Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM 2020)

Influence Maximization Using Influence and Susceptibility Embeddings

George Panagopoulos
École Polytechnique
george.panagopoulos@polytechnique.edu

Fragkiskos D. Malliaros
Université Paris-Saclay
CentraleSupélec, Inria
fragkiskos.malliaros@centralesupelec.fr

Michalis Vazirgiannis
École Polytechnique
AUEB
mvazirg@lix.polytechnique.fr

Abstract

Finding a set of users that can maximize the spread of information in a social network is an important problem in social media analysis — being a critical part of several real-world applications such as viral marketing, political advertising and epidemiology. Although influence maximization has been studied extensively in the past, the majority of works focus on the algorithmic aspect of the problem, overlooking several practical improvements that can be derived by data-driven observations or the inclusion of machine learning. The main challenges of realistic influence maximization is on the one hand the computational demand of the diffusion models' repetitive simulations, and on the other the accuracy of the estimated influence spread. In this work, we propose CELFIE, an influence maximization method that utilizes learnt influence representations from diffusion cascades to overcome the use of diffusion models. It comprises of two parts. The first is based on INF2VEC, an unsupervised learning model that embeds influence relationships between nodes from a set of diffusion cascades. We create a new version of the model, based on observations from influence analysis on a large scale dataset, to match the scalability needs and the purpose of influence maximization. The second part capitalizes on the learned representations to redefine the traditional live-edge model sampling for the computation of the marginal gain. For evaluation, we apply our method in the Sina Weibo and Microsoft Academic Graph datasets, two large scale networks accompanied by diffusion cascades. We observe that our algorithm outperforms various baseline methods in terms of seed set quality and speed. In addition, the proposed INF2VEC modification for influence maximization provides substantial computational advantages in the price of a minuscule loss in the influence spread.

Introduction

With the advent of the social web, social media have been broadly utilized to spread news, establish trends or even shape public opinions. This has motivated a substantial amount of research on the analysis of how social media users effect each other. Traces of such influence can be detected in the users sharing activity, such as tweets, Facebook posts, Instagram stories etc. Formally, social influence is defined as a directed measure between two users and represents how possible is for the target user to adapt the behavior or copy the action of the source user. The research on social influence can be divided into two main branches that address different problems: influence maximization and influence learning.

Influence maximization is typically formulated as a combinatorial optimization problem where the aim is to find the set of nodes in a network that would maximize the reach of a diffusion cascade starting from them (Kempe, Kleinberg, and Tardos 2003). Although the problem was inspired by viral marketing (Domingos and Richardson 2001), it can be applied to several other disciplines, such as limiting misinformation (Budak, Agrawal, and El Abbadi 2011), opinion polarization (Garimella et al. 2017) etc. Influence maximization algorithms face scalability issues primarily because the computation of the number of nodes infected by a set of seed users, called the influence spread, relies on repetitive simulations of diffusion models (Chen, Wang, and Wang 2010a). Apart from the computational hinder, diffusion models are based on several non-realistic assumptions that may guide the algorithm in unreliable results (Panagopoulos, Malliaros, and Vazirgiannis 2018). More specifically, influence maximization utilizes stochastic diffusion models such as the independent cascade and the linear threshold with uniform probabilities of influence between nodes. This random influence assignment, together with the simplistic assumptions governing the diffusion models, produces non-realistic estimations of the influence spread (Aral and Walker 2012). Moreover, influence dynamics do not depend solely on the structure of the network, but time is of essence as well (Karsai et al. 2011). In addition, diffusion models are Markovian, meaning that they fail to capture any higher order memory effects that are present in influence paths — a disadvantage we address further in this work. In general, these solutions depend on the structural information and oversimplified diffusion models, overlooking how information diffusion really take place over the network. However, along with new datasets that contain the network and diffusion cascade logs (Lerman, Ghosh, and Surachawala 2012), emerged the question of how to measure influence in a data-driven manner.

Influence learning has attracted significant attention the past few years. Initially, these methods were learning parameters that define the influence probabilities of diffusion models (Gomez-Rodriguez, Balduzzi, and Schölkopf 2011), or

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

the edge probabilities themselves (Goyal, Bonchi, and Lakshmanan 2010). However, these approaches suffered from overfitting, as the number of parameters was equal to the number of edges. To address this, more recently, an array of works utilize representation learning to capture influence between two nodes. A representative example is EMBEDD-IC (Bourigault, Lamprier, and Gallinari 2016), an adaptation of the independent cascade where the influence probabilities are expressed as a combination of the nodes' embeddings. Each node is associated with two embeddings, a source and a target, which combined construct the probability of influencing and getting influenced. This basic idea has been developed into involved neural architectures that can be used to predict the evolution of a diffusion cascade, such as the next infected node, and the time of the next infection (Du et al. 2016; Islam et al. 2018). One important problem with influence learning techniques that are based on diffusion models (Bourigault, Lamprier, and Gallinari 2016), is that they assume influence independence between edges. This assumption does not allow the model to capture influence similarities between nodes of similar status, which exist because of the assortative mixing property (Newman 2002). Moreover, it causes a substantial depreciation in the estimation of the influence spread (Aral and Dhillon 2018), which is why in this work, we utilize a model that learns influence and susceptibility embeddings based on the co-occurence in diffusion cascades (Feng et al. 2018).

In this study, we propose CELFIE, a fast influence maximization method based on diffusion probabilities instead of diffusion models. Our model capitalizes on the information contained in past diffusion cascades using influence and susceptibility representations learnt specifically for this setting. We start from the representation learning part. In order to improve the scalability of our model and compute diffusion instead of influence probabilities, we device a modification of the recently proposed influence embedding model INF2VEC (Feng et al. 2018), based on observations from a data analysis on a large scale dataset. As we will see in detail in the next section, INF2VEC computes influence and susceptibility embeddings based on time precedence in diffusion cascade history and follow relationships in the network. We adjust it to focus on initiators of the cascades and compute diffusion probabilities, which are influence probabilities between nodes with more than one hop distance in the network. We use the definition of this probability to reformulate the problem as influence maximization in a bipartite network. In this setting with no higher order paths, each seed directly infects a specific node, thus the stochastic existence of an influence edge effects only the infection of the node in the receiving end, and the need for diffusion models is eliminated. We take advantage of this to define a new marginal gain based on a sampling strategy similar to the live-edge model, but diminishing the sampling space due to past infections and reusing a seed's past samples to compute the new influence spread. Finally we propose a greedy solution that optimizes the marginal gain using the well-known Cost Effective Lazy Forward (CELF) algorithm. We evaluate our methodology in two large scale datasets, the Sina Weibo that contains both, a follow network and retweet cascades (Zhang et al. 2013), and the Microsoft Academic Graph for computer science papers, which is comprised of a co-authorship network, and citation cascades (Sinha et al. 2015) . The quality of the retrieved seed set is determined by a set of unseen diffusion cascades from future time steps, similar to a train and test split in machine learning (Panagopoulos, Malliaros, and Vazirgiannis 2018; Du et al. 2013). We have observed that our algorithm outperforms various baselines methods in terms of seed set quality, and that the proposed INF2VEC modification provides substantial computational advantages in the price of a minuscule loss in the influence spread.

