Introduction:

Analysis of social networks like Facebook, LinkedIn, Wikipedia, Authors publications etc is an interesting field of study because of its large, heterogeneous complex network structure. Link prediction is concerned with the problem of predicting the existence of links among nodes in a social network. It is a sub field of Social Network analysis, which uses different features of Social Network. Many Network analysis concentrate on the node. But the link prediction or recommendation is related to the study of links between the nodes in the social networks. We can pose the link prediction as a binary classification problem. It is used to determine whether or not a link exist between two given nodes. This lecture is going to consider a Graph Structure to analyze the problem of Link Prediction in Social Networks.

Problem Definition:

In Link prediction analysis mainly concentrate on the problem of predicting the future links between the nodes present in a social network. Since the Link Prediction problem deals with the social network, which are highly dynamic and sparse, the analysis of link prediction is a highly challenging task.

For instance, when we consider Facebook, which has over one billion nodes/members in its network, we have, (n choose 2) combination of nodes, for predicting the future links in the network. Thus, the probability of predicting the link is approximately equal to $(1/(\text{one billion})^2)$.

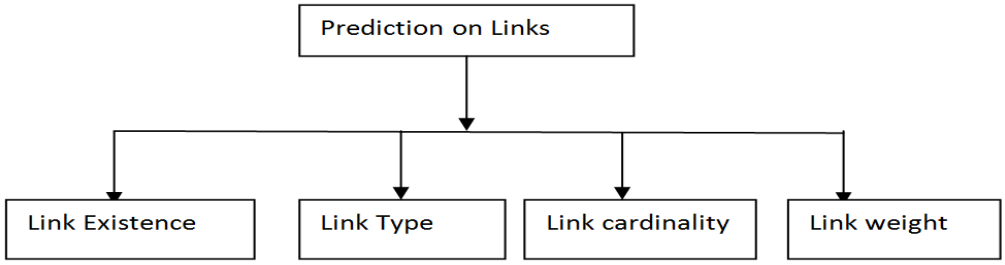
Also, the Link Prediction algorithm uses the criteria called '**Accuracy**' to evaluate to the accuracy of the algorithm, as it does not have any 'Mathematical Proof of Correctness' to evaluate the correctness of the algorithm. Accuracy is defined as "what percentage of the 'predicted links', will really exist in the future".

In link prediction we have four subcategories they are[4]

1) link existence problem: is there exist a link in future
2) link type problem: what type of link exist
3) link cardinality: more than one link between same pair of nodes in a social network
4) link weight: links have different weights associated with them

We first concentrate on the first two problems. We try to find the mechanism to find the link existence problem and the type of link. We can extend these two problems easily to the link cardinality and link weight problem.

Figure: Link Prediction Problem - Framework:



Link prediction and link recommendations have two main problems to find the recommendation links between two nodes in the social networks. They are sparseness and subtleness. The real social networks are very large. They have very large number of nodes and less number of edges between them. For example the Wikipedia has very large number pages in it, but not the links from it. The subtleness of the node features, we are not guaranteed that what extent we can use the existing features to predict the future links. The another issue is, we cannot always that the link prediction is going to exist with probability of 1, because of human nature. For example in Facebook, even if two persons has many common features (neighbors) and if they are strongly not willing to become friends, our prediction can be false. These are very rare and we will not consider them for our problem.

Formally link prediction is defined as: For a given graph G with E edges at given time t, we have to find the new edges E' after t` where t < t` in the graph G which are not present before time t. We can reduce this problem to link prediction for one particular node as predicting new links for a particular node.

The link prediction problem is associated with optimization problem. The optimization is in terms of the how many expected links or predicted links at time t for time t' by our algorithm are actually existed at time t'. For example at time t we predicted links for a node v after t', the evaluation of it only done after time t' by what percent of them are really created. Finding the features for link prediction is really a big task. Different methods are used them for the prediction are presented below.

