COMP4121 Project Report
How do I influence people on online social network?

Student: YiSong Jiang

ID: z5123920

Basic Part

Online Social network

A large dimension online social network (e.g. Facebook, Twitter, Instagram, Weibo...) is usually a huge system that millions of users connected with each other. Each of online social network can be treated as a map of social relations (e.g. friends, families, collaborators...). These online social networks are such powerful tools in young people that they have already become main communication style. One estimate says that within a day, Twitter processes over 250 million tweets. Twitter, in particular, provides what we call a direction node link. Because A may subscribe to B tweets but B don't subscribe to A. And indeed, Twitter's success builds on top of micro-blogging and group texting and one-sided or one direction social networking.

Given the popularity of these services, there is no shortage of interesting questions people have raised. One question is, how to measure the influential power of individuals on an online social network. There are variety of possibilities ways. For example, we can measure on Twitter the number of re-tweets of a user's tweets. As we identify through a party of via. We can also look at the number of re-posting of URL's that a user posted. If we have access to the topology of who follows whom, for example, A follow B's tweets, and B follow C's tweets, and so on. Then, this typology can also give us some information about influential power of different nodes which are different Twitter users.

And that relates to the second question, how to leverage the above knowledge of the individual influential power source and actually influence people online. For example, maybe we'll be able to seed some users and ask them to post or tweet certain information about our product or service. And the question is, which nodes to seed? If we know the topology, would it help us to answer this question more precisely? So, the two questions are indeed important.

Topology Based Influence Models

In order to answer the two questions, we have to, represent our important quantities with some symbols. So, we start with, of course, graphs.

In this project, we mostly focus on undirected graphs. That means all the links are *bidirectional*. Then we can build our *adjacency matrix* **A**. A given graph with N nodes will have an adjacency matrix of dimension N*N. A_{ij} is 1 if there is a link from node i to j, and 0 otherwise.

Measuring node importance

There several approaches to measuring importance of a node.

1. Degree: the number of nodes connected to a node. In a directed graph, it often has two degree --- in-degree and out-degree, represents the number of incoming edges and outgoing edges. Dunbar's number is used by sociologists as an estimate of the largest number of friends in a social network an average person can keep, usually quoted at around 150. But it is obviously not a good one since degree can misleading number to quantify node importance. Similar with page rank algorithm, if we consider each node as a page, then the importance of a node is much more about the number of connected pages. (More discussion about page rank and influence model will be in later part of this report)
2. Eigenvector centrality: Starting with an initialization vector $x(t) = A^t x\,(0)$. We can always write a vector as a linear combination of the eigenvectors $\{v_i\}$ of A, arranged in descending order and indexed by i, for some weight constants $\{c_i\}$. That means $x(0) = \sum_i c_i v_i$. So $x(t) = A^t \sum_i c_i v_i = \sum_i c_i v_i \lambda_i^t$.

   As $t \to \infty, x_i = \frac{1}{\lambda_1} \sum_j A_{ij} x_j\,, \forall i.$
3. Closeness centrality: closeness of a node is calculated by shortest distance to another node. In this project, the links' weights are treated equally as distance. The distance between node i and j that is the shortest one called as D_{ij}. So, we can tabulate these D_{ij} for all of the node i, j pairs. Pick the largest one. And we call that the *diameter* of this graph. It is the maximum D_{ij} across all D_{ij} pairs. It's the

largest to smallest path. Smallest for a given node pair, and largest across all node pairs. We can also look at the average instead of the max of these D_{ij}. So, we can sum up the D_{ij} and then normalize. That says the following, if we pick a node i here fix i and then look at all the other nodes j. And there are N-1 of them. Then look at the distance from node i to all the others. Closeness centrality is the reciprocal of this average:

$C_i = \frac{n-1}{\sum_j D_{ij}}$.