The rest of the paper is organized as follows. We start by briefly reviewing the required background. Then, we delineate influence representation learning, providing the data analysis study that served as motivation for the adjustment of INF2VEC. Then, we describe the reformulation of the live-edge model using diffusion probabilities and the proposed influence maximization algorithm. We present the experimental design, including the baseline methods and the justification of the results. In the last section, we conclude the paper with a contribution synopsis and suggestions for future work.

Background

Network Representation Learning

NODE2VEC (Grover and Leskovec 2016) is a popular representation learning technique that capitalizes on the structure of the network to derive node vectors (representations) that translate the structural correlations of the nodes into a real space. The learning algorithm resembles WORD2VEC (Mikolov et al. 2013), where each word is a node and each context is derived by a balanced random walk starting from that node. The intuition is that nodes that occur in the context of a node should have similar structural properties, hence their embeddings must be similar. These vectors can then be utilized for classification, clustering and visualization purposes with remarkable results. The INF2VEC model (Feng et al. 2018), similar to NODE2VEC, is a shallow neural network that receives as input a node of the network, and outputs a probability distribution over all other nodes getting influenced by this node. The training data is a pair of node-context derived by a set of diffusion cascades. More specifically, each cascade is first transformed into a diffusion network, by drawing directed edges from a node in the cascade, to other nodes that appear later in that cascade and have actual edges in the respective network (e.g., follow relationships). Subsequently, a node-context pair is constructed for each node in this diffusion network, where the context is comprised by 5 nodes derived by a random walk with restart starting from the node under examination, and 45 nodes randomly sampled from the diffusion network.

The hidden layer of the network represents the source embeddings O of the nodes, and the output layer the target T. The likelihood for a node u to influence v is given by:

$$f_{v,u} = O_u \cdot T_v + b_u + b'_v. \quad (1)$$

Note here the use of b_u and b'_v. These two one-dimensional embeddings substitute the bias and aim to capture the source

node's general capacity to influence other nodes and the susceptibility of the target node to be influenced. The output of the model, which is a probability distribution over all nodes, is created through a Softmax function $\phi(\cdot)$:

$$p_{v,u} = \phi\big(f_{v,u}\big) = \frac{e^{f_{v,u}}}{\sum_{u \in G} e^{f_{v,u}}}. \quad (2)$$

Due to the number of nodes, calculating the denominator of Eq. (2) is too computationally expensive, so a negative sampling approach is employed. The loss function samples randomly 10 nodes w that act as counter examples to the actual node u that occurred in the pair with v. The final log-likelihood uses the sigmoid s and is formulated as:

$$\log(p_{v,u}) = \log\big(s(f_{v,u})\big) + \sum_{w \in N} \log\big(s(-f_{v,w})\big). \quad (3)$$

Influence Maximization

The greedy algorithm for influence maximization (Kempe, Kleinberg, and Tardos 2003) starts with an empty seed set S and adds iteratively the node that provides the best marginal gain, i.e., the maximum increase of the set's influence spread. Influence spread is the number of nodes infected by a given seed set and is computed by simulating the progress of a diffusion cascade through the network using two basic models: the Independent Cascade (IC) and the Linear Threshold (LT). Instead of running simulations of these stochastic processes, the algorithm estimates the influence spread based on the live-edge model:

$$\sigma(S) = \sum_{g \subseteq G} P(g)\sigma_g(S) \quad (4)$$

$$P(g) = \prod_{e \in E(g)} p_e \prod_{e' \in E(G)/E(g)} (1 - p_{e'}) \quad (5)$$

where g is derived by randomly sampling edges from the network G. This Monte Carlo (MC) sampling facilitates converging to an unbiased estimate of the actual influence spread of both, IC and LT. An exact computation would require taking into account all possible realizations of the network, which amounts to $\sum_{k=0}^{e} \binom{e}{k} = 2^e$ combinations, where e is the number of edges. From the point of view of the diffusion process, this computation would be necessary because both stochastic processes rely on the influence probabilities of edges, hence both need to take into account all possibilities to compute the estimated spread. The influence spread $\sigma_g(S)$ under the specific network realization g, is computed by the nodes that are reachable through a path of live-edges from the seed set S. The probability of the realization, which is derived by the influence probabilities of the edges in Eq. (5), weighs the influence spread in order to quantify the likelihood of the specific spread occurring.

The function of the influence spread under the IC and LT diffusion models is monotonic and submodular. Submodularity means that as the seed set increases the contribution of a node to the influence spread (marginal gain) can only decrease or stay the same:

$$\sigma\big(S \cup \{w\}\big) - \sigma\big(S\big) \geq \sigma\big(S' \cup \{w\}\big) - \sigma\big(S'\big), \quad (6)$$

where $S \subseteq S'$. Due to this, the algorithm is guaranteed to reach a solution that approximates the optimal within a factor of $(1 - 1/e)$. However, the influence spread estimation is still prohibitively time consuming for real-world networks. Several methods have been developed that achieve notable acceleration and retain the theoretical guarantees using sketches (Cohen et al. 2014) or reverse reachable sets (Tang, Xiao, and Shi 2014). Moreover numerous heuristics have been proposed that, though lacking guarantees, exhibit remarkable success in practice (Goyal, Lu, and Lakshmanan 2011; Chen, Wang, and Wang 2010b). One general problem with most of these approaches is that they rely on the diffusion models, and the influence probabilities are derived by the weighted cascade model or uniformly at random (Kempe, Kleinberg, and Tardos 2003). This calls into question the quality of the retrieved seed set, as more and more studies indicate that structure alone can not capture the properties of influence and diffusion (Karsai et al. 2011; Pei, Morone, and Makse 2018). There has been a number of attempts that examine influence maximization using diffusion cascades, either directly (Goyal, Bonchi, and Lakshmanan 2011; Panagopoulos, Malliaros, and Vazirgiannis 2018) or by weighing the network based on the activities in diffusion cascades (Goyal, Bonchi, and Lakshmanan 2010), but overall, it is a largely overlooked field.

Influence Representation Learning

An important question in influence analysis is whether successful sharing or resharing contributes more to an influencer's popularity. In other words, does it suffice for an influencer to create massive cascades, or she has to participate in strong ones started by others as well? The sampling process of the INF2VEC algorithm described above, assumes the participation of a node in other cascades is important, as node-context pairs are derived for each node in each diffusion cascade. Although this is definitely a more complete approach, given that the average size of a cascade can surpass 60 nodes in today's datasets, it is very time consuming.

