Marvel Universe Social Network Analysis

Bao Jingcun G1902320B
Shi Xiangchan G1901943J
Tian Meng G1902196K
Wang Kunyu G1901931F

Content

1.Introduction and motivation 2

2.Community detection.....3

3.Key hero detection 9

4.Evaluation 12

5.Extension 13

6.Conclusion 14

1.Introduction and motivation

1.1 Motivation

The motivation of this project is to simulate a social network environment based on Marvel's characters, by establishing the Graph of social network and apply an appropriate Graph mining to figure out the possible community and detect the key player in each community. By detect the community and define the centralized nodes from the original social network graph, it could be helpful for solve the practical problems such as customer classification, and supply chain management, which mainly focus on build a community-based supply chain to maximize the efficiency.

1.2 Graph building and introduction

The Graph will be build based on the first 1000 nodes and edges due to the total amount of them could be time consumed. The original source of the data is non-structured, so the Turicreate package will be used in this project to structure the data.

```
import networkx as nx
import turicreate as tc
```

Turicreate is mainly doing the works of transfer the data into column-based structure which will help to identify relations and nodes.

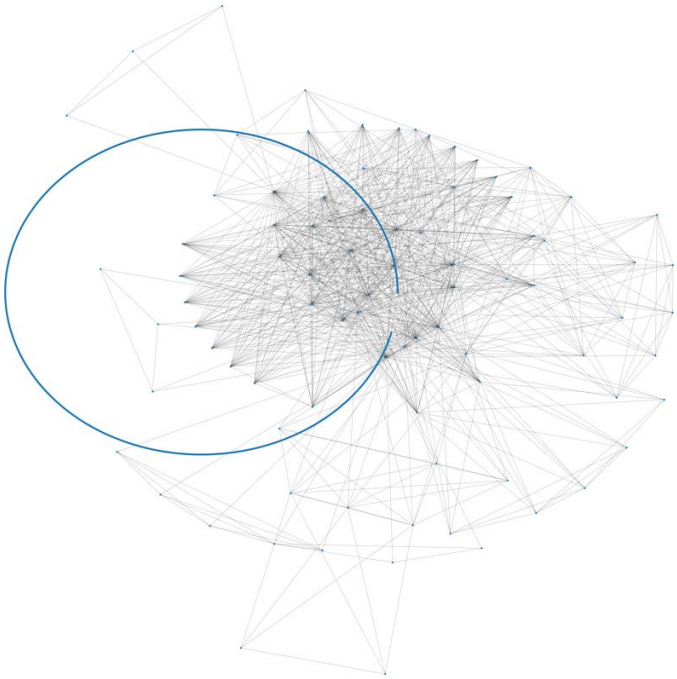


Figure 1-1: Graph with first 1000 nodes and edges

By evaluating graph, we can also figure out the degree density distribution, to identify the most popular nodes and how degree distributed.

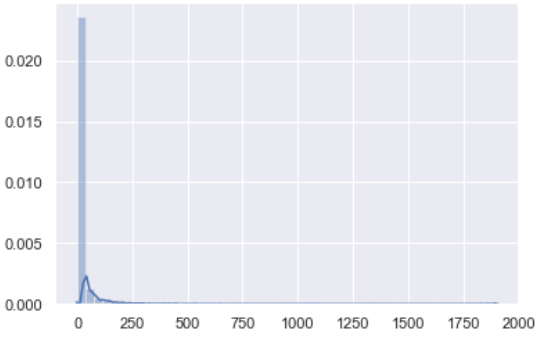


Figure 1-2 : Degree distribution

As the picture illustrate, the majority of nodes are located at degree from 0 to 1, which actually make sense since the marvel universe is basically build on serval key characters such as IRON MAN, and others will only be occurs once or only connected with one hero.