Different Heuristics for Link Prediction:

In this section we present different heuristics that are used for the link prediction. There are many methods for defining some function called score which is used to find the future links. The different methods are

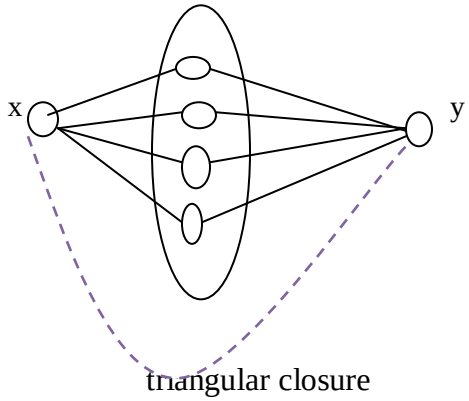
1) Node neighborhoods
2) Methods based on the ensemble of all paths

1) Node Neighborhoods:

We will use a notation Γ(x) to denote the set of neighbors of x in G(our social network as a graph). Many approaches are based on the idea that two nodes x and y are more likely to form a link in the future if their sets of neighbors Γ(x) and Γ(y) have large overlap. For example x and y represent authors with many colleagues in common, and hence are more likely to come into contact themselves. The score(x,y) is defined as the measure of probability of link between x and y. We define a score limit

i) ***Based on Common neighbors:***

The direct and common implementation of the idea for link prediction is to define score(x, y) := |Γ(x) ∧ Γ(y)|, Which is the number of neighbors that x and y have in common. The quantity in the context of collaboration networks, verify a correlation between the number of common neighbors of x and y at time t, and the probability that they will collaborate in the future.



In graph , if the nodes have more neighbors then they tend to form a triangular closure as shown above.

ii) ***Based on Jaccard's Coefficient***

The Jaccard coefficient is a commonly used similarity metric in information retrieval measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. The feature here is the common neighbor and the score is as below.

$$\text{score}(x,y) = \frac{|\Gamma(x) \wedge \Gamma(y)|}{|\Gamma(x) \vee \Gamma(y)|}$$

*iii)****Adamic/Adar***

Adamic and Adar consider a related measure, in the context of deciding when two personal home pages are strongly "related." To do this, they compute features of the pages, and define the similarity between two pages to be

$$\sum_{k:\ \text{feature shared by x,y}} \frac{1}{\log(\text{frequency}(k))}$$

Now the feature is common neighbor so the equation becomes

$$\text{score}(x,y) = \sum_{k \in \Gamma(x) \wedge \Gamma(y)} \frac{1}{\log \Gamma k}$$

The preferential attachment is defining score as is proportional to the number of neighbors to the present node.

2) Methods based on the grouping of all paths

It is depends on the number of shortest paths between the two nodes.

*i)**Collection of all paths(Katz)***

We directly sums all the paths between x and y to predict the link between them. We define a variable α which is less than 1 and greater than 0. We define score as

$$\text{score}(x,y): = \sum_{l=1}^{\beta} \alpha^{l} . |\text{paths}_{x,y}^{<l>}|.$$

where paths $|\text{paths}_{x,y}^{<l>}|$ is the set of all length - l paths from x to y.

The α plays an important role in score calculation. If we choose very less α, it is more likely to be considering neighborhood. because as the value l increases α^{l} value decreases drastically if $\alpha << 1$. In weighted and un weighted case the $|\text{paths}_{x,y}^{<l>}|$ defined as below

(1) unweighted, in which $|\text{paths}_{x,y}^{<l>}|=1$ otherwise 0.

(2) weighted, in which $|\text{paths}_{x,y}^{<l>}|$ is the number of times that x and y have collaborated.

*ii) **SimRank:***

This method is depends on features of the node. If a node y has all similar feature of x, then they are more likely to be friends.

The score based on the simrank is defined as:

similarity(x,x)= 1 and

$$\text{similarity}(x,y) = \gamma. \quad \frac{\sum_{a \in \Gamma(x)} \sum_{b \in \Gamma(y)} \text{similarity}(a,b)}{|\Gamma(x)|.\ |\Gamma(y)|}$$

iii) Hitting time, Page Rank, and variants

Link prediction based on random walk increasingly a better method for the link prediction. A random walk on graph G starts at a node x, and iteratively moves to a neighbor of x chosen uniformly at random. We find the ranks based on how likely the movement is towards the predicting node. We assign ranks based on the random walks as similar to the page ranks.

The hitting time $H_{x,y}$ from x to y is the expected number of steps required for a random walk starting at x to reach y. We perform it in both ways from x to y and y to x for the symmetry. If it is less, we give good score(x,y) as it is a converse to the prediction.

Optimization Problem: The 'RANDOMWALK' Algorithm

A new algorithm random walks which uses the nodes attributes as well as graph structure properties to define the score (or relevance) is proposed by University of Illinois in 2011. Which uses the attributes to connect the even the large distance people.

The below is sample social network where the circle type nodes are the persons and square type are the features/attribute nodes.