We examined this question in the diffusion cascades of the *Weibo* dataset, a network consisting of more than 1.7 million nodes and 0.4 billion edges, accompanied by a set of 300,000 retweet cascades spanning 3 years of recordings. The network is retrieved at the final month of these 3 years. The diffusion cascades are formed by gathering the past 1,000 tweets of each node in the network. The original posts in these tweets represent the start of a cascade, and any tweet with the same content, that is not an original post and is preceded by the initial tweet, is considered a retweet. We should underline here that the original post is not defined based on time precedence, we know which tweet is the original post because it is the only tweet of a given content that is not marked as a retweet.

In our case, we have separated the dataset in train and test cascades based on their time of occurrence. We keep the 18,652 diffusion cascades from the last month of recording as test set and the 97,034 from the previous 11 months as train set. We call them test and train set, similarly to machine learning, because we will use them for evaluation of

our algorithm, as we describe in the experimental section of the paper.

To evaluate our hypothesis that successful influencers mostly start diffusion cascades rather than participate in them, we will first rank all seed users found in the test set based on three measures of success: the number of test cascades they spawn, their cumulative size and the number of Distinct Nodes Influenced (DNI) (Panagopoulos, Malliaros, and Vazirgiannis 2018). We separate all users in three categories for each metric, based on how high they rank on it. Then, we compute for each category the total cascades they start in the training set and those they simply participate in. As we can see in Figure 1, all users that belong to the top $1/3$ of the test cascades in all three metrics are far more likely to start a train cascade than simply participate in it.

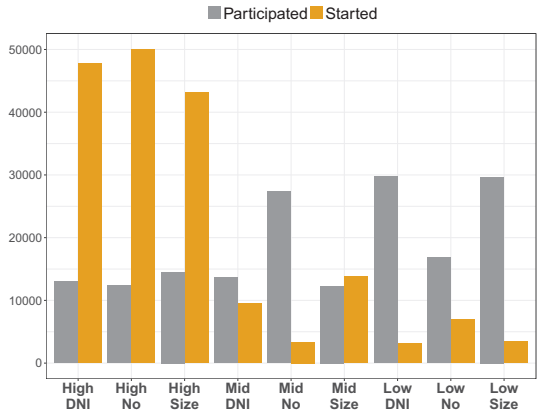

Figure 1: Influencer's initiating versus participating in diffusion cascades. **DNI** stands for distinct nodes influenced, **size** stands for cascade size, and **no** stands for number of cascades started. Each category in the x-axis defines where these users belong in the test cascades, and the bar indicates how many train cascades they start opposing to how many they just participate in.

To this end, we propose a new approach to the creation of the input to the neural network behind INF2VEC. Instead of deriving one node-context pair for each node in the diffusion, we can derive a single one only for the initiator of the cascade. Apart from the significant computational gain, we expect this approach to perform adequately due to the above observation. Moreover, instead of creating the diffusion network, which is time consuming and prone to noise due to the absence of edges (Panagopoulos, Malliaros, and Vazirgiannis 2018), we can perform another type of sampling to derive the context of the initiator.

A very important characteristic in the study of social influence is its temporal dynamics. In the case of information diffusions, the time passed between two node's activity is known to play a role in the amount of influence the source node exerts to the target (Goyal, Bonchi, and Lakshmanan 2010). Yet, the original INF2VEC algorithm does not capitalize over this attribute. In our dataset, we can observe this phenomenon by studying the copying times in the diffusion cascades of the influencers. As mentioned above, we focus on the nodes that have started a cascade in the test set and use as a measure of a node's influence the number of nodes it infects throughout all these cascades (DNI). We compute the average copying time of a cascade, for all train cascades of these nodes, and average it to get an estimate of how fast these nodes get "copied" during their cascades. Subsequently, we group these nodes based on their DNI and compute the average copying time of each group and plot it opposed to the DNI in Figure 2.

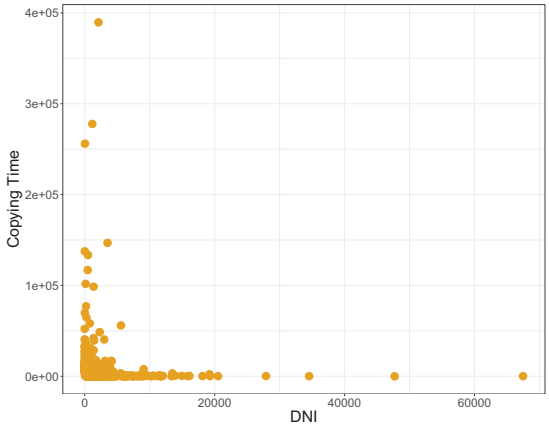

Figure 2: Average copying time in the train cascades relative to the number of distinct nodes influenced in the test cascades.

This plot indicates that nodes with higher influence tend to have much faster cascades than ordinary influencers. Of course, there are much more nodes with smaller DNI than high, but since we take the average of the copying time for each DNI, the result is not affected. Apart from confirming our intuition, this observation buttress several findings that underline the decay of influence as the copying time increases in real-world diffusion cascades (Goyal, Bonchi, and Lakshmanan 2010). To this end, we guide the sampling of a node's context based on the copying times in its cascades. More specifically, we form the context by performing a random sampling over the nodes in the cascade, where the probability is inversely proportional to the copying time between the candidate node and the initiator. For the current analysis, we do an oversampling of 120% to emphasize the importance of fast resharing in the depiction of influence. This creates node-context pairs that take into account the temporal dynamics of the cascades.

As a final step, we observed that the influenceability and susceptibility embeddings did not increase the accuracy of INF2VEC in our task, so we substituted them with a simple bias and used noise contrastive estimation over negative sampling (Gutmann and Hyvärinen 2010). The changes above were driven from observations in the data and aim primarily to alleviate the extensive computational demand of INF2VEC. We name the new model light INF2VEC (L-INF2VEC), and later on we will compare it to the original INF2VEC in our task. It should be noted that L-INF2VEC,

apart from significant computational advantages, is devoid of the use of the diffusion network, and consequently the underlying network itself, as it uses only the diffusion cascades. Moreover, we overcome the sampling process of INF2VEC which requires the creation of the propagation network that has a complexity of $\mathcal{O}\Big(c(\bar{n}(\bar{n}-1)/2)\Big)$, where c is the number of cascades and $\bar{n}$ is the average cascade size. This is the case because it requires going through every node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network. It should be noted here that our representation learning model's final purpose is influence maximization, which is why we focus so much on the activity of initiators and overlook the rest of the nodes. If we aimed for another task, such as predicting the course of a diffusion or recommending friendships, it is unclear whether this type of context-creation would be effective.

