HAL
open science

Multi-task Learning for Influence Estimation and Maximization

George Panagopoulos, Fragkiskos D. Malliaros, Michalis Vazirgiannis

To cite this version:

George Panagopoulos, Fragkiskos D. Malliaros, Michalis Vazirgiannis. Multi-task Learning for Influence Estimation and Maximization. IEEE Transactions on Knowledge and Data Engineering, 2020, 10.1109/TKDE.2020.3040028 . hal-03088930

HAL Id: hal-03088930

https://hal.science/hal-03088930

Submitted on 27 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multi-task Learning for Influence Estimation and Maximization

George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis

Abstract—We address the problem of influence maximization when the social network is accompanied by diffusion cascades. In the literature, such information is used to compute influence probabilities, which is utilized by stochastic diffusion models in influence maximization. Motivated by the recent criticism on diffusion models and the galloping advancements in influence learning, we propose IMINFECTOR (Influence Maximization with INFluencer vECTORs), a method that uses representations learned from diffusion cascades to perform model-independent influence maximization. The first part of our methodology is a multi-task neural network that learns embeddings of nodes that initiate cascades (influencer vectors) and embeddings of nodes that participate in them (susceptible vectors). The norm of an influencer vector captures a node's aptitude to initiate lengthy cascades and is used to reduce the number of candidate seeds. The combination of influencer and susceptible vectors form the diffusion probabilities between nodes. These are used to reformulate the computation of the influence spread and propose a greedy solution to influence maximization that retains the theoretical guarantees. We apply our method in three sizable datasets and evaluate it using cascades from future time steps. IMINFECTOR's scalability and accuracy outperform various competitive algorithms and metrics from the diverse landscape of influence maximization.

Index Terms—influence maximization, influence learning, multi-task learning

1 Introduction

Social networks have come to play a substantial role in numerous economical and political events, exhibiting their effect on shaping public opinion. The core component of their potential in micro level is how a user influences another user online, which is depicted in the users' connections and activity. 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. In viral marketing, influence is used to simulate how information flows through the network and find the optimal set of nodes to start a campaign from, the well-known influence maximization (IM) problem. In IM, the users are represented as nodes and the edges depict a relationship, like following, friendship, coauthorship, etc. A stochastic diffusion model, such as the independent cascade [1], governs how an epidemic traverses the users based on their connections. It is used to compute the number of users activated during a diffusion simulation, which is called the *influence spread*. The aim is to find the optimal set of k users that would maximize the influence spread of a diffusion cascade starting from them [1].

One of the main problems in the IM literature is that diffusion models utilize random or uniform influence parameters. Experiments have shown that such approach produces a less realistic influence spread compared to models with empirical parameters [2]. Even when the parameters are learned in an empirical manner from historical logs of diffusion cascades, the independent cascade (IC) model is utilized [3], [4], [5], which is problematic for two reasons. Apart from severe overfitting due to the massive number of parameters, this approach assumes influence independence throughout related nodes or edges of the same node. This assumption overlooks the network's assortativity, meaning that an influential node is more prone to effect a susceptible node then a less influential node, even when the edge of the latter to the susceptible is stronger than the edge of the former. This effect comes in contrast with the actual mechanics of influence [6]. In addition, diffusion models themselves suffer from oversimplifying assumptions overlooking several characteristics of real cascades [7] which can provide inaccurate estimations compared to actual cascades [8] while they are highly sensitive to their parameters [9].

To address these issues, we propose a method that learns influencer and susceptible embeddings from cascades, and uses them to perform IM without the use of a diffusion model. The first part of our method is INFECTOR (INfluencer Vectors), a multi-task neural network that learns influencer vectors for nodes that initiate cascades and susceptible vectors for those that participate in them. Our focus on the cascades' initiators is justified from empirical evidence on the predominantly initiating tendency of influencers and it is considerably faster compared to previous influence learning (IL) models [10]. INFECTOR additionally embeds the aptitude of an influencer to create sizable cascades in the norm of her embedding. From this, we derive an estimate of influencers' spread and use it to diminish the number of the

- *G. Panagopoulos is with the Computer Science Laboratory (LIX), École Polytechnique, Institute Polytechnique de Paris, 91120 Palaiseau, France*
  E-mail: george.panagopoulos@polytechnique.edu
- *F. D. Malliaros is with Paris-Saclay University, CentraleSupélec, Inria, Centre for Visual Computing (CVN), 91190 Gif-Sur-Yvette, France*
  E-mail: fragkiskos.malliaros@centralesupelec.fr
- *M. Vazirgiannis is with the Computer Science Laboratory (LIX), École Polytechnique, Institute Polytechnique de Paris, 91120 Palaiseau, France and with the Department of Informatics, Athens University of Economics and Business, 10434 Athens, Greece*
  E-mail: mvazirg@lix.polytechnique.fr

Manuscript received XXX; revised XXX.

candidate seeds for IM. Following suit from recent model-independent algorithms [11], [12], we overlook the diffusion model and connect each candidate seed (influencer) and every susceptible node with a diffusion probability using the dot product of their respective embeddings, forming a bipartite network. Apart from removing the time-consuming simulations, diffusion probabilities have the advantage of capturing higher-order correlations that diffusion models fail to due to their Markovian nature. Finally, we propose IMINFECTOR, a scalable greedy algorithm that uses a submodular influence spread to compute a seed set, retaining the theoretical guarantee of $1 - 1/e$.

To evaluate the performance of the algorithm, the quality of the produced seed set is determined by a set of unseen cascades from future time steps, similar to a train and test split in machine learning. We deem this evaluation strategy more reliable than traditional evaluations based on simulations because it relies on actual traces of influence. IMINFECTOR outperforms previous IM methods with learned edge weights, either in quality or speed in three sizable networks with cascades. This is the first time, to the best of our knowledge, that representation learning is used for IM with promising results.

Our contributions can be summarized as follows:

- INFECTOR: a multi-task learning neural network that captures simultaneously the influence between nodes and the aptitude of a node to create massive cascades.
- IMINFECTOR: a model-independent algorithm that uses the combinations and the norms of the learnt representations to produce a seed set for IM.
- A novel, thorough experimental framework with new large scale datasets, seed set size adapted to the network scale and evaluation based on ground truth cascades instead of simulations.

Reproducibility: The implementation and links to the datasets can be found online on github[1].

2 Related Work

The basic solution in the problem of IM is a greedy algorithm that builds a seed set by choosing the node that provides the best marginal gain at each iteration, i.e., the maximum increase of the set's influence spread [1]. The algorithm estimates the influence spread using the live-edge model, a Monte Carlo sampling of edges that estimates the result of simulating a diffusion model. The algorithm is guaranteed to reach a near optimal solution because of the submodularity if the marginal gain. Having said that, the 1,000 samplings needed for the sufficient estimation of the influence spread for each candidate seed in each iteration, renders the approach infeasible for real world networks. Several methods have been developed that achieve notable acceleration and retain the theoretical guarantees using sketches [13] or reverse reachable sets [14], [15]. Moreover numerous heuristics have been proposed that, though lacking guarantees, exhibit remarkable success in practice [16], [17].As the problem of IM became more popular, several methods were developed to derive more realistic versions by adding temporal [18], topic [19] and location constraints [20]. From the perspective of network immunization, the required seed set can be retrieved by finding the must pivotal nodes using optimal percolation [21]. In addition, if the outbreak has already started, a dynamic curing policy based on the graph's cutwidth can be used to control the epidemic [22]. Such methods are developed for epidemic containment and have not been utilized in the context of IM yet, hence are omitted in this paper.

The main problem of the aforementioned IM approaches is that the established influence spread may lead to seed sets of poor quality. This can be caused either by the assignment of simplistic influence weights or by the diffusion model's innate assumptions. To address the issue with the random influence weights, some novel influence-maximization approaches assume multiple rounds of influence maximization can take place over the graph, hence multiple simulations can be used to compute the influence probabilities while balancing between the number of nodes influenced in each round and learning influence for non examined parts of the network. Since this is an inherently exploration-exploitation problem these models are based on multi-armed bandits to use the feedback and update of the parameters [23], [24], [25]. Though useful, these algorithms are built based on the diffusion models as well, hence they share their aforementioned deficiencies. Recently more model-independent online learning [11] approaches that utilize regret functions without the use of diffusion models have been proposed [12], but they suffer from scalability issues and would not be able to scale in real-world social networks, such as the ones we examine in this work.

