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 (e.g. news, meme, virus) is a common event in online and offline social networks. Unfortunately, such networks, known as *diffusion networks*, are often unknown: that is, one can observe when subjects 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 through which connection the infection has been transmitted. The goal of this study is to infer such networks based on nodes infection contexts associated to diffusion events (cascades). Previous studies mostly relied on delay patterns between infection events of nodes to infer edges. It has been argued, however, that delay-agnostic approaches are also efficient for such inference. Motivated by this finding, we present a novel delay-agnostic algorithm that is largely inspired by representation learning of words in documents and nodes in networks. Moreover, unlike some delay-agnostic methods, we only consider infection context of nodes in a restricted window. After empirically observing a similarity between the distribution of words in documents and nodes in cascades, we employ the Skip-Gram model to learn a representation of nodes from cascades. The learned representation is then used to compute the probability that an edge exists 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, Information Diffusion, Representation Learning

ACM Reference format:
Zekarias T. Kefato, Nasrullah Sheikh, and Alberto Montresor. 2017. DeepInfer: Diffusion Network Inference through Representation Learning. In *Proceedings of MLG'17, Halifax, Nova Scotia, Canada, August 2017,* 8 pages.
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, 23]. 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 [23].

In order to effectively infer the underlying network over which diffusions have occurred, several studies [6–10, 14, 23] 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 can be identified. It has been argued that models based on delays are likely to miss most of the diffusion patterns, even in the presence of recurring ones, 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, 20, 22].

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 [25].

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 sample cascades with 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 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; finally, 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 $C = [u_1, u_2, \dots, u_c]$, 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 and context in which nodes are infected. Compelling arguments have been given [2, 14] as to why the relative order of node infection events *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, 25] 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

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

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

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

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 were 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 node appearances in cascades to their structural information in the diffusion network. That is, nodes that frequently co-occur together in cascades are similar or closely related in the diffusion network, e.g. have edges interconnecting them. In other words, strongly similar nodes (e.g. interested in related topics) tend to get infected by mostly related contagions, and hence are likely to have a direct link, or at least belong to a similar/same community.

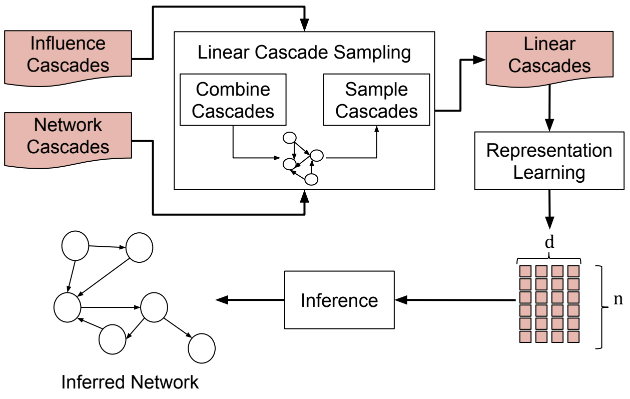

Figure 1: DeepInfer Architecture

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 *influence cascades* and *network cascades* that are transformed into linear cascades through a sampling process. 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.

A raw cascade is normally represented as a graph [19], and there are two type of representations of such a graph. The first is called an *influence cascade* (e.g. re-tweets and shares in Twitter and Facebook respectively), and the influence flow/propagation is explicit, i.e. as a result of the re-tweet/share action the source and the target of the propagation are known. We denote these types of raw cascades as $I_C = (V', V', T)$, and the graph representation as $I_G = (V', E')$. For any tuple $(u, v, t) \in I_C$: u, v, and t are the source, target and timestamp of an influence/information propagation respectively; and $(u, v) \in I_G$. The second is called a *network cascade* (e.g. hashtags in Twitter and Facebook), and in this case the propagation of influence is not explicit. It is not clear who has influenced a user to start using a hashtag; it could be any of her friends who have adopted the hashtag prior to her. Raw network cascades are denoted as $N_C = (V', T)$, and the associated graph as $N_G = (V', E')$. For any tuple $(u, t) \in N_C$: u and t are a node and the associated timestamp of infection of node u by a certain contagion. An edge $(u, v) \in N_G \Leftrightarrow (u, t_i) \in N_C \wedge (v, t_j) \in N_C \wedge t_i < t_j$.

