined as follows.

Definition II.4. Semantic Pattern. Given an HIN $\mathcal{N}$ with the object type mapping $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and the link mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$, a semantic pattern, denoted as $m = \{\mathcal{A}_m, \mathcal{R}_m\}$, is a directed graph defined over a subset of object types $\mathcal{A}_m \subset \mathcal{A}$, with edges as relations from $\mathcal{R}_m \subset \mathcal{R}$. ω should be formed *w.r.t.* prior knowledge about certain semantic proximities among entities of types in $\mathcal{A}_m$.

Note that the definition of semantic patterns is different from meta-path or meta-graph as in [7], whose exact shapes and sizes are fixed. Semantic patterns defines a way to sample or grow subgraph instances. Instead of acting as a distance measure, it captures semantic proximities among all entities on the particular sampled subgraphs. We will introduce some intuitive semantic patterns composed over our transportation HIN in Sec III.4 and study the impact of different semantic patterns for passenger experience prediction in Sec IV.2.

B. Problem Description

In this work, we aim at understanding passenger experience via predicting and gaining insight into the star-level ratings passengers make about their rides. While there are many metrics that can be used to describe user experience (*e.g.*,

NPS [21], satisfaction [22], *etc.*), they are often hard to collect through surveys and suffer from biased sampling because only highly motivated users tend to take part in the surveys. In our scenario, we have the first-hand information of passenger satisfaction, *i.e.* star-level ratings, of which the data are enormous and less biased. The problem is also important in production, because we observe that less than 20% rides receive star-level ratings from passengers, and it is desirable that the unrated rides can get properly understood with unhappy experiences timely detected and remedied.

Input. The basic input is the set of rides Υ. To describe a ride $r \in \Upsilon$, we should consider the involved driver d ($\phi(d) = \mathcal{D}$), passengers p ($\phi(p) = \mathcal{P}$), locations l ($\phi(l) = \mathcal{L}$) and timestamps t ($\phi(t) = \mathcal{T}$) . We denote it as $w = \{\vec{d}, \vec{p}, \vec{l}, \vec{t}, y\}$, where $\vec{d}$ usually only includes a single driver that operates the vehicle, $\vec{p}$ often includes a single passenger but can be multiple for shared rides, $\vec{l}$ and $\vec{t}$ are lists of visited and recorded locations and timestamps along the ride, and y is a ground-truth 5-level star rating, which is available in the training set and to be predicted in the testing one.

However, as illustrated in Table I, any of these entities alone does not have many relevant features. Moreover, their historical interactions are too complex to be modeled by classic recommendation algorithms like matrix factorization [23], [24]. Since the rich travel data we have are highly structured and entities interact in certain ways, we model them in a transportation HIN $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$, take it as additional input, and leverage $\mathcal{E}$, the set of heterogeneous edges that models the interactions among different entities, through novel pattern-aware HIN embedding.

Output. The basic output of our framework is the predicted star-level rating y for each ride r. To provide deeper insight into the rides and key factors of different travel experiences, we also output the embeddings $\mathbf{x}(\cdot)$ for every entity in $\mathcal{N}$. Ideally, these embeddings should capture the inherent relationships among different entities, such as the geographical distances among locations, the temporal distances among timestamps, the similarity in preferences and personalities among drivers and passengers, *etc.*

Objective. We design two types of objectives to 1) preserve passenger ratings over their rides in $\mathcal{W}$ and 2) capture the semantic proximities among different entities in $\mathcal{N}$. Therefore, we have the overall loss function

$$\mathcal{J} = \mathcal{J}_1 + \lambda \mathcal{J}_2, \tag{1}$$

where $\mathcal{J}_1$ is a supervised loss on passenger ratings and $\mathcal{J}_2$ is an unsupervised loss on entity similarities. λ controls the trade-off between the two objectives.

III. PHINE

A. Framework Overview

Figure 4 illustrates the overall neural embedding framework of PHINE. The goal of PHINE is to simultaneously learn the embeddings of all involved entities in rides (*i.e.*, drivers, passengers, locations, times) to predict passenger experience in terms of explicit ratings. It captures the historical interactions and semantic proximities among different entities through jointly preserving passengers' ratings over past rides and entities' contexts based on semantic patterns.

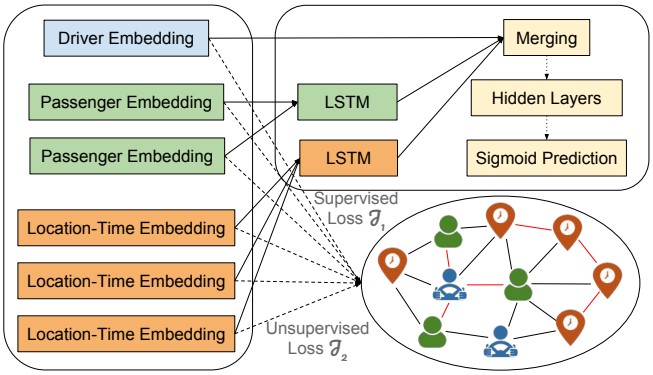


Fig. 4. The neural embedding framework of PHINE.

As discussed in Sec II, the input of PHINE is a set of rides $\mathcal{W}$ and a transportation HIN $\mathcal{N}$. To consider the original features describing a ride, we preprocess driver features (*e.g.*, age, sex, average rating, number of finished orders), passenger features (*e.g.*, average rating, number of finished orders, number of complaints, number of canceling) and spatiotemporal spot features (*e.g.*, longitude, latitude, timestamp) via binning (for numerical features) and one-hot encoding (for categorical features). After preprocessing, we insert a fully connected feedforward neural network to explore the high-dimensional sparse entity features and produce a low-dimensional dense embedding vector for each entity, *i.e.*, $\mathbf{x}(\mathbf{d})$ for a driver, $\mathbf{x}(\mathbf{p})$ for a passenger, and $\mathbf{x}(\mathbf{s})$ for a spatiotemporal spot. These embeddings are further constrained by a supervised loss $\mathcal{J}_1$ to recover ground-truth ratings as well as an unsupervised loss $\mathcal{J}_2$ to preserve semantic similarities among entities. The two losses are iteratively optimized in a joint training framework that runs efficiently in parallel on CPUs or GPUs.

