MONSTOR: An Inductive Approach for Estimating and Maximizing Influence over Unseen Social Networks

Jihoon Ko¹, Kyuhan Lee¹, Kijung Shin¹ and Noseong Park²
¹KAIST, Graduate School of AI and the School of Electrical Engineering
²Yonsei University, Department of AI
{jinhoonko, kyuhan.lee, kijungs}@kaist.ac.kr, npark9@gmu.edu

Abstract

Influence maximization (IM) is one of the most important problems in social network analysis. Its objective is to find a given number of seed nodes who maximize the spread of information through a social network. Since it is an NP-hard problem, many approximate/heuristic methods have been developed, and a number of them repeats Monte Carlo (MC) simulations over and over, specifically tens of thousands of times or more, to reliably estimate the influence of a seed set, i.e., the number of infected nodes. In this work, we present an inductive machine learning method, called Monte Carlo Simulator (MONSTOR), to predict the results of MC simulations on networks unseen during training. MONSTOR can greatly accelerate existing IM methods by replacing repeated MC simulations. In our experiments, MONSTOR achieves near-perfect accuracy on unseen real social networks with little sacrifice of accuracy in IM use cases.

1 Introduction

Viral marketing via influence maximization has received considerable attention over the last two decades, as social networks have become an essential part of our daily lives. Many people connect to and acquire information from social networks on a daily basis, and thus information diffusion over such social networks is often more effective than that over conventional media, such as newspapers and television.

Influence maximization (IM) is to find a certain number of seed nodes who maximize the spread of information through a social network. There exist several real-world applications where influence maximization played a key role, such as 2010 U.S. congressional elections, attempts to raise awareness about HIV among homeless youth, and so on [2, 18].

Unfortunately, finding optimal seed nodes in a social network is NP-hard. While it is a submodular optimization problem for which a greedy approach guarantees the approximation ratio of 63% [13], even the greedy approach is practically prohibitive. This is because it repeats Monte Carlo (MC) simulations of information cascade processes many times, more specifically tens of thousands of times or more, to accurately estimate the influence of a seed set, i.e., the number of infected nodes from the seed set through multi-hop cascade processes [20].

While numerous heuristic/approximated methods for IM have been proposed, many of them, including CELF [5] and UBLF [20], still rely heavily on repeated MC simulations. The time complexity of an MC simulation is \(O(|E|)\), where \(|E|\) is the number edges, and thus the time complexity of estimating the influence of a seed set via \(d\) simulations is \(O(d|E|)\) — UBLF uses \(d = 10,000\). Note that \(O(d|E|)\) operations are required for a single seed set, while a large number of seed sets are considered during an execution of the IM methods.

In this work, we propose a neural network-based method, called Monte Carlo Simulator (MONSTOR), for predicting MC simulation results. MONSTOR is inductive, i.e., it is capable of predicting MC simulation results in social networks unseen during training. After being trained, it can significantly speed up existing IM methods by replacing their computational bottleneck, i.e., repeated MC simulations, as illustrated in Figure 1. Specifically, in our experiments, MONSTOR’s prediction time is similar to the runtime of performing about 100 – 200 MC simulations.

There are two popular information cascade models under
which influence of users are measured: the independent cascade (IC) and linear threshold (LT) models [11]. In this work, we predict MC simulation results of IC and leave LT as our future work since we found that prediction under LT requires different approaches as discussed in Section 5. Note that prediction under IC is also challenging, and addressing it requires non-trivial solutions.

We conduct experiments with three real-world social networks. One strong point in our experiments is that we use real influence probabilities of edges. That is, we weight each directed edge \((u, v)\) with the probability that user \(u\) influences \(v\). We calculate the probabilities from retweet logs, and to this end, we collect tweet and retweet logs using commercial Gnip APIs [1] or from previous studies. Note that most previous studies on influence maximization simply used random/uniform/degree-based probabilities [5, 7, 14, 17, 20], which are very different from real ones.

In our experiments, MONSTOR yields near-perfect predictions. Specifically, it achieves the Pearson’s correlation coefficients of 1.000 and the Spearman’s Rank correlation coefficient of 1.000. To evaluate the efficacy of MONSTOR in real-world influence maximization use cases, we modify the greedy-based methods in [5, 20], which are known as the most accurate methods, by substituting their MC simulation sub-routines with MONSTOR. The greedy-based algorithms equipped with MONSTOR yield almost the same influence maximization results as the original algorithms based on MC simulations on real social networks unseen during training. Moreover, our algorithms are more accurate than state-of-the-art approximated/heuristic methods in many cases.

