A Diffusion Model for Maximizing Influence Spread in Large Networks

Tu-Thach Quach(✉) and Jeremy D. Wendt(✉)

Sandia National Laboratories, Albuquerque, NM, USA
{tong,jdwendt}@sandia.gov

Abstract. Influence spread is an important phenomenon that occurs in many social networks. Influence maximization is the corresponding problem of finding the most influential nodes in these networks. In this paper, we present a new influence diffusion model, based on pairwise factor graphs, that captures dependencies and directions of influence among neighboring nodes. We use an augmented belief propagation algorithm to efficiently compute influence spread on this model so that the direction of influence is preserved. Due to its simplicity, the model can be used on large graphs with high-degree nodes, making the influence maximization problem practical on large, real-world graphs. Using large Flixster and Epinions datasets, we provide experimental results showing that our model predictions match well with ground-truth influence spreads, far better than other techniques. Furthermore, we show that the influential nodes identified by our model achieve significantly higher influence spread compared to other popular models. The model parameters can easily be learned from basic, readily available training data. In the absence of training, our approach can still be used to identify influential seed nodes.

1 Introduction

Social networks often show that different users have varying levels of influence. As an example, tweets from some users are more likely to spread than from others. In a network of friends, an individual adopting a product may cause others to do the same. Identifying these influential nodes has important applications. For instance, in marketing, an organization wants to identify which small set of nodes will return the highest influence spread given a limited budget. Finding the seed nodes that maximize influence spread is called *influence maximization*.

Influence maximization requires two inputs: a graph (with nodes representing individuals and edges representing relationships between any two individuals), and a diffusion model. Given the graph and the diffusion model, influence maximization finds k seed nodes such that the expected number of nodes influenced is maximized [11].

A variety of diffusion models have been proposed and analyzed. Two popular diffusion models are the independent cascade (IC) model and linear threshold (LT) model [11]. In the IC model, each active node i has one opportunity to

E. Spiro and Y.-Y. Ahn (Eds.): SocInfo 2016, Part I, LNCS 10046, pp. 110–124, 2016.
DOI: 10.1007/978-3-319-47880-7_7

activate a neighboring node j with probability p_{ij}. In the LT model, each node j is influenced jointly by all neighboring nodes $i \in N(j)$ ($N(j)$ is the set of neighbors of node j). Each node j is influenced by each neighbor i with weight p_{ij} such that the sum of all incoming weights to j is at most 1. Each j determines a threshold t_j. If the sum of the incoming weights exceeds t_j, then j is activated.

A major drawback of these models is the computation: to get reasonable estimates of influence spread for a single node, these diffusion models require running Monte-Carlo simulations on the network many times (typically 10,000). This is clearly feasible only on small networks. In an effort to minimize this problem, several heuristics have been proposed to estimate the spread without resorting to Monte-Carlo simulations [2,3,14]. Others have proposed entirely new diffusion models. A probabilistic voter model found that the optimal seed nodes are those with the highest degree [6]. Markov models have also been proposed [5, 17]. Unlike cascade models, which capture the evolution of influence over time, Markov models capture the interactions of nodes as a set of interdependent random variables. Abandoning diffusion models altogether, the credit assignment approach uses historical logs to directly compute the influence of a node [8]. Many of these methods have parameters that need to be defined as well. When training data is available, a model's parameters can be learned [7,18]. In the absence of training data, constants and heuristics, such as weighted cascade where p_{ij} is inversely proportional to the in-degree of node j, are often used instead.

Given the existence of several proposed diffusion methods, practitioners must determine which diffusion model is the right one to use in any given situation. We propose three considerations for identifying which model to use:

1. A diffusion model should match well with ground-truth data when available. This validates the model and justifies its use for influence maximization.
2. A diffusion model's parameters can be learned from readily available data. In other words, the training data required should be practical to obtain. Furthermore, the model should still be usable when no training data is available.
3. A diffusion model should be computationally efficient so that it scales to large networks. Thus, practical diffusion models cannot rely on costly Monte Carlo simulations.

With these considerations in mind, we present a new diffusion model based on pairwise factor graphs that predicts influence spread for a given seed set. An efficient belief propagation algorithm is used to compute influence spread; it can be used on large real-world graphs with high-degree nodes. We provide experimental results showing that our model predictions match ground-truth spreads using a large Flixster dataset [10] and an Epinions dataset [19]. We then investigate the influence maximization problem under our model and show that the influential nodes identified by our model achieve higher influence spread compared to other popular models. The model parameters can easily be learned entirely from data. Moreover, the type of training data required is simple and practical. In the absence of training data, our model can still identify influential seeds.

