On Representation Learning for Road Networks

MENG-XIANG WANG, School of Computer Science and Engineering, Northeastern University, China
WANG-CHIEN LEE and TAO-YANG FU, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, USA
GE YU, School of Computer Science and Engineering, Northeastern University, China

Informative representation of road networks is essential to a wide variety of applications on intelligent transportation systems. In this article, we design a new learning framework, called Representation Learning for Road Networks (RLRN), which explores various intrinsic properties of road networks to learn embeddings of intersections and road segments in road networks. To implement the RLRN framework, we propose a new neural network model, namely Road Network to Vector (RN2Vec), to learn embeddings of intersections and road segments jointly by exploring geo-locality and homogeneity of them, topological structure of the road networks, and moving behaviors of road users. In addition to model design, issues involving data preparation for model training are examined. We evaluate the learned embeddings via extensive experiments on several real-world datasets using different downstream test cases, including node/edge classification and travel time estimation. Experimental results show that the proposed RN2Vec robustly outperforms existing methods, including (i) *Feature-based methods*: raw features and principal components analysis (PCA); (ii) *Network embedding methods*: DeepWalk, LINE, and Node2vec; and (iii) *Features + Network structure-based methods*: network embeddings and PCA, graph convolutional networks, and graph attention networks. RN2Vec significantly outperforms all of them in terms of F1-score in classifying traffic signals (11.96% to 16.86%) and crossings (11.36% to 16.67%) on intersections and in classifying avenue (10.56% to 15.43%) and street (11.54% to 16.07%) on road segments, as well as in terms of Mean Absolute Error in travel time estimation (17.01% to 23.58%).

CCS Concepts: • **Computing methodologies → Neural networks**; **Learning latent representations**; **Model development and analysis**;

Additional Key Words and Phrases: Road network, representation learning, intelligent transportation systems

ACM Reference format:
Meng-Xiang Wang, Wang-Chien Lee, Tao-Yang Fu, and Ge Yu. 2020. On Representation Learning for Road Networks. *ACM Trans. Intell. Syst. Technol.* 12, 1, Article 11 (December 2020), 27 pages.
https://doi.org/10.1145/3424346

This work is supported in part by the National Science Foundation under Grant No. IIS-1717084. Ge Yu and Meng-xiang Wang are supported by the National Natural Science Foundation of China (U1811261) and the State Scholarship Fund of the China Scholarship Council (201806080042).

Authors' addresses: M.-X. Wang and G. Yu, School of Computer Science and Engineering, Northeastern University, No.195 Changxin Road, Hun'nan District, Shenyang, P.R. China, 110169; emails: wmx0425@gmail.com, yuge@mail.neu.edu.cn; W.-C. Lee and T.-Y. Fu, Department of Computer Science and Engineering, The Pennsylvania State University, W332 Westgate Building, University Park, PA 16802; emails: {wlee, txf225}@cse.psu.edu.

2157-6904/2020/12-ART11 $15.00
https://doi.org/10.1145/3424346

1 INTRODUCTION

Owing to population growth, urbanization, and technological advances, efforts for smartening the various infrastructures of cities have been planned in many countries. As an important branch of the smart city, intelligent transportation systems (ITS) [19] have received significant interests from academia, industry and governments, owing to the growing demands of solutions to address various transportation issues, such as traffic congestion, pollution, and accidents.

The core of ITS includes functionalities of analysis, mining, predictions, and management built upon data collected from various sources (e.g., road networks, vehicle trajectories, traffic signal control systems, etc.) Among them, road networks are arguably the most basic due to the ubiquitous needs in ITS. For instance, road networks are essential to digital maps (e.g., Google Map and OpenStreetMap (OSM) [4]), online ride-hailing services (e.g., Uber and Lyft), self-driving cars, and various traffic analysis tasks. Typically, a road network is modeled as a graph, which consists of *intersections* (or junctions) as nodes[1] and *road segments* as edges.[2] Additionally, both the intersections and road segments in road networks contain not only geo-spatial information (e.g., the coordinates) but also various information about road characteristics and transportation facilities, such as traffic signal and stop signs on intersections, road categories (e.g., primary and motorway), number of lanes of road segments, and so on. Take the small portion of the San Francisco road network in Figure 1 as an example. The blue lines are road segments and blue circles are intersections. Also shown are information tagged by volunteers for intersections (e.g., traffic signal, crossings, stop signs, and motorway junctions) and road segments (e.g., the primary way in orange and motorway links in pink). As ITS mainly focus on road transportation, capturing network structure and associated information to represent road networks are important for various applications, such as travel time estimation [27], destination prediction [32], and intelligent speed adaptation [10].

To achieve good performance in these ITS applications, high-quality features of intersections or road segments that capture intrinsic properties of road networks are much needed. This conventionally requires labor-intensive feature engineering effort by domain experts. Recently, representation learning techniques [35], aiming to automatically learn useful latent feature vectors (also called *embeddings*)[3] of data objects as inputs to machine learning or data mining algorithms, have been developed and well received in the fields of speech recognition [17], computer vision [18], natural language processing (NLP) [9, 25], and so on. In this article, inspired by the successes in these fields, we investigate the issue of representation learning for real-world road networks, aiming to learn useful road network embeddings for general support of various ITS applications. Here we focus on learning embeddings for intersections and road segments, to capture rich feature information inherent to those components in preparation for *general use* in ITS applications.

To learn representations of road networks, it is natural to apply existing network representation learning methods [8, 21, 23, 24, 33] by treating a road network as a general network. However, this idea is impractical. First, while existing methods capture the topological structure of networks, they do not consider the spatial properties of road networks, e.g., the geo-locality of intersections/road segments. Second, they mostly apply random walks to sample the structure of networks, which fails to capture the moving behaviors of mobile road users who tend to take the shortest paths to move from the sources to destinations. Finally, they do not incorporate various homogeneity relationships of intersections/road segments (e.g., the sharing of some common

[1] In this article, we use the terms "intersection," "junction," and "node" interchangeably.
[2] In this article, we use the terms "road segment" and "edge" interchangeably.
[3] We use the terms "latent feature vectors," "embeddings," and "representations" interchangeably in this article.

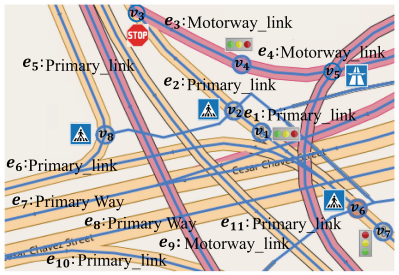


Fig. 1. A portion of San Francisco road network.

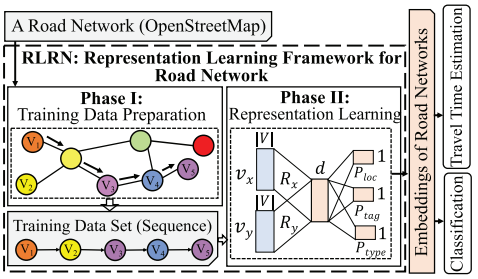


Fig. 2. Overview of the RLRN framework.

properties such as traffic signal by two intersections or motorways by two road segments) into their models.

To fill this gap between the conventional network representation learning methods and one tailored specifically for road networks, we design a new learning framework, called Representation Learning for Road Network (RLRN), as depicted in Figure 2, which explores the intrinsic properties of a road network to learn discriminative embeddings of its intersections and road segments, two essential and most important elements of road networks. In specific, the RLRN framework consists of two phases: (I) Training data preparation and (II) Representation learning. In Phase I, we propose a novel data preparation module that generates training data by sampling *shortest paths* on road networks to simulate user moving behaviors. Additionally, we supplement the above training data using real-world trajectory data for model training. In Phase II, we propose a novel neural network model, namely, Road Network to Vector (RN2Vec), to learn the embeddings of intersections and road segments jointly. More specifically, RN2Vec consists of three sub-modules, namely, Intersection of Road Network to Vector (IRN2Vec) [28],[4] Segment of Road Network to Vector (SRN2Vec), and Intersection and Segment of Road Network to Vector (ISRN2Vec), which learn embeddings of intersections and road segments by predicting various relationships among (i) intersections, (ii) road segments, and (iii) between intersections and road segments, respectively.[5] Finally, we evaluate the effectiveness of the learned road network embeddings by five predictive ITS applications, including traffic signal classification and crossing classification for intersection embeddings; street classification and avenue classification for road segment embeddings; and travel time estimation for both.

The main contributions of this work are as follows.

- **Novel ideas for road network representation learning.** We analyze the intrinsic properties of the road networks and explore novel ideas to learn embeddings for intersections and road segments, which support various ITS applications with excellent performance.
- **A new representation learning framework for road networks.** We propose RLRN, a two-phase framework for road network representation learning. In Phase I, we sample shortest paths, supplemented by paths travelled by road users, for training data preparation. In Phase II, we propose RN2Vec, a novel neural network model that learns embeddings for intersections and road segments by encoding the network structural information and predicting various relationships among intersections and road segments jointly.

[4]The preliminary research result on IRN2Vec is reported in SIGSPATIAL 2019.
[5]Although IRN2Vec and SRN2Vec are the sub-modules in RN2Vec, they can be adapted to learn embeddings of intersections and road segments separately.

- **Empirical study on multiple predictive applications using real-world data.** We conduct extensive experiments to evaluate RN2Vec on various missing tag prediction tasks and a travel time estimation task on multiple real-world datasets. Experimental results show that the proposed RN2Vec robustly outperforms three categories of existing methods, including (i) Feature-based methods: raw features and principal components analysis (PCA); (ii) Network embedding methods: DeepWalk, LINE, and Node2vec; and (iii) Features + Network structure-based methods: network embeddings and PCA, graph convolutional networks, and graph attention networks. Specifically, RN2Vec significantly outperforms them, in terms of *F1-score* in classifying traffic signals (11.96% to 16.86%) and crossings (11.36% to 16.67%) for intersections, and classifying avenue (10.56% to 15.43%) and street (11.54% to 16.07%) for road segments, as well as in terms of *Mean Absolute Error* in travel time estimation (17.01% to 23.58%).

The remainder of this article is organized as follows. We first review the related work in Section 2 and then present our problem formulation and data preprocessing in Section 3. Next, we detail the proposed neural network model RN2Vec in Section 4 and report the evaluation result on real-world data in Section 5. Finally, we conclude the article in Section 6 and discuss future research directions.

2 RELATED WORK

In past several years, the topic of representation learning [1, 35] has received significant interests in the fields of machine learning and data mining, owing to its advantages in reducing the labor-intensive effort in feature engineering. Compared with traditional feature engineering methods (e.g., feature extraction and principal components analysis), the goal of representation learning is to automatically transform raw data into low-dimensional latent vectors (i.e., embeddings), as input features to machine learning and data mining algorithms. Recently, neural network-based representation learning models have achieved great success in various domains, including NLP [9], speech recognition [17] and computer vision [18], and so on.

Inspired by these advances, research on representation learning has been extended to network data. Network embedding methods [8, 21, 23, 24, 33] have been proposed to embed network nodes and edges into a latent space as feature vectors that preserve their roles and structural properties in the network. These works typically assume that nearby nodes on network are relevant and thus tend to place their learned representations close to each other in the latent feature space. *DeepWalk* [21] learns node representations by sampling nodes in the network via uniform random walks to train *Skip-gram* model [16] by maximizing the likelihood of predicting whether two nodes are within *k-hop* to each other. *Node2vec* [8] also aims to learn node representations but focuses on the issue of sampling the network neighborhood by applying parameterized random walks rather than uniform random walks. The parameterized random walks have two parameters: return parameter p in and out parameter q, which controls the walking process based on the probability of returning back to the previous node and the probability of selecting the next node away from the previous node to simulate *BFS* and *DFS* search strategies, respectively. Instead of employing random walks, *LINE* [24] basically samples node relevance in accordance with the frequencies of their 1-hop and 2-hop connectivity. It captures first-order similarity (similarity between adjacent nodes) and second-order similarity (similarity between nodes in terms of their common neighbors), separately, to learn two representations of nodes that are used together by concatenation.