2 Preliminaries & Related Work

We review graph convolutional networks (GCNs), which the proposed MONSTOR method is based on. We also describe the basic concepts of the influence maximization problem.

2.1 Graph Convolutional Networks

Given a graph \(G = (V, E)\) and a node feature vector \(f_v\) for each node \(v \in V\), GCNs perform classification or regression for each node \(v\) — we use boldface to denote vectors.

Several GCN methods have been proposed, and each of them is described by the following equation with a specific choice for each of the aggregation function \(agr\) and the combination function \(cmb\):

\[
\begin{align*}
\mathbf{a}_v^i &= agr(\{\mathbf{h}_u^{i-1} | u \in Nei(v)\}), \\
\mathbf{h}_v &= cmb(\mathbf{a}_v^i, \mathbf{h}_v^{i-1}),
\end{align*}
\]

where \(i\) denotes the \(i\)-th convolutional layer, and \(Nei(v)\) is a set of neighbors of \(v\). After \(k\) convolutional layers, one performs classification or regression for each node \(v\) using \(h_v^k\). To this end, one more activation layer, such as sigmoid, softmax, and so forth, is needed at the end.

The aggregation and combination functions of some widely-used GCN methods are as follows: First, LGCN [3] uses \(l\)-Max_Pooling to aggregate neighbor features, which selects the \(l\)-largest values in each feature dimension from neighboring nodes. A simple concatenation and convolutional layer are used for the combination function. Second, GraphSAGE [6] introduced three.aggregators: Mean_Pooling, LSTM, and Max_Pooling. A simple concatenation and one linear layer are used to combine \(a_v^i\) and \(h_v^{i-1}\). In these perspectives, GraphSAGE has more options for the aggregation function, while it uses a simpler combination function than LGCN. Third, Graph Attention Network (GAT [16]) attends neighbors, i.e., it weights neighbors while aggregating their features. It uses linear layers for the aggregation and combination.

2.2 Influence Maximization

Influence maximization, i.e., to find a certain number of seed nodes who maximize the information spread through a social network, is an NP-hard problem. There are two major issues regarding influence maximization: i) how to model the information cascade, and ii) how to solve the NP-hard problem.

Independent cascade (IC) and linear threshold (LT) are the two most popular information cascade models. In the IC model, once a node \(u\) is activated (or influenced), it attempts once to activate each neighbor \(v\) with probability \(p(u,v)\). In the LT model, a node \(v\) is activated if a sufficient number of its neighbors (larger than a threshold) are activated. In this work, we focus on the IC model.

Numerous methods have been proposed for influence maximization. Since it is an NP-hard problem, all these methods try to approximate optimal seed nodes. They can be categorized into the following three types: i) simulation-based, ii) proxy-based, and iii) sketch-based methods. Among them, simulation-based methods are known to be able to find better seeds than the other methods.

In the simulation-based methods, MC simulations are explicitly repeated to estimate the influence of a seed set [5, 20]. These methods focus on pruning unnecessary (redundant) simulations to minimize the required number of simulations. SSA [14] and D-SSA [14] are strong among sketch-based method, and IRIE [8] and PMIA [17] are strong among proxy-based methods. See a recent survey [11] for details.

There exist two on-going studies [10, 19] relevant to ours. Most importantly, their machine-learning models are transductive, i.e., they are not capable of predicting influence on social networks unseen during training. For inference in a social network, they require the same network, potentially with different influence probabilities, for training and testing (see the next subsection for the definition of influence probability). We are more ambitious, and we aim at designing an inductive method, which is capable of predicting influence in social networks whose connections and influence probabilities are completely unseen during training. In addition, the former [10] directly searches seed nodes, while our method

\footnote{1This does not mean that our task definition is trivially easy. The influence measurement in IM is a research field on its own and many approximated methods have been proposed [8, 14, 17]. The IM method based on MONSTOR outperforms them in our experiments.}

\footnote{2One is not published yet, and the other is an extended abstract with lack of details. Their source code is not provided.}
predicts MC simulation results. Thus, their prediction task is different from ours.