This paper is organized as follows. Details of our diffusion model and the associated algorithms are presented in Sect. 2. Experimental results are provided in Sect. 3. Concluding thoughts are provided in Sect. 4.

2 Influence Spread

Given a graph $G = (\mathcal{V}, \mathcal{E})$ of n nodes, a set of seed nodes $\mathcal{S} \subseteq \mathcal{V}$, and a diffusion model Ω, the influence spread, $\sigma_\Omega(\mathcal{S})$, is the expected number of influenced or activated nodes. Here we adopt a factor graph, which can represent general graphical models including Markov networks and Bayesian networks. Our diffusion model consists of unary (ϕ) and pairwise (ψ) factors (potential functions). Specifically, each node $i \in \mathcal{V}$ has a corresponding state $x_i \in \{0, 1\}$ that indicates whether or not node i adopts the product (e.g., $x_i = 1$ means node i adopts the product). The adoption probability, $p_i(x_i)$, depends on not only i's preference, but also the states of its neighbors. The joint probability distribution of the states of the network is

$$p(x_1, \ldots, x_n) \propto \prod_{i \in \mathcal{V}} \phi_i(x_i) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(x_i, x_j). \tag{1}$$

Note that the unary potential function expresses the state preference of node i independent of its neighbors. The pairwise potential function expresses the dependency between neighboring nodes i and j whenever an edge between i and j exists in $\mathcal{E}$. The pairwise potential function depends on only two nodes and allows the model to deal with high-degree nodes directly (instead of pruning excess edges on high-degree nodes as in [14]). With this model, the marginal probability of each node i is

$$p_i(x_i) = \sum_{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n} p(x_1, \ldots, x_n). \tag{2}$$

Computing the marginal probabilities can be done efficiently using belief propagation [21].

For undirected graphs, the above model can be used to compute the marginal probability of each node, which corresponds to its adoption probability. For directed graphs, an edge's direction indicates the *direction* of influence. Therefore, we propose to adapt the above model for directed influence. Consider the graph shown in Fig. 1. In this case, the state of node 2 depends on node 1's state, but not on node 3's. This is not true of an undirected model. To capture directionality, we compute the forward probabilities [16] instead of the marginal probabilities. For a chain (such as the one shown in Fig. 1) the forward probability of the state of node i is

$$f_i(x_i) = \sum_{x_1, \ldots, x_{i-1}} p(x_1, \ldots, x_{i-1}, x_i), \tag{3}$$

where the joint probability is

$$p(x_1, \ldots, x_i) \propto \prod_{j \in \mathcal{V}: j \leq i} \phi_j(x_j) \prod_{(j,k) \in \mathcal{E}: j \leq i, k \leq i} \psi_{jk}(x_j, x_k). \tag{4}$$

It is clear that the forward probability of node i considers only those nodes that can influence it (e.g., the previous nodes in the chain), which is consistent with the meaning of directed edges. To compute the forward probabilities, we augment the belief propagation algorithm so that messages are only sent from node i to node j if there is an edge $(i, j) \in \mathcal{E}$. We provide further implementation details in Subsect. 2.4.

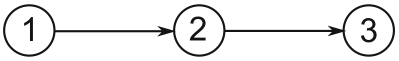


Fig. 1. A directed graph consisting of three nodes. An edge's direction determines the influencer-influencee relationship. In this case, node 2 is influenced by node 1, but not node 3.

We emphasize that an advantage of our approach is that efficient inference algorithms, such as belief propagation, can be used to approximate these forward probabilities [21]. Although exact computation of the forward probabilities is feasible only on graphs without loops, belief propagation is widely used on graphs with loops and generally provides good results [13,15,20].

