DeepInfer: Diffusion Network Inference through Representation Learning

Zekarias T. Kefato
University of Trento
Trento, Italy
zekarias.kefato@unitn.it

Nasrullah Sheikh
University of Trento
Trento, Italy
nasrullah.sheikh@unitn.it

Alberto Montresor
University of Trento
Trento, Italy
alberto.montresor@unitn.it

ABSTRACT

The diffusion of a contagion is a common phenomena in both the cyber and natural spaces. Irrespective of the contagion–a meme, a hashtag, a biological virus–the process is always the same: a *diffusion* or a *cascade* occurs as a result of interaction between agents over a *diffusion network*. Unfortunately, the diffusion network is often unknown: that is, one can observe when the agents are infected by a given contagion (e.g., when a piece of information arrives, when a product is adopted, when a virus is caught), but does not know how the infection has been transmitted. The goal of this study is to infer such a network starting from the contagion events and their relative ordering. Most of the previous approaches to this problem have relied on the delay between two infection events occurring at a pair of nodes, to infer the presence of an edge. It has been argued, however, that delay-agnostic approaches are sufficient to capture the diffusion patterns that lead to recovering of edges in several applications. The algorithm presented in this paper differs from existing delay-agnostic approaches in two aspects: first of all, our study is largely inspired by recent studies on *representation learning* of words in documents and nodes in networks. Second, we consider the relative ordering of diffusion patterns in a restricted window, rather than considering the entire history of events. Starting from the empirical observation that the occurrence of nodes in cascades could be compared to the occurrence of words in documents, we employ the Skip-Gram model to learn a representation of nodes from recorded cascades. The learned representation is then used to compute a probability for an edge to exist between a pair of nodes. Through extensive experiments we validate the effectiveness of our algorithm, showing that it is able to recover up to ≈ 95% of the hidden network in realistic datasets. We have also compared our algorithm to the state-of-the-art algorithm InfoPath, and achieved a large improvement on the quality of results.

KEYWORDS

Network Inference, Social Network, Representation Learning

ACM Reference format:
Zekarias T. Kefato, Nasrullah Sheikh, and Alberto Montresor. 2017. DeepInfer: Diffusion Network Inference through Representation Learning. In *Proceedings of KDD2017, Halifax, Nova Scotia, Canada, August 2017,* 9 pages.
DOI: 10.475/123_4

KDD2017, Halifax, Nova Scotia, Canada

DOI: 10.475/123_4

1 INTRODUCTION

In our daily life, being the observer of the diffusion of information is a pretty common event, both in the physical and in the cyber world. As examples, consider the diffusion of a virus in a population of individuals, the spread of a meme, the adoption of a product, or the diffusion of fake news in online social networks.

A common characteristic of all these diffusion processes, also called *cascades*, is that we are able to observe when a single individual has been infected by a virus, has adopted some product, or has posted/tweeted some information, but we do not get to observe *who* caused such individual to perform such actions.

In other words, the actual network over which the diffusion takes place is either partially or fully hidden [2, 6, 9, 22]. In the former case, we have observations that show different chunks of the diffusion network in the form of who copies from whom, as in retweet networks. In the latter case, our observations are limited at just infection timestamps, for example when using hashtags. This poses a major problem in several network analysis tasks, such as influence maximization and content prediction, that often rely on the knowledge of the diffusion network [22].

In order to effectively infer the underlying network over which diffusions have occurred, several studies [6–10, 14, 22] have been proposed. Some of the existing efforts, such as NetInf [9], NetRate [8], InfoPath [10] and KernelCascade [6], exploit infection rates based on delay patterns between infection timestamps.

Although interesting results have been obtained in such studies, several issues limiting their effectiveness have been identified. It has been argued that even in the presence of recurring diffusion patterns, models based on delays are likely to miss most of them, because of the size of the time intervals used in such models [14]. Delay-agnostic models, instead, have proven to be capable of capturing such patterns [14], as long as the partial order of infections is respected [2].

In this work, we tackle the problem of network inference using a novel approach, based on *representation learning*. Recent advances about word representation learning in the field of natural language processing [16, 17] have inspired several studies for learning representations in the context of social networks [2, 12, 18, 19, 21].

Our work is motivated by observations from previous studies [2, 9, 14], showing that users who frequently post together on related topics have a very good chance of being connected. Inversely, connected users are likely to be frequently infected by diffusions related to similar topics [24].

In other words, if we are given a particular infected user and we are able to observe her *infection contexts*, i.e. the sets of other users that are infected before or after her infections in multiple cascades, we could learn a representation for her that summarize

the users that most frequently appear in her infection contexts. This is equivalent to one of the fundamental assumptions for word representation learning, that is, we can distinguish a word by looking at its context [17]. Hence, equivalently to the famous quote "You shall know a word by the company it keeps" (J.R. Firth, 1957), we may say "You shall know a user by the company it tends to get infected with". We validate such equivalence via an empirical analysis of the distribution of nodes appearance in cascades and words appearance in natural language documents.

The main hypothesis of this study is that we should be able to infer the hidden network by leveraging representation of nodes learned from observed cascades. That is, given the representation of nodes, we infer the edges of the latent network by exploiting a specified similarity function over the representation space.

To prove our hypothesis, we developed a novel algorithm called DeepInfer that learns a representation of nodes from cascades using the Skip-Gram model [17]. The algorithm takes delay-agnostic observations of cascades as input. In each observation, a user is associated with an infection context, containing a bounded number of users who have been infected before and after her. Then, the algorithm learns a representation of the user in a low-dimensional space that preserves her infection context. The learned representation is then used to estimate the probability that an edge exists between every pair of nodes. We estimate such probability using the geometric distance between node representations, since representations capture both nodes infection context in cascades and their closeness in the hidden network.

The contribution of our study can be summarized as follows:

- We empirically analyzed several properties of cascades:
  - We provide a heuristic approach to generate synthetic cascades and show that the cascade size distribution follows a similar distribution as the observed ones.
  - We empirically demonstrate that the distribution of nodes in cascades is similar to the distribution of words in documents.
- We provide an algorithm for learning nodes representation from cascades.
- We provide a novel network inference algorithm based on representation learning that manages to recover up to $\approx 95\%$ of the edges and outperforms previous state-of-the-art by more than an order of magnitude.
- We provide a detailed performance analysis of the algorithm under different hypotheses.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts and notations and presents the problem statement. Section 3 discusses the proposed algorithm. Section 4 reports the experiments and results. Finally, Section 5 discusses related works; the paper is concluded in Section 6.