2.3 Influence Probability

We use interaction logs such as retweets among users to construct the network \( G = (V, E) \). The edge weight on a certain directional edge \( e = (u, v) \) represents the influence probability from the source node \( u \) to the destination node \( v \). We denote the weighted adjacency matrix (or influence probability matrix) as \( P \). We use boldface to denote vectors and matrices.

There are several methods to define the influence probability of an edge \( e = (u, v) \), denoted as \( p_e \) or \( p_{(u,v)} \), as follows:

1. Bernoulli Trial (BT): \( p_{(u,v)} = \frac{\text{actions}(u) \cap \text{actions}(v)}{\text{actions}(u)} \),
2. Jaccard Index (JI): \( p_{(u,v)} = \frac{|\text{actions}(u) \cap \text{actions}(v)|}{|\text{actions}(u) \cup \text{actions}(v)|} \),
3. Linear Probability (LP): \( p_{(u,v)} = \frac{|\text{actions}(u) \cap \text{actions}(v)|}{|\text{actions}(v)|} \),

where \( \text{actions}(x) \) denotes a set of actions done by node \( x \), e.g., a set of online postings retweeted or replied by node \( x \). Note that \( \sum_{x \in N_{\text{nei}}(v)} p_{(v,x)} = 1 \) in the last LP definition, while this is not the case in the other definitions.

Each probability definition has its own meaning. The first BT definition corresponds to the maximum likelihood of the Bernoulli trial of \( u \) attempting to infect \( v \) [4]. The JI definition use the Jaccard index which was originally defined to compare the similarity of two sets [4]. The LP definition was first introduced by Kempe et al. in their seminal paper introducing the linear threshold cascade model [9].

In this work, therefore, we consider all the three different definitions of influence probability, and as a result, we can define three different influence probability matrices, \( P_{BT} \), \( P_{JI} \), and \( P_{LP} \), from a social network.

Definition 1 (Influence Probability). The influence probability \( p_{(u,v)} \) means the probability that node \( u \) directly influences node \( v \) during the information cascade process.

Definition 2 (Influence Probability Matrix). Given a social network \( G = (V, E) \), each \((u,v)\)-th entry of the influence probability matrix \( P \in [0,1]^{V \times V} \) is the influence probability \( p_{(u,v)} \). Note that \( P \) is sparse with many zeros in general.

3 Problem Definition

Given a set of seed nodes \( S \subseteq V \), we predict \( \rho(x) \), the infection probability that a node \( x \in V \) is infected directly or indirectly from \( S \). It is obvious that \( \rho(x) = 1 \) if \( x \in S \).

Definition 3 (Infection Probability). Given a seed set \( S \) of infected nodes, the infection probability \( \rho(x) \) denotes the probability that a node \( x \) is infected directly or indirectly from \( S \) during the information cascading process.

Our goal is to design a general inductive model to predict \( \rho(x) \) in a social network that is not necessarily a part of training data. In other words, our model aims to predict infection probabilities in social networks unseen during training.

Figure 2: An example of how to collect training data. Given a network \( G \) whose influence probability matrix is \( P \) and a seed set, we perform many simulations and collect \( \pi_0 \) to \( \pi_h \). We repeat these steps with many seed sets. Note that the inner product \( \langle 1, \pi_h \rangle \) is the influence, i.e., the number of people infected by the input seed set. We also note the length of MC simulations can be varied even for the same seed set — if a simulation ends early at step \( i \), we consider the cascade is fixed after that when calculating \( \pi_j \), \( j \geq i \).

4 Proposed Method

We first sketch our proposed method, MONSTOR, and then describe its details. Below, we define two concepts necessary for describing MONSTOR. Note that seed nodes are infected in the 0-th step of the information cascading process, and those infected directly by seed nodes are infected in the 1-st step. Therefore, a step means one-hop cascade process from the nodes currently infected.

Definition 4 (Infection Probability within Limited Steps). Given a seed set \( S \subseteq V \), \( \rho_i(x) \) denotes the infection probability during the first \( i \) steps of the information cascading process. Note that \( \rho_i(x) \approx \rho(x) \) if \( i \) is sufficiently large, and \( \rho_i(x) = \rho(x) \) if \( i \) is greater than equal to the longest path in the social network.

Definition 5 (Infection Probability Vector). Let \( \rho(x) \in [0,1]^{|V|} \) be the vector of \( \rho(x) \), \( \forall x \in V \). We denote it as \( \pi \equiv \rho(x) \) and similarly \( \pi_i \equiv \rho_i(x) \).