These aforementioned methods leverage only network structural information to obtain node embeddings, without considering homogeneity information or attributes associated with vertices in networks. Content enhanced embedding methods assume node content information is available

and exploit both topological information and content features simultaneously. Text-Associated DeepWalk (TADW) [33] combines the network and text information associated with nodes to learn node embeddings. Content-Enhanced Network Embedding [23] aims to learn various context-aware embeddings for a vertex by incorporating information from its neighbors. In our work, we explore common tags and types of edges and nodes in road networks to establish relationships in our model. Recently, Graph Neural Network (GNN) models [12, 26] have achieved great success in various graph analytics applications, e.g., node classification, in social, citation and biological networks. Based on the concept of label propagation and information diffusion in a network, these models propose to learn embeddings of nodes by aggregating information (e.g., raw attributes of nodes or their embeddings) from their neighborhood. Graph Convolutional Network (GCN) [12] treats the edges of a given network as links of neurons between two layers in a neural network to carry out label propagation in the network. By adopting multiple layers in the model, GCN performs the label propagation and graph convolution (i.e., aggregate neighbors' embeddings) within k-hop for each node to enable the final prediction. However, Graph Attention Network (GAT) [26] adopts attention mechanisms to learn the weights between two connected nodes, to perform weighted combination of neighbors' embeddings. However, these GNN models are designed to tackle a prediction task in an end-to-end manner instead of targeting on the representation learning problem. Moreover, they are designed for general networks, not for road networks.

So far, there are only a few works considering road network representation learning for ITS [5, 6, 11, 14]. A case study on Danish road network applies *Node2vec* to empirically learn road segment embeddings for road category classification and speed limit classification [11]. DeepWalk is employed to prepare road segment embeddings for multi-step trajectory prediction [5]. Road2Vec [14] quantifies the traffic interactions among road segments and employs the ideas of Word2Vec and DeepWalk to learn embeddings of road segments. Instead of traffic interactions, attribute information from nodes and edges are incorporated in random walks to learn road segment embeddings [6]. These works either simply apply representation learning methods designed for general networks or extended with only some simple attributes of road segments, without considering the influence of intersections and the relationships between intersections and road segments. Moreover, these existing works are not evaluated by different ITS applications.

Different from existing works, the proposed model RN2Vec explores the intrinsic geo-spatial properties of intersections and road segments and various relationships established based on geo-closeness and common characteristics of intersections and road segments so as to learn an effective road network representation. Additionally, although some existing works explore the human behavior analysis via trajectory data [7, 20, 29–31, 36, 37], there exists an issue of incomplete coverage in road networks. Thus, the RLRN framework samples the training data based on both of real-world trajectories as well as shortest paths that resemble user moving behaviors. Finally, the embeddings learned by RLRN can generally support multiple ITS applications.

3 PRELIMINARIES

In this section, we define important terms used throughout the article, formulate the tackled problem, discuss the challenges, and detail the data preprocessing.

3.1 Terminology

Definition 1 (Road Network). A road network is an undirected graph $G = (V, E, \Psi, \Omega)$[6], where V is a set of nodes (i.e., intersections[7]); $E \subseteq V \times V$ is a set of edges (denoted as road segments);

[6]While road networks can be modeled as a directed graph, the design of our RLRN framework does not consider directions.
[7]Without loss of generality, we also consider terminal/end of a road as an intersection.

$\Psi : V \rightarrow A$ is a descriptor function that describes a node $v \in V$ by a set of pre-defined tags in A to capture its characteristics, i.e., $\Psi(v) \in A$; and $\Omega : E \rightarrow B$ is a descriptor function that describes an edge $e \in E$ by a set of pre-defined tags in B to capture its characteristics, i.e., $\Omega(e) \in B$.

Note that the characteristics sets A and B in Definition 1 may be derived from the road networks or made available by government agencies, volunteer-generated effort, or commercial services. We further discuss them later in Section 3.3.

Definition 2 (Trajectory). A trajectory $T = \{p_1, p_2, \ldots, p_{|T|}\}$ is a sequence of spatio-temporal sample points generated from the underlying route of a mobile road user. Each sample point p_i contains a location (x_i, y_i) (i.e., longitude and latitude) and a timestamp t_i.

Definition 3 (Road Network Sequence). A road network sequence $S_R = \{v_1, e_1, v_2, e_2, \ldots, v_{|S_R|}\}$ is a sequence of intersections and road segments where v_i and e_i are the ith intersection and road segment in the path of S_R generated by a path sampling method or transformed from a trajectory.

3.2 Problem Definition and Analysis

The goal of this work is to learn embeddings of intersections/road segments for use as input features to various predictive and analytical ITS applications. We formally define the problem as follows.

Definition 4 (Representation Learning on Road Networks). Given a road network, denoted as an undirected graph $G = (V, E, \Psi, \Omega)$, this problem learns two functions: (i) $f_V : V \rightarrow \mathbb{R}^d$, which projects each intersection $v \in V$ to a vector in a d-dimensional latent space $\mathbb{R}^d$, (ii) $f_E : E \rightarrow \mathbb{R}^d$, which projects each road segment $e \in E$ to a vector in a d-dimensional latent space $\mathbb{R}^d$, where $d \ll |V|$ and $d \ll |E|$.

In this work, we propose the RLRN framework and a neural network model RN2Vec to tackle the representation learning problem on road networks. Although there are a few works considering representation learning for road segments [5, 6, 11, 14], they are mostly applying some existing representation learning methods designed for general networks to the road network data, without exploring properties unique in road networks. Our design of RN2Vec consists of three sub-modules, namely, IRN2Vec, SRN2Vec, and ISRN2Vec, which are based on the same ideas and design principle but different focus, by exploring intrinsic geo-spatial properties and characteristics of intersections and road segments.

To realize the RLRN framework and the RN2Vec model, we face the following challenges. (1) *Model design:* A well-designed neural network model is essential for effective and efficient learning. (2) *Geo-spatial characteristics:* Different from general networks, a road network is characterized by specific information about roads, e.g., traffic signals and stop signs on intersections, the type of road segments (such as primary way and residential way), and so on. Thus, identifying information on intersections and road segments to explore their relationships (in terms of relevance and similarity) for representation learning of the road network elements requires careful study. (3) *Training data preparation:* Training data need to be prepared and tailored based on the learning logic behind the proposed RN2Vec model. For road network representation learning, a good network sampling method that captures the user moving behaviors in road networks is a mandate.

3.3 Data Preprocessing

In our work, we extract road networks and user-generated tags associated with intersections and road segments from the OSM [4], a publicly editable and accessible map service. We download the raw OSM data of several cities (e.g., San Francisco, Porto, and Tokyo) to generate the road

Table 1. Dataset Statistics for Road Network

Intersections' tags	San Francisco	Porto	Tokyo	Segments' categories	San Francisco	Porto	Tokyo
Turning_loop	51	17	84	Residential	26,519	78,535	121,289
Give_way	89	458	44	Sevice	28,937	18,005	27,784
Bus_stop	178	2,052	4,905	Unclassified	1,870	16,043	89,355
Turning_circle	1,198	454	36	Primary	4,364	10,301	11,627
Crossing	5,021	6,934	14,377	Secondary	5,645	8,776	10,180
Traffic_signals	2,582	1,646	8,470	Tertiary	6,411	13,976	33,260
Motorway_junction	300	506	257	Motorway	669	2,056	774
Stop	243	786	572	Trunk	612	1,386	5,286
Speed_camerca	26	82	53	Primary_link	157	443	1163
Mini_roundabout	148	131	61	Secondary_link	148	369	151
—	—	—	—	Tertiary_link	65	370	186
—	—	—	—	Motorway_link	1,119	1,834	852
—	—	—	—	Trunk_link	89	463	474
Total Coverage	16.84%	8.48%	13.29%	**Total Coverage**	99.81%	98.99%	98.87%

networks. Specifically, the OSM data mainly includes three sets of elements: *Nodes*, *Ways*, and *Relations*. Among them, *Nodes* contains the position of points in the geo-space, Ways denote the road segments or boundaries of areas, while *Relations* contains the relationship between elements. Since the focus of this article is on intersections and road segments in road networks, we extract intersections from Nodes and road segments from Ways to form road networks. In the following, we introduce how we extract tags and types of intersections/road segments.

3.3.1 Intersections. As mentioned previously, intersections may share some common characteristics. Thus, we extract two kinds of information on intersections: *intersection tags* and *intersection types*. Intersection tags are extracted from the "Highway" tag of Nodes in OSM data for an intersection. Specifically, a node consists of a unique ID, coordinate (i.e., longitude and latitude), and several key-value pairs of tag attributes, e.g., <node id="25496583" lat="51.5173639" lon="-0.140043"> <tag k="highway" v="traffic_signal"/> </node>. As shown in Table 1, we select the 10 most frequently appearing tags of nodes to learn embeddings. Second, intersections may be similar due to the same intersection types. In the real world, different types of intersections, such as T-juction and X-junction, exist. However, this junction-type information needs to be captured additionally as it is not readily available in the OSM data. One natural way to characterize types of intersections is by the number of road segments (or road ways) intersected at them. In this article, we calculate the number of road segments passing through an intersection as the *N-way* type of the intersection ($N = 2, 3, \ldots, 6$). For example, a T-junction are intersected by three road segments. In the real world, five-way and six-way intersections are less common but still exist, especially in suburban areas.

3.3.2 Road Segments. Similar with intersections, road segments also share some common characteristics, captured by road segment categories. Thus, we extract road segment categories from the OSM data. In OpenStreeMap, a way consists of a unique ID of the way, several key-value pairs of tag attributes and a list of references. The list of references contain IDs of the nodes that constitute the road segment, e.g., <edge id="30758272"> <tag k="highway" v="primary"/><reference list "3152008151," "767854126," "1229779432," "767854350," "65344423"> </way>. As shown in Table 1, we extract road segments' categories from the "Highway" tag of Ways in the OSM data and select 13 most frequently appearing categories of road segments. Figure 3 shows illustrative examples of road segments in different categories (highlighted as green lines) extracted from the

(a) Primary Road (b) Secondary Road (c) Motorway

Fig. 3. A portion of Tokyo road network.

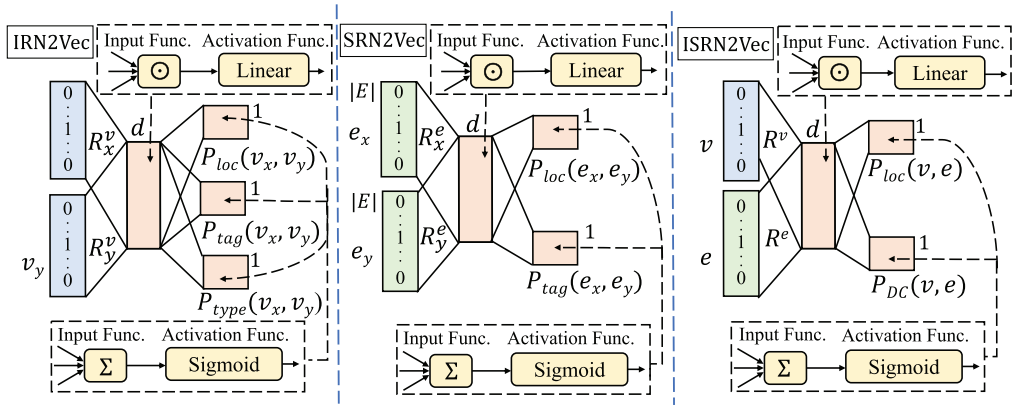


Fig. 4. The RN2Vec model.

OSM data in Tokyo. Statistics of the intersection tags and road segment categories are summarized in Table 1. Based on our observation, the coverage of the road segment categories on road segments is consistently much better than the intersection tags on intersection for different cities.

4 THE RN2VEC MODEL

In this section, we detail the design of RN2Vec. We first analyze the relationships considered among intersections and road segments, and then detail the design of sub-modules in RN2Vec, i.e., IRN2Vec, SRN2Vec, and ISRN2Vec, which respectively learn representations of intersections, road segments, and both of them jointly. Finally, we discuss the training data preparation for RN2Vec.

4.1 Relationships in Road Network

The main idea behind RN2Vec is to learn embeddings of two road network elements (i.e., intersections and road segments) by capturing the *geo-locality* and *homogeneity* relationships among them. Under the context of our study, the geo-locality relationship between two elements is established based on their closeness in terms of road distance within a given threshold. For example, using 100 m as a threshold, two intersections are considered to be local (or in the neighborhoods of each other) if they can reach each other by traveling within 100 m on road. However, the homogeneity relationship, existing only between two intersections or two road segments refers to the sharing of certain common properties. To systematically capture various geo-locality and homogeneity relationships within the road network, we design the RN2Vec model in three sub-modules, as shown in Figure 4.