B. Heterogeneous Network Construction

We construct a novel transportation HIN to encode the multi-type multi-dimensional proximities among different entities extracted from travel data. In the HIN, there are originally four types of entities: drivers, passengers, locations and timestamps. To account for the changing visitor behaviors and traffic conditions of locations during different times, and address the challenge of sparse variational spatiotemporal data, we further apply spatial-temporal spot binding and grid-based spot grouping before actually constructing the heterogeneous graph.

1) Spatial-Temporal Spot Binding: Existing HIN representation learning methods like [10] embed locations and timestamps separately. They then preserve the location-time co-occurrences and the neighborhood structures in the spatial

and temporal spaces. In this way, locations close in spatial distances and visited during similar times will be embedded as close. However, we stress that locations usually have changing functions at different times and thus different visitor behaviors and traffic conditions. *E.g.*, a busy shopping area during the daytime might become a quiet block with few visitors at night, but a cozy bar during the afternoon may turn into a center of nightlife activities after dark. Therefore, computing a single embedding for each location is insufficient to reflect such dynamics. Rather, it is intuitive to allow each location to have multiple embeddings *w.r.t.* different times of the day, while requiring these embeddings to be close in certain ways to reflect the location identities and spatial closeness.

Therefore, we propose a novel spatial-temporal spot binding process, to combine locations with timestamps at which they receive considerable visits. Specifically, for each location l visited at timestamp t, we replace l and t with a single spatiotemporal spot $s = \{l, t\}$. To capture the spatial-temporal proximities among spots that belong to nearby locations or have similar timestamps, we later require them to share similar (but not the same) embeddings.

Horeover, this process, while effectively differentiates locations with multiple functions at different times, indeed introduces a large amount of additional nodes. We deal with this overhead through a novel grid-based spot grouping.

2) Grid-Based Spot Grouping: Although we have rich travel history data, visitings around most individual spatiotemporal spot are sparse, as we can see from Figure 2. Besides, spatiotemporal data are also variational, because visits around the the same location for the same event might well be recorded with slightly different longitudes, latitudes and timestamps. Moreover, it is infeasible to compute an embedding for every spatiotemporal spot.

To address the issues of sparsity, variation and scalability, we design a novel grid-based spot grouping technique, to group spatiotemporal spots belonging to nearby locations with similar timestamps. Specifically, we design a 3-dimensional grid with tunable granularities of $[\alpha \times \beta \times \gamma]$. *E.g.*, we can group spatiotemporal spots based on $[0.001 \times 0.001 \times 10]$ grids, where the first two dimensions, α and β, are longitude and latitude in degree and the third, and γ is timestamp in minute. 0.001 degree in longitude and latitude is about 110 meters and time of day can be divided into $\frac{24 \times 60}{\gamma}$ bins. Therefore, such a gridding schema effectively groups the spatiotemporal spots in the core area of a gigantic city like Beijing into about 10 billion bins. This amount of bins, while manageable by current systems, is inefficient in computation and heavy in memory.

To deal with the large amount of bins, we further design a grid merging technique, based on the observation that most grids cover few and vacant spots if any, such as those within estates, mountains, lakes and other traffic-light areas. Specifically, for each grid cell, we recursively merge it with its nearby cells if the merged cell receives fewer than θ visits each day on average. θ is also tunable, while empirically setting it to 100 results in about 62 million bins, which significantly reduces the computational cost and further addresses data sparsity and variation.

Note that, our grid-based spot grouping technique is novel and different from existing urban modeling techniques like [10], because it is designed to consider both popular and vacant locations and times, which is crucial for the particular task of traffic modeling.

3) Heterogeneous Graph Construction: After binding and grouping the spatiotemporal spots, we use three types of nodes: driver, passenger and spatiotemporal spot in our transportation HIN, where each spatiotemporal node represents a merged grid cell in the spatiotemporal space. To construct the edges within our transportation HIN, we consider co-occurrence and neighborhood relationships similarly as in [10]. First, as each travel record contains driver, passenger and spot nodes, the co-occurrence relationship induces three types of edges: (1) *driver-passenger* edge; (2) *driver-spot* edges; (3) *passenger-spot* edge. Within each edge type, we set the edge weight to the normalized co-occurrence count. Second, the neighborhood relationship in spatial and temporal spaces induces one more type of edges: (4) *spot-spot* edge. For any spatiotemporal spot, we connect it with its spatial and temporal neighbors (each grid cell has 8 neighbor cells in the 3-dimensional spatiotemporal space before merging and two merged cells are connected if any pair of their member cells are connected). We set all weights of this type of edges to 1.

C. Pattern-Aware Embedding

To capture the semantic proximities among the three types of nodes in our transportation HIN, instead of preserving long random paths or complete local neighborhoods, we propose to leverage intuitive semantic patterns that match with prior knowledge about the properties of the particular HIN.

1) Semantic Pattern Definition: According to its definition in Sec II, HIN always contains multiple types of entities and their various relations. Therefore, it is almost impossible to enumerate every possible patterns on an HIN, and most of the arbitrarily enumerated patterns are meaningless. On the other hand, as we construct an HIN, we often have some prior knowledge over the contained entities and relations. Specifically, according to our prior knowledge about the transportation HIN, we intuitively define the semantic proximities and their corresponding semantic patterns as follows.

Definition III.1. (a) Spatial Proximity. Spots should be embedded as close if the underlying locations are close in the spatial space.

Definition III.2. (b) Temporal Proximity. Spots should be embedded as close if the underlying timestamps are close in the temporal space.

Definition III.3. (c) Trajectory Proximity. Spots should be embedded as close if they lie close on the same passenger trajectories, because passengers often take taxies to spots similar in certain ways (*e.g.*, restaurants and bars at night, *etc.*).

Definition III.4. (d) Preference Proximity. Drivers and passengers should be embedded as close if they have been to the same spots, because they have similar spatiotemporal travel preferences.

Definition III.5. (e) Compatibility Proximity. Drivers and passengers should be embedded as close if they have been in the same ride, because their travel objectives are compatible.

Figure 5 illustrates the five semantic patterns we derive for preserving proximities among the three types of entities in our transportation HIN. While other patterns might also be useful, these are the most significant based on our prior knowledge and empirical study on the travel data.

