# vec2Link: Unifying Heterogeneous Data for Social Link Prediction

Fan Zhou*
School of Information and Software Engineering, University of Electronic Science and Technology of China
fan.zhou@uestc.edu.cn

Bangying Wu
School of Information and Software Engineering, University of Electronic Science and Technology of China
wubangying580@gmail.com

Yi Yang
The Hong Kong University of Science and Technology
imyiyang@ust.hk

Goce Trajcevski
Iowa State University, Ames
gocet25@iastate.edu

Kunpeng Zhang
University of Maryland, College park
kzhang@rhsmith.umd.edu

Ting Zhong
University of Electronic Science and Technology of China
zhongting@uestc.edu.cn

## ABSTRACT

Recent advances in network representation learning have enabled significant improvements in the link prediction task, which is at the core of many downstream applications. As an increasing amount of mobility data becoming available due to the development of location technologies, we argue that this resourceful user mobility data can be used to improve link prediction performance. In this paper, we propose a novel link prediction framework that utilizes user offline check-in behavior combined with user online social relations. We model user offline location preference via probabilistic factor model and represent user social relations using neural network embedding. Furthermore, we employ locality-sensitive hashing to project the aggregated user representation into a binary matrix, which not only preserves the data structure but also speeds up the followed convolutional network learning. By comparing with several baseline methods that solely rely on social network or mobility data, we show that our unified approach significantly improves the performance.

## KEYWORDS

link prediction; network embedding; locality-sensitive hashing

**ACM Reference Format:**
Fan Zhou, Bangying Wu, Yi Yang, Goce Trajcevski, Kunpeng Zhang, and Ting Zhong. 2018. vec2Link: Unifying Heterogeneous Data for Social Link Prediction. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269244

*Corresponding author.


*CIKM '18, October 22–26, 2018, Torino, Italy*

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269244

## 1 INTRODUCTION

Link prediction [13] aims at identifying missing links, or links that are likely to be formed in the future in a network. It is at the core of many applications, including prediction of a friendship in social network, relations between entities in a knowledge graph, affinities between users and items in a recommender system, and potential biological interactions between drugs and diseases.

Recent years have brought advances in network representation learning, such as DeepWalk [16], Line [20], node2vec [8] and struc2vec [17]. These methods often learn low-dimensional representation of network nodes – also known as network embedding – to facilitate the link prediction task, where the propensity of forming a link is based on the nodes similarity. Concurrently, as increasing amount of mobility data becomes available due to the development of location technologies, there is a body of works focusing on inferring social relationships from user offline mobility data [1, 10, 22, 24]. As demonstrated in [21, 22], mobility data can indeed serve as a strong predictor for inferring social ties.

However, most previous studies either infer potential links between two users solely from their mobility data [1, 23] or learn user preference only from social network structures [8, 17, 20]. To our knowledge, taking advantage of both user location information *and* the social relationship for link prediction has not been exploited, and this paper presents an attempt to bridge this gap. Specifically, we propose a novel link prediction framework **vec2Link** which captures user offline location preference (learned from check-in behaviors) and online social preference (learned from network structures). vec2Link models user location preference via a probabilistic factor model (PFM), while leveraging neural network embedding (NE) method for social representation. It then employs a locality-sensitive hashing (LSH) to project the two representations into a binary vector in a manner that preserves similarities in both check-in behaviors and network structures. Subsequently, the matrix reshaped from the binary vector is fed into a convolutional neural network to perform prediction. This novel usage of LSH not only makes the convolution faster – compared to operations over original representations without LSH – but it also preserves well the latent patterns behind data, especially similarities, which yields performance improvement. We conducted experiments on two real-world LBSN datasets. By comparing with several baseline methods that solely rely on social network or mobility data,

we show that unifying both social and mobility information can significantly improve the prediction performance.

## 2 METHODOLOGY

The proposed model consists of three components – visit preference learning, social relationship embedding, and heterogeneous mapping – as illustrated in Figure 1.

### 2.1 Modeling Visit Preference

Given a set of Point-of-Interests (POIs) $\mathcal{L}$ ($|\mathcal{L}| = M$) and a set of users $\mathcal{U}$ ($|\mathcal{U}| = N$), each user $u_i \in \mathcal{U}$ is associated with a set of historical check-ins $\mathcal{L}^{u_i}$. Let $\mathbf{V} \in \mathbb{R}^{N \times M}$ be the visit frequency matrix whose element $v_{ij}$ denotes the number of visits of POI $l_j$ by user $u_i$. Matrices $\mathbf{U} \in \mathbb{R}^{N \times D}$ and $\mathbf{L} \in \mathbb{R}^{M \times D}$ denote the corresponding latent user and POI feature matrix, with row vectors $\mathbf{u}_i \in \mathbb{R}^D$ and $\mathbf{l}_j \in \mathbb{R}^D$ (both are $D$-dimensional vectors) representing user-specific and POI-specific latent factors, respectively.

