Information Spread and Influence Maximization in Social Networks

(Using Star Wars social network dataset)

Muhammad Altaf Agowun (A00448118)

ST: Complex Networks CSCI 4834

Somayeh Kafaie

2022/12/07

Table of Contents

Abstract

With the increase of daily active social media users and influencers, influence marketing has become increasingly impactful. Choosing the best influencer/s is key in running a successful influencer marketing campaign. However, that task is hard to do especially with how large the top social media platforms have become. Two of the most common diffusion models to help solve the issue of choosing the influencer/s are Independent Cascade and Linear Threshold, however these two diffusion models can take a long time to complete and are limited to how complex predicting whether a node will be influenced can be, since it will add to their already long runtime. To address these issues a new diffusion model is proposed, namely Stored Diffusion. The new diffusion model had considerably faster runtime and excluded the network specific determination of which nodes would be influence by which active node from the main simulation runtime, hence the algorithm use for determining the influence of a node can be as complex as needed, without impeding the runtime of the simulation.

1. Introduction

Modelling the spread of information and determining the influence of people or group of people in a social network has benefits for many fields especially in business for viral marketing or even in medical field by adapting the model to disease spreading and assessing cascading failures within complex systems. This report explores two of the most common diffusion models used to model influence in a network namely, Independent Cascade (IC) and Linear Threshold (LT), and two intuitive algorithms used to demonstrate how these models works. Finally, an algorithm storing the influence of individual nodes is proposed for simulating Independent Cascade (SD) in less time.

One of the main uses of solving Influence Maximization problem is to run marketing campaign which would utilize word-of-mouth to gain in popularity and raise the awareness of the product or service being advertised. One of the ways of achieving such a type of marketing is by advertising on social medias such as Facebook, Instagram and Twitter using one or more users known as influencers, however choosing the best influencer/s can be difficult since there are so many influencers and billion of users to consider. Using one of the common measures of centrality in network to select the best nodes does not provide the best outcome as shown at the beginning of the result section.

2. Methodology

2.1 Independent Cascade (IC)

In this diffusion model, nodes within the graphs can be in one of two states, Active or Inactive. A node is Active if they have been influenced and Inactive otherwise. At the start all the nodes are Inactive other than those who have been chosen to influence the network (seed set). Then in each iteration, the newly Active nodes n (nodes that were influence in the previous iteration) will try to influence the Inactive nodes v that are part of their neighbours using a probability, p(n, v). The probability p is different for each network and hence need to be determined by the user.

Pseudocode (independent cascade at time t to influence node for time t + 1)

```
iteration time = t
current_infectious = nodes that have being infected at time t
p(s,t) = probability that the infected (Active) node s will infect the Inactive node t
for n in current_infectious:
    for v in neighbours(n):
      if v not infected:
        if p(n,v) == true:
          v is infected for time t+1
```

2.2 Linear Threshold (LT)

In this diffusion model, nodes within the graphs can be in one of two states, Active or Inactive. A node is Active if they have been influenced and Inactive otherwise. At the start all the nodes are Inactive other than those who have been chosen to influence the network (seed set). Then in each iteration, all the active nodes p will add some tendency to be influence to their neighbour nodes v, the amount of tendency to be influence is determine by p(n,v), if the sum of all the influence is greater than their threshold amount to be influence t(v), then they are marked as influenced and will in turn try to influence their neighbour nodes in the next iteration.

The influence p(n, v) is different for each network and hence need to be determined by the user. Similarly, the threshold t(v) needs to also be determined by the user for the network being built, the threshold is usually set to a uniform random number across all the nodes.

Pseudocode (linear threshold at time t to influence node for time t + 1)

```
iteration time = t
current_infectious = nodes that have being infected at time t
p(s,t) = influence that the infected (Active) node s has on the Inactive node t
initialise threshold(s) for each node s to a uniform random number
initialise influence(s) amount for each node s to 0
for n in current_infectious:
    for v in neighbours(n):
      if v not infected:
         influence(v) += p(n,v)
         if influence(v) >= threshold(s):
            infection_times[v] = t+1
```

2.3 Stored Diffusion (SD)

In this algorithm we first loop through the nodes and calculate the influence of each node on their neighbours, the outcome is then stored to be used in the next step. Once the nodes influences have been calculated and stored, we carry out the simulation but instead of simulating the potential influence of each node on their neighbours during each time t in the simulation, we instead use the influence calculated and stored in the previous step. Hence, we mark the nodes predicted to get influence in the first step as influenced for the time t of the simulation.