Overall, RN2Vec aims to learn embeddings for intersections and road segments jointly by capturing specific relationships between two intersections (via IRN2Vec), between two road segments (via SRN2Vec) and between an intersection and a road segment (via ISRN2Vec). The specific homogeneity relationships considered in IRN2Vec and SRN2Vec are based on user-generated tags (see Table 1 for the complete list) for intersections and road segments, respectively. Additionally, in IRN2Vec, we also explore the road type of intersections, e.g., T-Junction, as a homogeneity relationship. However, the geo-locality relationship for pairs of intersections, road segments, and intersection/road segment are separately captured in their corresponding models. Note that in ISRN2Vec, in addition to the generic geo-locality mentioned above, we also explore an additional kind of geo-locality, called direct connection (DC), between an intersection and a road segment, representing their direct connection, i.e., the intersection is an endpoint of the road segment. We argue that a directly connected pair of intersection and road segment is more relevant than other indirectly connected pairs and thus incorporate this relationship in ISRN2Vec.

4.2 Representation Learning for IRN2Vec

In this section, we present IRN2Vec, the first sub-module of RN2Vec, to learn embeddings for intersections in a road network. As discussed earlier, our idea lies in capturing the geo-locality and homogeneity relationships between two intersections. Note that, in IRN2Vec, we divide the homogeneity relationships of intersections based on tags and road types, due to the nature of different data sources. If two intersections share at least one same tag, then we say they have the "same intersection tag" (or simply "same tag") relationship. However, if two intersections share the same road type, then we say they have the "same N-way type" (or simply "same type") relationship. Accordingly, we propose IRN2Vec that jointly predicts the targeted relationships between any given pair of intersections for intersection representation learning.

As shown in Figure 4, the IRN2Vec is a multi-task binary classifier that takes a pair of intersections $v_x, v_y \in V$ as the inputs to predict three relationships between them, including geo-locality, same intersection tag and same N-way type. In this model, the input layer takes in two one-hot vectors $\vec{v_x}$ and $\vec{v_y}$ of length $|V|$, representing node v_x and v_y, respectively. In the latent layer, $\vec{v_x}$ and $\vec{v_y}$ are transformed into latent vectors $R_x^{v\prime}\vec{v_x}$ and $R_y^{v\prime}\vec{v_y}$, where R_x^v and R_y^v are two $|V| \times d$ matrices consisting of all intersections' latent vectors, i.e., each row of this matrix denotes the vector for an intersection, $R_x^{v\prime}$ and $R_y^{v\prime}$ are their transpose matrices, and d is the dimensionality of the hidden space. Next, we use inner product followed by a *Sigmoid* function to predict whether two intersections v_x and v_y have a specific relationship. More specifically, to present these operations in the proposed neural network, we apply *Hadamard* function,[8] i.e., element-wise multiplication, to aggregate the two vectors, which is denoted by $R_x^{v\prime}\vec{v_x} \odot R_y^{v\prime}\vec{v_y}$, and then we apply the *Linear* function, i.e., f(x) = x, for activation. Finally, the output layer, taking *Summation* as the input function and *Sigmoid* function for activation, computes $Sigmoid(R_x^{v\prime}\vec{v_x} \cdot R_y^{v\prime}\vec{v_y})$, respectively, to predict the three relationships, correspondingly measured by (1) the probability $P_{loc}(v_x, v_y)$ for v_x and v_y to be located within a neighborhood of certain distance, (2) the probability $P_{tag}(v_x, v_y)$ for v_x and v_y to have the same intersection tag, and (3) the probability $P_{type}(v_x, v_y)$ for v_x and v_y to have the same N-way type. The three joint probabilities are derived as follows:

$$P_{loc}(v_x, v_y) = \sigma\left(\sum R_x^{v\prime}\vec{v_x} \cdot R_y^{v\prime}\vec{v_y}\right); P_{tag}(v_x, v_y) = \sigma\left(\sum R_x^{v\prime}\vec{v_x} \cdot R_y^{v\prime}\vec{v_y}\right); P_{type}(v_x, v_y)$$
$$= \sigma\left(\sum R_x^{v\prime}\vec{v_x} \cdot R_y^{v\prime}\vec{v_y}\right),$$

[8] Average and Minus are also tested empirically but Hadamard outperforms them.

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the *Sigmoid* function. In IRN2Vec, we make R_x^v and R_y^v to be the same matrix. In the training process, conceptually, if intersections v_x and v_y are observed in the training data to satisfy one of the targeted prediction tasks, then $R_x^{v\prime}\vec{v_x}$ and $R_y^{v\prime}\vec{v_y}$ are moved closer in the latent space. Otherwise, they are moved away in the latent space.

The learning of IRN2Vec model parameters (i.e., intersection embeddings) is realized by setting multiple optimization objectives and thus is critical to set the objective functions properly. To train IRN2Vec, a training dataset D_I, which contains training data entries in the form of $\langle v_x, v_y, S_{loc}(v_x, v_y), S_{tag}(v_x, v_y), S_{type}(v_x, v_y)\rangle$, extracted from the road network in the training data preparation phase (see Section 4.5). In a training data entry, $S_{loc}(v_x, v_y)$, $S_{tag}(v_x, v_y)$ and $S_{type}(v_x, v_y)$ are Boolean values, indicating whether intersections v_x and v_y are located within k-meter in the road network, whether v_x and v_y have the same intersection tag, and whether v_x and v_y have the same N-way type, respectively. With the training dataset D_I, the IRN2Vec model is trained by the backpropagation training algorithm in conjunction with asynchronous stochastic gradient ascent. It goes backward to adjust the weights in R_x^v and R_y^v for each data entry s in D_I, attempting to maximize the overall objective function O_{RN2Vec} of the *RN2Vec* model as follows:

$$O_{RN2Vec} = \lambda O_{ISRN2Vec} + \mu O_{IRN2Vec} + (1 - \lambda - \mu) O_{SRN2Vec}, \tag{1}$$

where λ and μ are weighting parameters.

While $O_{SRN2Vec}$ and $O_{ISRN2Vec}$ are to be discussed later in Section 4.3 and Section 4.4, here we define the objective function $O_{IRN2Vec}$ as a weighted combination of $O_{loc}(v_x, v_y)$, $O_{tag}(v_x, v_y)$, and $O_{type}(v_x, v_y)$ derived as follows:

$$O_{loc}(v_x, v_y) = \begin{cases} P_{loc}(v_x, v_y) & \text{if} \quad S_{loc}(v_x, v_y) = 1 \\ 1 - P_{loc}(v_x, v_y) & \text{if} \quad S_{loc}(v_x, v_y) = 0 \end{cases}, \tag{2}$$

$$O_{tag}(v_x, v_y) = \begin{cases} P_{tag}(v_x, v_y) & \text{if} \quad S_{tag}(v_x, v_y) = 1 \\ 1 - P_{tag}(v_x, v_y) & \text{if} \quad S_{tag}(v_x, v_y) = 0 \end{cases}, \tag{3}$$

$$O_{type}(v_x, v_y) = \begin{cases} P_{type}(v_x, v_y) & \text{if} \quad S_{type}(v_x, v_y) = 1 \\ 1 - P_{type}(v_x, v_y) & \text{if} \quad S_{type}(v_x, v_y) = 0 \end{cases}. \tag{4}$$

In Equations (2), (3) and (4), the functions $O_{loc}(v_x, v_y)$, $O_{tag}(v_x, v_y)$, and $O_{type}(v_x, v_y)$ quantify how IRN2Vec correctly predicts $S_{loc}(v_x, v_y)$, $S_{tag}(v_x, v_y)$, and $S_{type}(v_x, v_y)$ for a data entry s. Specifically, for a training data entry $s_I = \langle v_x, v_y, S_{loc}(v_x, v_y), S_{tag}(v_x, v_y), S_{type}(v_x, v_y)\rangle$, $O_{loc}(v_x, v_y)$ aims to maximize $P_{loc}(v_x, v_y)$, when $S_{loc}(v_x, v_y)$ is 1, and minimize $P_{loc}(v_x, v_y)$, otherwise. Similarly, $O_{tag}(v_x, v_y)$ and $O_{type}(v_x, v_y)$ aim to maximize $P_{tag}(v_x, v_y)$ and $P_{type}(v_x, v_y)$, when $S_{tag}(v_x, v_y)$ and $S_{type}(v_x, v_y)$ is 1, respectively, and minimize $P_{tag}(v_x, v_y)$ and $P_{type}(v_x, v_y)$, otherwise.

To ease the computation in the optimization process, we maximize $\log O_{loc}(v_x, v_y)$, $\log O_{tag}(v_x, v_y)$ and $\log O_{type}(v_x, v_y)$ instead of $O_{loc}(v_x, v_y)$, $O_{tag}(v_x, v_y)$, and $O_{type}(v_x, v_y)$. The objective functions are as follows:

$$\log O_{loc}(v_x, v_y) = S_{loc}(v_x, v_y) \log P_{loc}(v_x, v_y) + [1 - S_{loc}(v_x, v_y)] \log[1 - P_{loc}(v_x, v_y)], \tag{5}$$

$$\log O_{tag}(v_x, v_y) = S_{tag}(v_x, v_y) \log P_{tag}(v_x, v_y) + [1 - S_{tag}(v_x, v_y)] \log[1 - P_{tag}(v_x, v_y)], \tag{6}$$

$$\log O_{type}(v_x, v_y) = S_{type}(v_x, v_y) \log P_{type}(v_x, v_y) + [1 - S_{type}(v_x, v_y)] \log[1 - P_{type}(v_x, v_y)]. \tag{7}$$

Accordingly, the objective function $O_{IRN2Vec}$ is defined as follows:

$$O_{IRN2Vec} = \sum_{s_I \subseteq D_I} \Big\{\alpha \log O_{loc}(v_x, v_y) + \beta \log O_{tag}(v_x, v_y) + (1-\alpha-\beta) \log O_{type}(v_x, v_y)\Big\}, \quad (8)$$

where α and β are weighing parameters.

As mentioned, we apply backward propagation in conjunction with stochastic gradient ascent to maximize the overall objective function O_{RN2Vec}. Specifically, for each training data entry, it goes backward to adjust the weights of intersections v_x and v_y in $R_x^{v\prime}\vec{v_x}$ and $R_y^{v\prime}\vec{v_y}$ based on the gradients, respectively, as follows:

$$R_x^{v\prime}\vec{v_x} := R_x^{v\prime}\vec{v_x} + \eta[\alpha d \log O_{loc}(v_x, v_y) + \beta d \log O_{tag}(v_x, v_y) + (1-\alpha-\beta) d \log O_{type}(v_x, v_y)]/dR_x^{v\prime}\vec{v_x}, \quad (9)$$

$$R_y^{v\prime}\vec{v_y} := R_y^{v\prime}\vec{v_y} + \eta[\alpha d \log O_{loc}(v_x, v_y) + \beta d \log O_{tag}(v_x, v_y) + (1-\alpha-\beta) d \log O_{type}(v_x, v_y)]/dR_y^{v\prime}\vec{v_y}, \quad (10)$$

where η is the learning rate.

4.3 Representation Learning for SRN2Vec

Next, we present the second sub-module of RN2Vec, namely SRN2Vec, to learn embeddings for road segments in a road network. To generate the embedding of a road segment, one naive approach is to apply some aggregation functions, e.g., Hadamard, Addition, or Average, on the embeddings of its two end intersections (generated by IRN2Vec). However, this approach misses the information of road segments themselves, such as the geo-locations and road categories. Thus, following similar idea and the same design principle in IRN2Vec, we propose SRN2Vec by exploring the intrinsic properties of road segments for representation learning.

Given a road network, we learn road segment embeddings by exploring the relationships of geo-locality and same road category[9] between two road segments. As shown in Figure 4, SRN2Vec is also a multi-task binary classifier with a model architecture similar to IRN2Vec but targeting on different prediction objectives. More specifically, SRN2Vec takes a pair of road segments $e_x, e_y \in E$ as the inputs to predict the two aforementioned relationships between them. The input layer takes in two one-hot vectors $\vec{e_x}$ and $\vec{e_y}$ of length $|E|$, representing edge e_x and e_y, respectively. In the latent layer, $\vec{e_x}$ and $\vec{e_y}$ are transformed into latent vectors $R_x^{e\prime}\vec{e_x}$ and $R_y^{e\prime}\vec{e_y}$, where R_x^e and R_y^e are two $|E| \times d$ matrices consisting of all road segments' latent vectors, $R_x^{e\prime}$ and $R_y^{e\prime}$ are their transpose matrices, and d is the dimensionality of the hidden space. Next, we use inner product followed by a *Sigmoid* function to predict (1) the probability $P_{loc}(e_x, e_y)$ for e_x and e_y to be located within a neighborhood of certain distance and (2) the probability $P_{cate}(e_x, e_y)$ for e_x and e_y to have the same category relationship. The two joint probabilities are derived as follows:

$$P_{loc}(e_x, e_y) = \sigma\left(\sum R_x^{e\prime}\vec{e_x} \cdot R_y^{e\prime}\vec{e_y}\right); \qquad P_{cate}(e_x, e_y) = \sigma\left(\sum R_x^{e\prime}\vec{e_x} \cdot R_y^{e\prime}\vec{e_y}\right),$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the *Sigmoid* function. We make R_x^e and R_y^e identical.