From the above definitions, Proposition 1 trivially holds.

Proposition 1. The infection probability monotonically increases w.r.t. \( i \). Therefore, \( \pi_i \leq \pi_{i+1} \).
4.1 Overall Workflow

We describe the overall workflow in our method as follows:

1. We collect many example social networks \{G_1, G_2, \cdots \}.
2. From each \( G_j \), we collect a tuple of 
   \((\pi_{i-1}, \pi_{i-2}, \cdots, \pi_{i-e}, P_j)\), where \( e > 1 \) is a hyper-parameter, after choosing a seed set \( S \) randomly. \( P_j \) can be in BT, JI, or LP. We repeat this multiple times with different seed sets. These collected tuples from all social networks are our training set, as in Fig. 2.
3. We train our GCN-based model \( M \) that has \( l \) graph convolutional layers with the training set to predict \( \pi_i \) given \( \pi_{i-1}, \cdots, \pi_{i-e} \). In other words, \( M \) predicts one-step simulations rather than end-to-end simulations.
4. We stack the pre-trained model \( s \) times, and this stacked GCN predicts \( \pi_s \) from \( \pi_0 \). That is, it predicts end-to-end simulations. Hereinafter, MONSTOR means the stacked GCN, described in Fig. 3, unless otherwise stated.
5. We use MONSTOR to replace the MC simulation sub-routine of existing influence maximization algorithms, as described in Fig. 1 (b). This replacement can significantly accelerate the algorithms.

Note that, in the training phase, we use MC simulations of information cascade processes to obtain \( \pi \). Note that, in the training phase, we use MC simulations of information cascade processes to obtain \( \pi \). However, in the testing phase, MC simulations in (potentially unseen) target social networks are not needed.

4.2 Detailed Design

We describe our GCN architecture and its training method. We modify the aggregation function in Eq. (1) in the following way:

\[
\mathbf{a}_i^t = \text{agr}(\{p_{(u,v)}\mathbf{h}^{i-1}_u | u \in Nei(v)\}), \tag{2}
\]

where we multiply \( \mathbf{h}^{i-1}_u \) by \( p_{(u,v)} \).

Our particular choice of multiplying them makes sense considering the definition of IC where we keep multiplying the influence probabilities following a cascade route. For instance, the probability that \( u_1 \) infects \( u_2 \) and \( u_2 \) infects \( u_3 \) can be calculated by \( p_{(u_1,u_2)}p_{(u_2,u_3)} \).

As stated earlier, our model predicts \( \pi_i \) given \( \pi_{i-1}, \cdots, \pi_{i-e} \). Whereas one can try to directly predict the raw values in \( \pi_i \), we use the following more effective prediction method inspired by the monotonicity of Proposition (1):

\[
M(\pi_{i-1}, \cdots, \pi_{i-e}, P; \theta) \equiv \min\{\pi_{i-1} + g, U_i\}, \tag{3}
\]

\[
g = \sigma(H^tW), \tag{4}
\]

where \( M \) is our GCN model; \( \theta \) is its model parameters; \( \sigma(\cdot) \) is the ReLU activation; \( g \in \mathbb{R}^{\vert V \vert} \) is a vector of additive influence; \( U_i \) is the theoretical upper bound of \( \pi_i \) which we will define shortly. \( H^t \in \mathbb{R}^{\vert V \vert \times d} \) is the matrix whose rows are \( \mathbf{h}^t_u \in \mathbb{R}^d, u \in V \) and \( W \in \mathbb{R}^{d \times 1} \) is a training parameter of \( M \) to produce the vector \( g \). In the above definition, we predict a vector \( g \) and the right-hand side of Eq. (3) means the values in between \( \pi_{i-1} \) and \( U_i \). Its detailed workflow is in Fig. 4.

By teaching \( M \) that the possible maximum is \( U_i \), we can relieve the difficulty of the prediction task — in our preliminary studies, we found that directly predicting the raw values in \( \pi_i \) is not as successful as our design.

Of many possible loss definitions, we train our model \( M \) with the following error loss:

\[
L = \sum_{t \in T} \|M(t; \theta) - \pi_i\|_1 / |T| + \lambda \cdot \text{abs}(1 \cdot M(t; \theta) - 1 \cdot \pi_i) / \pi_i, \tag{5}
\]

where \( T \) is a training set, \( t = (\pi_i, \pi_{i-1}, \cdots, \pi_{i-e}, P) \in T \) is a training sample, and \( \cdot^t \) means the dot product.

