January 2015

Link Prediction in Dynamic Weighted and Directed Social Network using Supervised Learning

Ricky Laishram
Syracuse University

Abstract

Link Prediction is an area of great interest in social network analysis. Previous works in the area of link prediction have only focused on networks where the links once created cannot be removed. In many real world social networks, the links should be assigned strengths; for example, the strength of a link should decrease over time, if there are no interactions between the two nodes for a long time and increase if the two nodes interact often. In this thesis we modify existing methods of link prediction to apply to weighted and directed networks. The features, developed in previous works for unweighted and undirected networks, are extended to apply to networks whose links have weight and direction, and algorithms are developed to calculate them efficiently. These network features are used to train an SVM classifier to predict which nodes will be connected by a link and which links will be broken in the future. The results obtained using Twitter @-mention network demonstrate that the method developed in this thesis is very effective.

Link Prediction in Dynamic Weighted and Directed Social Network using Supervised Learning

by

Ricky Laishram

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Syracuse University

August 2015

Copyright

Acknowledgments

I am very grateful to Dr. Chilukuri K. Mohan who introduced me to the area of social network analysis and encouraged me to work on the current research. Without his help and encouragements, I might not have even considered doing this research.

I am also extremely grateful to Dr. Kishan Mehrotra for guiding me in my research and providing valuable feedback while writing this thesis. Without his guidance and help this thesis would not have been possible.

Contents

1 Introduction

A social network is a structure containing a set of actors and relationships; actors are represented as nodes, and relationships are represented as edges. For example, the edges may represent communications between the nodes, spread of diseases from one node to another, etc. Two basic changes can happen in a social network - addition or removal of nodes, and addition or removal of edges.

In this thesis, I propose a method to predict addition or removal of edges. This particular problem is called the Link Prediction Problem in Social Network. David Liben-Nowell and Jon Kleinsberg define the Link Prediction Problem as follows: "to what extent can the evolution of a social network be modeled using features intrinsic to the network itself?"[2].

In all the previous works addressing this problem, the network model is such that a link once created is not destroyed. However, in many real world networks, the links are not permanent. For example in a Twitter @-mention network, if two users have not communicated for a long time, the link between them should be removed. To address this problem, I consider the following:

1. A network model where the link weights change with time. In this model each new interaction between nodes strengthens the link between them, whereas lack of interaction decreases the strength of the link. An appropriate threshold is set such that if the weight of a link is below the threshold, the link is removed.
2. Link prediction in directed and weighted networks. I modify features, used in undirected and unweighted networks, to apply to weighted and directed networks.
3. Efficient algorithms to incrementally update the network feature scores are also developed.
4. The problem of class imbalance is handled by using the ensemble SVM method.

This thesis is organized as follows:

1. The problem of edge deletion is addressed in Chapter 3.

2. Feature modifications to address the weighted directed networks is addressed in Chapter 4.

3. Efficient algorithms to compute the feature scores is described in Chapter 5.

4. In Chapter 6, link prediction problem is formulated as a classification problem, and solved using SVM where class imbalance is also addressed.

In the next chapter, I examine the previous works done by other researchers and examine the shortcomings of the existing methods.

2 Previous Works

Many researchers have worked on the problem of link prediction in social networks in unsupervised as well as supervised context. In this chapter we look at some of the previous works done in the area of link prediction and their shortcomings.

2.1 Unsupervised Link Prediction

Liben-Nowell et al. [1] examined if links in a network can be predicted by considering only features intrinsic to the network. They observed a collaboration network at time $[t_0, t_1, t_2, \ldots T]$. In this network the set of nodes V remains fixed but edges are added over time. The network over time are $< V, E_0 >$, $< V, E_1 >$, $< V, E_2 > \ldots < V, E_T >$ such that $E_0 \subseteq E_1 \subseteq E_2 \ldots \subseteq E_T$. Out of these networks, they considered two networks $G_1 =< V, E_t >$ and $G_2 =< V, E_T - E_t >$ for time t such that $0 < t < T$, and let $N = |E_T - E_t|$. In other words, G_2 contains only the links that were formed after time t. They used G_1 as the training network and G_2 is assumed to be unknown. Their aim was to use G_1 to predict the edges in G_2.

They considered the following intrinsic features: common neighbors, preferential attachment count, Adamic-Adar Index, Katz Score, Hitting Time, Rooted PageRank and Commute Time for this evaluation.

Their method is described as below:

1. For all possible node pairs u and v such that $u, v \in V \wedge (u, v) \notin E_1$, the feature score $s_i(u, v)$ is calculated for all the features, $i = 1, 2, 3, \ldots 7$. Let $S_i = \{s_i(u, v) : \forall (u, v) \in V \wedge (u, v) \notin E_1\}$.

2. For $i = 1, 2, \ldots 7$, the elements of S_i are sorted in descending order of magnitude and the top N pairs of nodes (u, v) with highest $s_i(u, v)$ are selected. Let S_i^* represent this list of nodes. A link is predicted between (u, v) for all $(u, v) \in S_i^*$.

The performance P_i for a feature i is considered as the number of correctly predicted links. A link (u,v) in S_i^* in said to be correctly predicted if it is also in the set $E_T - E_t$.

$$P_i = |\{(u,v) : (u,v) \in S_i^* \wedge (u,v) \in (E_T - E_t)\}| \tag{1}$$

Liben-Nowell et al. compared the performances P_i against a random prediction. To generate randomly predicted set of N links in $E_T - E_t$, a set of N node pairs (u,v) is selected such that $u,v \in V \wedge (u,v) \notin E_t$. Let us denote this list by R, and Q be the number of correctly identified edges.

$$Q = |\{(u,v) : (u,v) \in R \wedge (u,v) \in (E_T - E_t)\}| \tag{2}$$

Liben-Nowell et al. applied their method to the collaboration network obtained from the ArXiv dataset. In this network, $|V| = 1253$ and $|E_T - E_t| = 1150$.

Feature	P_i/Q
Adamic-Adar	54.8
Katz Score	54.8
Common Neighbors	41.1
Hitting Time	23.8
Rooted PageRank	42.3
Commute Time	15.5

Table 1: P_i/Q for different features

Liben-Nowell et al. found that the ratio P_i/Q is in the range $[15,55]$ (Table 1) for all the features. This demonstrates that the performance of the predictions based on network intrinsic features is significantly higher than a random predictor. From this observation, they concluded that a network contains intrinsic information to predict links.

The method used by Liben-Nowell et al. has two drawbacks that make it difficult to apply in real world networks.

1. In real world networks, we do not know the number of links that will be created in the future.

So calculation of N will not be possible.

2. Even though their method outperforms random prediction by a huge margin, the ratio of correctly predicted links to the actual number of new links (P_i/N) is very low. In their work, Adamic-Adar Index and Katz Score were the best performing features, but the associated predictions are only 8% correct i.e. $P_i/N \leq 0.08$. The P_i/N ratios for all the features are shown in Table 2.

Feature	P_i/N
Adamic-Adar	0.080
Katz Score	0.080
Common Neighbors	0.060
Hitting Time	0.034
Jaccard's Coefficient	0.062
Rooted PageRank	0.060

Table 2: P_i/N for different features

2.2 Supervised Link Prediction

Al Hasan et al. [2] explored the use of supervised learning methods (Decision Tree, Naive Bayes, SVM, Multi-Layer Perceptron) to predict links in the co-authorship networks obtained from the BIOBASE and DBLP datasets.

As in Section 2.1, Al Hasan et al. considered two networks G_1 and G_2. The network G_1 is the training network and G_2 is the network for which links are to be predicted. They transform the link prediction problem into a two-class classification problem by assigning class labels to all pairs of nodes as:

$$class(u,v|G_t) = \begin{cases} 1, & \text{if } (u,v) \in E_t \qquad \text{(positive class)} \\ 0, & \text{if } (u,v) \notin E_t \qquad \text{(negative class)} \end{cases} \tag{3}$$

Using G_1, they calculate the feature scores associated with each pair (u,v). Let F_1 denote the

feature scores for all n-features from G_1,

$$F_1 = \{(s_1(u,v|G_1), s_2(u,v|G_1), ..., s_n(u,v|G_1)) : u,v \in V\} \tag{4}$$

Here $s_i(u,v|G_t)$ is a feature score between nodes u and v in network G_t for a feature i. They considered features that are both intrinsic to the network (sum of neighbors count, sum of secondary neighbors count, clustering index, shortest path) and extrinsic to the network (based on papers published - sum of keywords, sum of classification code, sum of papers, keyword match count).

Let C_1 be the set of all the class labels for network G_1. Then (F_1, C_1) forms the training data. Likewise testing data is calculated from G_2 for all pairs of nodes that do not have a link between them in G_1, i.e.

$$F_2 = \{(s_1(u,v|G_2), s_2(u,v|G_2), ..., s_n(u,v|G_2)) : u,v \in V \wedge (u,v) \notin E_t\} \tag{5}$$

The classifiers are trained using the training data, and used to predict the class labels for the testing data F_2.

Al Hasan et al. achieved very good results with their method. All the supervised learning algorithms they used have accuracy of more than 85%. However, there are some shortcomings with their method.

1. In a real world network, we do not know the network G_2. So calculation of the testing data F_2 (in equation 5) will not be possible.

2. Their method uses network extrinsic features, e.g., derived from similarity of the papers published. In other applications, it may not be possible to use extrinsic features.

2.3 Supervised Link Prediction in Sparse Network

The network that Al Hasan et al. [2] used in their work has positive and negative class of approximately equal size. However, many real world networks are sparse - the size of the positive class is much smaller compared to that that of the negative class. Lichtenwaltert et al. [3] improved upon the works of Al Hasan et al. as follows:

1. In their method, they do not use any extrinsic features. They used only network intrinsic features: common neighbors count, Katz Score, Jaccard's Coefficient, PropFlow and Preferential Attachment.

2. They overcame the problem of class imbalance by oversampling from the positive class and undersampling from the negative class to generate the training data. They used SVM to classify the test data into positive or negative class, and compared the results to those obtained using unsupervised method (Section 2.1).

Lichtenwaltert et al. [3] obtained very good results with their method (AUC > 70%) on sparse networks. This demonstrates two important ideas:

1. The network extrinsic features, like the ones used by Al Hasan et al. [2] are not always necessary in a supervised learning method of link prediction.

2. Supervised link prediction performs very well in sparse networks after adjusting for class imbalance.

3 Network Model with Weighted and Directed Links

In this chapter, we describe a network model where the links are assigned weights, representing its strength, and direction. Consider a network which has a link between nodes u and v at time t. If there is an interaction between u and v at time $(t+1)$, the strength of the link between u and v should increase. If there is an interaction between two nodes, the weight of the link between them is increased. Otherwise, the weight of the link between them is decreased by a factor δ. A weight threshold θ is defined such that a link is considered broken if its weight is lower than θ.

For example, in a Twitter @-mention network, an interaction is a tweet sent by one node to another. Each link in the network has a strength associated with it. Depending upon the strength of a link, from node u to v, the link could be destroyed assuming there are no more interactions between u and v [4, 5] for a long time.

Suppose that $G_t =< V, E_t >$ is the network at time t. If $I_{t,t+1}$ is the set of node pairs that interact with each other in the time interval $[t, t+1]$, the set of links at time $t+1$ is generated using the method described below.

1. A set of links is generated:

$$T_{t+1} = E_t \cup \{(u,v) : (u,v) \in I_{t,t+1} \wedge (u,v) \notin E_t\} \tag{6}$$

2. The weight of each link in T_{t+1} is calculated:

$$w(u,v|t+1) = \begin{cases} \delta \cdot w(u,v|t) + 1, & \text{if } (u,v) \in I_{t,t+1} \\ \delta \cdot w(u,v|t), & \text{if } (u,v) \notin I_{t,t+1} \end{cases} \tag{7}$$

3. The links with weight less than θ are removed from the set T_{t+1}. The resulting set E_{t+1} is

the set of links in the network at time $(t+1)$.

$$E_{t+1} = T_{t+1} - \{(u,v) : (u,v) \in T_{t+1} \wedge w(u,v|t+1) \leq \theta\} \tag{8}$$

The value of θ and δ depend on the network, and $0 < \theta < 1$ and $0 \leq \delta \leq 1$.

In the next chapter, we will describe how the existing feature scores are modified to apply to the network described in this chapter.

4 Modification of Feature Scores for Weighted and Directed Network

As mentioned in Chapter 2, previous works on link prediction have been performed on unweighted and undirected networks. So, most of the feature scores are independent of the link weight and direction. In this chapter we will extend the existing feature scores to apply to weighted and directed networks.

Notations:

1. In an undirected network, the set of neighbors of a node u is represented by $\Gamma(u)$, and $w(u,v)$ is the weight of the link from u to v.
2. In a directed network, a node u has two types of neighbors - neighbors with links that are directed away from u, represented by $\Gamma_o(u)$, and neighbors with links that are directed to u, represented by $\Gamma_i(u)$.
3. The set $\rho(u,v|l)$ denotes the set of paths of length l from node u to v,

In the following sections, we modify the features discussed in Chapter 2 for weighted and directed networks:

4.1 Common Neighbor Score

Newman's work [8] on collaboration networks shows that there is a positive correlation between the number of common neighbors of two nodes and the probability that there will be a link between them in the future.

In an undirected network, the Common Neighbor Score of two nodes u and v is given by,

$$CN(u,v) = |\Gamma(u) \cap \Gamma(v)| \tag{9}$$

In a directed network, there are two Common Neighbor Scores - one based on the in-neighbors and the other based on the out-neighbors.

$$CN_o = |\Gamma_o(u) \cap \Gamma_o(v)| \tag{10}$$

$$CN_i = |\Gamma_i(u) \cap \Gamma_i(v)| \tag{11}$$

To take into account the weight of the links, the sum of the average weights of the links from the nodes u and v to the common neighbors is taken instead of simply the number of neighbors. So we propose to extend the Common Neighbor Score as follows:

$$\boxed{CN_{o,weighted} = \sum_{z \in (\Gamma_o(u) \cap \Gamma_o(v))} \frac{w(u,z) + w(v,z)}{2}} \tag{12}$$

$$\boxed{CN_{i,weighted} = \sum_{z \in (\Gamma_i(u) \cap \Gamma_i(v))} \frac{w(z,u) + w(z,v)}{2}} \tag{13}$$

4.2 Preferential Attachment Score

Work by Barabasi et al. [10] on collaboration networks suggest that there is a positive correlation between the probability of collaboration between two nodes and the product of the number of neighbors of the two nodes.