To train SRN2Vec, a training dataset D_S, which contains training data entries in the form of $\langle e_x, e_y, S_{loc}(e_x, e_y), S_{cate}(e_x, e_y)\rangle$ is extracted from the road network in the training data preparation phase (see Section 4.5). In a training data entry, $S_{loc}(e_x, e_y)$ and $S_{cate}(e_x, e_y)$ are Boolean values, indicating whether road segments e_x and e_y are located within k-meter in the road network, and whether e_x and e_y have the same road category relationship, respectively. With the training dataset D_S, the SRN2Vec model is trained by the backpropagation training algorithm in

[9]Similarly with the same tag relationship in IRN2Vec, we say two road segments have the "same road category" (or simple "same category") relationship if they share at least one same category.

conjunction with asynchronous stochastic gradient ascent to maximize the overall objective function O_{RN2Vec} in Equation (1). The objectives $O_{loc}(e_x, e_y)$ and $O_{cate}(e_x, e_y)$ are derived as follows:

$$O_{loc}(e_x, e_y) = \begin{cases} P_{loc}(e_x, e_y) & \text{if } S_{loc}(e_x, e_y) = 1 \\ 1 - P_{loc}(e_x, e_y) & \text{if } S_{loc}(e_x, e_y) = 0 \end{cases};$$

$$O_{cate}(e_x, e_y) = \begin{cases} P_{cate}(e_x, e_y) & \text{if } S_{cate}(e_x, e_y) = 1 \\ 1 - P_{cate}(e_x, e_y) & \text{if } S_{cate}(e_x, e_y) = 0 \end{cases}.$$

In specific, for a training data entry $s_S = \langle e_x, e_y, S_{loc}(e_x, e_y), S_{cate}(e_x, e_y)\rangle$, $O_{loc}(e_x, e_y)$ aims to maximize $P_{loc}(e_x, e_y)$, when $S_{loc}(e_x, e_y)$ is 1, and minimize $P_{loc}(e_x, e_y)$, otherwise. $O_{cate}(e_x, e_y)$ aim to maximize $P_{cate}(e_x, e_y)$, when $S_{cate}(e_x, e_y)$ is 1, and minimize $P_{cate}(e_x, e_y)$, otherwise.

To ease the computation, we maximize $\log O_{loc}(e_x, e_y)$ and $\log O_{cate}\ (e_x, e_y)$ instead of $O_{loc}(e_x, e_y)$ and $O_{cate}\ (e_x, e_y)$. The objective functions are as follows:

$$\log O_{loc}(e_x, e_y) = S_{loc}(e_x, e_y) \log P_{loc}(e_x, e_y) + [1 - S_{loc}(e_x, e_y)] \log[1 - P_{loc}(e_x, e_y)], \quad (11)$$

$$\log O_{cate}(e_x, e_y) = S_{cate}(e_x, e_y) \log P_{cate}(e_x, e_y) + [1 - S_{cate}(e_x, e_y)] \log[1 - P_{cate}(e_x, e_y)]. \quad (12)$$

Finally, the objective function $O_{SRN2Vec}$ is defined as follows:

$$O_{SRN2Vec} = \sum_{s_S \subseteq D_S} \Big\{\gamma \log O_{loc}(e_x, e_y) + (1 - \gamma) \log O_{cate}(e_x, e_y)\Big\}, \quad (13)$$

where γ is a weighing parameter.

We apply backpropagation in conjunction with asynchronous stochastic gradient ascent to maximize the objective function $O_{SRN2Vec}$ based on the gradients derived as follows:

$$R_x^{e\prime}\vec{e_x} := R_x^{e\prime}\vec{e_x} + \eta[\gamma d \log O_{loc}(e_x, e_y) + (1 - \gamma) d \log O_{cate}(e_x, e_y)]/dR_x^{e\prime}\vec{e_x}, \quad (14)$$

$$R_y^{e\prime}\vec{e_y} := R_y^{e\prime}\vec{e_y} + \eta[\gamma d \log O_{loc}(e_x, e_y) + (1 - \gamma) d \log O_{cate}(e_x, e_y)]/dR_y^{e\prime}\vec{e_y}, \quad (15)$$

where η is the learning rate.

4.4 Representation Learning for ISRN2Vec

Please note that RN2Vec learns embeddings for intersections and road segments jointly. As IRN2Vec and SRN2Vec target on intersections and road segments independently,[10] there is no interactions between them. Thus, in addition to explore the geo-locality relationships between a pair of intersection and road segment, ISRN2Vec plays an essential and critical role to bridge IRN2Vec and SRN2Vec in RN2Vec, by sharing the learning parameters, i.e., intersection embeddings and road segment embeddings, with IRN2Vec and SRN2Vec.

Similarly, ISRN2Vec is a binary-task classifier that takes an intersection $v \in V$ and a road segment $e \in E$ as the inputs to predict their geo-locality, i.e., located within a certain distance. More specifically, the input layer takes in two one-hot vector $\vec{v}$ and $\vec{e}$ of length $|V|$ and $|E|$, respectively, representing node v and edge e. In the latent layer, $\vec{v}$ and $\vec{e}$ are transformed into latent vectors $R^{v\prime}\vec{v}$ and $R^{e\prime}\vec{e}$, where R^v and R^e are $|V| \times d$ and $|E| \times d$ matrices representing the transformation, $R^{v\prime}$ and $R^{e\prime}$ are their transpose matrices, and d is the dimensionality of the hidden space. Next, we use inner product followed by a *Sigmoid* function to predict (1) the probability $P_{loc}(v, e)$ for v and e to be located within a certain distance; and (2) the probability $P_{DC}(v, e)$ for v and e to be directly connected. As mentioned, in ISRN2Vec, the matrix R^v and R^e, consisting of embeddings of all intersections and road segments, are identical to R_x^v (= R_y^v) of IRN2Vec and R_x^e (= R_y^e) of SRN2Vec, respectively, which effectively connect all sub-modules of RN2Vec.

[10] Actually, IRN2Vec and SRN2Vec may be used as standalone models to learn intersection embeddings and road segment embeddings separately by using $O_{IRN2Vec}$ and $O_{SRN2Vec}$ as the training objective, respectively.

To train the ISRN2Vec, a training dataset D_{IS} that contains training data entries in the form of $\langle v, e, S_{loc}(v,e), S_{DC}(v,e)\rangle$ is extracted from the road network. In a training data entry, $S_{loc}(v,e)$ and $S_{DC}(v,e)$ are Boolean values, indicating whether node v and edge e are located within k-meter and whether directly connected in road networks. Accordingly, ISRN2Vec, trained by the backpropagation training algorithm in conjunction with asynchronous stochastic gradient ascent, goes backward to adjust the weights in R^v and R^e, attempting to maximize the overall objective function O_{RN2Vec} in Equation (1). In ISRN2Vec, the objectives $O_{loc}(v,e)$ and $O_{DC}(v,e)$ quantify how $S_{loc}(v,e)$ and $S_{DC}(v,e)$ in a data entry are correctly predicted, respectively, are derived as follows:

$$O_{loc}(v,e) = \begin{cases} P_{loc}(v,e) & \text{if} \quad S_{loc}(v,e) = 1 \\ 1 - P_{loc}(v,e) & \text{if} \quad S_{loc}(v,e) = 0 \end{cases}; \quad O_{DC}(v,e) = \begin{cases} P_{DC}(v,e) & \text{if } S_{DC}(v,e) = 1 \\ 1 - P_{DC}(v,e) & \text{if } S_{DC}(v,e) = 0 \end{cases}.$$

To ease the computation, we adopt $\log O_{loc}(v,e)$ and $\log O_{DC}(v,e)$ as follows:

$$\log O_{loc}(v,e) = S_{loc}(v,e) \log P_{loc}(v,e) + [1 - S_{loc}(v,e)] \log[1 - P_{loc}(v,e)], \tag{16}$$

$$\log O_{DC}(v,e) = S_{DC}(v,e) \log P_{DC}(v,e) + [1 - S_{DC}(v,e)] \log[1 - P_{DC}(v,e)]. \tag{17}$$

The objective function $O_{ISRN2Vec}$ for ISRN2Vec is defined as follows:

$$O_{ISRN2Vec} = \sum_{s_{IS} \subseteq D_{IS}} \Big\{ \omega \log O_{loc}(v,e) + (1-\omega) \log O_{DC}(v,e) \Big\}, \tag{18}$$

where ω is a weighting parameter.

During training, we apply stochastic gradient ascent to update $R^{v\prime}\vec{v}$ and $R^{e\prime}\vec{e}$ based on the gradients derived as follows:

$$R^{v\prime}\vec{v} := R^{v\prime}\vec{v} + \eta[\omega d \log O_{loc}(v,e) + (1-\omega) d \log O_{loc}(v,e)]/dR^{v\prime}\vec{v}, \tag{19}$$

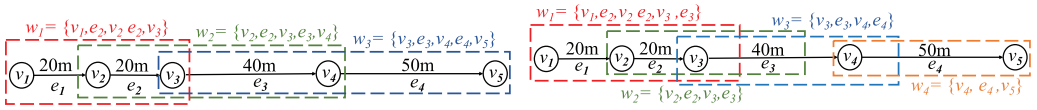
$$R^{e\prime}\vec{e} := R^{e\prime}\vec{e} + \eta[\omega d \log O_{DC}(v,e) + (1-\omega) d \log O_{DC}(v,e)]/dR^{e\prime}\vec{e}, \tag{20}$$

where η is the learning rate.

4.5 Training Data Preparation

To meet the need of model training, in Phase I of RLRN, we sample pairs of elements (i.e., intersections and road segments) in the road network to prepare training data for RN2Vec. Our design decision represents a tradeoff in data collection between the computational efficiency (i.e., sampling instead of enumeration) and the quality (i.e., the training data should cover as many intersections and road segments as possible). Conventionally, network representation learning techniques adopt random walks to sample the network structure. However, we argue that random walks-based sampling is not suitable for road networks, because *mobile road users do not move randomly*. Therefore, we adopt shortest path-based scheme to sample the road network, which is more aligned to the moving behaviors of mobile road users. Specifically, we apply Dijkstra algorithm to find the shortest paths between randomly selected pairs of road network elements.

After generating a set of shortest paths, we employ sliding windows on each path to prepare training data entries for RN2Vec in the form of $s_I = \langle v_x, v_y, S_{loc}(v_x, v_y), S_{tag}(v_x, v_y), S_{type}(v_x, v_y)\rangle$, $s_S = \langle e_x, e_y, S_{loc}(e_x, e_y), S_{tag}(e_x, e_y)\rangle$ and $s_{IS} = \langle v, e, S_{loc}(v,e), S_{DC}(v,e)\rangle$ to generate three datasets, D_I, D_S, and D_{IS}. More specifically, along a sampled shortest path, represented as a sequence of interleaving intersections and road segments (i.e., $S_R = \{v_1, e_1, v_2, e_2, \ldots, v_{|S_R|}\}$, we examine the notions of *hop-based neighborhood* and *distance-based neighborhood* to define the geo-locality between road network elements. Conventionally, network embedding techniques typically adopt hop-based neighborhood, which does not capture the geo-spatial characteristics of road networks very well. Considering an example of road network sequence $S_{RN} = \{v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_5\}$



(a) Hop-based Neighborhood: 2-hop (b) Distance-based Neighborhood: 50m

Fig. 5. An example of different sliding window size in the RN2Vec model.

(see Definition 3), sampled via shortest path between v_1 and v_5, where a sliding window of 2-hop is used. As shown in Figure 5(a), we have three 2-hop neighborhood windows $w_1 = \{v_1, e_1, v_2, e_2, v_3\}$, $w_2 = \{v_2, e_2, v_3, e_3, v_4\}$ and $w_3 = \{v_3, e_3, v_4, e_4, v_5\}$. As shown in Figure 5(b), intersections v_3 and v_5 are distant but they are treated as close to each other by a 2-hop window. Generally speaking, near things are more related than distant things. Therefore, in RN2Vec, we explore *distance-based neighborhood*. For the same road network sequence in Figure 5, where a 50m window is used to sample along the shortest path, resulting in $w_1 = \{v_1, e_1, v_2, e_2, v_3, e_3\}$, $w_2 = \{v_2, e_2, v_3, e_3\}$, $w_3 = \{v_3, e_3, v_4, e_4\}$, and $w_4 = \{v_4, e_4, v_5\}$.