To understand visiting preferences, we employ the Probabilistic Factor Model (PFM) [5, 14, 15]. Let $\mathbf{Z} \in \mathbb{R}^{N \times M}$ be the matrix of expected frequency with the same dimensions as $\mathbf{V}$. Every observed visit frequency $v_{ij} \in \mathbf{V}$ is assumed to follow a Poisson distribution with the mean $z_{ij}$ in $\mathbf{Z}$ which, in turn, is factorized into two matrices $\mathbf{U} \in \mathbb{R}^{N \times D}$ and $\mathbf{L} \in \mathbb{R}^{M \times D}$. Each element $u_{ik} \in \mathbf{U}$ ($k = 1, \cdots, D$) encodes the preference of a user $u_i$ for latent topic $k$, and each $l_{jk} \in \mathbf{L}$ reflects the affinity of POI $l_j$ to topic $k$. The priors of $u_{ik}$ and $l_{jk}$ follow Gamma distributions $\Gamma(\alpha, \beta)$. Thus, we have following generative process of observed user-POI visit frequency $v_{ij}$:

1. Generate user latent factor $u_{ik} \sim \Gamma(\alpha_u, \beta_u)$.
2. Generate POI latent factor $l_{jk} \sim \Gamma(\alpha_l, \beta_l)$.
3. Generate visit frequency $v_{ij} \sim \text{Poisson}(\mathbf{u}_i^\intercal \mathbf{l}_j)$.

where the probability of visit frequency $\mathbf{V}$ follows a Poisson distribution:

$$p(\mathbf{V}|\mathbf{U}, \mathbf{L}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \frac{(\mathbf{u}_i^\intercal \mathbf{l}_j)^{v_{ij}} \exp(-\mathbf{u}_i^\intercal \mathbf{l}_j)}{v_{ij}!}. \quad (1)$$

Since $\mathbf{Z} = \mathbf{U}\mathbf{L}^\intercal$, the latent matrices $\mathbf{U}$ and $\mathbf{L}$ can be estimated with MAP (maximum a posteriori) as:

$$p(\mathbf{U}, \mathbf{L}|\mathbf{V}, \alpha_u, \beta_u, \alpha_l, \beta_l) \propto p(\mathbf{V}|\mathbf{U}, \mathbf{L}) p(\mathbf{U}|\alpha_u, \beta_u) p(\mathbf{L}|\alpha_l, \beta_l). \quad (2)$$

The log of the posterior distribution over the user and POI latent factors is inferred as:

$$\begin{aligned} \mathcal{E}(\mathbf{U}, \mathbf{L}|\mathbf{V}) = & \sum_{i=1}^{N} \sum_{k=1}^{D} ((\alpha_u - 1) \ln u_{ik} - \frac{u_{ik}}{\beta_u} - \alpha_u \ln \beta_u + \ln \beta_u) \\ & + \sum_{j=1}^{M} \sum_{k=1}^{D} ((\alpha_l - 1) \ln l_{jk} - \frac{l_{jk}}{\beta_l} - \alpha_l \ln \beta_l + \ln \beta_l) \\ & + \sum_{i=1}^{N} \sum_{j=1}^{M} (v_{ij} \ln(\mathbf{u}_i^\intercal \mathbf{l}_j) - \mathbf{u}_i^\intercal \mathbf{l}_j) + C. \end{aligned} \quad (3)$$

The latent factors in $\mathbf{U}$ can be inferred by taking derivatives on $\mathcal{E}$ with respect to $u_{il}$. Finally, $u_{ik}$ can be iteratively updated using gradient descent. The visit preference for user $u_i$ can be obtained as $\mathbf{u}_i = u_{i1}, ..., u_{iD}$. We denote the user visit preference representation as $\mathbf{u}_i^v$.