Influence Maximization with Influence Embeddings

The likelihood influence score $p_{v,u}$ is derived from Eq. (1) and Eq. (2). From this we can build the *influence likelihood matrix* (ILM) as:

$$ILM = \begin{bmatrix} \phi(D_1) \\ \vdots \\ \phi(D_I) \end{bmatrix}, \quad D = OT^\top + B + B', \qquad (7)$$

where $\phi(\cdot)$ is the Softmax function, O is the hidden layer matrix (embeddings of source nodes), T is the output layer matrix (embeddings of target nodes), B is a matrix containing the embeddings that capture the ability of each node to influence others (b_u in Eq. (1)) repeated for I rows, where I is the number of nodes that started more then one cascade, and B' is the embeddings of influence susceptibility repeated for N columns. The Softmax function from Eq. (2) is applied in every row of the matrix to derive the diffusion probabilities. In the case of L-INF2VEC, we compute ILM in the same manner as in Eq. (7), without any bias.

Reformulation of the Problem

The main idea is to substitute the aforementioned computation of the influence spread that is based on the diffusion models, with a function that utilizes the influence likelihood scores derived by INF2VEC. As a first step, we observe that the influence spread under a specific live-edge sampling in Eq. (4) is determined by the paths formed by the sampled edges. Intuitively, in our case the influence likelihood score in Eq. (7) stands for the probability of a node appearing in a diffusion started by a seed (source node), independently of the two nodes' distance in the network. We can name this diffusion probability, although it has been used in the past to describe influence probability between two nodes in the network (Kalimeris et al. 2018), their difference is clear. This means that these probabilities *implicitly* include the underlying influence paths from the seed to the infected node and we can reformulate the live-edge model to take advantage of it. More specifically, we can interpret the ILM matrix as a fully connected bipartite network, where each of the nodes in the left side is connected to all other nodes in the right side. The left side nodes are the candidate seeds, and each of them can influence every node in the right side. In this manner, all the paths with length more than 1 are removed, so we do not require a diffusion model to estimate the spread. This is also the reason why we have no connections between the nodes at the same side, as there are no higher order influence paths and each seed can influence a node only through their direct edge.

For example, in the traditional setting a node u might be able to influence another node z by influencing node v between them, as shown in Figure 3. However, in our case, if u could indeed induce the infection of z in a direct or indirect manner, it would be depicted by the diffusion probability $p_{u,z}$. Thus we can remove all influence paths with length more than 1, as they are captured by the diffusion probabilities. From a data-driven perspective, this approach captures the case when v appears in the diffusions of u and z appears in the diffusions of v but not in u's. This happens in cases where node v publishes or reshares different types of content. When v reshares content from u, it diffuses in different directions than z. Thus, it would be wrong to assume u's infection is able to eventually cause z's. Typical influence maximization algorithms that rely on diffusion models fail to capture this effect of higher order correlations. A diffusion model acts in a Markovian manner and can spread the infection from u to z. The biggest advantage of this "shortcut" is that it allows us to overlook the complexity induced by the diffusion models.

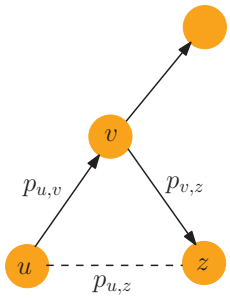

Figure 3: Toy example of higher order influence to depict the difference between influence and diffusion probabilities between nodes.

Assuming the influence probabilities are independent, the probability of a node u getting influenced by a seed set S is the complementary probability of not getting influenced by each node $v \in S$. Summing this over all non-seed nodes can give a new influence spread:

$$\sigma'(S) = \sum_{u \in G} \Big(1 - \prod_{v \in S} (1 - p_{v,u})\Big). \qquad (8)$$

To transform this into a set function that computes the infected nodes, we can use a threshold, meaning that a node will get infected if its probability of getting influenced by the seed set at this step is equal to or more than 0.5, which is the value used in classifiers with softmax output. Unfortunately, it is easy to see that this function is not submodular.

Think of a toy example with two source nodes a, b that can influence three other nodes with probabilities $[0.6, 0.3, 0.5]$ and $[0.3, 0.4, 0.4]$ respectively. In the first step, the algorithm will choose a as it gives 2 infected nodes opposing to b that gives 0. In the second step, following our definition, the addition of b will infect the second and final node, thus the property of diminishing returns does not stand for this influence spread and we can not utilize the greedy algorithm. Although the set function defined above is not submodular, function $\sigma'(\cdot)$ in Eq. (8) itself is convex. From it and Eq. (7) we can derive that:

$$\sigma'(S) = |G| - \sum_{u \in G} \prod_{v \in S} \big(1 - ILM[v, u]\big), \tag{9}$$

which leads to the following optimization problem:

$$\text{argmax}_S(\sigma'(S)) = \text{argmin}_S\Big(\sum_{u \in G} \prod_{v \in S} \big(1 - ILM[v, u]\big)\Big). \tag{10}$$

Since there is no easy way to directly optimize this function with guarantees, we will rely on a greedy heuristic. The method simply iterates throughout all nodes in each turn and adds in the seed set the node that minimizes the function above, taking into account the seed set up to that point. This will serve as a baseline for our experiments.

CELF with Influence Embeddings

As we saw above, directly minimizing the probability of a node getting influenced by the seed set is not a straight forward option. Thus, we can adopt the traditional approach to influence maximization and assume there is an underlying diffusion model, but instead of running over an influence network, it is based on the learned diffusion probabilities. To be more specific, we can use the bipartite network as defined by matrix ILM and follow the greedy algorithm (Kempe, Kleinberg, and Tardos 2003) to find the optimum seed set. Moreover, we can tailor the algorithm to increase significantly the speed of the process by taking advantage of the network's form. Particularly, one can observe that a seed's influence spread in the bipartite network is related only to its edges and the edges of the rest of seed set. This characteristic can induce a drastic change to the computation of the influence spread in Eq. (4).

In the typical live-edge model, edges are randomly sampled from the network and their joint probability in Eq. (5) is used to weigh the influence spread. During this process, edges that are unrelated to the seeds can be sampled, because they may form a new path from a seed to multiple uninfected nodes, which causes the computation of the joint probability of the whole network in Eq. (5). However, in our case sampling an edge that is unrelated to the seed set will never produce an infected node, because we do not have any paths. Thus, in the sampling process of this model, we do not have to take into account the probability of the whole network, but only of the seed set under consideration. We can update $P(g)$ in Eq. (4) to depict this effect as follows:

$$P(g) = \prod_{s \in S} \prod_{e \subseteq \bar{N}} p_e \prod_{e' \in N(s) \setminus \bar{N}} (1 - p_{e'}), \tag{11}$$

where $N(s)$ are the neighbors of the seed s, and $\bar{N} \sim N(s)$ is a sample of them of size ϵ. This not only allows to restrain our sampling to the seed at hand, but also separate the sampling of the seed set with the sampling of the candidate seed. In this way instead of sampling from the whole network, computing the sample's joint probability and finding the live paths from the seed set in each simulation, we can sample solely for each candidate seed and combine it with the sampling of the seed set in the respective simulation from the previous iterations. To this end, we can redefine the computation of the marginal gain and the algorithm itself.

