Bilkent University
Computer Engineering Department

Project Report

TABLE OF CONTENTS:

INTRODUCTION

Marvel Comics, together with DC Comics, have been for many decades the two main comic book publishing companies in the world. Marvel Comics first became popular during the Golden Age of comics.Some of the characters created in this period, such as Spider-Man, Captain America, X-Men and Fantastic Four became worldwide and have become cultural icons of the western society of the last forty years.

We looked at the Marvel Universe collaboration network,where two Marvel characters are considered linked if they jointly appear in the same Marvel comic book. This network is created artificially by Marvel's writers especially by Stan Lee. However, this network mimics real-life networks. A hero with its' own title series appearing in an issue of another hero's series were common in the Golden Age period. Characters lived their adventures in the same fictional cosmos, and they interacted each other like real actors.Same villains and secondary characters would appear in comic books of different issues and there were continuous references to the events that were simultaneously happening, or had happened.

Most pairs of characters that have jointly appeared in the same comic book have fought shoulder to shoulder or each other, or have had some other strong relationship, like family ties, kidnapping and archenemies. Thus, this network has the same nature of a real scientific collaboration network.

We have investigated communities in the Marvel Universe collaboration network. It was interesting to compare our results with the knowledge we have about the Marvel Universe. We have found many Marvel characters that were overlapping in many communities because of the crossover of the Marvel characters in the comic books.

DATASET

The dataset we have used in our project is Marvel Chronology Project (MCP) [1]. This dataset catalogs every appearance of significant Marvel characters by comic books starting from November 1961. We have found that it was common for characters to take different personalities. For example, Hank Pym who is one of the most popular Avengers, had been known as Ant-Man, Giant-Man, Goliath, YellowJacket, and as

world's greatest biochemist Dr. Henry Pym in many comic books. Similarly, many characters had the same personality from time to time. For instance, there were at least three Goliath's: Hank Pym, Clint Barton and Erik Josten. Fortunately, the MCP assigns a node for each person independently of the nickname or personality under which it appears in each comic book.

We had in total 6486 Marvel characters and 12942 comic books. We had two input files. One was the name of the characters and the titles of issues of the Marvel comic books. The other one was showing which characters appeared in each of the comic books. We merged these input files to create a character to character undirected weighted graph. There was an edge between two characters if they both appeared in at least one comic book. Of course the weight of this edge is the number of comic books they both have appeared in.

PAGERANK

After the preprocess of the dataset, to use in the community detection algorithms and to interpret our data, pagerank algorithm is used. Our datasets is used to create an adjacency matrix, the with using this matrix and networkX library, a graph is created. During random walk process, each weight is considered as an indicator which shows the probability of the edge. So, if an edge from a specific node has high weight, the most probably, random walker will use that edge in that iteration. So, the nodes which have high weighted edges between them will have high pagerank at the end. This situation explains that why some superheroes (Spiderman, Ironman etc.) have the highest pagerank among almost 7000 heroes.

Also, to create a grand truth for community detection algorithms, SimRank is used. This algorithm created a pagerank vector with using a specific teleport set which consist of single superhero. By doing so, a candidate for probable community of that superhero is created. When a community is detected, the result of SimRank is used to understand that community is valid or not.

For implementation, networkX library is used since it is already handle weighted undirected graph while calculating pagerank[3].

HIERARCHICAL CLUSTERING

Hierarchical clustering finds non-overlapping communities in a dataset. It has two different methods. First one is agglomerative method which combines nodes with a bottom-up approach by finding minimum distance between centroid nodes. Second method is Divisive which recursively splits nodes to get clusters with a top-down approach.

We used agglomerative method to detect communities faster. It runs in $O(n^2 \log(n))$ time complexity and since our dataset does not contains millions of nodes, this complexity finds communities in a short time.

We implemented agglomerative method by modifying according to our needs.

Below pseudo code represents our implementation:

Given:
A set of X objects {x1, x2, ..., xn}
A distance dictionary dist[(xi, xj)] = jaccard(xi, xj)
Sort dist dictionary by distance values
for i = 1 to n
  ci = Community(xi)
end for
C = {c1,....., cn}

```
min_dist, centroid1, centroid2 = find_min_dist(dist)
while(min_dist < threshold_distance):
  comm = combine_communities(centroid1, centroid2)
  find_centroid(comm)
  min_dist, centroid1, centroid2 = find_min_dist(dist)
```

Here is the explanation: We have a matrix which have characters as rows and comic books as columns. Since we have a document-word like dataset, jaccard similarity is appropriate to calculate distances between characters. Therefore we get a distance dictionary which stores key(char, char), value(distance) pairs for all character pairs. Then we sort this dictionary by its distance values to get minimum distance in optimal time later in the implementation. However, this sort cost us $O(n^2 \log(n))$ time complexity. We treat all character nodes as communities and store them in a list. At the beginning all character nodes have same community number with its node number. Now, we set a distance threshold to combine communities. If distance between communities less than threshold, we can combine communities. Distance between communities calculated by distance between their centroids. Centroids are determined by calculating each node's sum of square distance to other nodes in its community. Node with smallest sum of squares distance become the centroid of that community. In each iteration in the while loop we find the communities with minimum distance and combine them and assign a centroid to our new community. We repeat this process in each iteration. Therefore we can find non-overlapping communities with specified distance threshold.