<u>Pseudocode for the first step of Stored Diffusion (calculating and storing the influence of each node)</u>

```
Initialize Global array nodes_influence

For n in graph.nodes:
    nodes_influence.append({n, influence(n)})

Func influence(n):
    // determine the nodes that will be influenced by the node n
    return [list of influenced nodes]
```

<u>Pseudocode (Stored Diffusion at time t to influence node for time t + 1)</u>

```
iteration time = t
current_infectious = nodes that have being infected at time t

for n in current_infectious:
    for v in nodes_influence[n]:
      if v not infected:
        v is infected for time t+1
```

3. Results/findings

The performance presented in this section is based on a fictional social network created using the Star Wars franchise and consists of 110 nodes and 444 edges [2]. The nodes represent characters in the franchise and the edges represents the two nodes being seen in the same scene, the weight being the number of scenes they can be seen together. The algorithm was run on google Colab.

3.1 Using common measures of centrality for influence maximization

Using the top nodes from measures of centrality and importance of nodes such as Degree, Weighted Degree, page rank and betweenness centrality the below virality rate* were obtained, see Appendix A.

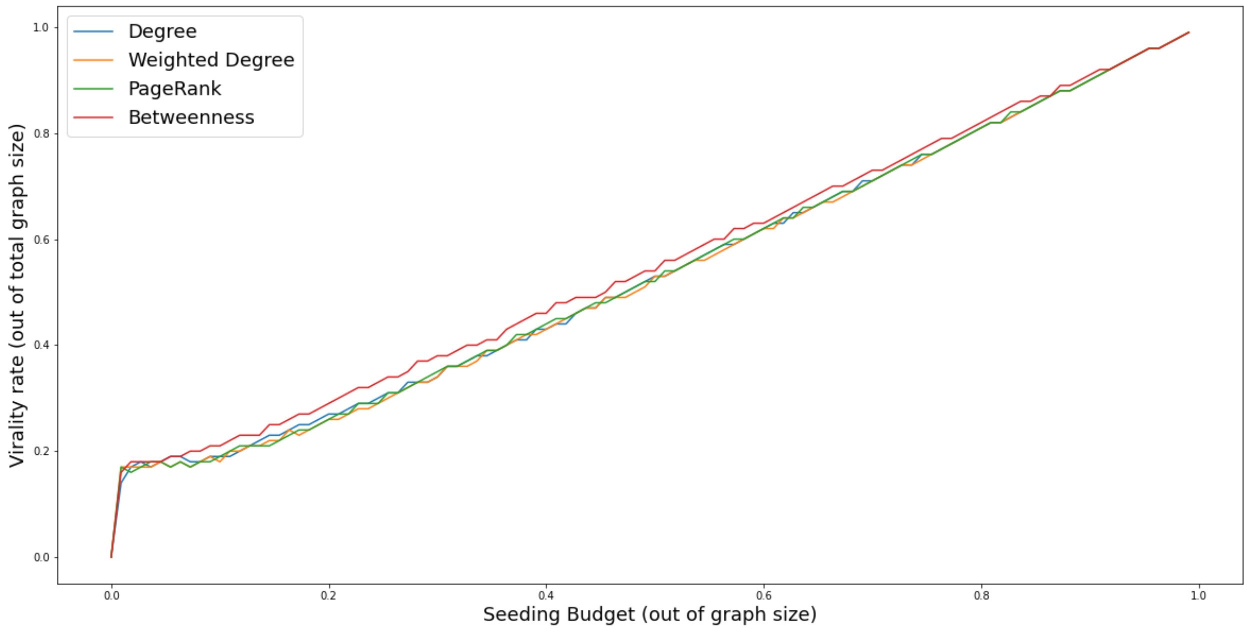


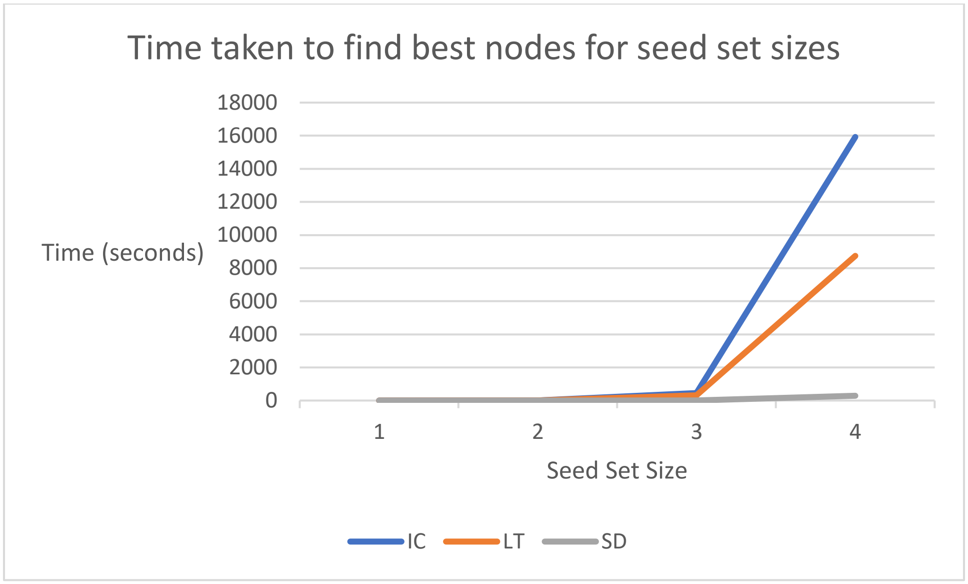
Figure 1 showing the virality rate for different seed set sizes (using the top nodes in the above-mentioned centrality measures)*

* Virality rate: the average percentage of the graph that was influence across the trials for each seed set size.

