Knowledge-Based Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Deep Collaborative Embedding for information cascade prediction ☆

Yuhui Zhao [a], Ning Yang [a,*], Tao Lin [a,*], Philip S. Yu [b]

[a] College of Computer Science, Sichuan University, China
[b] Department of Computer Science, University of Illinois at Chicago, USA

ARTICLE INFO

Article history:
Received 15 May 2019
Received in revised form 7 January 2020
Accepted 9 January 2020
Available online xxxx

Keywords:
Information cascade prediction
Deep Collaborative Embedding
Network embedding

ABSTRACT

Recently, information cascade prediction has attracted increasing interest from researchers, but it is far from being well solved partly due to the three defects of the existing works. First, the existing works often assume an underlying information diffusion model, which is impractical in real world due to the complexity of information diffusion. Second, the existing works often ignore the prediction of the infection order, which also plays an important role in social network analysis. At last, the existing works often depend on the requirement of underlying diffusion networks which are likely unobservable in practice. In this paper, we aim at the prediction of both node infection and infection order without requirement of the knowledge about the underlying diffusion mechanism and the diffusion network, where the challenges are two-fold. The first is what cascading characteristics of nodes should be captured and how to capture them, and the second is that how to model the non-linear features of nodes in information cascades. To address these challenges, we propose a novel model called Deep Collaborative Embedding (DCE) for information cascade prediction, which can capture not only the node structural property but also two kinds of node cascading characteristics. We propose an auto-encoder based collaborative embedding framework to learn the node embeddings with cascade collaboration and node collaboration, in which way the non-linearity of information cascades can be effectively captured. The results of extensive experiments conducted on real-world datasets verify the effectiveness of our approach.

1. Introduction

In recent years, as more and more people enjoy the services provided by Facebook, Twitter, and Weibo, etc., information cascades have become ubiquitous in online social networks, which has motivated a huge amount of researches [1–5]. An important research topic is information cascade prediction, whose purpose is to predict who will be infected by a piece of information in the future [6–9], where infection refers to the actions that users reshare (retweet) or comment a tweet, a photo, or other piece of information [10].

While lots of methods have been proposed for information cascade prediction [6,11–14], the existing works often suffer from three defects. First, the existing works often focus on predicting the probability that whether a node will be infected in the future given nodes infected in the past, but ignore the prediction of infection order, i.e., which nodes will be infected earlier or later than others. However, predicting the infection order is important in many scenarios. For example, it is helpful for blocking rumor spread to know who will be the next infected node [15,16]. Second, the existing methods often assume that information diffusion follows a parametric model such as Independent Cascade (IC) model [17] and Susceptible–Infected (SI) model [18]. In real world, however, information diffusion processes are so complicated that we seldom exactly know the underlying mechanisms of how information diffuses [19]. At last, the existing works often assume that the explicit paths along which information propagates between nodes are observable. Yet in many scenarios we can only observe that nodes get infected but cannot know who infects them [12]. For example, in viral marketing, one can track whether a customer buys a product but it is difficult to exactly determine who influences her/him.

In this paper, we aim at the problem of information cascade prediction without requirement of the knowledge about the underlying diffusion mechanism and the diffusion network. This is not easy due to the following two major challenges:

- **Cascading Characteristics** The probability that a node is infected by a cascade and the relative infection order mainly

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2020.105502.

* Corresponding authors.
E-mail addresses: zhaoyuhui@stu.scu.edu.cn (Y. Zhao), yangning@scu.edu.cn (N. Yang), lintao@scu.edu.cn (T. Lin), psyu@uic.edu (P.S. Yu).

https://doi.org/10.1016/j.knosys.2020.105502
0950-7051/

depend on its cascading characteristics that reveal its relation to other nodes in that cascade. The existing methods often just take into consideration the static structural properties of nodes, for example, the node neighborship in a static social network. However, the cascading characteristics of a node intuitively vary in different cascades, and different cascades can contain totally different infection ranges or orders of nodes. For example, in some cascades, one node may often get infected by certain nodes, but in other cascades, it may be more susceptible to different nodes, even though the node structural properties remain the same. Intuitively, different contents often lead to different cascading characteristics of a node and result in different underlying mechanisms in different cascades. However, in many situations it is not easy to recognize the content (i.e., what is diffused) and its underlying diffusion mechanism (i.e., why and how it is diffused). For example, we often do not know what virus is being propagated in a plague, but when and which nodes are infected can be observed. To make prediction for cascades in such situations, we have to explicitly model the observable cascading characteristics which arguably implicitly captures the effect of the unobservable content and underlying mechanism as well. Therefore, what cascading characteristics of nodes should be captured and how to capture them are crucial to our purpose.

- **Cascading Non-linearity** Information cascades are often non-linear. The non-linearity comes from two perspectives. One is the non-linearity of the dynamics of the information cascades, and the other is the non-linearity of the structure of the social networks on which cascades exist. The non-linearity will cause the problem when nodes spread the content of a cascade, they exhibit non-linear cascading patterns (e.g., emergence pattern) that the existing shallow models cannot effectively recognize. How to capture the non-linear features of nodes in information cascades is also a critical challenge for our problem.

Inspired by the impressive network representation learning ability of deep learning that has been demonstrated by the recent works [20–22], we propose a novel model called Deep Collaborative Embedding (DCE) for prediction of infection and infection order in cascades, which can learn the embeddings without assumption about the underlying diffusion model and diffusion networks. The main idea of DCE is to collaboratively embed the nodes with a deep architecture into a latent space where the closer the embeddings of the two nodes are, the more likely the two nodes will be infected in the same cascade and the closer their infection time will be.

Different from the traditional network embedding methods [20,23–25], which mainly focus on preserving the static structural properties of nodes in a network, DCE can capture not only the node structural property but also two kinds of node *cascading characteristics* that are important for the prediction of node infection and infection order. One is the *cascading context*, which reveals the temporal relation of nodes in a cascade. The cascading context of one node consists of two aspects, including the potential influence it receives from earlier infected nodes and their temporal relative positions in a cascade. The other kind of cascading characteristic captured by DCE is the *cascading affinity*, which reveals the co-occurrence relation of nodes in cascades. Cascading affinity essentially reflects the probability that two nodes will be infected by the same cascade. Higher cascading affinity between two nodes indicates that it is more likely for them to co-occur in a cascade. Intuitively, the cascading characteristics of nodes reflect the effect of the unobservable underlying diffusion mechanisms and diffusion networks. Therefore, by explicitly preserving the node cascading characteristics, the learned embeddings also implicitly capture the effect of unobservable underlying diffusion mechanisms and diffusion network, which makes it feasible to make cascade predictions in terms of the similarity between embeddings in the latent space. As we will see later in the experiments, due to the ability to capture the cascading characteristics, the embeddings learned by DCE show a better performance in the task of infection prediction.

To effectively capture the non-linearity of information cascades, we introduce an *auto-encoder based collaborative embedding* architecture for DCE. DCE consists of multi-layer non-linear transformations by which the non-linear cascading patterns of nodes can be effectively encoded into the embeddings. DCE can learn embeddings for nodes in a collaborative way, where there are two kinds of collaborations, i.e., *cascade collaboration* and *node collaboration*. At first, in light of the observation that a node often participates in more than one cascade of different contents, for a node DCE can collaboratively encode its cascading context features in each cascade into its embedding. In other words, the embedding of a node is learned with the collaboration of the cascades the node participates, which we call the cascade collaboration. At the same time, DCE can concurrently embed the nodes, during which the embedding for a node is generated under the constraints of its relation to other nodes, i.e., its cascading affinity to other nodes and its neighborship in social networks. In other words, the embeddings of nodes are learned with the collaboration of each other, which we call the node collaboration.

The major contributions of this paper can be summarized as follows:

1. We propose a novel model called Deep Collaborative Embedding (DCE) for information cascade prediction without requirement of the knowledge about the underlying diffusion mechanism and the diffusion network. The node embeddings learned by DCE are beneficial to not only the infection prediction but also the prediction of infection order of nodes in a cascade.
2. We propose an auto-encoder based collaborative embedding framework for DCE, which can collaboratively learn the node embeddings, preserving the node cascading characteristics including cascading context and cascading affinity, as well as the structural property.
3. The extensive experiments conducted on real datasets verify the effectiveness of our proposed model.

The rest of this paper is organized as follows. We give the preliminaries in Section 2. The cascading context is defined and modeled in Section 3. In Section 4 we illustrate our proposed model and in Section 5 we analyze the experiments results. Finally, we briefly review the related work in Section 6 and conclude in Section 7.

