Influence Maximization in Social Networks

Project Presentation

Oscar Aguilar, Louis Gautier, Eva Zhang

Stanford University

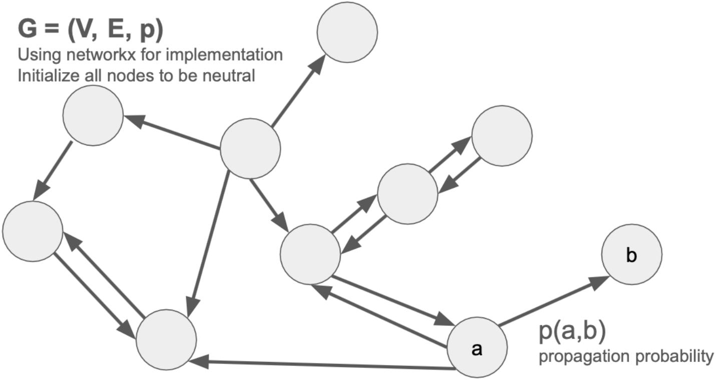
December 12, 2023

Independent Cascade Model with Negative Opinions

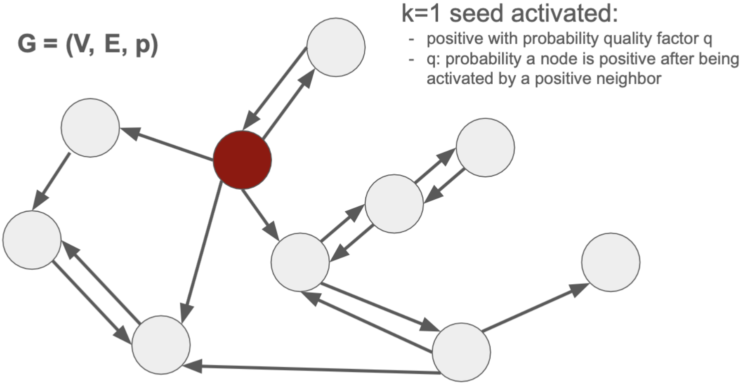
- Paper "Influence Maximization in Social Networks When Negative Opinions May Emerge and Propagate" by Wei Chen et al.
  - Influence maximization defined by Kempe, Kleinberg, and Tardos (2003)
  - First in influence maximization to consider negative opinions
  - Incorporates negativity bias from social psychology

Independent Cascade Model with Negative Opinions

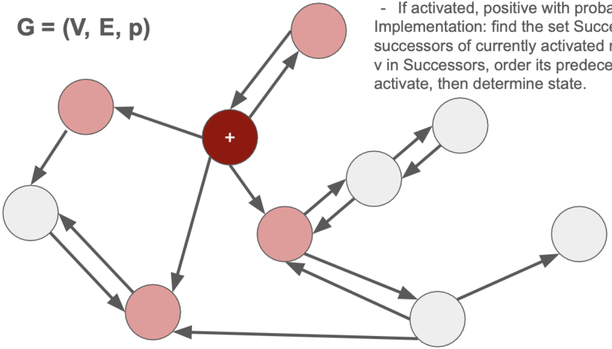
- IC-N model (Independent Cascade Model with Negative Opinions)



Independent Cascade Model with Negative Opinions



Independent Cascade Model with Negative Opinions

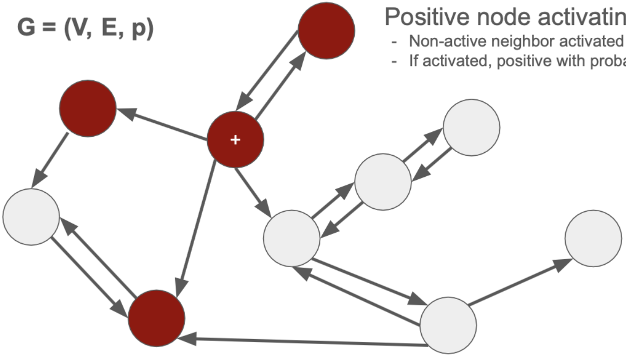


Positive node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, positive with probability q

Implementation: find the set Successors, neutral successors of currently activated nodes. For each node v in Successors, order its predecessors randomly and activate, then determine state.