4. <u>Betweenness centrality</u>: If a node is on the (shortest) paths of many other pairs of nodes, then the node is important. Let g_{st} be the total number of shortest paths between two different nodes, source s and destination t (neither of which is node i itself), and n_{st}^i be the number of such paths that node i sits on. Then betweenness centrality of node i is

$$B_i = \sum_{si} \frac{n_{st}^i}{g_{st}}$$

(More discussion about $C_i \; and \; B_i$ will be in the later part of this report)

Measuring link importance

What about link's importance? First of all, what is a link? It's not as easy to define. For example, on Facebook. I may have 300 friends, but only have communicated with 100 of them. And maybe mutual wall posting, other than happy birthday, only with 20 of them. So, do I have a link with 20, 100, or 300 people?

A link could also <u>bidirectional</u> or <u>unidirectional</u>. For example, on Twitter, as already mentioned, a link certain to be unidirectional. Some of these links are <u>strong</u>. Some of these links are <u>weak</u>. As we mentioned that if you define links as weak links if the connect nodes that are not otherwise very well connected. Actually, they may be strong in the sense of disseminating information.

And we say if a link is between two nodes that does not have many short path among them other than this direct link. Now we say this is a <u>locally</u> important link. And if this link is connecting two important nodes in a local important way, we can call that a <u>global</u> important link. So, in fact, we can also quantify similar metrics of link's importance as node importance. In particular, we can look at link betweenness. So instead of counting how many shortest path of node pair (s, t) does node i reside on. We can look at on how many shortest path between node pair (s, t) does a particular link (i, j) sits on top. Then we can take the sum or average of that proportion and call that the betweenness important of link (i, j).

$$B_{(i,j)} = \sum_{si} \frac{n_{st}^{(i,j)}}{g_{st}}$$

(Examples of above content are in the python notebook)

Contagion Models

Equipped with a basic discussion of importance scores, we can move on to two more influence model with network topology as part of the picture. However, we have to assumed the graph is given and static. In practice, it's often difficult to get an accurate topology. And topology is often time varying, too. So, we're skipping those important practical issues. A contagion is very similar to the tipping idea. In this 2-state model, the initialization has a subset of the nodes adopting one state, for example, the state of 1, while the rest of the nodes adopt the other state, the state of 0. We can consider the state-1 nodes as the early adopters of a new product, service, or trend, so it is likely they would be a small minority in the network and not necessarily aggregated together.
We basically said that the adoption percentage p next to discreet time slot is a function of the current one. And this function captures the macroscopic effect of each individual's decision by looking at a global adoption percentage.
Instead of global percentage let's look at local percentages. A node will only flip for example from an iPad has user to an Apple watch user. If the known p of is neighbors flipped. So, rather than looking at a percentage of the entire network. Let us just look at the neighbors those that one node have a direct $\frac{1}{2}$ connection. Let's assume these are bidirectional links. And there are 6 of them. If 3 or more of them have adopted Apple watch then the node will also adopt Apple watch.
This is a local density driven decision making. Now each node may have a different threshold of the percentage of local neighbor it needs to see adopting before he decides to adopt. So, this factor p here, may differ from each node to each node. Since it's not about a whole network, global adoption percentage, we have to ask, will the whole network actually flip? To answer this, the first one to talk about is a notion in a graph

called a cluster. A cluster is really a sub set of nodes with a certain density p. A class of density p is a set of nodes such that each of these nodes has a least a fraction of p of his neighbors also in the set. So, for example, in the graph below, the set of nodes forming the lower left triangle is a cluster with density 0.5, whereas the set of nodes forming the square is a cluster with density 2/3.

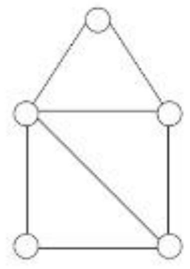


Figure 1

If the entire network flips, then that means there cannot be a cluster of density 1-p or higher among those who have not yet flipped.

Prove: If we have a cluster of density 1-p or higher, all in the state of not flipped. Then this cluster with such a density would become a blocking cluster to prevent the flipping to completely happen inside the network. Because they have enough inertia among themselves, this cluster, this subset, no matter what the rest of network looked like. Their density is 1- p or higher, so, 1- p or more of them will always point to themselves, which means we only have less than p of their neighbors being outside of them. Even if the rest of network all flips, they will still say, look at each other and say well, we do not have yet a high enough density of flipped nodes as our neighbors. So, we will still not flip. At least the only part of this theorem is easy to see.