2. Preliminaries and problem definition

2.1. Basic definitions

We denote a social network as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the nodes set comprising N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edges set. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ be the set of M information cascades. An information cascade C_m $(1 \leq m \leq M)$ observed on a social network $\mathcal{G}$ is defined as a set of timestamped infections, i.e., $C_m = \left\{(v, t_v^{(m)}) | v \in \mathcal{V} \wedge t_v^{(m)} < \infty\right\}$, where $(v, t_v^{(m)})$ represents node v is infected by cascade C_m at time $t_v^{(m)}$. We also say $v_i \in C_m$ if node v_i participates in cascade C_m. Additionally, we use $C_m(t) = \{(v, t_v^{(m)}) | v \in \mathcal{V} \wedge t_v^{(m)} < t\}$ to denote the set of nodes infected by cascade C_m before time t, and $\overline{C}_m(t) = \mathcal{V} \backslash C_m(t)$ the set of nodes which have not been infected before t. Note that the nodes in $\overline{C}_m(t)$ might or might not be infected by C_m after t. (See Table 1.)

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

Table 1
Notations.

Symbol	Description
N	The number of nodes
M	The number of cascades
$\mathcal{G}$	Network
$\mathcal{V}$	The set of nodes
$\mathcal{E}$	The set of edges
$\mathcal{C}$	The set of cascades
$\boldsymbol{X}^{(m)}$	The cascading context matrix of cascade C_m, $\boldsymbol{D}^{(m)} \in \mathbb{R}^{N\times N}$
$\boldsymbol{A}$	The cascading affinity matrix, $\boldsymbol{A} \in \mathbb{R}^{N\times N}$
$\boldsymbol{S}$	The structural proximity matrix, $\boldsymbol{S} \in \mathbb{R}^{N\times N}$
$t_v^{(m)}$	The infections time of node v_i in cascade C_m
$\boldsymbol{x}_v^{(m)}$	The row vector of node v in $\boldsymbol{X}^{(m)}$
$\boldsymbol{z}_v$	The learned embedding vector of node v

2.2. Problem definition

The target problem of this paper can be formulated as: given a set of information cascades $\mathcal{C} = (C_1, C_2, \ldots, C_M)$ observed on a given social network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we want to learn embeddings for nodes in $\mathcal{V}$, where the learned embeddings can preserve the cascading characteristics and structural property of nodes, so that closer embeddings indicate that the corresponding nodes are more likely to be infected by the same cascade with the closer infection time.

3. Modeling cascading characteristics

Cascading characteristics of a node reveal its relation to other nodes in information cascades, which are crucial to the prediction of node infection and infection order. In this section, we will define two kinds of cascading characteristics, the cascading context and the cascading affinity, which will be encoded into the learning embeddings.

3.1. Cascading context

As mentioned before, the cascading context of a node in a cascade is supposed to capture its temporal relation to other nodes in that cascade, which includes the potential influence imposed by other nodes and their temporal infection order. There are three factors we have to consider for the definition of cascading context. First, the infection of a node is intuitively caused by the potential influence of all the nodes infected before it, and the influence declines over time. Second, the cascading context should be specific to a cascade, as one node might have different cascading contexts in different cascades. Finally, in the same cascade, the infection of one node can be influenced neither by the nodes that are infected after it, nor by the nodes that are not infected at all. Based on these ideas, we can define the cascading context as follow:

Definition 1 (*Cascading Context*)**.** Given the set of M cascades on a social network $\mathcal{G}$ of N nodes, $\mathcal{C} = (C_1, C_2, \ldots, C_M)$, the cascading context of the nodes involved in cascade C_m $(1 \le m \le M)$ is defined as a matrix $\boldsymbol{X}^{(m)} \in \mathbb{R}^{N\times N}$. The entry at the uth row and the vth column of $\boldsymbol{X}^{(m)}$ represents the potential influence from node v to u, which is defined as

$$x_{u,v}^{(m)} = \begin{cases} \exp(-\frac{t_u^{(m)} - t_v^{(m)}}{\tau}) & , \quad t_v^{(m)} < t_u^{(m)}, \\ 0 & , \quad t_v^{(m)} \ge t_u^{(m)}, \end{cases} \tag{1}$$

where $t_u^{(m)}$ is the infection time of u in cascade C_m and τ is the decaying factor. The cascading context of node u in cascade C_m is defined as the row vector $\boldsymbol{x}_u^{(m)} = \boldsymbol{X}_{u,*}^{(m)}$.

As we will see later, $\boldsymbol{x}_u^{(m)}$ will be fed into our model as it quantitatively captures u's temporal relation (including the influence and the relative infection position) to the other nodes in a cascade C_m.

3.2. Cascading affinity

As mentioned before, cascading affinity of two nodes measures the similarity of them with respect to the cascades, which can be defined in terms of their co-occurrences in historical cascades as follow:

Definition 2 (*Cascading Affinity*)**.** Given the set of M cascades on a social network $\mathcal{G}$ of N nodes, i.e., $\mathcal{C} = (C_1, C_2, \ldots, C_M)$, the cascading affinity of two nodes u and v is represented by the entry at the uth row and the vth column of the cascading affinity matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, which is defined as the ratio of the cascades involving both u and v, i.e.,

$$a_{u,v} = \frac{|\{C_k | u \in C_k, v \in C_k, C_k \in \mathcal{C}\}|}{|\mathcal{C}|}. \tag{2}$$

Definition 2 tells us that for two given nodes, the more number of cascades involving both of them, the higher their cascading affinity, and intuitively the more similar their preferences to the contents of cascades. In this sense, cascading affinity of two nodes implies that how close their embeddings should be in the latent space.

4. Deep collaborative embedding

In this paper, we propose an auto-encoder based Deep Collaborative Embedding (DCE) model, which can learn embeddings for nodes in a given social network, based on the M cascades $C_1, \ldots, C_M$ observed on the network, so that the learned embeddings can be used for cascade prediction without knowing the underlying diffusion mechanisms and the explicit diffusion networks. In this section, we first present the architecture of the Deep Collaborative Embedding (DCE) model in detail, and then we describe the objective function and the learning of DCE.

4.1. Architecture of DCE

The architecture of DCE is shown in Fig. 1. As we can see from Fig. 1, DCE learns the embeddings through two collaborations, the cascade collaboration and the node collaboration. With the cascade collaboration, DCE can generate the result d-dimensional embedding $\boldsymbol{z}_v \in \mathbb{R}^d$ for a node v by collaboratively encoding its M cascading contexts, $\boldsymbol{x}_v^{(m)}$ $(1 \le m \le M)$. At first, DCE will learn M intermediate embeddings $\boldsymbol{y}_v^{(1)}, \ldots, \boldsymbol{y}_v^{(M)}$ for v by M auto-encoders, respectively, each of which corresponds to a cascade. The auto-encoder for cascade C_m $(1 \le m \le M)$ takes the v's cascading context $\boldsymbol{x}_v^{(m)}$ in the cascade C_m as input, and then generates the intermediate embedding of v in cascade C_m, $\boldsymbol{y}_v^{(m)}$, through its encoder part consisting of L non-linear hidden layers defined by the following equations:

$$\begin{aligned} \boldsymbol{y}_v^{(m),1} &= \sigma\left(\boldsymbol{W}^{(m),1}\boldsymbol{x}_v^{(m)} + \boldsymbol{b}^{(m),1}\right), \\ \boldsymbol{y}_v^{(m),l} &= \sigma\left(\boldsymbol{W}^{(m),l}\boldsymbol{y}_v^{(m),l-1} + \boldsymbol{b}^{(m),l}\right), \quad \forall l \in \{2, 3, \ldots, L\}, \end{aligned} \tag{3}$$

where $\boldsymbol{y}_v^{(m),l}$ is the output vector of lth hidden layer of mth auto-encoder taking $\boldsymbol{x}_v^{(m)}$ as input, $\boldsymbol{W}^{(m),l}$ is the parameter matrix of that layer, and $\boldsymbol{b}^{(m),l}$ is the corresponding bias.