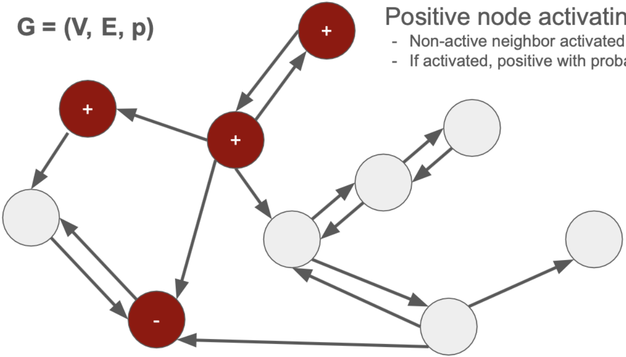
Independent Cascade Model with Negative Opinions



Positive node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, positive with probability q

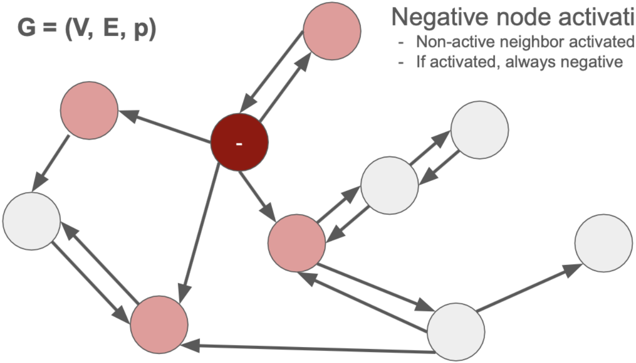
Independent Cascade Model with Negative Opinions



Positive node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, positive with probability q

Independent Cascade Model with Negative Opinions

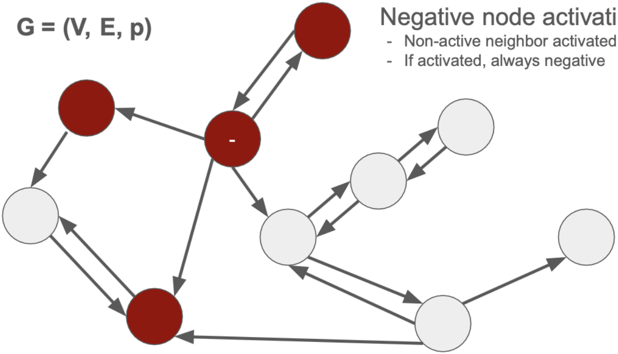


Negative node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, always negative

Independent Cascade Model with Negative Opinions

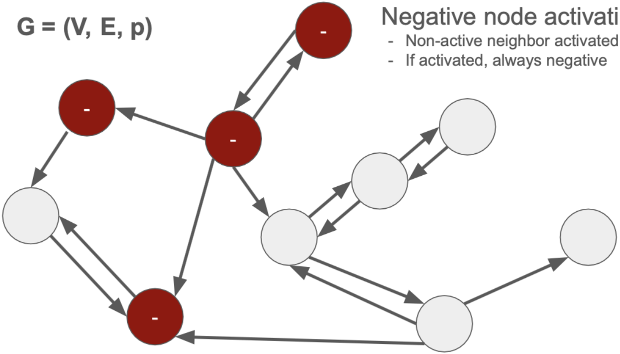
G = (V, E, p)



Negative node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, always negative

Independent Cascade Model with Negative Opinions



Negative node activating a neighbor:

- Non-active neighbor activated with probability p
- If activated, always negative

Simulation-based Greedy Algorithm

Algorithm 1 Greedy(k, f)

```
initialize S = ∅
for i = 1 to k do
    select u = arg max_{w∈V\S} (f(S ∪ {w}) − f(S))
    S = S ∪ {u}
end for
output S
```

Greedy algorithm, where f is a Monte Carlo simulation of the IC-N model with seed set S

MIA-N algorithm

Goal: Develop a local search heuristic that avoids evaluating the influence of each node on the whole graph at each iteration like in the greedy algorithm.

- Use arborescences to approximate local influence regions of every node in the graph
- Use efficient methods to compute the positive influence spread in these arborescences.

Computing influence in arborescences

Computational Complexity: Computing influence spread in general for IC model is $\#$*P-hard*

Positive Influence Spread: $\sigma_A(S, q) = \sum_u pap(u)$

Positive Activation Prob: $pap(u) = 0$ if $path(u) = \emptyset$, else

$$pap(u) = \prod_{e \in E(path(u))} p(e) \cdot q^{\|path(u)\|+1}$$

where $path(u)$ is the path from seed s in S to u in out-arborescence A

Insight: We can compute the positive influence spread in one traversal of A.