### 2.2 Modeling Social Representation

The social representation of a user $u_i (\in \mathbf{U})$ can be obtained through network embedding (NE). Given a graph $\mathcal{G} = (\mathcal{V}, E)$, and a predefined dimensionality of the embedding $d$ ($d \ll |\mathcal{V}|$), the problem of NE is to encode $\mathcal{G}$ into a $d$-dimensional space ($\texttt{ENC}(\mathcal{V}) \rightarrow \mathbb{R}^d$), in which the network property is preserved as much as possible. Formally, the NE problem can be defined in the framework of autoencoder [9] as:

$$\texttt{DEC}(\texttt{ENC}(v_i), \texttt{ENC}(v_j)) = \texttt{DEC}(\mathbf{u}_i^n, \mathbf{u}_j^n) \approx \mathcal{P}(v_i, v_j), \quad (4)$$

where $v_i, v_j \in \mathcal{V}$; $\texttt{DEC}(\cdot, \cdot) \rightarrow \mathbb{R}^+$ decodes pairs of node embeddings to a real-valued graph proximity measure; and $\mathcal{P}(\cdot)$ is a user-defined, graph-based proximity measure between nodes. The graph property can be quantified using a proximity measure such as the first- [16] and higher-order [20] proximity. Each graph is represented as either one d-dimensional vector (for a whole graph) or a set of $d$-dimensional vectors where each vector represents the embedding of a part of the graph (e.g., node, edge, substructure) [2].

We note that, in our model, we do not rely on specific implementation of network embedding. Instead, we leverage the output vector $\mathbf{u}_i^n$ of NE as the social representation for each user $u_i$ – and we incorporate several state-of-the-art NE models to investigate the link prediction performance.

### 2.3 The Joint Model with LSH

After the above two steps of learning preference representations, we obtain the vectors $\mathbf{u}_i^v \in \mathbb{R}^D$ and $\mathbf{u}_i^n \in \mathbb{R}^d$ for $u_i$, respectively denoting user's visit preference over POIs and social representations. However, we need a way to preserve data properties as much as possible while boosting the link prediction. One of the practical ways to facilitate efficient feature retrieval and to reduce computational cost is to convert the feature vectors to binary codes. In this work, we propose to embed the domain specific representation with the Locality-Sensitive Hashing (LSH) [12]. Recently, LSH has been employed to reduce the amount of computation needed to train and test neural networks through hashing weights and biases into binary representations and preserving inner products [6, 11, 19]. In this paper, we apply LSH to encode continuous embedding vectors into binary ones while preserving similarities for efficient link prediction. Specifically, a hash $h$ is called $(S, cS, p_1, p_2)$-sensitive if, for any two points $x, y \in \mathbb{R}$, $h$ satisfies the following:

1. if $sim(x, y) \geq S$, then $Pr_h(h(x) = h(y)) \geq p_1$;
2. if $sim(x, y) \leq cS$, then $Pr_h(h(x) = h(y)) \leq p_2$;

where $S$ is a threshold of interest, $c < 1$ and $p_1 > p_2$ for efficient approximate nearest neighbor search. $sim$ denotes a similarity measure, which is defined as the cosine similarity between two vectors in our case. The LSH algorithm aims at maximizing the probability that similar data are mapped to similar binary codes and could benefit many applications such as Maximum Inner Product Search (MIPS) [18].

Then, we perform the following operations for every possible pair of users $e_{ij} = (u_i, u_j)$ in $\mathcal{G}$:

1. Concatenate $\mathbf{u}_i^v$ and $\mathbf{u}_i^n$ as a vector $\mathbf{u}_i^{vn} = [\mathbf{u}_i^v; \mathbf{u}_i^n] \in \mathbb{R}^{D+d}$.
2. Generate Hamming code $\mathbf{m}_i \in \{0, 1\}^m$ for $\mathbf{u}_i^{vn}$ with LSH.