There have been few attempts to address influence maximization with learned influence parameters from real past cascades, which serve as benchmark comparisons for our proposed approach [26], [27]. These models suffer from overfitting due the number of parameters which is proportional to the number of edges. To reduce the number of parameters, a more practical approach is to express the influence probabilities as a combination of the nodes' influence and susceptibility embeddings and learn them from cascades [4]. This method however relies on IC, and consequently suffers from the aforementioned influence independence assumption. More recent IL methods are devoid of diffusion models and learn the embeddings based on co-occurrence in cascades [10], similarly to node representation learning. This type of IL has not been used yet for IM. Although more accurate in diffusion prediction, the input of this model consists of node-context pairs derived from the propagation network of the cascade, a realization of the underlying network (e.g., follow edges) based on time-precedence in the observed cascade (e.g., retweets). This requires looping through each node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network, which has a complexity of $\mathcal{O}(c\bar{n}(\bar{n}-1)/2)$, where c is the number of cascades and $\bar{n}$ is the average cascade size. This search is too time-consuming as we analyze more on the experimental section.

It should be noted that apart from pure IL, multiple methods have been developed to predict several aspects of a cascade, such as TOPO-LSTM and HIDAN to predict

1. https://github.com/geopanag/IMINFECTOR

the next infected node [28], [29], RMTPP, DEEPDIFFUSE and CYANRNN for next node and time of infection [30], [31], [32], DEEPCAS to predict cascade size [33], and FOREST to predict next node and size together [34]. These methods have advanced end-to-end neural architectures focused each on their respective tasks. However, their hidden representations can not be utilized in the context of IM, because they are derived by a sequence of nodes, hence we can not form individual influence relationships between two distinct nodes. In other words, an IM algorithm requires a node-to-node influence relationship, which is not clearly provided by the aforementioned references. Moreover, methods that do learn node-to-node influence but capitalize on other information such as sentiment of the context [35] can not be utilized as well due to our datasets missing content, but it is a promising future approach.

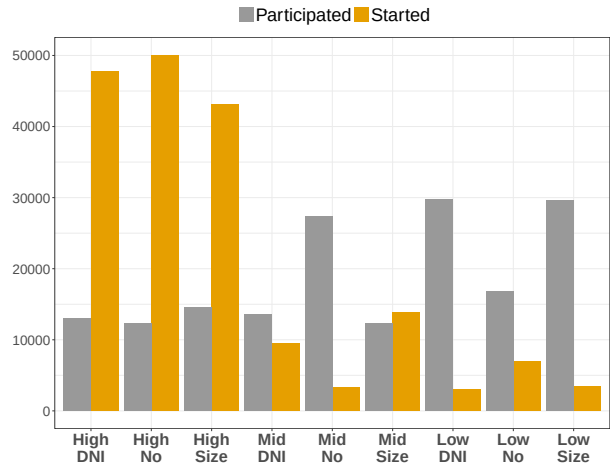

Figure 1: Influencer's initiating versus participating in diffusion cascades. **DNI** stands for distinct nodes influenced, **size** stands for cascade size, and **no** stands for number of cascades started.

3 LEARNING INFLUENCER VECTORS

3.1 Node-Context Extraction

Our goal in the first part of our methodology is to create a model that learns representations suitable for scalable model-independent IM. Due to the lack of a diffusion model, the network's structure becomes secondary and online methods tend to rely only on the activity of individual nodes [11], [12]. We follow suit and propose an IL method that focuses mainly on influencers.

We start from the context creation process that produces the input to the network. In previous node-to-node IL models, initiating a cascade is considered equally important with participating in a cascade created by someone else [10], thus the context of a node is derived by the nodes occurring after it in a cascade. This process requires the creation of the propagation network, meaning going through every node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network. This 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. Given that the average size of a cascade can surpass 60 nodes, it is a very time consuming for a scalable IM algorithm. To overcome this we focus on the nature of IM, meaning the final chosen seed users will have to exert influence over other nodes, thus we may accelerate the process by capitalizing on the differences between influencers and simple users. Intuitively, we expect the influencers to exhibit different characteristics in sharing content than the simple/susceptible nodes [36]. We argue that an influencer's strength lies on the cascades she initiates, and evaluate this hypothesis through an exploratory analysis. We utilize the cascades of *Sina Weibo* dataset, a large scale social network accompanied by retweet cascades to validate our hypothesis. The dataset is split in train and test cascades based on their time of occurrence and each cascade represents a tweet and its set of retweets. We keep the 18,652 diffusion cascades from the last month of recording as a test set and the 97,034 from the previous 11 months as a train set. We rank all users that initiated a cascade 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) [3], [7], [37], which is the set of nodes that participated in these test cascades. We bin the users into three categories based on their success in each metric, and for each category we compute the total cascades the users start in the training set opposed to those they participate in.

Symbol	Meaning	Type
N	number of nodes	scalar
t_u	time of node' u post	scalar
C_u	cascade initiated by u	set
$\mathbf{x}, \mathbf{y}_t$	one-hot embedding of a node	vector
y_c	cascade length	scalar
X^t	$\mathbf{x}, \mathbf{y}$ pairs extracted from cascade	list
$\mathbf{O}$	origin embeddings of INFECTOR	matrix
$\mathbf{T}$	target embeddings of INFECTOR	matrix
C	constant embeddings of INFECTOR	vector
$p_{u,v}$	probability of u influencing v	scalar
$\mathbf{z}$	hidden layer output	vector
f_t	softmax	function
f_c	sigmoid	function
$\boldsymbol{\varphi}_t$	output for node influenced	vector
φ_c	output for cascade size	scalar
$\boldsymbol{\Phi}$	jacobian of INFECTOR's output	matrix
L	loss of INFECTOR	vector
D	INFECTOR classification probabilities	matrix
λ_u	number of nodes to be influenced by u	scalar
$\hat{D}_s$	D sorted for the row of node s	matrix
$\sigma'(s)$	influence spread of node set s	scalar
S	Set of seed nodes	set

Table 1: Table of symbols.

Here, we examine the contrast between the behavior of the influencers and normal users, meaning how more probable is for an influencer to start cascades compared to normal users. As we see in Figure 1, users that belong to the top category of the test set are much more prone to create cascades compared to the ones who belong to the mid and low categories. Since our end goal is to find influencers for our algorithm, this observation allows us to focus solely on the initiators of the cascades, rather than every node in the cascade. Moreover, we observe that influencers in Weibo are more prone to create cascades opposed to participating in them. This means that by overlooking their appearances

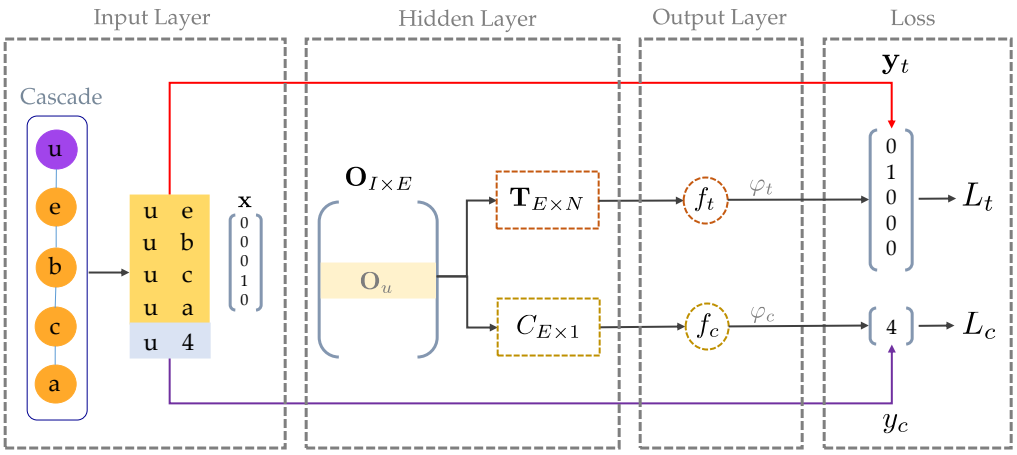

Figure 2: Schematic representation of the INFECTOR model. Given the cascade, a sequence of pairs is extracted, each pair consisting of the initiator node $\mathbf{x}$ (one-hot embedding) which is the *input*, and one of the "infected" nodes $\mathbf{y}_t$ i.e. e,b etc. which is the *output*. The last pair is the initiator and the cascade size y_c i.e. 4 as output. After the initator's embedding lookup through the origin embeddings $\mathbf{O}$, the vector passes through $\mathbf{T}$ and f_t if the output is another node or through $\mathbf{C}$ and f_c if the output is scalar. The loss functions are log-loss L_t for node output or mean squared error L_c for scalar output.

inside the cascades of others, we do not lose too much information regarding their influence relationships, as most of them start the cascades.