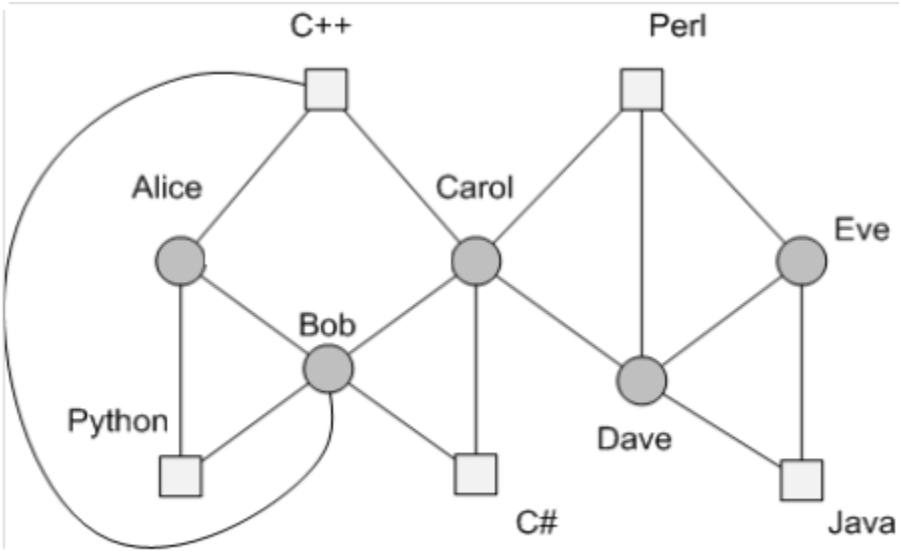


Figure: Social Network with Person nodes and Attribute nodes

First we construct the new graph by considering the features, we present the different types of features and how they will impact the link prediction in randomized walking algorithm for the prediction, the algorithm, and we describe how it is likely to optimize the link prediction in last we are going to explain the algorithm using an example graph which is shown above.

- **Graph Construction:**

Given the original social graph G(V, E), we construct a new graph G'(V',E'), augmented based on G. Specifically, for each node in graph G, we create a corresponding node in G' , called person node. For each edge in E in graph G, we create a corresponding edge in G' . For each attribute a, we create an additional node in G' , called attribute node. V' = Vp ∪ Va (Vp union Va) where Vp is the person node set and Va is the attribute node set. For every attribute of a person, we create a corresponding edge between the person node and the attribute node.

The edge weights in G' are defined by the uniform weighting scheme. The weight w(a,p) of the edge from attribute node a to person node p is defined as follows.

$$w(a,p)=\frac{1}{|Np(a)|}$$

where Np(a) denotes the set of person nodes connected to attribute node a.

In a social network the below are some features which are used for the link recommendation.

1) **Rarity:** The rare attributes are likely to be more important, whereas the common attributes are less important. E.g., only Alice and Bob love Hiking, but thousands of people, including Alice and Carol, are interested in Football.

2) **Social Influence:** The attributes shared by a large percentage of friends of a particular person are important for predicting potential links for that person. E.g., most of the people linked to Alice like Football, and Bob is interested in Football but Carol is not.

3)Homophily: Two persons who share more attributes are more likely to be linked than those who share fewer attributes. E.g., Alice and Bob both like Football and Tennis, and Alice has no common interest with Carol.

4) **Social Closeness:** The potential friends are likely to be located close to each other in the social graph. E.g., Alice and Bob are only one step away from each other in social graph, but Alice and Carol are five steps apart.

5) **Common Friendship:** The more neighbors two persons share, the more likely it is that they are linked together. E.g., Alice and Bob share over one hundred friends, but Alice and Carol have no common friend.

6) **Preferential attachment:** A person is more likely to link to a popular person rather than to a person with only a few friends. E.g., Bob is very popular and has thousands of friends, but Carol has only ten friends.

- **Terminology:**

Given person node p, attribute node a connected to p and person node p' connected to node p, the edge weight w(p,a) from person node p to attribute node a and the edge weight w(p,p') from person node p to person node p' are defined as follows.

$$w(p,a)=\begin{cases} \frac{\rho}{|Na(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) > 0 \\ \frac{1}{|Na(p)|} & \text{for } Np(p) = 0 \text{ and } Na(p) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w(p,p')=\begin{cases} \frac{1-\rho}{|Np(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) > 0 \\ \frac{1}{|Np(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) = 0 \\ 0 & \text{otherwise} \end{cases}$$

The larger ρ is, the more the algorithm uses attribute properties for link recommendation. Specifically, if ρ = 1, the algorithm makes use of the attribute features only. If ρ = 0, it is based on structural properties only.