2 PRELIMINARIES

In this section we provide the basic definitions needed to describe the problem we want to tackle. A *cascade* occurs when a certain contagion, such as a meme, an innovation, or any on-line content in general, has originated from a source and spreads through a diffusion network. The diffusion network, however, is usually hidden and it is what we aspire to infer.

The hidden network is represented as a graph $H = (V, E)$, where V is a set containing n nodes and E is a set containing m edges.

We consider a collection $\mathscr{C}$ of linear cascades, where each linear cascade C captures the sequence of events in which a finite set of nodes have been infected by a given contagion. More formally, we define a *linear cascade* as a sequence $C = [u_1, u_2, \ldots, u_c]$, where u_i are distinct nodes belonging to V. We use $C(i)$ to denote the i-th node of the cascade C. We say that a node u is infected before node v in a cascade C, and we write $u \prec_C v$, if and only if $u = C(i)$ and $v = C(j)$ and $i < j$.

Though infection events are usually associated with an actual timestamp, we are only interested in the relative ordering in which nodes are infected. Compelling arguments have been given [2, 14] as to why the relative order of nodes infection time *per se* is sufficient to solve the network inference problem in several domains.

We assume that a node usually tends to get infected together with other nodes who are very similar to itself, which we refer to as his *infection context*. Unlike rare and viral contagions, most cascades exhibit homophilic behavior, in the sense that the infected nodes are strongly related or very close to each other [4, 24] in the network. For example they could have interconnections between them or belong to the same community within the network.

Each node u has two types of infection contexts based on the order they get infected in a given cascade:

- the *influencer context* $C(u;s)^{\preceq}$ of node u in cascade C contains the s nodes that immediately precede u in C

$$C(u;s)^{\preceq} = \{v : v = C(i) \wedge u = C(j) \wedge j - s \leq i \leq j - 1\}$$

- the *influenced context* $C(u;s)^{\succeq}$ of node u in cascade C contains the s nodes that immediately succeed u in C:

$$C(u;s)^{\succeq} = \{v : v = C(i) \wedge u = C(j) \wedge j + 1 \leq i \leq j + s\}$$

For example, given $s = 2$ and a cascade

$$C = \{a, b, c, u, v, w, x, y\},$$

then $C(u;2)^{\preceq} = \{b, c\}$ and $C(u;2)^{\succeq} = \{v, w\}$. The two sets $C(u;s)^{\preceq}$ and $C(u;s)^{\succeq}$, can be loosely interpreted as the candidate influencer nodes of u and the candidate nodes to be influenced by u, respectively.

An important insight of our study (see Section 4) is that, as we increase the window size above a certain value, it becomes more and more difficult to observe recurring patterns and consequently the performance of a model will be hampered. In previous studies [2, 14] no restriction where in place regarding this size while learning influence propagation probabilities, introducing unnecessary noise in the learned representation.

The problem we want to solve is the following: given a set of observed cascades $\mathscr{C}$ over a hidden network $H = (V, E)$, we want to infer a network $G = (V, E')$ such that $E' \approx E$ as much as possible. We will evaluate the quality of our results based on precision, recall and F1 score.

3 DEEPINFER

Our study is based on the hypothesis that there is a mapping from nodes appearances in cascades to their structural information in the diffusion network. That is, nodes that frequently co-occur together

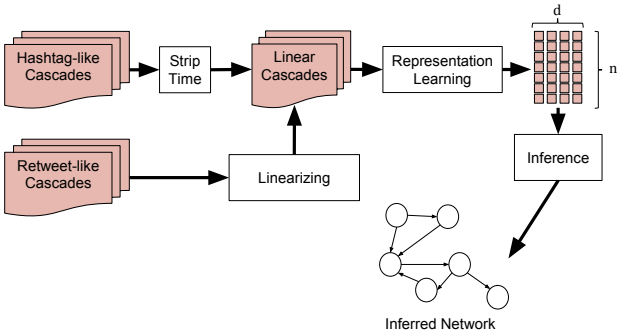
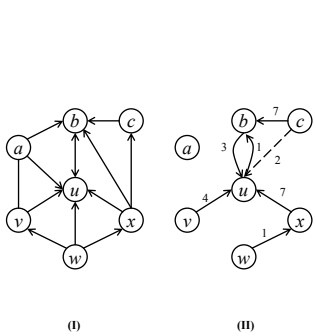


Figure 1: DeepInfer Architecture



User	Contagion	Infection Time
u	1	0
b	1	1
a	1	3
x	1	4
w	1	6
x	2	0
w	2	2
x	4	1
v	4	3
w	4	4

(III)

Figure 2: An example of a (I) diffusion network, (II) retweet cascade (the weights correspond to the number of retweets), and (III) hashtag cascade

in cascades are closely related in the diffusion network, e.g. have edges interconnecting them.

Based on this insight, we propose the DeepInfer algorithm, whose architecture is depicted in Fig. 1. In the first phase, we consider two possible sources of input, namely *hashtag-like cascades* and *retweet-like cascades* (referred as *hashtag cascades* and *retweet cascades* for the sake of brevity). These sources are transformed into linear cascades. Linear cascades are then fed to the *representation learning* module, whose goal is to learn a representation of the nodes based on the infection contexts extrapolated from the linear cascades. Then, the *inference module* is run over the learned representation to reconstruct the diffusion network.

Hashtag cascades. Usually infections are described as a log of timestamped events, representing the first time a user u has been infected by a specific piece of information. As a possible use case, consider Twitter: each hashtag T is associated with a distinct cascade C_T, and a node-timestamp pair $(u, t) \in C_T$ represent the first time at which user u has started to use T in her tweets.

As an example, consider Fig. 2(III), representing three cascades (identified by ids 1,2,4) and a collection of users with their respective timestamps. Note that nodes w and x were infected close to each other in all cascades; in fact, they form an edge $(w, x) \in H$ (Fig. 2(I)).