Accordingly, for each pair of road network elements (i.e., (v_x, v_y), (e_x, e_y) or (v, e)) within a window, we create a positive sample reflecting their geo-locality and homogeneity. For instance, for the first window $w_1 = \{v_1, e_1, v_2, e_2, v_3, e_3\}$, we create positive training samples $\langle v_1, e_1, 1, same_{DC}(v_1, e_1)\rangle$, $\langle v_1, v_2, 1, same_{tag}(v_1, v_2), same_{type}(v_1, v_2)\rangle$, $\langle v_1, e_2, 1, same_{DC}(v_1, e_2)\rangle$, $\langle v_1, v_3, 1, same_{tag}(v_1, v_2), same_{type}(v_1, v_2)\rangle$, $\langle v_1, e_3, 1, same_{DC}(v_1, e_2)\rangle$, $\langle e_1, v_2, 1, same_{DC}(e_1, v_2)\rangle$, and so on, where 1 indicates the positive locality between the pair of network elements, $same_{DC}(v, e)$ or $same_{DC}(e, v)$ are further determined by checking whether v and e are directly connected; $same_{tag}(v_x, v_y)$ and $same_{type}(v_x, v_y)$ are determined by checking whether v_x and v_y have the same intersection tag and same N-way type, respectively; $same_{cate}(e_x, e_y)$ is determined by checking whether e_x and e_y have the same road category. In addition to the positive training samples, the RN2Vec model also needs negative data samples for learning. Thus, we generate negative data samples following the idea of *Negative Sampling* [16], replacing the second element by randomly selecting another element.

In our study, we show that the shortest path-based sampling approach is better than the conventional random walk-based network sampling. Nevertheless, we argue that using real-world trajectory data, which captures the moving behaviors of mobile road users, to sample the road network is better than existing approaches that randomly generates samples. However, publicly available trajectory datasets do not cover the whole corresponding road networks, i.e., not all the intersections and road segments in the road networks are visited by the collected trajectories. Thus, in this article, we supplement shortest path sampling with real-world trajectories for model training.

5 PERFORMANCE EVALUATION

In this section, we conduct extensive experiments to evaluate RN2Vec by using multiple real-world road network datasets and five downstream applications, including *intersections classification*, *road segments classification* as well as *travel time estimation* of travel paths on road networks for all these models. For comparison, we include three categories of existing methods, including (i) Feature-based methods, (ii) Network embedding methods, and (iii) Features + Network structure-based methods in our evaluation. We also perform parameter tuning on RN2Vec and its sub-modules, examine several issues and demonstrate the generality and robustness of RLRN framework. Finally, we visualize and compare embeddings generated by RN2Vec and some selected models.

Table 2. Statistics of Road Network Datasets

Name	San Francisco	Porto	Tokyo	Seattle	Chicago
#Intersections	58,404	119,769	217,117	200,213	261,727
#Road segments	76,744	154,128	305,874	247,184	377,973

Table 3. Statistics of Trajectory Datasets

Name	#GPS Points	#Trajectories	Avg. Time Gap
San Francisco	11,219,955	443,406	14.12 s
Porto	74,269,739	1,233,766	15.11 s
Tokyo	68,275,641	273,046	15.00 s

5.1 Datasets and Compared Methods

Our evaluation involves two types of real-world datasets, i.e., road network datasets and trajectory datasets. Specifically, road networks are fed for training/testing of representation learning models, while some tags/categories of nodes/edges (i.e., the ground truth) are used for classification tasks. Meanwhile, trajectory datasets are used to provide the travel paths and their corresponding travel times (i.e., the ground truth) for travel time estimation.

5.1.1 Road Network Data. We extract five road networks, including San Francisco, Porto, Tokyo, Seattle and Chicago, from the raw network data downloaded from OpenStreetMap website [4]. Note that in road segment classification tasks, we use additional road segment types from TIGER data [2] including avenue, drive, boulevard, street, court, and so on, which are not part of road segment categories in the OSM data. As the TIGER data, produced by the US Census Bureau, only contains information on US road networks, we use San Francisco, Seattle and Chicago road networks in classification tasks for evaluation of road segment embeddings. Some statistics of the road networks extracted are summarized in Table 2.

5.1.2 Trajectory Data. We collect three publicly accessible trajectory datasets as follows. Some statistics of these trajectory datasets are summarized in Table 3.

San Francisco [22]. This dataset collects 11 million GPS points from 536 taxis running in San Francisco for a 30-day period, and every taxi has two statuses (occupied or not). We select the sequences of GPS points by occupied taxis to form trajectories and remove trajectories with fewer than 5 sample points, which yields 0.4 million trajectories.

Porto [3]. This dataset collects 1.7 million trajectories (containing 74 million GPS sample points) from 442 taxis running in Porto City over a complete year. Each taxi reports its location every 15 second. We remove trajectories with fewer than 10 sample points to yield 1.23 million trajectories.

Tokyo [34]. This dataset collects 68 million GPS sample points from 617,040 users in Tokyo taking different vehicles, such as bike, train, and car. We consider sequences of GPS sample points by users taking bicycle and bus and segment them into trajectories when there is no sample point for 45 s or more (which is longer than about 99% time gaps). Then, we remove trajectories with fewer than 5 sample points, which yields 0.27 million trajectories.

5.1.3 Compared Methods. We use Unique ID as a baseline for our evaluation. In addition, we examine three categories of existing methods, including (i) Feature-based methods: raw features and PCA; (ii) Network embedding methods: DeepWalk, LINE, and Node2vec; and (iii) Features + Network structure-based methods: network embeddings and PCA, graph convolutional networks, and graph attention networks.

Unique ID (UID) uses one-hot vectors as unique IDs of intersections and road segments in a network. Without learning embeddings, this method serves as a baseline.

Raw Features (RF) directly uses intersection tags and road types as feature vectors of intersection and uses road segment categories as feature vectors of road segments.

PCA linearly transforms the raw features of intersections and road segments into a new space where the top k (i.e., $k = 4$) components are used as embeddings.

DeepWalk [21] learns d-dimensional node vectors by exploring the relationship of nodes within w-hop neighborhood via uniform random walks in the network.

LINE [24] learns d-dimensional node vectors by by capturing the first-order proximity in d/2 dimensions and the second-order proximity on the other dimensions.

Node2vec [8] is generalized from DeepWalk. It learns d-dimensional node vectors by capturing node pairs within w-hop neighborhood via parameterized random walks in the network.

Network embedding and PCA (NE+PCA) concatenates the best network embeddings in experiments and the PCA embeddings to form combined embeddings.

GCN [12] learns embeddings of intersections/road segments in a road network by aggregating information from their neighborhoods. Note that GCN is not designed specifically for representation learning. To compare GCN as a general network embedding method, we train a GCN model for a classification task not targeted in our downstream applications (e.g., turning circles classification for intersection embedding and boulevard classification for road segment embedding), then we extract the output of the last layer of the GCN model as the embeddings for evaluation (denoted as GCN_E). However, to observe how embedding methods fare with the best performance of GCN, we also directly train an end-to-end GCN model (denoted as GCN_A) for each of the targeted classification tasks for comparison.

GAT [26] adopts the attention mechanism to learn the weights between two connected nodes for information aggregation. Similarly to GCN, we also consider two GAT variants, GAT_E and GAT_A, in the classification tasks.

Notice that although several existing works are proposed for learning road segment embeddings, they simply apply network embedding methods (e.g., DeepWalk and Node2vec). Thus, we compare DeepWalk, LINE, and Node2vec instead of them for evaluation. RN2Vec is implemented in C. All experiments are run on the Ubuntu 18.04 operating system with an Intel Core i5-8400 CPU.

5.2 Parameter Tuning and Unique Issues

In this section, we tune the parameter settings on the sub-modules (i.e., IRN2Vec, SRN2Vec, and ISRN2Vec) and RN2Vec. Then we study unique issues arise in RN2Vec.

5.2.1 IRN2Vec. To study the impact of parameter setting on IRN2Vec, we vary the values of important parameters to observe how the F1-score changes in traffic signal classification.

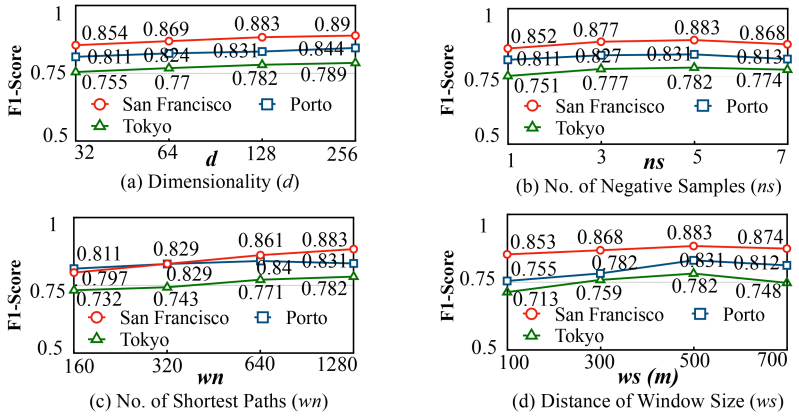
After learning the intersection embeddings,[11] we use the 2,582, 1,646, and 8,470 intersections with the "traffic signal" tag in the San Francisco, Porto, and Tokyo road networks as positive samples. Then, we randomly add equal number of negative samples (i.e., intersections without the "traffic signal" tag) to form three balanced labeled datasets, which are randomly split as the training set (90%) and test set (10%). We use the training set to train a classifier by using linear SVM and evaluate the performance using the test set. In Figure 6, we report the F1-score of these sensitivity tests.

Dimensionality (d). Generally speaking, a small dimensionality is not sufficient to capture the various relationships between intersections, but too large a dimensionality may lead to overfitting. Figure 6(a) shows that setting d at 128 for IRN2Vec is reasonable.

No. of Negative Samples (ns). We test various number of negative samples per positive sample. As shown in Figure 6(b), the best performance are achieved when ns is set to 5 in datasets.

No. of Shortest Paths Per Node (wn). Generally speaking, more shortest paths generate more training data, leading to better performance in downstreaming tasks. Figure 6(c) suggests that the performance continues to improve when wn increases and it converges at $wn = 1280$.

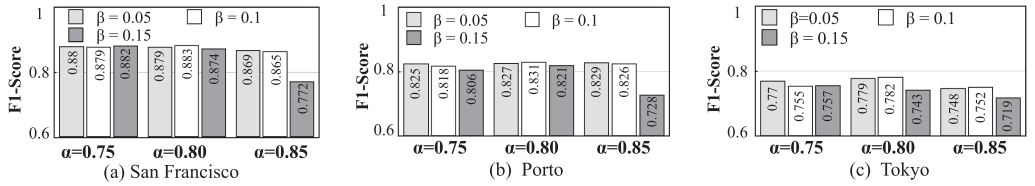
[11] We remove all "traffic signal" tags while learning intersection embeddings.



(a) Dimensionality (d) (b) No. of Negative Samples (ns)

(c) No. of Shortest Paths (wn) (d) Distance of Window Size (ws)

Fig. 6. Parameter analysis for IRN2Vec.



(a) San Francisco (b) Porto (c) Tokyo

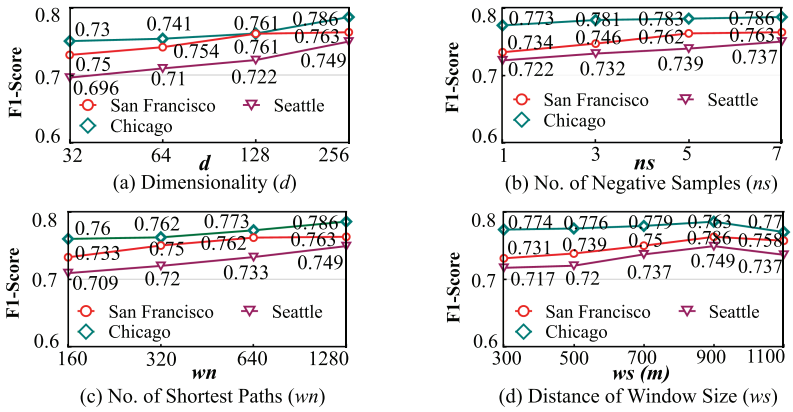
Fig. 7. Evaluation of α and β for IRN2Vec.