In an undirected and unweighted network, the Preferential Attachment Score between two nodes u and v is given by,

$$PA(u,v) = |\Gamma(u)| \cdot |\Gamma(v)| \tag{14}$$

In a directed network, since there are two neighbor sets, the two Preferential Attachment Scores will be given by,

$$PA_i(u,v) = |\Gamma_i(u)| \cdot |\Gamma_i(v)| \tag{15}$$

$$PA_o(u,v) = |\Gamma_o(u)| \cdot |\Gamma_o(v)| \tag{16}$$

In a directed weighted network, to incorporate the weights of the links into the Preferential Attachment Score, it can be redefined as the product of the average link weights of the two neighbors of the two nodes, written as,

$$\boxed{PA_{i,weighted}(u,v) = \left(\frac{\sum_{z\in\Gamma_i(u)} w(z,u)}{|\Gamma_i(u)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_i(v)} w(z,v)}{|\Gamma_i(v)|}\right)} \tag{17}$$

$$\boxed{PA_{o,weighted}(u,v) = \left(\frac{\sum_{z\in\Gamma_o(u)} w(u,z)}{|\Gamma_o(u)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(v)} w(v,z)}{|\Gamma_o(v)|}\right)} \tag{18}$$

4.3 Adamic-Adar Index

Adamic et al. [9] studied web pages and developed a measure to calculate the similarity between web pages. They found that rare features are more important than common ones while calculating the similarity between web pages. Their work can be easily applied to social networks - nodes take the place of web pages and links take the place of web links.

In an undirected network, the Adamic-Adar Index of two nodes u and v is given by,

$$AA(u,v) = \sum_{z\in(\Gamma(u)\cap\Gamma(v))} \frac{1}{\log(|\Gamma(z)|)} \tag{19}$$

If the network is directed, we will have two Adamic-Adar Index values based on the two neighbor sets.

$$AA_o(u,v) = \sum_{z\in(\Gamma_o(u)\cap\Gamma_o(v))} \frac{1}{\log|\Gamma_o(z)|} \tag{20}$$

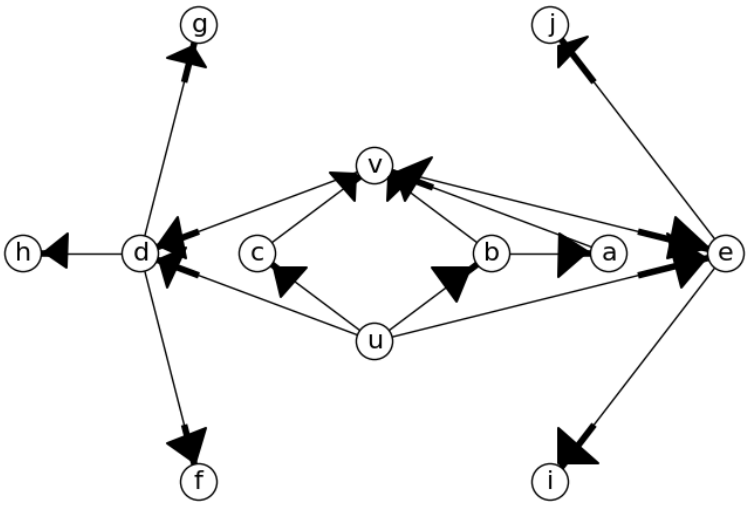


Figure 1: An example graph

$$AA_i(u,v) = \sum_{z \in (\Gamma_i(u) \cap \Gamma_i(v))} \frac{1}{\log |\Gamma_i(z)|} \tag{21}$$

To illustrate, consider the network in Figure 1. If we want to calculate the out-Adamic-Adar Index between nodes u and v, the contributions of the common nodes d and e will be given by,

$$\gamma(d) = \frac{1}{\log(w(d,f) + w(d,g) + w(d,h))} \tag{22}$$

$$\gamma(e) = \frac{1}{\log(w(e,i) + w(e,j))} \tag{23}$$

The contributions of $\gamma(d)$ and $\gamma(e)$ towards the final out-Adamic-Adar Index need to take into account the average weights of links to node d and e respectively.

$$AA_{o,weighted}(u,v) = \frac{1}{2}\left(\frac{w(u,d)+w(v,d)}{\gamma(d)} + \frac{w(u,e)+w(v,e)}{\gamma(e)}\right) \tag{24}$$

Equation 24 can be generalized as,

$$\boxed{AA_{o,weighted}(u,v) = \frac{1}{2}\left(\sum_{z\in(\Gamma_o(u)\cap\Gamma_o(v))} \frac{w(u,z)+w(v,z)}{\log(\sum_{x\in\Gamma_o(z)} w(z,x))}\right)} \tag{25}$$

Similarly, the in-Adamic-Adar Index can be derived to be,

$$\boxed{AA_{i,weighted}(u,v) = \frac{1}{2}\left(\sum_{z\in(\Gamma_i(u)\cap\Gamma_i(v))} \frac{w(z,u)+w(z,v)}{\log(\sum_{x\in\Gamma_i(z)} w(x,z))}\right)} \tag{26}$$

4.4 Katz Score

Katz [11] defined a similarity measure based on the paths between two nodes. In the measure that Katz defined, the contribution of shorter paths to the final score is higher than that of the longer paths.

Mathematically, the Katz Score of (u,v) is given by,

$$KS(u,v) = \sum_{l=1}^{\infty} \beta^l \cdot |\rho(u,v|l)| \tag{27}$$

where β is the damping factor $(0 \leq \beta \leq 1)$ and $|\rho(u,v|l)|$ is the number of paths of length l from node u to v. Liben-Nowell et al. [1] have shown that performance of the Katz Score is best for $\beta \leq 0.005$.

As seen in equation 27, the Katz Score depends on only the path between the nodes. So the same equation can be applied to directed networks. However, it still needs to be adjusted for link weights. To account for the link weights, the number of paths in equation 27 is replaced by the

sum of average link weights.

Consider the network in Figure 1. The sets of paths from node u to v are,

$$\rho(u,v|1) = \phi \tag{28}$$

$$\rho(u,v|2) = \{\{(u,b),(b,v)\},\{(u,c),(c,v)\}\} \tag{29}$$

$$\rho(u,v|3) = \{\{(u,b),(b,a),(a,v)\}\} \tag{30}$$

The contribution of each path set towards the final Katz Score is as follows:

$$\gamma(1) = \beta^1 \cdot 0 \tag{31}$$

$$\gamma(2) = \beta^2 \cdot \left(\frac{w(u,b)+w(b,v)}{2} + \frac{w(u,c)+w(c,v)}{2}\right) \tag{32}$$

$$\gamma(3) = \beta^3 \cdot \left(\frac{w(u,b)+w(b,a)+w(a,v)}{3}\right) \tag{33}$$

The Katz Score from node u to v is then given by:

$$KS_{weighted}(u,v) = \gamma(1) + \gamma(2) + \gamma(3) \tag{34}$$

Equation 34 can be generalized as,

$$\boxed{KS_{weighted}(u,v) = \sum_{l=1}^{\infty} \beta^l \cdot \sum_{p \in \rho(u,v|l)} \frac{\sum_{(x,y)\in p} w(x,y)}{l}} \tag{35}$$

4.5 Shortest Path Score

In an undirected and unweighted network, the Shortest Path score of (u,v) is simply the reciprocal of the length of the shortest path from u to v.

To extend the Shortest Path Score to weighted and directed network, let us take the network in Fig. 1 as an example. Let l_{min} represent the length of the shortest path. In the example network, the length of the shortest path from u to v is 2.

The set of paths of length l_{min} is,

$$\rho(u,v|2) = \{\{(u,b),(b,v)\},\{(u,c),(c,v)\}\} \tag{36}$$

The set of average link weights in the path set $\rho(u,v|2)$ is given by,

$$\overline{\rho}(u,v|2) = \left\{\frac{w(u,b)+w(b,v)}{2}, \frac{w(u,c)+w(c,v)}{2}\right\} \tag{37}$$

The Shortest Path Score of (u,v) is then given by the average of the elements of the set $\overline{\rho}(u,v|2)$,

$$SP_{weighted}(u,v) = \frac{1}{2}\left(\frac{w(u,b)+w(b,v)}{2} + \frac{w(u,c)+w(c,v)}{2}\right) \tag{38}$$

Equation 38 can be generalized as follows:

$$\boxed{SP_{weighted}(u,v) = \frac{1}{|\rho(u,v|l_{min})|}\sum_{p\in\rho(u,v|l_{min})}\frac{\sum_{(x,y)\in p} w(x,y)}{l_{min}}} \tag{39}$$

4.6 Rooted PageRank

Rooted PageRank is a modification of the PageRank algorithm developed by Brin et al. [12]. The Rooted PageRank between nodes u and v is defined as the probability that we will land on the node v when we perform a random walk starting from node u under the following conditions:

1. There is a $(1-\alpha)$ probability of moving to a randomly chosen neighbor node from the current node.
2. There is an α probability of the random walk restarting from the root u again.

To account for the link weights the probability of selecting the next neighbor node is proportional to the associated link weight. This is shown in more detail in the algorithm for calculating the Rooted PageRank (Algorithm 1).

Algorithm 1 Weighted PageRank Algorithm

procedure WEIGHTED–PAGERANK(u, v)
    N = 1000 ▷ The number of walks to perform
    i = 0
    n = 0
    currentNode = u
    while i < N **do**
        if random(0,1) > α **then** ▷ Reset the random walk
            currentNode = u
        else ▷ Select a neighbor node
            probDist = []
            for all n $\in \Gamma_o(currentNode)$ **do** ▷ Probability of next node
                probDist[n] = $w(currentNode, n)/\sum_{z \in \Gamma_o(currentNode)} w(currentNone, z)$
            end for
            currentNode = selectNode($\Gamma_o(currentNode), probDist$)
            if currentNode == v **then** ▷ Check if node v is selected
                n = n+1
            end if
            i = i+1
        end if
    end while
    return n/i
end procedure

5 Incremental Method for Calculating Feature Scores

When the network feature scores are modified to take into account the link weights and direction (Chapter 4), the time complexity of calculating them is higher compared to the unweighted and undirected feature scores. In this chapter, we describe methods to calculate the weighted features by iteratively updating them as new interactions between nodes and removal of links are observed.

Notations:

1. The network is observed at time $[1,2,3,\ldots T]$.
2. The decay factor of the network is δ per time interval.
3. $I_{t,t+1}$ is the set of node pairs that interact in the time interval $[t,t+1]$.
4. S is the set of all possible node pairs.

5.1 Common Neighbor Score

Let $CN_o(u,v|t)$ and $CN_i(u,v|t)$ be the Common Neighbor Score of node pair (u,v) at time t based on out-neighbor (equation 10) and in-neighbor (equation 11).

To update Common Neighbor Scores, we need to consider three cases:

1. Decay of link weight
2. Removal of link
3. Interaction between nodes

5.1.1 Effect of decay of link weight

If there were no interactions between the common neighbors of x and y at time interval $[t, t+1]$, the weight of the link between two nodes a and b is,

$$w(a,b|t+1) = \delta \cdot w(a,b|t) \tag{40}$$

Consider the in-Common Neighbor Score between nodes x and y at time t.

$$CN_i(x,y|t) = \sum_{z \in (\Gamma_i(x) \cap \Gamma_i(y))} \frac{w(z,x|t) + w(z,y|t)}{2} \tag{41}$$

The in-Common Neighbor Score of (x,y) at time $t+1$ is

$$CN_i(x,y|t+1) = \sum_{z \in (\Gamma_i(x) \cap \Gamma_i(y))} \frac{w(z,x|t+1) + w(z,y|t+1)}{2} \tag{42}$$

$$= \delta \cdot \sum_{z \in (\Gamma_i(x) \cap \Gamma_i(y))} \frac{w(z,x|t) + w(z,y|t)}{2} \tag{43}$$

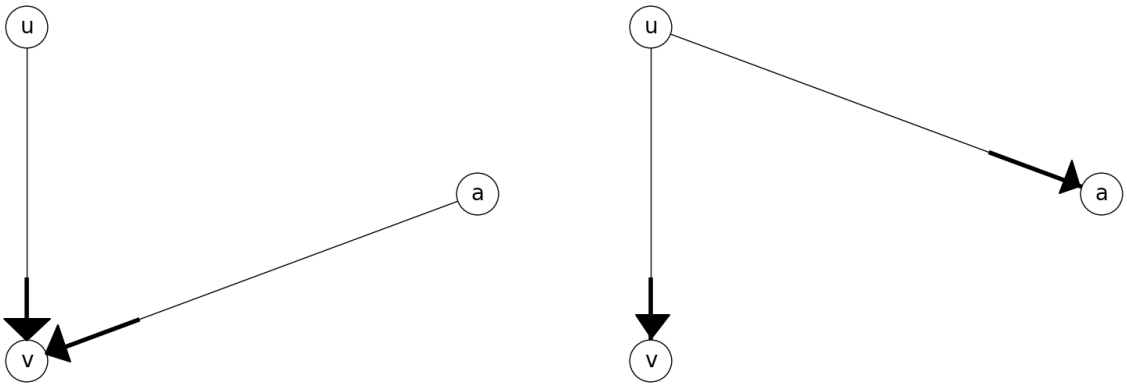
$$\boxed{CN_i(x,y|t+1) = \delta \cdot CN_i(x,y|t)} \tag{44}$$

Similarly, the out-Common Neighbor Score is,

$$\boxed{CN_o(x,y|t+1) = \delta \cdot CN_i(x,y|t)} \tag{45}$$

5.1.2 Effect of removal of link

Suppose that there was a link between nodes u and v at time t but it has been removed at time $(t+1)$. The removal of the link (u,v) will affect $CN_o(u,a)$ and $CN_i(v,a)$ for a node a that was connected to u or v as shown in Figures 2a and 2b. Consider the node a connected as shown in



(a) Case 1 (b) Case 2

Figure 2: Node pairs that interaction between (u,v) will affect

Figure 2a and recall that,

$$CN_o(u,a|t) = \frac{w(u,v|t) + w(a,v|t)}{2} + \kappa, \tag{46}$$

where κ is the component of $CN_o(u,a|t)$ that is independent of node v.

If the link (u,v) is removed at time $(t+1)$,

$$CN_o(u,a|t+1) = \delta \cdot \kappa, \tag{47}$$

which can be written as,

$$CN_o(u,a|t+1) = \delta \cdot \left(\frac{w(u,v|t) + w(a,v|t)}{2} + \kappa \right) - \delta \cdot \frac{w(u,v|t) + w(a,v|t)}{2} \tag{48}$$

$$\boxed{CN_o(u,a|t+1) = \delta \cdot CN_o(u,a|t) - \delta \cdot \frac{w(u,v|t) + w(a,v|t)}{2}} \tag{49}$$

Removal of the link (u,v) will not affect the in-Common Neighbor Score of (u,a).

Similarly, for the case where node a is connected as shown in Figures 2a, it can be shown that

the out-Common Neighbor Score of (a,v) will not be affected by the removal of the link (u,v) and,

$$\boxed{CN_i(a,v|t+1) = \delta \cdot CN_i(a,v|t) - \delta \cdot \frac{w(u,v|t) + w(u,a|t)}{2}} \tag{50}$$

5.1.3 Effect of interaction between nodes

If a pair of nodes (u,v) interact with each other in $[t,t+1]$, it will affect $CN_o(u,a)$ and $CN_i(v,a)$ for node a that is connected to u or v, as shown in Figures 2a and 2b, consider the node a, as shown in Figure 2a, at time $(t+1)$.

If there is a link from u to v at time t, then,

$$CN_o(u,a|t) = \frac{w(u,v|t) + w(a,v|t)}{2} + \kappa, \tag{51}$$

where κ is the component of $CN_o(u,a|t)$ that is independent of node v.

