Link Prediction in Social networks and Affiliation networks

Nagarajan Natarajan, vishvAs vAsuki

Contents

Abstract

Social network analysis, in particular the link prediction problem, has attracted much attention recently. Often, an affiliation network is associated with a social network, containing auxiliary information about various nodes. In this project, we investigate how information in the affiliation networks can be used in link prediction on the social network. Conversely, we also investigate ways of using information in the social network to perform link prediction in the affiliation network.

In this report, we detail efforts at preparing suitable data for our experiments. We report on some attempts at exploiting information in affiliation networks for link prediction in the social network. However, our experiments suggest that, while there is useful information for this task in the affiliation network beyond what may be determined from the social network itself, the benefit from using this information is insignificant. Similarly, we also report on our attempts to use information from the social network for link prediction in the affiliation network. For this task, we were able to effectively exploit information from the social network.

Chapter 1

Introduction

1.1 Social and affiliation networks

A social network is a directed or undirected graph of nodes that represent people (sometimes referred to as users) and the links that represent some relationship among users.

An affiliation network is an undirected bipartite graph in which the nodes can be partitioned into two disjoint sets of entities: Users and Groups, and every edge is a link between a user and a group. An link in the affiliation network is indicative of the membership of the user in the group.

1.1.1 Terminology

We now introduce some terminology useful in the presentation of various link prediction techniques.

Associated with social and affiliation networks are the adjacency matrices referred to as UserUser and UserGroup respectively. A network among users can be derived from the affiliation network by adding edges between users whenever they share an affiliation, and then removing the nodes corresponding to groups. This process is called 'folding' [3]. The folding operation on a bipartite graph G is denoted by $fold(G)$.

We refer to the network in which we intend to perform link prediction as the *primary network*. Any other network used as an auxiliary source of information in making this prediction is called a *secondary network*. For example, suppose that we are attempting to do link prediction over the social network, while using the affiliation network as an auxiliary source of information. Here, the social and affiliation networks play the role of primary and secondary networks respectively.

1.2 Link prediction problem

The problem of link prediction is fundamental in the analysis of social networks. It has attracted serious attention in the last decade. The problem is to infer new relationships among nodes in a network that are likely to develop in the future, given the current state of the network. The problem can be interpreted as inferring missing links from a given snapshot of the network. Most of the efforts at solving this problem have relied on a single network. We try to use other auxiliary networks to enhance the quality of link prediction.

1.2.1 Common link prediction techniques

Link prediction methods generally assign a proximity score to each of the possible links and produce a list of scores in the descending order. *katz* is a subtle measure of proximity that out-performs many direct measures. It is explained in a later chapter. A more direct method is to assign the number of shared neighbors as the score (*Common neighbors*).

A straight-forward way to evaluate any link prediction technique would be to compare its performace against random prediction[5]. We determine the extent to which a method exploits the secondary network by comparing against a random predictor. We also present some methods to incorporate the information from the secondary network in a *katz*-like proximity measure, and we evaluate their performance against *katz*. We use *katz* measure as the baseline for our preliminary experiments.

1.2.2 How good is a prediction?

Two commonly used measures of quality of solutions in information retrieval and classification tasks are *Precision* and *Recall*. *Precision* measures the exactness or fidelity of the solution while *Recall* measures the completeness of the solution. We define *Precision* as the ratio of the number of correct predictions to the total number of predictions made, and *Completeness* (or *Recall*) as the ratio of the number of correct predictions to the total number of possible correct predictions i.e. number of edges in the test set.

1.3 Related work

There has been prolific work on analysis of social networks in the recent past, with particular attention to link prediction. But the same cannot be said about using different sources of information for link prediction or seeking beyond the social network. Most of the existing methods for link prediction try to exploit the structure and topology of the primary network. Liben-Nowell and Kleinberg[5] evaluate various measures of proximity between nodes in a given network, by predicting the links that are likely to be formed.

We take a similar approach to link prediction, but we digress from their approach in the fact that we derive proximity from the secondary network.

Savas et al propose a model[7] that uses multiple retroactive steps and auxiliary sources to improve link prediction. Similar to this approach, we intend to predict links in the primary network using a hybrid measure of proximity: proximity in primary network combined with the proximity in the secondary network.