2.1 Learning

The unary (ϕ_i) and pairwise (ψ_{ij}) potential functions are our model's parameters. The pairwise potential function is shown in Table 1, where p_{ij} is an edge-specific influence probability. If causal information between nodes were available, these could be learned individually. In the absence of good causal information, p_{ij} can be set using a heuristic, such as those based on the degree of a node, or some constant [2,3,11,18]. In our model, we set $p_{ij} = 0.5 + 0.5/\text{in-degree}(j)$. As the in-degree of a node increases, $p_{ij} \to 0.5$. This essentially implies that for any high in-degree node, each influencer exerts less influence on it. The rationale for this function is that when a node has many influencers, each influencer, individually, has a smaller impact on the decision of that node, allowing the node to make a decision based on the aggregate of the states of the influencers. Note that the potential functions express the fact that when $x_i = 0$, node i does not influence node j because both states are equally likely; the lack of adoption does not spread influence. Note that pairwise potential functions are general enough to accommodate other situations, including the case where the lack of adoption could spread influence. For completeness, we also consider the case when p_{ij} is a constant in our experiments.

Each node's unary potential function can be learned from training data, if available. Specifically, the unary potential is any node's adoption probability

Table 1. Pairwise potential functions.

$\psi_{ij}(x_i, x_j)$	$x_j = 0$	$x_j = 1$
$x_i = 0$	0.5	0.5
$x_i = 1$	$1 - p_{ij}$	p_{ij}

Table 2. Unary potential functions.

	$\phi_i(x_i)$
$x_i = 0$	$1 - \rho_i$
$x_i = 1$	ρ_i

independent of all other nodes. Therefore, given any training dataset, we can compute a node's adoption probability as simply the number of historical node adoptions divided by the total number of possible adoptions. Let ρ_i be this probability for node i. If ρ_i is too small, we set it to a minimum value (that is, $\rho_i \geq 10^{-5}$). This allows nodes that are not activated in the training set to still participate in influence propagations in the test set. The unary potential function is shown in Table 2.

The type of data required for training our model is minimal and generally available. In particular, it is far more realistic to assume that we can obtain historical states of the nodes in a network than to capture other higher-level information, such as the causal spread of information from one node to another, as required by the credit assignment model [8]. That is, it is easier to capture *which* nodes are activated than *how* nodes are activated. Nonetheless, in our experiments, we also consider the situation when no training data is available and set ρ_i to a small positive constant.

2.2 Computing Influence Spread

Using our diffusion model, we can compute the influence spread of seed set $\mathcal{S}$. For each seed node $i \in \mathcal{S}$, we set $\phi_i(1) = 1$ and $\phi_i(0) = 0$. We then run our forward belief propagation algorithm to compute $f_i(x_i)$ as defined by (3). Since social networks tend to have loops, belief propagation requires several iterations to converge (we use a maximum of 20 iterations in our experiments). Once converged, or the maximum number of iterations is reached, the influence spread of seed set $\mathcal{S}$ is quantified by

$$\sigma_\Omega(\mathcal{S}) = \sum_{i \in \mathcal{V}} f_i(x_i = 1). \tag{5}$$

2.3 Influence Maximization

The influence maximization problem is to find seed set $\mathcal{S}$ of specified size k that maximizes the influence spread [11]. A greedy approach can be used to

Algorithm 1. Greedy Influence Maximization

```
Input: $G = (\mathcal{V}, \mathcal{E})$, $k$, $\sigma_\Omega$
Output: $\mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
while $|\mathcal{S}| < k$ do
    $u \leftarrow \arg\max_{v \in \mathcal{V} \setminus \mathcal{S}} \sigma_\Omega(\mathcal{S} \cup v) - \sigma_\Omega(\mathcal{S})$
    $\mathcal{S} \leftarrow \mathcal{S} \cup u$
```

Algorithm 2. CELF

```
Input: $G = (\mathcal{V}, \mathcal{E})$, $k$, $\sigma_\Omega$
Output: $\mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
for $u \in \mathcal{V}$ do
    $u.priority \leftarrow \sigma_\Omega(\{u\})$
    $u.n \leftarrow 0$
    enqueue($u$)
while $|\mathcal{S}| < k$ do
    $u \leftarrow$ dequeue()
    if $u.n = |\mathcal{S}|$ then
        $\mathcal{S} \leftarrow \mathcal{S} \cup u$
    else
        $u.priority \leftarrow \sigma_\Omega(S \cup u) - \sigma_\Omega(S)$
        $u.n \leftarrow |\mathcal{S}|$
        enqueue($u$)
```

approximate the influence maximization problem, which is NP-hard in general. The greedy algorithm, taken from [8], is shown in Algorithm 1.