Another most useful question is about seeding a contagion, seeding a flipping. Let's say that we have fixed a budget of $10,000 and want to stage a viral campaign to sell certain new product by word of mouth online. We want to seed or buy off very important knows, could be people whose tweets are watched and re-tweeted, by many other people. Always, they command a certain prize to be seeded to help us. Let's assume these prices are also increasing functions, of the influential power as perceived by or themselves. In order to fitting the budget, we cannot buy off everyone. Buy off a small number of them, and the question is which ones to buy off? Even without considering the price to pay them, just consider each is one unit of a dollar. It's a graph theoretic and combinatorial optimization problem

There is a trade-off between seeding nodes that they are concentrated enough. And they can have a chance of flipping at least some cluster. Versus seeding nodes so that they are far apart and therefore they may be able to flip entire network. We want to spread them out so that they can reap different corners, and yet we don't want them spread out so much that they lose the chance of flipping an even small corner of the network.

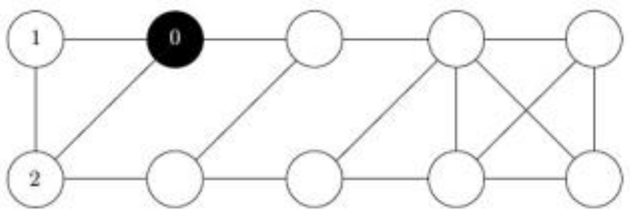


Figure 2

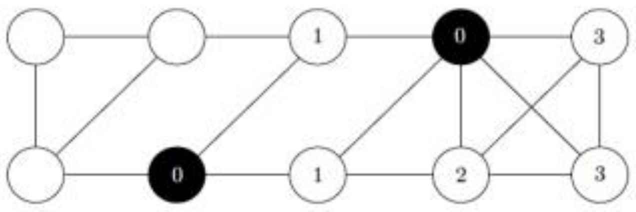


Figure 3

Another point is that seeding is not necessarily successfully refineable. There is a small example, it's supposed to want to seed a contagion in this graph here. And we've got ten nodes, with the links drawn in Figure2. Then which one shall we seed?

If we can only seed one node: At times zero shade it dark meaning that it's flipped from, not buying Apple watch to buying one for example. Flipped from one state of to the other state of the mind. Assume every other node has a flipping threshold of 0.49. If 49% or higher of node neighbors flipped, then the node will flip next as well. Then for this node. It's got two neighbors and one of the neighbors flipped. After two iterations, two of its neighbors flipped and none of the other five nodes would flip. This is the best we can do. If we can only seed one node, seed this node and we can flip two more as the equilibrium are flipping in the contagion.

But suppose we can actually seed two nodes as we got a slightly bigger budget. It turns out the optimal way to seed is the choice in Figure3. You seed these two nodes. Even though they are not directly connected, and this is the best that we can do, as it turns out.

The interesting point is that the optimal seeding strategy not successfully refineable. We cannot say, if allow me seed of seed one node, I will seed a particular node. And then when allow me to seed one more node, I could just add one more node.

Infection Models

Unlike the other models, in the infection models, we'll see state transition across more than two states. It's not just flipping from one to the other states. We're going got see three states, for example. And they are, variants with even more states. Also, instead of looking at discrete, we look at continuous time through the more convenient mathematical representation of differential equation.
We'll talk about three variants of the infection model. S stands for susceptible, I stands for infected and R stands for recovered population. We'll be looking at also the percentage of the overall population rather than the absolute values. And in the Figure4 we see that if you are susceptible to an infection disease, this is where that infection model started, originally, thus these particular choices of that node have a chance of being infected. And the rate of that happening is represented by β.

