

A Report For Influence Maximazation Problem

1. Preliminaries

Inuence maximization is the problem of nding a small subset of nodes (seed nodes) in a social network that could maximize the spread of inuence. Recently many large-scale online social network sites, such as Facebook and Friendster, become successful because they are very effective tools in connecting people and bringing small and discon- nected ofine social networks together. Moreover, they are also becoming a huge dissemination and marketing platform, allowing information and ideas to inuence a large population in a short pe- riod of time. However, to fully utilize these social networks as mar- keting and information dissemination platforms, many challenges have to be met. [1]

1.1. Software

This project is written in Python using IDE PyCharm. The libraries being used includes NumPy, sys, time, random, agrparse

1.2. Algorithm

The algorithm being used in this project includes Degree Discount Algorithm and Cost Effective Lazy Forward(CELF).

2. Methodology

The general idea is as follows. let v be a neighbor of vertex u . If u has been selected as a seed, then when considering selecting v as a new seed based on its degree, we should not count the edge \overline{vu} towards its degree. Thus we should discount v 's degree by one due to the presence of u in the seed set, and we do the same discount on v 's degree for every neighbor of v that is already in the seed set.

2.1. Representation

This project contains some main data need to be maintain during the process:

- nodes:defaultdict(dict):to store information
- nodes_amount: the nodes amount
- edges_amount :the edge amount
- nodes_set: the set of all nodes which have out-degree
- start_time:the time program start
- end_time = -sys.maxint:the time program halt

2.2. Architecture

Here list functions in the Python file ISE.py with their usage.

- *open_network_file(filename)*: read the network file.
- *open_seed_file(filename)*:open the seed set file
- *ise_ic_sample(amount)*:estimate ic sample
- *ise_lt_sample(amount)*: estimate lt sample
- *ise_parse_command_line()*: read command

Here list functions in the Python file IMP.py with their usage.

- *open_network_file(filename)*: read the network file.
- *ic_model(size, p)*:degree discount
- *lazy_forward(k,model_type,sample_type)*::lazy_forward algorithm
- *ise_ic_sample(amount)*:estimate ic sample
- *ise_lt_sample(amount)*: estimate lt sample
- *imp_parse_command_line()*: read command

The last of the Python file, use *main()* to test all the codes.

2.3. Detail of Algorithm

Degree Discount Algorithm performs well to solve big data problem.

Algorithm 1 Degree Discount[1]

```
1: initialize  $S = \emptyset$ 
2: for each vertex  $v$  do
3:   compute its degree  $d_v$ 
4:    $dd_v = d_v$ 
5:   initialize  $t_v$  to 0
6: end for
7: for  $i = 1$  to  $k$  do
8:   select  $u = \arg \max_v \{dd_v \mid v \in V \setminus S\}$ 
9:    $S = S \cup \{u\}$ 
10:  for each neighbor  $v$  of  $u$  and  $v \in V \setminus S$  do
11:     $t_v = t_v + 1$ 
12:     $dd_v = d_v - 2t_v - (d_v - t_v)t_v p$ 
13:  end for
14: end for
15: output  $S$ 
```

I made some changes base on this algorithm, because this algorithm used to solve undirected graph. d_v means the out degree, t_v means the amount of v 's neighbours which

to

$$dd_v = d_v - 2t_v - (d_v - tv)t_v p \quad (1)$$

$$dd_v = 2t_v + (d_v - tv)t_v p[2] \quad (2)$$

I also try to change the heuristic amount, I want to make the all the nodes the v can reach, but it doesn't work well.

```
def count_dv(node):
    has_count = num.zeros(nodes_amount+1)
    queue = list()
    count = 0
    queue.append(node)
    while queue:
        node = queue.pop(0)
        has_count[node] = 1
        for i in nodes[node]:
            if has_count[i] == 0:
                queue.append(i)
            count += 1
    return count
```

The CELF is an improved algorithm for traditional greedy algorithm. It do not search for every change, it only change the biggest value.

Algorithm 2 CELF[3]

```

Function: LazyForward( $\mathcal{G} = (\mathcal{V}, \mathcal{E}), R, c, B, type$ )
 $\mathcal{A} \leftarrow \emptyset$ ; foreach  $s \in \mathcal{V}$  do  $\delta_s \leftarrow +\infty$ ;
while  $\exists s \in \mathcal{V} \setminus \mathcal{A} : c(\mathcal{A} \cup \{s\}) \leq B$  do
    foreach  $s \in \mathcal{V} \setminus \mathcal{A}$  do  $cur_s \leftarrow \text{false}$ ;
    while true do
        if  $type = UC$  then  $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \mathcal{A}, c(\mathcal{A} \cup \{s\}) \leq B}{\operatorname{argmax}} \delta_s$ ;
        if  $type = CB$  then  $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \mathcal{A}, c(\mathcal{A} \cup \{s\}) \leq B}{\operatorname{argmax}} \frac{\delta_s}{c(s)}$ ;
        if  $cur_{s^*}$  then  $\mathcal{A} \leftarrow \mathcal{A} \cup s^*$ ; break;
        else  $\delta_{s^*} \leftarrow R(\mathcal{A} \cup \{s^*\}) - R(\mathcal{A})$ ;  $cur_{s^*} \leftarrow \text{true}$ ;
    return  $\mathcal{A}$ ;

```

```

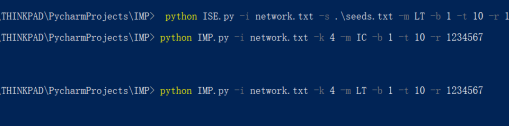
Algorithm: CELF( $\mathcal{G} = (\mathcal{V}, \mathcal{E}), R, c, B$ )
 $\mathcal{A}_{UC} \leftarrow \text{LazyForward}(\mathcal{G}, R, c, B, UC);$ 
 $\mathcal{A}_{CB} \leftarrow \text{LazyForward}(\mathcal{G}, R, c, B, CB);$ 
return  $\text{argmax}\{R(\mathcal{A}_{UC}), R(\mathcal{A}_{CB})\}$ 

```

3. Empirical Verification

Empirical verification is compared with the given ans in the dat file:

. result



```
PS C:\Users\THINKPAD\PycharmProjects\IMP> python ISE.py -i network.txt -s .\seeds.txt -m IC -b 1 -t 10 -r 1234567
5,0144
PS C:\Users\THINKPAD\PycharmProjects\IMP> python ISE.py -i network.txt -s .\seeds.txt -m LT -b 1 -t 10 -r 1234567
5,0263
PS C:\Users\THINKPAD\PycharmProjects\IMP> python IMP.py -i network.txt -k 4 -m IC -b 1 -t 10 -r 1234567
56
57
58
59
60
61
62
63
64
65
66 C:\Users\THINKPAD\PycharmProjects\IMP> python IMP.py -i network.txt -k 4 -m LT -b 1 -t 10 -r 1234567
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
PS C:\Users\THINKPAD\PycharmProjects\IMP>
```

Acknowledgments

I would like to thank Zhou,Ni, Li,Zheng for exchanged some thoughts with me on some vital algorithms although I did not successfully finished this project.

References

- [1] Wei Chen, Yajun Wang, and Siyu Yang. 2009. Efficient influence maximization in social networks. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09). ACM, New York, NY, USA, 199-208.
- [2] K. Sivasailam, V. K. Sebastian, D. M. Jacob and S. Bhattacharya, "Discount heuristics and heterogeneous probabilities for optimal influence in social networks," 2014 6th International Conference on Computational Aspects of Social Networks, Porto, 2014, pp. 13-18.
- [3] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA, 420-429.