Thus, for our purpose, the context is extracted exclusively for the initiator of the cascade and will be comprised of all nodes that participate in it. Moreover, we will take into account the copying time between the initiator and the reposter, based on empirical observation on the effect of time in influence [27].To be specific, all cascade initiators are called influencers, an influencer u's context will be created by sampling over all nodes v in a given cascade c that u started, with probability inversely proportional to their copying time. In this way, the faster v's retweet is, the more probable it will appear in the context of u, following this formula $P(v|\mathcal{C}_u) \sim \frac{(t_u - t_v)^{-1}}{\sum_{v' \in \mathcal{C}_u} (t_u - t'_v)^{-1}}$. It is a general principle to utilize temporal dynamics in information propagation when the network structure is not used [3], [38], [39] and is based on empirical observation on the effect of influence in time [27]. In our case we do an oversampling of 120% to emphasize the importance of fast copying. This node-context creation has a complexity of $\mathcal{O}(cn)$, which is linear to the cascade's size and does not require searching in the underlying network. Even more importantly, this type of context sets up the model to compute diffusion probabilities, i.e. influence between nodes with more than one hop distance in the network.

3.2 The INFECTOR Model

The diffusion probabilities (DPs) are the basis for our model-independent approach, and exhibit important practical advantages over influence probabilities, as we analyze further below. We use a multi-task neural network [40] to learn simultaneously the aptitude of an influencer to create long cascades as well as the diffusion probabilities between her and the reposters. We chose to extend the typical IL architecture in a multi-task learning setting because (i) the problem could naturally be broken in two tasks, and (ii) theoretical and applied literature suggests that training linked tasks together improves overall learning [41], [42]. In our case, given an input node u, the first task is to classify the nodes it will influence and the second to predict the size of the cascade it will create. An overview of our proposed INFECTOR model can be seen in Figure 2. There are two types of inputs. The first is the training set comprised of the node-context pairs. As defined in the previous section, given a cascade t with length m, we get a set $\mathbf{X}^t = \{(\mathbf{x}^1, \mathbf{y}_t^1), (\mathbf{x}^1, \mathbf{y}_t^2), ..., (\mathbf{x}^m, \mathbf{y}_t^m)\}$, where $\mathbf{x} \in \mathbb{R}^I$ and $\mathbf{y}_t \in \mathbb{R}^N$ are one hot encoded nodes, with I the number of influencers in the train set and N the number of nodes in the network. The second is a similar set X^c, where instead of a vector e.g. $\mathbf{y}_t^1$, there is a number $\mathbf{y}_c^1$ denoting the length of that cascade, initiated by the $\mathbf{x}^1$. To perform joint learning of both tasks we mix the inputs following the natural order of the data; given a cascade, we first input the influencers-context pairs extracted from it and then the influencers-cascade length pair, as shown in Figure 2. $\mathbf{O} \in \mathbb{R}^{I \times E}$, with E being the embeddings size, represents the source embeddings, $\mathbf{O}_u \in \mathbb{R}^{1 \times E}$ the embeddings of cascade initiator u, $\mathbf{T} \in \mathbb{R}^{E \times N}$ the target embeddings and $C \in \mathbb{R}^{E \times 1}$ is a constant vector initialized to 1. Note that $\mathbf{O}_u$ is retrieved by the multiplication of the one-hot vector of u with the embedding matrix $\mathbf{O}$. The first output of the model represents the diffusion probability $p_{u,v}$ of the source node u for a node v in the network. It is created through a softmax function and its loss function is the cross-entropy. The second output aims to regress the cascade length, which has undergone min-max normalization relative to the rest of the cascades in this set, and hence a sigmoid function is used to bound the output at $(0, 1)$. The respective equations can be seen in Table 2. Here y_t is a one-hot representation of the target node and y_c is the normalized cascade length. We employ a non-linearity instead of a simple regression because without it, the updates induced to the hidden layer from the second output would heavily overshadow the ones from the first. Furthermore, we empirically observed that the update of a simple regression would cause the gradient to explode eventually. Moreover, variable C is a constant

vector of ones that is untrainable, in order for the update of the regression loss to change only the hidden layer. This change was motivated by experimental evidence.

	Classify Influenced Node	*Regress Cascade Size*
Hidden	$\mathbf{z}_{t,u} = \mathbf{O}_u \mathbf{T} + b_t$	$\mathbf{z}_{c,u} = \mathbf{O}_u C + b_c$
Output	$\boldsymbol{\varphi}_t = \frac{e^{(\mathbf{z}_{t,u})}}{\sum_{u' \in G} e^{\mathbf{z}_{t,u'}}}$	$\boldsymbol{\varphi}_c = \frac{1}{1+e^{-(\mathbf{z}_{c,u})}}$
Loss	$L_t = \mathbf{y}_t \log(\boldsymbol{\varphi}_t)$	$L_c = (y_c - \varphi_c)^2$

Table 2: The layers of INFECTOR.

The training happens in an alternating manner, meaning when one output is activated the other is idle, thus only one of them can change the shared hidden layer at a training step. For example, given a node u that starts a cascade of length 4 like in Figure 2, $\mathbf{O}_u$ will be updated 4 times based on the error of $\frac{\partial L_t}{\partial \mathbf{O}}$, and one based on $\frac{\partial L_c}{\partial \mathbf{O}}$, using the same learning rate e for the training step τ:

$$\mathbf{O}_{u,\tau+1} = \mathbf{O}_{u,\tau} - e\frac{\partial L}{\partial \mathbf{O}} \tag{1}$$

The derivatives from the chain rule are defined as:

$$\frac{\partial L}{\partial \mathbf{O}} = \frac{\partial L}{\partial \varphi}\frac{\partial \varphi}{\partial z}\frac{\partial z}{\partial \mathbf{O}}. \tag{2}$$

The derivatives in Eq. (2) differ between regression and classification. For the loss function of classification L_c, we have that

$$\frac{\partial \mathbf{z}_c}{\partial \mathbf{O}} = \mathbf{T}^\top \tag{3}$$

$$\frac{\partial \boldsymbol{\varphi}_t}{\partial \mathbf{z}_t} = \begin{cases} \boldsymbol{\varphi}_t^u(1-\boldsymbol{\varphi}_t^v), & u = v \\ -\boldsymbol{\varphi}_t^u \boldsymbol{\varphi}_t^v, & u \neq v \end{cases} \tag{4}$$

where u is the input node and v is each of all the other nodes, represented in different dimensions of the vectors $\boldsymbol{\varphi}_t$. If we create a matrix consisting of N replicates of $\boldsymbol{\varphi}_t$, we can express the Jacobian as this dot product:

$$\frac{\partial \boldsymbol{\varphi}_t}{\partial \mathbf{z}_t} = \boldsymbol{\Phi}_t \cdot (I - \boldsymbol{\Phi}_t)^\top, \quad \boldsymbol{\Phi}_t = \begin{bmatrix} \boldsymbol{\varphi}_t \\ \vdots \\ \boldsymbol{\varphi}_t \end{bmatrix} \in \mathbb{R}^{N \times N} \tag{5}$$

$$\frac{\partial L_t}{\partial \boldsymbol{\varphi}_t} = \mathbf{y_t}\left(-\frac{1}{\boldsymbol{\varphi_t}}\right) \tag{6}$$

The derivatives for the regression task are:

$$\frac{\partial \mathbf{z}_c}{\partial \mathbf{O}} = C^\top \tag{7}$$

$$\frac{\partial \varphi_c}{\partial \mathbf{z}_c} = \frac{1}{1+e^{-\mathbf{z}_c}}\left(1 - \frac{1}{1+e^{-\mathbf{z}_c}}\right) = \varphi_c(1-\varphi_c) \tag{8}$$

$$\frac{\partial L_c}{\partial \varphi_c} = 2(y_c - \varphi_c)\frac{(-\varphi_c)}{\partial \varphi_c} = -2(y_c - \varphi_c) \tag{9}$$

From Eqs. (2), (3), (5), (6), (7), (8),(9) we have:

$$\frac{\partial L}{\partial \mathbf{O}} = \begin{cases} -\frac{y_t}{\boldsymbol{\varphi}_t}(\boldsymbol{\Phi}_t \cdot (I - \boldsymbol{\Phi}_t)^\top)\mathbf{T}^\top \\ -2(y_c - \varphi_c)(\varphi_c(1-\varphi_c))\mathbf{C}^\top \end{cases} \tag{10}$$

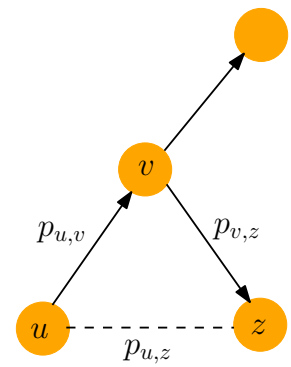

Figure 3: Example of higher order influence showing the difference between influence and diffusion probabilities.