As noted in the previous sections, information about timestamps are stripped away and cascades are just represented by an ordered sequence of nodes.

Retweet cascades. In some cases, infections could be described by a log of directed connections between users, representing partial knowledge about who influenced whom during the infections. As a possible use case, consider retweets and mentions in Twitter: a pair $(v \to u)$ could mean that user v has retweeted a message originally tweeted by u (the direction of the arrow means that v is following u).

In Fig. 2, for example, (I) shows the full diffusion network, while (II) depicts the associated retweet infections. In Fig. 2(II), each edge is associated with weights that correspond to the number of times the retweet-like action has been performed over the directed edge.

An important problem in this case is that we could have edges that do not exist in the diffusion network. Consider again Twitter as an example: let u be a user that originally tweets a message T, let v be a user that re-tweeted T after having seen it from u, and let w be a user that re-tweeted T after having seen it from v. Unfortunately, Twitter does not provide the retweet network as $(v \to u), (w \to v)$; instead, it logs $(v \to u), (w \to u)$, to mark the fact that the message originated from u in both cases. Without loss of generality, we can account for edges like this, e.g the dotted edge (c, u) in Fig. 2(II) corresponds to an instance as such.

Such directed network representation does not provide linear infection sequences that enables us to capture the infection contexts formalized in Section 2. For this reason, we need a strategy to derive linear cascades from retweet cascades. Such strategy is described in Section 3.1.

Representation learning and inference. The next step is given by the representation learning module, that takes a collection $\mathscr{C}$ of linear cascades as input and learn a representation $\Phi : V \to \mathbb{R}^d$ mapping nodes to a d-dimensional euclidean space. This module is described in Section 3.2. Finally, the learned representation is used to infer the hidden diffusion network, as described in Section 3.3

3.1 Linearizing Retweet cascades

Retweet cascades are often provided as directed graphs obtained by merging individual cascades. Once cascades are merged, there is no straightforward way of obtaining the linear infection orders. Sometimes the original retweet cascades may not be accessible, and one needs to reconstruct the retweet cascades from the retweet network. In fact, even if the retweet cascades are readily available, the linear infection order is not as immediate as in the hashtag cascades. Therefore, for datasets where we face this kind of situation, we reconstruct the infection order of the retweet cascades by assuming that some kind of diffusion model have generated them; more specifically, we consider the well- known discrete time independent cascade (*IC*) model [13]. We thus convert the retweet cascades to linear cascades.

Every cascade has a root where the infection has originated, and the set of cascade roots is denote by R. Nodes of the retweet network where there is at least one retweet originating from them, in other words with at least one incoming edge, are deemed to be the possible roots. For example, for the retweet network in Fig. 2 (II), we have $R = \{b, u, x\}$. Then for each cascade root node $r \in R$, we generate at most $c \propto \sum_{(z,r) \in E(G_R)} in(z, r)$ cascades, where G_R

Algorithm 1 Linear Cascade Generator

Require: G_R: retweet network
Require: R: Cascade roots
Ensure: $\mathscr{C}$: list of linear cascades
1: **procedure** GENERATELINEARCASCADES
2: $\overline{G_R} = reverse(G_R)$ ▷ Influence flow network
3: $\mathscr{C} \leftarrow \emptyset$
4: **for** $r \in R$ **do** ▷ for each root r
5: $cascadeCounter = 0$
6: **while** $cascadeCounter \leq in_{sum}(r)$ **do**
7: $lb, ub = Eq.\ 1$
8: $ipp = sampleUniform(lb, ub)$
9: $C = icSim(\overline{G_R}, r, ipp)$
10: $C = stripTimestamps(C)$
11: $append\ C\ to\ \mathscr{C}$
12: $cascadeCounter++$
13: **return** $\mathscr{C}$

is the retweet network and $in(z, r)$ is the weight of r's incoming edge (z, r) .

The complete procedure to generate the linear cascades is given in Algorithm 1. The first step in the algorithm (line 2) is to reverse the edge of G_R so as to obtain $\overline{G_R}$, i.e. the *influence/information flow network*. The core of the algorithm is line 9, i.e. the sub-routine $icSim(\overline{G_R}, r, ipp)$. This method generates a cascade C by simulating the influence propagation over the edges of $\overline{G_R}$ starting from a root r according to the *IC* model [13]. For each simulation (cascade) from the root r, an influence propagation probability ipp is sampled uniformly at random between the interval $[lb(r), ub(r)]$ specified in Eq. 1 (line 8).

$$lb(r) = \frac{in_{avg}(r)}{\max_{z \in V(G_R)} in_{sum}(z)}$$
$$ub(r) = \frac{in_{sum}(r)}{\max_{z \in V(G_R)} in_{sum}(z)} \quad (1)$$

where the overloaded $in_f(z)$ of any $z \in V(G_R)$ is the incoming weight of z summarized according to the function f, e.g when f is *sum* then the summary is the sum of the incoming weights of z, *i.e.* $in_{sum}(z) = \sum_{(y,z) \in E(G_R)} in(y, z)$. The variable ipp corresponds to the probability of a cascade spreading from any node $u \in V(\overline{G_R})$ that was infected at time t to any of the non-infected out-neighbors $\{z : (u, z) \in E(\overline{G_R})\}$ of u in the next time step $t+1$ over the influence flow network.

The lb and ub bounds in Equation 1 are chosen heuristically; nonetheless, Fig 3 shows that the simulated cascades have a similar property as the observed cascades distribution. More concretely, there are a very few number of large cascades, and a large number of small cascades; they follow a power-law distribution. The intuition behind the heuristic is that, as in real world social networks each node u has to face a competition to generate popular contents or cascades that go viral. Our formulas model a situation where nodes with small (large) in-degree weights have low (high) likelihoods to generate large cascades, respectively. This is what Eq. 1 is intended to achieve, e.g in Fig 2 (II) x is less likely where as u and b are more likely to generate large cascades.