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

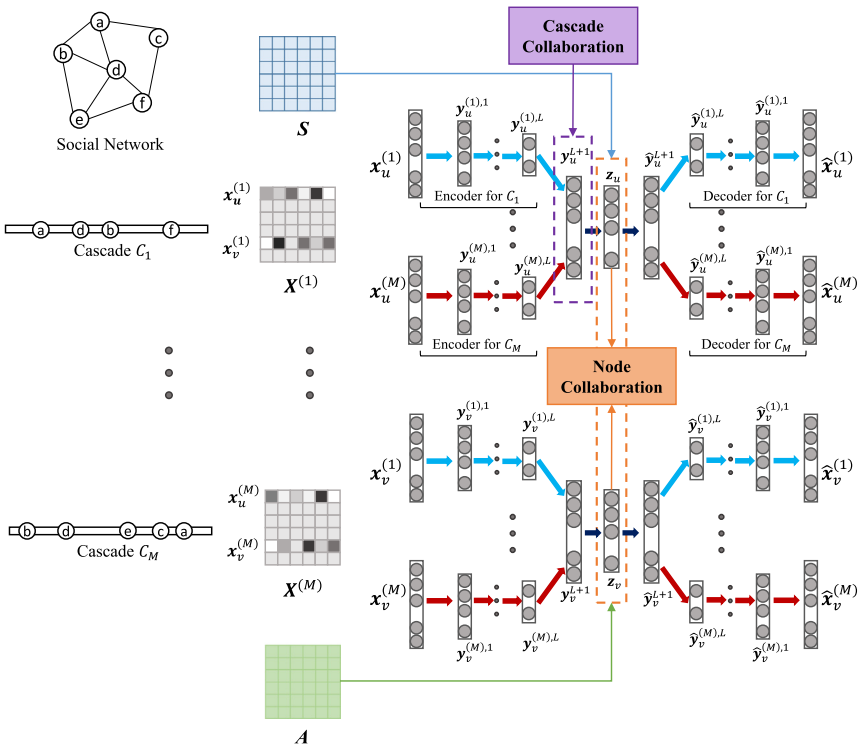


Fig. 1. Architecture of DCE.

At last, the result embedding $\boldsymbol{z}_v$ is generated by fusing the M intermediate embeddings $\boldsymbol{y}_v^{(m),L}$ $(1 \le m \le M)$ through the following non-linear mappings:

$$\begin{aligned} \boldsymbol{y}_v^{L+1} &= \sigma\left(\sum_{m=1}^{M}(\boldsymbol{W}^{(m),L+1}\boldsymbol{y}_v^{(m),L} + \boldsymbol{b}^{(m),L+1})\right), \\ \boldsymbol{z}_v &= \sigma\left(\boldsymbol{W}^{L+2}\boldsymbol{y}_v^{L+1} + \boldsymbol{b}^{L+2}\right). \end{aligned} \tag{4}$$

Symmetrically, the decoder part of the auto-encoder for cascade C_m is defined by the following equations:

$$\begin{aligned} \hat{\boldsymbol{y}}_v^{L+1} &= \sigma\left(\hat{\boldsymbol{W}}^{L+2}\boldsymbol{z}_v + \hat{\boldsymbol{b}}^{L+2}\right), \\ \hat{\boldsymbol{y}}_v^{(m),L} &= \sigma\left(\hat{\boldsymbol{W}}^{(m),L+1}\hat{\boldsymbol{y}}_v^{L+1} + \hat{\boldsymbol{b}}^{(m),L+1}\right), \\ \hat{\boldsymbol{y}}_v^{(m),l-1} &= \sigma\left(\hat{\boldsymbol{W}}^{(m),l-1}\hat{\boldsymbol{y}}_v^{(m),l} + \hat{\boldsymbol{b}}^{(m),l-1}\right), \quad \forall l \in \{2, 3, \ldots, L\}, \\ \hat{\boldsymbol{x}}_v^{(m)} &= \sigma\left(\hat{\boldsymbol{W}}^{(m),1}\hat{\boldsymbol{y}}_v^{(m),1} + \hat{\boldsymbol{b}}^{(m),1}\right). \end{aligned} \tag{5}$$

In the above Eqs. (3), (4), and (5), the parameter matrices $\boldsymbol{W}$ and $\hat{\boldsymbol{W}}$, and the bias vectors $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ are the parameters that will be learned from training data.

At the same time, with the node collaboration, DCE can concurrently embed the nodes into latent space, by which the similarity between nodes in the social network can be captured into the learned embeddings. Particularly, to regulate the closeness between any two embeddings $\boldsymbol{z}_u$ and $\boldsymbol{z}_v$, DCE will impose the constraints of the cascading affinity $a_{u,v}$ and structural proximity $s_{u,v}$ between u and v via Laplacian Eigenmaps, which will be described in detail in next subsection.

4.2. Optimization objective of DCE

4.2.1. Loss function for cascade collaboration

At first, as described in last subsection, M auto-encoders defined by Eqs. (3), (4), and (5) fulfill the cascade collaboration for embedding v by reconstructing its M cascading contexts $\boldsymbol{x}_v^{(m)}$. The optimization objective for this part is to minimize the reconstruction error between $\boldsymbol{x}_v^{(m)}$ and $\hat{\boldsymbol{x}}_i^{(m)}$, of which the loss function is defined as follow:

$$\begin{aligned} \mathcal{L}_x &= \sum_{m=1}^{M}\sum_{v\in\mathcal{V}}\left\|(\boldsymbol{x}_v^{(m)} - \hat{\boldsymbol{x}}_v^{(m)})\right\|_2^2 \\ &= \sum_{m}^{M}\left\|(\boldsymbol{X}^{(m)} - \hat{\boldsymbol{X}}^{(m)})\right\|_F^2, \end{aligned} \tag{6}$$

where $\boldsymbol{X}^{(m)}$ and $\hat{\boldsymbol{X}}^{(m)}$ are the original cascading context matrix and the reconstructed cascading context matrix of cascade C_m, respectively, which are defined in Definition 1.

The cascading context vectors $\boldsymbol{x}_v^{(m)}$ are often sparse, which may leads to undesired $\mathbf{0}$ vectors in the embeddings $\boldsymbol{z}_v$ and the reconstructed $\boldsymbol{x}_v^{(m)}$ if the sparse vectors $\boldsymbol{x}_v^{(m)}$ are straightforwardly fed into DCE. To overcome this issue, inspired by the idea used in the existing works [20,26] which assign more penalty (corresponding to larger weight) to the loss incurred by non-zero elements than that incurred by zero elements, the $\mathcal{L}_x$ can be redefined as

$$\begin{aligned} \mathcal{L}_x &= \sum_{m}^{M}\sum_{v\in\mathcal{V}}\left\|(\boldsymbol{x}_v^{(m)} - \hat{\boldsymbol{x}}_v^{(m)}) \odot \boldsymbol{p}_v^{(m)}\right\|_2^2 \\ &= \sum_{m}^{M}\left\|(\boldsymbol{X}^{(m)} - \hat{\boldsymbol{X}}^{(m)}) \odot \boldsymbol{P}^{(m)}\right\|_F^2, \end{aligned} \tag{7}$$

where $\odot$ denotes the Hadamard product, and the uth column vector of the matrix $\boldsymbol{P}^{(m)} \in \mathbb{R}^{N\times N}$ is the weight vector $\boldsymbol{p}_u^{(m)} = \{p_{u,v}^{(m)}\}_{v\in\mathcal{V}}$ assigned to cascading context $\boldsymbol{x}_u^{(m)}$. An entry $p_{u,v}^{(m)} = \rho > 1$ if $x_{u,v}^{(m)} \neq 0$, otherwise $p_{u,v}^{(m)} = 1$.

4.2.2. Loss functions for node collaboration

Next we introduce the loss function for node collaboration. As mentioned in last subsection, through the node collaboration the embeddings $\boldsymbol{z}_i$ will preserve the cascading affinity of nodes

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

in cascades and the structural proximity of nodes in social network. Following the idea of Laplacian Eigenmaps, we weight the similarity between two embeddings with the cascading affinity of their corresponding nodes, which leads to the following loss function:

$$\mathcal{L}_a = \sum_{u,v\in\mathcal{V}} a_{u,v} \left\| \boldsymbol{z}_u - \boldsymbol{z}_v \right\|_2^2, \tag{8}$$

where $a_{u,v}$ is the cascading affinity between u and v defined in Eq. (2). The insight of Eq. (8) is that a penalty will be imposed when two nodes with high cascading affinity are relocated far away in the latent space.