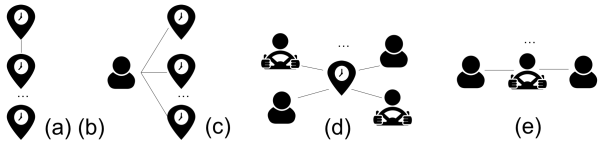


Fig. 5. Intuitive semantic patterns on our travel HIN.

2) Pattern Sampling Algorithm: Algorithm 1 formally describes the distributed pattern sampling algorithm we design to efficiently generate context data for the pattern preserving objective. The algorithm takes the input of the HIN $\mathcal{N}$, the number of patterns to be sampled T, the entities to be sampled on each pattern L, and a multinomial distribution to sample different numbers of instances for the five semantic patterns. The output $\mathcal{M}$ is a set of instances Ω, where each instance is a set of sampled entities.

Algorithm 1 PHINE Pattern Sampling

procedure PHINESAMPLE
  Input: $\mathcal{N}, [\mu_a, \mu_b, \mu_c, \mu_d, \mu_e], T, L$
  Output: $\mathcal{M}$
  $\mathcal{M} \leftarrow \emptyset$
  for $t \leftarrow 1$ to T **do**
    $\Omega \leftarrow \emptyset$
    Sample a pattern Λ *w.r.t.* $[\mu_a, \mu_b, \mu_c, \mu_d, \mu_e]$
    repeat
      Sample a node $v_1 \in \mathcal{V}$ *w.r.t.* the uniform dist
    until $\phi(v_1) = \mathcal{S}$ for pattern (a)(b)(d), $\mathcal{P}$ or $\mathcal{D}$ for pattern (c) and $\mathcal{P}$ for pattern (e)
    $\Omega \leftarrow \Omega + v_1$
    repeat
      Sample a node $v_c \in \Omega$ *w.r.t.* the uniform dist
      Sample a neighbor v_i of v_c *w.r.t.* the weights of the edges selected by Λ and add v_i to Ω
    until $|\Omega| = L$
    $\mathcal{M} \leftarrow \mathcal{M} + \Omega$
  end for
end procedure

3) Pattern Preserving Objective: Based on the sampled instances of semantic patterns, we design a pattern preserving objective to capture the semantic proximities among different types of entities in the HIN. To leverage the classic Skipgram model for word embedding [25], within each pattern instance Ω, we randomly sample pairs of entities into entity-context pairs. Following the derivations in [8], we have

$$\begin{aligned}\mathcal{J}_2 &= -\sum_{(v_i,v_c)} \log p(v_c|v_i) \\ &= -\sum_{(v_i,v_c)} \log(\mathbf{w}_c^T \mathbf{u}_i - \log \sum_{u'_c \in \mathcal{C}} \exp(\mathbf{w}_{c'}^T \mathbf{u}_i)), \end{aligned} \quad (2)$$

where $\mathbf{u}_i = \mathbf{x}(v_i)$ is the embedding of entity v_i, $\mathbf{W}$ include the parameters in the pattern preservation module, and $\mathcal{C}$ is the set of all entity contexts.

To efficiently optimize this objective, we follow the popular negative sampling approach [25] to approximate the intractable normalization over the whole context space $\mathcal{C}$. Specifically, with probability ρ, we sample a positive pair of $(v_i, v_c, \pi = 1)$ from Ω, and with probability $1 - \rho$, we uniformly corrupt the context v_c to sample a negative pair $(v_i, v_c, \pi = -1)$. Therefore, for each sampled entity pair (v_i, v_c, π), we minimize the cross entropy loss of classifying (v_i, v_c) to the binary label π:

$$\begin{aligned}\mathcal{J}_2 &= -\mathbb{I}(\pi = 1) \log \sigma(\mathbf{w_c}^T \mathbf{u}_i) - \mathbb{I}(\pi = -1) \log \sigma(-\mathbf{w}_c^T \mathbf{u}_i) \\ &= -\mathbb{E}_{(v_i,v_c,\pi)} \log \sigma(\pi \mathbf{w}_c^T \mathbf{u}_i), \end{aligned} \quad (3)$$

where σ is the sigmoid function defined as $\sigma(x) = 1/(1 + e^{-x})$, $\mathcal{I}(\cdot)$ is an indicator function and the expectation is taken *w.r.t.* the distribution of $p(v_i, v_c, \pi)$, which is conditioned on the HIN and encodes the distributional information in the semantic patterns.

D. Joint Training

As we aim to capture both passenger past experience and the semantic proximities among drivers, passengers and spots, we design a joint training framework to simultaneously optimize the two losses in Eq. 1.

1) End-to-end Supervision: The entity embeddings should reflect passengers' ratings over existing rides. To achieve this, we design the embedding framework with end-to-end rating supervision. As illustrated in Figure 4, since each ride w can involve multiple passengers (for shared rides) and multiple spatiotemporal spots (along the ride), after the individual embedding of entities, we further connect two LSTM layers to the lists of passenger embeddings $\mathbf{x}(\mathbf{p})$ and spot embeddings $\mathbf{x}(\mathbf{s})$, respectively. The LSTM cells model the sequential information among passengers and spots in a ride, such as distances among locations, travel times among timestamps, differences in passengers' personal preferences, *etc.* So we have

$$\mathbf{g}(\vec{p}) = \mathbf{g}(\mathbf{x}(\vec{p})) = \mathbf{g}(\mathbf{x}(\mathbf{p}_1), \ldots, \mathbf{x}(\mathbf{p}_m)), \quad (4)$$

$$\mathbf{g}(\vec{s}) = \mathbf{g}(\mathbf{x}(\vec{s})) = \mathbf{g}(\mathbf{x}(\mathbf{s}_1), \ldots, \mathbf{x}(\mathbf{s}_k)). \quad (5)$$