At time $t+1$, the weights $w(u,v)$ and $w(a,v)$ are updated, and will be,

$$w(u,v|t+1) = \delta \cdot w(u,v) + 1 \tag{52}$$

$$w(a,v|t+1) = \delta \cdot w(a,v) \tag{53}$$

Consequently, at time $(t+1)$,

$$CN_o(u,a|t+1) = \frac{w(u,v|t+1) + w(a,v|t+1)}{2} + \delta \cdot \kappa \tag{54}$$

$$= \frac{\delta \cdot w(u,v|t) + 1 + \delta \cdot w(a,v|t)}{2} + \delta \cdot \kappa \tag{55}$$

$$= \delta \cdot \left(\frac{w(u,v) + w(a,v)}{2} + \kappa \right) + \frac{1}{2} \tag{56}$$

$$= \delta \cdot CN_o(u,a|t) + 0.5 \tag{57}$$

On the other hand, if there was no link from u to v at time t,

$$CN_o(u,a|t) = \kappa \tag{58}$$

$$CN_o(u,a|t+1) = \frac{w(u,v|t+1)+w(a,v|t+1)}{2} + \delta \cdot \kappa \tag{59}$$

$$\boxed{CN_o(u,a|t+1) = \frac{w(u,v|t+1)+w(a,v|t+1)}{2} + \delta \cdot CN_o(u,a|t)} \tag{60}$$

Likewise, if there is a node a as shown in Figure 2b at time $(t+1)$, if there is a link from u to v at time t, it can be similarly shown that,

$$CN_i(v,a|t+1) = \delta \cdot CN_i(v,a|t) + 0.5 \tag{61}$$

On the other hand, if there is no link from u to v at time t, then

$$\boxed{CN_i(a,v|t+1) = \frac{w(u,v|t+1)+w(u,a|t+1)}{2} + \delta \cdot CN_i(a,v|t)} \tag{62}$$

5.2 Preferential Attachment Score

Let $PF_o(u,v|t)$ and $PF_i(u,v|t)$ be the Preferential Attachment Score of node pair (u,v) at time t. To update $PF_o(u,v|t)$ and $PF_i(u,v|t)$, we need to take into account the decay in the link weight, changes due to links that are removed and the effect of interactions between two nodes.

5.2.1 Effect of decay of link weight

Consider the out-Preferential Attachment Score between nodes u and v at time t.

$$PF_o(u,v|t) = \left(\frac{\sum_{z\in\Gamma_o(u|t)} w(u,z|t)}{|\Gamma_o(u|t)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(v|t)} w(v,z|t)}{|\Gamma_o(v|t)|}\right) \tag{63}$$

If there was no interactions between u and v and any of their neighbors in time interval $[t, t+1]$,

$$\Gamma_o(u|t+1) = \Gamma_o(x|t) \tag{64}$$

$$\Gamma_o(v|t+1) = \Gamma_o(x|t) \tag{65}$$

$$w(u,x|t+1) = \delta \cdot w(u,x|t+1) \tag{66}$$

$$w(v,x|t+1) = \delta \cdot w(v,x|t+1) \tag{67}$$

The out-Preferential Attachment Score at time $(t+1)$ is,

$$PF_o(u,v|t+1) = \left(\frac{\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)}{|\Gamma_o(u|t+1)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(v|t+1)} w(v,z|t+1)}{|\Gamma_o(v|t+1)|}\right) \tag{68}$$

$$= \left(\frac{\sum_{z\in\Gamma_o(u|t)} \delta \cdot w(u,z|t)}{|\Gamma_o(u|t)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(v|t)} \delta \cdot w(v,z|t)}{|\Gamma_o(v|t)|}\right) \tag{69}$$

$$= \delta^2 \cdot \left(\frac{\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)}{|\Gamma_o(u|t+1)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(v|t+1)} w(v,z|t+1)}{|\Gamma_o(v|t+1)|}\right) \tag{70}$$

$$\boxed{PF_o(u,v|t+1) = \delta^2 \cdot PF_o(u,v|t)} \tag{71}$$

Similarly, the in-Preferential Attachment Score at time $(t+1)$ can be show to be,

$$\boxed{PF_i(u,v|t+1) = \delta^2 \cdot PF_i(u,v|t)} \tag{72}$$

5.2.2 Effect of removal of link

Suppose that there was a link between nodes u and v at time t but it is broken at $(t+1)$.

Consider the Preferential Attachment Score between nodes u and x at time t.

$$PF_o(u,x|t) = \left(\frac{\sum_{z\in\Gamma_o(u|t)} w(u,z|t)}{|\Gamma_o(u|t)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|}\right) \tag{73}$$

The Preferential Attachment Score of (u,x) at time $(t+1)$,

$$PF_o(u,x|t+1) = \left(\frac{\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)}{|\Gamma_o(u|t+1)|}\right) \cdot \left(\frac{\sum_{z\in\Gamma_o(x|t+1)} w(x,z|t+1)}{|\Gamma_o(x|t+1)|}\right) \quad (74)$$

Since the link (u,v) is broken at time $t+1$,

$$PF_o(u,x|t+1) = \left(\frac{\sum_{z\in\Gamma_o(x|t)} \delta\cdot w(x,z|t)}{|\Gamma_o(x|t)|}\right) \cdot \left(\frac{\left(\sum_{z\in\Gamma_o(u|t)} \delta\cdot w(u,z|t)\right) - \delta\cdot w(u,v|t)}{|\Gamma_o(u|t)-1|}\right) \quad (75)$$

$$= \delta^2\cdot\frac{|\Gamma_o(u|t)|}{|\Gamma_o(u|t)|-1}\cdot\left(\frac{\sum_{z\in\Gamma_o(u|t)} w(u,z|t)}{|\Gamma_o(u|t)|}\right)\cdot\left(\frac{\sum_{z\in\Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|}\right) - \delta\cdot\left(\frac{w(u,v|t)}{|\Gamma_o(u|t)|-1}\right)\cdot\left(\frac{\sum_{z\in\Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|}\right) \quad (76)$$

$$\boxed{PF_o(u,x|t+1) = \delta^2\cdot\frac{|\Gamma_o(u|t)|}{|\Gamma_o(u|t)|-1}\cdot PF_o(u,x|t) - \delta\cdot\left(\frac{w(u,v|t)}{|\Gamma_o(u|t)|-1}\right)\cdot\left(\frac{\sum_{z\in\Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|}\right)} \quad (77)$$

Similarly,

$$\boxed{PF_i(x,v|t+1) = \delta^2\cdot\frac{|\Gamma_i(v|t)|}{|\Gamma_i(v|t)|-1}\cdot PF_i(x,v|t) - \delta\cdot\left(\frac{w(u,v|t)}{|\Gamma_i(v|t)|-1}\right)\cdot\left(\frac{\sum_{z\in\Gamma_i(x|t)} w(z,x|t)}{|\Gamma_i(x|t)|}\right)} \quad (78)$$

5.2.3 Effect of interaction between nodes

Consider the out-Preferential Attachment Score between nodes u and x at time t.

$$PF_o(u,x|t) = \left(\frac{\sum_{z\in\Gamma_o(u|t)} w(u,z|t)}{|\Gamma_o(u|t)|}\right)\cdot\left(\frac{\sum_{z\in\Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|}\right) \quad (79)$$

Let,

$$P = \left(\frac{\sum_{z \in \Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)|} \right) \tag{80}$$

Assume there is an interaction between nodes u and v at time interval $[t, t+1]$ and the link (u,v) existed at time t,

$$PF_o(u,x|t+1) = \left(\frac{\left(\sum_{z \in \Gamma_o(u|t)} \delta \cdot w(u,z|t)\right) + 1}{|\Gamma_o(u|t)|} \right) \cdot \left(\frac{\sum_{z \in \Gamma_o(v|t)} \delta \cdot w(x,z|t)}{|\Gamma_o(x|t)|} \right) \tag{81}$$

$$= \delta^2 \cdot \left(\frac{\sum_{z \in \Gamma_o(u|t)} w(u,z|t)}{|\Gamma_o(u|t)|} \right) \cdot P + \frac{\delta \cdot P}{|\Gamma_o(u|t)|} \tag{82}$$

$$\boxed{PF_o(u,x|t+1) = \delta^2 \cdot PF_o(u,x|t) + \delta \cdot \left(\frac{\sum_{z \in \Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(u|t)| \cdot |\Gamma_o(x|t)|} \right)} \tag{83}$$

If the link (u,v) did not exist at time t,

$$PF_o(u,x|t+1) = \left(\frac{\left(\sum_{z \in \Gamma_o(u|t)} \delta \cdot w(u,z|t)\right) + 1}{|\Gamma_o(u|t)| + 1} \right) \cdot \left(\frac{\sum_{z \in \Gamma_o(v|t)} \delta \cdot w(x,z|t)}{|\Gamma_o(x|t)|} \right) \tag{84}$$

$$= \left(\frac{\left(\sum_{z \in \Gamma_o(u|t)} \delta \cdot w(u,z|t)\right) + 1}{|\Gamma_o(u|t)|} \right) \cdot \delta \cdot P \cdot \left(\frac{|\Gamma_o(u|t)|}{|\Gamma_o(u|t)| + 1} \right) \tag{85}$$

$$= \left(\frac{\delta^2 \cdot P \cdot \left(\sum_{z \in \Gamma_o(u|t)} w(u,z|t)\right) + \delta \cdot P}{|\Gamma_o(u|t)|} \right) \cdot \left(\frac{|\Gamma_o(u|t)|}{|\Gamma_o(u|t)| + 1} \right) \tag{86}$$

$$\boxed{\begin{aligned} PF_o(u,x|t+1) = \delta \cdot \left(\frac{\sum_{z \in \Gamma_o(x|t)} w(x,z|t)}{|\Gamma_o(x|t)| \cdot (|\Gamma_o(u|t)| + 1)} \right) \\ + \delta^2 \cdot \left(\frac{|\Gamma_o(u|t)|}{|\Gamma_o(u|t)| + 1} \right) \cdot PF_o(u,x|t) \end{aligned}} \tag{87}$$

Similarly, for the node pair (v,x) if (u,v) existed at time t, the in-Preferential Attachment is,

$$\boxed{PF_i(v,x|t+1) = \delta^2 \cdot PF_i(v,x|t) + \delta \cdot \left(\frac{\sum_{z \in \Gamma_i(x|t)} w(z,x|t)}{|\Gamma_i(v|t)| \cdot |\Gamma_i(x|t)|} \right)} \tag{88}$$

If (u,v) did not exist at time t,

$$\boxed{\begin{aligned} PF_i(v,x|t+1) = \delta \cdot \left(\frac{\sum_{z \in \Gamma_i(x|t)} w(z,x|t)}{|\Gamma_i(x|t)| \cdot (|\Gamma_i(v|t)|+1)} \right) \\ + \delta^2 \cdot \left(\frac{|\Gamma_i(v|t)|}{|\Gamma_i(v|t)|+1} \right) \cdot PF_i(v,x|t) \end{aligned}} \tag{89}$$

5.3 Adamic Adar Index

Let $AA_o(u,v|t)$ and $AA_i(u,v|t)$ be the Adamic-Adar Index of node pair (u,v) at time t. To update $AA_o(u,v|t)$ and $AA_i(u,v|t)$, we need to take into account the decay in the link weight, the changes due to links that are removed and the effect of interactions between two nodes.

5.3.1 Effect of decay of link weight

Assume that for a pair of nodes u and v, there are no interactions between nodes that affect the Adamic-Adar Index of (u,v) at time $(t+1)$.

Consider the out-Adamic-Adar Index between nodes u and v at time t.

$$AA_o(u,v|t) = \frac{1}{2} \left(\sum_{z \in (\Gamma_o(u|t) \cap \Gamma_o(v|t))} \frac{w(u,z|t) + w(v,z|t)}{\log(\sum_{x \in \Gamma_o(z|t)} w(z,x|t))} \right) \tag{90}$$

Let,

$$A(u,v,z|t) = w(u,z|t) + w(v,z|t) \tag{91}$$

$$B(z|t) = \log(\sum_{x \in \Gamma_o(z|t)} w(z,x|t)) \tag{92}$$

At time $(t+1)$,

$$AA_o(u,v|t+1) = \frac{1}{2}\left(\sum_{z\in(\Gamma_o(u|t+1)\cap\Gamma_o(v|t+1))} \frac{(w(u,z|t+1)+w(v,z|t+1))}{\log(\sum_{x\in\Gamma_o(z|t+1)} w(z,x|t+1))}\right) \tag{93}$$

$$= \frac{1}{2}\left(\sum_{z\in(\Gamma_o(u|t)\cap\Gamma_o(v|t))} \frac{\delta\cdot(w(u,z|t)+w(v,z|t))}{\log(\delta)+\log(\sum_{x\in\Gamma_o(z|t)} w(z,x|t))}\right) \tag{94}$$

$$= \frac{1}{2}\left(\sum_{z\in(\Gamma_o(u|t)\cap\Gamma_o(v|t))} \frac{\delta\cdot A(u,v,z|t)}{\log(\delta)+B(z|t)}\right) \tag{95}$$

Let us assume that there is an F such that,

$$\frac{\delta\cdot A(u,v,z|t)}{\log(\delta)+B(z|t)} = F\cdot\frac{A(u,v,z|t)}{B(z|t)} \tag{96}$$

Then,

$$F = \frac{\delta}{\log(\delta)/B(z|t)+1} \tag{97}$$

For $\delta\to 1$,

$$\frac{\log(\delta)}{B(z|t)} \approx 0 \tag{98}$$

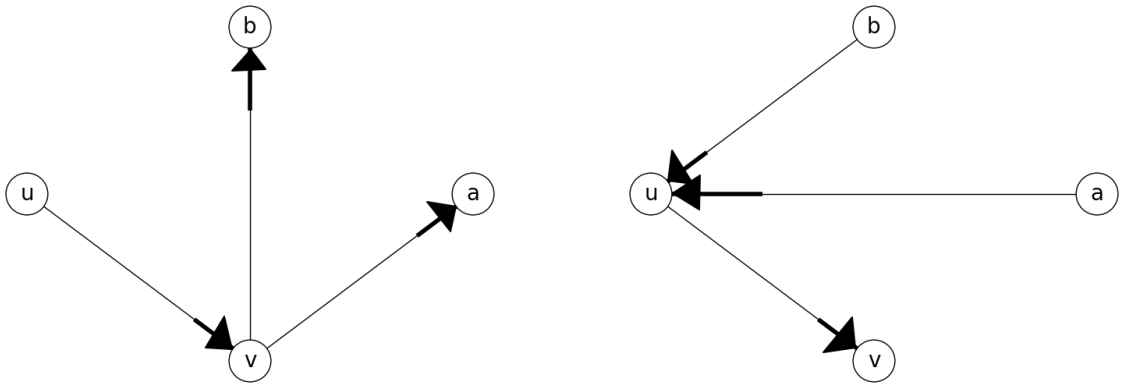
$$F \approx \delta \tag{99}$$

So,

$$\boxed{AA_o(u,v|t+1) \approx \delta\cdot AA_o(u,v|t)} \tag{100}$$

Similarly,

$$\boxed{AA_i(u,v|t+1) \approx \delta\cdot AA_i(u,v|t)} \tag{101}$$



(a) Case 1 (b) Case 2

Figure 3: Node pairs that interaction between (u,v) will affect

5.3.2 Effect of removal of links

Suppose that there was a link between u and v at time t, but it is removed at time $(t+1)$. The removal of (u,v) will affect the Adamic-Adar Index of (u,a) and (v,a) for a node a which was connected to u or v as shown in Figure 2a and 2b at time t. If there are two nodes a and b that was connected to u or v at time t as shown in Figure 3a and 3b, it will affect the Adamic-Adar Index of (a,b).