Similarly, we also weight the similarity between two embeddings with the structural proximity of their corresponding nodes, which leads to the following loss function:

$$\mathcal{L}_s = \sum_{u,v\in\mathcal{V}} s_{u,v} \left\| \boldsymbol{z}_u - \boldsymbol{z}_v \right\|_2^2, \tag{9}$$

where $s_{u,v}$ is the structural proximity between u and v in social network. Note that it does not matter how to define $s_{u,v}$, and theoretically, the node structural proximity of any order can be used for $s_{u,v}$. In this paper, we employ the first-order proximity [23] to define $s_{u,v}$. To be more specific, $s_{u,v} = 1$ if u and v are connected by a link in the network, otherwise $s_{u,v} = 0$.

Let $\boldsymbol{L}^{(a)}$ be the laplacian matrix of the cascading affinity matrix $\boldsymbol{A}$, i.e., $\boldsymbol{L}^{(a)} = \boldsymbol{D}^{(a)} - \boldsymbol{A}$, where $\boldsymbol{D}^{(a)}$ is diagonal and $\boldsymbol{D}^{(a)}_{u,u} = \sum_v^N a_{u,v}$. Let $\boldsymbol{S}$ be the structural proximity matrix whose entry at uth row and vth column is $s_{u,v}$, and similarly, let $\boldsymbol{L}^{(s)}$ be its laplacian matrix, i.e., $\boldsymbol{L}^{(s)} = \boldsymbol{D}^{(s)} - \boldsymbol{A}$, where $\boldsymbol{D}^{(s)}$ is also diagonal and $\boldsymbol{D}^{(s)}_{u,u} = \sum_{v\in\mathcal{V}} s_{u,v}$. Then we can rewrite the Eqs. (8) and (9) with their matrix forms:

$$\mathcal{L}_a = 2tr(\boldsymbol{Z}^T \boldsymbol{L}^{(a)} \boldsymbol{Z}), \tag{10}$$

and

$$\mathcal{L}_s = 2tr(\boldsymbol{Z}^T \boldsymbol{L}^{(s)} \boldsymbol{Z}), \tag{11}$$

where $\boldsymbol{Z}$ is the embedding matrix whose ith column is $\boldsymbol{z}_i$.

4.2.3. The complete loss function

By combining $\mathcal{L}_x$, $\mathcal{L}_a$, and $\mathcal{L}_s$, we can define the **complete loss function** of DCE as follow:

$$\mathcal{L} = \mathcal{L}_x + \alpha\mathcal{L}_a + \beta\mathcal{L}_s + \gamma\mathcal{L}_{reg}, \tag{12}$$

where $\mathcal{L}_{reg} = \sum_l^{L+2} \sum_m^M (\left\| \boldsymbol{W}^{(m),l} \right\|_2^2 + \left\| \hat{\boldsymbol{W}}^{(m),l} \right\|_2^2)$ is a $\mathcal{L}2$-norm regularizer term to avoid overfitting, and α, β, and γ are nonnegative parameters used to control the contributions of the terms.

4.3. Learning of DCE

DCE model can be learned using Stochastic Gradient Descent (SGD), the gradients of which are given by the follow equations:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(m),l}} = &\sum_m^M 2(\hat{\boldsymbol{X}}^{(m)} - \boldsymbol{X}^{(m)}) \odot \boldsymbol{P}^{(m)} \cdot \frac{\partial \hat{\boldsymbol{X}}^{(m)}}{\partial \boldsymbol{W}^{(m),l}} + \gamma \cdot \frac{\boldsymbol{W}^{(m),l}}{2} + \\ &\alpha \cdot (2(\boldsymbol{L}^{(a)} + \boldsymbol{L}^{(a)^T}) \cdot \boldsymbol{Z}) \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{(m),l}} + \\ &\beta \cdot (2(\boldsymbol{L}^{(s)} + \boldsymbol{L}^{(s)^T}) \cdot \boldsymbol{Z}) \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{W}^{(m),l}},\end{aligned} \tag{13}$$

Algorithm 1 learning algorithm of DCE

Input:
The set of cascading context matrices $\mathcal{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots \boldsymbol{X}_M)$, cascading affinity matrix $\boldsymbol{A}$, structural proximity matrix $\boldsymbol{S}$, and the parameters α, β and γ.

Output:
Node embeddings $\boldsymbol{Z}$.

1: Initialize parameters $\boldsymbol{W}$, $\hat{\boldsymbol{W}}$, $\boldsymbol{b}$, and $\hat{\boldsymbol{b}}$.
2: **repeat**
3: Compute $\boldsymbol{Z}$, $\hat{\boldsymbol{X}}$ according to Eqs. (3)–(5).
4: Compute total loss $\mathcal{L}$ according to Eq. (12).
5: Update $\boldsymbol{W}$, $\hat{\boldsymbol{W}}$, $\boldsymbol{b}$, and $\hat{\boldsymbol{b}}$ according to Eqs. (13) to (16) using SGD.
6: **until** $\mathcal{L}$ converges.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}^{(m),l}} = &\sum_m^M 2(\hat{\boldsymbol{X}}^{(m)} - \boldsymbol{X}^{(m)}) \odot \boldsymbol{P}^{(m)} \cdot \frac{\partial \hat{\boldsymbol{X}}^{(m)}}{\partial \boldsymbol{b}^{(m),l}} + \gamma \cdot \frac{\boldsymbol{b}^{(m),l}}{2} + \\ &\alpha \cdot (2(\boldsymbol{L}^{(a)} + \boldsymbol{L}^{(a)^T}) \cdot \boldsymbol{Z}) \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{b}^{(m),l}} + \\ &\beta \cdot (2(\boldsymbol{L}^{(s)} + \boldsymbol{L}^{(s)^T}) \cdot \boldsymbol{Z}) \cdot \frac{\partial \boldsymbol{Z}}{\partial \boldsymbol{b}^{(m),l}},\end{aligned} \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{W}}^{(m),l}} = \sum_m^M 2(\hat{\boldsymbol{X}}^{(m)} - \boldsymbol{X}^{(m)}) \odot \boldsymbol{P}^{(m)} \cdot \frac{\partial \hat{\boldsymbol{X}}^{(m)}}{\partial \hat{\boldsymbol{W}}^{(m),l}} + \gamma \cdot 2\hat{\boldsymbol{W}}^{(m),l}, \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{b}}^{(m),l}} = \sum_m^M 2(\hat{\boldsymbol{X}}^{(m)} - \boldsymbol{X}^{(m)}) \odot \boldsymbol{P}^{(m)} \cdot \frac{\partial \hat{\boldsymbol{X}}^{(m)}}{\partial \hat{\boldsymbol{b}}^{(m),l}} + \gamma \cdot 2\hat{\boldsymbol{b}}^{(m),l}, \tag{16}$$

where the partial derivatives on the right side of the equations can be computed using back-propagation.

The learning process is given in Algorithm 1. Note that in each iteration, the parameters are updated (Line 5) once the embeddings $\boldsymbol{z}_v$, $v \in \mathcal{V}$ are concurrently generated (Line 3). Such concurrent embedding scheme ensures the cascading context can be encoded into the embeddings as well as the cascading affinity and the structural proximity of nodes can be preserved at the same time.

5. Experiments

In this section, we will present the details of experiments conducted on real-world datasets. The experiments include two parts, the tuning of the hyper-parameters and the verifying of DCE. Particularly, to verify the effectiveness of DCE, we will check whether the embeddings learned by DCE improve the performance of the prediction of information cascades on the real world datasets.

5.1. Settings

5.1.1. Datasets

We verify the effectiveness of our method through experiments conducted on three real datasets, Digg, Twitter, and Weibo, which are described as follows:

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

Digg is a website where users can submit stories and vote for the stories they like [27]. The dataset extracted from Digg contains 3553 stories, 139,409 users, and 3,018,197 votes with timestamps. A vote for a story is treated as an infection of that story, and the votes for the same story constitute a cascade. In addition, a social link exists between two users if one of them is watching or is a fan of the other one.

Twitter is a social media network which offers microblog service [28]. The dataset extracted from Twitter comprises 510,795 users and 12,054,205 tweets with timestamps, where each tweet is associated with a hashtag. If the hashtag is adopted in one user's tweet, we consider it infects that user. The tweets sharing the same hashtag are treated as a cascade, and 1,345,913 cascades are contained in the dataset. In addition, the users are linked by their following relationships.

Weibo is a Twitter-like social network [29]. The dataset extracted from Weibo contains 1,340,816 users and their 31,444,325 tweets with timestamps. A retweeting action of a user is viewed as an infection of the retweeted tweet to that user. The retweetings of the same tweet constitute a cascade, and the dataset contains 232,978 cascades of different tweets. The users in Weibo network are also connected by following relationships.