The degree analysis could also illustrate the most popular or core hero in the universe,

Such as the links for one particular nodes

```
print("There are %s superheroes connected to Black Panter"  %
      d["BLACK PANTHER/T'CHAL"])
```

```
There are 711 superheroes connected to Black Panter
```

Or the nodes with highest density

```
import operator
max(dict(d).items(), key=operator.itemgetter(1))
```

```
('CAPTAIN AMERICA', 1908)
```

2.Community detection

In order to find the communities of marvel universe, we used Newman Fast Greedy Algorithm and Label Propagation Algorithm.

2.1. Newman Fast Algorithm

This method is to find communities in graph using Clauset-Newman-Moore greedy modularity maximization. It currently supports the Graph class and does not consider edge weights.

Here is the step of this algorithm:

1) Separate each vertex solely into n community.
2) Calculate ΔQ for all possible community pairs.
3) Merge the pair of the largest increase in Q.
4) Repeat 2 & 3 until all communities merged in one community.
5) Cross cut the dendogram where Q is maximum

Note: The modularity Q is a measure for assessing communities which shows the quality of a particular partition of the network.

The modularity Q is defined as:

$$Q = \sum_i (e_{ii} - a_i^2)$$

And e_{ii} = % edges in module i, a_i = % edges with at least 1 end in module i

For example, the Figure 1 shows that Zachary network can get the highest modularity Q when it is split into 2 communities and each community has 17 nodes.

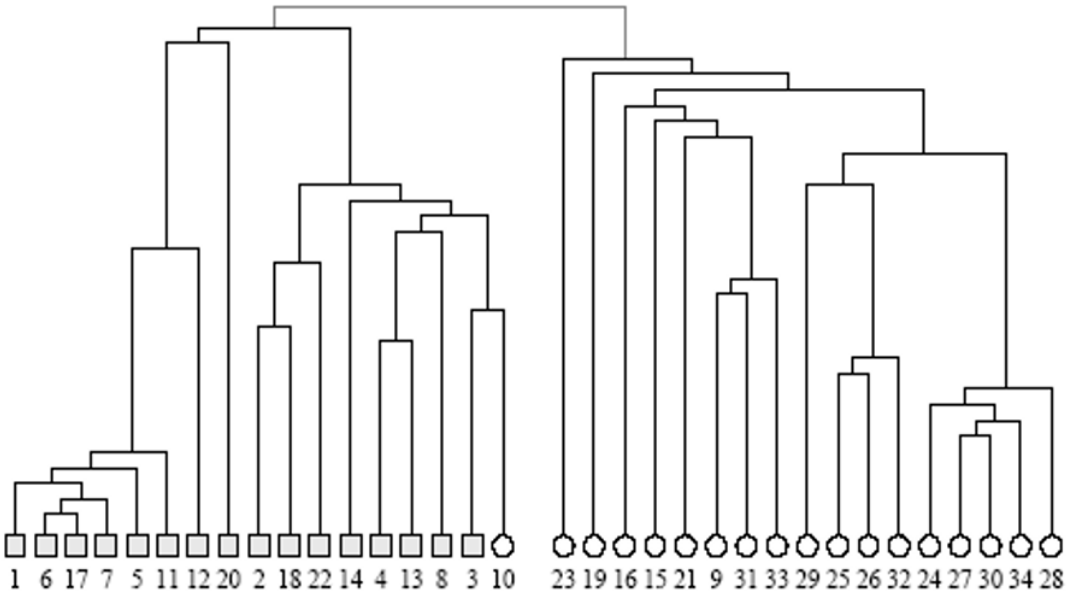


Figure 2-1: Dendrogram of the communities found by Newman Fast Algorithm

2.2. Label Propagation Algorithm

The Label Propagation algorithm (LPA) is a fast algorithm for finding communities in a graph. It detects these communities using network structure alone as its guide, and doesn't require a pre-defined objective function or prior information about the communities.

The process is formally described as Algorithm 1:

1 **Initialize labels:** for each $v \in V$, $l_v(0) = v$;
2 i = 0;
3 **while** *the stop criterion is not met* **do**
4 i++;
5 **Propagation:**
6 **foreach** $v \in V$ **do**
7 $l_v(i) = \arg\max_l \sum_{u \in N(v)} [l_u(i-1) == l]$,
8 **end**
9 **end**
10 **return** Final labeling: $l_v(t)$ for each $v \in V$, where t is the last executed step.