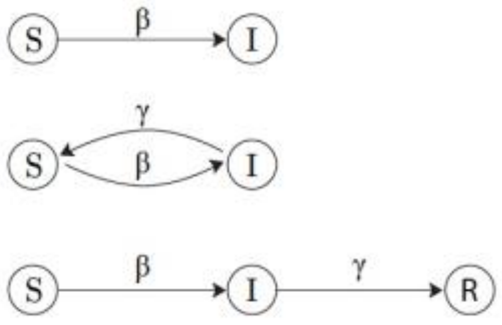

$$\begin{tikzcd} S \arrow[r, "\beta"] & I \end{tikzcd}$$
$$\begin{tikzcd} S \arrow[r, bend right, "\beta"'] & I \arrow[l, bend right, "\gamma"'] \end{tikzcd}$$
$$\begin{tikzcd} S \arrow[r, "\beta"] & I \arrow[r, "\gamma"] & R \end{tikzcd}$$

Figure 4

And if a node is infected, then we have a choice of deciding what to say in the model. In SI model, the simpler ones, says that then the node just stay infected forever. In another SIS model, the node may actually be able to recover. Or at least go back to susceptible with a certain rate γ. And in the SIR models, the node actually truly recover and become immunized from this infectious disease with a certain rate γ.

Math detail for the three model

The calculate details and examples are based on math symbol and graph, so please check the python notebook for this part.

Infection with topology

We have assumed that the impact is on the entire global population. If somebody is in China and I am in Australia, I can also get infected. Obviously for certain influence models such as infectious disease itself is hardly realistic. So, we have to take into account the impact of topology.

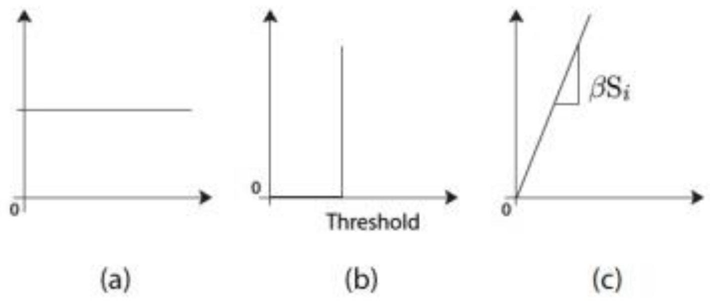


Figure 5

Figure5 has three different graphs. What plotting on the x axis is the neighbor's influence, under y axis is node I state.

The (a) graph is actually for random walk model, that we'll talk more about in this report's advanced material part.

(b) and (c) are depicting contagion and infection respectively. For contagion, there is a certain threshold above which a node flip, so it's a binary state. And for infection however, we'll get to see that an impact on topology can be modeled by looking at node neighbors' influence on the node state in a probabilistic way, with a certain slope which is βSi. When we look at a given graph of connectedness with associated adjacency matrix A, then we can write down the infection influence model's dynamics through a different set of differential equations, incorporating this adjacency matrix. Assume this is a bidirectional graph, so $A_{ij} = A_{ji}$.

Let's take the simple SI model. Then, for node I is not a general global population percentage anymore. It is the probability that node I is in the state of being susceptible at time t. So, we have to interpret this as the probability that node I is in state S. And that probability evolves over time, where the -β beta times the summation of all the neighbors:

$$\frac{dS_i(t)}{dt} = -\beta \sum_j A_{ij}[S_i(t)I_j(t)]$$

$$\frac{dI_i(t)}{dt} = \beta \sum_j A_{ij}[S_i(t)I_j(t)]$$

A_{ij} could be 0 or 1, depending whether j's neighbor of i or not. If it's zero, the rest of the term doesn't matter. This expression is in fact incorrect. It's wrong because, we can't actually write this down as such. What we actually need to write down is the joint probability. That mode I in the state of S and node J in the state of I. And what will that probability be? Then we need to know the probability that some neighbor of node J, other than node I itself, got to the current state of infected. Got to the current state by being infected and then making node J infected. So, we have to look at all the neighbors of node J, that induces this joint probability of node I and J in this state. But in order to do that we have to look at these neighbor's neighbors. And the trajectory of the state transition in the past have led to this state. As we can see this quickly gets out of hand, it's no longer tractable. So, what we're actually doing is to stop this propagation tracing back, and approximate.