- **The Algorithm - Step-by-Step Analysis:**

Random walks (Graph: A social graph G(V, E), person attribute profile, input person node: p^*, and two parameters ρ and α) are the inputs to the Random Walk Algorithm. Let us analyze the algorithm, step by step, as follows.

{

1. **Step1:** Construct a new graph G'(V',E') based on G(V,E) and attribute profiles, where V'= Vp υVa, Vp is a person node, and Va is an attribute node.
2. **Step2:** Set the edge weights with ρ in the augmented graph G' using below Equations

$$w(p,a)=\begin{cases} \frac{\rho}{|Na(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) > 0 \\ \frac{1}{|Na(p)|} & \text{for } Np(p) = 0 \text{ and } Na(p) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w(p,p')=\begin{cases} \frac{1-\rho}{|Np(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) > 0 \\ \frac{1}{|Np(p)|} & \text{for } Np(p) > 0 \text{ and } Na(p) = 0 \\ 0 & \text{otherwise} \end{cases}$$

3. **Step3:** Iterate to update r_p and r_a with according to below Equations for each person node and attribute node

$$r_p = (1-\alpha)\sum_{p' \in Np(p)} w(p', p) rp' + (1-\alpha) \sum_{a' \in Na(p)} w(a', p) ra' + \alpha rp(0)$$

$$r_a = (1-\alpha)\sum_{p' \in Np(a)} w(p', a) rp'$$

until the convergence.

$r_p(0) = 1$ if $p = p^*$, $r_p(0) = 0$ otherwise.

where r_p is the link relevance of person p with regard to p^*.

r_a is the relevance of attribute a with regard to p^*.

4. **Step4:** Let r_p* is the stationary value of $r_p^{(t)}$. Output the nodes in V_p based on decreasing order of r_p^* where the nodes connected to p* in G are connected.
5. **Step5:** Return a ranked list of recommended candidates (to be linked with) for person p*

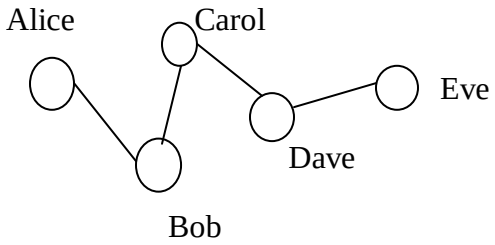
}

- **Time complexity:**

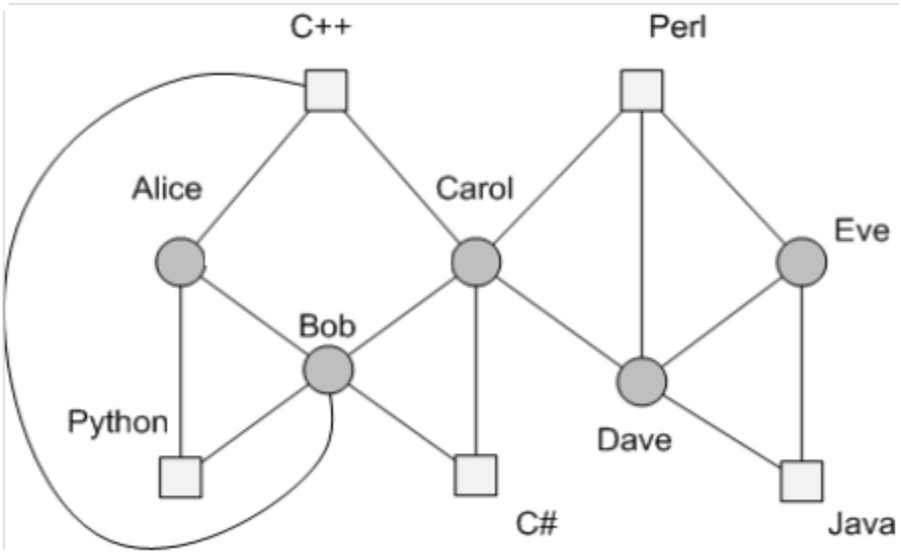
- In the first step, graph creation takes time for looking into each node and forming new edges accordingly, hence, the time complexity is O(V'E').
- The second step takes at most O(V') time for calculation of edge 'weight function' for each node with all other nodes to which it is connected to.
- In the third step, time complexity, depends on the convergence. If the convergence is order of K then the time complexity is $O(KV'^2)$. That is, for each vertex we are calculating the r_p, in summation we are also visiting each vertex which are in p adjacent list.
- The fourth step takes O(V'logV') time for sorting, say, we quick sort.

