Influence Maximization Problem Project

Wentao Ning 11610914
Computer Science and technology
Southern University of Science and Technology
11610914@mail.sustc.edu.cn

1. Preliminaries

Description: Influence Maximization Problem (IMP) is the problem of finding a small subset of nodes (referred to as seed set) in a social network that could maximize the spread of influence. The influence spread is the expected number of nodes that are influenced by the nodes in the seed set in a cascade manner. The Linear Threshold and Independent Cascade Models are two of the most basic and widely-studied diffusion models[1]

Application: The IMP is often used to calculate the influence of a social network and so on. A social network, the graph of relationships and interactions within a group of individuals, plays a fundamental role as a medium for the spread of information, ideas, and influence among its members[1]

1.1. Software

I wrote this project in Python 3.6 using editor Sublime Text 2 and IDE PyCharm.

1.2. Algorithm

At first I wrote a preprocess function to preprocess the file and get the node number and all edges. Then I used CELF algorithm which is a kind of greedy search algorithm based on submodularity to get the final result.

2. Methodology

2.1. Representation

There are 3 global variables in both ISE.py and IMP.py.

- **graph_list**: A dict which records the adjacent list of each node
- **graph_parent_list**: A dict which records the parent nodes of each node
- **neighbor_prob**: A dict which records the weight of each node

2.2. Architecture

There are several methods in my ISE.py file.

- **preprocess**: Preproccess the given file and construct the graph.
- **calculate_influence**: To calculate the influence based on different diffusion model.
- **one_IC_sample**: To calculate the influence based on IC diffusion model.
- **one_LT_sample**: To calculate the influence based on LT diffusion model.

There are some other functions in my IMP.py file.

- **CELF**: Using a kind of greedy algorithm called CELF to calculate the result set.
- **print_results**: To print result as the required format.

2.3. Detail of Algorithm

There are 3 main functions in my ISE.py file.

- **one_IC_sample**: To calculate the influence based on IC diffusion model.

Algorithm 1 one_IC_sample

$AcitivitySet \leftarrow SeedSet$
$count \leftarrow len(ActivitySet)$
while ActivitySet is not empty **do**
  $newActivitySet \leftarrow \emptyset$
  for each seed in ActivitySet **do**
    for each inactive neighbor in seed **do**
      seed tries to activate neighbors with probability
      if activated **then**
        newActivitySet.add(neighbor)
      end if
    end for
  end for
  $count \leftarrow count + len(newActivitySet)$
  $AcitivitySet \leftarrow newActivitySet$
end while
return count

- **one_LT_sample**: To calculate the influence based on LT diffusion model.

Algorithm 2 one_LT_sample

```
AcitivitySet ← SeedSet
Randomly set threshold for each node
count ← len(ActivitySet)
while ActivitySet is not empty do
    newActivitySet ← ∅
    for each seed in ActivitySet do
        for each inactive neighbor in seed do
            calculate the weight of activated neighbors: w_total
            if w_total ≥ neighbor.threshold then
                update the state as Active
                newActivitySet.add(neighbor)
            end if
        end for
    end for
    count ← count + len(newActivitySet)
    AcitivitySet ← newActivitySet
end while
return count
```

- **calculate_influence**: To calculate the influence based on 10000 times iterations on LT or IC diffusion model.

Algorithm 3 calculate_influence(seed, model)

```
sum ← 0
times ← 10000
if model.upper() = "IC" then
    for i = 1 to times do
        sum ← sum + one_IC_sample(seeds)
    end for
    return sum / times
else if model.upper() = "LT" then
    for i = 1 to times do
        sum ← sum + one_LT_sample(seeds)
    end for
    return sum / times
end if
```

There is one main function in my IMP.py file.

CELF: Using a kind of greedy algorithm called CELF to calculate the result set.

Algorithm 4 CELF

```
influence_dict ← empty dict
for i = 1 to len(neighbor_prob) do
    influence_dict[i] ← calculate_influence(i, model)
end for
seeds ← empty set
tmp_max_node ← the key in influence_dict having the max value
add tmp_max_node2 to seeds
pop tmp_max_node from influence_dict
while len(seeds) < seed_num do
    tmp_max_node ← the key in influence_dict having the max value
    influence_dict[tmp_max_node] ← the influence increment after adding tmp_max_node
    tmp_max_node2 ← the key in influence_dict having the max value
    if tmp_max_node2 = tmp_max_node then
        add tmp_max_node2 to seeds
        pop tmp_max_node2 from influence_dict
    else
        continue
    end if
end while
return seeds
```

3. Empirical Verification

3.1. Design

In the Empirical Verification part, I use the network.txt which is given by teacher as the input graph and the seeds.txt and seeds2.txt as input seed set to verify the ISE model. Then I use network.txt as input to verify the IMP model.

3.2. Data and data structure

- **Data Set**: NetHEPT.txt, network.txt, seeds.txt and seeds2.txt
- **Data Structure**: list, set, dict and numpy 2D matrix.

3.3. Performance & Hyperparameters

Performance Measurement: I run my algorithm on the given files, network.txt and NetHEPT.txt. I test my ISE.py file to calculate the correctness using seeds.txt and seeds2.txt file. I test my IMP.py file to get 5 seeds and 10 seeds respectively.

Hyperparameters setting: At first, I set the ISE Algorithm's iteration times to 10000 in my IMP.py file, When I run the IMP.py file on the NetHEPT.txt, I found it is too slow and can not get a good result on the given time. So I change the iteration times to 1000, I found the result is not change but the run time decreased dramaticly. Then I changed it to 500, the result is also unchanged but the run time decreased half. Then I continue decreasing the iteration times, but this time when I decrease the iteration times, the performance of this algorithm will be worse. So I set the iteration times to 500 finally.

3.4. Result

In the ISE test part, I let the one_IC_sample or one_LT_sample iterate 10000 times and recorded the result and used time. The result is below.

	IC	LT
seeds.txt	5.019 / 0.23s	5.033 / 0.05s
seeds2.txt	30.47 / 0.25s	37.07 / 0.64s

TABLE 1. INFLUENCE RESULT AND USED TIME OF EACH DIFFUSION MODEL AND SEED SET

In the IMP test part, I set the seed set as 5 and 10 respectively, the result and the used time is below.

	IC	LT
5	30.674 / 1.959s	30.719 / 1.831s
10	42.75 / 5.587s	42.862 / 6.616s

TABLE 2. INFLUENCE RESULT AND USED TIME OF EACH DIFFUSION MODEL AND SEED SIZE

3.5. Conclusion

This project is to solving a problem called Influence Maximum Problem. I used a kind of greedy algorithm to solve this problem. Admittedly, the general greedy algorithm is not effective, so I used an algorithm called CELF which consists of greedy algorithm plus prunning based on submodularity. The advantage of my algorithm is that this algorithm can have a good performance on the gragh which is not too large, and it is very fast to get the results. In my opinion, the disadventage of my algorithm is that my algorithm can not get a good performance on the large graph,it will take several minutes or hours to run on the large graph, which is what I need to improve.

Acknowledgments

References

[1] D. Kempe, J. Kleinberg and E. Tardos, ”Maximizing the Spread of Influence through a Social Network”, Cs.cornell.edu, 2003. [Online]. Available: https://www.cs.cornell.edu/home/kleinber/kdd03-inf.pdf. [Accessed: 01- Dec- 2018].