The problem with the greedy approach is that it searches all nodes in the network at each iteration to find the best node. This can be prohibitively expensive, especially if the diffusion model uses Monte Carlo simulations. Several methods have been proposed to improve the greedy algorithm so as to reduce the number of nodes evaluated [3,9,12,22]. In particular, CELF (Cost-Effective Lazy Forward) significantly reduces the number of nodes to evaluate, resulting in 700 times speedup [12]. It uses a priority queue to greedily select the node that has the largest gain in influence spread at each iteration so as to minimize the number of nodes evaluated. In our experiments, we use CELF. For completeness, the CELF algorithm is shown in Algorithm 2. We note that CELF is not the only algorithm that can be used for seed selection. Other algorithms, such as CELF++ [9], can serve as alternatives. We prefer CELF due to its simplicity and find it sufficient, as computing spread on our model via belief propagation is fast.

2.4 Implementation Details

We now briefly describe an implementation of belief propagation and our modification for directed graphs so that the direction of influence is preserved. For a more thorough treatment, see [21].

To solve the marginal probabilities from (2), belief propagation defines a per-edge message from i to j about the likelihood of node j being in state x_j from the perspective of node i:

$$m_{ij}(x_j) = \sum_{x_i} \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \tag{6}$$

where ϕ_i and ψ_{ij} are potentials as defined before, and the final term is the product of all messages sent to i by its neighbors (excluding j). These messages are initialized to a fixed value at all nodes (usually 1). At each iteration, new messages are computed from the previous iteration's messages in both directions along each edge. Iterations continue until either all messages converge to a steady value or a maximum number of iterations is reached.

Once converged, the belief over the states of node i is

$$b_i(x_i) \propto \phi_i(x_i) \prod_{j \in N(i)} m_{ji}(x_i). \tag{7}$$

The normalized belief $b_i(x_i)$ corresponds to the marginal probability $p_i(x_i)$.

The solution to both (6) and (7) may have numerical issues if any node involved has high degree. That is, the product of hundreds or thousands of messages with values between $[0, 1]$ leads to products that are unrepresentable by finite-precision machines. The solution to this problem is to use the well-known log trick.

Since $m_{ij}(x_j)$ can be normalized by an arbitrary positive constant c_{ij} (that is fixed for all values of x_i and x_j on an edge), we can reformulate (6) as

$$\begin{aligned} m_{ij}(x_j) &= \sum_{x_i} c_{ij}\, \phi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \\ &= \sum_{x_i} \exp\left[\ln(c_{ij}) + \ln(\phi_i(x_i)) + \ln(\psi_{ij}(x_i, x_j)) + \sum_{k \in N(i) \setminus j} \ln(m_{ki}(x_i)) \right]. \end{aligned} \tag{8}$$

By exploiting log space, the products become sums, and we avoid numerical underflow during message computation. We can ensure the result is within the representable double-precision range before exponentiation by setting $\ln(c_{ij})$ to an appropriate value. Specifically, we use

$$\ln(c_{ij}) = -\max_{x_i, x_j} \{ \ln(\phi_i(x_i)) + \ln(\psi_{ij}(x_i, x_j)) + \sum_{k \in N(i) \setminus j} \ln(m_{ki}(x_i)) \}. \tag{9}$$

A similar trick is used for computations involving (7).

We found that a graph containing a node with degree greater than 750 would underflow with the default implementation. With the exponentiated version, we have tested up to degree 20,000 with no numerical issues.

Finally, the above belief propagation works on undirected graphs. However, as already described, the influence problem can be directed or asymmetric along edges – that is, i may influence j more than j does i. The only alteration required in the implementation to solve for (3) instead of (2) is to send messages downstream only. The beliefs now correspond to the forward probabilities.

We have implemented the above algorithms and models in Java and integrated them into the open-source Algorithm Foundry package.[1] For a commented example of how to run our code, see the class InfluenceSpread in the GraphExamples Component.

3 Experiments

We demonstrate the utility of our model using two datasets: Flixster [10] and Epinions [19]. The Flixster dataset contains movie reviews with timestamps and a network of friends. The edges are undirected, but to allow asymmetric influence along edges, we convert each edge into two opposite directed edges. The Epinions dataset contains product reviews with timestamps and a directed network of trust among reviewers. The basic statistics of the two datasets are summarized in Table 3.

Table 3. Statistics of the two datasets.