Then we collect the driver embedding $\mathbf{x}(d)$, passenger list embedding $\mathbf{g}(\vec{p})$ and spot list embedding $\mathbf{g}(\vec{s})$ through vector concatenation $\mathbf{x}(r) = [\mathbf{x}(d), \mathbf{g}(\vec{p}), \mathbf{g}(\vec{s})]$ and connect multiple (Q) layers of nonlinear feedforward neural networks to fully

explore the interactions among the three types of entities in a ride. So we have

$$\mathbf{h}(r) = \mathbf{h}^Q(\dots \mathbf{h}^1(\mathbf{h}^0(\mathbf{x}(r)))\dots), \tag{6}$$

where $\mathbf{h}^q(r) = ReLU(\mathbf{W_h}^q \mathbf{h}^{q-1}(r) + \mathbf{b_h}^q)$ and $\mathbf{h}^0(r) = \mathbf{x}(r)$. The predicted rating is produced by adding a sigmoid regression layer at the end of ride embedding as $\hat{y}(r) = 5\sigma(\mathbf{h}(r))$, because ratings are continuous values in the range of $[0, 5]$. Finally, we use MSE as the supervised loss and get

$$\mathcal{J}_1 = \frac{1}{|\Upsilon|} \sum_r |\hat{y}(r) - y(r)|^2. \tag{7}$$

2) Joint Training Algorithm: Algorithm 2 describes the joint training algorithm we design to simultaneously learn all parameters $\Phi = \{\Phi_e, \Phi_l, \Phi_h, \Phi_c\}$ in the PHINE model. $\Phi_e, \Phi_l, \Phi_h, \Phi_c$ are the parameters in the entity embedding layers, LSTM cells, rating prediction layers and pattern preservation layers, respectively. The inputs T_1 and T_2 are number of iterations used to approximate the weighting factor λ and B is the batch size. We implement it with PyTorch by performing SGD with mini-batch Adam [26]. We deploy the algorithm in DiDi Big Data Center to deliver high-quality passenger experience predictions and perform extensive performance comparison experiments.

Algorithm 2 PHINE Joint Training

procedure PHINETRAIN
  Input: T_1, T_2, B
  repeat
    for $t \leftarrow 1$ to T_1 **do**
      Sample a batch of $r = (d, \vec{p}, \vec{s}, y)$ of size B
      $\mathcal{J}_1 = \frac{1}{B} \sum_{r \in \mathcal{B}_1} |\hat{y}(r) - y(r)|^2$
      Take a gradient step to optimize Φ_e, Φ_l, Φ_h
    end for
    for $t \leftarrow 1$ to T_2 **do**
      Sample a batch of $p = (v_i, v_c, \pi)$ of size B
      $\mathcal{J}_2 = \frac{1}{B} \sum_{(v_i, v_c, \pi \in \mathcal{B}_2} \log \sigma(\pi \mathbf{w}_c^T \mathbf{u}_i)$
      Take a gradient step to optimize Φ_e, Φ_c
    end for
  until $\mathcal{J}$ converges or is sufficiently small
end procedure

IV. EXPERIMENTS

In this section, we evaluate PHINE for passenger experience learning with extensive experiments on multiple real-world travel datasets.

A. Experimental Setup

1) Datasets: We process and use 12 travel datasets of travel orders in different cities and time ranges. The data are generated on a popular online taxicab platform in China. The basic statistics of the datasets are shown in Table II.

The 12 datasets include orders generated in three cities of different scales in China, *i.e.*, Beijing, Chengdu and Shenzhen. According to available public statistics, the populations of the three cities are about 20 million, 15 million and 10 million, respectively, by the end of 2014. For each city, we randomly sampled a certain percent of all orders generated in one day, one week and one month, to generate the three datasets with the *random_* prefix. To comprehensively evaluate the algorithm performance under difference scenarios, we further generate three groups of datasets, by filtering out orders generated around less active drivers (*driver_*), less active passengers (*passenger_*), and less popular spots (*spot_*). Each group of datasets also have different scales (*day, week* and *month*). Note that we intentionally do not report the exact sampling ratio and filtering thresholds due to confidential reasons, and therefore the reported figures here do not indicate real business scales and distributions.

Dataset	#orders	#driver	#passenger	#spots
random_day	1,178,478	79,486	712,210	253,371
random_week	7,762,695	129,713	2,634,123	640,206
random_month	39,252,188	184,221	6,286,167	1,319,718
driver_day	962,688	49,844	612,128	187,228
driver_week	6,680,968	88,805	2,346,616	485,584
driver_month	29,536,359	130,671	5,237,177	955,376
passenger_day	823,707	48,738	403,727	158,149
passenger_week	5,750,024	76,482	1,721,320	434,964
passenger_month	27,528,423	103,812	4,250,002	918,252
spot_day	889,515	53,758	503,234	106,429
spot_week	5,825,985	103,650	1,865,124	289,007
spot_month	20,596,488	152,626	5,375,703	606,278

TABLE II
THE STATISTICS OF THE SAMPLED DATASETS. NOTE THAT THE DATASETS ARE ALL SUBSETS RANDOMLY SAMPLED FROM REAL TRANSACTION DATA. WE DO NOT REPORT THE EXACT SAMPLING AND FILTERING PROCESS DUE TO CONFIDENTIAL REASONS, AND THEREFORE THE REPORTED FIGURES HERE DO NOT INDICATE REAL BUSINESS SCALES.

Drivers, passengers and spots are anonymous. Driver features are 52-dimensional, including gender, age, type, car level, finished order count, late order count and so on. Passenger features are 51-dimensional, including gender, finished order count, canceled order count, complaint count and *etc.* Spots are associated with only three attributes, *i.e.*, longitude, latitude and timestamp.

We partition each dataset into training set and test set. For each user, we use the earliest 80% orders as the training data, and the most recent 20% orders as the test data.

2) Evaluation Metrics: To quantitatively evaluate the performances of PHINE and compare with baseline methods, we compute the MSE (mean squared error) averaged over all testing samples between the predicted ratings and ground-truth ratings. Since more than 80% orders receive 5 stars and ratings are highly unbalanced, we also compute MSE on all testing samples that do not receive 5 star ratings, to highlight the algorithms' ability to capture unsatisfactory rides.

3) Compared Algorithms: We compare the following algorithms to comprehensively study the impacts of different learning and embedding schemas on our passenger experience prediction problem with the rich travel data.

- **NOBIND**: it does not bind and group spatiotemporal

spots but uses the raw longitude, latitude and timestamps as features and take every spot as a distinct entity. The travel HIN cannot be constructed due to vast amounts of spots.