The statistics of the datasets are summarized in Table 2. On each dataset, we randomly select 60% of the total cascades as training set, 20% as validating set, and the remaining 20% as testing set.

5.1.2. Baselines

In order to demonstrate the effectiveness of DCE, we compare it with the following baseline methods:

NetRate NetRate is a generative cascade model which exploits infection times of nodes without assumptions on the network structure [30]. It models information diffusion process as discrete networks of continuous temporal process occurring at different rates, and then infers the edges of the global diffusion network and estimates the transmission rates of each edge that best explain the observed data.

CDK CDK maps nodes participating in information cascades to a latent representation space using a heat diffusion process [10]. It treats learning diffusion as a ranking problem and learns heat diffusion kernels that defines, for each node of the network, its likelihood to be reached by the diffusing content, given the initial source of diffusion. Here we adopt the without-content version of CDK considering that other baselines and our approach are not designed to deal with diffusion content.

Topo-LSTM Topo-LSTM uses directed acyclic graph as the diffusion topology to explore the diffusion structure of cascades rather than regarding it as merely a sequence of nodes ordered by their infection timestamps [8]. Then it puts dynamic DAGs into a LSTM-based model to generate topology-aware embeddings for nodes as outputs. The infection probability at each time step will be computed according to the embeddings.

Embedded-IC Embedded-IC is a representation learning technique for inference of Independent Cascade (IC) model [12]. Embedded-IC can embed users in cascades into a latent space and infer the diffusion probability between users based on the relative positions of the users in the latent space.

DCE-C DCE-C is a special version of the proposed DCE, where the node collaborations of cascading affinity and structural proximity are removed while only the cascade collaboration of cascading contexts is kept.

5.2. Cascade prediction

In this paper, we evaluate the learned embeddings by applying them to the task of information cascade prediction, the details of which are described as follows.

For a testing cascade C, given a set of seed nodes which are infected before, we predict the infection probabilities for the remaining nodes and their infecting order. To be more specific, the size of the seed set will be 1% of the total number of the nodes. Let $V_t \subset \mathcal{V}$ be the set of nodes that are predicted before time step $t+1$, and then the probability that one node $u \in \mathcal{V} \backslash V_t$ will be infected at $t+1$ is

$$P(u|V_t) = 1 - \prod_{v \in V_t} \big(1 - P(u|v)\big), \tag{17}$$

where $P(u|v)$ is the probability that u is infected by v. Our idea of computing $P(u|v)$ is based on the similarity between the embeddings, which is defined as

$$P(u|v) = \frac{1}{1 + \exp\big(\|\boldsymbol{z}_v - \boldsymbol{z}_u\|_2^2\big)}, \tag{18}$$

where $\boldsymbol{z}_u$ and $\boldsymbol{z}_v$ are embedding vectors of nodes u and v, respectively, and the similarity is measured by Euclidean distance. For each uninfected node $u \in \overline{C}(t)$, its infection probability can be computed according to Eq. (17), and we can obtain a list $\hat{R}_C$ of the nodes in descending order of their infection probabilities. Comparing $\hat{R}_C$ with the ground truth R_C, we can evaluate the performance of the prediction with two metrics, Mean Average Precision (MAP) and order-Precision.

As a metric originating from information retrieval, MAP can evaluate the prediction of information cascades by taking positions of nodes in the predicting list into consideration. We first define the top-n precision of $\hat{R}_C$ as the hit rate of the first n nodes of $\hat{R}_C$ over the ground truth, i.e.,

$$p_{C,n} = \frac{|\hat{R}_{C,n} \cap R_C|}{|\hat{R}_{C,n}|}, \tag{19}$$

where $\hat{R}_{C,n}$ is the set of first n nodes of $\hat{R}_C$. Then based on $p_{C,n}$, we can define the average precision of $\hat{R}_C$ as

$$AP_C = \frac{\sum_{v \in R_C} p_{C,r_{C,v}}}{|R_C|}, \tag{20}$$

where $r_{C,v}$ denotes the rank of node v in $\hat{R}_C$ and $p_{C,r_{C,v}}$ is the top-$r_{C,v}$ precision of $\hat{R}_C$. From Eqs. (19) and (20) we can see that, it will lead to a low AP_C if too many nodes which occur in R_C but rank low in $\hat{R}_C$. What is more, we set the size k of the predicted list $\hat{R}_C$ in {100, 300, 500, 700, 900} to compute $AP_C@k$ among the first k nodes. Finally, $MAP@k$ can be defined as the average of $AP_C@k$ over testing set $\mathcal{C}_t$, i.e.,

$$MAP@k = \frac{1}{|\mathcal{C}_t|} \sum_{C \in \mathcal{C}_t} AP_C@k. \tag{21}$$

To evaluate the prediction of infection order, we propose a new metric, order-Precision, which is defined as

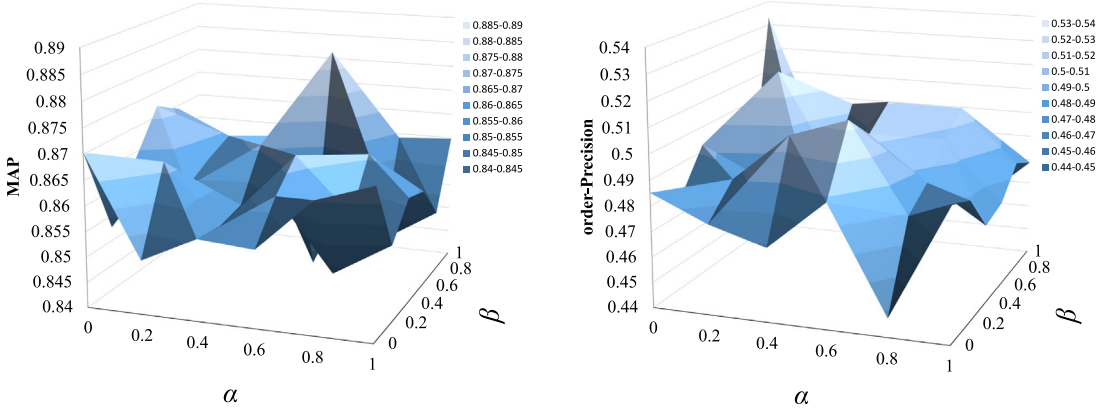
$$P_o = \frac{1}{|\mathcal{C}_t|} \sum_{C \in \mathcal{C}_t} \frac{1}{|R_C|} \sum_{v \in R_C \cap \hat{R}_C} \frac{|\hat{R}_C(\hat{t}_v^C) \cap R_C(t_v^C)|}{|\hat{R}_C(\hat{t}_v^C) \cap R_C|}, \tag{22}$$

where t_v^C is the true infection time of v and $\hat{t}_v^C$ is the predicted one, and $R_C(t_v^C)$ and $\hat{R}_C(\hat{t}_v^C)$ denotes the sets of nodes infected before node v_v in the ground truth list and the predicted list respectively. The idea of Eq. (22) is that the more nodes with more similar relative orders of nodes in R_C and $\hat{R}_C$, the higher the order-Precision of $\hat{R}_C$. First, to evaluate the similarity of node v's relative orders in R_C and $\hat{R}_C$, we consider a heuristic indicator, the number of the nodes that are infected before node v and shared by R_C and $\hat{R}_C$, i.e., $|\hat{R}_C(\hat{t}_v^C) \cap R_C(t_v^C)|$, and the larger this number is, the more similar the relative orders will be. Then we can obtain

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

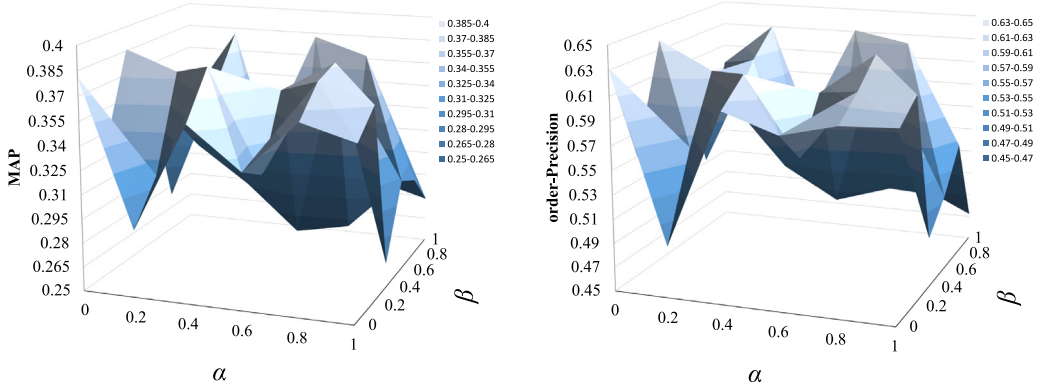
Table 2
The statistics of datasets.