	Flixster	Epinions
# Nodes	800K	18K
# Directed Edges	12M	1.2M
# Products/Movies	49K	262K
Avg. Degree	30	64
Max. Degree	2K	4K

In the following, we use the word propagations to refer to movies in the Flixster dataset and products in the Epinions dataset. For each dataset, we split the propagations into two sets: training (80 %) and test (20 %). As in [8], to ensure a fair distribution of the propagation sizes across the training and test sets, we order all propagations by size and assign every fifth propagation into the test set. The training set is used to learn the unary potential functions, ρ_i.

[1] https://github.com/algorithmfoundry/Foundry/.

3.1 Diffusion Model Validation

We use the test sets to quantify how well our diffusion model predicts actual influence spreads using the method proposed in [8]. Specifically, for a given seed set $\mathcal{S}$, we calculate the predicted spread, $\sigma_\Omega(\mathcal{S})$, using our diffusion model. We can compare our predictions against ground-truth spreads. As in [8], for each propagation in the test set, the seed set is the set of users who are first to review among their immediate friends. The ground-truth spread is the actual number of users who reviewed the propagation.

We consider two strategies for choosing the pairwise potential functions (ψ_{ij}, Table 1):

- **DW**: Degree Weighted – $p_{ij} = 0.5 + 0.5/\text{in-degree}(j)$.
- **CW**: Constant Weight – p_{ij} is set to a constant of 10^{-3}. Note that for the Flixster dataset the weights are symmetric on all edges and the resulting model is undirected.

We consider two strategies for the unary potential functions (ϕ_i, Table 2):

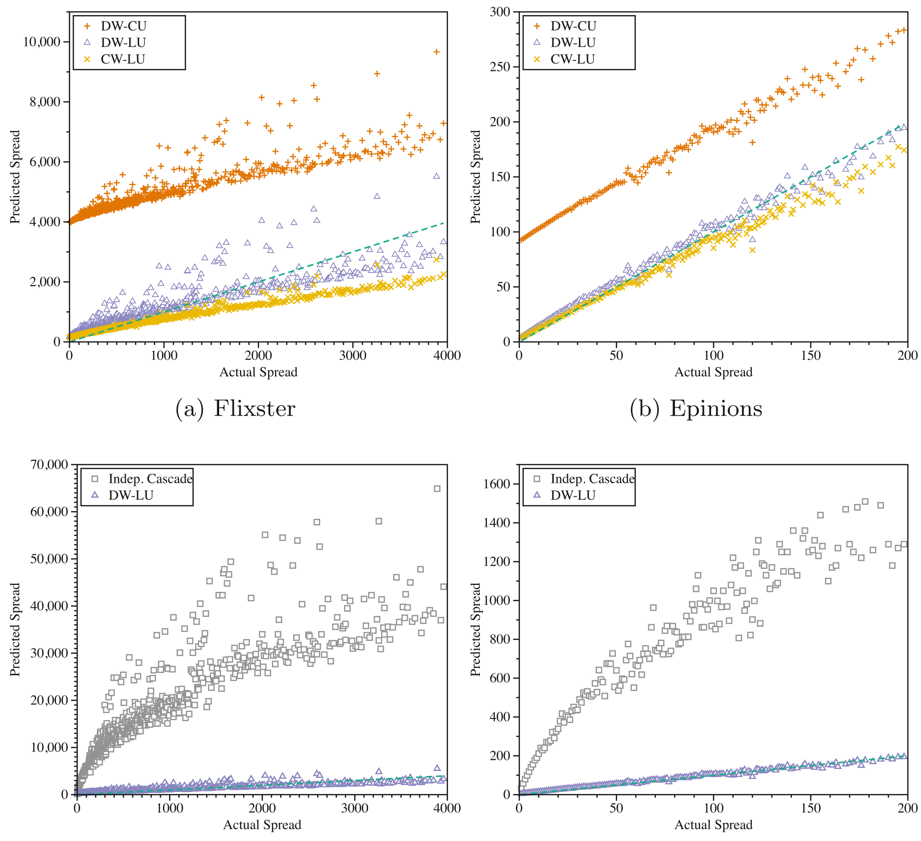
- **LU**: Learned Unary – ρ_i is number of reviewed propagations by node i divided by the total number of propagations in the training set; must be at least 10^{-5}.
- **CU**: Constant Unary – ρ_i is set to a constant (5×10^{-3}).