- **NONET**: it only explores entity features to predict ratings through a feedforward neural network, without constructing a travel HIN to model the interactions among entities (drivers, passengers and spots).
- **PATH** [12]: besides modeling entity features, it leverages the travel HIN by jointly preserving ratings and entity contexts based on path sampling [8].
- **NEIGHBOR** [14]: besides modeling entity features, it leverages the travel HIN by jointly preserving ratings the first and second order proximities among entities [9].
- **PHINE**: besides modeling entity features, it leverages the travel HIN by jointly preserving ratings and entity contexts based on sampled instances of some intuitively composed patterns.

4) Parameter Settings: When comparing the algorithms, we set the parameters of PHINE to the following default values: for the loss function, we set the weighting factors $\lambda = 0.1$; for the hidden layers under $\mathcal{J}_1$, we set the number of layers to 3 and the sizes of the layers to $64 \rightarrow 32 \rightarrow 16$; for the context sampling process under $\mathcal{J}_2$, we set the pattern size to 8, the number of entity context pairs on each pattern to 10, and the sampling ratio of five patterns to $[1, 1, 1, 1, 1]$; for the entity embedding layer, we set the embedding size to 64; for the joint training process, we sample 5 negative samples for each positive label, and we set the batch size to 10,000 and the learning rate to 0.001. In addition to these default values, we also evaluate the effects of different parameters on the performance of PHINE in Sec. IV.B.

To ensure fair comparison, we use the same set of features and the same neural embedding architecture for all five compared algorithms. For NEIGHBOR and PATH, we use the same HIN as for PHINE. For NOBIND, NONET and NEIGHBOR, the size of entity embedding, number of feedforward network layers and sizes of feedforward networks are set as the same with PHINE, and no more parameters remain to be set; for PATH, we further set the length of sampled paths to 8 and the number of entity context pairs on each path to 10.

B. Experimental Results

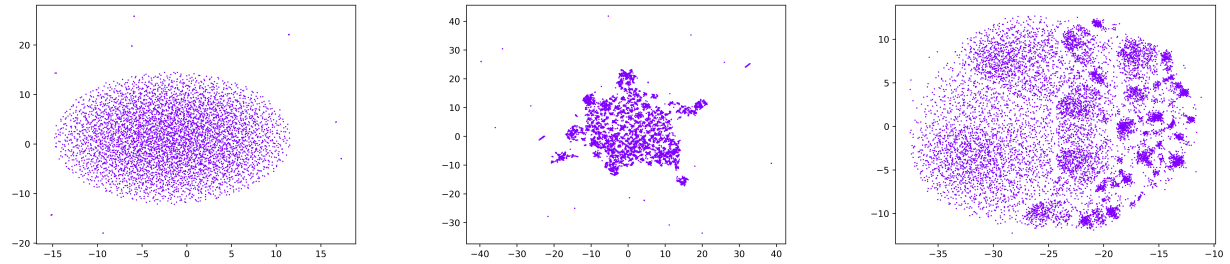
1) Embedding Visualization: A good network embedding should be able to capture the intrinsic distribution and functional roles of entities, so as to properly lay them out in a low dimensional space. To empirically analyze the ability of different HIN embedding schemas for our passenger experience learning objective, we visualize the embedding of driver and passenger networks on part of our random_day dataset. Laying out these subnetworks is very challenging as the algorithm needs to capture the intrinsic features and interactions in the rich HIN that lead to different ratings. We first compute the low dimensional embeddings of drivers and passengers in three joint training frameworks with different context preserving schemas, and then map the entity embeddings into a 2-dimensional space with the LargeVis package [27], which is more efficient than traditional t-SNE [28].

Figure 6 compares the visualization results generated by the different embedding algorithms, *i.e.*, PATH, NEIGHBOR and PHINE. Entities in Figure 6 (a)(b)(d)(e) (generated by PATH and NEIGHBOR) are uniformly distributed or highly cluttered together in the embedding space. This is because they consider general distance on the network regardless of entity types, and random paths or complete neighborhoods usually do not effectively capture meaningful semantic patterns in HINs. On the other hand, PHINE performs quite well and generates meaningful layouts of the subnetworks, as entities in Figure 6 (c)(f) clearly reside on certain separable manifold structures, probably characterizing users' multi-dimensional behavioral conventions. Such embeddings reflect the intrinsic geometric distribution of drivers and passengers in the latent space, which is captured by PHINE under the help of semantically meaningful HIN patterns.

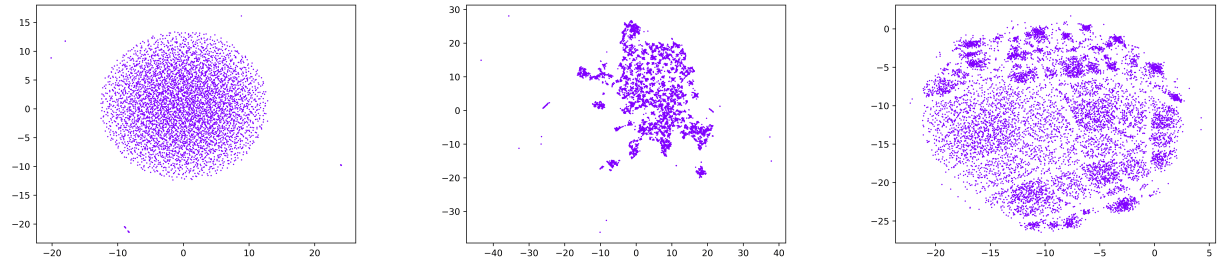
2) Quantitative Evaluation: We quantitatively evaluate PHINE against the baselines on the passenger experience prediction task. Table III shows the performance in terms of MSE and runtime on the 12 travel datasets. PHINE constantly outperforms the strongest baseline by a large margin of around 25% relative performance. The improvements of PHINE over the compared algorithms all passed the paired t-tests with significance value $p < 0.001$. Such consistent improvements clearly demonstrate the advantage of PHINE.