1. Let us assume that the nodes u, v and a were connected to each other as shown in Figure 2a at time t.

   At time t,

   $$AA_o(u,a|t) = \frac{w(u,v|t) + w(a,v|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t)} w(v,z|t)\right)} + \kappa \tag{102}$$

   where κ is the component of the $AA_o(u,a|t)$ that is independent of node v.

   At time $(t+1)$,

   $$AA_o(u,a|t+1) = \delta \cdot \kappa \tag{103}$$

$$AA_o(u,a|t+1) = \delta \cdot \left(\frac{w(u,v|t) + w(a,v|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t)} w(v,z|t)\right)} + \kappa \right) - \frac{\delta \cdot (w(u,v|t) + w(a,v|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t)} w(v,z|t)\right)} \tag{104}$$

$$\boxed{AA_o(u,a|t+1) = \delta \cdot AA_o(u,a|t) - \frac{\delta \cdot (w(u,v|t) + w(a,v|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t)} w(v,z|t)\right)}} \tag{105}$$

2. If the nodes u, v and a were connected to each other as shown in Figure 2b at time t, it can be similarly shown that,

$$\boxed{AA_i(v,a|t+1) = \delta \cdot AA_i(v,a|t) - \frac{\delta \cdot (w(u,v|t) + w(u,a|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(u|t)} w(z,u|t)\right)}} \tag{106}$$

3. If the nodes u, v, a and b were connected to each other as shown in Figure 3b at time t, it can be shown that,

$$\boxed{AA_o(a,b|t+1) = \delta \cdot AA_o(a,b|t) - \frac{\delta \cdot (w(a,u|t) + w(b,u|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(u|t)} w(u,z|t)\right)} + \frac{\delta \cdot (w(a,u|t) + w(b,u|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(u|t+1)} w(u,z|t+1)\right)}} \tag{107}$$

4. If the nodes u, v, a and b were connected to each other as shown in Figure 3a at time t,

$$\boxed{AA_i(a,b|t+1) = \delta \cdot AA_i(a,b|t) - \frac{\delta \cdot (w(v,a|t) + w(v,b|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(v|t)} w(z,v|t)\right)} + \frac{\delta \cdot (w(v,a|t) + w(a,b|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(v|t+1)} w(z,v|t+1)\right)}} \tag{108}$$

5.3.3 Effect of interaction between nodes

If a pair of nodes (u,v) interact in $[t,t+1]$, it will affect the Adamic-Adar Index of (u,a) and (v,a) where a is a node connected to u or v as shown in Figure 2a and 2b. If there are two nodes a and b

that are connected to u or v as shown in Figure 3a and 3b, it will affect the Adamic-Adar Index of (a,b)

1. Let us consider the nodes u,v and a connected to each other as shown in Figure 2a at time $(t+1)$. If the link (u,v) exist at time t,

$$AA_o(u,a|t) = \frac{w(u,v|t)+w(a,v|t)}{2\cdot\log\left(\sum_{z\in\Gamma_o(v|t)}w(v,z|t)\right)} + \kappa \quad (109)$$

Then, at time $(t+1)$,

$$AA_o(u,a|t+1) = \frac{\delta\cdot w(u,v|t)+\delta\cdot w(a,v|t)+1}{2\cdot\log\left(\sum_{z\in\Gamma_o(v|t)}\delta\cdot w(v,z|t)\right)} + \delta\cdot\kappa \quad (110)$$

$$\boxed{AA_o(u,a|t+1) = \delta\cdot AA_o(u,a|t) + \frac{1}{2\cdot\log\left(\sum_{z\in\Gamma_o(v|t)}\delta\cdot w(v,z|t)\right)}} \quad (111)$$

If the link (u,v) did not exist at time t, at time t,

$$AA_o(u,a|t) = \kappa \quad (112)$$

Then, at time $(t+1)$,

$$AA_o(u,a|t+1) = \frac{1+\delta\cdot w(a,v|t)}{2\cdot\log\left(\sum_{z\in\Gamma_o(v|t)}\delta\cdot w(v,z|t)\right)} + \delta\cdot\kappa \quad (113)$$

$$\boxed{AA_o(u,a|t+1) = \frac{1+\delta\cdot w(a,v|t)}{2\cdot\log\left(\sum_{z\in\Gamma_o(v|t)}\delta\cdot w(v,z|t)\right)} + \delta\cdot AA_o(u,v|t)} \quad (114)$$

2. Suppose the nodes u,v and a are connected to each other as shown in Figure 2b at time $(t+1)$. If (u,v) existed in time t, it can be similarly shown that,

$$\boxed{AA_i(a,v|t+1) = \delta\cdot AA_i(a,v|t) + \frac{1}{2\cdot\log\left(\sum_{z\in\Gamma_i(u|t)}\delta\cdot w(z,u|t)\right)}} \quad (115)$$

If (u,v) did not existed in time t,

$$\boxed{AA_i(a,v|t+1) = \delta \cdot AA_i(a,v|t) + \frac{1+\delta \cdot w(a,u|t)}{2 \cdot \log\left(\sum_{z\in\Gamma_i(u|t)} \delta \cdot w(z,u|t)\right)}} \tag{116}$$

3. Let us consider the nodes u,v,a and b connected as shown in Figure 3b. At time t,

$$AA_o(a,b|t) = \left(\frac{w(a,u|t)+w(b,u|t)}{\log\left(\sum_{z\in\Gamma_o(u|t)} w(u,z|t)\right)}\right) + \kappa, \tag{117}$$

where κ is the component of $AA_o(a,b|t)$ that is independent of the node u.

At time $(t+1)$,

$$AA_o(a,b|t+1) = \left(\frac{w(a,u|t+1)+w(b,u|t+1)}{\log\left(\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)\right)}\right) + \delta \cdot \kappa \tag{118}$$

$$= \left(\frac{\delta \cdot w(a,u|t)+\delta \cdot w(b,u|t)}{\log\left(\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)\right)}\right) + \delta \cdot \kappa \tag{119}$$

Then,

$$\boxed{\begin{aligned} &AA_o(a,b|t+1) = \delta \cdot AA_o(a,b|t) \\ &+ \delta \cdot \left(\frac{w(a,u|t)+w(b,u|t)}{\log\left(\sum_{z\in\Gamma_o(u|t+1)} w(u,z|t+1)\right)} - \frac{w(a,u|t)+w(b,u|t)}{\log\left(\sum_{z\in\Gamma_o(u|t)} w(u,z|t)\right)}\right) \end{aligned}} \tag{120}$$

4. If the nodes u,v,a and b are connected as show in Figure 3a at time $(t+1)$.

$$\boxed{\begin{aligned} &AA_i(a,b|t+1) = \delta \cdot AA_i(a,b|t) \\ &+ \delta \cdot \left(\frac{w(v,a|t)+w(v,b|t)}{\log\left(\sum_{z\in\Gamma_i(v|t+1)} w(z,v|t+1)\right)} - \frac{w(a,u|t)+w(b,u|t)}{\log\left(\sum_{z\in\Gamma_i(u|t)} w(z,v|t)\right)}\right) \end{aligned}} \tag{121}$$

5.4 Katz Score

Let is assume that $KS(u,v|t)$ is the Katz Score of (u,v) at time t and β^l is the damping factor for the Katz Score. Work by Libel-Nowell et al. [1] have shown that $\beta < 0.005$. So for paths of length four or above, the damping factor becomes 6.25×10^{-10} or lower. So calculations of Katz Score in real world networks are approximated by considering only paths with length three or less.

5.4.1 Effect of decay of link weight

Assume that $KS(u,v|t)$ is the Katz Score at time t and there is no interactions among any nodes that affect $KS(u,v|t)$.

$$KS(u,v|t) = \sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,v|l|t)} \left(\frac{\sum_{(x,y)\in p} w(x,y|t)}{l} \right) \tag{122}$$

Then, at time $(t+1)$,

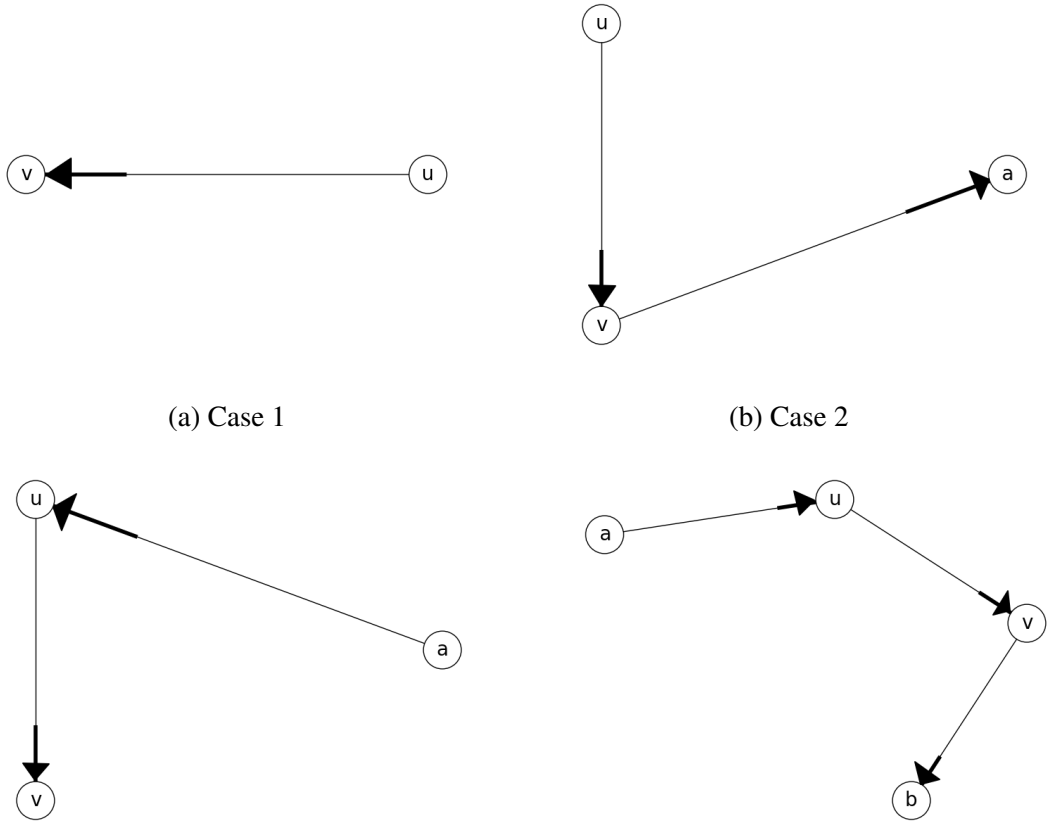
$$KS(u,v|t+1) = \sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,v|l|t)} \left(\frac{\sum_{(x,y)\in p} w(x,y|t+1)}{l} \right) \tag{123}$$

$$= \delta \cdot \sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,v|l|t)} \left(\frac{\sum_{(x,y)\in p} w(x,y|t)}{l} \right) \tag{124}$$

$$\boxed{KS(u,v|t+1) = \delta \cdot KS(u,v|t)} \tag{125}$$

5.4.2 Effect of removal of link

Let us assume that there was a link between nodes u and v at time t, but it is removed at time $(t+1)$. This will affect the Katz score between (u,v), (u,a),(a,v) and (a,b) if they were connected as shown in Figure 4 at time t.



(a) Case 1

(b) Case 2

(c) Case 3

(d) Case 4

Figure 4: Node pairs for which interaction between (u, v) will affect the Katz Score

1. Case 1. (Figure: 4a)

   If the Katz Score of (u,v) at time t is,

$$KS(u,v|t) = \beta \cdot w(u,v) + \kappa, \tag{126}$$

   where κ is the component of $KS(u,v|t)$ that is independent of the path (u,v).

   At time t, if the link (u,v) has been removed,

$$KS(u,v|t+1) = \delta \cdot \kappa \tag{127}$$

$$= \delta \cdot (\beta \cdot w(u,v|t) + \kappa) - \delta \cdot \beta \cdot w(u,v|t) \tag{128}$$

$$\boxed{KS(u,v|t+1) = \delta \cdot KS(u,v|t) - \delta \cdot \beta \cdot w(u,v|t)} \tag{129}$$

2. Similarly in Case 2. (Figure: 4b),

$$\boxed{KS(u,a|t+1) = \delta \cdot KS(u,a|t) - \delta \cdot \beta^2 \cdot \frac{w(u,v|t) + w(v,a|t)}{2}} \tag{130}$$

3. Similarly in Case 3. (Figure: 4c),

$$\boxed{KS(a,v|t+1) = \delta \cdot KS(a,v|t) - \delta \cdot \beta^2 \cdot \frac{w(u,v|t) + w(a,u|t)}{2}} \tag{131}$$

4. Similarly in Case 4. (Figure: 4d),

$$\boxed{KS(a,b|t+1) = \delta \cdot KS(a,b|t) - \delta \cdot \beta^3 \cdot \frac{w(a,u|t) + w(u,v|t) + w(v,b|t)}{3}} \tag{132}$$

5.4.3 Effect of interaction between nodes

Let us assume that interaction between (u,v) affects the Katz score between two nodes. The interaction will affect the Katz score between (u,v), (u,a),(a,v) and (a,b) if they are connected as shown in Figure 4.