Thus, for any experiment, we must select both a unary and a pairwise strategy. Hereafter, we refer to a combined strategy as a *pairwise-unary* strategy. As an example, the DW-LU strategy uses the degree weighted pairwise and learned unary strategies. For completeness, we also consider the weighted cascade IC model, a first in itself for the large Flixster graph. For the IC model, we use 10,000 Monte Carlo simulations to compute the spread of each propagation.

We show the scatter plots of the predicted and actual spread of each of these strategies on the test sets in Fig. 2. To improve the readability of the scatter plots, if there are several propagations that have the same actual spread, we report the average predicted spread. The ideal spread is shown as the green dashed line. The CW-LU strategy consistently underestimates the actual spreads. The DW-CU strategy overestimates (or underestimates, depending on the constant ρ_i) the actual spreads. The IC model significantly overestimates the spreads. This is consistent with past observations on smaller networks [8]. The DW-LU strategy performs the best – surrounding the actual spread.

In Fig. 3, we show the same scatter plots of DW-LU along with the corresponding seed sizes. The plots show that this model is able to take the initial seeds and spread their influence to other nodes in the network.

We believe DW-LU performs well on both datasets for two reasons. First, the learned unary strategy is able to incorporate node-specific data. For instance, some social media users are much more likely to produce content than others: incorporating this into the model improves results. This also implies that nodes that tend to adopt products on their own should not be targeted, as resources are better spent on other nodes. Second, as mentioned earlier, the choice of DW for p_{ij} implies that influencers of high in-degree nodes have small impact on their



(a) Flixster (b) Epinions

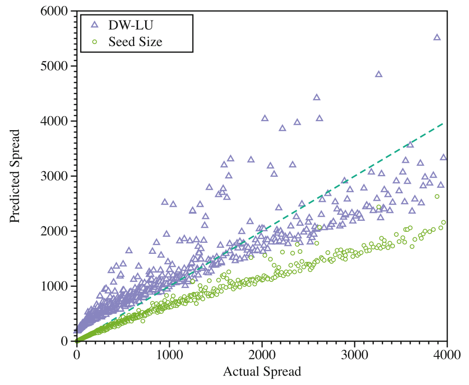
(c) Flixster (d) Epinions

Fig. 2. Scatter plots of predicted spread vs. actual spread for different choices of potential functions: DW-CU (degree weighted and constant unary), CW-LU (constant weight and learned unary), and DW-LU (degree weighted and learned unary), as well as IC for the Flixster and Epinions datasets. The green line shows the ideal predictions. The DW-LU model best predicts the actual spread. Best viewed in color. (Color figure online)

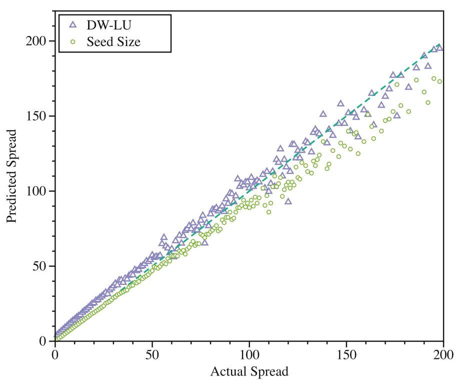
influencees individually, allowing the influencees to make their decisions based on the aggregate of the states of their influencers.

3.2 Influence Maximization

Since our results establish that the DW-LU model is the best in predicting the spread of a seed set, we now investigate the influence maximization problem to determine how much spread is achieved under the DW-LU model on seeds selected by various models, obtained by running CELF on each model as appropriate. In addition, we also investigate the similarity between the seeds selected

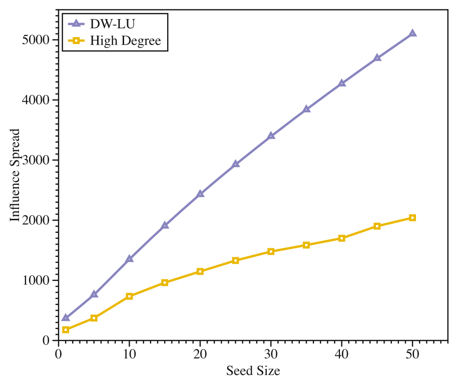


(a) Flixster

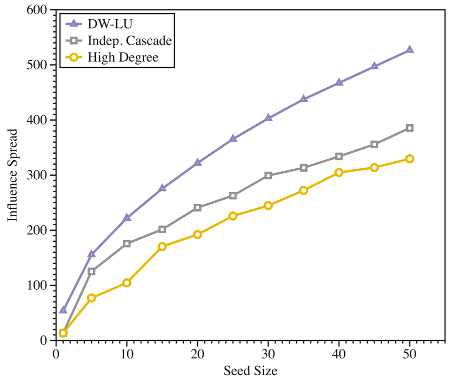


(b) Epinions

Fig. 3. Scatter plots of predicted spread using DW-LU along with the corresponding seed sizes used to spread influence on (a) Flixster and (b) Epinions. The green line shows the ideal predictions. The DW-LU model is able to spread the influence of seed nodes to other nodes. Best viewed in color. (Color figure online)