As mentioned above, the aim here is for the embedding of an influencer (hidden layer $\mathbf{O}$) to not only capture who she influences but also her overall aptitude to create lengthy cascades. To be specific, for each node v inside the cascade, O_u and T_v will undergo updates using the upper formula from Eq. (10) to form the diffusion probabilities, depending on $\boldsymbol{\varphi_t}$ and $\boldsymbol{y_t}$. We see that the upper update is formed based on a combination of the gradients, hence each dimension of the update vector might be positive or negative. In contrast, when updating the parameters for the regression task, we multiply a constant vector C with a scalar value. Thus, the update will increase or decrease the overall norm of the embedding analogously to the difference of the predicted and the actual cascade size.

The main difference between INFECTOR and similar node-to-node IL methods is that it computes diffusion probabilities and does not require the underlying social network, in contrast to influence probabilities which are assigned to edges of the network [4], [5], [10]. Intuitively, DP is the probability of the susceptible node appearing in a diffusion started by the influencer, independently of the two nodes' distance in the network. This means that the underlying influence paths from the seed to the infected node is included *implicitly*, which changes drastically the computation of influence spread. For example, in a setting with diffusion models and influence probabilities, 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 $p_{u,z}$.

This approach captures the case when v appears in the cascades of u and z appears in the cascades of v but not in u's. This is a realistic scenario that occurs when v reshares various types of content, in which case content from u might diffuse in different directions than z. Hence, it would be wrong to assume that u's infection would be able to eventually cause z's. Typical IM algorithms fail to capture such higher order correlations because diffusion models act in a Markovian manner and can spread the infection from u to z. Apart from capturing higher order correlations, DPs will allow us to overcome the aforementioned problems induced in IM by the diffusion models as well as surpass the computational bottleneck of the repeated simulations.

Please note that our method's final purpose is influence maximization, which is why we focus so much on the activity of influencers and overlook the rest of the nodes. If we aimed for a purely prediction task, such as cascade size or next infected node prediction, it is not clear that our learning mechanism would be effective.

4 Influence Maximization with Influencer Vectors

4.1 Diffusion With Influencer Vectors

In the second part of the methodology, we aim to perform fast and accurate IM using the representations learnt by INFECTOR and their properties. Initially, we will use the combination of the representations which form the *diffusion probability* of every directed pair of nodes. The DPs derived from the network can define a *matrix*:

$$\mathbf{D} = \begin{bmatrix} f_t(\mathbf{O}_1 T) \\ \vdots \\ f_t(\mathbf{O}_I T) \end{bmatrix} \tag{11}$$

which consists of the nodes that initiate train cascades (influencers) in one dimension and all susceptible nodes in the other. Even though influencers are fewer than the total nodes, $\mathbf{D}$ can still be too memory-demanding for real-world datasets. To overcome this, we can keep the top $P\%$ influencers based on the norm of their influencer embedding $|\mathbf{O}_u|$ to reduce space, depending on the device. Recall that, the embeddings are trained such that their norm captures the influencers' potential to create lengthy cascades, because $\mathbf{O}_u C = \sum_i^E \mathbf{O}_u(i) = |O|$, since C is constant.

Subsequently, $\mathbf{D}$ can be interpreted as a bipartite network where the left side nodes are the candidate seeds for IM, and each of them can influence every node in the right side, where the rest of the network resides. Since all edges are directed from left to right, no paths with length more than 1 exist. This means that the probability of an edge can only define the infection of one node, and it is independent of the infection of the rest—hence, we do not require a diffusion model to estimate the spread.

To be specific regarding our assumptions and the difference with diffusion models, we clarify that typically simulations are performed in order to account for all 2^e possible combinations of edges. This happens because an edge might produce the infection of more than the target node e.g. when it is a bridge between two communities. This can not happen in our setting by the definition of diffusion probability: if a node can influence another node, no matter how far that is in the original network, it will be depicted in their edge weight. That is clear in the first iteration, where the infection of a seed can only cause the infection of its neighbors hence simulations are not required. To keep this property in the subsequent iterations (where the candidate's influenced spread overlaps with previous seeds) we make the assumption that each node will influence for sure a certain number of nodes, based on INFECTOR's representations. Hence these nodes can be removed, as they should not be recalculated in the influence spread of subsequent candidate seeds.

To formulate that influence spread, we will utilize the embedding of candidate seed u's to compute the fraction of all N nodes it is expected to influence based on its $|\mathbf{O}_u|$:

$$\lambda_u = \left\lceil N \frac{||\mathbf{O}_u||_2}{\sum_{u' \in \mathcal{I}} ||\mathbf{O}_{u'}||_2} \right\rceil \tag{12}$$

where $\mathcal{I}$ is the set of candidate seeds. The term resembles the norm of u relative to the rest of the influencers and the size of the network. It is basically computing the amount of the network u will influence. Since we know a seed s can influence a certain number of nodes, we can use the diffusion probabilities to identify the top λ_s nodes it connects to. Moreover, we have to take into account the values of the DPs, to refrain from selecting nodes with big λ_s but overall small probability of influencing nodes. Consequently, the influence spread is defined by the sum of the top λ_s DPs:

$$\sigma'(s) = \sum_j^{\lambda_s} \hat{\mathbf{D}}_{s,j} \tag{13}$$

where $\hat{\mathbf{D}}_s$ is the DPs of seed s sorted in descending order. Once added to the seed set, as mentioned above the seed's influence set is considered infected and is removed from $\mathbf{D}$. This means that a seed's spread will never get bigger in two subsequent rounds, and we can employ the CELF trick [43] to accelerate our solution. It should be noted that one can either model alternatively the influence spread based on the total probability of each node getting influenced, which results in a non-convex function, or approach the problem as weighted bipartite matching. Both these cases are analyzed briefly in the Appendix A.2

To retain the theoretical guarantees of the greedy algorithm $(1 - 1/e)$, we need to prove the monotonicity and submodularity of the influence spread that we optimize in each iteration. To do this, we need to separate the influence spread of the seed set S throughout the different steps of the algorithm. Let $i(s)$ be the step before node s was inserted in S, S^i be seed set at step i, $\mathbf{D^i}$ the DP matrix at step i and u the current candidate seed.

Corollary 1. The influence spread is monotonic increasing.

Proof:

$$\begin{aligned} \sigma'(S^{i(u)} \cup \{u\}) = \sum_{s \in S^{i(u)} \cup \{u\}} \sum_j^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} = \\ \sum_{s \in S^{i(u)}} \sum_j^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} + \sum_j^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} = \\ \sigma'(S^{i(u)}) + \sum_j^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \geq \sigma'(S^{i(u)}) \end{aligned} \tag{14}$$

□

Corollary 2. The influence spread is submodular.