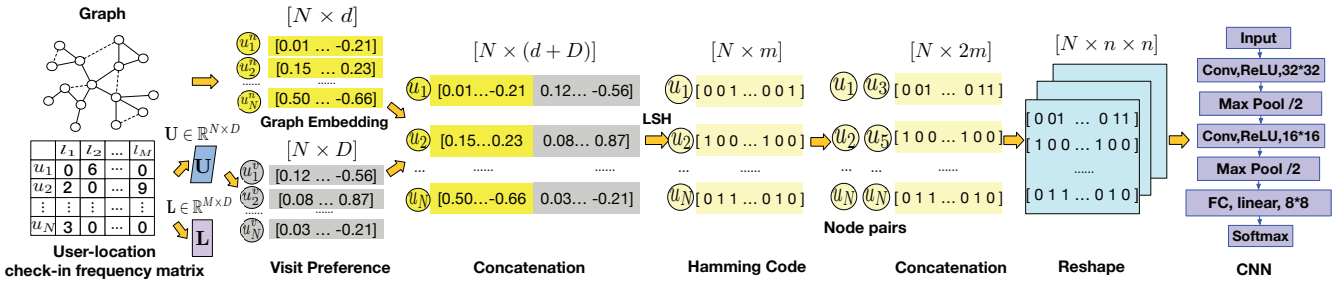


**Figure 1: The Overview of our proposed method vec2Link.**

3. Concatenate $\mathbf{m}_i$ and $\mathbf{m}_j$ to form a binary vector $\mathbf{m}_{ij}(\in \{0,1\}^{2m}) = [\mathbf{m}_i; \mathbf{m}_j]$.

In step 1, we set $D = d$ for ease of computation and in the sequel we use $\mathbf{u}_i$ in place of $\mathbf{u}_i^{vn}$, without ambiguity. Step 2 works as a Sign Random Projection (SRP) [4, 7]: Given a vector $\mathbf{u}_i$, SRP utilizes a random vector $\mathbf{r}$ with each bit $\mathbf{r}(b)$ generated from independent identically distributed normal, i.e., $\mathbf{r}(b) \sim N(0,1)$, and only stores the sign of the projection.

Now we have a binary vector $\mathbf{m}_i \in \{0,1\}^m$ aggregating the heterogeneous information consisting of social representation and visit preference of user $u_i$, where $m$ is the number of bits of random vector $\mathbf{r}$, i.e., the number of hyperplanes used for hash projection. For each link (or edge) between $u_i$ and $u_j$, we concatenate $\mathbf{m}_i$ and $\mathbf{m}_j$ to obtain $\mathbf{m}_{ij} \in \{0,1\}^{2m}$ in step 3.

## 2.4 Efficient Representation via Reshape

After mapping the heterogeneous embedding into a Hamming space, we reshape the one dimensional vector $\mathbf{m}_{ij}$ to a binary matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$. Then we apply a convolution operation on $\mathbf{B}$ with a kernel $\mathbf{K} \in \mathbb{R}^{w \times w}$ to obtain the feature maps $\mathbf{M} = \{\mathbf{m}_1, ..., \mathbf{m}_k\}$. Max-over-time pooling is used on the feature maps to learn the underlying representation of matrix $\mathbf{B}$. Note that we also try one dimensional convolution on $\mathbf{m}_{ij}$ but found that reshaping to a matrix performs better.

# 3 EXPERIMENTS

We now present our experimental observations, comparing vec2Link with several baseline methods on two public datasets commonly used in link prediction.

The dimensionality of location preference vector $D$ and network embedding vector $d$ are both set to 128, while the length $m$ of the Hamming code is 512. In our implementation, we use two layer CNN and 16 (32) filters in the first (second) layer. The kernel size in both layers are $5 \times 5$. The activation function is ReLU. For all experiments, we generate training and testing set of edges following [8]: To obtain positive instances, we remove 50% of edges chosen randomly from the network while ensuring that the remaining network obtained after the edge removals is connected. To generate negative examples, we randomly sample an equal number of non-existing edges. For all experiments, we randomly choose 80% of the nodes and corresponding edges for training and the rest for testing.

**Table 1: Descriptives of datasets.**

| Dataset | #check_ins | #POIs | #edges | #users |
|---|---|---|---|---|
| **Foursquare@NYC** | 22,563 | 1,992 | 5,810 | 588 |
| **Foursquare@TKY** | 38,742 | 2,212 | 9,624 | 1,055 |
| **Gowalla@DC** | 13,594 | 4,795 | 5,826 | 880 |
| **Gowalla@CHI** | 10,314 | 3,269 | 2,542 | 627 |

**Datasets**: We used two public datasets – Foursquare and Gowalla[1]. For Foursquare, we extract the users and friendship networks and check-in history in the area of New York City (NYC) and Tokyo (TKY). Similarly, we extract the data in Washington DC (DC) and Chicago (CHI) from Gowalla. Data statistics are shown in Table 1.

**Metrics**: We use Area Under Curve (AUC) to measure the performance of link prediction. In addition, we use four binary operators proposed in [8], i.e., *Average, Hadamard, Weighted-L1 and Weighted-L2*, for generating a representation of the edge between any two nodes in the graph.