Distance of Window Size (ws). Figure 6(d) shows that the performance improves while ws increases from 100 to 700 m (with a step of 200 m). The best performance is achieved at ws = 500m.

Weights of α and β. Finally, we evaluate the weights α, β, and $1-\alpha-\beta$ corresponding to various objectives of IRN2Vec, i.e., $O_{loc}(v_x, v_y)$, $O_{tag}(v_x, v_y)$, and $O_{type}(v_x, v_y)$ in Equation (8). We perform grid search to tune the performance. As shown in Figure 7, the performance of IRN2Vec in different datasets does not change much when α is set between 0.75 to 0.8 and β is set to between 0.05 to 0.1. Consistently, the best performance is achieved when $\alpha = 0.8$ and $\beta = 0.1$. Note that when $\alpha = 0.85$ and $\beta = 0.15$ (i.e., it does not consider the same-type relationship between intersections in IRN2Vec), the experiments have the worst results, which suggests that the same-type relationship between intersections is quite useful for representation learning and traffic signal classification.

5.2.2 SRN2Vec. Next, we study the impact of important parameters on the performance of SRN2Vec in avenue classification, which infers whether a road segment has an avenue tag on it or not. We use the 11.2k, 22.2k, and 68.7k road segments tagged as "avenue" in Tiger data on San Francisco, Seattle, and Chicago road networks, respectively, as positive samples and randomly select equal number of non-avenue road segments as negative samples to form labeled datasets. For each dataset, we randomly split it into 90% and 10% as the training set and test set, respectively. We use the training set to train a linear SVM model as a binary classifier and evaluate its performance using the test set. We report the *F1-score*.

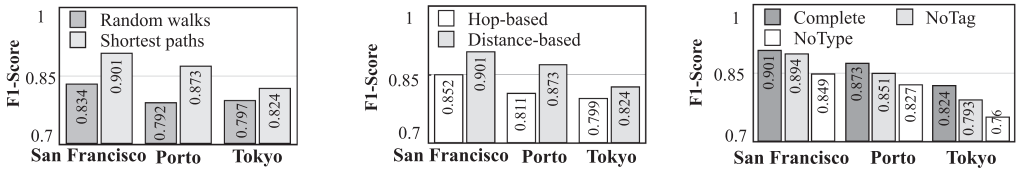
As shown in Figure 8, SRN2Vec performs the best when the dimensionality d is set to 256; the number of negative samples per positive sample ns is set to 7; the number of shortest paths sampled per road segment wn is set to 1280; the window size ws is set to 900 m. Moreover, we test the parameter γ and $1-\gamma$, the weighs for $\log O_{loc}(e_x, e_y)$, and $\log O_{cate}(e_x, e_y)$ in Equation (13). The best performance is achieved when the γ is set to 0.8.



(a) Dimensionality (d) (b) No. of Negative Samples (ns)

(c) No. of Shortest Paths (wn) (d) Distance of Window Size (ws)

Fig. 8. Parameter analysis for SRN2Vec.



(a) Sampling Schemes of Training Data (b) Definition of Window Size (c) Intersection Tags and N-way Types

Fig. 9. Unique issues in traffic signal classification.

5.2.3 ISRN2Vec. By using the best parameter settings for IRN2Vec and SRN2Vec, we tune ω for weighing $\log O_{loc}(v, e)$ in ISRN2Vec (i.e., Equation (18)) on both traffic signal classification and avenue classification tasks. The tuning is based on the overall RN2Vec performance as ISRN2Vec does not work alone like IRN2Vec and SRN2Vec. Due to space limit, we do not plot the experimental result. The performance is the best when ω is set to 0.75.

5.2.4 RN2Vec. Finally, we perform grid search to tune the performance of λ, μ, and $1 - \lambda - \mu$, which are the weights for $O_{ISRN2Vec}$, $O_{IRN2Vec}$, and $O_{SRN2Vec}$ in Equation (1). Based on the results, the weights of λ and μ are set to 0.6 and 0.2. In summary, in RN2Vec, the dimensionality of all embeddings is set to 256. The number of negative samples per positive sample is set to 7. The distance of window size is set to 900m, the number of shortest paths sampled is set to 1280, the weights of α, β, γ, ω, λ, and μ are set to 0.8, 0.1, 0.8, 0.75, 0.6, and 0.2.

5.2.5 Study of Unique Issues. As discussed, several unique issues arise in the design of RN2Vec. In this section, we perform experiments on traffic signal classification to examine the following issues: (1) Sampling schemes for training data, (2) distance-based vs. hop-based neighborhoods, and (3) ablation study on tags and N-way types.

To validate our argument that `shortest paths` better reflect the moving behaviors of mobile road users than `random walks`, we compare the proposed RN2Vec[12] with a variant that uses training data prepared by random walk sampling. As shown in Figure 9(a), `shortest paths` outperforms `random walks` by 3.38% to 10.23% in traffic signal classification, which suggests that using shortest paths sampling in RN2Vec is significantly more effective than using random walks, because mobile users naturally follow the more economic shortest paths while moving on roads.

[12]In this article, shortest path is considered by default as the data sampling scheme of the proposed RN2Vec.

To validate our idea of adopting distance to capture geo-locality in road networks, we compare the proposed `distance-based neighborhood` (with ws = 900 m) for road networks representation learning against the `hop-based neighborhood` (with hop number k = 5), which is widely used in conventional network representation learning. Figure 9(b) shows that `distance-based neighborhood` outperforms `hop-based neighborhood` in all three datasets (improving by 3.13% to 7.64%), which suggests that `distance-based neighborhood`, capturing the geo-spatial characteristics of the road network, is a more natural choice for road network representation learning.

Finally, as we separate two homogeneity relationships in IRN2Vec (due to the nature of data sources), we perform an ablation study to find the impact of same-tag and same-type relationships between intersections on the performance of the proposed RN2Vec. In this study, we consider three variants of RN2Vec: (1) `Complete` is the complete version with all factors considered; (2) `NoTag` does not consider the same-tag relationship in IRN2Vec; and (3) `NoType` does not consider the same-type relationship. Figure 9(c) shows that `Complete` outperforms `NoTag` and `NoType` for about 0.78% to 8.24% in the three datasets, which suggests that taking both tags and N-way types of intersections into account are beneficial for RN2Vec to learn better intersection representations.

In Figure 9(c), we observe that `NoType`, which uses tags but not intersection types, performs the worst in all datasets. This means the N-way intersection types we derived is more informative than the tags generated by OSM volunteers. This may be due to the incomplete/inaccurate tag information associated with intersections in the OSM data. We dig further into this issue and find significant amount of missing tags in open street maps. For example, there are only 243 stop signs in San Francisco road network, which is unreasonably few. To investigate this issue and to validate our idea of using tags in learning, we collect additional 3,727 stop signs from the official website of San Francisco government [13] as supplementary stop sign tags of intersections in the San Francisco road network for representation learning. The new result is improved for about 2.94% compared with the old result using only stop sign tags from OSM dataset. This supports our argument that the information of intersection tags, if available and more accurate, is indeed useful for representation learning. Thus, the RN2Vec embeddings have room to get better as the volunteer-generated tags grow or official/proprietary geographical information describing the road networks is used.

5.3 Intersection Classification

In this section, we evaluate the intersection embeddings generated by RN2Vec by two intersection classification tasks (i.e., traffic signal classification and crossing classification) on three real-world road network datasets (i.e., the San Francisco, Porto, and Tokyo datasets). They are both binary classification tasks that infer whether an intersection has a traffic signal tag and has a crossing tag or not, respectively. We follow the same experimental setup mentioned in Section 5.2.1 and also report the F1-score for comparison.

The parameter configurations of compared methods are tuned experimentally. The dimensionality of raw feature, PCA and NE+PCA are set to 14, 4, and 132, respectively. In DeepWalk and Node2vec, we set the window size w = 5, the number of walks = 20 and the walk length = 1,280. For Node2vec, we set the two parameters p = 1 and q = 4 for parameterized random walks. For GCN_A and GCN_E the number of hidden layers is set to 2 followed by a fully connected layer. The dimensionality and learning rate are set to 128 and 0.01. For GAT_A and GAT_E, the dimensionality is 128 (i.e., 4 attention heads and the embedding size = 32 for each head) and the learning rate is set to 0.005.

Evaluation by Traffic Signal Classification. The performance of all evaluated models on traffic signal classification is reported in Table 4. Please note that RF, GCN_A, and GAT_A are application-specific methods, which do not use embeddings generated for general support of

Table 4. Performance of Traffic Signal Classification

	San Francisco	Porto	Tokyo
UID	0.532	0.511	0.505
RF	0.751	0.746	0.729
GCN_A	0.789	0.798	0.755
GAT_A	0.801	0.812	0.769
PCA	0.733	0.709	0.701
DeepWalk	0.711	0.661	0.639
LINE	0.573	0.553	0.557
Node2vec	0.713	0.680	0.624
NE+PCA	0.771*	0.760*	0.736*
GCN_E	0.695	0.679	0.652
GAT_E	0.711	0.680	0.641
IRN2Vec	0.883	0.831	0.782
RN2Vec	**0.901 (16.86%)**	**0.873 (14.87%)**	**0.824 (11.96%)**

Table 5. Performance of Crossing Classification

	San Francisco	Porto	Tokyo
UID	0.527	0.516	0.509
RF	0.739	0.672	0.717
GCN_A	0.770	0.712	0.775
GAT_A	0.798	0.721	0.783
PCA	0.719	0.650	0.741
DeepWalk	0.692	0.635	0.718
LINE	0.554	0.567	0.577
Node2vec	0.717	0.644	0.734
NE+PCA	0.757*	0.684*	0.769*
GCN_E	0.703	0.638	0.717
GAT_E	0.698	0.631	0.729
IRN2Vec	0.779	0.719	0.800
RN2Vec	**0.843 (11.36%)**	**0.798 (16.67%)**	**0.869 (13.00%)**

multiple applications. While the goal of this evaluation is to compare against the state-of-the-art embedding learning methods, we include them as references to see how the general RN2Vec method fare with those specialized approaches.

As shown, RN2Vec soundly outperforms all the compared models. The improvement ratio (compared with the best of the embedding methods, marked by $*$) ranges from 11.96% to 16.86% in three datasets. Based on the results, we have the following observations. (1) UID, without exploring features or relationships in road networks, has the worst performance. (2) The feature-based methods achieve better performance than conventional network embedding methods learned by encoding network structure information, which prove that the properties (features) of intersections are very useful, especially in classification tasks where features are even more important than structural information. (3) Among network embedding methods, while DeepWalk and Node2vec are better than LINE, they are still significantly inferior to IRN2Vec and RN2Vec. It may suggest that LINE, only capturing the information of 1-hop or 2-hop neighborhood of nodes in networks, is not suitable for road network representation learning, which has a broader neighborhood and more general relationships. Moreover, all DeepWalk, LINE, and Node2Vec, only capturing network structural information, are insufficient for road network representation learning. (4) NE+PCA, aiming to combine features and structural information in embeddings, is effective. It is consistently the best among all valid embedding methods but still significantly inferior to IRN2Vec and RN2Vec. GCN_E and GAT_E, supposed to encode both feature and structural information as well, do not perform as well as NE+PCA, probably because they are geared toward their training target too much instead of being able to generalize for general embedding learning. On the contrary, RN2Vec is able to explore structural information, various relationships on road networks, distance-based neighborhood, and shortest path sampling to achieve excellent performance. (5) RN2Vec and IRN2Vec perform convincingly better than specialized application solutions (i.e., RF, GCN_A, and GAT_A), demonstrating the feasibility of general-purpose intersection embeddings. (6) RN2Vec outperforms IRN2Vec, which suggests that learning embeddings of intersections and road segments jointly is more effective than learning intersection embeddings alone as IRN2Vec does.

Evaluation by Crossing Classification. To demonstrate the robustness of RN2Vec for intersection embeddings, we evaluate the models by performing an alternative intersection classification task that infers whether an intersection in a road network has a crossing on it. The experimental setup is the same as that in Section 5.2.1, except for having the "crossing" tags as positive

samples (5,021, 6,934, and 14,377 intersections in San Francisco, Porto, and Tokyo datasets, respectively.) The performance of all evaluated models on the task of crossing classification is reported in Table 5. As shown, RN2Vec robustly outperforms all the compared methods, with improvement ratios over the competing embedding methods ranging from 11.36% to 16.67% among the three datasets.

5.4 Road Segment Classification

In this section, we evaluate RN2Vec by two road segment classification tasks (i.e., avenue classification and street classification) on three real-world road networks (i.e., San Francisco, Chicago, and Seattle).[13] In experiments, we also evaluate road segments embeddings generated by intersection embeddings, IRN2Vec$_H$, by applying Hadamard function[14] on intersection embeddings generated by IRN2Vec.