Proof:

$$\begin{aligned} \sigma'(S^{i(u)-1} \cup \{u\}) - \sigma'(S^{i(u)-1}) = \\ \sum_{s \in S^{i(u)-1} \cup \{u\}} \sum_j^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} - \sum_{s \in S^{i(u)-1}} \sum_j^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} = \sum_j^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)-1} \end{aligned} \tag{15}$$

$$\sigma'(S^{i(u)} \cup \{u\}) - \sigma'(S^{i(u)}) = \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \tag{16}$$

$\hat{\mathbf{D}}_{u,j}^{i(u)}, \forall j \in \lambda_u$ are the top DPs of u for nodes that are left uninfected from the seeds in $S^{i(u)}$. Note here that, $\hat{\mathbf{D}}_{u,j}^{i(u)}, \forall j \in \lambda_u$ are the top DPs of u for nodes that are left uninfected from the seeds in $S^{i(u)}$. So, the influence spread of u is reverse analogous to the influence spread of the seed set up to that step:

$$S^{i(u)-1} \subseteq S^{i(u)} \Leftrightarrow \sum_{s \in S^{i(u)-1}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} \leq \sum_{s \in S^{i(u)}} \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}^{i(s)} \Leftrightarrow \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)-1} \geq \sum_{j}^{\lambda_u} \hat{\mathbf{D}}_{u,j}^{i(u)} \tag{17}$$

□

4.2 IMINFECTOR

The algorithm is given at Algorithm 1 and is an adaptation of CELF using the proposed influence spread and updates on $\mathbf{D}$. We keep a queue with the candidate seeds and their attributes (line 2) and the uninfected nodes (line 3). The attributes of a candidate seed are its influence set (line 5) and influence spread (line 6), as defined by eq. (13). We sort based on the influence spread (line 8) and proceed to include in the seed set the candidate seed with the maximum marginal gain, which is Ω after the removal of the infected nodes. Once a seed is chosen, the nodes it influences are marked as infected (line 14). The DPs of the candidate seeds are reordered after the removal of the infected nodes (line 17) and the new marginal gain is computed (line 18).

An alternative to Eq. (12) would be to keep the actual percentage of nodes influenced by each seed, i.e., $||\mathbf{O}_u||_2/N$. This approach, however, suffers from a practical problem. In real social networks, influencers can create cascades with tens or even hundreds of thousands of nodes. In this case, $|\lambda_u|$ can be huge, so when our algorithm is asked to come up with a seed set of e.g., 100 nodes, matrix $\mathbf{D}$ will be emptied in the first few iterations that will include seeds covering the whole network. This would constrain our algorithm to small seed set sizes, in which case simple ranking metrics tend to be of equal strength with IM, as we argue further on the evaluation methodology. Hence, we use a normalization that alleviates moderately the large differences between influence spreads and allows for bigger seed set sizes.

To give a concrete example of how the algorithm works, let's imagine a simple network with 8 nodes, as depicted in Figure 4, with the respective weights from each candidate seed S to each susceptible node N. In the first step, λ of a seed S defines how many of the S's top susceptible nodes should be taken into account in the computation of σ'. In this case, we keep the top 3 for $S3$ and top 2 for $S2$ and $S1$. The top 3 for $S3$ correspond to $\{N1, N3, N4\}$, as indicated by the green edges in the bipartite network, whose sum gives a $\sigma' = 0.9$. For $S2$, we could either take $\{N1, N2\}$ or $\{N1, N3\}$, both giving a $\sigma' = 0.6$, which is lower than $S3$, and the same applies for $S1$. Thus $S3$ is chosen as the seed for step 1 and the influenced nodes are removed.

Algorithm 1 IMINFECTOR

Input: Probability diffusion matrix D, expected influence spread λ, seed set size ℓ
Output: Seed set $\mathcal{S}$

```
    procedure IMINFECTOR(D, λ, ℓ)
2:      set q ← [ ], S ← ∅
        F = 0 : dim(D)[1]
4:      for s = 0; s < dim(D)[0]; s++ do
            R ← argsort(D[s, F])[0 : λ[s]]
6:          Ω ← sum(D[s, R])
            q.append([s, Ω, R, 0])
8:      q ← sort(q, 1)
        while |S| < ℓ do
10:         u ← q[0]
            if u[3] == |S| then
12:             S.add(u[0])
                R ← u[2]
14:             F ← F − R
                q.delete(u)
16:         else
                R ← argsort(D[u[0], F])[0 : λ[u[0]]]
18:             Ω ← sum(D[u[0], R])
                q[0] ← [u[0], Ω, R, |U|]
20:             q ← sort(q, 1)
        return S
```

Seed	$N1$	$N2$	$N3$	$N4$	$N5$	λ	σ'
S1	0.1	0.3	0.2	0.2	0.2	2	0.5
S2	0.4	0.2	0.2	0.1	0.2	2	0.6
S3	0.5	0.1	0.2	0.2	0	3	0.9

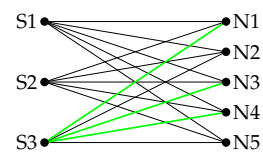

Figure 4: Example of IMINFECTOR: step 1.

Seed	$N2$	$N5$	λ	σ'
S1	0.3	0.2	2	0.5
S2	0.2	0.2	2	0.4

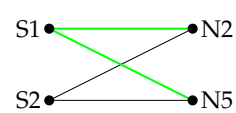

Figure 5: Example of IMINFECTOR: step 2.

In the next step, matrix $\mathbf{D}$ is updated to remove the influenced nodes and σ' is recomputed. In this case, there are only two nodes left for each seed, so their influence spread is computed by the sum of all their edges, since their $\lambda = 2$. An important point here is that while in the first step $S2$ had a stronger σ' than $S1$, $S1$ is stronger now, due to the removal of $N1$, which was strongly susceptible to the influence of $S2$, but infected in the previous round by $S3$.

Thus, the chosen seed in step 2 is $S1$.

Running time complexity. The first step of the algorithm is to compute the influence spread for all candidate seeds I in $\mathbf{D}$, which takes $I \cdot N \log N$, because we sort the DPs of each candidate seed to take the top λ. The sorting of q which has $\mathcal{O}(I \cdot \log I)$ before the algorithm's iterations start, adds constant time to the total complexity. With that in mind, and taking into acount the seed set size ℓ, the complexity is analogous to $\mathcal{O}(\ell \cdot I(I \log I) \cdot (N \log N))$, similar to CELF, but with sorting N in every evaluation of the influence spread. In practice, in every iteration, N is diminished by an average of λ because of the removed influence set, so the final logarithmic term is much smaller. Finally, due to the nature of CELF, much fewer influence spread evaluations than I take place in reality (line 16 in the algorithm), which is why our algorithm is fast in practice.

5 Experimental Evaluation

5.1 Datasets

It is obvious that we can not use networks from the IM literature because we require the existence of ground truth diffusion cascades. Although such open datasets are still quite rare, we managed to assemble three, two social and one bibliographical, to provide an estimate of performance in different sizes and information types. Table 3 gives an overview of the datasets.

	Digg	*MAG*	*Sina Weibo*
Nodes	279,631	1,436,158	1,170,689
Edges	2,251,166	15,928,078	225,877,808
Cascades	3,553	181,020	115,686
Avg Cascade size	847	29	148

Table 3: Summary of the datasets used.

- *Digg*: A directed network of a social media where users follow each other and a vote to a post allows followers to see the post, thus it is treated as a retweet [44].
- *MAG Computer Science*: We follow suit from [45] 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 the employ Microsoft Academic Graph [46] and filter to keep only papers that belong to computer science. We remove cascades with length less than 10.
- *Sina Weibo*: A directed follower network where a cascade is defined by the first tweet and the list of retweets [47]. We remove from the network nodes that do not appear in the cascades i.e. they are practically inactive, in order to make more fair the comparison between structural and diffusion-based methods, since the evaluation relies on unseen diffusions.

5.2 Baseline Methods

Most traditional IM algorithms, such as greedy [1] and CELF [43], do not scale to the networks we have used for evaluation. Thus, we employ only the most scalable approaches:

- K-Cores: The top nodes in terms of their core number, as defined by the undirected k-core decomposition. This metric is extensively used for influencer identification and it is considered the most effective structural metric for this task [48], [49].
- AVG Cascade Size: The top nodes based on the average size of their cascades in the train set. This is a straightforward ranking of the nodes that have proven effective in the past [50].
- SimPath: A heuristic that capitalizes on the locality of influence pathways to reduce the cost of simulations in the influence spread [16]. (The *threshold* is set to 0.01).
- IMM: An algorithm that derives reverse reachable sets to accelerate the computation of influence spread while retaining theoretical guarantees [15]. It is considered state-of-the-art in efficiency and effectiveness. (Parameter ϵ is set to 0.1).
- Credit Distribution: This model uses cascade logs *and* the edges of the network to assign influence credits and derive a seed set [26]. (Parameter λ is set to 0.001).
- CELFIE: The previous version of the proposed algorithm, which relies on influence learning and sampling-based influence spreading [51].
- IMINFECTOR: Our proposed model based solely on cascades, with embedding size equal to 50, trained for 5 epochs with a learning rate of 0.1. The reduction percentage P is set to 10 for *Weibo* and *MAG*, and 40 for *Digg*.