(a) Flixster



(b) Epinions

Fig. 4. Influence spread achieved under the DW-LU model by seed sets selected by various models: DW-LU, IC, and High Degree on (a) Flixster and (b) Epinions.

by DW-LU and other models. We consider several models: DW-LU, DW-CU, IC, and High Degree which selects the top k nodes as seeds based on degree (CELF is not needed). Since the IC model is computationally demanding, we run it only on the Epinions dataset, which is the smaller of the two datasets. Even then, it takes 22 days to find 50 seeds.

The plot of the influence spreads of seeds selected by DW-LU, IC, and High Degree are shown in Fig. 4. The results show that the seeds identified by our DW-LU model achieve significantly higher influence spread than High Degree and IC.

Table 4. Number of overlapping seeds between DW-LU and other models for various seed sizes.

		10	20	30	40	50
Flixster	High Degree	0	1	3	4	5
	DW-CU	10	20	30	40	49
Epinions	High Degree	3	6	10	16	18
	DW-CU	10	18	29	36	45
	IC	6	9	14	18	24

Table 4 shows the number of overlapping seeds between DW-LU and the other models. It is clear that the seeds selected by High Degree have low overlap with DW-LU. Even with $k = 50$ seeds, the overlap between High Degree and DW-LU is only 5 for the Flixster dataset and only 18 for the Epinions dataset. For the Epinions dataset, with 50 seeds, the number of overlapping seeds between DW-LU and IC is 24. The DW-CU and DW-LU models have almost identical seeds, which is why we did not plot DW-CU in Fig. 4 as those two curves are on top of each other. An important consequence of this result is that the social network analyst can leverage the DW-CU model to identify influential seed nodes in the absence of any training data. This is significant as training data may not be available in some applications.

We examine various graph metrics for the seeds selected by DW-LU, IC, and High Degree.

- **Community Overlap**: We run Louvain community detection [1] on the Flixster and Epinions graphs to identify the community assignment for each of the identified seeds. Since we have 50 seeds and between 15 and 25 communities identified on each graph, there is some overlap in community assignment for seed nodes. However, the High Degree technique selects far more nodes from its two most common communities (24 and 10 on Flixster; 30 and 9 on Epinions) than our DW-LU model (12 and 9 on Flixster; 20 and 14 on Epinions). On Epinions, IC is approximately the same as DW-LU (19 and 14).
- **Average Distance**: We compute each seed's average distance to all other seeds using Dijkstra's Algorithm [4]. In both graphs, our DW-LU model selects nodes that are farther apart on average (2.38 vs. 2.14 edges apart on Flixster; 1.82 vs. 1.39 edges apart on Epinions). On Epinions, IC averages 1.58–further than degree, but closer than DW-LU.
- **Node Degree**: We investigate the degrees of seed nodes in the order selected by CELF. Although the degree of each seed selected by IC and DW-LU varies from the degrees of the seeds selected before or after it, when we fit a line to the seeds' degrees, there is a clear negative trend. The degrees of the seeds are mostly well above the average degree on both graphs (one of the seeds selected by DW-LU for Epinions is just below the graph-wide average degree). IC consistently selects higher degree nodes than DW-LU.

These results indicate several interesting features for effective seed nodes. First, while high degree seems to be a useful feature for a seed node (nearly all DW-LU seeds have high degree), it is not sufficient (High Degree achieves lower influence spread and IC's higher degree nodes achieve lower influence as shown in Fig. 4). The community overlap and average distance measures indicate a second critical feature: the best seeds spread out from each other. Note that the most spread out set of nodes are among the leaves, but those nodes do not have a high enough degree to spread influence. Thus, there must be a balance between spread and high degree. Both IC and DW-LU balance these two features, although DW-LU balances them better.