The parameters of compared methods are tuned experimentally. The dimensionality of raw feature, PCA and NE+PCA are set to 13, 4, and 260, respectively. In DeepWalk and Node2vec, we set the window size $w = 5$; the number of walks = 20 and the walk length = 1,280. For Node2vec, the two parameters p and q for parameterized random walks are set to 1 and 5, respectively. For GCN_A and GCN_E, the number of hidden layers is set to 2 followed by a fully connected layer. The dimensionality and learning rate are set to 256 and 0.01. For GAT_A and GAT_E, the dimensionality is set to 256 (i.e., 4 attention heads and an embedding size = 64 for each attention head) and the learning rate is both set to 0.005.

Evaluation by Avenue Classification. The performance of all evaluated methods on avenue classification is reported in Table 6. We observe that all the embedding-based methods including IRN2Vec$_H$ outperform UID. Among embedding methods, IRN2Vec$_H$ performs the worst, suggesting that capturing information associated with road segments is better than simply aggregating information in intersection embeddings. Other observations are consistent with our findings from evaluation by traffic signal classification. Again, the proposed RN2Vec soundly outperforms all the compared methods, including specialized GCN_A and GAT_A. The improvement ratio (compared with the best of these embedding models, marked by $*$) ranges from 10.56% to 15.43% in the three datasets. Also, the road segment embedding learned by RN2Vec outperforms those learned by SRN2Vec, i.e., jointly learning embeddings of intersections and road segments in road networks is beneficial.

Evaluation by Street Classification. To demonstrate the robustness of road segment embeddings learned by RN2Vec, we evaluate the road segment embeddings by compared methods to infer whether a road segment is tagged as "street" in Tiger data. The experimental setup is the same with avenue classification, except for using road segments with the "Street" tag (there are 14,145, 19,803, and 55,796 such road segments in San Francisco, Seattle, and Chicago datasets, respectively). The result of evaluation is reported in Table 7. As shown, SRN2Vec and RN2Vec robustly outperform all the compared methods, with improvement ratios (compared with the best of these embedding models, marked by $*$) ranging from 11.54% to 16.07% in these three datasets.

5.5 Travel Time Estimation

In addition to classification tasks, we further evaluate our proposed RN2Vec model on an application of different type, *travel time estimation* for paths in road networks. In the following, we

[13] Tiger dataset do not cover Porto and Tokyo so we extract Seattle and Chicago road network for the experiments instead. Those binary classification tasks infer whether a road segment has an avenue/street tags or not. We follow the same experimental setup mentioned in Section 5.2.2.

[14] Average and Addition are also tested but Hadamard outperforms them.

Table 6. Performance of Avenue Classification

	San Francisco	Seattle	Chicago
UID	0.507	0.537	0.520
IRN2Vec$_H$	0.525	0.554	0.539
RF	0.690	0.667	0.719
GCN_A	0.748	0.724	0.749
GAT_A	0.757	0.739	0.766
PCA	0.667	0.643	0.702
DeepWalk	0.642	0.613	0.693
LINE	0.592	0.566	0.547
Node2vec	0.631	0.628	0.664
NE+PCA	0.700*	0.669*	0.720*
GCN_E	0.650	0.639	0.691
GAT_E	0.649	0.631	0.682
SRN2Vec	0.763	0.749	0.768
RN2Vec	**0.808 (15.43%)**	**0.752 (12.41%)**	**0.796(10.56%)**

Table 7. Performance of Street Classification

	San Francisco	Seattle	Chicago
UID	0.541	0.560	0.524
IRN2Vec$_H$	0.563	0.580	0.544
RF	0.738	0.743	0.751
GCN_A	0.751	0.758	0.743
GAT_A	0.762	0.769	0.752
PCA	0.718	0.712	0.720
DeepWalk	0.707	0.698	0.709
LINE	0.516	0.540	0.537
Node2vec	0.711	0.718	0.699
NE+PCA	0.745*	0.754*	0.759*
GCN_E	0.703	0.709	0.698
GAT_E	0.689	0.697	0.714
SRN2Vec	0.785	0.802	0.797
RN2Vec	**0.831 (11.54%)**	**0.854(13.23%)**	**0.881(16.07%)**

first introduce the experimental setup and then report the experimental results obtained by using intersection/road segment embeddings learned by RN2Vec, its variants, and other methods.

5.5.1 Experimental Setup. Travel time estimation is a regression task that predicts the travel time of a given moving path in road networks. Publicly available trajectory datasets, including San Francisco, Porto, and Tokyo, are processed for preparation of training and used as ground truth for testing.[15] To avoid the learned regression model simply counting the number of sample points in a trajectory to exploit the relatively fixed time gap between consecutive sample points for travel time estimation, we do not directly use raw trajectories for travel time estimation. Instead, we adopt Barefoot [15], a state-of-the-art road network matching technique, to map raw trajectories onto their underlying road network to obtain sequences of intersections or road segments (which depict corresponding moving paths of the trajectories in the road network). Thus, this task aims to estimate the travel time of a given moving path (i.e., as a sequence of intersections or a sequence of road segments). To achieve the goal, we apply a Long Short-Term Memory (LSTM) model followed by three fully connected layers (with dimensions 256, 256, and 1, respectively) as a regression model. More specifically, given a travel path as the input, the LSTM model takes the embedding of each intersection or road segment in the travel path as a state to estimate its travel time.

We use mean absolute error (MAE) between the predicted result and the ground truth (in seconds) as the metric to evaluate the travel time estimation. In the experiments, we adopt the best embeddings obtained previously in the classification tasks for each method.

5.5.2 Evaluation of Models. The results of travel time estimation obtained using intersection embeddings and road segment embeddings learned by the evaluated models are reported in Table 8 and Table 9, respectively. As shown, RN2Vec robustly outperforms all the compared methods by 11.24% to 18.52% in MAE compared with the best among existing works (marked by *) on the three datasets. From the results, we have the following observations. (1) As shown, all the embedding-based methods outperform the UID method, which validates the use of general embeddings (for both intersections and road segments) for travel time estimation. (2) The results show that the feature-based methods (i.e., RF and PCA) do not perform well, because they only explore features

[15]While raw trajectory data contains the arrival times at GPS points, they are not used for training and testing.

Table 8. Performance of Intersection Embedding in Travel Time Estimation

	San Francisco	Porto	Tokyo
UID	75.62	169.49	106.39
RF	67.33	164.07	100.75
PCA	69.09	167.22	103.12
DeepWalk	64.97	163.17	95.74
LINE	67.05	161.99	99.86
Node2vec	61.37	160.65	96.31
NE+PCA	58.87*	157.66*	92.25*
GCN_E	63.25	159.79	94.26
GAT_E	62.87	161.16	96.89
IRN2Vec	49.54	141.96	86.29
RN2Vec	**47.97 (−18.52%)**	**137.99 (−12.48%)**	**81.88 (−11.24%)**
RN2Vec$_{RN}$	**44.99 (−23.58%)**	**125.03 (−20.70%)**	**76.28 (−17.31%)**

Table 9. Performance of Road Segment Embedding in Travel Time Estimation

	San Francisco	Porto	Tokyo
UID	80.61	168.37	107.64
RF	69.33	164.79	98.28
PCA	73.09	167.85	100.56
DeepWalk	75.32	158.69	99.57
LINE	68.65	167.79	104.66
Node2vec	62.97	163.13	93.18
NE+PCA	60.25*	152.69*	91.92*
GCN_E	64.18	161.59	97.78
GAT_E	63.95	163.33	96.05
SRN2Vec	51.05	137.28	85.91
RN2Vec	**46.89 (−22.17%)**	**133.64 (−12.48%)**	**82.86 (−9.86%)**
RN2Vec$_{RN}$	**44.99 (−25.33%)**	**125.03 (−18.12%)**	**76.28 (−17.01%)**

but not network structural information. (3) Compared with existing network embedding methods (i.e., DeepWalk, LINE, and Node2vec), NE+PCA achieves better performance, because it not only contains the intersection or road segment features but also the network structural information. (4) While GCN_E and GAT_E are supposed to capture both features and structural information, they do not perform as good as NE+PCA and RN2Vec. This observation may suggest that although GCN/GAT methods contain node/edge features and structural information, their ways of integrating feature propagation and graph convolution in terms of hop-based neighborhood may optimize their models towards the training target instead of capturing inherent structural properties of the network as the representation learning models do. As a result, network embeddings coupled with feature embeddings (i.e., NE+PCA) performs better. (5) For intersection embeddings (see Table 8), the result shows that RN2Vec outperforms all the compared methods by 11.24% to 18.52% in MAE compared with the best among existing works (marked by *) on the three datasets. This demonstrates that RN2Vec is able to capture the intrinsic properties of the road networks and thus achieve better performance. Moreover, RN2Vec is better than IRN2Vec by 1.39% to 6.84%, suggesting that capturing the various relationships among intersections and road segments is particularly useful. Similarly, as shown in Table 9, the road segment embeddings learned by SRN2Vec outperforms all the existing models by 6.54% to 15.25%; and RN2Vec is even more impressive, outperforming them by 9.86% to 22.17%. (6) It provides new evidence that the embeddings for intersections and road segments learned by our RN2Vec can support various applications of different types.

Finally, we study the effect of using *both* intersection and road segment embeddings, denoted as $\text{RN2Vec}_{\text{RN}}$, for travel time estimation. As shown in Table 8 and Table 9, $\text{RN2Vec}_{\text{RN}}$ improves the best existing work for intersection embeddings by 17.31% to 23.58% on the three datasets and that for road segment embeddings by 17.01% to 25.33%. As shown, $\text{RN2Vec}_{\text{RN}}$ is also better than using only either intersection embeddings or road segments embeddings generated by RN2Vec. It validates that intersections and road segments in road networks are both important and complementary to each other in travel time estimation.

5.6 Enhancement by Trajectory Data

To validate our argument for using real-world trajectory data as a supplement of the adopted shortest data sampling by default, we compare RN2Vec (denoted by RN2Vec_{S}) with two variants,

Table 10. Trajectories for Intersection Classification

	Traffic Signal Classification (F1-score)		
	San Francisco	Porto	Tokyo
Best Baseline	0.771	0.760	0.736
$RN2Vec_T$	0.853	0.845	0.811
$RN2Vec_S$	0.901 (16.86%)	0.873 (14.87%)	0.824 (11.96%)
$RN2Vec_{S+T}$	**0.907 (17.64%)**	**0.887 (16.71%)**	**0.830 (12.78%)**
	Crossing Classification (F1-score)		
	San Francisco	Porto	Tokyo
Best Baseline	0.757	0.684	0.769
$RN2Vec_T$	0.818	0.767	0.833
$RN2Vec_S$	0.843 (11.36%)	0.798 (16.67%)	0.869 (13.00%)
$RN2Vec_{S+T}$	**0.876 (15.72%)**	**0.817 (19.45%)**	**0.878 (14.17%)**

Table 11. Trajectories for Travel Time Estimation

	Intersection Embeddings (MAE)		
	San Francisco	Porto	Tokyo
Best Baseline	58.87	157.66	92.25
$RN2Vec_T$	50.66	146.07	84.34
$RN2Vec_S$	47.97 (−18.52%)	137.99 (−12.48%)	81.88 (−11.24%)
$RN2Vec_{S+T}$	**46.11 (−21.67%)**	**135.04 (−14.35%)**	**78.87 (−14.50%)**
	Road Segment Embeddings (MAE)		
	San Francisco	Porto	Tokyo
Best Baseline	60.25	152.69	91.92
$RN2Vec_T$	48.22	139.76	85.52
$RN2Vec_S$	46.89 (−22.17%)	133.64 (−12.48%)	82.86 (−9.86%)
$RN2Vec_{S+T}$	**44.36 (−26.37%)**	**130.77 (−14.36%)**	**80.10 (−12.86%)**

$RN2Vec_T$ and $RN2Vec_{S+T}$, in intersection classification and travel time estimation,[16] which have the same parameter settings as RN2Vec except that $RN2Vec_T$ samples real trajectories for training data while $RN2Vec_{S+T}$ samples both shortest paths and real trajectories instead. The performance of RN2Vec, its variants and the best compared model, i.e., NE+PCA, denoted by `Best Baseline`, are summarized in Table 10 and Table 11.