The parameters are the same as in the original papers. Comparing network-based methods such as SimPath and IMM with methods that use both the network and cascades, is not a fair comparison as the latter capitalize on more information. To make this equitable, each network-based method is coupled with two influence learning (IL) methods that provide the IM methods with influence weights on the edges, using the diffusion cascades:

- Data-based (DB): Given that the follow edge $u \rightarrow v$ exists in the network, the edge probability is set to $A_{u \rightarrow v}/A_u$, i.e., the number of times v has copied (e.g., retweeted) u, relative to the total activity (e.g., number of posts) of u [27]. Any edge with zero probability is removed, and the edges are normalized such that the sum of weighted out-degree for each node is 1.
- Inf2vec: A shallow neural network performing IL based on the co-occurence of nodes in diffusion cascades and the underlying network [10]. It was proven more effective than similar IL methods [4], [5] in the tasks of next node prediction and diffusion simulation.

5.3 Evaluation Methodology

We split each dataset into train and test cascades based on their time of occurrence. The methods utilize the train cascades and/or the underlying network to define a seed set. The train cascades amount for the first 80% of the whole set and the rest is left for testing. The evaluation is twofold: computational time and seed set quality, similar to previous literature in influence maximization. We evaluate the quality of the predicted seed set using the number of *Distinct Nodes Influenced* (DNI) [3], [7], [37], which is the combined set of nodes that appear in the test cascades that are initiated from each one of the chosen seeds. To be specific, each predicted seed in the seed set has started some cascades in the test set. We compute the set of all infected nodes (DNI) by simply adding every node appearing in these test cascades, in a unified set. The size of this set indicates a measure of how successful was that seed set on the test set. To understand the choice of DNI we need to take into consideration that each seed can create several test cascades. For example, another metric we could use is the sum of the average cascade length of each seed's test cascades [52]. With this approach though, we fail to counter for the overlap between different seeds' spreads, and hence we can end up with influential seeds whose influence spreads overlaps a lot, resulting in an eventually smaller number of infected nodes. As mentioned above, DNI maintains a set of all distinct nodes participating in cascades started by each of the seeds, hence tackling the potential overlap issue. Overall, the most significant advantage over other standard IM evaluations is that it relies on actual spreading data instead of simulations. Although not devoid of assumptions, it is the most objective measure of a seed set's quality, given the existence of empirical evidence.

Since our datasets differ significantly in terms of size, we have to use different seed set size for each one. For *MAG* which has 205,839 initiators in the train set, we test it on 10,000, *Weibo* with 26,158 is tested on 1,000, and *Digg* with 537 has a seed set size of 50. This modification is crucial to the objective evaluation of the methods because small seed sets favor simpler methods. For example, the top 100 authors based on K-CORES in *MAG* would unavoidably work well because they are immensely successful. However, increasing the seed set size allows the effect of influence overlap to take place, and eventually, simplistic methods fall short. The experiments were run on a machine with Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz, 252GB ram, and an Nvidia Titan V. Any method that required more memory or more than seven days to run, it is deemed unable to scale.

5.4 Results

Figure 6 shows the computational time of the examined methods, separated based on different parts of the model, and Figure 7 shows the estimated quality of each seed set. In terms of quality, we can see that IMINFECTOR surpasses the benchmarks in two of the three datasets. In *Digg*, CREDIT DISTRIBUTION performs better but is almost 10 times slower, because of the average cascade size in the dataset. All methods could scale in *MAG* except for SIMPATH with INF2VEC weights, because in contrast to DB, INF2VEC retains all the edges of the original *MAG* network, for which SIMPATH takes more than one week to run. Moreover, SIMPATH with both types of weights is not able to scale in the *Weibo* dataset, due to the number of edges. CREDIT DISTRIBUTION also fails to scale in *Weibo* due to the network's and cascade size. IMM with INF2VEC can not scale due to the high demand in memory. It can still scale with DB weights (close to 1M nodes), but it performs purely. The only method that scales successfully in Weibo was ranking by the k-core decomposition, which took about 650 seconds. IMM's performance is lower than expected, highlighting the difference between data-driven means of evaluation and the traditional simulations of diffusion models, which have been used in the literature due to the absence of empirical data. IMM optimizes diffusion simulations as part of its solution, so it is reasonable to outperform other methods in this type of evaluation. However, it performance is not equal in a metric based on real influence traces. This can be attributed largely to the aforementioned problems of diffusion models and their miscalculation of the influence spread.

Compared to CELFIE, the previous version of our algorithm, IMINFECTOR has superior performance as the seed set size increases. One of the CELFIE's core disadvantages is that the marginal influence spread, which is computed by sampling repetitively from a node's neighbors and summing the edge's weights, decreases very fast during the seeding rounds. This has as a result the whole network being covered with few rounds, and hence the retrieved seed set size to always be bounded. In IMINFECTOR, each node has an expected influence spread as shown in Eq. (12), which we have defined based on a node's influencer vector, compared to the rest of the initiator's representations. In other words, this formulation imposes the constraint that the network is shared among all candidate seeds, depending on their influencer vector, rendering infeasible to cover the whole graph without putting all initiators in the seed set. Moreover, it allows for a fixed computation of a node's influence spread (e.g., without resampling), which renders it faster the CELFIE which retrieves only 127 seeds for Weibo, 2059 for MAG and 4 for Digg.

In terms of IL methods, DB outperformed INF2VEC because it removes edges with no presence in the cascades, diminishing significantly the size of the networks, some even close to 90%. However, it also had a severe shortcoming most notable in the social networks. Due to the general lack of consistent reposters and the huge amount of posts, the total activity of a node (the denominator in edge weight) was much larger than the number of reposts by a follower (numerator). Hence, the edge weights were too small with insignificant differences, alleviating the computation of the simulated influence spread. This effect is not so strong in the bibliographical network, because authors tend to cite the same popular authors more consistently. A side observation is the heavy computational burden of INF2VEC. It accounts for most of the computational time (blue color), which justifies the context creation mechanism of INFECTOR, while the accuracy validates our hypothesis that influencers' success lies more on the cascades they initiate rather their reposts.

In general, we see that IMINFECTOR provides a fair balance between computational efficiency and accuracy. Most importantly, it exhibits such performance using *only*

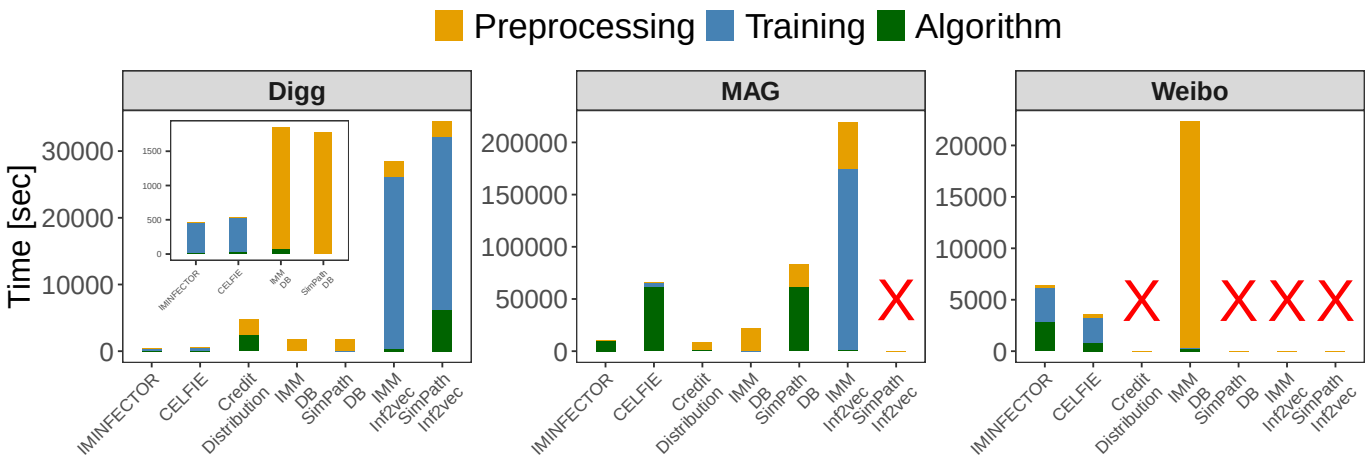

Figure 6: Time comparison between the methods. Methods that could not scale are marked with X.