Consider the diffusion network in Fig. 2 (A), an influence cascade $I_C = \{(u, v, 1), (u, w, 2), (v, x, 4)\}$, a network cascades $N_C = \{(u, 1), (v, 1), (w, 2), (x, 2)\}$, and another network cascade $N'_C = \{(u, 1), (w, 2), (v, 3), (x, 4)\}$. Fig. 2 shows the graphs associated to each cascade as follows: $I_C \equiv (B)$, $N_C \equiv (C)$, and $N'_C \equiv (D)$.

For our purpose, cascades can be simply ordered according to the first time of infection to obtain the associated *linear cascade*. However, observed patterns of frequently co-occurring nodes in cascades are very sparse. In addition, most cascades are incomplete or it is difficult to find them as a coherent atomic element. It has been noted that a significant fraction of the true cascade information is missing [19]. Clearly such problems will propagate to the subsequent network inference task. Therefore, instead of directly

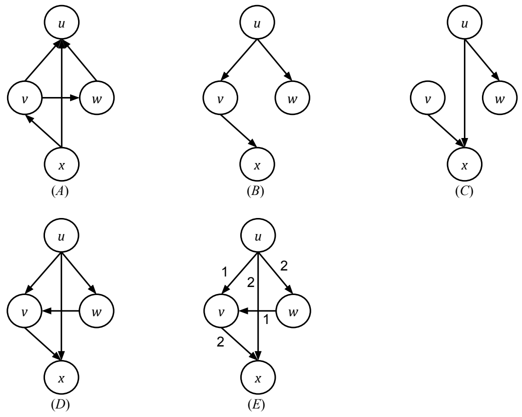

Figure 2: An example of (A) a diffusion network ≡ follower network, (B) ≡ I_C, an influence cascade, (C) ≡ N_C, a network cascade, (D) ≡ N'_C a network cascade, and (E) a weighted graph constructed by combining (C) and (D)

generating a linear cascade associated to the graph structure, we combine all cascade graphs of a particular cascade type into a single graph and sample a number of linear cascades sufficient enough to observe recurring node co-occurrence patterns. Fig. 2(E) shows an example of the combination of the network cascades in Fig. 2 (C) and (D), the weight of an edge indicate the frequency of the edge in the corresponding cascades.

Combining Cascades. We combine the raw cascades of a particular type (i.e. influence or network) into a single unified graph for sampling linear cascades. We shall focus on how the influence cascade graphs are combined briefly, but the same principle holds for the network cascade graphs. Suppose the set of all influence cascade graphs is denoted by $\{I_G\}$. Then we define a function $W : V \times V \rightarrow \mathbb{Z}^+$ that keeps track of the frequency of pairs $(u, v) \in I'_G \in \{I_G\}$ in all the influence cascade graphs. An example is given in Fig. 2(E) for the network cascade graphs in Fig. 2 (C) and (D). For instance, the edge (u, w) is observed in (C) and (D), and hence we add the edge $(u, w, 2)$ in the combined graph (E). The combined graphs for influence and network cascades are denoted by $\mathcal{I}_\mathcal{G}$ and $\mathcal{N}_\mathcal{G}$, respectively.