Recommendation systems have also been an important object of study in recent years, due to their commercial applications as in the Netflix movie recommendation system. The problem of link prediction on the affiliation network can be viewed as a group recommendation problem. There has been prior work on this problem [1] from the collaborative filtering perspective, whereas we consider the same problem from the link prediction point of view.

1.4 Overview of the report

We begin with a chapter on our dataset. It focuses on the subtle issues in preparation of the data, obtaining the small data samples, challenges faced in the process and their resolution.

The next chapter elaborates on our attempts at using information from the affiliation network for link prediction in the social network. We explain the baseline prediction methods against which we compare the performance of our methods. We describe experiments, where, using information from the affiliation network and its clusters, we show improvement in performance over the random predictor. Thereby, we show that there is useful information in the affiliation network which can potentially be used in link prediction.

Next, we have a chapter on our attempts at using information from the social network for link prediction in the affiliation network. This chapter is similar in structure to its predecessor.

The report concludes with an examination of future directions for research.

Chapter 2

Data

2.1 Data preparation

We expended much effort in acquiring and preparing data required for this project. For the purpose of this project, we need not only a social network among users, but also an associated affiliation network showing the memberships of users in various groups. In this section, we describe our data preparation efforts.

2.1.1 The Orkut online social network

Orkut is Google's popular online social network service. In Orkut, every user is associated with a list of friends and a list of 'communities'. This is precisely the kind of data we are looking for.

However, companies which operate such websites are reluctant to give away this data. We were unable to acquire data for our experiments directly from Google. Instead, by crawling the Orkut network and harvesting these associations, one can construct a social and an affiliation network. Mislove and coauthors did exactly this for their work [6]. They then made this data available to us at our request.

2.1.2 Reformatting the data

The main part of the data we received from Dr Mislove consisted of two text files. One text file contained a list of UserUser links, and the other contained a list of UserGroup links. There were over 3 million users and 8 million groups. We refer to this as the 'raw data'.

So, if a user with id 25 is connected to a user with id 45, in the first file, there would either be a row which read '45 25' or a row which read '25 45' or both. If a user with id 25 is connected to groups with id 400 and 500, there would be two rows in the second file which read '25 400' and '25 500'.

Statistic	Orkut(Big)	Orkut(Small)	Youtube(Big)	Youtube(Small)
Total number of users	2783019	10133	79206	10228
Total number of groups	8730831	75551	27734	7102
Total number of edges in SS	221723034	459884	784702	230872
Total number of edges in **A**	327036181	510789	263633	56132
Average number of groups per user	117.5	50.4	3.3	5.5
Mode of number of groups per user	1	6	1	1
Average number of users per group	37.5	6.8	9.5	7.9
Mode of number of users per group	1	2	1	2
Minimum number of users per group	1	2	1	2
Average number of friends per user	79.7	45.4	9.9	22.6
Mode of number of friends per user	18	2	1	1

Table 2.1: Statistics on the Orkut and Youtube datasets. The datasets marked 'Big' are the actual social networks obtained from [??]. The datasets marked 'Small' are extracted from the actual datasets for experiments. Note the extracted datasets(columns 2 and 4) preserve the properties of actual social networks (columns 1 and 3).

A graph is conveniently represented by an adjacency matrix. Furthermore, common link prediction and clustering algorithms are naturally specified as linear algebra manipulations. Furthermore, we intended to use Matlab for this project. So, our first step was to convert the raw data to two adjacency matrices corresponding to the social network and the affiliation network.

2.1.3 Cleaning up the data

As we ran our experiments, we found many inconsistencies in our assumptions about the raw data. Below, we list the data-preparation problems we encountered and solved before we could proceed with our experiments.

Even though all links in the Orkut social network are undirected, in many cases, only one directed link was recorded in the raw data, but in many other cases two directed links would be recorded. For example, if a user with id 25 is connected to a user with id 45, in the first file, there would either be a row which read '45 25' or a row which read '25 45' or both. So, we had to deal with this inconsistency in representation while preparing our adjacency matrices.

There were many users without affiliations, groups without members and users without links to other users in the social network. Link prediction and the use of affiliation network data in link prediction is the central theme of our project. So, we decided to remove such users and groups.

Maintaining data consistency

While cleaning up the data, or in extracting smaller test-networks to experiment on, we needed to remove users and groups from the network. However, in doing so, we needed to be careful in order to maintain the consistency of the network. For example, every time we remove a user from the network, it may happen that some group may become empty. Or, when we remove a group, it may turn out that some user ends up with no affiliations.