To measure the marginal gain of a candidate seed c, we use Algorithm 1. We perform a fixed number of simulations, where each one consists of sampling a fixed number of c's edges (line 4), according to their probabilities. Subsequently, we unite these nodes with the ones that have been infected by the seed set in that respective simulation (line 5), which gives the candidates influenced set. The cumulative quantity from all simulations subtracted by the same quantity of the seed set up to now defines the marginal gain of c, as in line 18 and 19 of the main algorithm (2), which is an adapted implementation of CELF (Leskovec et al. 2007). Variables l is the number of simulations and ϵ is the number of edges to sample during each simulation. The list q is used to store a node's id and influence set, as well as the iteration it was last updated. The CELF with Influence Embeddings (CELFIE) algorithm, like CELF, capitalizes on the submodularity of the marginal gain to minimize the computations. The intuition is that a node that has higher marginal gain during iteration t than other nodes have at $t-1$, will retain its lead at t. Hence, the computation of the marginal gain is eliminated for those nodes that remain lower in the list though they have not been evaluated with the current seed set yet. This is achieved by taking always the first node in the list q, if it was updated with the current seed set (line 11) and resorting the list when a candidate seed is evaluated with the current seed set (lines 16 & 21).

Algorithm 1 MARGINAL INFLUENCE SET (MIS)

Input: Influence likelihood matrix ILM, index of candidate seed c, influenced set up to now inf_set,number of simulations l, number of edges to sample ϵ
Output: Updated influenced set inf_set

procedure MIS($ILM, c, inf\_set, l, \epsilon$)
set gain = 0, n = size(ILM)[0]
for $i \leftarrow 0; i < l; i++$ **do**
  $c\_inf \leftarrow Sample(\epsilon, n, ILM[c, :])$
  $inf\_set[i] = inf\_set[i] \cup c\_inf$
return inf_set

Experiments

Evaluation Methodology

As mentioned above, we split each dataset into train and test cascades based on their time of occurrence. For the structural information required to form the diffusion network in the original INF2VEC algorithm, we utilize the

Algorithm 2 CELFIE

Input: Influence likelihood matrix ILM , number of simulations l, number of edges to sample ϵ, seed set size $size$
Output: Seed set S

procedure CELFIE($ILM, l, \epsilon, size$)
  set $q = [], S = []$
  for $i \leftarrow 0; i < l; i++$ **do**
    $inf\_set[i] \leftarrow \varnothing$
  for $c \leftarrow 0; c < N[1]; c++$ **do**
    $infs \leftarrow MIS(ILM, c[0], inf\_set, l, \epsilon)$
    $q.append([c, infs, 0])$
  $q \leftarrow Sort(q, 1, Desc = True)$
  while $S.size() < size$ **do**
    $c \leftarrow q[0]$
    if $c[2] == size(S)$ **then**
      $S.add(c[npos])$
      $inf\_set = c[1]$
      $q.delete(c)$
    else
      $infs \leftarrow MIS(ILM, c[0], inf\_set, l, \epsilon)$
      $mg \leftarrow 0$
      for $i \leftarrow 0; i < l; i++$ **do**
        $mg+ = |infs[i]| - |inf\_set[i]|$
      $c \leftarrow [mg, infs, |S|]$
      $q \leftarrow Sort(q, 1, Desc = True)$
  return S

	Weibo	*MAG-CS*
Nodes	1,170,689	1,436,158
Edges	225,877,808	15,928,078
Cascades	115,686	181,020
Avg Cascade size	148	29

Table 1: Summary of the datasets used.

underlying network. Our evaluation tactic is to compute the number of distinct nodes influenced in the test diffusion cascades by the predicted seed set (Du et al. 2013; Panagopoulos, Malliaros, and Vazirgiannis 2018). We consider as influenced every node that participates in a test set diffusion initiated from one of the predicted seeds. Meaning, each seed set has initiated a number of cascades in the test set. The set of infected nodes (DNI) is computed by adding every node appearing in these test cascades in a unified set. We use the size of this set to measure the success of the seed set in the test set. Since we measure the size of the distinct set, potential overlaps between diffusions of different seeds are taken into account, which would not be the case had we use more simplistic measures such as the sum of the seeds' average cascade size. Although not devoid of assumptions, it is the closest and most objective measure of a seed set's influence over a network at a given time span.

Datasets

- *Sina Weibo*: A directed follower network where a cascade is defined by the first tweet and the list of retweets (Zhang et al. 2013). We keep the last month of cascades as the test set and the previous 11 as train set, in an approximate 80%-20% split. We also remove from the network nodes that do not appear in the cascades, in order to make more fair the comparison between structural and diffusion-based methods, since the evaluation relies on diffusions.
- *MAG Computer Science*: We follow suit from (Qiu et al. 2018) and define a network of authors with undirected co-authorship edges, where a cascade happens when an author writes a paper and other authors cite it. In other words, a co-authorship is perceived as a friendship, a paper as a post and a citation as a repost. In this case, we employ Microsoft Academic Graph (Sinha et al. 2015) and filter it to keep only papers that belong to computer science. We remove cascades with length less than 10. The train and test split happens in a similar manner to *Weibo*, keeping the first 80% of diffusion cascades for train set and the next for test set.

Although the metaphor between the paper and tweet cascades is not perfect, since a citation between two nodes might happen without requiring any structural relationship (e.g. a path of co-authorships linking them) in contrast to a social network, the idea that scientific information diffuses like this through the network of authors is valid. A subtle point is that a citation cascade does not start from just one node, since in most cases a paper has numerous authors. We thus treat the citation cascade started by a paper as n separate cascades initiated by the n authors of the paper and spreading through the same nodes (the authors of the papers that cite the initial paper). To restrict the number of cascades, we remove those with length less than 10 i.e., papers with less then 10 citations.

Baselines

We should note here that juxtaposing the effectiveness of structural and diffusion metrics as well as influence maximization algorithms is something missing from the literature, to the best of our knowledge, especially in the scale of the networks utilized.