**Baselines**: We compare vec2Link with several state-of-the-art network embedding methods. Since vec2Link leverages mobility data for improving link prediction and does not rely on specific NE, the reported results are NE-related, i.e., the NE part of vec2Link is the same as the compared method. The baselines consist of:

- **walk2friends** [1]: is a Skip-Gram based model that employs mobility information for inferring social links.
- **DeepWalk** [16]: performs random walks over networks and employs Skip-Gram model to learn node embeddings.
- **LINE** [20]: learns node embeddings in networks considering both first-order and second-order proximities.
- **GraRep** [3]: learns node embeddings in large-scale networks considering both first-order and second-order proximities.
- **node2vec** [8]: performs the 2nd-order random walks to explore neighborhood architecture and embeds nodes with Skip-Gram model.
- **struc2vec** [17]: involves generating a series of weighted auxiliary graphs, capturing structural similarities between nodes' $k$-hop neighborhoods, and running node2vec for embedding the auxiliary graphs.

Note that walk2friends is the only baseline method that utilizes user mobility data, while all the other baselines rely on using online social relationship for link prediction. To test the efficiency impact of LSH, we also used *vec2Link-*, a framework that does not apply LSH.

[1] http://snap.stanford.edu

**Table 2: Area Under Curve (AUC) scores for link prediction.**

| Method | Operator | Dataset | | | |
|---|---|---|---|---|---|
| | | NYC | TKY | WA | CHI |
| Walk2friend | | 0.6648 | 0.6139 | 0.6156 | 0.6059 |
| DeepWalk | Average | 0.5735 | 0.5906 | 0.7115 | 0.5959 |
| | Hadamard | 0.8621 | 0.7995 | 0.7858 | 0.8227 |
| | Weighted-L1 | 0.9052 | 0.8732 | 0.9063 | 0.8144 |
| | Weighted-L2 | 0.9061 | 0.8731 | 0.9062 | 0.8107 |
| vec2Link-vec2Link | | 0.9441 | 0.9051 | 0.9232 | 0.8575 |
| | | **0.9533** | **0.9085** | **0.9361** | **0.8622** |
| LINE | Average | 0.6739 | 0.6537 | 0.6842 | 0.6240 |
| | Hadamard | 0.5185 | 0.5825 | 0.4997 | 0.5731 |
| | Weighted-L1 | 0.5477 | 0.6020 | 0.5333 | 0.5515 |
| | Weighted-L2 | 0.5358 | 0.5992 | 0.5326 | 0.5945 |
| vec2Link-vec2Link | | 0.8652 | 0.8332 | 0.7955 | 0.7847 |
| | | **0.8676** | **0.8421** | **0.8093** | **0.7914** |
| GraRep | Average | 0.6643 | 0.6656 | 0.6763 | 0.5670 |
| | Hadamard | 0.9327 | 0.9031 | 0.7063 | 0.8030 |
| | Weighted-L1 | 0.9171 | 0.8809 | 0.8821 | 0.8558 |
| | Weighted-L2 | 0.9096 | 0.8809 | 0.8995 | 0.8344 |
| vec2Link-vec2Link | | 0.9422 | 0.9183 | 0.9279 | 0.8650 |
| | | **0.9539** | **0.9258** | **0.9322** | **0.8852** |
| node2vec | Average | 0.5660 | 0.5737 | 0.7137 | 0.5857 |
| | Hadamard | 0.8752 | 0.8280 | 0.8392 | 0.8288 |
| | Weighted-L1 | 0.9189 | 0.8816 | 0.9169 | 0.8376 |
| | Weighted-L2 | 0.9228 | 0.8825 | 0.9177 | 0.8399 |
| vec2Link-vec2Link | | 0.9462 | 0.9052 | 0.9162 | 0.8375 |
| | | **0.9510** | **0.9129** | **0.9257** | **0.8642** |
| struct2vec | Average | 0.7834 | 0.7663 | 0.7895 | 0.6869 |
| | Hadamard | 0.7463 | 0.7146 | 0.7562 | 0.6311 |
| | Weighted-L1 | 0.7046 | 0.6815 | 0.7208 | 0.6129 |
| | Weighted-L2 | 0.7189 | 0.6844 | 0.7287 | 0.6267 |
| vec2Link-vec2Link | | 0.8328 | 0.8175 | 0.8208 | 0.7584 |
| | | **0.8458** | **0.8275** | **0.8319** | **0.7670** |

**Results**: Recall that vec2Link does not rely on any specific NE method for representing social network users in a low-dimensional space. Therefore, we compare it with the baseline using the same NE model – e.g., the NE method in vec2Link uses DeepWalk when comparing to DeepWalk. The experiment results are illustrated in Table 2, from which we can observe that:

(1) leveraging the user check-in preference can efficiently improve the link prediction performance. For example, vec2Link achieves 6.35%, 33.12%, 6.09%, 3.06% and 22.39% improvement over DeepWalk, LINE, GraRep, node2vec and struc2vec respectively on Gowalla@CHI data. As expected, walk2friends performs the worst due to lacking of the network information.