Algorithm 1: LP algorithm (synchronous)

The Figure 2 is an example of LP algorithm in community detection. There are 2 steps:

1) Each node in the network is initially given a unique label.
2) Then for each iteration, each node is updated by choosing the label which is the most frequent among its neighbours. If multiple choices are possible (as for example in the beginning), one among the candidate labels is picked randomly.

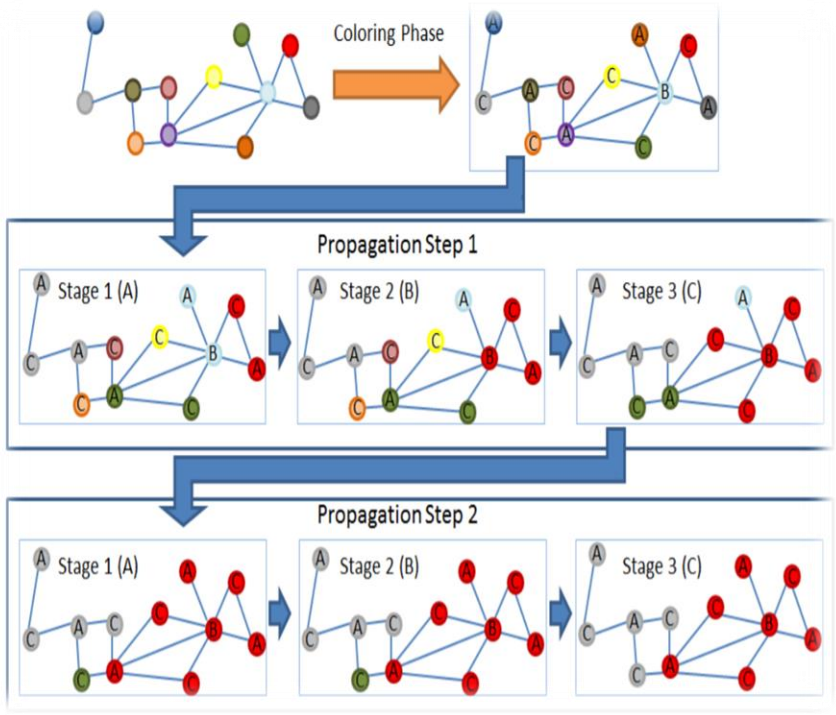


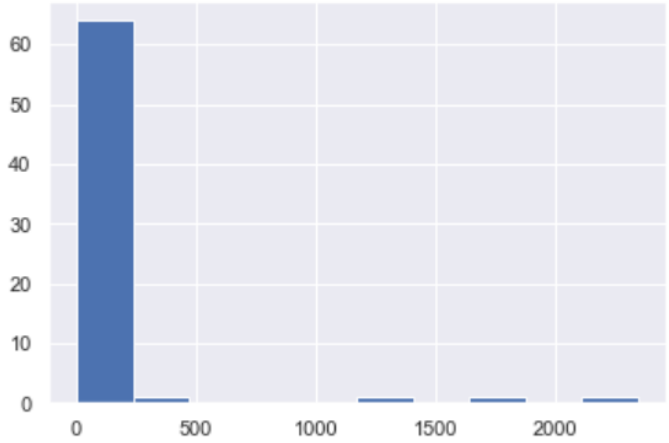
Figure 2-2: Example of LP algorithm

2.3. Marvel universe community detection with Newman Fast algorithm

We used *greedy_modularity_communities* of *networkx* library in python to do community detection. This is our code:

```
from networkx.algorithms.community import greedy_modularity_communities
cc = greedy_modularity_communities(h)
len(cc)
```

Then we draw the histogram to show the number of nodes in the communities we detected.



From the histogram, we can see that the number of nodes in most communities are less than 250, only a few communities has over 250 nodes. So we select the communities with nodes number between 100 and 500 to do visualization.