1. If the nodes u and v are connected as shown in Figure 4a,

$$w(u,v|t+1) = \delta \cdot w(u,v|t) + 1 \tag{133}$$

$$KS(u,v|t+1) = \sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,v|l|t+1)} \left(\frac{\sum_{(x,y)\in p} w(x,y|t+1)}{l} \right) \tag{134}$$

$$= \left(\sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,v|l)} \left(\frac{\sum_{(x,y)\in p} \delta \cdot w(x,y|t)}{l} \right) \right) + \beta \tag{135}$$

$$\boxed{KS(u,v|t+1) = \delta \cdot KS(u,v|t) + \beta} \tag{136}$$

2. If the nodes are connected as shown in Figure 4b, and there is a link from u to v at time t,

$$\rho(u,v|l|t+1) = \rho(u,v|l|t) \tag{137}$$

$$w(u,v|t+1) = \delta \cdot w(u,v|t) + 1 \tag{138}$$

$$KS(u,a|t+1) = \delta \cdot \left(\sum_{l=1}^{3} \beta^l \cdot \sum_{p \in \rho(u,a|l|t)} \left(\frac{\sum_{(x,y)\in p} w(x,y|t+1)}{l} \right) \right) + \frac{\beta^2}{2} \tag{139}$$

$$\boxed{KS(u,a|t+1) = \delta \cdot KS(u,a|t) + \frac{\beta^2}{2}} \tag{140}$$

If there was no link from u to v at time t,

$$\rho(u,a|l|t+1) = \rho(u,a|l|t) + \{(u,v),(v,a)\} \tag{141}$$

$$\boxed{KS(u,a|t+1) = \delta \cdot KS(u,a|t) + \beta^2 \cdot \frac{1+\delta \cdot w(v,a)}{2}} \tag{142}$$

3. Similarly, if the nodes are connected as shown in Figure 4c and there is a link from u to v at time t,

$$\boxed{KS(a,v|t+1) = \delta \cdot KS(a,v|t) + \frac{\beta^2}{2}} \tag{143}$$

If there was no link from u to v at time t,

$$\boxed{KS(a,v|t+1) = \delta \cdot KS(a,v|t) + \beta^2 \cdot \frac{1+\delta \cdot w(a,u|t)}{2}} \tag{144}$$

4. Similarly, if the nodes are connected as shown in Figure 4d, and there is a link from u to v at time t,

$$\boxed{KS(a,b|t+1) = \delta \cdot KS(a,b|t) + \frac{\beta^3}{3}} \tag{145}$$

If there was no link from u to v at time t,

$$\boxed{KS(a,b|t+1) = \beta^2 \cdot \frac{1+\delta \cdot w(a,u|t) + \delta \cdot w(v,b|t)}{3} + \delta \cdot KS(a,b|t)} \tag{146}$$

5.5 Shortest Path Score

Let $SP(u,v|t)$ be the shortest path score between two nodes at time t, $l_{min}(u,v|t)$ be the length of the shortest path from node u to v at time t and $\rho_{min}(u,v|t)$ be the set of paths of length $l_{min}(u,v|t)$ from u to v at t.

5.5.1 Effect of decay of link weight

The Shortest Path Score of (u,v) at time t is,

$$SP(u,v|t) = \frac{1}{|\rho_{min}(u,v|t)|} \sum_{p \in \rho_{min}(u,v|t)} \frac{\sum_{(x,y)\in p(x,y|t)} w(x,y|t)}{l_{min}(u,v|t)} \tag{147}$$

If the interactions between the nodes in time interval $[t,t+1]$ do not affect $l_{min}(u,v|t)$ or $\rho_{min}(u,v|t)$, the Shortest Path Score at time $(t+1)$ is,

$$SP(u,v|t+1) = \frac{1}{|\rho_{min}(u,v|t+1)|} \cdot \sum_{p \in \rho_{min}(u,v|t+1)} \frac{\sum_{(x,y)\in p(x,y|t+1)} w(x,y|t+1)}{l_{min}(u,v|t+1)} \tag{148}$$

$$= \frac{1}{|\rho_{min}(u,v|t)|} \sum_{p \in \rho_{min}(u,v|t)} \frac{\sum_{(x,y)\in p(x,y|t+1)} \delta \cdot w(x,y|t)}{l_{min}(u,v|t)} \tag{149}$$

$$\boxed{SP(u,v|t+1) = \delta \cdot SP(u,v|t)} \tag{150}$$

5.5.2 Effect of interaction between nodes

If there is an interaction between nodes (u,v) at time interval $[t,t+1]$,the interaction will change the Shortest Path Score of (x,y) under two cases.

1. The link (u,v) did not exist in E_t, and the new interaction creates a new shortest path.
2. The link (u,v) exist in E_t, nodes u and v lies in the shortest path from x to y.

Suppose the link (u,v) did not exist in E_t, if the interaction between (u,v) forms a new shortest path from x to y:

1. The new shortest path should contain (u,v).

2. The path from x to u in the new shortest path should be the shortest path from x to u.

3. The path from v to y in the new shortest path should be the shortest path from v to y.

So, the interaction between u and v will create a new shortest path from x to y if,

$$l_{min}(x,y|t) \geq l_{min}(x,u|t) + l_{min}(v,y|t) + 1 \tag{151}$$

If a new shortest path is created from x to y,

$$l_{min}(x,y|t+1) = l_{min}(x,u|t) + l_{min}(v,y|t) + 1 \tag{152}$$

$$\rho_{min}(x,y|t+1) = \{\rho_{min}(x,v|t) + (u,v) + \rho_{min}(v,y|t)\} \tag{153}$$

However, if the link (u,v) exist in E_t and if (u,v) is in the shortest path from x to y,

$$w(u,v|t+1) = \delta \cdot w(u,v|t) + 1 \tag{154}$$

$$\boxed{SP(x,y|t+1) = \delta \cdot SP(x,y|t) + \frac{1}{|\rho_{min}(x,y|t)| \cdot l_{min}(x,y|t)}} \tag{155}$$

If (u,v) is in not the shortest path from x to y,

$$\boxed{SP(x,y|t+1) = \delta \cdot SP(x,y|t)} \tag{156}$$

5.5.3 Effect of removal of link

Let us assume that there was a link between nodes u and v at time t, but not at time t. Let us assume that (u,v) was in a shortest path p from node x to y.

Then removal of (u,v) will break the path p, and its contribution towards the Shortest Path

Score from x to y should be removed.

At time t,

$$SP(x,y|t) = \frac{\left(\sum_{(a,b)\in p} w(a,b|t)\right) + \kappa}{|\rho(x,y|t)| \cdot l_{min}(x,y|t)} \tag{157}$$

where κ is the contribution from the other paths that do not contain (u,v).

At time $(t+1)$, if (u,v) is broken, the path p does not exist anymore. So,

$$|\rho(x,y|t+1)| = |\rho(x,y|t)| - 1 \tag{158}$$

$$SP(x,y|t) = \frac{\kappa}{(|\rho(x,y|t)|-1) \cdot l_{min}(x,y|t)} \tag{159}$$

$$= \delta \cdot \frac{|\rho(x,y|t)|}{|\rho(x,y|t)|-1} \cdot \left(\frac{\left(\sum_{(a,b)\in p} w(a,b|t)\right) + \kappa}{|\rho(x,y|t)| \cdot l_{min}(x,y|t)} \right) \tag{160}$$

$$- \delta \cdot \frac{\sum_{(a,b)\in p} w(a,b|t)}{(|\rho(x,y|t)|-1) \cdot l_{min}(x,y|t)}$$

$$\boxed{SP(x,y|t) = \delta \cdot \frac{|\rho(x,y|t)|}{|\rho(x,y|t)|-1} \cdot SP(x,y|t) - \delta \cdot \frac{\sum_{(a,b)\in p} w(a,b|t)}{(|\rho(x,y|t)|-1) \cdot l_{min}(x,y|t)}} \tag{161}$$

5.6 Rooted PageRank

Consider the Rooted PageRank algorithm given in 1. The teleportation probability α determines the probability that the random walk with reach a certain length.

We can approximate the Rooted PageRank by considering only paths that has a probability ot at least p of being reached by a random walk. Suppose that l_c is the maximum path length that has a probability p or more of being reached.

$$p \geq (1-\alpha)^{l_c} \tag{162}$$

$$l_c \leq \left\lfloor \frac{\log p}{\log(1-\alpha)} \right\rfloor \tag{163}$$

This means that if there is an interaction between (u,v), it will effect the Rooted PageRank of nodes x and y if they are at a maximum distance of l_c away from u or v.

Work by Liben-Nowell et al [1] have shown that the optimum value of α is 0.5. So, to approximate the Rooted PageRank by taking only paths of probability $p = 0.1$ or greater,

$$l_c \leq \left\lfloor \frac{\log(0.1)}{\log(1-0.5)} \right\rfloor \tag{164}$$

$$l_c \leq \lfloor 3.21 \rfloor \tag{165}$$

$$l_c \leq 3 \tag{166}$$

5.6.1 Effect of decay of link weight

Let I be the set of node pairs that interact in time interval $[t, t+1]$, R be the set of node pairs that had a link at time t but is broken in time $(t+1)$, C be the set of all nodes in the set of node pairs $(I \cup R)$ and V be the set of all node pairs.

In Algorithm 1, it can be seen that if the link weights are changed by the same factor, the value of the Rooted PageRank will not change. So, for any node pair (u,v) such that $u \notin C \wedge v \notin C$, the decay in link weight will not change the Rooted PageRank.

5.6.2 Effect of interactions between nodes and removal of links

Let D be the set of all node pairs that are within l_c distance from the nodes in C.

So, to update the Rooted PageRank, we need to only recalculate it for nodes in D. The algorithm for updating the Rooted PageRank is given in Algorithm 8.

6 Link Prediction as a Classification Problem

In this chapter, we discuss the problem of link prediction as a classification problem. We will also address the problem that arises naturally due to the imbalance between the negative and positive classes.

6.1 Link Prediction Method

As discusses earlier, the network observed over time are $< V, E_0 >$, $< V, E_1 >$, $\ldots < V, E_T >$. However, we consider only three snapshots; that is we consider the networks $G_1^* =< V, E_{t1} >$, $G_2^* =< V, E_{t2} >$ and $G_3^* =< V, E_{t3} >$ at time t_1, t_2 and t_3 such that $t_1 < t_2 < t_3$.[1]

Recall that a deficiency of the method described in Chapter 2.2 is that the network G_2 cannot be observed in observed in real world cases, and consequently the feature set cannot be used to predict edges. For this reason, in our method, the features for the training data are generated from G_1^* and the class labels for the training data are generated from G_2^*. In other words, the training set for a classifier consists of (F_1, C_2). The features for the testing data are generated from G_2^* to predict the links in G_3^*.

The training data and the testing data are generated as described below:

1. For all possible node pairs (u, v) the features for training are calculated from G_1^*.

$$F_{train} = \{(s_1(u, v|G_1^*), s_2(u, v|G_1^*), \ldots s_n(u, v|G_1^*)) : u, v \in V\} \tag{167}$$

   where, $s_f(u, v|G_t^*)$ is a score associated with node pair (u, v) in network G_t^* for a feature f.

2. The class label for the training data is calculated from G_2^*. A pair of nodes (u, v) is considered to be in the positive class if there is a link from u to v in G_2^*; otherwise it is considered to be

[1] It should be noted here that t_1 should not be very small. If t_1 is very small, the missing past effect [13] will come into play and the learning data will be inaccurate.

in the negative class.

$$C_{train} = \{class(u, v|G_2^*) : u, v \in V\} \tag{168}$$

3. The feature set F_{test} is calculated for all possible node pairs (u, v) using G_2^*. This is the testing data.

$$F_{test} = \{(s_1(u, v|G_2^*), s_2(u, v|G_2^*), \ldots s_n(u, v|G_2^*)) : u, v \in V\} \tag{169}$$

There are two differences between our method and the earlier methods:

1. While the method by Al Hasan et al. [2] uses two networks, G_1, G_2, our method uses three (G_1^*, G_2^* and G_3^*). This is to overcome the problem with the training network used by Al Hasan et al., where G_2 will be unknown in a real world network, as described in Section 2.2. In our method G_3^* is the unknown and G_2^* is used to calculate the testing data.

2. In the earlier methods of link prediction (Chapter 2), the size of the testing data is always smaller than the training data because existing links are not included in the testing data. If the links are permanent, the node pairs that already have a link between them in the training network are not included in the testing data. However, in our method, since links are not permanent, the training data and the testing data have the same size.

6.2 Handling Class Imbalance

As mentioned in Section 2.3, in many real world networks, the size of the positive class is much smaller than that of the negative class. So, when we look at link prediction as a classification problem, the training data will be highly imbalanced.

Previous works done in classification of extremely imbalanced data [14, 15, 16, 17] have shown that good results can be obtained using an ensemble of SVM classifiers trained with the minority class and different samples of the majority class. We implement the proposed ensemble approach as described below:

1. Suppose the training data is D_{train}. Consider its two components D_+ and D_- associated with the positive and negative class respectively, where $|D_-| >> |D_+|$.

2. To use an ensemble of m SVM classifiers, n samples $[S_1, S_2, \ldots S_n]$ each of size $|D_+|$, are taken from D_- without replacement. In our simulations, $m = 10$.

3. For $i = 1, 2, \ldots, m$, we train the classifier C_i using the sample S_i and $D+$, and class membership represented by 1 for D_+ and 0 for S_i.

4. Let P_i consists of the predictions by classifier C_i, $i = 1, 2, \ldots, m$.

5. The predictions $P_1, P_2, \ldots P_n$ are combined using majority voting to generate the final prediction P.

$$P(u,v) = \begin{cases} 1 & \text{if } (\sum_{i=0}^{m} P_i(u,v)) > n/2 \\ 0 & otherwise \end{cases} \tag{170}$$

7 Data

In this chapter, we describe the data used for simulation and testing, and the pre-processing done to the data.

7.1 Data Source

In this thesis, the data used for simulation is the Twitter @-mention network. This network is obtained from the Twitter Streaming API [6] by observing it for 28 days. It contains 91169 nodes and 13087872 interactions between node pairs over the 28 days.

In this network, nodes represent users and there is a link from node u to v if u sends a tweet mentioning v. The links in this network are not permanent and they are directed. In the Twitter network, the size of the positive class is much smaller than that of the negative class.

The dataset consists of networks observed every hour $< V, E_1 >$, $< V, E_2 >$, $\ldots < V, E_{672} >$ which are generated using the network model described in Chapter 3. In this network the values of θ and δ are 0.2 and 0.9 per hour respectively.

Spammers and bots are removed from this network by using the Local Outlier Factor density-based outlier detection [20] using the in-degree and out-degree as features.

7.2 Data Preparation

As mentioned in Section 7.1, the Twitter data has approximately 10^5 nodes. So, there are approximately 10^{10} possible links.

Because of the size of the Twitter network, it is computationally very expensive to perform link prediction over the entire network. So five sets of sample nodes $V_1, V_2, \ldots V_5$ are taken from the set

V, each containing 500 nodes. The five samples are taken from areas of the network with different densities. [2]

Sample	$\lvert E_{t1}\rvert$	$\lvert E_{t2}\rvert$	$\lvert E_{t3}\rvert$	Density at t_3
V_1	729	810	824	0.0033
V_2	780	893	973	0.0039
V_3	1251	1314	1298	0.0056
V_4	1407	1463	1497	0.0060
V_5	1564	1768	1699	0.0068

Table 3: Sizes of the sample networks

To apply the link prediction method described in Section 6.1, $t_1 = 336$, $t_2 = 504$ and $t_3 = 672$.The observations were made every hour, so the value of t_i is in hours. Thus we are using the networks observed on day 14 and day 21 to predict the links in the network on day 28.

[2]The density of a set of nodes is defined as the ratio of the number of connections between the nodes to the number of potential connections. The density of a set of nodes V_i at time t will be given by,

$$Density(V_i|t) = \frac{|\{(u,v) : (u,v \in V_i) \wedge ((u,v) \in E_t)\}|}{|V_i| \cdot (|V_i| - 1)} \tag{171}$$

8 Link Prediction Results

In this chapter, we present the results of from the simulation done using the data described in Chapter 7. As mentioned in Section 7.1, the positive class is much smaller than the negative class. So, the ensemble method described in Section 6.2 is used.

The link prediction (Section 6.1) is performed on the five sets of nodes mentioned in Section 7.2. Let P_1, P_2, P_3, P_4 and P_5 represent the predictions for node sets V_1, V_2, V_3, V_4 and V_5.