Taking a closer look at the scores of different algorithms, we observe that 1) NOBIND generally has the worst performance, indicating that directly taking the features of entities involved in the rides does not lead to satisfactory predictions. It is worthwhile to develop more complicated, intuitive and powerful models to leverage the interactions and distributions of the entities. 2) NONET simply applies our specifically designed spot binding and grouping method, and achieves slightly better performance than NOBIND on the *random, driver, passenger* datasets. It indicates that binding and grouping spatiotemporal spots are generally helpful, especially when some spatiotemporal spots receive sparse visits. This method also discretize and reduces the number of spatiotemporal spots, which enables the construction of the travel HIN and further leverage of context preserving techniques like PATH, NEIGHBOR and PHINE. 3) Both PATH and NEIGHBOR achieve significant improvements over NONET on all datasets, which justifies our intuition of modeling the travel data in an HIN and demonstrates the effectiveness of jointly preserving passenger experience and entity structural contexts on the HIN. NEIGHBOR generally performs better than PATH, probably because randomly sampling long paths on the HIN breaks down semantically meaningful structures and introduces more noise than blindly preserving the complete neighborhoods. 4) PHINE further outperforms both PATH and NEIGHBOR by a large margin, indicating the advantage of preserving HIN structures through pattern-aware embedding.

The relative performance of the compared algorithms across



(a) Driver embedding generated by PATH (b) Driver embedding generated by NEIGHBOR (c) Driver embedding generated by PHINE



(d) Passenger embedding generated by PATH (e) Passenger embedding by NEIGHBOR (f) Passenger embedding generated by PHINE

Fig. 6. **Driver and passenger embeddings generated by different algorithms on part of our random_day dataset.**

	random				driver		
Algorithm	**day**	**week**	**month**	**Algorithm**	**day**	**week**	**month**
NOBIND	$33.14(327s)$	$32.47(22.6m)$	$31.86(1.6h)$	**NOBIND**	$33.68(287s)$	$32.58(19.8m)$	$32.05(1.4h)$
NONET	$32.94(325s)$	$32.26(22.5m)$	$31.78(1.6h)$	**NONET**	$33.21(288s)$	$32.19(19.8m)$	$31.92(1.4h)$
PATH	$25.24(568s)$	$23.35(35.8m)$	$21.77(2.8h)$	**PATH**	$24.60(529s)$	$23.14(31.6m)$	$21.49(2.6h)$
NEIGHBOR	$22.62(769s)$	$21.75(51.7m)$	$20.83(4.1h)$	**NEIGHBOR**	$22.03(732s)$	$20.91(49.6m)$	$20.07(3.9h)$
PHINE	$15.36(482s)$	$15.01(30.8m)$	$14.81(2.3h)$	**PHINE**	$14.92(430s)$	$14.54(28.4m)$	$14.28(2.1h)$
	passenger				**spot**		
Algorithm	**day**	**week**	**month**	**Algorithm**	**day**	**week**	**month**
NOBIND	$32.42(238s)$	$31.93(17.0m)$	$31.05(1.3h)$	**NOBIND**	$32.96(297s)$	$32.14(17.2m)$	$31.88(1.1h)$
NONET	$32.00(235s)$	$31.48(17.0m)$	$30.89(1.3h)$	**NONET**	$32.94(297s)$	$32.21(17.3m)$	$31.90(1.1h)$
PATH	$24.72(512s)$	$22.98(28.8m)$	$21.21(2.5h)$	**PATH**	$23.56(489s)$	$22.15(26.4m)$	$20.97(2.2h)$
NEIGHBOR	$21.63(714s)$	$20.38(48.8m)$	$19.79(3.9h)$	**NEIGHBOR**	$20.44(686s)$	$19.25(43.1m)$	$18.70(3.4h)$
PHINE	$14.26(398s)$	$13.56(25.4m)$	$13.13(2.0h)$	**PHINE**	$14.48(329s)$	$13.31(20.6m)$	$12.81(1.8h)$

TABLE III
PERFORMANCES OF PHINE IN TERMS OF MSE ($\times 1000$) AND RUNTIME COMPARED WITH OTHER BASELINE ALGORITHMS.

different datasets are also interesting. All algorithms constantly perform better when the orders are filtered by removing the ones around inactive drivers, passengers and spots, which indicates the challenge of data sparsity and motivates the modeling of entity proximity. Moreover, the sparsity of spots seem to influence the overall performance the most, followed by passengers and drivers, because filtering out orders around the unpopular spots constantly leads to the best performance of almost all algorithms. Finally, while it is quite natural that richer historical data allows better performance, the performance gaps among datasets of orders generated within one day, one week and one month are not so big compared with those among the results of different algorithms, which is promising and implies the opportunity of fast online prediction with less training data.

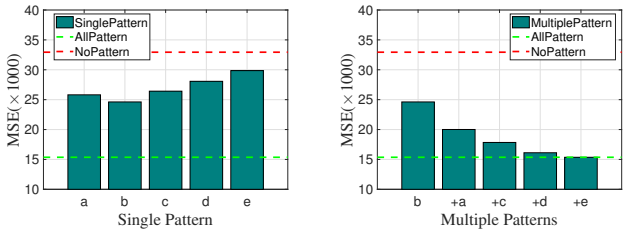
3) Runtime Comparison: We also record the runtime of different algorithms to conduct comprehensive runtime analysis. All algorithms are deployed on the servers in DiDi Big Data Center. Each machine has 12 Intel Xeon E5-2620 2.40GHz CPU cores and 64GB memory. The algorithms are implemented with PyTorch[3] on Anaconda[4] using 10 threads. As can be observed from Table III, algorithms with a sole embedding module like NOBIND and NONET are faster, and adding context preserving module leads to computation overhead. The overhead mainly comes from the joint optimization process, because the computations of sampling paths for PATH, computing neighborhoods for NEIGHBOR and sampling patterns for PHINE can be done offline with data preprocessing. The overhead of PHINE is the smallest compared with PATH and NEIGHBOR, both due to the highly parallelable context preserving process based on sampled

[3]http://pytorch.org/
[4]https://www.continuum.io/what-is-anaconda

target-context pairs (similar to PATH) and fast convergence (faster than PATH, probably because the semantics captured by regular patterns are more consistent).