Cascade Sampling. Once cascade graphs are combined, we sample a number of linear cascades by simulating a number of diffusion processes over $\mathcal{I}_\mathcal{G}$ or $\mathcal{N}_\mathcal{G}$, again we focus on $\mathcal{I}_\mathcal{G}$. Towards this, we assume the original cascades are generated according to the independent cascade (IC) [13] model; thus we employ IC to sample linear cascades. According to IC, a cascade is generated by triggering a diffusion process from a set S_0 of seed nodes, $|S_0| \geq 1$, at time $t = 0$. At each subsequent time step $t > 0$, each node $u \in S_{t-1}$ makes an attempt to spread influence to an outgoing neighbor $v \in \mathcal{I}^{out}_\mathcal{G}(u)$ of u with a probability proportional to $W(u, v)$, if v has not been infected. If the attempt succeeds, we add v to S_t, and each node $u \in S_{t-1}$ is given only one chance to influence each of its uninfected outgoing neighbor. Each linear cascade sample is then added to $\mathscr{C}$ that is utilized during the representation learning module discussed in Section 3.1.

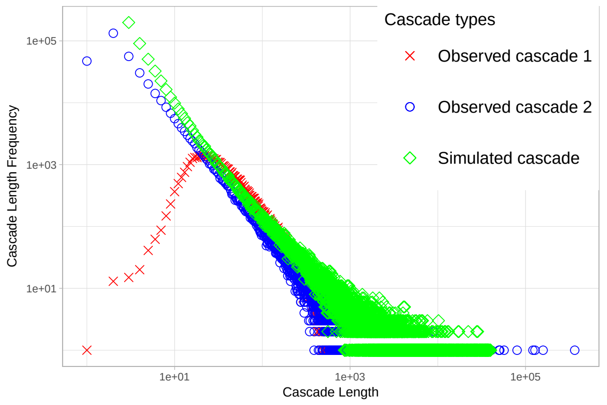

Figure 3: Cascade size distributions for three of our datasets

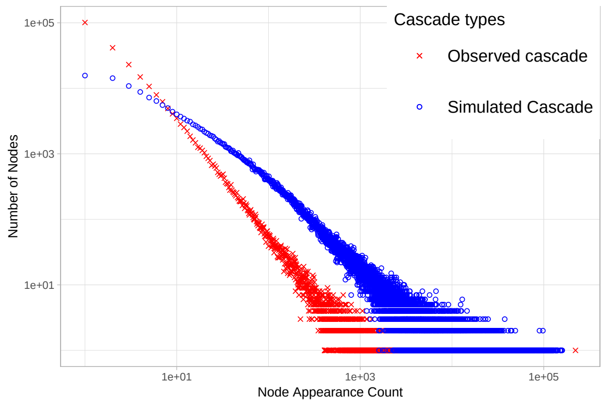

Figure 4: The distribution of nodes occurrence in cascades.

As shown in Fig. 3 and 4, our sampling strategy leads to distributions similar to the original cascades in terms of cascade size and nodes appearances, respectively. Besides, later in the experiments, we demonstrate that this strategy indeed gives better results.

3.1 Representation Learning from Cascades

Essentially, our algorithm at its core leverages representation of nodes learned from linear 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 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, formally the SKIP-GRAM model optimizes the objective in Eq. 1 with respect to the model parameter Φ.

$$\max_{\Phi} \sum_{u \in V} \log Pr(C(u;s)^{\preceq} \mid \Phi(u)) + \log Pr(C(u;s)^{\succeq} \mid \Phi(u)) \quad (1)$$

$$\log Pr(C(u;s)^{D} \mid \Phi(u)) = \sum_{v \in C(u;s)^{D}} \log Pr(v|\Phi(u)) \quad (2)$$

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. 2 is specified using the softmax function as follows:

$$Pr(v|\Phi(u)) = \frac{\exp(\Phi(v)^T \cdot \Phi(u))}{\sum_{w \in N} \exp(\Phi(w)^T \cdot \Phi(u))} \quad (3)$$

Nonetheless, directly estimating the conditional probability in Eq. 3 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 $\log Pr(v|\Phi(u))$ adjusted according to the negative sampling strategy is shown in Eq. 4.

$$\log Pr(v|\Phi(u)) = \log \sigma(\Phi(v)^T \Phi(u)) + neg(u;l) \quad (4)$$

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

$$neg(u;l) = \sum_{i=1}^{l} \mathbb{E}_{w_i \sim \mathcal{N}(u)}[-\log \sigma(\Phi(w_i)^T \Phi(u))] \quad (5)$$

In a nutshell, a good model should minimize the noise model – $neg(u;l)$, and maximize the data-model – the first term on the right-hand-side of Eq. 4.