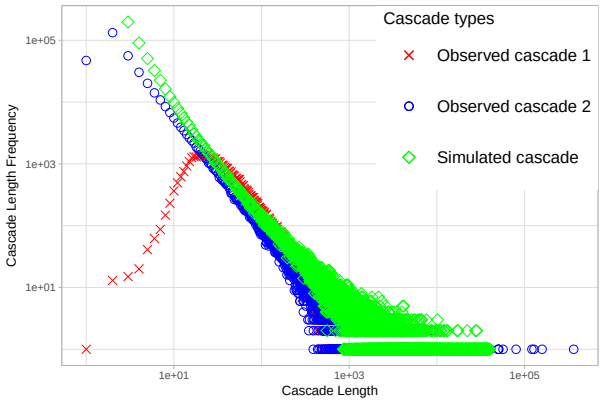


Figure 3: Cascade size distributions for three of our datasets

To better understand how the $icSim(\overline{G_R}, root, ipp)$ sub-routine in Algorithm 1 works, consider the following illustration. Suppose we are generating cascades by fixing a cascade root at b; then, since $in_{sum}(b) = 8$ we generate at most 8 cascades starting from b, controlled by the loop at line 6. The other quantities relevant for our purpose are $in_{avg}(b) = 4$ and $max_{z \in V(G_R)}\ in_{sum}(z) = in_{sum}(u) = 16$, consequently $lb(b) \approx 0.25$ and $ub(b) \approx 0.5$. Suppose we have sampled $ipp = 0.4 \in [0.25, 0.5]$ (line 8), then the *icSim* subroutine proceeds as follows:

(1) At time $t = 0$ it initiates a cascade by infecting the current root, $C = \{(b, 0)\}$, and maintain a queue $q = [b]$.
(2) At time $t = t + 1$ the node y at the head of q tries to infect each out neighbor $(y, z) \in E(\overline{G_R})$ with probability ipp. In the first round b does this with $ipp = 0.4$, and suppose it succeeds to do so, thus $C = \{(b, 0), (u, 1), (c, 1)\}$
(3) Remove the head of q and add newly infected nodes (for our example, during the first round $q = [u, c]$).
(4) Then for the current node at the head of q (for example u) start from step 2 until no infection is possible ($q \neq \emptyset$).
(5) Suppose only v is infected by u in the following rounds; hence the final state of $C = \{(b, 0), (u, 1), (c, 1), (v, 2)\}$ is returned to the caller.
(6) Finally the timestamps are stripped (line 10), and hence a linear cascade

As we shall show later in the experiments, this strategy is useful not only when the cascades information is merged, but also when it is available. We get better results using the simulated (for large number simulations) cascades than the observed cascades. One possible explanation for this could be that such cascades might capture diffusion patterns that already happened and are likely to happen in the future.

3.2 Representation Learning from Cascades

Essentially, our algorithm at its core leverages representation of nodes learned from cascades. The representation learning aspect of our algorithm is heavily inspired by word representation learning [16, 17] in natural languages. The state-of-the-art word representation learning techniques employ the so-called "*learning by*

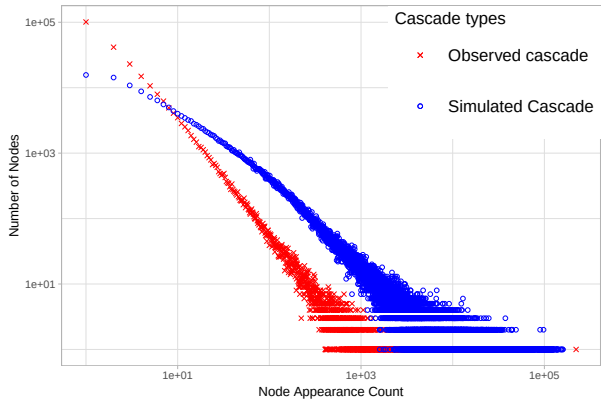


Figure 4: The distribution of nodes occurrence in cascades.

prediction" strategy [1]. In a nutshell, the idea is to learn a representation of words that enables us to predict their context, where the context of a word is specified by those words that regularly co-occur with it. This notion motivates us to hypothesize that cascades can be considered as documents in natural languages, and nodes as words. As a result, we can exploit algorithms for word representation learning to learn representations for nodes. We validate our hypothesis that nodes in cascades have an equivalence mapping to words in documents based on the distribution of words and nodes appearance in documents and cascades, respectively. For instance, it has been shown [18] that words occurrence in Wikipedia documents follows a power-law distribution, and as shown in Fig. 4 nodes occurrence in cascades also follows a power-law distribution. Therefore, we tackle the node representation learning task by employing the Skip-Gram model [12, 16–18] used for word and network representation learning. In the following we discuss this model in relation to our context.

Skip-Gram Model. Given a center node $u \in C$, this model maximizes the log probability of observing context nodes $v \in C(u;s)^{\preceq}$ and $w \in C(u;s)^{\succeq}$ within a window size s. Based on the assumption that the likelihood of observing each context node given a center node is independent, more formally the Skip-Gram model optimizes the objective in Eq. 2 with respect to the model parameter Φ.

$$\max_{\Phi} \sum_{u \in V} \log Pr(C(u;s)^{\preceq} \mid \Phi(u)) + \log Pr(C(u;s)^{\succeq} \mid \Phi(u)) \quad (2)$$

$$\log Pr(C(u;s)^{D} \mid \Phi(u)) = \sum_{v \in C(u;s)^{D}} \log Pr(v|\Phi(u)) \quad (3)$$

where D is either $\preceq$ or $\succeq$, and $\Phi(u) \in [0,1]^d$ is a $d-$dimensional representation of u. The right-hand side term in Eq. 3 is specified using the softmax function as follows:

$$Pr(v|\Phi(u)) = \frac{\exp(\Phi(v)^T \cdot \Phi(u))}{\sum_{w \in N} \exp(\Phi(w)^T \cdot \Phi(u))} \quad (4)$$

Nonetheless, directly estimating the conditional probability in Eq. 4 is expensive, because of the normalization constant that needs to be computed for every node. For this reason, different approximation strategies have been suggested in the literature; in this work, we adopt the "Negative Sampling" strategy [17] that characterizes a good model by its power to discriminate appropriate context nodes from noise. Then, the computation of $\log Pr(v|\Phi(u))$ using the negative sampling strategy is shown in Eq. 5.

$$\log Pr(v|\Phi(u)) = \log \sigma(\Phi(v)^T \Phi(u)) + neg(u;l) \quad (5)$$

σ is the logistic function, and we need the model to effectively differentiate v from the l negative examples drawn from some noise distribution $\mathcal{N}(u)$ of u, where $neg(u;l)$ is the noise model and is defined as

$$neg(u;l) = \sum_{i=1}^{l} \mathbb{E}_{w_i \sim \mathcal{N}(u)}[-\log \sigma(\Phi(w_i)^T \Phi(u))] \quad (6)$$

Numerically, a good model should produce a small expected probability for the noise model and larger probability for the data model (the first term on the right-hand-side of Eq. 5).