Work by Fawcett [7] suggests that for datasets with imbalanced classes, the area under curve (AUC) of the receiver operating curve (ROC) is a more accurate measure of performance than accuracy and recall. So the predictions made using our method (P_i) is compared to the baseline prediction (Q_i) using the AUC of the ROC.

To provide a baseline to compare for comparison with our method, the Twitter network is assumed to be unweighted and undirected, and the link prediction method described by Leichtenwalter et al. [3] is applied. The baseline link prediction is performed for node sets V_1, V_2, V_3, V_4 and V_5. Let Q_1, Q_2, Q_3, Q_4 and Q_5 represent these predictions.

The comparison between the AUC of P_i and Q_i is given in Table 4. The comparison of the ROC for V_1, V_2, V_3, V_4 and V_5 are shown in Figures 5, 6, 7, 8 and 9.

Sample	AUC of Q_i	AUC of P_i
V_1	0.52	0.77
V_2	0.73	0.79
V_3	0.74	0.82
V_4	0.78	0.83
V_5	0.83	0.87

Table 4: AUC comparison between prediction using our method P_i and the baseline method Q_i

Sample	Number of correct predictions in Q_i	Number of correct predictions in P_i
V_1	190	577
V_2	195	632
V_3	844	1064
V_4	1168	1272
V_5	1393	1529

Table 5: Number of correct predictions comparison between prediction using our method P_i and the baseline method Q_i

