Influence Maximization Problem

Keyuan Lu

December 2017

1 Preliminaries

Influence maximization problem is the problem of finding a small subset of nodes(seed nodes) in a social network to maximize the spread of influence.

1.1 Software

This project is written in Python using IDE Pycharm. The library being used is Numpy.

1.2 Algorithm

The algorithms in this project includes DFS search. The method in this project includes recursion. There are seven functions in this project

2 Methodology

This part includes representation, architecture and details of the algorithms

2.1 Representation

This project includes two models of influence maximization, **Independent Cascade Model**(IC model) and **Linear Threshold Model**(LT Model). The **DegreeDiscountIC** is an algorithm to calculate the best seed set in the given graph. Input a graph of the social network, the size of seed set k and the time to calculate in the two models n, the output is the k-sized seed set and the average number of people influenced after n times of calculation in both models.

2.2 Architecture

Here is the list of the functions in this project:

- **input()**
- **find-all-neighbors()**
- **find-inactive-neighbors**
- **find-active-neighbors()**
- **DegreeDiscountIC()**
- **IC()**
- **LT()**

2.3 Details of the algorithms

- find-all-neighbors()

This algorithm finds all the neighbors $n$of vertex v and store them in S

Algorithm 1 find-all-neighbors(**G,n**)

Initialize S
for each vertex v in G **do**
  if v is neighbor of n **then**
    S.append(v)
  end if
end for
return S

- find-inactive-neighbors()

This algorithm finds all the inactive neighbors $n$of vertex v and store them in S. And this algorithm also gets the probability p that the neighbor be actived and stores them in P.

Algorithm 2 find-inactive-neighbors(**G,n,p**)

Initialize S, P
for each neighbor n in **G do**
  if n is not active **then**
    S.append(n)
    P.append(p)
  end if
end for
return S, P

- find-active-neighbors()

This algorithm finds all the active neighbors $n$of vertex v and store them in S. And this algorithm also gets the sum of the degrees p and stores them in *Sum*.

Algorithm 3 find-active-neighbors(**G,n,s**)

```
Initialize S, Sum
for each neighbor n in G do
  if n is active then
    S.append(n)
    Sum.append(p)
  end if
end for
return S, Sum
```

- IC()

This is a sample of the IC model. Use the given seedset and the graph to get the number of people which the seedset can spread to.

Algorithm 4 IC(G, *SeedSet*)

```
ActivitySet = SeedSet
count = ActivitySet.length
while ActivitySet is not empty do
  newActivitySet
  for seed s in ActivitySet do
    for inactive neighborn in seed s do
      active neighborn at probability p
      if (success) then
        Update n
        newActivitySet.add(n)
      end if
    end for
  end for
  count=count+newActivity.length
  ActivitySet=newActivitySet
end while
return count
```

- LT()

This is a sample of the LT model. Use the given seedset and the graph to get the number of people which the seedset can spread to.

Algorithm 5 LT(G,$SeedSet$)

```
ActivitySet = SeedSet
Initialize T
count = ActivitySet.length
while ActivitySet is not empty do
    newActivitySet
    for seed s in ActivitySet do
        for inactive neighborn in seed s do
            get w-total
            if w-total ¿=n.T then
                Update n
                newActivitySet.add(n)
            end if
        end for
    end for
    count=count+newActivity.length
    ActivitySet=newActivitySet
end while
return count
```

Algorithm 6 DegreeDiscountIC(G,k)

```
initialize S
for each vertex v do
    compute its degree d_v
    dd_v=d_v
    initialize t_v to 0
end for
for i=1 to k do
    select u=argmax(dd_v where v ∈ Setu)
    S.append(u)
    for each neighbor v of u and v ∈ Setu do
        t_v=t_v+1
        dd_v=d_v-2t_v-(d_v-t_v)t_vp
end for
return S
```

3 Empirical Verification

Empirical Verfication can be reached when given different Graphs, the program will output an nearly correct answer(due to the random part of the project, a total correct answer can't be reached every time.)

3.1 Design

This program contains two simple model algorithm and an algorithm to get a seed set

3.2 Data

The data type used in this project contains np.array, list. Txt file is used to test.

3.3 Performance

The performance of this project can be measured by setting the time that the program runs and calculate the accrancy of the output.

3.4 Result

The result is shown after running the program.

3.5 Analysis

This program is able to calculate andulate and get a seedset of a graph which have a wide spread range. However, when given a large graph(with over 10000 nodes), the performance of the program becomes not so good and need to be improved.

4 Reference

[1] Wei Chen, Yajun Wang, Siyu Yang Available:http://snap.stanford.edu/class/cs224w-readings/chen09influence.pdf