This is our code to find these selected communities

```
selected_community_list = [c for c in cc if 500 > len(c) > 100]
len(selected_community_list)
```

The result of this code is '2', which means that there are 2 communities are selected for visualization.

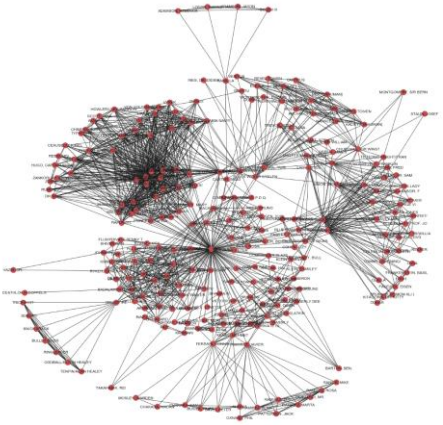


Figure 2-3: Community 1-1

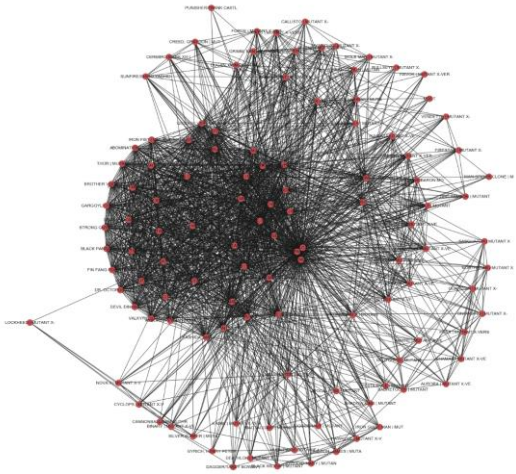


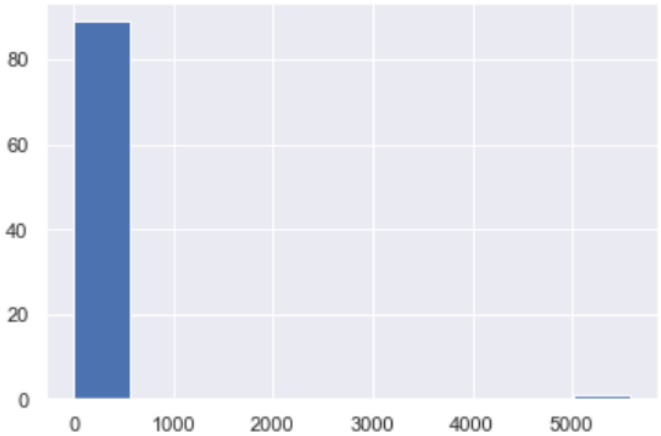
Figure 2-4: Community 1-2

2.4. Marvel universe community detection with Label Propagation algorithm

We used *label_propagation_communities* of *networkx* library in python to do community detection. This is our code:

```
from networkx.algorithms.community.label_propagation import
label_propagation_communities
cc2 = list(label_propagation_communities(h))
len(cc2)
```

Then we draw the histogram to show the number of nodes in the communities we detected.



From the histogram, we can see that the number of nodes in most communities are less than 500, only one community has over 500 nodes. So we select the communities with nodes number between 30 and 300 to do visualization.

This is our code to find these selected communities

```
selected_community_list2 = [c for c in cc2 if 300>len(c) > 30]
len(selected_community_list2)
```

The result of this code is ‘5’, which means that there are 5 communities are selected for visualization.