The SI is not dependent on J. So, we can pull this out of the equation and start doing approximation.

$$\frac{dI_i(t)}{dt} = \beta S_i(t) \sum_j A_{ij} I_j(t) = \beta(1 - I_i(t)) \sum_j A_{ij} I_j(t)$$

And now we can write this down as a vector equation.

$$\frac{d\vec{I}(t)}{dt} = \beta \vec{A}\vec{I}(t)$$

With a probability of each known may finds itself in the infected state at time t. This is almost the exact same as before the scalar evolution, except now is a vector evolution with A. A is weighting by this adjacency matrix and representing the topology's impact. Again, we can apply the trick to represent the S, a weight of the sum of eigenvectors of the adjacency matrix as before. Skipping the derivation, we see that the solution of this vector of probabilities is

$$I(t) = \sum_k w_k(0) e^{\beta\lambda_k t} v^k$$

v^k are the eigenvectors of the adjacent symmetric A of this given draft index by k. $e^{\beta\lambda_k t}$ is our weighted by this exponential factor e to the β (the given disease spreading right times) times the corresponding eigenvalue. So, as time goes on, the largest eigenvalue will dominate effect.

Case study: measles

Measles is an infectious disease that causes about 1,000,000 deaths worldwide each year. But in developed countries the population is sufficiently vaccinated and it effects very few people. Because there was called a herding immunity: There is enough immunized population that the infection would not cause a pandemic.

What is a herd immunity then? We want $S(0) * \sigma < 1$ to prevent the infect population from flaring up. That means we need $S(0) < \frac{1}{\sigma}$. That means when you need initial recovered presumably through vaccination program population percentage $\mathrm{R}(0) > 1 - \frac{1}{\sigma}$.

From missiles people have estimated from previous outbreaks of the infection that the σ is very big about 16.67. That means this initial vaccinated rate $\mathrm{R}(0) = 94\%$. Which is big but not that close to 100%. However, the vaccination is only about 95% effective. So, that translates we actually need the vaccination rate $\mathrm{R}(0) = 99\%$ in order to achieve herd immunity in this condition.

Back in 1963 in the US, A measles population started drop because of the introduction of the measles vaccination but then still stayed around 50,000 people every year. In 1978, The US government tried to, make the immunization coverage wider to eliminate measles but it dropped to about 5,000, but stayed around there. In fact, sometimes it went up to 15,000. So, just increasing the coverage of immunization didn't help. In 1989, US Government introduced the two-dosage program. So, a child has to get one dosage when you're around one year old, and another around five years old before going to school all the time. And this time, the two-dosage program, which is much more expensive than the single dosage program, was able to achieve the vaccination rate, 99% needed to counter this large sigma of 16.65 for measles in order to satisfy this condition, and thereby, achieving Herd Immunity. Since then, the number of reported case of measles dropped to under 100 within a few years' time. This is a very interesting story, showing the power of the particular differential equation model we just developed for infection.

Extended Part

Random Walk

Let G be an undirected graph and each node has an edge to itself. Then for each node its degree is greater than zero. Now consider an object placed at node i. At each stage the object must move to an adjacent node. The probability that it moves to node j is $\frac{1}{\deg(\mathrm{I})}$, if (i, j) is an edge on G. Otherwise, the probability is 0.

Therefore, if we define $\mathrm{a}_{ij} = \frac{1}{\deg(\mathrm{I})}(i, j\ connected)/0(otherwise)$. Then A = (a_{ij}) is a Markov matrix. As each stage occurs, a sequence of adjacent node is produced. This sequence represents the position of the object at a given stage. Moreover, this sequence is a walk in the graph. We call such a walk a random walk on the graph G. Using Markov matrix, we see that the i, j entry of M^k represents the probability that a random walk of length k starting at node i and ends at node j. The steady-state vector will correspond to the probability of being at a given node after a sufficient number of random walks.
(Example in the python notebook.)