COMMUNITY DETECTION

We used community detection algorithm which was written by Zongqing Lu, Yonggang Wen and Guohong Cao who are from Nanyang Technological University [2]. By using this algorithm we found communities in weighted graphs. There is an example in the graph below [Figure 1]. When we apply algorithm on this graph, communities would be { {b,d}, {e, f, c, a, g}, { g, h, i} }. As it be seen from example there may be

overlapping in communities, for example node of g is included in both {e, f, c, a, g} and {g, h, i} communities. This is the difference of community detection algorithms from clustering algorithms.

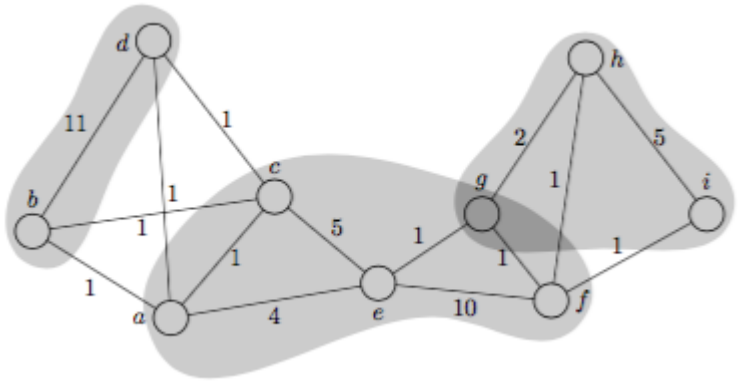

Figure 1: Example Graph for Community Detection

We'll give more details about algorithm and explain some implementation methods.

Algorithm 1: Detection Algorithm

Input : $G = (V, E)$
Output: $\mathcal{C}$
Initialize: $\mathcal{C} = \emptyset$;
while $E \neq \emptyset$ **do**
  $C = \{u, v\}$, where $(u, v) = \arg\max_{(u,v)\in E} w_{uv}$;
  while $N_C \neq \emptyset$ **do**
    $C' = C \cup \arg\max_{w\in N_C} B(w, C)$;
    if $\Phi(C') < \Phi(C)$ **then**
      $C = C'$;
    else
      break;
    end
  end
  $E = E \backslash E_C$;
  $\mathcal{C} = \mathcal{C} \cup C$;
end

Figure 2. Pseudocode for algorithm

There is the pseudocode for algorithm in the [Figure 2]. Firstly we choose the edge which has the maximum weight with compared to other edges and create a new

community from that edge's both vertices. After that we choose the node from neighbors of community, that node has the maximum belonging degree to that community. Belonging degree to community is calculated as:

$$B(u,C) = \frac{\sum_{v \in C} w_{uv}}{k_u}.$$

and

$$k_u = \sum_{v \in N_u} w_{uv}.$$

So belonging degree of node u to community C is found by dividing sum of weight of edges which connects u to the node in the community of C to sum of weight of all edges which belong to node u. More belonging degree means more possibility for get involved in community C.

After finding the node with maximum belonging degree for community C, we create temporary community which is created by adding that to the current community C. We calculate conductance of each community which are current community and new community. Conductance of community is calculated by:

$$\Phi(C) = \frac{cut(C, G \backslash C)}{w_C},$$

Cut(C, G\C) is equal to sum of weights of cutting edges of C from all graph, which means sum of weights of all edges which connects two nodes which apply that condition: one of them should be in community and other should not be in community. w_C is sum of weights of all edges of nodes which are in the community. For example in the above example for community C = {b,d}, cut(C, G\C) = cut({b,d}, {a,c,e,f,g,h,i,j}) is equal to 4 and $w_{\{b,d\}}$ is equal to 15.

If conductance of new community is smaller than the conductance of the current community, we add that node to current community. Otherwise, if conductance of new community is not smaller than current conductance, extending community is finished.

After finishing the process of extending community, we extract edges in community from current edges list which we are looking for finding edge with maximum weight. Lastly continue to finding new communities and we repeat all the process which

begins with finding the edge with maximum weight in order to continue to find other communities.

RESULTS

Hierarchical Detection

Results of the hierarchical clustering is very promising. It find very small local communities. However, these small communities are the most important communities which includes most of the their members. To illustrate, when we analyzed results, we found Avengers, Spider-man, Fantastic Four and many other local but non-popular communities. Since these result does not include overlapping communities we cannot relate one character to multiple communities.