- K-CORE: The top 100 nodes in terms of their core number, as defined by the undirected k-core decomposition (Malliaros et al. 2020).
- AVERAGE CASCADE SIZE: The average number of retweets that a user has in the train set.
- DIFFUGREEDY: An influence maximization algorithm that uses directly the diffusion cascades of a node to compute the marginal gain (Panagopoulos, Malliaros, and Vazirgiannis 2018).
- CREDIT DISTRIBUTION: Uses cascade logs to assign influence credits based on time precedence *and* the edges of the network to derive a seed set for IM (Goyal, Bonchi, and Lakshmanan 2011). (Parameter λ is set 0.001).
- HEURISTIC: The heuristic defined in the "Reformulation of the Problem" section using the L-INF2VEC embeddings.

- CELFIE: CELFIE with the influence embeddings from the original INF2VEC model.
- L-CELFIE: CELFIE with L-INF2VEC embeddings.

We cannot employ traditional influence maximization algorithms like greedy or CELF and CELF++, because their scalability is limited to networks with less then 200K edges. Even fast heuristics like SIMPATH (Goyal, Lu, and Lakshmanan 2011) or PMIA (Chen, Wang, and Wang 2010a), can not scale in networks with tens of millions of edges. Moreover, as mentioned above, methods that rely on simulations rather than data-driven observations of influence suffer from the aforementioned disadvantages of diffusion models, which is why we compare to models that utilize diffusion cascades and possibly the network. In terms of other learning methods that we could compare with, CONTINEST (Du et al. 2013) uses an adjacency matrix stored in memory, so it can not scale in these networks, while EMBED-IC (Bourigault, Lamprier, and Gallinari 2016) was proven experimentally inferior to INF2VEC in predicting the evolution of a cascade (Feng et al. 2018). A practical problem with CELFIE is that a typical large network can consist of more then $N = 1,000,000$ nodes and $I = 100,000$ influencers, hence computing ILM might be infeasible. To overcome this, we can filter out nodes that are not candidate seeds based on a fast criteria. In our case, $I = 9,755$ in *Weibo* (nodes with more then 1 cascade), and we define as the final candidate influencers the top 1,000 nodes of them based on the size of the node's cascade in the train set. The experiments were run on a machine with Intel i7, 16Gb RAM and an Nvidia GeForce GTX 1050 GPU.The code can be found online [1].

Results

Figures 4 and 5 show the quality of the seed set in different sizes for each method, for the *Weibo* and *MAG-CS* datasets respectively. One can see that the proposed CELFIE algorithm outperforms the baselines by a significant margin using both INF2VEC and L-INF2VEC embeddings (L-CELFIE). The increased DNI stems from the combination of the learnt representations and the algorithm that derives the seed set. The advantage of representation learning compared to simple influence maximization using past diffusion cascades is more obvious from the gap between L-CELFIE and DIFFUGREEDY. In that case, an algorithm that uses past diffusion cascades exhibits a consistently inferior DNI than the model that learns the diffusion probabilities from them. However, influence and susceptibility representations are not enough, as indicated by the accuracy of the HEURISTIC approach, which is also lower to L-CELFIE in both cases. Hence, it is important to take into account the influence spread overlap in an effective way, such that the seed set's accuracy is guaranteed to increase.

The embeddings of L-INF2VEC provide a slightly worse seed set than simple INF2VEC, the difference being more clear in the experiments in *MAG-CS*. However, Figure 6 provides an overlook over the computational time required for

[1]https://github.com/GiorgosPanagopoulos/IMINFECTOR

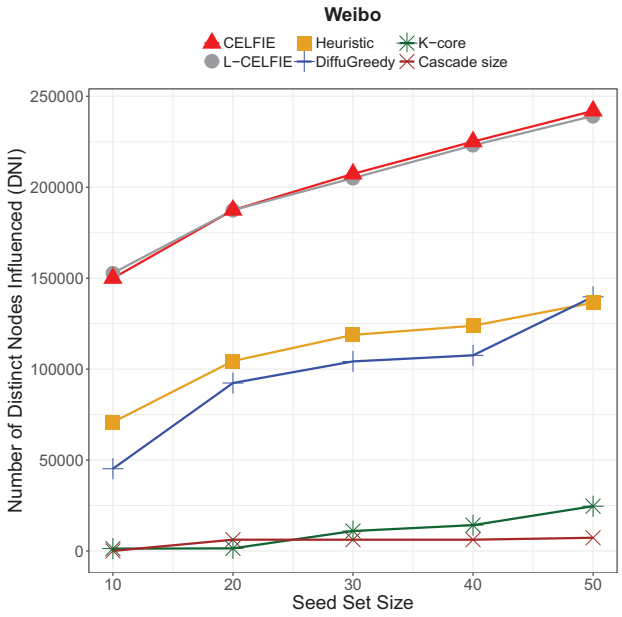

Figure 4: The quality of the seed set in different sizes for each method tested in the *Weibo* dataset. The legend is sorted from left to right and up down, based on descending average DNI computed throughout all seed set sizes. CELFIE and L-CELFIE are significantly better than the baselines. The comparison with DIFFUGREEDY underlines the importance of using influence and suseptibility embeddings compared to simple influence maximization based on diffusion cascades. The comparison with the HEURISTIC indicates that simply relying on the learnt representations is not enough and an influence maximization algorithm such as CELFIE is required to increase accuracy.

the two algorithms; we can observe that L-CELFIE is much faster than CELFIE in both datasets. Of course, the main difference lies in the preprocessing step, where INF2VEC creates a propagation network from each cascade and derives node-context pairs from each node in it, while L-INF2VEC simply samples from the cascade based on the copying times. The training of the representation learning model is also faster due to the reduced number of input samples from the previous step. Overall, the computational burden of the propagation network is prevalent in *Weibo* for both, the Credit Distributon and CELFIE. One can see how the HEURISTIC is slower than CELFIE in *MAG-CS* but becomes faster in *Weibo*, due to the huge number of edges and the combination of number and average size of cascades that renders the context creation mechanism (preprocessing step of CELFIE) very slow. This is also obvious in the only baseline that performs closely to L-CELFIE, the CREDIT DISTRIBUTION model. This model assigns influence credits based on time precedence in the cascades and the follow edges of the network, which has an analogous complexity of the propagation network. Hence, although it scales easily in *MAG-CS*, in which case it performs worse than L-CELFIE, it is not able to scale in *Weibo* (i.e., takes more than 5 days)