(2) vec2Link with LSH slightly increases ($p < 0.05$) the prediction accuracy compared to directly concatenating two heterogeneous vectors, i.e., vec2Link without LSH. The reason behind this improvement lies in that LSH can distinguish the dissimilar user representations while preserving the similar ones, which facilitates the feature learning in CNN. Moreover, training a convolution neural network on binary matrix is order of magnitude faster than training on dense real-value vectors, which can benefit large-scale network learning.

(3) We also observe that node2vec performs the second best and weighted-L2 is the best operator for learning edge features for NE methods – in fact, vec2Link can be thought of as a method for incorporating heterogeneous information into learning edge features.

## 4 CONCLUSIONS

We presented vec2Link – a novel link prediction framework, capturing both social network relationship and location-based check-in information of the users. We demonstrated vec2Link's superior performance over the state-of-the-art link prediction methods which rely either on online user social relationship or offline user mobility data. Currently, we are extending vec2Link to incorporate the user check-in preference to a broader range of spatio-temporal contexts such as geographical influence and sequential patterns of human mobility.

## ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No.61602097 and No.61472064), NSF grants III 1213038 and CNS 1646107, and ONR grant N00014-14-10215.

## REFERENCES

[1] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. 2017. walk2friends: Inferring Social Links from Mobility Profiles. In *CCS*.
[2] Hongyi Cai, Vincent W. Zheng, and Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *TKDE* (2018).
[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *CIKM*.
[4] Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *STOC*.
[5] Ye Chen, Michael Kapralov, Dmitry Pavlov, and John F Canny. 2009. Factor Modeling for Advertisement Targeting. In *NIPS*.
[6] Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. 2017. A neural algorithm for a fundamental computing problem. *Science* 358, 6364 (2017), 793–796.
[7] Michel X Goemans and David P Williamson. 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM* (1995).
[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*.
[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40, 3 (2017), 52–74.
[10] Hsun-Ping Hsieh, Rui Yan, and Cheng-Te Li. 2015. Where You Go Reveals Who You Know: Analyzing Social Ties from Millions of Footprints. In *CIKM*.
[11] Qinghao Hu, Peisong Wang, and Jian Cheng. 2018. From Hashing to CNNs: Training Binary Weight Networks via Hashing. In *AAAI*.
[12] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*.
[13] David Liben-Nowell and Jon M Kleinberg. 2003. The link prediction problem for social networks. In *CIKM*.
[14] Bin Liu, Hui Xiong, Spiros Papadimitriou, Yanjie Fu, and Zijun Yao. 2015. A General Geographical Probabilistic Factor Model for Point of Interest Recommendation. *TKDE* (2015).
[15] Hao Ma, Chao Liu, Irwin King, and Michael R Lyu. 2011. Probabilistic factor models for web site recommendation. In *SIGIR*.
[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*.
[17] L. F. Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *KDD*.
[18] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NIPS*.
[19] Ryan Spring and Anshumali Shrivastava. 2017. Scalable and Sustainable Deep Learning via Randomized Hashing. In *KDD*.
[20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*.
[21] Cheng Wang, Jieren Zhou, and Bo Yang. 2017. From Footprint to Friendship: Modeling User Followership in Mobile Social Networks From Check-in Data. In *SIGIR*.
[22] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-László Barabási. 2011. Human mobility, social ties, and link prediction. In *KDD*.
[23] Hongjian Wang, Zhenhui Li, and Wang-Chien Lee. 2014. PGT: Measuring Mobility Relationship Using Personal, Global and Temporal Factors. In *ICDM*.
[24] Pinghui Wang, Feiyang Sun, Di Wang, Jing Tao, Xiaohong Guan, and Albert Bifet. 2017. Inferring Demographics and Social Networks of Mobile Device Users on Campus From AP-Trajectories. In *WWW*.