Measuring subgraph connectedness

Suppose we divide a graph into two parts and we want to measure how close are those nodes in same part.
Consider an undirected graph, and pick node i with degree d_i. Each of its neighbors, indexed by j, has a degree d_j. Let us pick one of node i's edges. The chance that on the other hand is node j is $\frac{d_j}{L}$(L is the total number of edges in the graph). Multiply the

number of links node I has and sum over node pairs (i, j) of the same part: $\sum_{ij\in same\ part}\frac{d_j d_i}{L}$.

Then we use a new notion Q to represent the modularity of a give graph:

$$Q = \frac{1}{L}\sum_{ij\in same\ part}(A_{ij} - \frac{d_j d_i}{L})$$

Q can be positive, in which case we have associative mixing: people of the same type connect more with each other (relative to a random drawing). It can be negative, in which case we have dissociative mixing: people of different types connect more with each other. With the normalization by L in its definition, we know $Q \in [-1,1]$.
(Example in python notebook.)

More about centrality

Page Rank

Page rank is an algorithm we have learnt for search engine. We can use it to measure the node importance because we can treat any node like a web page.
(The page rank algorithm is covered in this course, so I will skip the introduce.)
The Basic PageRank of a node can be interpreted as the probability that a random walk lands on the node after i random steps (Random walk is introduced at the begin of extend part).
Basic PageRank has the problem that, in some networks, a few nodes can suck up all the PageRank from the network. To fix this problem, Scaled PageRank introduces a parameter α, such that the random walker chooses a random node to jump to with probability $1-\alpha$. Typically we use α between 0.8 and 0.9. Then we can get a reasonable scaled page rank for each node.

Hubs and Authorities

This is another way to find central nodes in the network. Just like PageRank, this way was also developed in the context of how a search engine might go about finding important web pages given a query using the hyperlink structure of the web. So, the first step will be to find a set of relevant webpages. For example, web pages that contain the query string in the text of the web page or for some reason the search engine thinks these might be an important page to look at. These are potential authorities. Potential pages that are important given the query that the user submitted. This will be called the root set. And the next step will be to find all the web pages that link to any page in the root set, and these pages will be potential hubs. Hubs are pages that are not themselves necessarily relevant to the query that the user submitted, but they link to pages that are relevant. They're pages that are good at pointing at things that may be relevant.

Then we run HITS algorithm. The HITS algorithm just like PageRank works by computing k iterations and keeping track of the score for every node.

Basic steps:

1. Assign each node an authority and hub score of 1. (not repeat)
2. Apply the Authority Update Rule: each node's authority score is the sum of hub scores of each node that points to it.
3. Apply the Hub Update Rule: each node's hub score is the sum of authority scores of each node that it points to.
4. Normalize Authority and Hub scores: $auth(j) = \frac{auth(j)}{\sum_{i \in N} auth(i)}$
5. Repeat k times.

For most networks, as k gets larger, authority and hub scores converge to a unique value. (Example in python notebook.)

Comparing Centrality Measures

We've seen a number of different ways of finding central nodes in a network. Then we're going to look at an example where we compare how the different centrality measures that we've looked at, rank nodes differently.

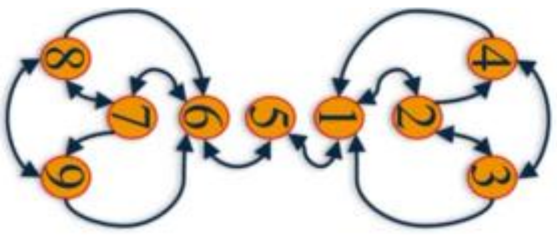


Figure 6

We're going to run the different algorithms that we looked at on this particular network Figure6. (Test in python notebook)

And then we got the result ranking table for these nodes by several algorithms.

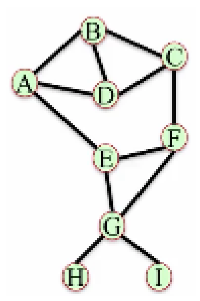
In-Deg	Closeness	Betweenness	PageRank	Auth	Hub
1	5	5	1	1	2
6	1	1	6	6	5
2	6	6	5	4	7
3	2	2	2	9	3
4	3	7	7	3	8
5	7	3	3	8	4
7	8	8	8	2	9
8	4	4	4	7	1
9	9	9	9	5	6

To summarize:

- In this example, no pair of centrality measures produces the exact same ranking of nodes, but have some commonalities.
- Centrality measures make different assumptions about what it means to be a central node. Thus, they produce different rankings.
- The best centrality measure depends on the context of the network one is analyzing.
- When identifying central nodes, it is usually best to use multiple centrality measures instead of relying on single one.