Dataset	#Nodes	#Links	Avg. degree	#Cascades	#Infections	Avg. cascade length
Digg	139,409	1,731,658	12.4	3,553	3,018,197	849.5
Twitter	510,795	14,273,311	27.9	1,345,913	12,054,205	9.0
Weibo	1,340,816	308,489,739	230.1	232,978	31,444,325	135.0



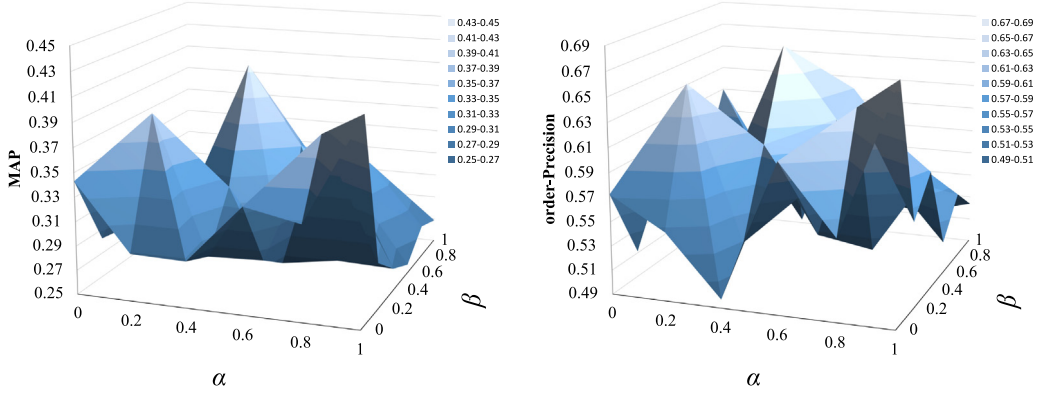
(a) MAP (b) order-Precision

Fig. 2. Tuning the parameter α and β on Digg.



(a) MAP (b) order-Precision

Fig. 3. Tuning the parameter α and β on Twitter.



(a) MAP (b) order-Precision

Fig. 4. Tuning the parameter α and β on Weibo.

Table 3
MAP@k on Digg, Twitter and Weibo datasets.

Dataset	Method	MAP@k (%)				
		@100	@300	@500	@700	@900
Digg	NetRate	1.108	5.749	10.933	16.618	24.043
	CDK	27.951	39.766	52.032	65.220	80.408
	Embedded-IC	2.084	9.073	23.314	47.249	78.066
	Topo-LSTM	2.444	17.535	25.812	42.779	69.534
	DCE-C	32.356	55.308	63.546	66.823	86.879
	DCE	**47.497**	**72.952**	**76.694**	**84.250**	**91.362**
Twitter	NetRate	0.140	2.550	6.724	15.058	30.572
	CDK	9.512	22.724	34.701	48.162	63.315
	Embedded-IC	0.751	4.740	12.568	24.985	43.347
	Topo-LSTM	0.665	5.084	13.681	26.083	42.050
	DCE-C	15.983	27.846	37.427	53.617	65.858
	DCE	**16.376**	**29.773**	**40.690**	**56.301**	**69.863**
Weibo	NetRate	0.469	2.696	7.724	15.280	25.583
	CDK	1.124	11.510	25.348	41.810	54.429
	Embedded-IC	0.185	3.988	9.706	18.965	30.738
	Topo-LSTM	0.005	0.268	2.204	7.084	19.774
	DCE-C	3.466	28.526	52.084	62.684	71.339
	DCE	**10.506**	**30.986**	**53.555**	**64.533**	**72.746**

the relative order similarity for one single testing cascade C by taking the average over all nodes shared by R_C and $\hat{R}_C$. Finally, the overall order-Precision is the average of the relative order similarities over all testing cascades in $\mathcal{C}_t$.

5.3. Hyper-parameter tuning

In this subsection, we investigate the hyper-parameters α and β in Eq. (12) on the validation set, which control the influence of the cascading affinity and the structure proximity on the embedding learning, respectively.

For simplicity, we fix $\gamma = 0.002$ and adopt a grid search in the range of [0, 1] with a step size of 0.2 to determine the optimal values of α and β. Figs. 2, 3, and 4 show the results of MAP and order-Precision over different combinations of α and β on three datasets. Through a comprehensive comparison, we can find that, in most cases the MAPs and order-Precisions at non-zero α and β are better than those at zero α and β. Taking Fig. 2(a) as an instance, the MAP value at (0.6, 0.8) is 0.8835, which is higher than 0.8703 at (0.0, 0.0). It verifies that appropriately applying cascading affinity and structural proximity as constrains can improve the learned embeddings for information cascade prediction. The combinations of α and β at which the sum of MAP and order-Precision achieve the highest are chosen for the remaining experiments. Based on this criterion, we set (α, β) as (0.1, 0.9) for Digg, (0.6, 0.8) for Twitter, and (0.8, 0.2) for Weibo.

5.4. Effectiveness

In this section, we will analyze the experiments results in the tasks of infection prediction and infection order prediction, which are presented in Table 3 and Fig. 5 respectively.

5.4.1. Infection prediction

Table 3 gives the MAPs of different methods for infection prediction task, with the best ones in each case being boldfaced. From Table 3 we can make some analyses as follows:

1. The proposed DCE-C and DCE always outperform all baselines, giving improvements on the best baselines by 5.989% (Twitter, MAP@500) to 33.186% (Digg, MAP@300) relatively across all datasets. We can also find that DCE achieves better results than DCE-C in every case, and it proves that by using node collaborations as constrains, DCE can better characterize relations between nodes, which are important in information cascades.
2. The results show that, through collaboratively mapping the nodes into a latent space with a deep architecture, DCE can better capture deep and non-linear features of nodes in information cascades than Netrate, which estimates infection probability directly with a shallow probabilistic model.
3. In contrast with embedding baselines CDK, Embedded-IC, and Topo-LSTM, DCE's deep collaborative embedding architecture can better preserve the cascading characteristics and structural properties of nodes, which are crucial for infection prediction. Unlike CDK which assumes unrealistically that information diffusion is driven by the relations between source node and the others, in DCE all infected nodes are thought to have potential influence on the not yet infected ones and cascading context is employed to model their temporal relations. And as DCE makes no assumption about the underlying diffusion mechanism, it can better utilize the cascading contexts of nodes than Embedded-IC which is based on the IC model. Compared with Topo-LSTM that also adopts a deep model, DCE does not rely on the knowledge of the underlying diffusion network, which is usually difficult to obtain.

5.4.2. Infection order prediction

In Fig. 5 the order-Precisions of different methods for infection order prediction are presented, based on which several analyses can be made as follows:

1. We can see that the proposed DCE-C and DCE achieve best performance in all three datasets. The reason is that with the proposed cascading context, DCE is able to not only better preserve the temporal relations, but also better capture the infection order characteristics in information cascades than baselines. And DCE's superior results over DCE-C reveals that, even though cascading affinity and structural property do not indicate nodes infection orders explicitly, they can lead to further improvements when they are used as constrains in DCE.
2. To be more specific, NetRate is incapable of capturing the infection order features with its shallow probabilistic model. While CDK exploits heat diffusion kernel to formulate a ranking problem, where infection orders are kind of modeled, it cannot fully characterize node infection order features like the proposed cascading context in DCE. For embedded-IC, nodes infection orders do not get any attention in this IC-based model and certainly cannot be captured, which results in its bad performance. Notwithstanding Topo-LSTM's adoption of diffusion topology can encode the nodes infection orders to some extent, it still cannot get rid of the dependence on the underlying diffusion network, which cannot always be satisfied.

6. Related work

In this section, we briefly review two lines of related works with our research, including network embedding and information cascade prediction.