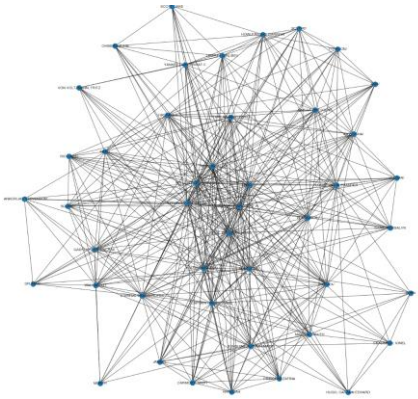


Figure 2-5: Community 2-1

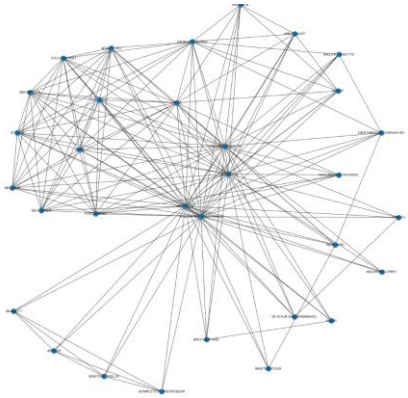


Figure 2-6: Community 2-2

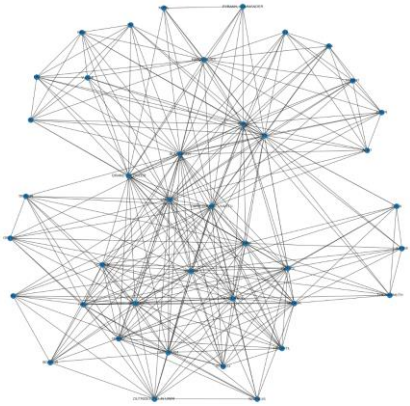


Figure 2-7: Community 2-3

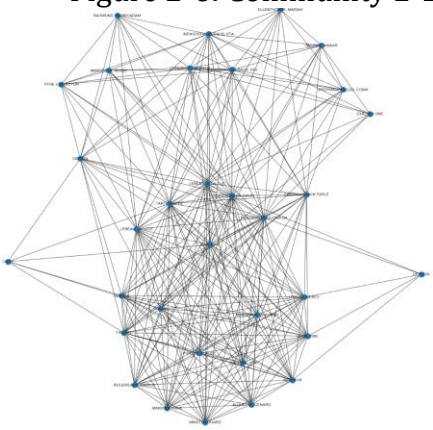


Figure 2-8: Community 2-4

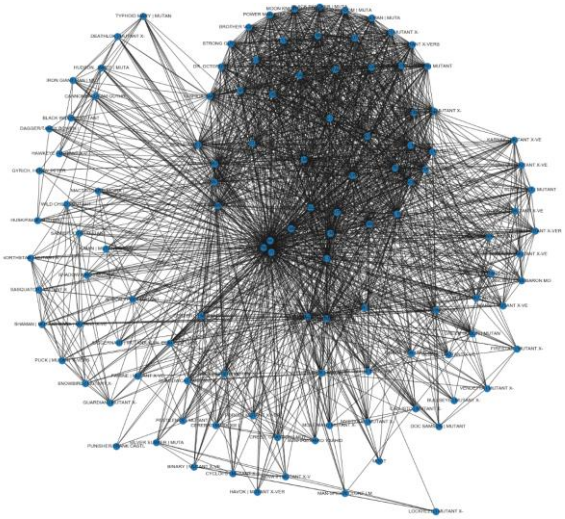


Figure 2-9: Community 2-5

3.Key hero detection

We use three methods to find key hero for each community. The three methods are degree centrality, closeness centrality and PageRank.

3.1 Key hero detection with degree centrality

Degree centrality is the simplest centrality measure to compute. Recall that a node's degree is simply a count of how many social connections (i.e., edges) it has. The degree centrality for a node is simply its degree. A node with 10 social connections would have a degree centrality of 10.

In the python library networkx, the program will convert those numbers into a 0-1 scale. In such cases, the node with the highest degree in the network will have a degree centrality of 1, and every other node's centrality will be the fraction of its degree compared with that most popular node. For example, if the highest-degree node in a network has 20 edges, a node with 10 edges would have a degree centrality of 0.5 (10 ÷ 20).