Dynamic Program to estimate influence

Idea: Recursively compute prob. each node unactivated at t

$$ap(u,t) = \begin{cases} 1 & \text{if } t = 0 \wedge u \in S \\ 0 & \text{if } t = 0 \wedge u \notin S \\ 0 & \text{if } t > 0 \wedge u \in S \\ \displaystyle\prod_{w \in N^{in}(u)} [1 - \sum_{i=0}^{t-2} ap(w,i)p(w,u)) \\ \quad \displaystyle - \prod_{w \in N^{in}(u)} [1 - \sum_{i=0}^{t-1} ap(w,i)p(w,u))] & \text{if } t > 0 \wedge u \notin S \end{cases}$$

$$\sigma_A(S,q) = \sum_{u \in V} pap(u) = \sum_{u \in V} \sum_{t \geq 0} pap(u,t) = \sum_{u \in V} \sum_{t \geq 0} ap(u,t) \cdot q^{t+1}$$

Efficiency bump: Flatten out the recursion of $pap(u)$

MIA-N algorithm

Maximum Influence In (resp. Out)-Arborescences

$$MIIA(v, q, \theta) = \bigcup_{u \in V, ppp(MIP(u,v)) \geq \theta} MIP(u, v)$$

$$MIOA(v, q, \theta) = \bigcup_{u \in V, ppp(MIP(v,u)) \geq \theta} MIP(v, u)$$

where $MIP(u, v)$ is the path between u and v with highest positive propagation probability: $ppp(P) = \prod_{i=1}^{m-1} p(p_i, p_{i+1}) q^m$

Critical assumption: The influence from S to v is only propagated through $MIIA(v, q, \theta)$: the positive influence spread of S is $\mu(S, q) = \sum_{u \in V} pap(v, S, MIIA(v, q, \theta), q)$.

Efficiency improvement: At each step, we only need to update the incremental influence spread of nodes $w \in MIIA(v, q, \theta)$, for $v \in MIOA(u, q, \theta)$.

Our MIA-N implementation

- Fully implemented the algorithm in Python with compatibility with NetworkX
- MIIAs and MIOAs computations: Uses Dijkstra algorithm to compute all shortest paths in a graph with modified edge weights
- Completely generic to the input graph and with user-friendly interaction and convenient results evaluation

```
Number of nodes: 500
Number of edges: 2141
Average in-degree: 4.282
Average out-degree: 4.282
Initializing MIAN
Computing all shortest paths
Completed computing all shortest paths
Computing initial MIIA and MIOA
Computing initial PAP
k: [====================] 100% | v: [====================] 100%
```

Example log from the implementation

MIA-N vs Simulation: Analysis

Complexity

Given a social network with n nodes and m edges:

- MIA-N algorithm: $\mathcal{O}(kn_o n_i^2 h)$ iterations, with $h, n_o, n_i \ll n$
- Greedy simulation-based algorithm: $\mathcal{O}(knMm)$, where M is the number of times to run the Monte Carlo simulation

In practice, huge difference of run time on large graphs

Robustness

- Sensitivity of MIA-N to the hyperparameter θ and to the structure of the network
- Sensitivity of the simulation-based algorithm to M

Datasets

Epinions dataset

- Graph representing a consumer review website
- Nodes: Users ($|V| = 75,879$)
- Directed Edges: Trust relationships ($|E| = 508,837$)

NetHPT dataset

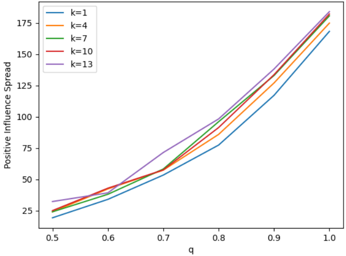
- High Energy Physics - Theory collaboration network
- Nodes: Researchers ($|V| = 9,877$)
- Edges: Co-authorship links ($|E| = 25,598$)

Random Walk Subgraph

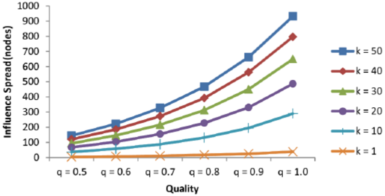
The sizes of Epinions and NetHPT were too much for the scope of the project. Thus, random walk subgraphs were constructed:

- Start at a random node
- Until the desired subgraph size is reached:
  - Choose a random neighbor to add to the subgraph
  - If there are no neighbors, check the size of current subgraph. If size < 10, random node is selected from the entire graph to restart the sampling process. If size ≥ 10, it restarts by selecting a random node from the current subgraph.

Results: Positive Influence Spread vs q

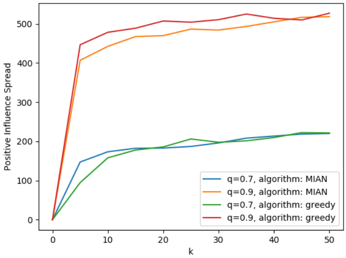


(a) Our implementation tested on a 200-node subgraph of the Epinions dataset

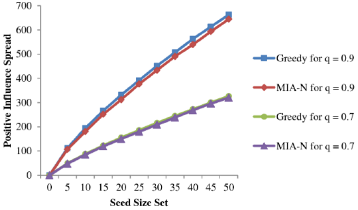


(b) Paper on the full NetHPT graph

Results: Influence Spread vs k, Greedy and MIA-N



(a) On a 500-node subgraph of the NetHPT graph



(b) Paper on the full NetHPT graph