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

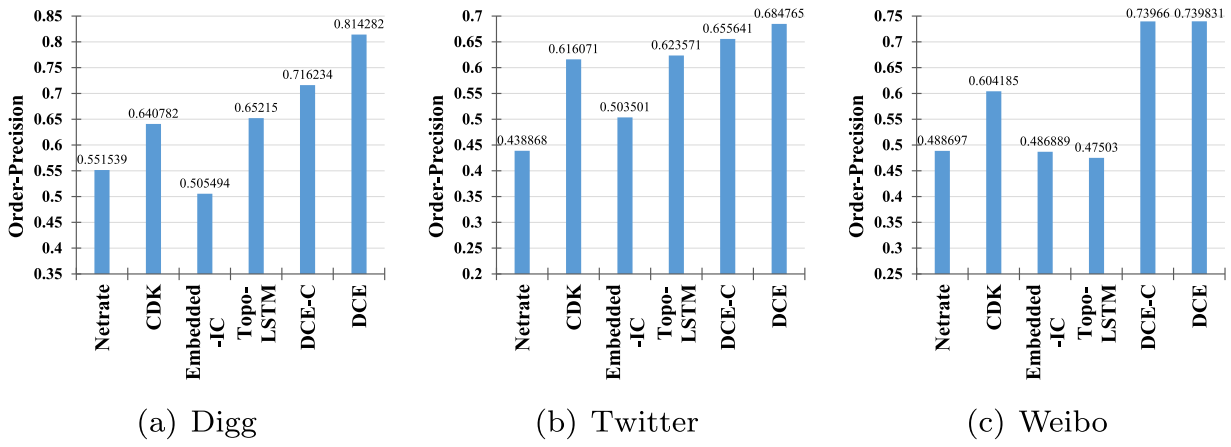


Fig. 5. Order-Precision on Digg, Twitter and Weibo datasets.

6.1. Network embedding

With the wide employment of embedding methods in various machine learning tasks [31–35], network embedding also gains more and more attentions and applications [36,37]. Network embedding refers to assigning nodes in a network to low-dimensional representations and effectively preserving the network structure [36]. Intuitively, nodes can be represented by their correspondent raw or column feature vectors in the adjacent matrix of a network. However, sometimes these vectors are sparse with high dimensions, which brings challenges to machine learning tasks. As a result, a set of traditional network embedding methods [38–41] are proposed mainly for dimension reduction. Nevertheless, these methods can only work well on networks of relatively small sizes and suffer from high computation cost when coping with online social networks with huge numbers of nodes.

Recent works like DeepWalk [25] and LINE [23] are proposed to learn low-dimensional representations for nodes through an optimization process instead of directly transforming the original feature vectors, where the scaling problem also can be well handled. Inspired by word2vec [31,32], DeepWalk considers the nodes in network as the words in natural language and utilizes random walks to generate node sequences, based on which the node representations are learned following the procedure of word2vec. As a more generalized version of DeepWalk, node2vec is proposed in [42] with biased random walks to control the generation of nodes' contexts more flexibly. LINE produces embeddings for nodes with the expectation to preserve both the first-order and second-order proximities of the network neighborhood structure. Under the influence of these researches, a collection of network embedding methods are proposed for different scenarios. For instances, [43] modifies DeepWalk for heterogeneous networks by introducing meta-path based random walks, and [44] incorporates a harmonious embedding matrix to further embed the embeddings that only encode intra-network edges. As the deep neural network has shown remarkable effectiveness in many machine learning tasks, there also emerges a series of works which perform network embedding with a deep model. For example, [20] adopts a semi-supervised deep autoencoder model to exploit the first-order and second-order proximities jointly to preserve the network structure. [21] learns nodes representations by keeping both the structural proximity and attribute proximity with a designed multilayered perceptron framework. And in [22], the researchers use a highly nonlinear multilayered embedding function to capture the complex interactions between the heterogeneous data in a network.

However, most of these network embedding methods [24,45,46] are not applicable to information cascade prediction. In our work, we employ an auto-encoder based collaborative embedding architecture to learn embeddings from nodes' cascading contexts with constrains.

6.2. Information cascade prediction

Information cascade phenomena have been widely investigated in the context of epidemiology, physical science and social science, and the development of online social network has greatly promoted related researches [47,4,14]. Most of the early researches [48] analyze information cascade based on fixed models, the representatives among which are Independent Cascade (IC) [17] model and the Linear Threshold (LT) [49] model. Classic IC model treats the diffusion activity of information as cascades while the LT model determines infections of users according to thresholds of the influence pressure incoming from the neighborhood. Both of them can be unified into a same framework [48], and a series of extension work has been proposed [6,50,51,7,52–54]. For example, [50] extends the IC model to formulate a generative model that can take time delay into consideration. However, information diffusion processes are so complicated that we seldom exactly know the underlying mechanisms of how information diffuses. What is more, these works are often based on the assumption that the explicit paths along which information propagates between nodes are observable, which is difficult to satisfy.

A collection of methods are proposed to infer the most possible links that can best explain the observed diffusion cascades without knowing the explicit paths. For instance, NetInf [11] and Connie [55] use greedy algorithms to find a fixed number of links between users that maximize the likelihood of a set of observed diffusions under an IC-like diffusion hypothesis. And a more general framework called NetRate [30] has been proposed, which also occurs in our experiments as a baseline. NetRate models information diffusion process as discrete networks of continuous temporal process occurring at different rates, and then infers the edges of the global diffusion network and estimates the transmission rates of each edge that best explain the observed data [30]. There are also further variants of this framework being proposed [56,57]. However, most of these works still rely on the assumption that information diffusion follows a parametric model.

In recent years, a set of researches [10,9,12,8,58] which adopt network embedding techniques to handle information cascade prediction have be proposed. These methods usually embed nodes in a latent feature space, then the diffusion probabilities between nodes are computed based on their positions in the

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

space. CDK proposed in [10] treats information diffusion as a ranking problem and maps nodes to a latent space using a heat diffusion process. However, it assumes the infected nodes orders of a cascade is influenced by the relations between source node and the other nodes, which is not realistic. [12] follows the mechanism of IC model to embed users in cascades into a latent space. [8] puts dynamic directed acyclic graphs into an LSTM-based model to generate topology-aware embeddings for nodes, which depends a lot on the network structure information. In contrast, our proposed method DCE collaboratively embed the nodes with a deep architecture into a latent space, without requirement of the knowledge about the underlying diffusion mechanisms and the explicit paths of diffusions on the network structure.

7. Conclusions

In this paper, we address the problem of information cascade prediction in online social networks with the network embedding techniques. We propose a novel model called Deep Collaborative Embedding (DCE) for information cascade prediction which can learn embeddings for not only infection prediction but also infection order prediction in a cascade, without the requirement to know the underlying diffusion mechanisms and the diffusion network. We propose an auto-encoder based collaborative embedding architecture to generate the embeddings that preserve the node structural property as well as the node cascading characteristics simultaneously in the learned embeddings. The results of extensive experiments conducted on real datasets verify the effectiveness of the proposed method.

CRediT authorship contribution statement

Yuhui Zhao: Conceptualization, Methodology, Software, Writing - original draft. **Ning Yang:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Tao Lin:** Supervision. **Philip S. Yu:** Conceptualization, Methodology, Writing - original draft.

Acknowledgments

This work is supported by National Natural Science Foundation of China under grant 61972270, and in part by National Science Foundation under grants III-1526499, III-1763325, III-1909323, CNS-1930941, and CNS-1626432.