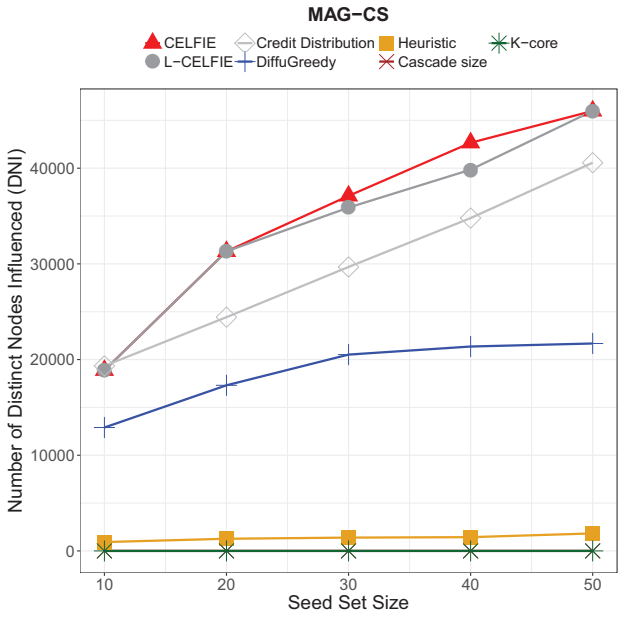

Figure 5: The quality of the seed set in different sizes for each method tested in *MAG-CS* dataset. The legend is sorted from left to right and, based on descending average DNI computed throughout all seed set sizes. The difference between INF2VEC and L-INF2VEC embeddings becomes a little more clear, as the seed set from L-CELFIE is slightly inferior to that of CELFIE. The comparison with another strong influence maximization algorithm that uses the diffusion cascades and the network (CREDIT DISTRIBUTION) indicates the effectiveness of our representation learning approach.

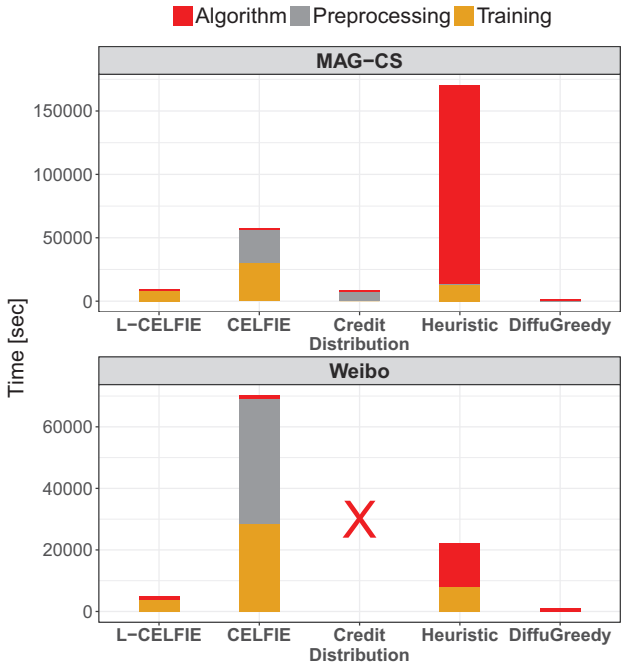

Figure 6: The computational time required by all algorithms for both datasets, broken down to the different steps. One can see the immense difference that the number of edges and average cascade size of *Weibo* causes to the computational time of CELFIE and CREDIT DISTRIBUTION. The difference between L-CELFIE and CELFIE's time is tripled between *Weibo* and *MAG-CS*, while the CREDIT DISTRIBUTION fails to run within the scope of five days. This is because of the propagation network creation that requires looping through all nodes in the cascade and retrieving follow edges from the network. The metric baselines were not included as they are significantly faster but exhibit a very bad DNI.

due to the combination of cascade number and size as well as the number of edges. Of course. there is also a practical advantage in relying just on the the diffusion cascades to derive the embeddings, because the availability of the underlying network is not guaranteed. L-CELFIE uses solely the diffusion cascades and exhibits an adequate or better performance than methods that utilize both, the network and the cascades, such as CELFIE and CREDIT DISTRIBUTION. To sum up, L-CELFIE's speed can be attributed on the one hand to the context-creation mechanism and on the other to the use of diffusion probabilities and the updated definition of marginal gain, which circumvents the need to run simulations of diffusion models, a constant burden to other influence maximization algorithms. In addition, using the CELF approach instead of running the pure greedy algorithm provides a substantial acceleration.

The most important parameter in our model is the size of the train set. When applied in the real world, the relationship between the size of the input and the time range of the resulting seed set is very crucial and indicative of a model's robustness. To evaluate it, we performed a sensitivity analysis by training in different percentages of the cascades and evaluating with a cascade set from the future. Note here that in machine learning, training with different train set sizes means the test set is adjusted as well. In this case however, the DNI would become unavoidably lower as you increase the train set, because there will be fewer test cascades. Moreover, due to the sequential nature of the prediction, we want to refrain from any type of overlapping between the train set and the test set. Thus, we chose to use different train set sizes (60, 70, 80, 90) but test them in the same test set of the last 10%, because in every other case, the train set would include some part of the test set. We plot the results for different seed set sizes in Figure 7. One can see that, the seed set's accuracy does not change significantly throughout different train sets, which indicates the robustness of L-CELFIE. The only exception is the 70% seed set in *MAG-CS*, that has an increased accuracy relative to the rest. The intuition behind this is that seeds that are more successful in the test set, were more active in the first 70% of train set. Afterwards, they were surpassed by other nodes that are not as successful in the test set, but their difference is still not significant. The robustnesss of the model mostly stems from the small size

of the seed set we predict, meaning that the top 50 nodes will unavoidably be very influential in a pool of millions of nodes.

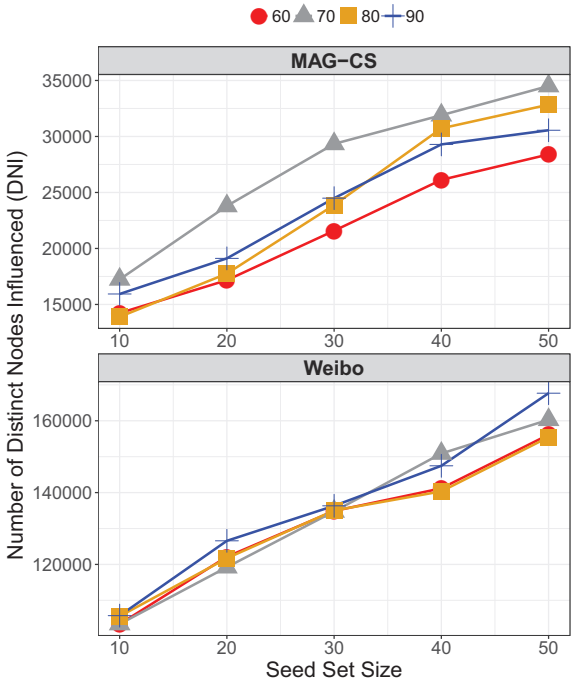

Figure 7: The changes of the accuracy relative to the size of the train set for L-CELFIE are almost the same throughout different sizes of input for both datasets.