4) Pattern Selection: Our PHINE framework leverages intuitive HIN patterns constructed based on prior knowledge about the data. We study how different patterns influence the overall prediction performance and present the experimental results in Figure 7. As can be seen in Figure 7(a), considering each single pattern alone generally results in much better performance than considering no pattern at all, while some patterns (*e.g.*, *a* and *b*) are more significant than other patterns (*e.g.*, *d* and *e*). Since there are too many ways of combining multiple patterns into consideration, we further analyze the performance of PHINE *w.r.t.* different pattern sets by additively adding the most significant single pattern. Figure 7(b) clearly demonstrates that considering more useful patterns generally leads better performance, and the performance converges as more nonsignificant patterns are added.

Note that this evaluation process also implies a general and efficient way to select useful patterns for effective HIN embedding, *i.e.*, 1) come up with a set of candidate patterns based on prior knowledge about the data; 2) evaluate their performance one by one; 3) greedily grow a set of multiple patterns based on the performance of single patterns until the overall performance does not increase. The effectiveness of this pattern selection process has also been demonstrated in a recent work that considers meta-path selection on HIN [29].



(a) PHINE with a single pattern (b) PHINE with multiple patterns

Fig. 7. **Pattern selection performance: the performance largely improves as important patterns are added and converges as more nonsignificant patterns are added.**

5) Neural Architecture: Now we study the effects of different embedding architectures on the performance of PHINE, through an in-depth analysis of the key components under various parameter settings.

Since there is no previous work on travel data that uses neural networks to model heterogeneous entities like drivers, passengers and spots, we are interested in how the deep architectures can benefit the modeling of entity features and interactions. To this end, we study the effectiveness of neural architectures by applying different neural components for entity combination and varying the number of hidden layers for interaction exploration. The analysis is done on the *random_day* dataset.

For entity combination, since each order can involve various numbers of passengers and spots, we use max pooling, summation and LSTM cells to combine the embedding of multiple entities. For interaction exploration, we fix the size of the last hidden layer to 16 and vary the number of hidden layers from 0 to 4. The sizes of hidden layers are doubled upwards and the size of entity embedding is half of the first hidden layer. *E.g.*, if the number of hidden layers is 3, the sizes of layers are $64 \rightarrow 32 \rightarrow 16$, and the size of entity embeddings is 32. The other model parameters are set to their defaults as described in Sec. IV.A.

Table IV shows the model performance with different neural architectures, which clearly demonstrate the effectiveness of applying the expressive neural networks for modeling the features and interactions of HIN entities. Specifically, stacking more nonlinear layers generally improves model performance, while the improvements are more significant when there are fewer layers. LSTM is the best model for the combination of multiple passengers and spots, probably because it captures the sequential order among involved passengers and spots.

Architecture	MaxPooling	Summation	LSTM
Concatenation	188.26	186.10	184.03
$\mathbf{16}$	90.38	90.89	84.57
$\mathbf{32 \rightarrow 16}$	25.12	24.76	21.06
$\mathbf{64 \rightarrow 32 \rightarrow 16}$	15.88	16.18	15.36
$\mathbf{128 \rightarrow 64 \rightarrow 32 \rightarrow 16}$	15.94	16.44	16.62

TABLE IV
PERFORMANCE WITH DIFFERENT NEURAL ARCHITECTURES.

6) Impact Analysis: Besides producing accurate passenger satisfaction predictions, we aim to understand what factors lead to good/bad rides. To this end, we borrow the idea of sensitivity analysis [30]. To account for the various factors (features) associated with drivers, passengers and spots, we adopt the OFAT (One-Factor-At-a-Time) approach described in [31], [32]. Specifically, we leverage our learned PHINE model by changing one input factor at a time while fixing all other factors to their original values, and monitoring how much the output prediction score changes. The way we change the particular input factor is simply to introduce corruption to the original data by randomly sampling a value for the factor uniformly from its all feasible values. Then we rank the top 10 key factors we have found that by themselves mostly affect passenger experience.

Table V shows the top 10 key factors we have found significant and their corresponding sensitivity scores, which are normalized among all factors. As we can observe, features related to spots are the generally more significant, followed by a mixture of passenger and driver features, among which driver features seem to win by a notch. The results are a bit surprising, because conventional belief may attribute passenger experience mostly to drivers as the service providers. However, the results nonetheless make sense, because spots related features are more likely to encode signals about traffic conditions such as travel times, speeds and distances, which substantially influence passengers' experiences. Drivers' historical service quality (indicated by the number of complaints received from passengers and the amounts of fees not paid by passengers) certainly has a significant influence, but passengers also have

their own rating standards and conventions and their future ratings can be predicted based on past behaviors (indicated by number of complaints received from drivers and number of phone complaints they make) to some extent.

Feature Name	Related Entity	Normalized Sensitivity
finish_time	spot	1
departure_time	spot	0.9826
starting_lng	spot	0.8357
passenger_complaints	driver	0.8131
dest_lng	spot	0.8069
starting_lat	spot	0.8048
dest_lat	spot	0.7874
7days_no_pay_orders_fee	driver	0.6770
driver_complaints	passenger	0.6518
7days_phone_complaints	passenger	0.6452

TABLE V
TOP 10 FACTORS INFLUENCING PHINE PREDICTIONS.

We are aware that OFAT is unable to capture all interactions among factors because it fixes all but one input variable at each time. However, our analysis is done on a large dataset with over 1 billion transactions, which effectively explores various combination of corrupted data. It is also possible to further study important combination of features through screening based on our learned PHINE model, which we defer as future work.

V. Related Work

On the application side, we review related work in transportation data mining, which extensively analyzes transportation data for different prediction tasks. On the technique side, we compare our framework with existing network embedding methods, which also model data in networks to leverage their proximity for downstream applications.

A. Transportation Data Mining

Transporation data mining has been attracting ever-increasing attention over the past few years. The first related line of work along this direction is urban activity modeling, which explores transportation data and GPS records to model people's activities in different locations and time or discover the functions of different geographical areas in the urban space. Meanwhile, researchers have also integrated transportation data with external data sources (*e.g.*, energy consumption, air pollution) to model the behaviors of moving objects in rich contexts.