Link Prediction Problem

We've been looking at networks as dynamic structures that change over time. Now given a new problem: Given a fixed network, can we predict which edges are going to be formed in the future? This problem has a lot of applications, for example if we're Facebook and we want to create a friend recommendation algorithm or a friend recommender to tell people who they should friend. Then, basically we're solving this problem. We need to look at the current Facebook network and try to predict which new friendships are likely to form in the future.



Let's take this small network as an example. We take a specific pair of nodes and want to assess whether or not they're likely to connect. 7 ways to measures them is going to be introduced:

Measure 1: Common Neighbors

Triadic closure: the tendency for people who share connections in a social network to become connected themselves.

Triadic closure actually gives us a hint for what the first measure that we're going to look at. And that is very simple measure, just look at the number of common neighbors that the two nodes have.

Let’s define the number of common neighbors of nodes X and Y:

$$common\_neighbour(X, Y) = |N(X) \cap N(Y)|$$

where $N(X)$ is the set of neighbors of node X

$$common\_neighbour(A, C) = |\{B, D\}| = 2$$

We then compute the number of common neighbors for each pair of nodes that are not linked. Sort them like a list, and the pair with largest value is very likely to be linked in future. And, we will get (A, C) are mostly likely to connect each other.

Measure 2: Jaccard Coefficient

The next measure we're going to look at is called the Jaccard coefficient. And what it does is that it looks at the number of common neighbors but it normalizes it by the total number of neighbors of the two nodes.

The Jaccard coefficient of nodes X and Y is:

$$jacc\_coeff(X,Y) = \frac{|N(X) \cap N(Y)|}{|N(X) \cup N(Y)|}$$

$$jacc\_coeff(A,C) = \frac{|\{B,D\}|}{|\{B,D,E,F\}|} = 0.5$$

So, if we sort this $jacc\_coeff$ list, we now find that (I, H) have the highest Jaccard coefficient of 1. And that's because I and H are both connected to a single neighbor which is a common neighbor G.

Measure 3: Resource Allocation

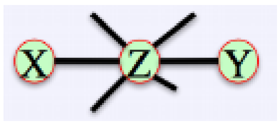
The intuition behind this measure is that it considers a fraction of a resource. For example, information or something else that a node can send to another node through their common neighbors.

The Resource Allocation index of nodes X and Y is:

$$res\_alloc(X,Y) = \sum_{u \in N(X) \cap u \in N(Y)} \frac{1}{|N(u)|}$$

$$res\_alloc(A,C) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

Example:



Z has n neighbors, X sends 1 unit to Z, Z distributes the unit evenly among all neighbors. So, Y receives 1/n of the unit.

And If we sort Resource Allocation list, we can see the Resource Allocation index of all pairs in this network. And if we look at the two pairs following, we see that in this case, (A, G) has a higher Resource Allocation index than (I, H). And that is because I, H has a common neighbor which is G. But G has a high degree of 4. Whereas A, G, they're common neighbor is E, and E has a slightly smaller degree of 3. So, that will give (A, G) a larger Resource Allocation index than (I, H).

Measure 4: Adamic-Adar Index

This measure is similar to resource allocation index, but with log in the denominator.
The Adamic-Adar index of nodes X and Y is:

$$adamic\_adar(X,Y) = \sum_{u \in N(X) \cap u \in N(Y)} \frac{1}{\log(|N(u)|)}$$

$$adamic\_adar(A,C) = \frac{1}{\log(3)} + \frac{1}{\log(3)} = 1.82$$

And again, we can compare (A, G) and (H, I) but in this case, it's going to be very similar to the Resource Allocation index. All we did was to put the log in the denominator.