3.2 Using Diffusion models for influence maximization

The influence predicted by the methods were approximately the same across the different approaches, see Appendix B, however it is important to note that the algorithm for Independent Cascade does not provide the same outcome every time.

Finding the best seed set using influence maximization algorithms usually takes a lot of time since it contains combination based on the number of nodes N and the seed set size S, $C_{N,S}$. Using Independent cascade model with the brute force approach took the most amount of time since it has to do a few trials for each seed sets (number of trials used 15), and the runtime performance obtained was 0, 9, 443 and 15925 seconds for seed sets of 1, 2, 3 and 4 respectively. Linear Threshold using a brute force approach took the second most amount of time even with only one trial 0, 9, 312 and 8741 seconds for seed sets of 1, 2, 3 and 4 respectively. Finally, the Stored Diffusion algorithm took the least with 0, 0, 10 and 289 seconds for seed sets of 1, 2, 3 and 4 respectively, however it took the most amount of memory storage since it has to store the influence of the nodes, 4.6015625 MB and will take time upfront to create and store the influence capabilities of the nodes*.



* It took 0 seconds for our small network. However, we expect the process to take O(n^2) where n is the number of nodes in the network, since it has to loop through all the nodes and try to influence their respective neighbours.

4. Discussion

The output that was received from the influence maximization algorithm are only “technically elegant” since the simulation is only based on little information about how the nodes influence each other, the only information used was the number of scenes that the nodes were seen together and making the assumption that being in the same scene would mean that they can influence each other. For example, in this fictional data above, Darth Vader (the main enemy in the Star Wars franchise) was being influence by OBI-WAN (who is a good character in Star Wars) this would most likely not happen in the fictional world, however it happens in our simulation.

Diffusion models are better than common measures of centralities mainly because they can be designed to account for reasons why a node would be influenced based on studies. The two most common methods Independent Cascade and Linear Threshold are both problems that takes a lot of time to solve although algorithms have been created to find approximates such as Stop and Stare [3] to reduce the time taken by “up to 1200 times faster than the SIGMOD’15 best method”.

One of the advantages of the SD is that it can be updated over time and once updated can be used by the algorithm to run a simulation. Moreover, each node’s influence can be calculated independently thus multi-processing or multi-threading could be implemented as well as the algorithm used to determine the influence can be complex since it is calculated outside of the simulation.

5. Conclusion/recommendations

Further work is still required to develop even faster algorithm but more importantly a more accurate approach to determining the best seed set, since the SD approach proposed still takes a lot of time and storage space.

SD can use different definition and approach to determining whether a neighbour will be influence or not. This is up to the developer of the network to define how to determine whether a node is influenced or not. Determining the best definition of influence is network dependent and still requires further research, for example we can define a better diffusion model for social media such as Instagram by using information gathered by the application such as followers, likes and others information, such as presented in the article “Influence Maximization Diffusion Models Based On Engagement and Activeness on Instagram” by Kristo Radion Purba; David Asirvatham; Raja Kumar Murugesan [4].

References

[1] Report: https://github.com/dimgold/pycon_social_networkx/blob/master/info_spread_got.ipynb

[2] Star Wars social network dataset: https://www.kaggle.com/datasets/ruchi798/star-wars?select=starwars-full-interactions-allCharacters-merged.json

[3] Stop-and-Stare: Optimal Sampling Algorithms for Viral Marketing in Billion-scale Networks by Hung T.Nguyen, My T. Thai, Thang N. Dinh: https://doi.org/10.48550/arXiv.1605.07990

[4] Influence Maximization Diffusion Models Based On Engagement and Activeness on Instagram by Kristo Radion Purba; David Asirvatham; Raja Kumar Murugesan: https://doi.org/10.1016/j.jksuci.2020.09.012

Appendices

Appendix A (Influence of top seed sets based on common measures of centrality)

Seed Set Size	Degree	Weighted Degree	PageRank	Betweenness
1	0.16	0.18	0.16	0.15
2	0.18	0.17	0.17	0.17
3	0.18	0.17	0.18	0.18
4	0.18	0.17	0.18	0.18

Appendix B (influence of top seed sets)

Seed Set Size	IC	LT	SD
1	0.18	0.18	0.21
2	0.22	0.20	0.23
3	0.22	0.21	0.25
4	0.25	0.22	0.25