Conclusions

This paper proposed CELFIE, an algorithm to perform influence maximization using representations learned from diffusion cascades, such as a tweet and its retweets or a post and its reshares that are spread over a friendship or follower network. The algorithm outperformed several baseline methods in retrieving a set of users that can influence the largest part of the network, based on a data driven evaluation in a large scale dataset. The overall contributions can be summarized as follows:

- Analysis of influencer activity that provided two new insights: Successful influencers mostly start diffusion cascades rather then participate in them, and their cascades tend to be faster then those of a non-influencer.
- Capitalized on the aforementioned findings to modify INF2VEC into L-INF2VEC, a faster model that learns diffusion probabilities for influence maximization by focusing the node-context creation on the initiator of the cascade and the copying times.
- Use of the diffusion probabilities to redefine the computation of the influence spread by substituting diffusion models' simulations on the original network to simple neighborhood sampling on a fully connected bipartite network with the learned edge probabilities.
- Developed a version of CELF that utilizes this setting to optimize the influence spread and retrieve the seed set.

This is the first time, to the best of our knowledge, that representation learning is used towards solving influence maximization, and it proved promising in terms of both accuracy in the seed set and speed. However, our method does not take into account conditional probabilities of influence. In our case, we work with direct diffusion probabilities that implicitly include the transitive influence, but in general it could be argued that a more accurate approach would model the influence probability conditioned on who is already influenced. Thus, models that learn embeddings taking into account the sequence of the cascades (Du et al. 2016; Islam et al. 2018; Wang et al. 2017) could be used to create a more accurate diffusion model to compute the influence spread in the iterations of greedy, and we leave this as future work.

References

Aral, S., and Dhillon, P. S. 2018. Social influence maximization under empirical influence models. *Nature Human Behaviour* 2(6):375.

Aral, S., and Walker, D. 2012. Identifying influential and susceptible members of social networks. *Science* 337(6092):337–341.

Bourigault, S.; Lamprier, S.; and Gallinari, P. 2016. Representation learning for information diffusion through social networks: an embedded cascade model. In *9th ACM International Conference on Web Search and Data Mining*, 573–582. ACM.

Budak, C.; Agrawal, D.; and El Abbadi, A. 2011. Limiting the spread of misinformation in social networks. In *20th international conference on World wide web*, 665–674. ACM.

Chen, W.; Wang, C.; and Wang, Y. 2010a. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1029–1038. ACM.

Chen, W.; Wang, C.; and Wang, Y. 2010b. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1029–1038. ACM.

Cohen, E.; Delling, D.; Pajor, T.; and Werneck, R. F. 2014. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 629–638. ACM.

Domingos, P., and Richardson, M. 2001. Mining the network value of customers. In *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 57–66. ACM.

Du, N.; Song, L.; Rodriguez, M. G.; and Zha, H. 2013. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*, 3147–3155.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1555–1564. ACM.

Feng, S.; Cong, G.; Khan, A.; Li, X.; Liu, Y.; and Chee, Y. M. 2018. Inf2vec: Latent representation model for social influence embedding. In *2018 IEEE 34th International Conference on Data Engineering*.

Garimella, K.; Gionis, A.; Parotsidis, N.; and Tatti, N. 2017. Balancing information exposure in social networks. In *Advances in Neural Information Processing Systems*, 4663–4671.

Gomez-Rodriguez, M.; Balduzzi, D.; and Schölkopf, B. 2011. Uncovering the temporal dynamics of diffusion networks. In *28th International Conference on Machine Learning*, 561–568.

Goyal, A.; Bonchi, F.; and Lakshmanan, L. V. 2010. Learning influence probabilities in social networks. In *Proceedings of the third ACM International Conference on Web Search and Data Mining*, 241–250. ACM.

Goyal, A.; Bonchi, F.; and Lakshmanan, L. V. 2011. A data-based approach to social influence maximization. *In Very Large DataBases Endowment* 5(1):73–84.

Goyal, A.; Lu, W.; and Lakshmanan, L. V. 2011. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 211–220. IEEE.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 855–864. ACM.

Gutmann, M., and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *13th International Conference on Artificial Intelligence and Statistics*, 297–304.

Islam, M. R.; Muthiah, S.; Adhikari, B.; Prakash, B. A.; and Ramakrishnan, N. 2018. Deepdiffuse: Predicting the'who'and'when'in cascades. In *IEEE International Conference on Data Mining*, 1055–1060. IEEE.

Kalimeris, D.; Singer, Y.; Subbian, K.; and Weinsberg, U. 2018. Learning diffusion using hyperparameters. In *International Conference on Machine Learning*, 2425–2433.

Karsai, M.; Kivelä, M.; Pan, R. K.; Kaski, K.; Kertész, J.; Barabási, A.-L.; and Saramäki, J. 2011. Small but slow world: How network topology and burstiness slow down spreading. *Physical Review E* 83(2):025102.

Kempe, D.; Kleinberg, J.; and Tardos, É. 2003. Maximizing the spread of influence through a social network. In *9th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 137–146. ACM.

Lerman, K.; Ghosh, R.; and Surachawala, T. 2012. Social contagion: An empirical study of information spread on digg and twitter follower graphs. *arXiv preprint arXiv:1202.3162*.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N. 2007. Cost-effective outbreak detection in networks. In *13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 420–429. ACM.

Malliaros, F. D.; Giatsidis, C.; Papadopoulos, A. N.; and Vazirgiannis, M. 2020. The core decomposition of networks: theory, algorithms and applications. *Very Large Data Bases Journal* 29(1):61–92.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

Newman, M. E. 2002. Assortative mixing in networks. *Physical review letters* 89(20):208701.

Panagopoulos, G.; Malliaros, F. D.; and Vazirgiannis, M. 2018. Diffugreedy: An influence maximization algorithm based on diffusion cascades. In *International Workshop on Complex Networks and their Applications*, 392–404. Springer.

Pei, S.; Morone, F.; and Makse, H. A. 2018. Theories for influencer identification in complex networks. In *Complex Spreading Phenomena in Social Systems*. Springer. 125–148.

Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; and Tang, J. 2018. Deepinf: Modeling influence locality in large social networks. In *24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sinha, A.; Shen, Z.; Song, Y.; Ma, H.; Eide, D.; Hsu, B.-j. P.; and Wang, K. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, 243–246. ACM.

Tang, Y.; Xiao, X.; and Shi, Y. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 75–86. ACM.

Wang, Y.; Shen, H.; Liu, S.; Gao, J.; and Cheng, X. 2017. Cascade dynamics modeling with attention-based recurrent neural network. In *26th International Joint Conference on Artificial Intelligence*, 2985–2991. AAAI Press.

Zhang, J.; Liu, B.; Tang, J.; Chen, T.; and Li, J. 2013. Social influence locality for modeling retweeting behaviors. In *IJCAI*, volume 13, 2761–2767.