Community Detection

Firstly when we run the algorithm on our dataset of Marvel Characters which is composed of more than 6000 nodes. We found more than 1000 communities, so we need to put threshold which prevents creating redundant communities. In order to prevent this we calculated mean value of all edges, after that when we are finding edge with maximum weight we used that mean value. If maximum weight in current edges is smaller than mean values of all edges, we stopped creating new communities.

After that improvement we reduced total community count to 150s, however while showing results 150 communities could not be seen well so we take first 5 community for showing results. That shows us most significant and characteristic communities. First communities are more significant with compared to other communities which are found later, because of the prioritization of the algorithm.

Also while we are showing results we used result of PageRank algorithm on our data, we used PageRank for analyzing popularities of characters. We showed most popular 10 characters with high node size such as radius of 150, and showed next 100 populars with radius of 30, and drawed other 2000 node with radius of 10. We showed character names on the figure for only most popular 10 character. In addition to that, we

showed communities with different colors in order to make divergent. The reason of these modifications is that make results more understandable.

Here are the final results:

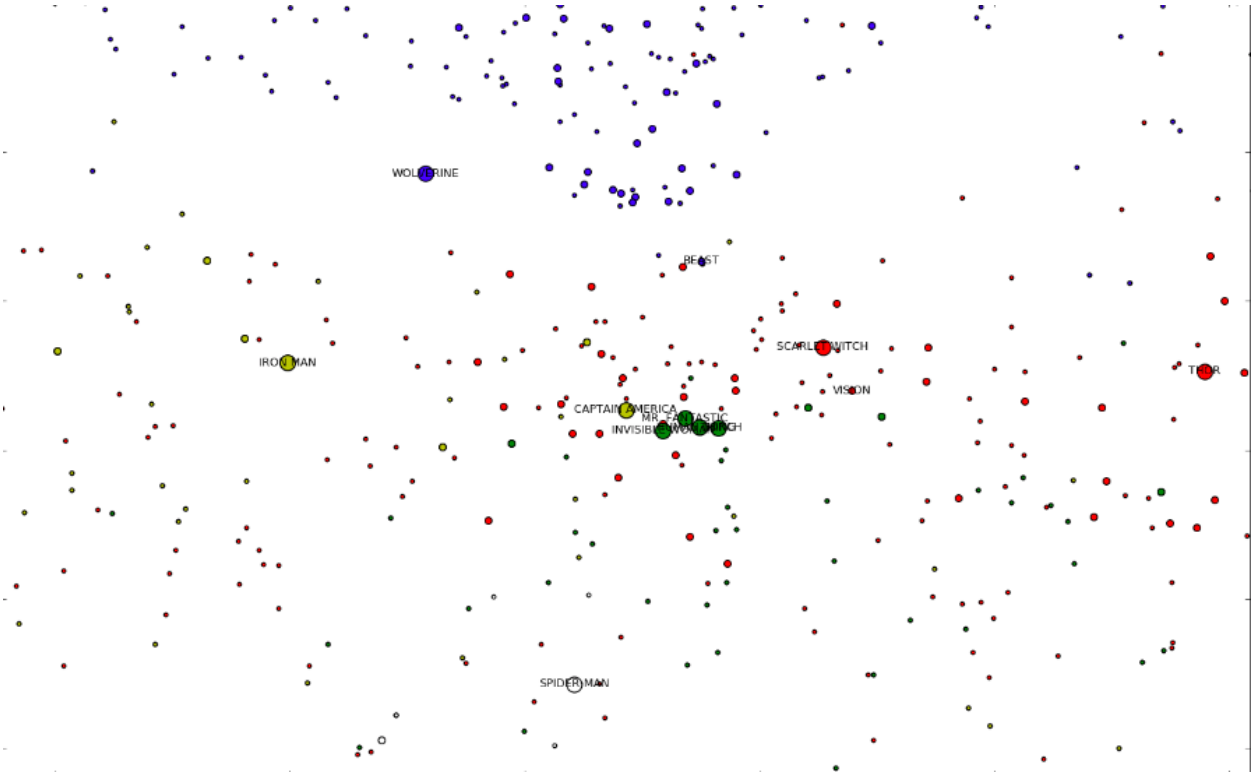

Figure 3: Sample result for Community Detection on Marvel Heroes

As it can be seen in [Figure 3], algorithm find communities which included most popular heroes. People who are a bit interested in Marvel can understand correctness of these results easily. For example, in result figure Ironman and Captain America are in same community(shown in figure by yellow). There are 4 big green nodes whose names are Mr. Fantastic, Invisible Woman, Thing and Human Torch these are the group of famous Fantastic Four and these are shown in same community(all of them has green color) and also location of them is very close to each other.

REFERENCES

[1] Marvel Chronology Project. http://www.chronologyproject.com/

[2] http://mcn.cse.psu.edu/paper/zongqing/percom-zongqing13.pdf

[3]https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.link_analysis.pagerank_alg.pagerank.html