We employ the stochastic gradient descent algorithm to minimize, instead, the negative log-likelihood of Eq. 1 that utilizes the negative sampling strategy just described. Finally, when the objective is optimized, we obtain the parameters $\Phi \in V \longrightarrow [0,1]^d$.

3.2 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, similar nodes or 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 similar/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:

$$E' = \{(u,v) : p(u,v) \geq \theta\} \tag{6}$$

where

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

Now, we can infer the edges between every pair of nodes by using Eq. 6. 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) [25]. 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.

4 EXPERIMENTS AND RESULTS

In this section, we first introduce the 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, for which a summary of the basic characteristics is provided in Table 1.

(1) Twitter #1 [25]: 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): Retweet information related to the Higgs boson.
   (b) Retweet cascade #2 (RT_2_2 - Simulated cascades): Mentions related to the Higgs boson.

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

(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. In 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: 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 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

[1]http://snap.stanford.edu/data/higgs-twitter.html

[2]http://www.memetracker.org/data.html

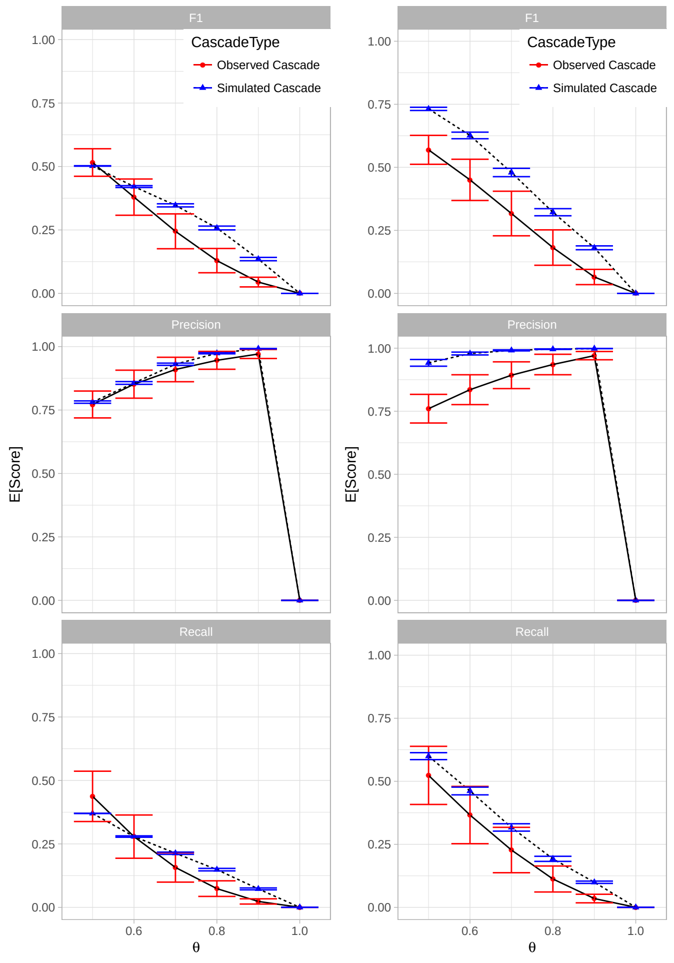

Figure 5: Observed vs Simulated Cascades Performance Evaluation for RT_1_1 (right column) and RT_1_2 (left column). The number of cascades in the observed case of RT_1_1 is 226,488 and RT_1_2 is 217,653.

algorithm; even when the quality is scarcely affected, the training time increases significantly [17]. Moreover performance is also related to the size of the training data, Note that for the small datasets (RT_2_1, RT_2_2) the error-rate is almost as good as the baseline (50%).