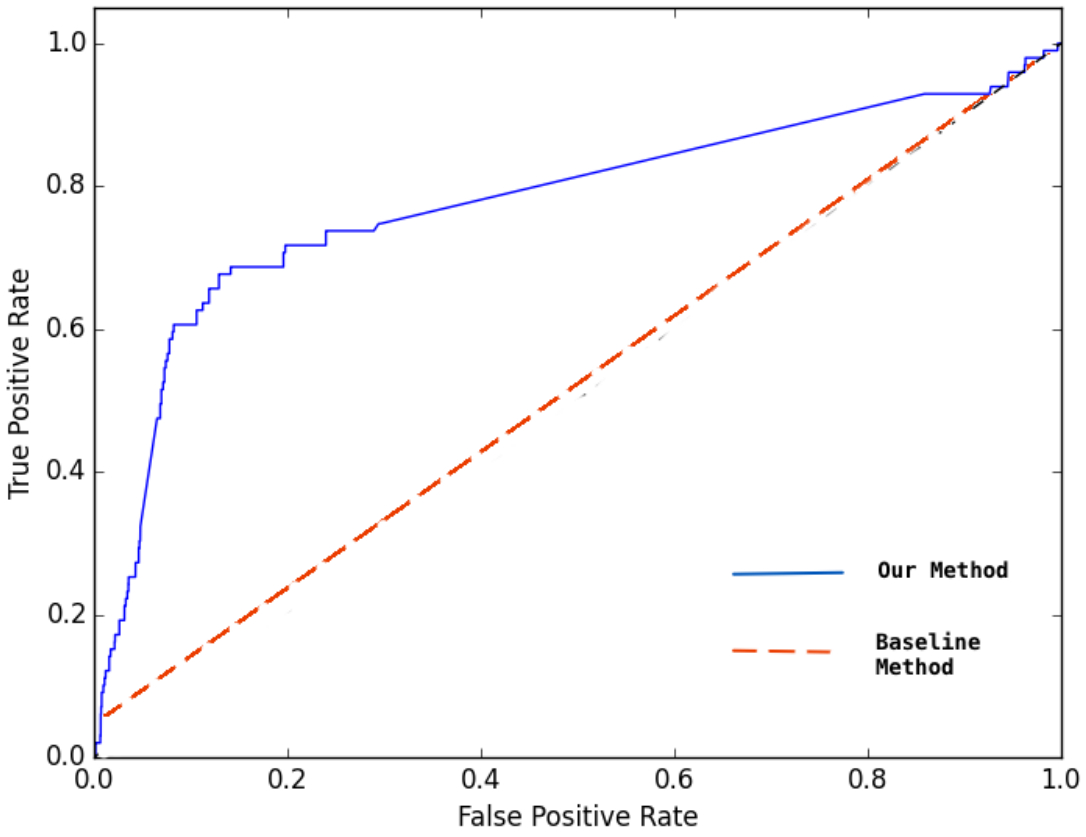


Figure 5: ROC for V_1

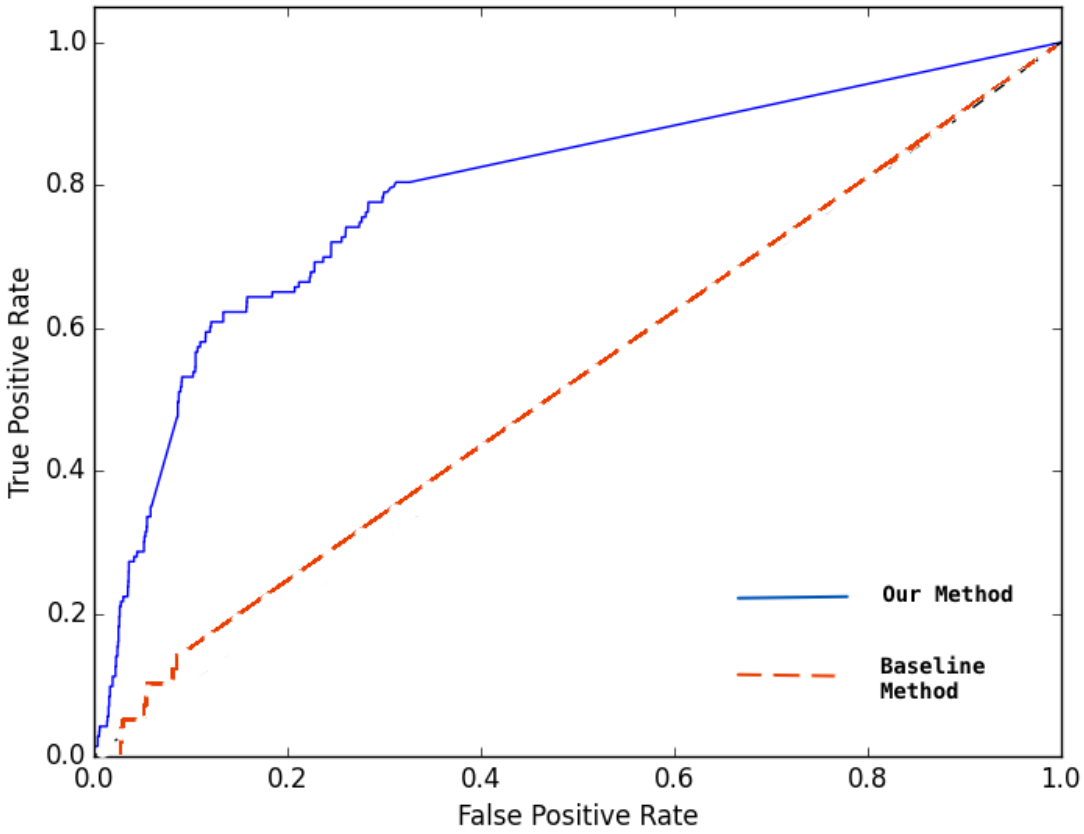


Figure 6: ROC for V_2

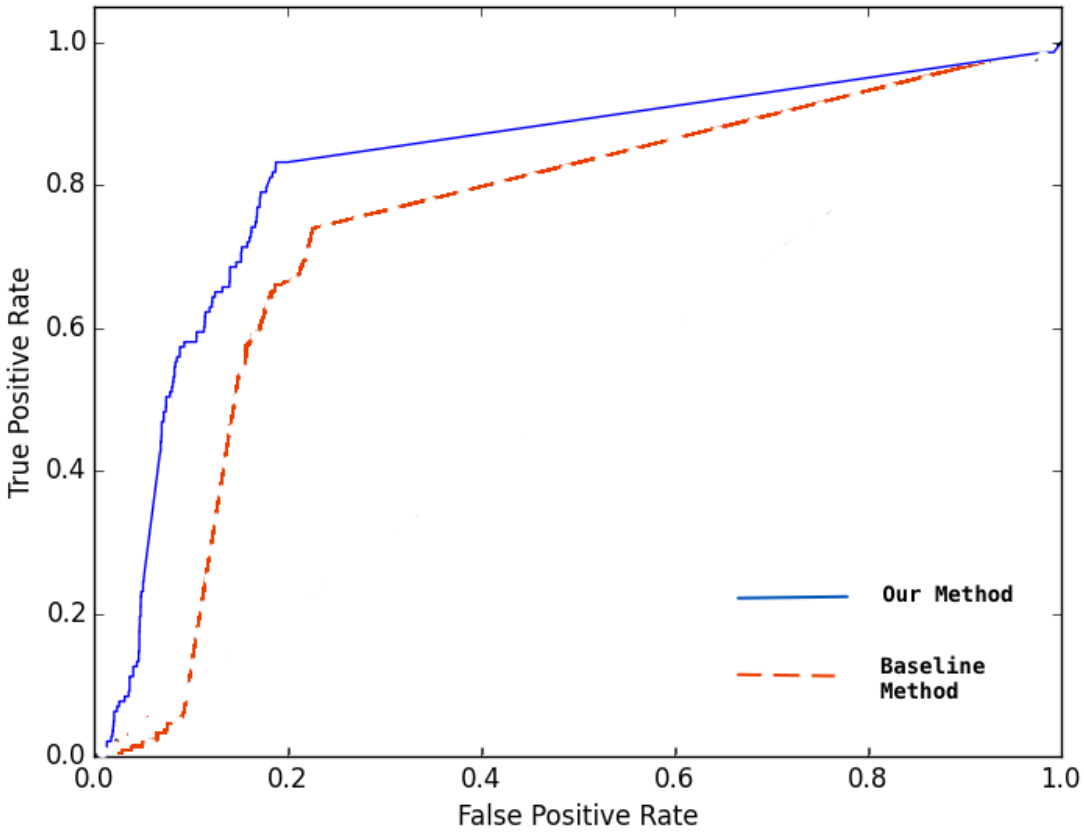


Figure 7: ROC for V_3

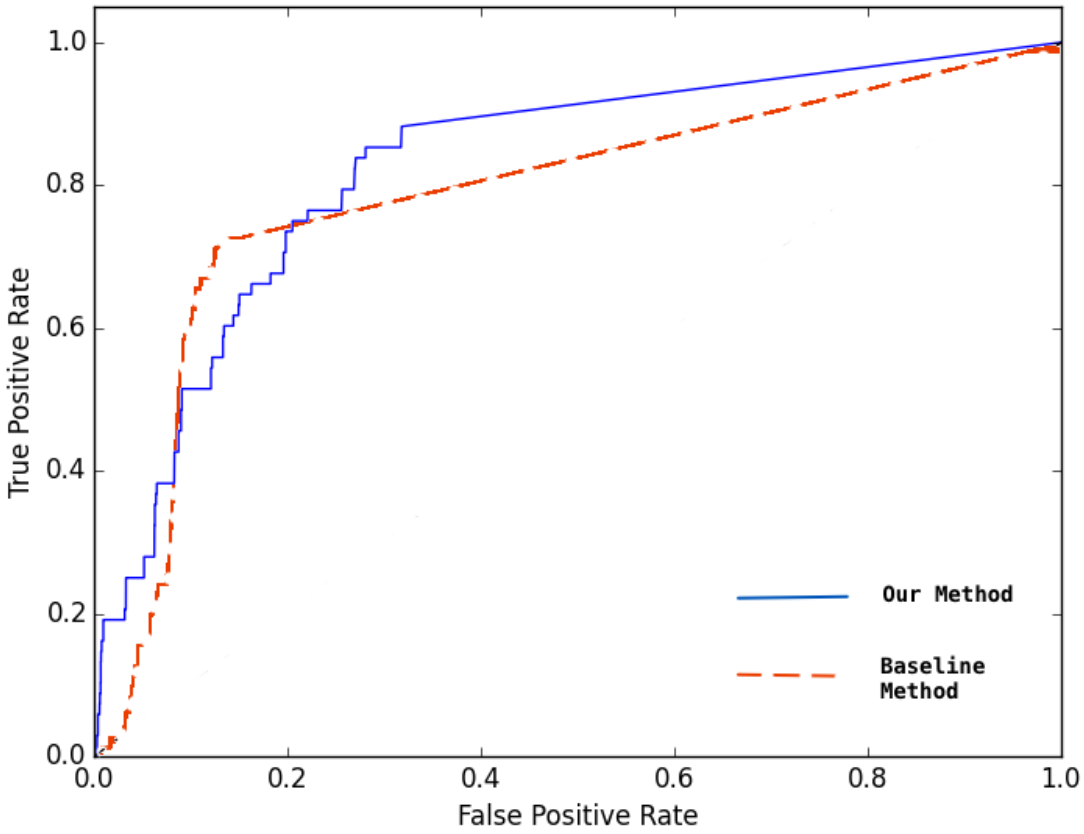


Figure 8: ROC for V_4

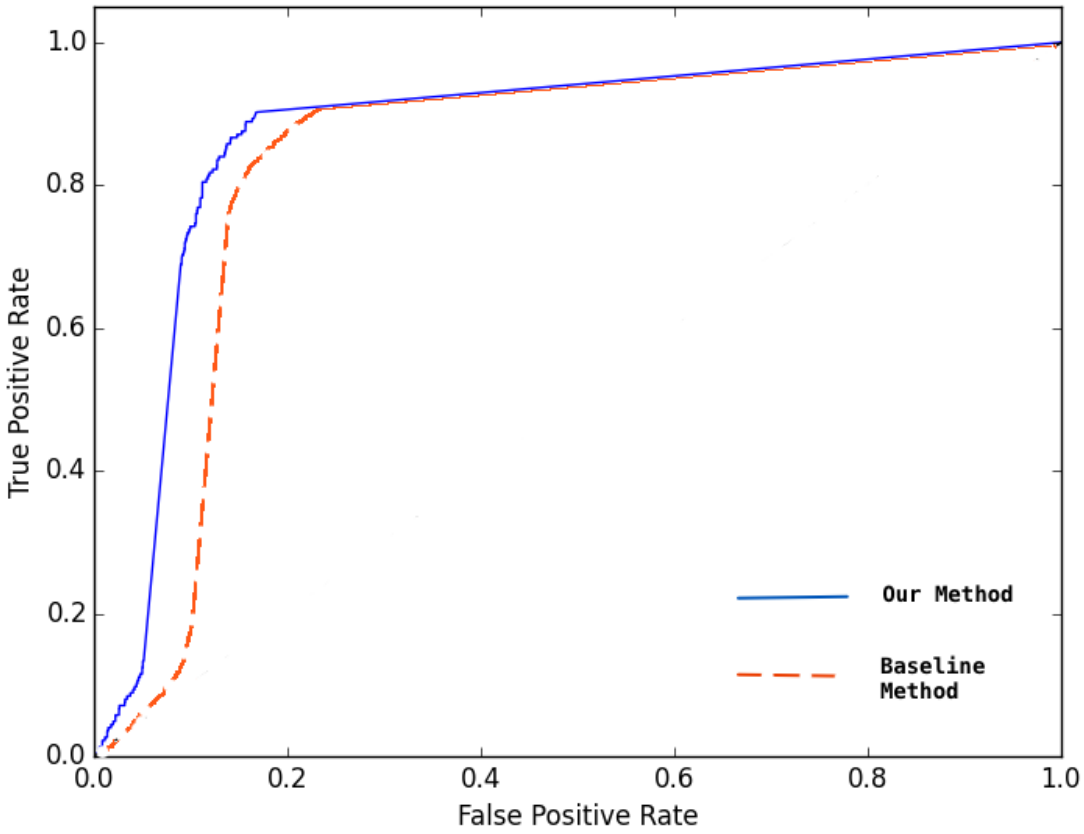


Figure 9: ROC for V_5

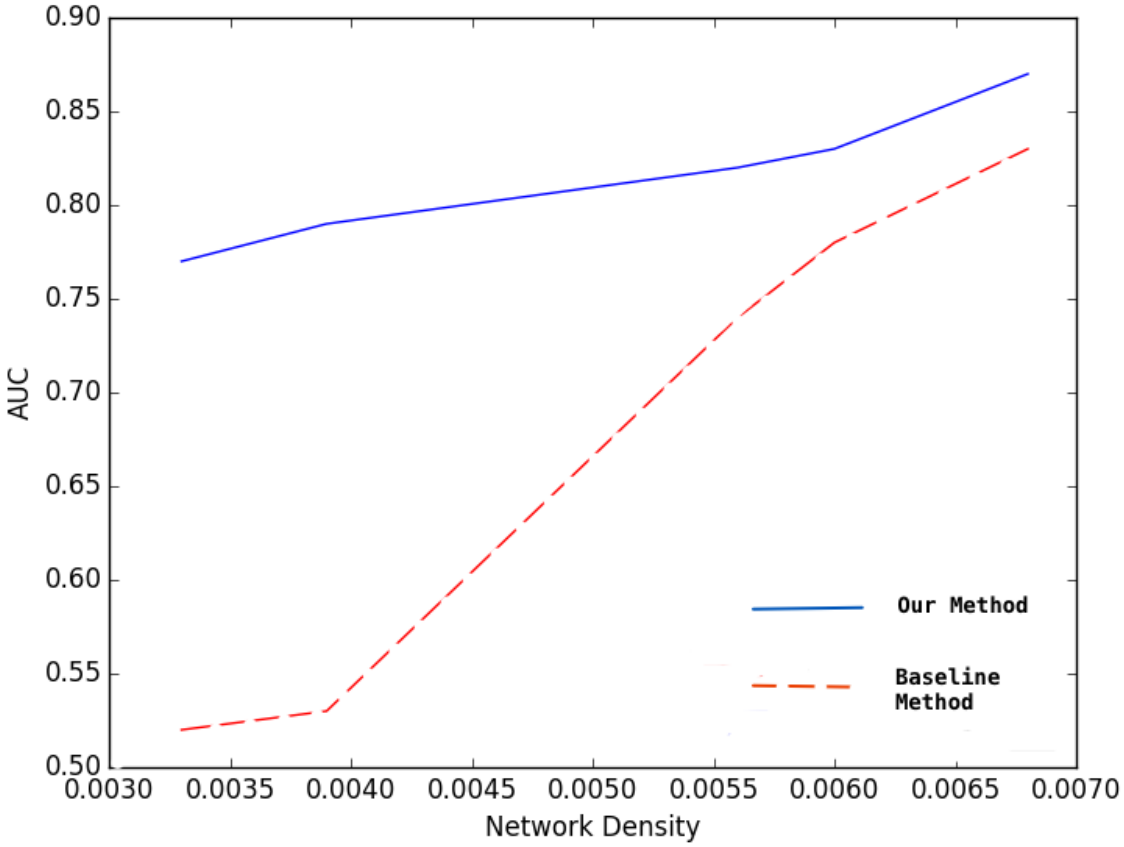


Figure 10: Variation of AUC with Density

9 Analysis

The results in Chapter 8 show that our method performs better than the baseline method in all cases. Our method takes the link weight and directions into consideration, but the baseline method assumes that the network is undirected and unweighted.

The graph in Figure 10 shows the comparison of AUC for our method and the baseline method against the density of the sample nodes. It can be seen that the performance of our method is much better than the baseline method if the density is lower, i.e., the network is sparse.

10 Conclusion

When we consider networks in which the links are not permanent, the method of link prediction described in this thesis can achieve performance better over other supervised learning methods. While the improvement over the baseline method is around 3% AUC for a dense network, the difference is very substantial (around 25% AUC) when the network is very sparse.

As mentioned in Chapter 2, most of the real world networks are sparse, and existing methods do not take into consideration links that can be removed.

The features (4) for classification used in this thesis depend on only the network topology. This means that this method can be applied easily to other type of networks.

Appendices

A Algorithms for Calculating Feature Scores

In Chapter 5, methods to incrementally update the feature scores are described. In this section, we describe the algorithms to calculate the feature sores by using the methods described in Chapter 5.

A.1 Common Neighbor Score

Let I be the set of node pairs that interact with each other in time interval $[t, t+1]$ and let R be the set of node pairs whose links are broken in time interval $[t, t+1]$.

Equations 44, 45 49, 50, 57, 60, 61 and 62 are used to develop an algorithm to iteratively update the common neighbor score for all possible node pairs. This is given in Algorithm 2.

A.2 Adamic-Adar Index

Let us assume that I is the set of node pairs that interact in time interval $[t, t+1]$, R be the set of node pairs which had a link at time t but is removed at time $(t+1)$ and let S be the set of all possible node pairs. The algorithm for updating the Adamic Adar Index is given in Algorithm 3 and 4.

A.3 Preferential Attachment Score

Let I be the set of node pairs that interact in time interval $[t, t+1]$, R be the set of node pairs which had a link at time t but do not at time $(t+1)$, N be the set of all nodes, S be the set of all possible

Algorithm 2 Common Neighbor Score Update Algorithm

procedure COMMONNEIGHBORSCOREUPDATE(I)
  for all $(u,v) \in S$ **do** ▷ Update all scores for decay
    $CN((u,v)|t+1) = \delta \cdot CN((u,v)|t)$
  end for
  for all $(u,v) \in I$ **do** ▷ Update for I
    for all $x \in (\Gamma_i(v|t+1) - \{u\})$ **do** ▷ Update CN_o
      if $(u,v) \notin E_t$ **then**
        $CN_o(u,x|t+1) = CN_o(u,x|t+1) + 0.5(1 + w(x,v|t+1))$
      else
        $CN_o(u,x|t+1) = CN_o(u,x|t+1) + 0.5$
      end if
    end for
    for all $x \in (\Gamma_o(u|t+1) - \{v\})$ **do** ▷ Update CN_i
      if $(u,v) \notin E_t$ **then**
        $CN_i(x,v|t+1) = CN_i(x,v|t+1) + 0.5(1 + w(u,x|t+1))$
      else
        $CN_i(x,v|t+1) = CN_i(x,v|t+1) + 0.5$
      end if
    end for
  end for
  for all $(u,v) \in R$ **do** ▷ Update for R
    for all $x \in (\Gamma_i(v|t) - \{u\})$ **do** ▷ Update CN_o
      $CN_o(u,x|t+1) = CN_o(u,x|t+1) - 0.5 \cdot \delta((w(u,v|t) + w(x,v|t))$
    end for
    for all $x \in (\Gamma_o(u|t) - \{v\})$ **do** ▷ Update CN_i
      $CN_i(x,v|t+1) = CN_i(x,v|t+1) - 0.5 \cdot \delta(w(u,v|t) + w(u,x|t))$
    end for
  end for
end procedure

Algorithm 3 out-Adamic Adar Index Update Algorithm

procedure OUTADAMICADARUPDATE(I)
  for all $(u,v) \in I$ **do** ▷ Update for interactions
    if $(u,v) \in E_t$ **then**
      for all $x \in \Gamma_i(v|t+1)$ **do**
        $AA_o(u,x|t+1) = \delta \cdot AA_o(u,x|t) + \frac{1}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t+1)} w(v,z|t+1)\right)}$
      end for
    else
      for all $x \in \Gamma_i(v|t+1)$ **do**
        $AA_o(u,x|t+1) = \delta \cdot AA_o(u,x|t) + \frac{1+\delta \cdot w(a,v|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t+1)} w(v,z|t+1)\right)}$
      end for
    end if
    for all $x,y \in \Gamma_i(u|t+1)$ **do**
      $AA_o(x,y|t+1) = \delta \cdot \left(\frac{w(x,u|t)+w(y,u|t)}{\log\left(\sum_{z \in \Gamma_o(u|t+1)} w(u,z|t+1)\right)} - \frac{w(x,u|t)+w(y,u|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_o(u|t)} w(u,z|t)\right)} \right) + \delta \cdot AA_o(a,b|t)$
    end for
  end for
  for all $(u,v) \in R$ **do** ▷ Update for removed links
    for all $x \in \Gamma_i(v|t+1)$ **do**
      $AA_o(u,x|t+1) = \delta \cdot AA_o(u,x|t) - \frac{\delta \cdot (w(u,v)+w(x,v|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_o(v|t)} w(v,z|t)\right)}$
    end for
    for all $x,y \in \Gamma_i(u|t+1)$ **do**
      $AA_o(a,y|t+1) = \delta \cdot AA_o(x,y|t) - \frac{\delta \cdot (w(x,u|t)+w(y,u|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(u|t)} w(u,z|t)\right)} + \frac{\delta \cdot (w(x,u|t)+w(y,u|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_o(u|t+1)} w(u,z|t+1)\right)}$
    end for
  end for
  for all $(u,v) \in S$ **do** ▷ Update all the remaining node pairs
    if $AA_o(u,v|t+1)$ do not exist **then**
      $AA_o(u,v|t+1) = \delta \cdot AA_o(u,v|t)$
    end if
  end for
end procedure

Algorithm 4 in-Adamic Adar Index Update Algorithm

procedure INADAMICADARUPDATE(I)
  for all $(u,v) \in I$ **do** ▷ Update for interactions
    if $(u,v) \in E_t$ **then**
      for all $x \in \Gamma_o(u|t+1)$ **do**
        $AA_i(v,x|t+1) = \delta \cdot AA_i(v,x|t) + \frac{1}{2 \cdot \log\left(\sum_{z \in \Gamma_i(u|t+1)} w(z,u|t+1)\right)}$
      end for
    else
      for all $x \in \Gamma_o(u|t+1)$ **do**
        $AA_i(v,x|t+1) = \delta \cdot AA_i(v,x|t) + \frac{1+\delta \cdot w(u,a|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_i(u|t+1)} w(z,u|t+1)\right)}$
      end for
    end if
    for all $a,b \in \Gamma_o(v|t+1)$ **do**
      $AA_i(a,b|t+1) = \delta \cdot AA_i(a,b|t) + \frac{\delta \cdot (w(v,a|t)+w(v,b|t))}{\log\left(\sum_{z \in \Gamma_i(v|t+1)} w(z,v|t+1)\right)} - \frac{\delta \cdot (w(v,a|t)+w(v,b|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(v|t)} w(z,v|t)\right)}$
    end for
  end for
  for all $(u,v) \in R$ **do** ▷ Update for removed links
    for all $x \in \Gamma_o(u|t+1)$ **do**
      $AA_i(v,x|t+1) = \delta \cdot AA_i(v,x|t) - \frac{\delta \cdot (w(u,v)+w(v,x|t)}{2 \cdot \log\left(\sum_{z \in \Gamma_i(u|t)} w(z,u|t)\right)}$
    end for
    for all $x,y \in \Gamma_o(v|t+1)$ **do**
      $AA_i(a,b|t+1) = \delta \cdot AA_i(a,b|t) - \frac{\delta \cdot (w(v,a|t)+w(v,b|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(v|t)} w(z,v|t)\right)} + \frac{\delta \cdot (w(v,a|t)+w(v,b|t))}{2 \cdot \log\left(\sum_{z \in \Gamma_i(v|t+1)} w(z,v|t+1)\right)}$
    end for
  end for
  for all $(u,v) \in S$ **do** ▷ Update all the remaining node pairs
    if $AA_i(u,v|t+1)$ do not exist **then**
      $AA_i(u,v|t+1) = \delta \cdot AA_i(u,v|t)$
    end if
  end for
end procedure

node pairs and $SUM_o(x|t)$ and $SUM_i(x|t)$ be the sum of the weights of the out-links and in-links of node x. The algorithm for updating the Preferential Attachment Score is given in Algorithm 5.

Algorithm 5 Preferential Attachment Score Update Algorithm

procedure PREFERENTIALATTACHMENTUPDATE(I)
  for all $x \in N$ **do**
    for all $(u,v) \in I$ **do** ▷ Update for node interactions
      if $(u,v) \in E_t$ **then**
        $PF_o(u,x|t+1) = \delta^2 \cdot PF_o(u,x|t) + \frac{\delta \cdot SUM_o(x|t)}{|\Gamma_o(u|t)| \cdot |\Gamma_o(x|t)|}$
        $PF_i(v,x|t+1) = \delta^2 \cdot PF_i(v,x|t) + \frac{\delta \cdot SUM_i(x|t)}{|\Gamma_i(v|t)| \cdot |\Gamma_i(x|t)|}$
      else
        $PF_o(u,x|t+1) = \frac{|\Gamma_o(u|t)|}{|\Gamma_u(u|t)|+1} \cdot \delta^2 \cdot PF_o(u,x|t) + \frac{\delta \cdot SUM_o(x|t)}{(|\Gamma_o(u|t)|+1) \cdot |\Gamma_o(x|t)|}$
        $PF_i(v,x|t+1) = \frac{|\Gamma_i(v|t)|}{|\Gamma_i(v|t)|+1} \cdot \delta^2 \cdot PF_i(v,x|t) + \frac{\delta \cdot SUM_i(x|t)}{(|\Gamma_i(v|t)|+1) \cdot |\Gamma_i(x|t)|}$
      end if
    end for
    for all $(u,v) \in R$ **do** ▷ Update for links removed
      $PF_o(u,x|t+1) = \frac{\delta^2 \cdot |\Gamma_o(u|t)| \cdot PF_o(u,x|t)}{|\Gamma_o(u|t)-1|} - \frac{\delta \cdot w(u,v|t) \cdot SUM_o(x|t)}{(|\Gamma_o(u|t)|-1) \cdot |\Gamma_o(x|t)|}$
      $PF_i(v,x|t+1) = \frac{\delta^2 \cdot |\Gamma_i(v|t)| \cdot PF_i(v,x|t)}{|\Gamma_i(v|t)-1|} - \frac{\delta \cdot w(u,v|t) \cdot SUM_i(x|t)}{(|\Gamma_i(v|t)|-1) \cdot |\Gamma_i(x|t)|}$
    end for
  end for
  for all $(u,v) \in S$ **do** ▷ Update all the remaining node pairs
    if $PF_i(u,v|t+1)$ do not exist **then**
      $PF_i(u,v|t+1) = \delta^2 \cdot PF_i(u,v|t)$
    end if
    if $PF_o(u,v|t+1)$ do not exist **then**
      $PF_o(u,v|t+1) = \delta^2 \cdot PF_o(u,v|t)$
    end if
  end for
end procedure

A.4 Katz Score

Let I be the set of node pairs that interact in time interval $[t,t+1]$, R be the set of node pairs that had a link between them at time t but not at $(t+1)$ and S be the set of all possible node pairs. The algorithm for updating the Katz Score is given in Algorithm 6.