In Table 10, we observe that $RN2Vec_T$ outperforms the `Best Baseline` in intersection classifications, even though it does not perform as well as $RN2Vec_S$ due to the issue of incomplete coverage. We observe that $RN2Vec_S$ outperforms $RN2Vec_T$ by 1.6% to 5.63% and 3.06% to 4.32% in traffic signal classification and crossing classification, respectively. However, $RN2Vec_{S+T}$ brings the performance up one notch to outperform `Best Baseline` by 12.78% to 17.64% and 14.17% to 19.45%, respectively. In Table 11, a similar trend exists in travel time estimation. $RN2Vec_{S+T}$ consistently performs the best, improving 14.35% to 21.67% and 12.86% to 26.37% with intersection and road segment embeddings over the `Best Baseline` in the three datasets. It demonstrates that while the trajectory data are incomplete, the real-world moving behavior of users is useful for representation learning on road networks.

5.7 Test on Generality and Robustness

As mentioned earlier, this work aims to learn useful road network embeddings for *general support* of various ITS applications. To demonstrate the robustness and generality of RLRN and RN2Vec in learning *general purpose* embeddings in support of various ITS applications, we use RN2Vec to learn a set of generic intersection/road segment embeddings (denoted by $RN2Vec_G$) without using information relevant to the classification and travel time estimation tasks. More specifically, we remove the same intersection tag objective in the IRN2Vec sub-module and do not use trajectory data. We following the same parameter settings in previous experiments and set $\alpha = 0.8$ and $\beta = 0.2$. The performance results of $RN2Vec_G$ and the best model among compared models, denoted by `Best Baseline`, are summarized in Table 12 and Table 13. The results show that $RN2Vec_G$ robustly outperforms `Best Baseline` in all experiments, which demonstrate that, although the performance of $RN2Vec_G$ is slightly worse than the RN2Vec, these generic embeddings are generally useful and effective for all applications, and thus exhibiting great generality and robustness.

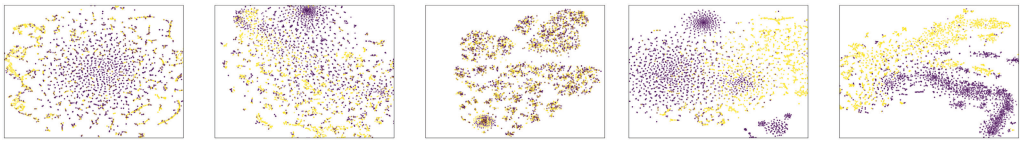
[16] We do not perform experiments on road segment classification due to the lack of trajectory data for Seattle and Chicago.

Table 12. Generic Intersection Embeddings

	Traffic Signal Classification (F1-score)		
	San Francisco	Porto	Tokyo
Best Baseline	0.771	0.760	0.736
$RN2Vec_G$	0.894(15.95%)	0.851(11.97%)	0.79(7.33%)
	Crossing Classification (F1-score)		
	San Francisco	Porto	Tokyo
Best Baseline	0.757	0.684	0.769
$RN2Vec_G$	0.803(6.08%)	0.756(10.53%)	0.824(7.15%)
	Travel Time Estimation (MAE)		
	San Francisco	Porto	Tokyo
Best Baseline	58.87	157.66	92.25
$RN2Vec_G$	48.99(−16.78%)	138.21(−12.34%)	83.47(−9.52%)

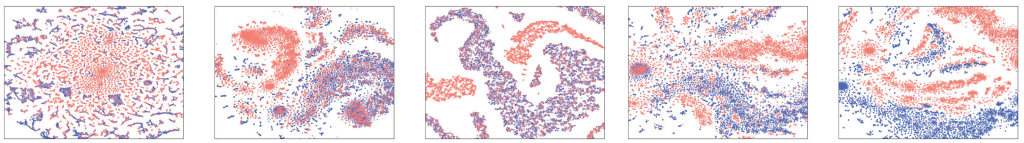
Table 13. Generic Road Segment Embeddings

	Avenue Signal Classification (F1-score)		
	San Francisco	Seattle	Chicago
Best Baseline	0.700	0.669	0.720
$RN2Vec_G$	0.781(11.57%)	0.743(11.06%)	0.777(7.92%)
	Street Classification (F1-score)		
	San Francisco	Seattle	Chicago
Best Baseline	0.745	0.754	0.759
$RN2Vec_G$	0.812(8.99%)	0.835(10.74%)	0.861(13.44%)
	Travel Time Estimation (MAE)		
	San Francisco	Porto	Tokyo
Best Baseline	60.25	152.69	91.92
$RN2Vec_G$	48.87(−18.89%)	136.09(−10.87%)	83.20(−9.49%)



(a) Node2vec (b) NE+PCA (c) GAT (d) IRN2Vec (e) RN2Vec

Fig. 10. Visualization of intersection embeddings with/without traffic signal.



(a) DeepWalk (b) NE+PCA (c) GCN (d) SRN2Vec (e) RN2Vec

Fig. 11. Visualization of road segment embeddings of avenue/non-avenue.

5.8 Visualization of Embeddings

To get a sense of how the learned embeddings are positioned in latent feature space, in this section, we visualize the intersection and road segment embeddings generated by RN2Vec, along with some selected competitive models. We map the learned intersections and road segments embeddings to a two-dimensional space using the t-SNE algorithm and color them based on whether they are labelled with traffic signal and avenue, respectively, i.e., intersections equipped with traffic signal are colored in purple and road segments categorized as avenue are colored in blue. As shown in Figure 10 and Figure 11, the purple-colored intersections and blue-colored road segments, respectively, are much better clustered in embedding space of RN2Vec than in those spaces of other models, explaining why RN2Vec achieves excellent performance superior to others.

6 CONCLUSION AND FUTURE WORK

In this article, we study the problem of road network representation learning, which is of interest to the academic and industrial communities due to its practical value to various ITS applications. In the proposed RLRN framework, we develop a novel RN2Vec model by exploiting various geo-locality and homogeneity relationships in road network to learn road network embeddings, which

are general-purposed, effective, and reusable. In addition to the modeling ideas behind RN2Vec, insights obtained from this work, e.g., shortest path sampling and distance-based neighborhood, provide guidance for training and use of the road network embeddings in ITS. The result has significant implications and potential applications on a variety of forecasting problems, e.g., accident occurrence, air pollution, traffic control, traffic congestion, and so on. The automatically learned embeddings are practically useful for many ITS applications that involve road networks, e.g., map information management, intelligent speed adaptation, route planning, travel time estimation, and so on.

Several issues arising in the course of this study reveal some limitations of our work. (i) Data quality: We have exploited OSM and trajectory data in our study. While OSM data are rich and valuable, the user-generated tags tend to be incomplete or inaccurate. Moreover, both OSM and trajectory data have low coverage on road networks. (ii) Relationship variety: Only a few relationships that capture intrinsic properties of road networks are exploited. (iii) Directed graphs: The proposed model is designed based on undirected graphs, which may exist a gap from real-world road networks. Further research on above issues have a potential to improve the performance of RN2Vec and bring more advances in the direction of road network representation learning.

In the future work, we plan to extend our work for missing tag recommendation and inaccurate information detection on OSM data and explore ideas of transfer learning to address the low coverage issues. Moreover, we plan to exploit more variety of data, e.g., aerial images, Lidar, Hyper-spectral images for road networks, traffic data, human mobility data, and explore meaningful metapaths in road networks to identify and incorporate more variety of relationships in our models. Finally, we will study the representation learning problem for urban regions, aiming to encode the underlying patterns and structures of regions for smartening of modern cities.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (2013), 1798–1828.

[2] The US Census Bureau. 2019. TIGER Data. Retrieved from https://wiki.openstreetmap.org/wiki/TIGER.

[3] Taxi Trajectory Prediction Challenge. 2015. Trajectory Prediction. Retrieved from http://www.geolink.pt/ecmlpkdd2015-challenge/.

[4] Steve Coast. 2004. OpenStreetMap. Retrieved from https://www.openstreetmap.org/.

[5] Jingze Cui, Xian Zhou, Yanmin Zhu, and Yanyan Shen. 2018. A road-aware neural network for multi-step vehicle trajectory prediction. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. 701–716.

[6] M. Fruensgaard and T. S. Jepsen. 2017. Improving cost estimation models with estimation updates and road2vec: A feature learning framework for road networks. *Master's thesis, Aalborg University* (2017).

[7] Yanjie Fu, Guannan Liu, Yong Ge, Pengyang Wang, Hengshu Zhu, Chunxiao Li, and Hui Xiong. 2018. Representing urban forms: A collective learning model with heterogeneous human mobility data. *IEEE Trans. Knowl. Data Eng.* 31, 3 (2018), 535–548.

[8] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[9] Weiwei Guo, Huiji Gao, Jun Shi, Bo Long, Liang Zhang, Bee-Chung Chen, and Deepak Agarwal. 2019. Deep natural language processing for search and recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3199–3200.

[10] Carlos Herranz Perdiguero and Roberto J. López Sastre. 2018. ISA: Intelligent speed adaptation from appearance. *The Computing Research Repository* (2018).

[11] Tobias Skovgaard Jepsen, Christian S. Jensen, Thomas Dyhre Nielsen, and Kristian Torp. 2018. On network embedding for machine learning on road networks: A case study on the danish road network. In *Proceedings of the IEEE International Conference on Big Data (Big Data'18)*. 3422–3431.

[12] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[13] Jason Lally. 2019. Open Data in San Francisco. Retrieved from https://data.sfgov.org/Transportation/Stop-Signs/4542-gpa3.
[14] Kang Liu, Song Gao, Peiyuan Qiu, Xiliang Liu, Bo Yan, and Feng Lu. 2017. Road2vec: Measuring traffic interactions in urban road system from massive travel routes. *Int. J. Geo-Inf.* 6, 11 (2017), 321.
[15] Sebastian Mattheis. 2016. Barefoot. Retrieved from https://github.com/bmwcarit/barefoot/.
[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems*. 3111–3119.
[17] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. 2019. Speech recognition using deep neural networks: A systematic review. *IEEE Access* 7 (2019), 19143–19165.
[18] Mark Nixon and Alberto Aguado. 2019. *Feature Extraction and Image Processing for Computer Vision*. Academic Press.
[19] U.S. Department of Transportation. 2017. Intelligent Transportations Systems. Retrieved from https://www.its.dot.gov.
[20] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1720–1730.
[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. 701–710.
[22] Michal Piorkowski and Matthias Grossglauser. 2009. Dataset of mobility traces of taxi cabs in San Francisco. Retrieved from https://crawdad.org/epfl/mobility/20090224/.
[23] Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv:1610.02906*. Retrieved from https://arxiv.org/abs/1610.02906.
[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*. 1067–1077.
[25] Soujanya Poria Tom Young, Devamanyu Hazarika and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* 13, 3 (2018), 55–75.
[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv:1710.10903*. Retrieved from https://arxiv.org/abs/1710.10903.
[27] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Trans. Intell. Syst. Technol.* 10, 2 (2019), 1–22.
[28] Meng-xiang Wang, Wang-Chien Lee, Tao-yang Fu, and Ge Yu. 2019. Learning embeddings of intersections on road networks. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 309–318.
[29] Pengyang Wang, Yanjie Fu, Hui Xiong, and Xiaolin Li. 2019. Adversarial substructured representation learning for mobile user profiling. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 130–138.
[30] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Xiaolin Li, and Dan Lin. 2018. Learning urban community structures: A collective embedding perspective with periodic spatial-temporal mobility graphs. *ACM Trans. Intell. Syst. Technol.* 9, 6 (2018), 1–28.
[31] Pengyang Wang, Yanjie Fu, Jiawei Zhang, Pengfei Wang, Yu Zheng, and Charu Aggarwal. 2018. You are how you drive: Peer and temporal-aware representation learning for driving behavior analysis. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2457–2466.
[32] Jiajie Xu, Jing Zhao, Rui Zhou, Chengfei Liu, Pengpeng Zhao, and Lei Zhao. 2019. Destination prediction a deep learning based approach. *IEEE Trans. Knowl. Data Eng.* (2019). DOI: 10.1109/TKDE.2019.2932984
[33] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.
[34] Yanbo Pang. Yoshihide Sekimoto. 2017. Open datasets for Tokyo trajectory. Retrieved from https://github.com/sekilab/OpenPFLOW.
[35] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2020. Network representation learning: A survey. *IEEE Trans. Big Data* 6, 1 (2020), 3–28.
[36] Yunchao Zhang, Yanjie Fu, Pengyang Wang, Xiaolin Li, and Yu Zheng. 2019. Unifying inter-region autocorrelation and intra-region structures for spatial embedding via collective adversarial learning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1700–1708.
[37] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 1–41.

Received February 2020; revised July 2020; accepted September 2020