The next experiment analyzes 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

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

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 ≈ 96% and ≈ 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 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

Representation Learning. Recent advances in neural network models have attracted researches from several communities such as

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

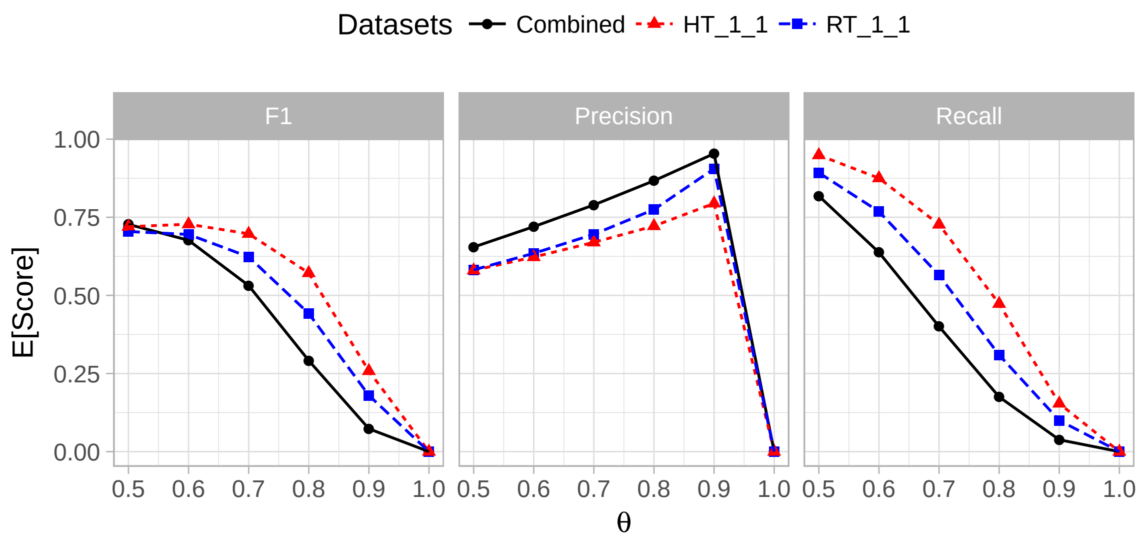

Figure 6: Combined vs separate cascades performance evaluation on window size, $s = 5$