**Upper Bound of Infection Probability** In this paragraph, we analyze i) the upper bound \( U \) of \( \pi \) and ii) the upper bound \( U_i \) of \( \pi_i \). Recall that \( \pi \approx \pi_h \) when \( h \) is sufficiently large.

The following proposition follows [20].

**Proposition 2.** Given a seed set \( S \), the vector \( \pi \) is bounded by \( U \equiv \min(1, \pi_0 \cdot (1 - P)^{-1}) \), where \( \pi_0 \in \{0, 1\}^{|V|} \) is the multi-hot vector representing \( S \), i.e., \( \pi_0 \) has 1 at the index of each seed user and 0 for others, and \( I \) is a unit matrix.

The following Thm. 1 is one of our key theoretical results. Since our prediction relies on \( U_i \) (cf. Eq. (3)), it is crucial to make \( U_i \) be as tight as possible.

**Theorem 1.** Given a seed set \( S \) and \( \pi_{i-1} \), an upper bound \( U_i \) of \( \pi_i \) can be defined as follows:

\[
\pi_i \leq U_i \equiv \pi_{i-1} + (\pi_{i-1} - \pi_{i-2})P \tag{6}
\]

**Proof.** During a MC simulation, let \( N \) be a set of users newly infected at cascade step \( i - 1 \) and \( o \) be a multi-hot vector representing them. The expected number of their neighbors newly infected at step \( i \) will be no larger than \( oP \) because some neighbors cannot be activated by the definition of IC, i.e., if a user \( u \) fails to activate its neighbor \( v \) at a step earlier than \( i - 1 \), \( u \) cannot activate \( v \) after that. The same argument can be applied among \( \pi_i, \pi_{i-1}, \pi_{i-2} \) as well.

**Upper Bound of Prediction Error** When \(|P| < 1\) and \(|P|_\infty < 1\), similar analyses give an upper bound of \( \pi - \pi_s \) as follows:

**Theorem 2.** Given a seed set \( S \), if \(|P| < 1\) and \(|P|_\infty < 1\), then

\[
\pi - \pi_s \leq \sum_{i=1}^{\lfloor |V\setminus S| - s \rfloor} \pi_s \cdot P^i. \tag{7}
\]

**Proof.** Given an influence probability matrix \( P \), \( P^i \equiv \prod_{j=1}^i P \) represents the influence probability through exactly
i step propagation. Thus, \( \sum_{k=1}^{s} |V \setminus S - s| P^k \) contains the influence probability through all possible infection paths (whose lengths are \( |V \setminus S| - s \)) when there are no failures in infecting neighbors. Therefore, \( \pi_s + \sum_{k=1}^{s} |V \setminus S| - s \pi_s P^k \) is an upper bound of \( \pi \) and the theorem is proved. \( \square \)

However, the right-hand side converges quickly even with a small \( i \) since in many cases, \( p(u,v) \in P \) is small and \( P^i \) becomes close to zero quickly (exponentially) as \( i \) increases. Therefore, we can say that \( \pi_s \) is a reasonable approximation of \( \pi \) if \( s \) is sufficiently large. Our model \( M \) predicts \( \pi_s \), as we stack trained GCNs \( s \) times. Therefore, our model provides a reasonable approximation of \( \pi \) if it is able to accurately predict \( \pi_s \) since \( \pi - \pi_s \) becomes close to zero very quickly as \( s \) increases.

### 5 Discussion on Linear Threshold

We discuss why we focus on the IC model but exclude the LT model in this work. As described earlier, we first produce \( g \) and convert to the final prediction (cf. Eqs. (3) and (4)). For the LT cascade model, it is not easy to find tight upper bounds \( [20] \). If \( U_i \) is not tight, our design not to directly predict the raw values will be discouraged. For this reason, we leave influence prediction under LT as an open problem.