Finally, we employ the stochastic gradient descent algorithm to optimize the objective in Eq. 2 based on the negative sampling strategy in Eq. 5 and 6 and obtain the complete model parameters $\Phi \in V \rightarrow [0,1]^d$.

3.3 Network Inference

Once obtained a representation $\Phi(u)$ for every $u \in V$, the next step is to seek to infer the hidden diffusion network H. Note that, the driving premise behind our algorithm is that nodes that are close to each other in the diffusion network are likely to get infected together in most cascades. Inversely, nodes that tend to co-appear in most infection cascades are likely to be closely related to each other in the diffusion network, for example belong to the same community. Based on this assumption, we utilize the representation learned from the cascades to infer edges in the hidden network.

If node closeness in the hidden network is captured in the representation, we can simply compute some geometric distance of nodes representation to infer edges between a pair of nodes. In particular, we estimate a probability $p(u,v)$ for every pair of nodes $\langle u,v \rangle$ in each cascade C based on the cosine similarity of $\Phi(u)$ and $\Phi(v)$. Then, an edge (u,v) is inferred if the similarity is above a certain threshold. More formally, let $\theta \in (0,1]$ be a threshold, and $G = (V, E')$ be the inferred network, then G is constructed such that:

$$E' = \{(u,v) : p(u,v) \geq \theta\} \quad (7)$$

where

$$p(u,v) = \frac{\Phi(u) \cdot \Phi(v)}{\| \Phi(u) \| \| \Phi(v) \|}$$

Now, one can infer edges between every pair of nodes by using Eq. 7. However this is very expensive for very large networks, for this reason we avoid such pair-wise computations by leveraging an interesting property of cascades. That is, nodes in large (viral) cascades usually belong to a lot of communities (they do not have correlations), where as nodes in small cascades usually belong to a single or to very few communities (they are tightly connected) [24]. Hence, first we reduce the set of cascades by ignoring large cascades and then we scan each of the remaining cascades to infer an edge between each pair of nodes in the corresponding cascade. Yet again we do not have to test for a possible edge between each pair of nodes in a given cascade. We can ignore pairs that co-occur merely a few times in cascades and probe pairs that are relatively frequent.

Dataset	#Cascades	#Nodes in H	#Edges in H
HT_1_1	397,681	595,460	14,273,311
RT_1_1	1,860,000	595,460	14,273,311
RT_1_2	540,385	595,460	14,273,311
RT_2_1	144,704	456,626	14,855,842
RT_2_2	54,785	456,626	14,855,842
Memetracker	71,568	3,836,314	15,540,787

Table 1: Dataset statistics

4 EXPERIMENTS AND RESULTS

In this section, we first introduce the real and synthetic datasets on which DeepInfer is evaluated, and then we provide the main results.

4.1 Datasets

In order to evaluate our model, we use the following six real and synthetic datasets, and the basic summary of each dataset is provided in Table 1.

(1) Twitter #1 [24]: A Twitter dataset containing infections related to hashtags; there are three variants of this dataset:
   (a) Hashtag cascade (HT_1_1 - Observed cascades): A record of hashtag users. Each entry corresponds to a hashtag and its adopters.
   (b) Retweet cascade #1 (RT_1_1 - Simulated cascades): Records containing users retweeting other's tweets.
   (c) Retweet cascade #2 (RT_1_2 - Simulated cascades): Records containing users mentioning other users.

(2) Twitter #2 [5]: A Twitter dataset collected before, during, and after the announcement of the Higgs boson particle. We use the the following two kinds of datasets [1]
   (a) Retweet cascade #1 (RT_2_1 - Simulated cascades): Contains users retweet information related to the Higgs boson.
   (b) Retweet cascade #2 (RT_2_2 - Simulated cascades): Contains users mentioning other users related to the Higgs boson.

(3) Memetracker [15](Observed Cascades): A dataset containing news and blog posts mentioning memes. We construct hashtag-like infections by utilizing the meme cluster dataset [2]. Each cluster id is considered as a contagion, and a news or blog post is said to be infected by the contagion if it mentions a meme that belongs to the cluster.

For all the datasets we have a ground truth, that is, the diffusion network against which we are going to compare the inferred network. For the Memetracker dataset the ground truth is constructed by considering hyperlinks between pages.

4.2 Results and Discussion

In the first set of experimental results we seek to empirically verify our assumptions. Towards this end, we generate a new extended graph H' that has an additional false edge $(u, v') \notin E$ per every true edge $(u, v) \in E$ in the ground truth network H. Next we analyze the performance of DeepInfer in correctly classifying the edges in H' according to different hypotheses. Based on the classification results of each experiment, the performance of the model according to a specific hypothesis is measured using one or more of the following four evaluation metrics, which are precision, recall, F1, and error-rate. For all the metrics but error-rate we seek to achieve higher scores. Note that we have a balanced number of valid (true) and invalid (false) edges in the extended ground truth - H', hence the worst performance in terms of the error-rate is 50%. That is, if our algorithm is unable to detect true edges correctly, it will be correct in 50% of the times for classifying every edge as false.