References

[1] J. Cheng, L. Adamic, P.A. Dow, J.M. Kleinberg, J. Leskovec, Can cascades be predicted? in: Proceedings of the 23rd International Conference on World Wide Web, ACM, 2014, pp. 925–936.
[2] C. Li, J. Ma, X. Guo, Q. Mei, DeepCas: An end-to-end predictor of information cascades, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 577–586.
[3] Y. Sun, C. Qian, N. Yang, P.S. Yu, Collaborative inference of coexisting information diffusions, in: 2017 IEEE International Conference on Data Mining, ICDM, IEEE, 2017, pp. 1093–1098.
[4] Y. Li, J. Fan, Y. Wang, K. Tan, Influence maximization on social graphs: A survey, IEEE Trans. Knowl. Data Eng. 30 (10) (2018) 1852–1871.
[5] S. Liu, Q. Qu, S. Wang, Heterogeneous anomaly detection in social diffusion with discriminative feature discovery, Inform. Sci. 439–440 (2018) 1–18.
[6] K. Saito, R. Nakano, M. Kimura, Prediction of information diffusion probabilities for independent cascade model, in: Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Springer-Verlag, 2008, pp. 67–75.
[7] A. Guille, H. Hacid, A predictive model for the temporal dynamics of information diffusion in online social networks, in: Proceedings of the 21st International Conference on World Wide Web, ACM, 2012, pp. 1145–1152.
[8] J. Wang, V.W. Zheng, Z. Liu, C.C. Chang, Topological recurrent neural network for diffusion prediction, in: 2017 IEEE International Conference on Data Mining, ICDM, IEEE, 2017, pp. 475–484.
[9] S. Gao, H. Pang, P. Gallinari, J. Gup, N. Kato, A novel embedding method for information diffusion prediction in social network big data, IEEE Trans. Ind. Inf. 13 (4) (2017) 2097–2105.
[10] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, P. Gallinari, Learning social network embeddings for predicting information diffusion, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, ACM, 2014, pp. 393–402.
[11] M. Gomez-Rodriguez, J. Leskovec, A. Krause, Inferring networks of diffusion and influence, ACM Trans. Knowl. Discov. Data 5 (4) (2011) 1019–1028.
[12] S. Bourigault, S. Lamprier, P. Gallinari, Representation learning for information diffusion through social networks: an embedded cascade model, in: Proceedings of the 9th ACM International Conference on Web Search and Data Mining, ACM, 2016, pp. 573–582.
[13] X. Zhang, Y. Su, S. Qu, S. Xie, B. Fang, P.S. Yu, IAD: Interaction-aware diffusion framework in social networks, IEEE Trans. Knowl. Data Eng. 31 (7) (2019) 1341–1354.
[14] D. Varshney, S. Kumar, V. Gupta, Predicting information diffusion probabilities in social networks: A Bayesian networks based approach, Knowl.-Based Syst. 133 (2017) 66–76.
[15] A. Guille, H. Hacid, C. Favre, D.A. Zighed, Information diffusion in online social networks:a survey, ACM SIGMOD Rec. 42 (2) (2013) 17–28.
[16] L. Yang, Z. Li, A. Giua, Containment of rumor spread in complex social networks, Inform. Sci. 506 (2020) 113–130.
[17] J. Goldenberg, B. Libai, E. Muller, Talk of the network: A complex systems look at the underlying process of word-of-mouth, Mark. Lett. 12 (3) (2001) 211–223.
[18] J. Radcliffe, The mathematical theory of infectious diseases and its applications, J. R. Stat. Soc. Ser. C. Appl. Stat. 26 (1) (1977) 85–87.
[19] G.V. Steeg, A. Galstyan, Information-theoretic measures of influence based on content dynamics, in: Proceedings of the 6th ACM International Conference on Web Search and Data Mining, ACM, 2013, pp. 3–12.
[20] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1225–1234.
[21] L. Liao, X. He, H. Zhang, T.S. Chua, Attributed social network embedding, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2257–2270.
[22] S. Chang, W. Han, J. Tang, G.J. Qi, C.C. Aggarwal, T.S. Huang, Heterogeneous network embedding via deep architectures, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 119–128.
[23] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE:Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.
[24] Y. Xie, M. Gong, A.K. Qin, Z. Tang, X. Fan, TPNE: Topology preserving network embedding, Inform. Sci. 504 (2019) 20–31.
[25] B. Perozzi, R. Alrfou, S. Skiena, DeepWalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 701–710.
[26] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, P.S. Yu, BL-MNE: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder, in: 2017 IEEE International Conference on Data Mining, ICDM, IEEE, 2017, pp. 605–614.
[27] T. Hogg, K. Lerman, Social dynamics of digg, EPJ Data Sci. 1 (1) (2012) 1–26.
[28] L. Weng, F. Menczer, Y.Y. Ahn, Virality prediction and community structure in social networks, Sci. Rep. 3 (8) (2013) 2522.
[29] J. Zhang, B. Liu, J. Tang, T. Chen, J. Li, Social influence locality for modeling retweeting behaviors, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, AAAI Press, 2013, pp. 2761–2767.
[30] M.G. Rodriguez, D. Balduzzi, B. Scholkopf, Uncovering the temporal dynamics of diffusion networks, in: Proceedings of the 28th International Conference on Machine Learning, Omnipress, 2011, pp. 561–568.
[31] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, Comput. Sci. (2013).
[32] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, Curran Associates Inc., 2013, pp. 3111–3119.
[33] M. Pota, F. Marulli, M. Esposito, G.D. Pietro, H. Fujita, Multilingual POS tagging by a composite deep architecture based on character-level features and on-the-fly enriched word embeddings, Knowl.-Based Syst. 164 (2019) 309–323.
[34] M. Esposito, E. Damiano, A. Minutolo, G.D. Pietro, H. Fujita, Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering, Inform. Sci. 514 (2020) 88–105.

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.

[35] T. Deng, D. Ye, R. Ma, H. Fujita, L. Xiong, Low-rank local tangent space embedding for subspace clustering, Inform. Sci. 508 (2020) 1–21.
[36] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (5) (2017) 833–852.
[37] H. Cai, V.W. Zheng, K.C. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, IEEE Trans. Knowl. Data Eng. 30 (9) (2018) 1616–1637.
[38] J.B. Tenenbaum, V.D. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[39] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[40] M. Belkin, P. Niyogi, Laplacian Eigenmaps and spectral techniques for embedding and clustering, in: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, MIT Press, 2001, pp. 585–591.
[41] M.A. Cox, T.F. Cox, Multidimensional scaling, J. R. Stat. Soc. 46 (2) (2001) 1050C1057.
[42] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 855–864.
[43] A. Swami, A. Swami, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 135–144.
[44] L. Xu, X. Wei, J. Cao, P.S. Yu, Embedding of embedding (EOE):Joint embedding for coupled heterogeneous networks, in: Proceedings of the 10th ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 741–749.
[45] P. Goyal, S.R. Chhetri, A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, Knowl.-Based Syst. 187 (2020) 104816.
[46] F. Huang, X. Zhang, J. Xu, C. Li, Z. Li, Network embedding by fusing multimodal contents and links, Knowl.-Based Syst. 171 (2019) 44–55.
[47] C. Chou, M. Chen, Learning multiple factors-aware diffusion models in social networks, IEEE Trans. Knowl. Data Eng. 30 (7) (2018) 1268–1281.
[48] D. Kempe, J. Kleinberg, Maximizing the spread of influence through a social network, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 137–146.
[49] M. Granovetter, Threshold models of collective behavior, Am. J. Sociol. 83 (6) (1978) 1420–1443.
[50] K. Saito, M. Kimura, K. Ohara, H. Motoda, Generative models of information diffusion with asynchronous timedelay, in: Proceedings of 2nd Asian Conference on Machine Learning, Vol. 13, PMLR, 2010, pp. 193–208.
[51] K. Saito, K. Ohara, K. Ohara, H. Motoda, Learning continuous-time information diffusion model for social behavioral data analysis, in: Asian Conference on Machine Learning: Advances in Machine Learning, Springer-Verlag, 2009, pp. 322–337.
[52] L. Wang, S. Ermon, J.E. Hopcroft, Feature-enhanced probabilistic models for diffusion network inference, in: European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD12, 2012, pp. 499–514.
[53] J. Ding, W. Sun, J. Wu, Y. Guo, Influence maximization based on the realistic independent cascade model, Knowl.-Based Syst. (2019) 105265.
[54] F. Gursoya, D. Gunnec, Influence maximization in social networks under deterministic linear threshold model, Knowl.-Based Syst. 161 (2018) 111–123.
[55] S.A. Myers, J. Leskovec, On the convexity of latent social network inference, in: Proceedings of the 23rd International Conference on Neural Information Processing Systems, Curran Associates Inc., 2010, pp. 1741–1749.
[56] M. Gomez-Rodriguez, J. Leskovec, Modeling information propagation with survival theory, in: Proceedings of the 30th International Conference on Machine Learning, PMLR, 2013, pp. 666–674.
[57] S. Wang, X. Hu, P.S. Yu, Z. Li, MMRate: inferring multi-aspect diffusion networks with multi-pattern cascades, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 1246–1255.
[58] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, J. Tang, DeepInf: Social influence prediction with deep learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2018, pp. 2110–2119.

Please cite this article as: Y. Zhao, N. Yang, T. Lin et al., Deep Collaborative Embedding for information cascade prediction, Knowledge-Based Systems (2020) 105502, https://doi.org/10.1016/j.knosys.2020.105502.