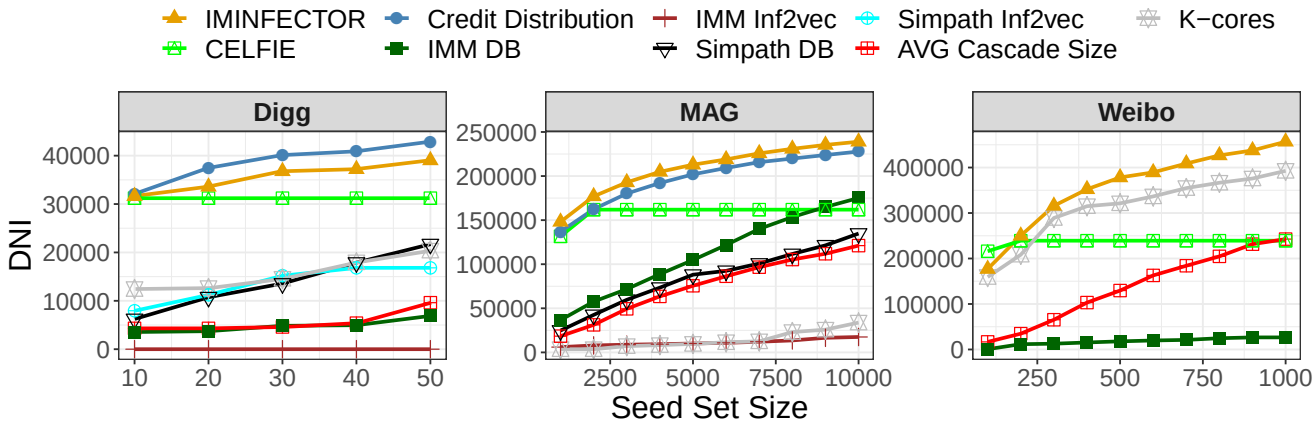

Figure 7: The quality of the seed set derived by each method in different sizes, measured by the seed set's number of Distinct Nodes Influenced (DNI) in the test cascades. Algorithms that failed to scale for the minimum seed set size, are not included.

the cascades, while the rest baselines use both, the network and cascades. Being unaffected by the network size, IMINFECTOR scales with the average cascade size and the number of cascades, which makes it suitable for real world applications where the networks are too big and scalability is of utmost importance.

6 Conclusion

In this paper, we have proposed IMINFECTOR, a model-independent method to perform influence maximization using representations learned from diffusion cascades. The machine learning part learns pairs of representations from diffusion cascades, while the algorithmic part uses the representations' norms and combinations to extract a seed set. The algorithm outperformed several methods based on a data-driven evaluation in three large scale datasets.

It should be noted that our method is not devoid of assumptions. Firstly, we assume that the optimum seed set is comprised solely of nodes that initiate cascades. Given the task of influence maximization, this seems a valid assumption, with the exception of cases where very popular people in the real world enter the social network and get hyperconneccted without sharing content. Apart from this assumption, our method is highly scalable; a system though with too many cascade initiators (close to the number of users), will increase the overall complexity. A simple solution to this is to remove initiators with minuscule activity, which, given the exponential distribution characterizing the users' activity, will diminish severely the number. Finally, we tested our method in three datasets of varying sizes and characteristics, but there are further types of social networks that we need to experiment with to certify our method's effectiveness to the fullest.

A potential future direction is to examine deeper and more complex architectures for INFECTOR to capture more intricate relationships. From the algorithmic part, we can experiment with bipartite matching algorithms, given sufficient resources. Overall, the main purpose of this work is primarily to examine the application of representation

learning in the problem of influence maximization and secondarily to highlight the importance of data-driven evaluation (e.g., DNI). In addition, we want to underline the strength of model-independent methods and evaluations, given the recent findings on the severe shortcomings of diffusion models. We hope this will pave the way for more studies to tackle influence maximization with machine learning means.

REFERENCES

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.

[2] S. Aral and P. S. Dhillon, "Social influence maximization under empirical influence models," *Nature Human Behaviour*, vol. 2, no. 6, p. 375, 2018.

[3] N. Du, L. Song, M. G. Rodriguez, and H. Zha, "Scalable influence estimation in continuous-time diffusion networks," in *NeurIPS*, 2013.

[4] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation learning for information diffusion through social networks: an embedded cascade model," in *WSDM*, 2016.

[5] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Learning continuous-time information diffusion model for social behavioral data analysis," in *ACML*, 2009.

[6] S. Aral and D. Walker, "Identifying influential and susceptible members of social networks," *Science*, vol. 337, no. 6092, pp. 337–341, 2012.

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

[8] M. E. G. Rossi and M. Vazirgiannis, "Exploring network centralities in spreading processes," in *International Symposium on Web AlGorithms (iSWAG)*, 2016.

[9] N. Du, Y. Liang, M. Balcan, and L. Song, "Influence function learning in information diffusion networks," in *ICML*, 2014, pp. 2016–2024.

[10] S. Feng, G. Cong, A. Khan, X. Li, Y. Liu, and Y. M. Chee, "Inf2vec: Latent representation model for social influence embedding," in *ICDE*, 2018.

[11] P. Lagrée, O. Cappé, B. Cautis, and S. Maniu, "Algorithms for online influencer marketing," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 1, p. 3, 2018.

[12] S. Vaswani, B. Kveton, Z. Wen, M. Ghavamzadeh, L. V. Lakshmanan, and M. Schmidt, "Model-independent online learning for influence maximization," in *ICML*, 2017.

[13] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, "Sketch-based influence maximization and computation: Scaling up with guarantees," in *International Conference on Conference on Information and Knowledge Management (CIKM)*, 2014, pp. 629–638.

[14] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *International Conference on Management of Data (SIGMOD)*, 2014, pp. 75–86.

[15] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*, 2015.

[16] A. Goyal, W. Lu, and L. V. Lakshmanan, "Simpath: An efficient algorithm for influence maximization under the linear threshold model," in *ICDM*, 2011.

[17] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010, pp. 1029–1038.

[18] B. Liu, G. Cong, Y. Zeng, D. Xu, and Y. M. Chee, "Influence spreading path and its application to the time constrained social influence maximization problem and beyond," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1904–1917, 2013.

[19] Y. Li, J. Fan, Y. Wang, and K.-L. Tan, "Influence maximization on social graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1852–1872, 2018.

[20] K. Zhang, J. Zhou, D. Tao, P. Karras, Q. Li, and H. Xiong, "Geodemographic influence maximization," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2764–2774.

[21] S. Pei, J. Wang, F. Morone, and H. A. Makse, "Influencer identification in dynamical complex systems," *Journal of Complex Networks*, vol. 8, no. 2, p. cnz029, 2020.

[22] K. Drakopoulos, A. Ozdaglar, and J. N. Tsitsiklis, "An efficient curing policy for epidemics on graphs," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 2, pp. 67–75, 2014.

[23] Z. Wen, B. Kveton, M. Valko, and S. Vaswani, "Online influence maximization under independent cascade model with semi-bandit feedback," in *NeurIPS*, 2017.

[24] L. Sun, W. Huang, P. S. Yu, and W. Chen, "Multi-round influence maximization," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2249–2258.

[25] Q. Wu, Z. Li, H. Wang, W. Chen, and H. Wang, "Factorization bandits for online influence maximization," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 636–646.

[26] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "A data-based approach to social influence maximization," *VLDB*, 2011.

[27] ——, "Learning influence probabilities in social networks," in *WSDM*, 2010.

[28] J. Wang, V. W. Zheng, Z. Liu, and K. C.-C. Chang, "Topological recurrent neural network for diffusion prediction," in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 475–484.

[29] Z. Wang and W. Li, "Hierarchical diffusion attention network," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3828–3834.

[30] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 1555–1564.

[31] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan, "Deepdiffuse: Predicting the 'who' and 'when' in cascades," in *ICDM*, 2018.

[32] Y. Wang, H. Shen, S. Liu, J. Gao, and X. Cheng, "Cascade dynamics modeling with attention-based recurrent neural network," in *IJCAI*, 2017.

[33] C. Li, J. Ma, X. Guo, and Q. Mei, "DeepCas: An end-to-end predictor of information cascades," in *International Conference on World Wide Web (The WebConf)*, 2017, pp. 577–586.