Measure 5: Pref. Attachment

This measure is going to be the preferential attachment score. The preferential attachment model has the feature that nodes that have very high degree are more likely to get more neighbors. And so, what it does, is very simply, to take the product of the degree of the two nodes.
The preferential attachment score of nodes X and Y is:

$$pref\_attach(X,Y) = |N(X)||N(Y)|$$

$$pref\_attach(A,C) = 3 * 3 = 9$$

We've been following, (A, G) and (H, I). And we see that (A, G) has a higher preferential attachment score than (I, H) and that's because A has a degree of 3 and G has a degree of 4, which makes the preferential attachment score of 12. Whereas, I and H both only have one neighbor and so they have a score of 1.

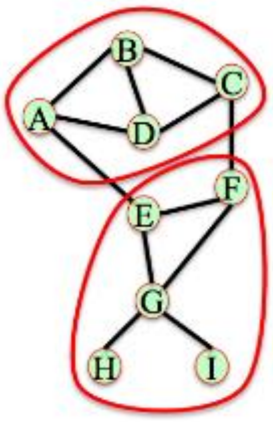
Community Structure

Next, we're going to look at two other measures, that take in to account the community structure of the network. And here we'll think of communities as sets of nodes and we'll make the assumption that each node belongs only to one community.

And what these two measures do, is they make the assumption that if two nodes belong to the same community, and they have many neighbors that also belong to the same community. Then they're more likely to form an edge, than if they had neighbors that belong to different communities, or if the two nodes themselves were in different communities. That's sort of the basic assumption that we make. And this allows us to compute new measures, and so let's go over two of them.

Measure 6: Community Common Neighbors

The sixth measures that we are going to look at is the number of common neighbors. But with a bonus for neighbors that belong to the same community as the two nodes we're looking at. This is called the Common Neighbor Soundarajan-Hopcroft score.



The Common Neighbor Soundarajan-Hopcroft score of nodes X and Y is:

$$cn\_soundarajan\_hopcroft(X,Y) = |N(X) \cap N(Y)| + \sum\nolimits_{u \in N(X) \cap N(Y)} f(u),$$

$$where\ f(u) = \frac{1, u\ in\ same\ community}{0, otherwise}$$

$$cn\_soundarajan\_hopcroft(A,C) = 2 + 2 = 4$$

$$cn\_soundarajan\_hopcroft(E,I) = 1 + 1 = 2$$

$$cn\_soundarajan\_hopcroft(A,G) = 1 + 0 = 1$$

Then we can sort score list and find which nodes have the highest score. And if we look at the two edges that we've been following, then we find that (I, H) has a score of 2, because their common neighbor G belongs to the same community as A do, whereas (A, G) has a score of 1.

Measure 7: Community Resource Allocation

The last measure we're going to look at is a measure that is similar to the Resource Allocation index but it only takes into account nodes that are in the same community as the two nodes we're looking at.

The Resource Allocation Soundarajan-Hopcroft score of nodes X and Y is:

$$ra\_soundarajan\_hopcroft(X,Y) = \sum\nolimits_{u \in N(X) \cap N(Y)} \frac{f(u)}{|N(u)|},$$

$$where\ f(u) = \frac{1, u\ in\ same\ community}{0, otherwise}$$

$$ra\_soundarajan\_hopcroft(A,C) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$$

$$ra\_soundarajan\_hopcroft(E,I) = \frac{1}{4}$$

$$ra\_soundarajan\_hopcroft(A,G) = 0$$

Then we can sort score list and find which nodes have the highest score. And here, we find that (I, H) has a score of 0.25, whereas (A, G) has score of 0. Again because (A, G), their common neighbor is not in the same community, at least one of A and G. And in this case again, we find that (I, H) has a higher score.

(All 7 measures are in python notebook)

Reference

[1] Mung Chiang: Networked Life, Cambridge University Press, 2012;

[2] M. E. J. Newman, Networks: An Introduction, Oxford University Press, 2010.

[3] NetworkX documents: https://networkx.github.io/documentation/stable/index.html