### 6 Complexity and Runtime Analyses

The complexity of \( d \) MC simulations is \( O(d|\mathcal{E}|) \) as we stated earlier. When using GraphSAGE, one of the most widely-used GCN models as of now, the inference time complexity of MONSTOR becomes \( O(ls|\mathcal{E}|) \), where \( s \) is the number of stacks and \( l \) is the number of graph convolutional layers per stack. In general, \( ls \) is few orders of magnitude smaller than \( d \). In our experiments, for instance, \( l = 3 \) and \( s = 2, 3, 5 \) are used, which leads to \( ls = 6, 9, 15 \). Whereas it is hard to achieve reliable results only with such a small MC simulation numbers, our proposed method is able to infer in a reliable manner. The standard configuration proposed by UBLF \( [20] \) is \( d = 10,000 \), although it can be different depending on the number of vertices/edges and the scale of influence probabilities. Our MONSTOR's prediction time amounts to the runtime of performing 100 - 200 MC simulations in our experiments, which is two orders of magnitude smaller than the standard configuration of UBLF.

### 7 Experiments

We present our experimental settings and results.

#### 7.1 Experimental Environments

**Datasets** We use three real-world social networks: Extended, WannaCry, and Celebrity. For Extended, we use the data used in \([15]\) and crawled more tweets and retweets on our own. WannaCry is collected by us and contains many online postings about WannaCry, the first ransomware, and their cascade patterns in Twitter. Celebrity is used in \([12]\) and contains many postings by online celebrities and their cascade patterns. All the datasets span for more than 12 months and have timestamps.
Table 2: The quality of MONSTOR’s influence predictions. Note that in the rows in blue, a testing social network is missing in its training data. We write the training social networks inside the parentheses and E, W, and C means Extended, WannaCry, and Celebrity, respectively. This is to know how MONSTOR reacts to unseen social networks.

| In Exchange | Influence Maximization | Pearson Correlation | Spearman’s Rank |
|-------------|------------------------|---------------------|-----------------|
| BT          | Greedy (UBLF)          | 0.829               | 0.829           |
|             | Greedy (CELF)          | 0.829               | 0.829           |
|             | CELF-MON (E+W)         | 0.829               | 0.829           |
|             | CELF-MON (E+C)         | 0.829               | 0.829           |
|             | CELF-MON (W+C)         | 0.829               | 0.829           |
|             | D-SSA                  | 0.564               | 0.564           |
|             | SSA                    | 0.477               | 0.477           |
|             | IRIE                   | 0.421               | 0.421           |
|             | PMIA                   | 0.416               | 0.416           |
|             | Greedy (UBLF)          | 0.829               | 0.829           |
|             | Greedy (CELF)          | 0.829               | 0.829           |
|             | CELF-MON (E+W)         | 0.829               | 0.829           |
|             | CELF-MON (E+C)         | 0.829               | 0.829           |
|             | CELF-MON (W+C)         | 0.829               | 0.829           |
|             | D-SSA                  | 0.564               | 0.564           |
|             | SSA                    | 0.477               | 0.477           |
|             | IRIE                   | 0.421               | 0.421           |
|             | PMIA                   | 0.416               | 0.416           |

Table 3: The influence maximization results with various seed numbers $k = 10, 50, 100$. We consider all the possible combinations of the train-test splits and the influence probability definitions. UBLF and CELF are the state-of-the-art methods with the best accuracy. The best results (except for UBLF and CELF) are highlighted in boldface. We also highlight the predictions for unseen social networks in blue.

| In Exchange | Influence Maximization | Pearson Correlation | Spearman’s Rank |
|-------------|------------------------|---------------------|-----------------|
| BT          | Greedy (UBLF)          | 0.829               | 0.829           |
|             | Greedy (CELF)          | 0.829               | 0.829           |
|             | CELF-MON (E+W)         | 0.829               | 0.829           |
|             | CELF-MON (E+C)         | 0.829               | 0.829           |
|             | CELF-MON (W+C)         | 0.829               | 0.829           |
|             | D-SSA                  | 0.564               | 0.564           |
|             | SSA                    | 0.477               | 0.477           |
|             | IRIE                   | 0.421               | 0.421           |
|             | PMIA                   | 0.416               | 0.416           |

7.2 Experimental Results

We focus on evaluating the prediction accuracy of MONSTOR. Theoretical and practical runtime analyses can be found in Sec. 6.

Influence Prediction

As there are no inductive models for similar prediction tasks on unseen social networks, we compare MONSTOR only with ground truth. Table 2 summarizes the influence prediction results in terms of the Pearson’s correlation coefficients and the Spearman’s Rank correlation, which are to measure the similarity of true and predicted influences. We first calculate the ground truth influences and the rankings of seed sets for all the testing cases with the large enough number (i.e., $d = 10,000$) of MC simulations. Our predictions shine with the almost perfect values close to 1.0. However, this doesn’t mean that our task is trivial. Straightforward approaches, such as linear regression, do not perform well and even the famous IM methods, such as D-SSA, IRIE, and so forth, do not measure the influence as accurately as MONSTOR as will be described shortly.