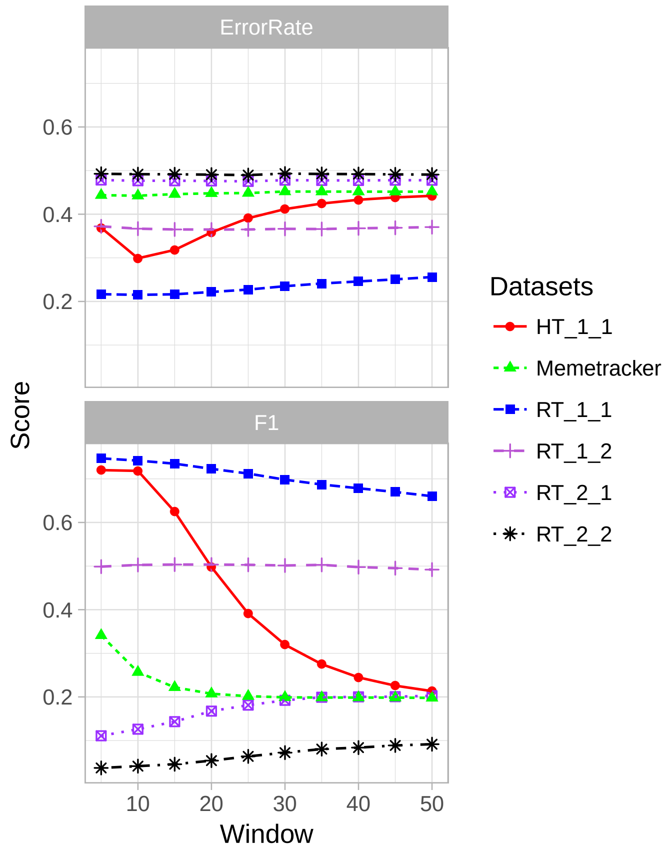

Figure 7: Evaluation on the effect of increasing window size

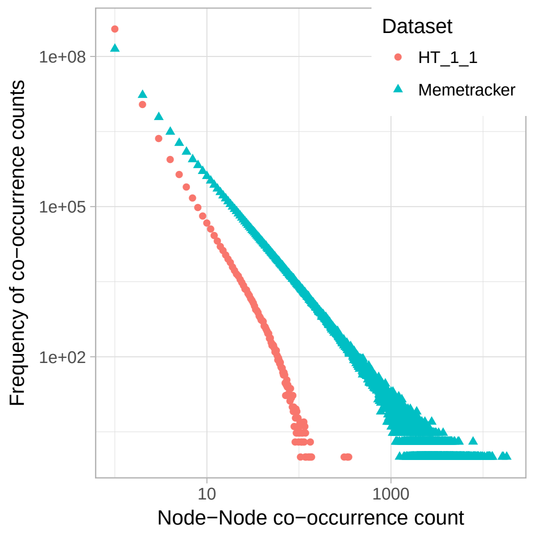

Figure 8: Distribution of node-node pair co-occurrences

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, 20, 22], and community structures [21, 24]. 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.

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 based on 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 a number of 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 adopt our framework to deep-models and also 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 *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 238–247.

[2] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. 2016. Representation Learning for Information Diffusion Through Social Networks: An Embedded Cascade Model. In *Proc. of the 9th ACM Int. Conf. on Web Search and Data Mining (WSDM '16)*. ACM, 573–582.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *Proc. of the 24th ACM Int. on Conf. on Information and Knowledge Management (CIKM '15)*. ACM, 891–900.

[4] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can Cascades Be Predicted?. In *Proc. of the 23rd Int. Conf. on World Wide Web (WWW '14)*. ACM, 925–936.

[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 *Proc. of the 25th Int. Conf. on Neural Information Processing Systems (NIPS'12)*. Curran Associates Inc., 2780–2788.

[7] Nathan Eagle, Alex (Sandy) Pentland, and David Lazer. 2009. Inferring friendship network structure by using mobile phone data. *Proc. of the National Academy of Sciences* 106, 36 (2009), 15274–15278. 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 *Proc. of the 28th Int. Conf. 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 *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '10)*. ACM, 1019–1028.

[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 *Proc. of the 22Nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 855–864.

[13] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the Spread of Influence Through a Social Network. In *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '03)*. ACM, 137–146.

[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 *Proc. of the 2015 IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining 2015 (ASONAM '15)*. ACM, 178–185.

[15] Jure Leskovec, Lars Backstrom, and Jon Kleinberg. 2009. Meme-tracking and the Dynamics of the News Cycle. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '09)*. ACM, 497–506.

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

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proc. of the 26th Int. Conf. on Neural Information Processing Systems (NIPS'13)*. Curran Associates Inc., 3111–3119.

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '14)*. ACM, 701–710.

[19] Eldar Sadikov, Montserrat Medina, Jure Leskovec, and Hector Garcia-Molina. 2011. Correcting for Missing Data in Information Cascades. In *Proc. of the Fourth ACM Int. Conf. on Web Search and Data Mining (WSDM '11)*. ACM, 55–64.

[20] 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).

[21] 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

[22] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proc. of the 22Nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 1225–1234.

[23] Liaoruo Wang, Stefano Ermon, and John E. Hopcroft. 2012. Feature-Enhanced Probabilistic Models for Diffusion Network Inference. In *Proc. of the 2012 European Conf. on Machine Learning and Knowledge Discovery in Databases - Volume Part II (ECML PKDD'12)*. Springer-Verlag, 499–514.

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

[25] L. Weng, F. Menczer, and Y.-Y. Ahn. 2013. Virality Prediction and Community Structure in Social Networks. *Sci. Rep.* 3, 2522 (2013).