In our first experiment we shall demonstrate the claim that using the simulated (given large number of simulations) rather than the observed cascades leads to a better performance. For the two datasets, RT_1_1 and RT_1_2, we have the corresponding observed cascades, and hence we report the performance comparison of observed vs simulated cascades across different measures as shown in

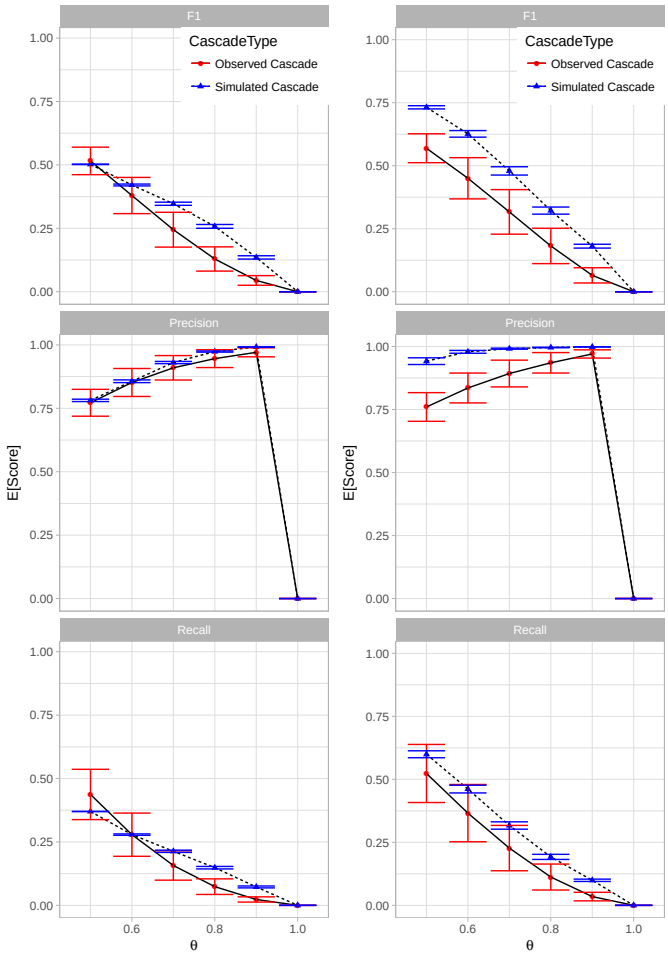


Figure 5: Observed vs Simulated Cascades Performance Evaluation for RT_1_1 (right column) and RT_1_2 (left column). The number of cascades in the observed case of RT_1_1 is 226,488 and RT_1_2 is 217,653.

[1] http://snap.stanford.edu/data/higgs-twitter.html
[2] http://www.memetracker.org/data.html

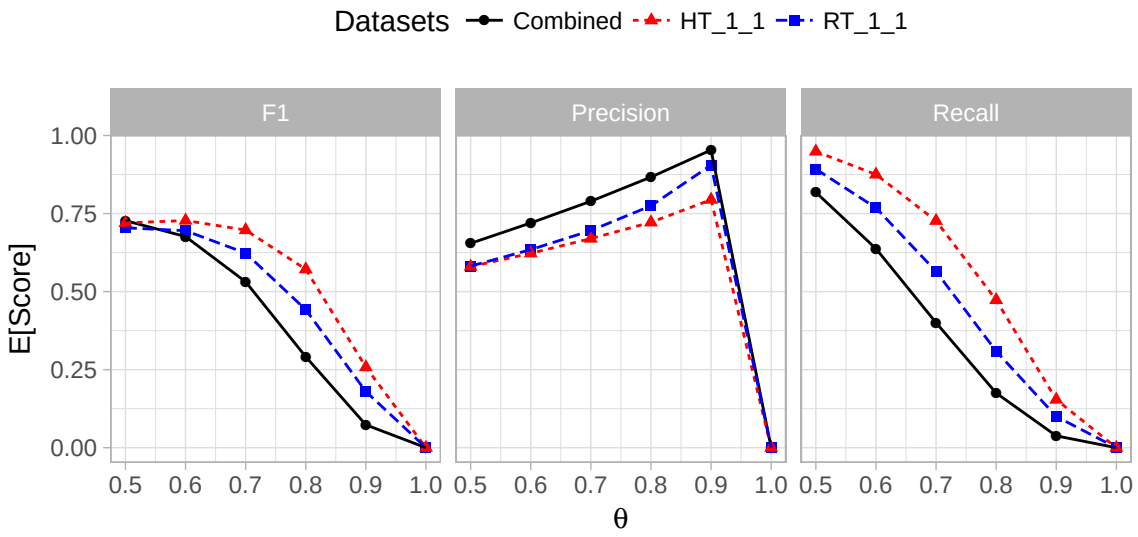


Figure 6: Combined vs separate cascades performance evaluation on window size, $s = 5$

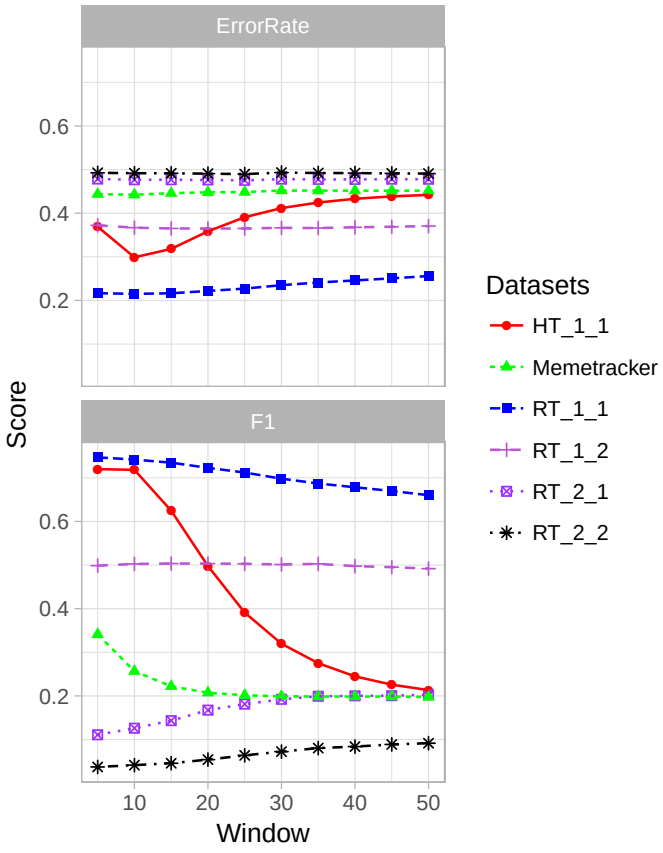


Figure 7: Evaluation on the effect of increasing window size

Fig. 5. In the figure, we consider the expected value for each threshold θ summarized over the range of window size 5, 10, 15, 20, 25. In fact we do achieve better performance by using the simulated ones. It is important, however, to note that the larger the number of simulations (cascades) the better the performance. Observe that the number of cascades for RT_1_2 is much smaller than RT_1_1 as shown in Table 1, and hence the performance improvements are more vivid in the later case.