3.3 Computing Resources

Our model uses an augmented belief propagation algorithm to compute influence spread. The runtime and memory requirement are both bounded by $O(|\mathcal{E}| + |\mathcal{V}|)$. The quantities provided here are relevant to the Flixster dataset, which is the larger of the two. Our Java implementation uses 3.4 GB of memory. As for computing time, the most expensive operation in influence maximization is the computation of influence spread of each node as a seed node, which is required by CELF. For this, we use a compute cluster of 60 compute nodes to run our model, which took 12 h to complete. Note that we do this only once. Once done, we select the top 50 nodes using CELF on a single workstation. The total time to find the top 50 nodes is approximately 16 min. On average, propagating each seed set takes 4 s. In contrast, using 10,000 Monte Carlo simulations to compute the spread of each seed set under the IC model takes 6 min on average.

4 Discussion and Conclusion

Influence maximization is a relevant and important problem in social network analysis. As such, it is important to have models that are efficient, provide a certain level of validation against ground-truth data, and can be learned from readily available data. To this end, we have presented a model that addresses these concerns. Our model uses belief propagation instead of Monte Carlo simulations to compute influence spread. Our model parameters can be learned from basic training data, such as frequency of adoptions, which we believe is more readily available in practical applications than other models that require causal relationships. In the absence of training data, our model can still identify influential seeds. As mentioned earlier, we use a heuristic based on in-degree to set the pairwise potentials. Our model, however, is general enough that these pairwise functions can be set to arbitrary values, including those learned from a training dataset, if available.

The results of this work raise an important question: what intrinsic graph properties are important in identifying influential seeds? As we have seen, high degree alone is not sufficient. Yet, our model, using pairwise functions that are based on in-degree, identifies seed nodes that achieve high influence spread. Are seed nodes intrinsic to graph structures? We hope to provide further insight into these questions in our future work.

Acknowledgment. We are grateful to Cristopher Moore for discussions on belief propagation and implementation considerations, Rich Field for improving the quality of the paper, and Dave Zage for discussions on implementation considerations. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp. **10**, P10008 (2008)
2. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1029–1038. ACM (2010)
3. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 199–208 (2009)
4. Dijkstra, E.W.: A note on two problems in connection with graphs. Numer. Math. **1**, 269–271 (1959)
5. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 57–66. ACM (2001)
6. Even-Dar, E., Shapira, A.: A note on maximizing the spread of influence in social networks. Inf. Proces. Lett. **111**(4), 184–187 (2011)
7. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 241–250. ACM (2010)
8. Goyal, A., Bonchi, F., Lakshmanan, L.V.: A data-based approach to social influence maximization. Proc. VLDB Endow. **5**, 73–84 (2011)
9. Goyal, A., Lu, W., Lakshmanan, L.V.: CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th International Conference Companion on World Wide Web, pp. 47–48. ACM (2011)
10. Jamali, M., Ester, M.: A matrix factorization technique with trust propagation for recommendation in social networks. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 135–142. ACM (2010)
11. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146. ACM (2003)
12. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 420–429. ACM (2007)

13. Mooij, J.M., Kappen, H.J.: Sufficient conditions for convergence of the sum-product algorithm. IEEE Trans. Inf. Theory **53**(12), 4422–4437 (2007)
14. Nguyen, H., Zheng, R.: Influence spread in large-scale social networks – a belief propagation approach. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 515–530. Springer, Heidelberg (2012)
15. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Burlington (2014)
16. Rao, V., Teh, Y.W.: Fast MCMC sampling for Markov jump processes and extensions. J. Mach. Learn. Res. **14**(1), 3295–3320 (2013)
17. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 61–70. ACM (2002)
18. Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 67–75. Springer, Heidelberg (2008)
19. Tang, J., Gao, H., Liu, H., Sarma, A.D.: eTrust: understanding trust evolution in an online world. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 253–261. ACM (2012)
20. Weiss, Y.: Correctness of local probability propagation in graphical models with loops. Neural Comput. **12**(1), 1–41 (2000). http://dx.doi.org/10.1162/089976600300015880
21. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Understanding belief propagation and its generalizations. Technical report. TR2001-22, Mitsubishi Electric Research Laboratories, November 2001
22. Zhou, C., Zhang, P., Zang, W., Guo, L.: On the upper bounds of spread for greedy algorithms in social network influence maximization. IEEE Trans. Knowl. Data Eng. **27**(10), 2770–2783 (2015)