So total time complexity of the algorithm is $O(KV'^2+V'logV'+E'V'+V')$.

- **Explaining algorithm using an example:**



In the above graph Alice, Bob, Carol Dave, Java, Eve are the person nodes and let the c++, c#, Java, Perl, Python are the feature. Using algorithm we first design a new graph G' as below. Let us predict the links for Alice (A) in future.



In step 2 we will compute the edge weights as given : let $\rho=0$ for simplicity (we are not taking into consideration, the attribute nodes, at all) and $\alpha=0.5$.

For each edge in G' we will compute the W(p,p') using the above equation.

for edge W(A,B)= 1/1=1

for edge W(B,A)=1/2=0.5

for edge W(B,C)=1/2=0.5

for edge W(C,B)=1/2=0.5

for edge W(C,D)=1/2=0.5

for edge W(D,C)=1/2=0.5

for edge W(D,E)=1/2=0.5

for edge W(E,D)=1/1=1

In the next step 3, we calculate r_p and r_a for each node until the convergence hold. $r_p^{(0)}=1$ if $p = p^*$ .Otherwise, $r_p^{(0)}=0$. Here simply we define convergence is that for all nodes p which are connected to pp , r_p should be greater than all other non connected nodes p'. $r_p > r_p'$

Now for B, C, D, E

$r_a=1$

$r_b= 0.5(0.5+0)=0.25$

$r_c=0.5(0.25*0.5)=0.125$

$r_d=0.5(0.125*0.5+0)=0.03125r$

$r_e=0.5(1*0.03125)=0.015625$

The convergence is satisfying and let the predicting edges should have value at least greater than some value let r_b / 10, then we output the C and D as output for the link prediction from A.

by observing the graph, the B is already connected to A. The C and D are near to A than E. As this applies same for attributes, if $\rho>0$ then also it gives the same result because of linearity.

Some Properties of the above Algorithm:

(1) If one attribute is rare, there are fewer out links for the corresponding attribute node. Therefore, the weight of each out link is larger and the probability of a random walk originating from a person and reaching the other person node via this attribute node is larger.

(2) If two persons share more attributes, the corresponding person nodes in the graph will have more connected attribute nodes in common. Therefore, the random walk probability from one person node to the other via those common attribute nodes is high.

(3) If two persons share many friends, these two person nodes have a large number of common neighbors in the graph. Therefore, the random walk probability from one person node to the other is high.

(4) If one attribute is shared by many of the existing linked persons of the given person, the random walk will pass through the existing linked person nodes to this attribute node.

(5) If two person nodes are close to each other in the graph, the random walk probability from one to the other is likely to be larger than if they are far away from each other.

Applications:

- Link prediction Algorithm, is applicable to a wide areas like Facebook, bibliographic domain, molecular biology, criminal investigations and recommender systems.
- 'The Facebook - Most Favorite Social Network' in particular, essentially uses the 'Supervised Random Walks' method which is similar to the Randomwalk algorithm, discussed in this lecture. This algorithm, basically provides the link recommendation feature in Facebook.
- The other applications which uses the 'Supervised Random Walks' are in Crime Detection, Co-authorship suggestion, etc,. in the Social Networks.
- The Random Walk algorithm, discussed in this paper, is essentially used for 'Movie Recommendation'. The algorithm, based on the network structure, recommends, movies to the users.

REFERENCES:

[1] http://cs.stanford.edu/people/jure/pubs/linkpred-wsdm11.pdf

[2] http://www.stanford.edu/class/cs224w/proj/skurian_Finalwriteup_v2.pdf

[3]http://svolkova.weebly.com/uploads/1/6/7/1/1671882/cis_830_final_project_-_link_prediction_in_social_networks_-_part_1.pdf

[4] http://www.cs.cornell.edu/home/kleinber/link-pred.pdf

[5] http://www.cs.uiuc.edu/~hanj/pdf/asonam10_zyin.pdf

[6]http://delivery.acm.org/10.1145/2070000/2063741/p1147-li.pdf?ip=128.138.65.249&acc=ACTIVE%20SERVICE&CFID=98940813&CFTOKEN=40692190&__acm__=1335247850_29f81d2d040765c03008e8d19a282858

[7]http://blogs.cornell.edu/info2040/2011/10/31/random-walk-algorithms-for-movie-recommendation/