Algorithm 6 Katz Score Update Algorithm

procedure KATZUPDATE(I)
  for all $(u,v) \in I$ **do** ▷ Update for interaction
    if $(u,v) \in E_t$ **then**
      $KS(u,v|t+1) = \delta \cdot KS(u,v|t) + \beta$
      for all $x \in \Gamma_o(v)$ **do**
        $KS(u,x|t+1) = \delta \cdot KS(u,x|t) + 0.5 \cdot \beta^2$
      end for
      for all $x \in \Gamma_i(u)$ **do**
        $KS(x,v|t+1) = \delta \cdot KS(x,v|t) + 0.5 \cdot \beta^2$
      end for
      for all $y \in \Gamma_o(v)$ **do**
        for all $x \in \Gamma_i(u)$ **do**
          $KS(x,y|t+1) = \delta \cdot KS(x,y|t) + \frac{\beta^2}{3}$
        end for
      end for
    else
      $KS(u,v|t+1) = \delta \cdot KS(u,v|t) + \beta$
      for all $x \in \Gamma_o(v)$ **do**
        $KS(u,x|t+1) = \delta \cdot KS(u,x|t) + \beta^2 \cdot \frac{1+\delta \cdot w(v,x|t)}{2}$
      end for
      for all $x \in \Gamma_i(u)$ **do**
        $KS(x,v|t+1) = \delta \cdot KS(x,v|t) + \beta^2 \cdot \frac{1+\delta \cdot w(x,u|t)}{2}$
      end for
      for all $y \in \Gamma_o(v)$ **do**
        for all $x \in \Gamma_i(u)$ **do**
          $KS(x,y|t+1) = \delta \cdot KS(x,y|t) + \beta^3 \cdot \frac{1+\delta \cdot w(x,u)+\delta \cdot w(v,y)}{3}$
        end for
      end for
    end if
  end for

for all $(u,v) \in R$ **do** ▷ Update for removel of links
  $KS(u,v|t+1) = KS(u,v|t+1) - \delta \cdot \beta w(u,v|t)$
  for all $x \in \Gamma_o(v)$ **do**
    $KS(u,x|t+1) = KS(u,x|t+1) - \delta \cdot \beta^2 \cdot \frac{w(u,v|t)+w(v,x|t)}{2}$
  end for
  for all $x \in \Gamma_i(u)$ **do**
    $KS(x,v|t+1) = KS(x,v|t+1) - \delta \cdot \beta^2 \cdot \frac{w(u,v|t)+w(x,u|t)}{2}$
  end for
  for all $y \in \Gamma_o(v)$ **do**
    for all $x \in \Gamma_i(u)$ **do**
      $KS(x,y|t+1) = KS(x,y|t+1) - \delta \cdot \beta^3 \cdot \frac{w(u,v|t)+w(x,u)+w(v,y)}{3}$
    end for
  end for
end for
for all $(u,v) inS$ **do** ▷ Update remaining node pairs
  if $KS(u,v|t+1)$ do not exist **then**
    $KS(u,v|t+1) = \delta \cdot KS(u,v|t)$
  end if
end for
end procedure

A.5 Shortest Path Score

Let us assume that $P_s(a)$ is the set of all nodes b such that there is a valid shortest path from a to b. Also suppose that $P_e(a)$ is the set of all nodes b such that there is a valid shortest path from b to a. Let $Q(u,v)$ be the set of all node pairs whose shortest paths passes through (u,v).

Let I be the set of node pairs that interact with each other in time interval $[t,t+1]$, R be the set of all node pairs which had a link at time t but not at time $(t+1)$ and S be the set of all possible node pairs. The algorithm to update the Shortest Path Score is given in Algorithm 7.

A.6 Rooted PageRank

To update the Rooted PageRank we only need to recalculate the Rooted PageRank for the nodes that are within a distance of l_c from the nodes which interacted or which had links that are broken.

Algorithm 7 Shortest Path Score Update Algorithm

procedure SHORTESTPATHUPDATE(I,R)
  for all $(u,v) \in I$ **do** ▷ Update for node interactions
    for all $x \in P_e(u)$ **do**
      for all $y \in P_s(v)$ **do**
        if $(u,v) \in E_t$ **then**
          $SP(x,y|t+1) = \delta \cdot SP(x,y|t) + \frac{1}{|\rho_{min}(x,y|t)| \cdot l_{min}(u,v|t)}$
        else
          if $l_{min}(x,y|t) \geq (l_{min}(x,u|t) + l_{min}(v,y|t) + 1)$ **then**
            $l_{min}(x,y|t+1) \geq l_{min}(x,u|t) + l_{min}(v,y|t) + 1$
            $\rho_{min}(x,y|t+1) = \{\rho_{min}(x,v|t) + (u,v) + \rho_{min}(v,y|t)\}$
            Recalculate $SP(x,y|t+1)$ using $\rho_{min}(x,y|t+1)$
            Update $Q(x,y)$
          end if
        end if
      end for
    end for
  end for
  for all $(u,v) \in I$ **do** ▷ Update remaining node pairs
    if $SP(u,v|t+1)$ do not exist **then**
      $SP(u,v|t+1) = \delta \cdot SP(u,v|t)$
    end if
  end for
  for all $(u,v) \in R$ **do** ▷ Update for link removal
    for all $(x,y) \in Q(u,v)$ **do**
      $SP(x,y|t+1) = \frac{|\rho(x,y|t)|}{|\rho(x,y|t)|-1} \cdot SP(x,y|t+1)$
      for all $p \in \rho_{min}(x,y|t)$ **do**
        if $(u,v) \in p$ **then**
          $SP(x,y|t+1) = SP(x,y|t+1) - \delta \cdot \frac{\sum_{(a,b) \in p} w(a,b|t)}{(|\rho(x,y|t)|-1) \cdot l_{min}(x,y|t)}$
          $\rho(x,y|t+1) = \rho(x,y|t) - \{p\}$
        end if
      end for
      Update $Q(x,y) = Q(x,y) - \{(x,y)\}$
    end for
  end for
end procedure

The algorithm to recalculate the Rooted PageRank is given in Algorithm 8

Algorithm 8 Rooted PageRank Update Algorithm

```
1: procedure PAGERANKUPDATE(I)
2:     for all u ∈ N do
3:         for all v ∈ D do
4:             PR(u,v|t+1) = WeighedPageRank(u,v)
5:             PR(v,u|t+1) = WeighedPageRank(v,v)
6:         end for
7:     end for
8: end procedure
```

B Run Time Analysis

B.1 Common Neighbor Score

Let us assume n is the number nodes, i is the average number of node pairs that interact in time interval $[t,t+1]$, r is the number of links that were broken on $[t,t+1]$ and c is the average number of common neighbors between two nodes.

Then the time it takes to calculate the common neighbor scores for all possible node pairs using equation 12 and 13 is $(c \cdot n^2)$, and the time it takes to update the Common Neighbor Scores for all possible node pairs using Algorithm 2 is $n^2 + 2 \cdot c \cdot (i+r)$. For most real world networks, $n^2 >> (i+r)$ and $n >> c$. So, $n^2 >> 2 \cdot c \cdot (i+r)$. The running time for Algorithm 2 can then be approximated as $O(n^2)$.

B.2 Adamic-Adar Index

Let us assume n is the number nodes, i is the number of node pairs that interact in time interval $[t,t+1]$, r is the number of links that were broken in time interval $[t,t+1]$ and g is the average

number of neighbors for a node.

Then the time it takes to calculate the Adamic Adar Index for all possible node pairs using equation 25 and 26 is (gcn^2).

If we use Algorithm 3 and 4, the time it takes to update the Adamic Adar Index for all possible node pairs is $(n^2 + (i+r) \cdot (g+g^2))$. For a sparse network, g is small, and we can choose the time intervals such that the amount of changes in the network $(i+r)$ is not very high. Then, $n^2 >> (i+r)(g+g^2)$. So, the time it will take to run Algorithm 3 and 4 can be approximated as $O(n^2)$.

B.3 Preferential Attachment Score

Let us assume n is the number nodes, g is the average number of neighbors for a node, r is the number of links that were broken in in time interval $[t,t+1]$ and i is the number of node pairs that interact in time interval $[t,t+1]$.

Then the time it takes to calculate the Preferential Attachment Score for all possible node pairs using equation 17 and 18 is $(2 \cdot g \cdot n^2)$.

If we use Algorithm 5, the time it takes to update the Preferential Attachment Score for all possible node pairs is $n^2 + 2 \cdot n \cdot g \cdot (i+r)$. The interval can be chosen such $n > (i+r)$. For a sparse network, $n >> g$. So, the time it will take to run Algorithm 5 can be approximated as $O(n^2)$.

B.4 Katz Score

Let us assume n is the number nodes, g is the average number of neighbors for a node, r is the number of node pairs that had a link between them in time t but not in $(t+1)$ and i is the average number of node pairs that interact in time interval $[t,t+1]$.

Then the time it takes to calculate the Katz Score for all possible node pairs using equation 35 and paths of length three or less is $(g^3 \cdot n^2)$.

If we use Algorithm 6, the time it takes to update the Katz Score for all possible node pairs is $n^2 + (i+r) \cdot (2 \cdot g + g^2)$.

The time interval can be chosen so that $n >> (i+r)$. And for a sparse network, $n >> g$. So, for a real world network, the time $n^2 >> (i+r) \cdot (2 \cdot g + g^2)$. So, the it will take to run Algorithm 6 can be approximated as $O(n^2)$.

B.5 Shortest Path Score

Let us assume n is the number nodes, p is the average number of nodes that can be reached from each node, c is the average umber of paths of shortest length between two nodes, r be the number of node pairs that had a link at time t but not at $(t+1)$ and i is the average number of node pairs that interact in time interval $[t, t+1]$.

Then the time it takes to calculate the Shortest Path Score for all possible node pairs using equation 39 and paths of length three or less is $c \cdot n^3$.

If we use Algorithm 6, the time it takes to update the Shortest Path Score for all possible node pairs is $(n^2 + (i+r) \cdot p^2)$. The time interval can be chosen such that $(i+r)$ is small. And in a sparse network, $(n >> p)$. So, in a real world network, $(n^2 >> (i+r) \cdot p^2)$. So, the running time of the algorithm can be approximated as $O(n^2)$.

B.6 Rooted PageRank

Since the algorithm for calculating the Rooted PageRank between two nodes given in Algorithm 1 depends on random walk, it is difficult to calculate the running time for this algorithm. So, let us

assume that this running time is f.

If we are to calculate the Rooted PageRank for all node pairs at each time interval, the time it will take is $n^2 \cdot f$.

If we calculate the Rooted PageRank for only the nodes that are in the set D. To calculate the size of the set D, let us assume that each node has m neighbors on average. Then the size of the set D is,

$$|D| < 2(i+r) \cdot \left(m^{l_c} + m^{l_c - 1} + \ldots + 1 \right) \tag{172}$$

$$|D| < 2(i+r) \cdot \frac{(m-1)^{l_c+1} - 1}{m-2} \tag{173}$$

Then the approximate time it will take to update the Rooted PageRank using Algorithm 8 is, $|D| \cdot n \cdot f$.

Since $|D| \leq n$, Algorithm 8 will run in time less than $n^2 \cdot f$.

In most sparse networks, the average number of neighbors is low, and if we select $l_c \leq 3$, $|D| << n$. So in real world networks, running time of the Algorithm 8 will be much lower than $n^2 \cdot f$.

References

[1] David Liben-Nowell, Jon Kleinberg "The Link Prediction Problem in Social Networks" *Journal of the American Society for Information Science and Technology,* 2007.

[2] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Mohammed Zaki "Link Prediction using Supervised Learning" *SIAM International Conference on Data Mining,* 2011.

[3] Ryan N Lichtenwalter, Jake T Lussier, Nitesh V Chawla "New Perspectives and Methods in Link Prediction" *Proceeding of the 16th ACM,* 2010

[4] Julia Preusse, JÃl'rÃt'me Kunegis, Matthias Thimm, Sergej Sizov "DecLiNe - Models for Decay of Links in Networks", 2014

[5] Sitaram Asur, Bernardo A. Huberman, Gabor Szabo, Chunyan Wang âĂIJTrends in Social Media : Persistence and DecayâĂİ *Fifth International AAAI Conference on Weblogs and Social Media,* 2011

[6] Twitter Streaming API *https://dev.twitter.com/streaming/overview*

[7] Tom Fawcett "An Introduction to ROC Ananlysis" *Pattern Recognition Letters,* 2006

[8] M E J Newman "Clustering and Preferential Attachment in growing networks" *Physical Review Letters E, 64(025102),* 2001

[9] Lada A Adamic, Eytan Adar "Friends and neighbors on the web" *Social Networks,* 2003

[10] A L Barabasi, H Jeong, Z Neda, E Ravasz, A Schubert, T Vicsek "Evolution of the social network of scientific collaboration" *Physica,* 2002

[11] Leo Katz "A new status index derived from sociometric analysis" *Psychometrika,* 1953

[12] Sergey Brin, Larry Page "The anatomy of a large-scale hypertextual Web search engine" *Computer Networks and ISDN Systems,* 1998

[13] Jure Leskovec, Jon Kleinberg, Christos Faloutsos "Graph Evolution: Densification and Shrinking Diameters" *ACM Transactions on Knowledge Discovery from Data,* 2007

[14] Z. Lin, Z. Hao, X. Yang, X. Liu, "Several svm ensemble methods integrated with under-sampling for imbalanced data learning" *Proceedings of the 5th International Conference on Advanced Data Mining and Applications,* 2009

[15] P. Kang, S. Cho "Eus svms: ensemble of under-sampled svms for data imbalance problems" *Proceedings of the 13th international conference on Neural Information Processing,* 2006

[16] Y. Liu, A. An, X. Huang "Boosting prediction accuracy on imbalanced datasets with svm ensembles" *Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining,* 2006

[17] B. Wang, N. Japkowicz "Boosting support vector machines for imbalanced data sets" *Knowledge and Information Systems,* 2010

[18] John C. Platt "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods" *Advances in Large Margin Classifiers,* 1999

[19] David J Rogers, Taffee T Tanimoto "A Computer Program for Classifying Plants" *Science,* 1960

[20] MM Breunig, HP Kriegel, RT Ng, J Sander, "LOF: identifying density-based local outliers" *SIGMOD,* 2000

[21] KL Li, HK Huang, SF Tian, W Xu "Improving one-class SVM for anomaly detection" *Proceedings of the Second International Conference on Machine Learning and Cybernetics,* 2003

Vita

Ricky Laishram received a Bachelor of Engineering in Electronics and Communication Engineering from Birla Institute of Technology in December 2010. In 2013, he joined the Master of Science in Computer and Information Science program in Syracuse University.