Next, we shall proceed to illustrate the effects of increasing the window size. Fig 7 shows that increasing the window size up to a certain turning point (depending on the dataset) might result in a quality improvement. Increasing it beyond that turning point, however, may significantly reduce the quality obtained by the algorithm; even when the quality is scarcely affected, the training time increases significantly [17] .

In another experiment we analyze the effect of combining different kinds of cascades for the same dataset to understand if we can achieve a better performance. As shown in Fig. 6, there is some performance gain in terms of precision that comes at the expense of recall by combining the HT_1_1 and RT_1_1 cascades for Twitter #1 dataset. However as illustrated by the F1 measure, the overall quality decreases. The experiment is repeated for other cascades in other datasets, and we have observed the same property as in Fig 6.

In our last experiment we evaluate the performance of DeepInfer with the state-of-the-art method called InfoPath and show that DeepInfer performs better. There is a fundamental assumption for both algorithms that requires a pair of nodes to sufficiently co-occur in cascades so as to infer a possible edge. This is a valid assumption, as it can be observed from the distribution of node-node co-occurrence frequencies in Fig. 8. Most node-node pairs co-occur just a very few times, more precisely $\approx 96\%$ and $\approx 80\%$ of the pairs in HT_1_1 and Memetracker datasets respectively co-occur only once. Clearly no diffusion pattern can be learned from these

Algorithm	Precision	Recall	F1	K
DeepInfer	0.51	0.29	**0.36**	1,000
	0.38	0.35	**0.36**	5,000
	0.29	0.37	**0.33**	10,000
	0.18	0.34	**0.24**	50,000
	0.14	0.32	**0.19**	100,000
InfoPath	0.18	0.43	0.25	1,000
	0.12	0.24	0.16	5,000
	0.08	0.16	0.12	10,000
	0.07	0.07	0.07	50,000
	0.05	0.05	0.05	100,000

Table 2: Performance evaluation of DeepInfer and InfoPath (approximated to two decimal places) on the HT_1_1 dataset. Parameter setting, DeepInfer: $\theta = 0.5, s = 5$ and InfoPath: exponential influence model. Best performances are represented in bold. K is the value for the top-K frequently co-occurring node-node pairs

Algorithm	Precision	Recall	F1	K
DeepInfer	0.41	0.74	**0.53**	1,000
	0.28	0.52	**0.36**	5,000
	0.24	0.43	**0.30**	10,000
	0.20	0.26	**0.23**	50,000
	0.19	0.22	**0.20**	100,000
InfoPath	0.21	0.92	0.41	1,000
	0.24	0.58	0.34	5,000
	0.23	0.42	0.29	10,000
	0.15	0.18	0.16	50,000
	0.13	0.12	0.13	100,000

Table 3: Performance evaluation of DeepInfer and InfoPath (approximated to two decimal places) on the Memetracker dataset. Parameter setting, DeepInfer: $\theta = 0.5, s = 5$ and InfoPath: exponential influence model. Best performances are represented in bold. K is the value for the top-K frequently co-occurring node-node pairs

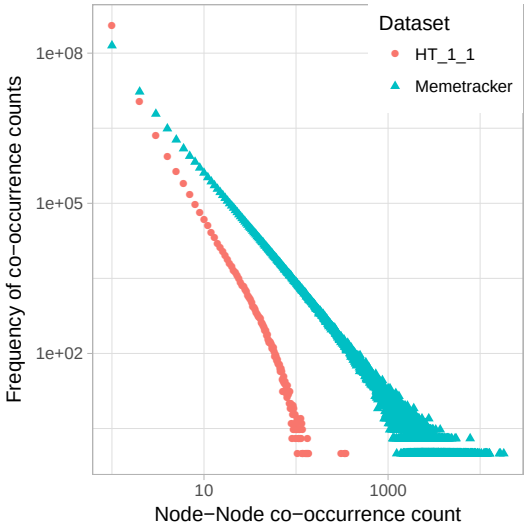


Figure 8: Distribution of node-node pair co-occurrences

pairs, and hence there is no benefit in probing them for possible edges. Therefore, in order to compare the two methods, we first compute a ranking of node-node pair co-occurrences in cascades. Next we perform several experiments for both methods by utilizing those cascades containing at least one of the top-K node-node pairs and a ground-truth network induced by the top-K pairs. The two observed datasets, HT_1_1 and Memetracker, are used for the evaluation; and the results are reported in Table 2 and 3. For InfoPath we use all the default configurations of the implementation[3].

5 RELATED WORK

In this section we discuss selected related works in both representation learning and network inference.

[3]http://snap.stanford.edu/infopath/software.html

5.1 Representation Learning

Recent advances in neural network models have attracted researches from several communities such as computer vision, NLP, and social network analysis. For our purpose, we only consider the literatures from the last two communities. The seminal work of Mikolov et al. [17] in representation learning (embedding) of words in documents using a shallow neural network model has inspired studies [12, 18] in network representation learning. Among the approaches introduced for word embedding, the Skip-Gram model [17] is the one that has been most largely used for network representation learning. The Skip-Gram model learns a representation of words by way of predicting context words.

The context of a node in a network, however, does not have a straightforward definition. Studies have introduced different strategies of capturing nodes context, for example using random walks [12, 18], pair-wise proximities [3, 19, 21], and community structures [20, 23]. Once a context is formalized different neural network, either shallow or deep models are employed for the representation learning task. Then the learned representations are utilized for downstream network analysis tasks.