[34] C. Yang, J. Tang, M. Sun, G. Cui, and Z. Liu, "Multi-scale information diffusion prediction with reinforced recurrent networks," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 4033–4039.

[35] S. Liu, H. Shen, H. Zheng, X. Cheng, and X. Liao, "Ct lis: Learning influences and susceptibilities through temporal behaviors," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 6, pp. 1–21, 2019.

[36] M. E. Newman, "Assortative mixing in networks," *Physical review letters*, vol. 89, no. 20, p. 208701, 2002.

[37] G. Panagopoulos, F. D. Malliaros, and M. Vazirgiannis, "Diffu-greedy: An influence maximization algorithm based on diffusion cascades," in *Complex Networks*, 2018.

[38] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf, "Uncovering the temporal dynamics of diffusion networks," in *International Conference on Machine Learning (ICML)*, 2011, pp. 561–568.

[39] Y. Zhang, T. Lyu, and Y. Zhang, "Cosine: Community-preserving social network embedding from information diffusion cascades," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 2620–2627.

[40] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[41] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *KDD*, 2004.

[42] G. Panagopoulos, "Multi-task learning for commercial brain computer interfaces," in *BIBE*, 2017.

[43] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *KDD*, 2007.

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

[45] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "DeepInf: Modeling influence locality in large social networks," in *KDD*, 2018.

[46] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-j. P. Hsu, and K. Wang, "An overview of microsoft academic service (mas) and

applications," in *International Conference on World Wide Web (The WebConf)*, 2015, pp. 243–246.
[47] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li, "Social influence locality for modeling retweeting behaviors." in *IJCAI*, 2013.
[48] F. D. Malliaros, M.-E. G. Rossi, and M. Vazirgiannis, "Locating influential nodes in complex networks," *Nature Scientific reports*, vol. 6, p. 19307, 2016.
[49] F. D. Malliaros, C. Giatsidis, A. N. Papadopoulos, and M. Vazirgiannis, "The core decomposition of networks: theory, algorithms and applications," *VLDB J.*, vol. 29, no. 1, pp. 61–92, 2020.
[50] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *WSDM*, 2011.
[51] G. Panagopoulos, F. D. Malliaros, and M. Vazirgianis, "Influence maximization using influence and susceptibility embeddings," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 511–521.
[52] M. Xie, Q. Yang, Q. Wang, G. Cong, and G. De Melo, "Dynadiffuse: A dynamic diffusion model for continuous time constrained influence maximization." in *AAAI*, 2015, pp. 346–352.
[53] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.
[54] G. Panagopoulos, C. Xypolopoulos, K. Skianis, C. Giatsidis, J. Tang, and M. Vazirgiannis, "Scientometrics for success and influence in the microsoft academic graph," in *International Conference on Complex Networks and Their Applications*. Springer, 2019, pp. 1007–1017.

APPENDIX A

A.1 Non-convex Influence Spread

Given the diffusion matrix $\mathbf{D}$, the probability of a node v getting influenced by a seed set $\mathcal{U}$ is the complementary probability of not getting influenced by each node $u \in \mathcal{U}$. Summing this over all non-seed nodes gives the number of nodes the $\mathcal{U}$ can influence:

$$\sigma(\mathcal{U}) = \sum_{v \in G/\mathcal{U}} \Big(1 - \prod_{u \in \mathcal{U}} (1 - p_{u,v})\Big), \tag{18}$$

which is non-convex. 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. 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. Moreover the optimization of this non-convex function is non-trivial.

A.2 Influence Spread by Bipartite Matching

The diffusion matrix $\mathbf{D}$ can be interpreted as a bipartite network with influencers (cascade initiators) on one side and susceptible nodes on the left. If we view the diffusion probabilities as edge weights, a simplification used extensively in similar context [16], [17], we can transform the problem into weighted bipartite matching. However, in this case, each candidate seed (left side) can be assigned to more than one nodes (right side). The matching can

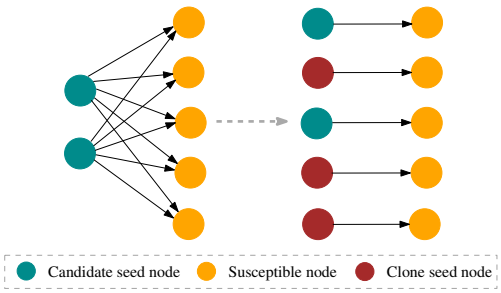

Figure 8: Example of cloning candidate seeds for perfect bipartite matching.

become perfect by creating a number of clone seeds for each candidate seed, and assigning them to the nodes the initial seed would be assigned to, in a one-to-one fashion, as can be seen in Figure 8.

The number of clones is basically equal to the expectation of a candidate seed u's influence spread λ_u. Cloning each node u for λ_u times transforms $\mathbf{D}$ into a balanced, weighted bipartite graph, for which we could use algorithms such as the Munkres assignment [53]. Unfortunately, it has a complexity of $\mathcal{O}(n^4)$ and requires constructing a non-sparse $N \times N$ matrix, which renders it prohibitive for real-world datasets. Though we can not use it directly, this interpretation remains useful for future directions.

A.3 The Microsoft Academic Graph Dataset

To construct the network and diffusion cascades from MAG, we utilized the tables PaperReferences, Author, PaperAuthorAffiliation,PaperFields and FieldsofStudy from the official from the official site[2]. Initially, we removed some one-paper authors based on a pattern we have identified in the past [54]. Subsequently, we kept only papers with fields that are included in the "Computer Science" category. To create the coauthorship network, we formed a graph based on the authors' coappearence in these papers using the "PaperAuthorAffiliation" table. To form the diffusion cascades between authors, we had to derive a sequence of papers (citing other papers) using "Paper References". Subsequently, we transform it into a cascade over our network by substituting the papers with their authors, and forming one separate cascade for each author of the initial paper (as all of them can be considered initiators). The authors of a certain paper that is included in the cascade appear in a random sequence accompanied with the same timestamp, which corresponds to the time between the initial paper's publication and the time this paper was published.

2. https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

George panagopoulos is a PhD candidate at LIX Ecole Polytechnique, France. He received his M.S. in Computer Science from University of Houston, US (2016), for which he received the best M.S. thesis award from the respective department. At that time, he was a research assistant at the Computational Physiology Lab and prior to that, at the Software Knowledge and Engineering Lab, NCSR Demokritos in Athens, Greece. His research interests lie in machine learning and its application to network science.

Fragkiskos D. Malliaros is an Assistant Professor at Paris-Saclay University, CentraleSupélec and associate researcher at Inria Saclay. He also co-directs the M.Sc. Program in Data Sciences and Business Analytics (CentraleSupélec and ESSEC Business School). Right before that, he was a postdoctoral researcher at UC San Diego (2016-17) and at École Polytechnique (2015-16). He received his Ph.D. in Computer Science from École Polytechnique (2015) and his M.Sc. degree from the University of Patras, Greece (2012). He is the recipient of the 2012 Google European Doctoral Fellowship in Graph Mining and the 2015 Thesis Prize by École Polytechnique. In the past, he has been the co-chair of various data science-related workshops, and has also presented ten invited tutorials at international conferences in the area of graph mining and data science. His research interests span the broad area of data science, with focus on graph mining, machine learning and network analysis.

Michalis Vazirgiannis is a Professor at LIX, Ecole Polytechnique in France. He has conducted research in GMD-IPSI, Max Planck MPI (Germany), in INRIA/FUTURS (Paris). He has been teaching in AUEB (Greece), Ecole Polytechnique, Telecom-Paristech, ENS (France), Tsinghua (China) and in Deusto University (Spain). His current research interests are on machine learning and combinatorial methods for Graph analysis and Text mining. He has active cooperation with industrial partners in the area of data analytics and machine learning for large scale data repositories in different application domains such as Web advertising and recommendations, social networks, medical data, aircraft logs, insurance data. He has supervised fourteen completed PhD theses - several of the supported by industrial funding including Google and Airbus. On the publication frontier, he has contributed chapters in books and encyclopedias, published three books and more than a hundred forty papers in international refereed journals and conferences. He is also co-author of three patents. He has received the ERCIM and the Marie Curie EU fellowships and lead a DIGITEO chair grant. Since 2015 M. Vazirgiannis leads the AXA Data Science chair.