Along another line, there have been work [33] on detecting patterns from trajectory data. Earlier works try to automatically discover locations and travel sequences that could be of interest to users from GPS trajectories. Later, different mobility patterns such as frequent sequential patterns [34], [35] and periodic patterns [36] have been introduced and mined to reveal the mobility regularities underlying people's movements. Finally, a considerable amount of research efforts have been devoted to taxi sharing [37], where the goal is to develop large-scale ridesharing algorithms that satisfy user needs in dynamic environment settings.

None of the above studies have considered the problem of estimating passenger experience. In contrast, we model our rich structured travel data in a transportation HIN and capture the semantic proximities among different entities based on a novel method of HIN embedding. While our framework fits well with the particular passenger experience prediction problem, it is also a general HIN embedding technique that can be easily applied to various learning problems over HINs.

B. Network Embedding

Recently, extensive developments have been witnessed in network embedding, most of which essentially adapt successful neural models such as Skipgram [8], [11], CNN [38], [39], [40] and RNN [41], [42] from word embedding [25], image processing [43] and natural language processing [44]. Among the popular ones, [8], [11], [13] recover pairwise distances by sampling the network into random paths, while [9], [15], [16] compute context distributions through preserving local neighborhoods. Although they can both capture high-order proximities to some extent, long random paths and complete local neighborhoods do not necessarily utilize semantically meaningful network patterns, especially under the HIN setting, where links and some particular network structures carry rather different semantic meanings [45], [7]. To deal with HIN, [29] proposes an embedding method based on meta paths, but the framework only works for a particular task because it does not jointly learn the representations of all related entities in the HIN. Moreover, they do not consider semantic patterns other than meta paths.

Different from all of them, the exact problem setting of supervised passenger experience prediction with rich context in our unique travel data motivates us to come up with intuitive semantic patterns, as well as the design of a novel and general joint learning framework that can receive end-to-end training *w.r.t.* the specific task while trivially take any semantic patterns into consideration to boost the overall performance.

VI. Conclusion

In this paper, we study the problem of passenger experience prediction and understanding. We model the various entities involved in a ride and their multi-type multi-dimensional interactions in a transportation HIN. To complement limited entity features and address data scarcity, we leverage the similarities among entities through PHINE, a novel HIN embedding algorithm based on spatial-temporal node binding and grouping, semantic pattern sampling and end-to-end joint training. We deploy PHINE in DiDi's Big Data Center to deliver high-quality passenger experience predictions and provide insightful interpretations into the key factors of good/bad rides. Extensive experiments on multiple travel datasets demonstrate the effectiveness of our approach. While PHINE fits well with our particular problem and unique data, it is a general HIN embedding technique and it is promising to apply it to other interesting downstream applications in the future.

REFERENCES

[1] X. Wang, F. He, H. Yang, and H. O. Gao, "Pricing strategies for a taxi-hailing platform," *Transportation Research Part E: Logistics and Transportation Review*, vol. 93, pp. 212–231, 2016.

[2] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 38, 2014.

[3] A. Noulas, C. Mascolo, and E. Frias-Martinez, "Exploiting foursquare and cellular data to infer user activity in urban environments," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, vol. 1. IEEE, 2013, pp. 167–176.

[4] M. L. Anderson, "Subways, strikes, and slowdowns: The impacts of public transit on traffic congestion," *The American Economic Review*, vol. 104, no. 9, pp. 2763–2796, 2014.

[5] M. Guarnieri and J. R. Balmes, "Outdoor air pollution and asthma," *The Lancet*, vol. 383, no. 9928, pp. 1581–1592, 2014.

[6] M. M. Waldrop *et al.*, "No drivers required," *Nature*, vol. 518, no. 7537, pp. 20–20, 2015.

[7] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 20–28, 2013.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701–710.

[9] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International World Wide Web Conference*, 2015, pp. 1067–1077.

[10] C. Zhang, K. Zhang, Q. Yuan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, and J. Han, "Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 361–370.

[11] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[12] Z. Yang, W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[13] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." in *International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[14] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1165–1174.

[15] S. Cao, W. Lu, and Q. Xu, "Grarep:learning graph representations with global structural information," in *Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[16] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[17] R. Kimball and M. Ross, *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.

[18] X. Ren, A. El-Kishky, C. Wang, F. Tao, C. R. Voss, and J. Han, "Clustype: Effective entity recognition and typing by relation phrase-based clustering," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 995–1004.

[19] X. Ren, M. Jiang, J. Shang, and J. Han, "Building structured databases of factual knowledge from massive text corpora," in *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017, pp. 1741–1745.

[20] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017.

[21] F. F. Reichheld and S. R. Covey, *The ultimate question: Driving good profits and true growth*. Harvard Business School Press Boston, MA, 2006.

[22] P. C. Smith *et al.*, "The measurement of satisfaction in work and retirement: A strategy for the study of attitudes." 1969.

[23] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[24] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[25] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[26] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[27] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 287–297.

[28] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[29] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 295–304.

[30] A. Saltelli, "Sensitivity analysis for importance assessment," *Risk analysis*, vol. 22, no. 3, pp. 579–590, 2002.

[31] J. Campbell, G. Carmichael, T. Chai, M. Mena-Carrasco, Y. Tang, D. Blake, N. Blake, S. Vay, G. Collatz, I. Baker *et al.*, "Photosynthetic control of atmospheric carbonyl sulfide during the growing season," *Science*, vol. 322, no. 5904, pp. 1085–1088, 2008.

[32] J. M. Murphy, D. M. Sexton, D. N. Barnett, G. S. Jones *et al.*, "Quantification of modelling uncertainties in a large ensemble of climate change simulations," *Nature*, vol. 430, no. 7001, p. 768, 2004.

[33] Y. Zheng, L. Zhang, X. Xie, and W. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, 2009, pp. 791–800.

[34] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *KDD*, 2007, pp. 330–339.

[35] C. Zhang, J. Han, L. Shou, J. Lu, and T. F. L. Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *PVLDB*, vol. 7, no. 9, pp. 769–780, 2014.

[36] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye, "Mining periodic behaviors for moving objects," in *KDD*, 2010, pp. 1099–1108.

[37] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *ICDE*, 2013, pp. 410–421.

[38] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," *Computer Science*, 2013.

[39] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[40] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[41] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *Computer Science*, 2015.

[42] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[44] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[45] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 3, p. 11, 2013.