5.2 Network Inference

There have been several works [6–11, 14] to infer diffusion networks. A large percentage of the studies in this area are motivated by the fact that diffusion networks are very often hidden. Therefore, the standard approach towards inferring such a network is by leveraging diffusion patterns in cascades. Most studies [6, 8–11] have focused on the delay between the infection times of a pair of nodes infected by a certain contagion. The main premise is that if a pair of nodes are frequently infected within a certain time window, then there is a diffusion pattern that is a likely indicator of connections. Some of these studies [8, 10] assume a fixed parametric form (e.g. exponential or power-law) of influence model on the edges of the network. Nonetheless, one particular study [6] has argued and empirically illustrated that such an assumption is too strong for capturing the complex diffusion patterns in real networks. For

this reason, Du et al. [6] have presented a method using survival analysis based on a kernelized hazard function.

The common assumption of most studies in this task, including us, is that the network is considered to be static. One particular study [10], however, has proposed an elegant solution for dynamic networks as well.

Besides delay-aware approaches, a delay-agnostic technique [14] has been proposed recently. Mainly they argue that diffusion patterns within a restricted window of time are difficult to extract. Therefore they propose a method that is based on the relative order of infection. However, their influence model considers all infected nodes before and after a certain node while learning infection probabilities. But as illustrated in our experimental results, it is difficult to extract meaningful diffusion patterns unless we consider nodes in a restricted window of infection context.

6 CONCLUSION AND FUTURE WORK

In this study we address the problem of diffusion network inference. Usually diffusion networks are hidden to us, and one has to first deal with the inference task before carrying out any downstream analysis on the network. Therefore to tackle this problem we propose a novel algorithm called DeepInfer using representation learning. The algorithm first learns a representation of nodes from cascades and then leverages such a representation to infer edges of the hidden network. By exploiting the empirical mapping between words in documents and nodes in cascades, DeepInfer uses the Skip-Gram model to learn the representation of nodes from cascades.

We have performed extensive experiments to prove most of our assumptions and also to compare the effectiveness of our algorithm with the state-of-the-art. We have managed to recover up to $\approx 95\%$ of the edges of the hidden network and achieve more than an order of magnitude improvement over the state-of-the-art. In a future work we would like to consider inference for dynamic networks.

REFERENCES

[1] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 238–247. http://www.aclweb.org/anthology/P14-1023

[2] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. 2016. Representation Learning for Information Diffusion Through Social Networks: An Embedded Cascade Model. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, New York, NY, USA, 573–582. DOI: http://dx.doi.org/10.1145/2835776.2835817

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, New York, NY, USA, 891–900. DOI: http://dx.doi.org/10.1145/2806416.2806512

[4] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can Cascades Be Predicted?. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. ACM, New York, NY, USA, 925–936. DOI: http://dx.doi.org/10.1145/2566486.2567997

[5] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. 2013. The Anatomy of a Scientific Rumor. *Scientific Reports* 3, 02980 (October 2013).

[6] Nan Du, Le Song, Alex Smola, and Ming Yuan. 2012. Learning Networks of Heterogeneous Influence. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. Curran Associates Inc., USA, 2780–2788. http://dl.acm.org/citation.cfm?id=2999325.2999445

[7] Nathan Eagle, Alex (Sandy) Pentland, and David Lazer. 2009. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences* 106, 36 (2009), 15274–15278. DOI: http://dx.doi.org/10.1073/pnas.0900282106 arXiv:http://www.pnas.org/content/106/36/15274.full.pdf

[8] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. 2011. Uncovering the Temporal Dynamics of Diffusion Networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Omnipress, 561–568.

[9] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. 2010. Inferring Networks of Diffusion and Influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. ACM, New York, NY, USA, 1019–1028. DOI: http://dx.doi.org/10.1145/1835804.1835933

[10] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schölkopf. 2012. Structure and Dynamics of Information Pathways in Online Media. *CoRR* abs/1212.1464 (2012). http://arxiv.org/abs/1212.1464

[11] Manuel Gomez-Rodriguez and Bernhard Schölkopf. 2012. Submodular Inference of Diffusion Networks from Multiple Trees. *CoRR* abs/1205.1671 (2012).

[12] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 855–864. DOI: http://dx.doi.org/10.1145/2939672.2939754

[13] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the Spread of Influence Through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. ACM, New York, NY, USA, 137–146. DOI: http://dx.doi.org/10.1145/956750.956769

[14] Sylvain Lamprier, Simon Bourigault, and Patrick Gallinari. 2015. Extracting Diffusion Channels from Real-World Social Data: A Delay-Agnostic Learning of Transmission Probabilities. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 (ASONAM '15)*. ACM, New York, NY, USA, 178–185. DOI: http://dx.doi.org/10.1145/2808797.2808865

[15] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the Dynamics of the News Cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 497–506. DOI: http://dx.doi.org/10.1145/1557019.1557077

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://arxiv.org/abs/1301.3781

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS'13)*. Curran Associates Inc., USA, 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 701–710. DOI: http://dx.doi.org/10.1145/2623330.2623732

[19] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. *CoRR* abs/1503.03578 (2015). http://arxiv.org/abs/1503.03578

[20] Cunchao Tu, Hao Wang, Xiangkai Zeng, Zhiyuan Liu, and Maosong Sun. 2016. Community-enhanced Network Representation Learning for Network Analysis. *CoRR* abs/1611.06645 (2016). http://arxiv.org/abs/1611.06645

[21] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1225–1234. DOI: http://dx.doi.org/10.1145/2939672.2939753

[22] Liaoruo Wang, Stefano Ermon, and John E. Hopcroft. 2012. Feature-Enhanced Probabilistic Models for Diffusion Network Inference. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II (ECML PKDD'12)*. Springer-Verlag, Berlin, Heidelberg, 499–514. DOI: http://dx.doi.org/10.1007/978-3-642-33486-3_32

[23] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. *AAAI* (2017).

[24] L. Weng, F. Menczer, and Y.-Y. Ahn. 2013. Virality Prediction and Community Structure in Social Networks. *Sci. Rep.* 3, 2522 (2013). http://dx.doi.org/10.1038/srep02522