We use *degree_centrality* from *networkx* library to find the key hero for each community.
This is our code:

```
for i in range(len(selected_community_list2)):
    c=h.subgraph(selected_community_list2[i])
    d = nx.degree_centrality(c)
    key_player = max(dict(d).items(), key=operator.itemgetter(1))
```

For each community, we find the key hero with highest degree centrality:

Community	Key hero	Degree centrality
Greedy Modularity Community 1	FURY, COL. NICHOLAS	0.437275985663082
Greedy Modularity Community 2	BRUTE | MUTANT X-VER	0.9914529914529916
Label Propagation Community 1	BLACK FOX/ROBERT W.	0.9772727272727273
Label Propagation Community 2	M'SHULLA	1.0
Label Propagation Community 3	UNI-LORD	0.8717948717948718
Label Propagation Community 4	BRUTE | MUTANT X-VER	0.9914529914529916
Label Propagation Community 5	COURIER/JACOB GAVIN	0.9375

Figure 3-1 Highest centrality

For degree centrality, higher values mean that the hero is more central. Degree centrality shows how many connections a hero has. They may be connected to a lot of heroes at the heart of the network, but they might also be far off on the edge of the network.

3.2 Key hero detection with closeness centrality

'Central' nodes are important, as they can reach the whole network more quickly. The importance measured by how close a node is to other nodes. People usually refer to its normalized form which represents the average length of the shortest paths instead of their sum. It is generally given by the formula multiplied by N-1, where N is the number of nodes in the graph. The normalized form of closeness centrality is:

$$C_C(v_i) = \left[\frac{1}{n-1}\sum_{j\neq i}^{n} g(v_i, v_j)\right]^{-1} = \frac{n-1}{\sum_{j\neq i}^{n} g(v_i, v_j)}$$

We use *closeness_centrality* from *networkx* library to find the key hero for each community.
This is our code:

```
for i in range(len(selected_community_list2)):
    c=h.subgraph(selected_community_list2[i])
    d = nx.closeness_centrality(c)
    key_player = max(dict(d).items(), key=operator.itemgetter(1))
```

For each community, we find the key hero with highest closeness centrality:

Community	Key hero	Closeness centrality
Greedy Modularity Community 1	FURY, COL. NICHOLAS	0.58985200845665
Greedy Modularity Community 2	BRUTE | MUTANT X-VER	0.9915254237288136
Label Propagation Community 1	BLACK FOX/ROBERT W.	0.9777777777777777
Label Propagation Community 2	M'SHULLA	1.0
Label Propagation Community 3	UNI-LORD	0.886363636363636
Label Propagation Community 4	BRUTE | MUTANT X-VER	0.9915254237288136
Label Propagation Community 5	COURIER/JACOB GAVIN	0.9411764705882353

Figure 3-2Highest closeness centrality

The benefits of closeness centrality are that it indicates those heroes as more central if they are closer to most of the heroes in the graph. This strongly

corresponds to visual centrality—a node that would appear toward the center of a graph when we draw it usually has a high closeness centrality.

3.3 Key hero detection with PageRank

PageRank is an algorithm that measures the transitive, or directional, influence of nodes. All other centrality algorithms we discuss measure the direct influence of a node, whereas PageRank considers the influence of your neighbors and their neighbors. For example, having a few influential friends could raise your PageRank more than just having a lot of low-influence friends.

PageRank is computed by either iteratively distributing one node's rank (originally based on degree) over its neighbors or by randomly traversing the graph and counting the frequency of hitting each node during these walks. The formula of PageRank is:

$$\mathrm{PR(A)}=(1-\mathrm{d})+\mathrm{d}(\mathrm{PR(t1)}/\mathrm{C(t1)}+\ldots+\mathrm{PR(tn)}/\mathrm{C(tn)})$$

In the formula, PR(A) is the PageRank of a page; d is a moderating factor (estimated to be 0.85); PR(t1) to PR(tn) is the PageRank of pages linking to A; C(t1) to C(tn) are the number of outgoing links in those pages.