Influence Maximization

We summarize the results of all influence maximization use cases in Table 3. For BT and JI, UBLF performs best. For LP, UBLF is not applicable, and CELF performs best. UBLF and CELF are most accurate, while they rely on MC simulations and thus take longer time to finish than other approximated/heuristic methods.

For BT, UBLF-MON (E+C) is most accurate (except for UBLF) in 4 out of 9 cases and its influence is very close to that of UBLF. IRIE is an effective approximated method, and it was most accurate in one case, i.e., WannaCry with $k = 100$ seeds.

In JI, there is no clear winner because each method (except SSA and D-SSA) has at least one case where it was most accurate. The main reason is that the influence probabilities are relatively smaller than the others, as shown in Table 1, e.g., 0.07974 for the BT training data of Extended vs. 0.03345 for the JI training data of the same social network. UBLF-MON and IRIE show reliable accuracy in almost all cases for JI.
UBLF-MON is clearly the most accurate for LP, which is the most challenging case. LP has the largest influence probabilities, as shown in Table 1, so small changes can decrease the influence significantly. CELF-MON shows the accuracy close to CELF. One interesting point is that IRIE does not work in LP as well as in BT and JI. In many cases of LP, D-SSA and SSA outperforms IRIE.

All in all, only UBLF-MON (or CELF-MON) performs reliably (even in unseen social networks). For all the influence probability definitions, its accuracy is close to the original method (i.e., UBLF or CELF) based on repeated MC simulations. The other baselines do not show reliable performance in at least one probability definition, e.g., IRIE in LP.

8 Conclusions
In this work, we tackled the inductive inference of the influence of seed nodes under the independent cascade model. We carefully designed a GCN-based model called MONSTER for our needs, and we tested it with three real-world social networks. MONSTER is able to predict the influence almost perfectly on unseen social networks. Moreover, in influence maximization use cases, greedy approaches equipped with MONSTER, which replaces repeated MC simulations, perform reliably, outperforming state-of-the-art methods in many cases.

References
[1] Gnip APIs. https://support.gnip.com/apis/, 2020.
[2] C. Aslay, L. V. Lakshmanan, W. Lu, and X. Xiao. Tutorial on influence maximization in online social networks. In WSDM, 2018.
[3] H. Gao, Z. Wang, and S. Ji. Large-scale learnable graph convolutional networks. In KDD, 2018.
[4] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In WSDM, 2010.
[5] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In WWW, 2011.
[6] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In NIPS, 2017.
[7] K. Jung, W. Heo, and W. Chen. IRIE: A scalable influence maximization algorithm for independent cascade model and its extensions. In ICDM, 2011.
[8] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In ICDM, 2012.
[9] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In KDD, 2003.
[10] H. Li, M. Xu, S. S. Bhowmick, C. Sun, Z. Jiang, and J. Cui. DISCO: influence maximization meets network embedding and deep learning. CoRR, abs/1906.07378, 2019.
[11] Y. Li, J. Fan, Y. Wang, and K.-L. Tan. Influence maximization on social graphs: A survey. IEEE Transactions on Knowledge and Data Engineering, 30:1852–1872, 2018.
[12] J. Liu, Y. Chen, D. Li, N. Park, K. Lee, and D. Lee. Predicting influence probabilities using graph convolutional networks. In IEEE BIG DATA, 2019.
[13] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions—i. Mathematical Programming, 14(1):265–294, 1978.
[14] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In SIGMOD, 2016.
[15] C. Sabottke, O. Suciu, and T. Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In USENIX Security, 2015.
[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In ICLR, 2018.
[17] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. Data Mining and Knowledge Discovery, 25, 11 2012.
[18] A. Yadav, H. Chan, A. Xin Jiang, H. Xu, E. Rice, and M. Tambe. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In AAMAS, 2016.
[19] B. Yan, K. Song, J. Liu, F. Meng, Y. Liu, and H. Su. On the maximization of influence over an unknown social network. In AAMAS, 2019.
[20] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo. Ublf: An upper bound based approach to discover influential nodes in social networks. In ICDM, 2013.