We use *PageRank* from *networkx* library to find the key hero for each community. This is our code:

```
for i in range(len(selected_community_list2)):
    c=h.subgraph(selected_community_list2[i])
    d = nx.pagerank(c)
    key_player = max(dict(d).items(), key=operator.itemgetter(1))
```

For each community, we find the key hero with highest PageRank:

Community	Key hero	PageRank
Greedy Modularity Community 1	FURY, COL. NICHOLAS	0.04476212954041
Greedy Modularity Community 2	BRUTE | MUTANT X-VER	0.0269580834772
Label Propagation Community 1	BLACK FOX/ROBERT W.	0.0445485614355
Label Propagation Community 2	M'SHULLA	0.08170797474584
Label Propagation Community 3	UNI-LORD	0.05017076912584
Label Propagation Community 4	BRUTE | MUTANT X-VER	0.0269580834772
Label Propagation Community 5	COURIER/JACOB GAVIN	0.0488564517079

Figure 3-3 Highest Pagerank

In networkx library, the function of PageRank is designed for directed graphs, and if the input graph is undirected like we built in the Marvel network, it will firstly convert each edge in the directed graph to two edges. PageRank counts the number and quality of links to a hero, which determines an estimation of how

important the hero is. The underlying assumption is that key heroes with high PageRank are more likely to receive a higher volume of connections from other influential heroes.

4.Evaluation

4.1 Purpose

In order to evaluate these two kinds of community detection, we use modularity to compare.

4.2 Method

Modularity is one measure of the structure of networks or graphs. It was designed to measure the strength of division of a network into modules.
Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.
Modularity is often used in optimization methods for detecting community structure in networks.

The definition of modularity is shown as below.

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w) = \sum_{i=1}^{c} (e_{ii} - a_i^2)$$

Here is the explanation for each parameter:
m : number of links
k_v : degree of node v
A : adjacency matrix.
A_{vw}= 0 --- no edge between node v and node w
A_{vw} =1 --- there is an edge between node v and node w
$\delta(c_v,c_w)$: whether node v and w are in the same community, 1 --- yes, 0 --- no
e_{ii} : the number of links in the same group connecting the vertices (intralinks)
$a_{i,}$ the sum of the number of links from the vertices in group *i* to another group *j* (interlinks)

4.3 Code for evaluation

```
import modularity
```

For Greedy Modularity Community

```
Greedy_modularity = community.modularity(cc_dic, h)
print('The global modularity of greedy modularity community: ',Greedy_modularity)
```

```
The global modularity of greedy modularity community:  0.35733142987873256
```

For Label Propagation Community

```
LP_modularity = community.modularity(cc2_dic, h)
print('The global modularity of label propagation community: ',LP_modularity)
```

```
The global modularity of label propagation community:  0.056376389471366604
```

4.4 Analysis

The modularity of greedy modularity community is approximately 0.36 and the modularity of label propagation modularity community is approximately 0.06. By this we conclude that there are higher proportion of Intralinks in greedy community than that in label propagation community, which could also be identified from the pictures in 2.3 and 2.4.

5.Extension

5.1 Brief of the possible application

Figure 5-1 Social network

Furthermore, we are inspired by this detection process and hope the similar technique could be used in real business environment, we can utilize the similar approach to make customer community detection, identifying the key player in each community as the targeted marketing objective and implementing further sales strategies towards other customers efficiently.

5.2 Example of application

Figure 5-2 Finding the key player

e.g. A internet company whose major product is a social network app wants to launch a new app with social networking in the form of short videos. To attract more users and to boost the daily active users, the company may initially identify those key players in each social network community, and then encourage these group of people to let their friends or fans know it. Company could give key players more rewards who can invite more new users download the app and register an account.

6.Conclusion

In this project, we try to find the communities in Marvel Universe. Firstly, we build the original graph and then make the vertices filtering. Then we applied two algorithms for detecting communities and selecting communities whose number of nodes is in [30,300] and get the results: 2 communities under Newman fast greedy algorithm; 6 communities under label propagation algorithm. Next, with three kinds of methods---degree centrality, closeness centrality and page-rank, we get the same key player in each community under two algorithms respectively. Finally, based on the evaluation result, we concluded that the greedy algorithm presents better community detection result. Moreover, we also propose possible application of this detection technique to help the company enhance performance efficiently and also present an example for companies whose products rely on social network relationship.