Thus, we carefully maintained the invariant that every user is linked to some other user within the network, that every group has a certain minimum number of members, and that every user is affiliated with some group. A group with no members or with just one member is not likely to yield much information about links in the social network. So, in general, we ensure that groups have at least two members.

2.1.4 Extracting small networks

Many of the clustering and link prediction algorithms we use in our experiments do not scale well to large networks. If n is the number of nodes in a network, they often have a time complexity of $O(n^2)$. As our work is exploratory in nature, we defer the problem of how to perform well on link prediction and affiliation network clustering while scaling well with the number of nodes to future work.

Extracting a small social and affiliation network pair from a large social and affiliation network pair can be done in many ways. But, one must be careful to ensure that the smaller networks preserve as faithfully as possible the important properties of the original large networks. For example, one must ensure that the small social network shows the characteristics of a social network. That is, one must be able to observe, for example, a power-law distribution of node degrees.

For this reason, naive methods such as picking the subgraph induced by a randomly and independently selected set of users does not work. This does not preserve or consider the important local structure in a social network, and as a result, the resultant graph is very sparse. One can instead cluster the large social network, and pick the small cluster. However, we choose an alternative approach. Starting at a random node in the large social network, we traverse the graph along its links, until we have visited a certain number of unique nodes. We then take the subgraph induced by this set of nodes as our small test network.

This approach for extracting a small network, in greater detail, is as follows. The input to our algorithm is the target number of nodes in the extracted graph. A node is said to be visited if it has been encountered by our algorithm. A node is said to be expanded if all its children are visited. At every step, maintain a set *visitedNodes* of nodes visited so far, and a set *expandedNodes* of expanded nodes. Initially, *visitedNodes* has only one randomly selected node, and *expandedNodes* is empty. At each step, an arbitrary (not necessarily random) node is selected from the set *visitedNodes* - *expandedNodes*, and it is expanded. All its children are added to *visitedNodes*. This process is continued until the size of *visitedNodes* exceeds the target number of nodes in the extracted graph.

We also experimented with various ways to limit the number of groups corresponding to the users in the small test network. For example, we considered the possibility of eliminating groups with small membership. However, ultimately, we found that the number of groups associated with a small set of users is not

Dataset name	Number of users	Number of groups
socialNet10133	10133	75551

Table 2.2: A small dataset

too large; so we did not find any need to further limit the number of groups associated with the small user set.

Another issue to consider while extracting a small test network from a large network is to ensure that certain good properties are satisfied. For example, for the purpose of our experiments, we don't want users without any group affiliations. Our approach to dealing with this has already been discussed in a previous subsection.

Given an isolated node, it is hard to predict where the next link will be formed. However, if the node is not isolated, it is possible to use the knowledge of prior links to more accurately predict where the next link will be formed. Anticipating that the use of our link prediction algorithms will be limited to such cases, we eliminated users with fewer than 2 friendship links.

The small data-set we use in our experiments is described in Table 2.2.

Some properties of this and other networks are discussed in the next subsection.

2.2 The structure in social and affiliation networks

Understanding the properties of a social network is important in motivating and designing various link prediction and clustering methods. Ensuring that properties expected of a social network hold in a small test network is also a good 'sanity check' for the process which generated this test network.

Properties of a social network have been widely studied empirically. Based on these, many theoretical models of network formation have been proposed. Leskovec's thesis [4] is a good reference in this regard. Here we comment on some interesting properties of our social and affiliation networks.

A-priori, one might anticipate that the total number group-affiliations of users will be much smaller than the total number of users. On the contrary, we find that over 8 million groups are associated with just 3 million users.

Below we list some statistics of the socialNet10133 network.

- Group-membership count of users. Mean = 50.4, Min = 0, Mode = 6.
- Friendship link count of users. Mean = 45.4, Min = 2, Mode = 2.
- Membership count of groups. Mean = 6.7, Max = 1119, Min = 2.

Chapter 3

Link prediction in the social network

3.1 The training and test sets

We use a sub-network extracted from the social network as training data. Our training data consists of 70% of edges from the social network that are randomly chosen. We use the remaining 30% edges as test data.

3.2 Baseline performance measures

3.2.1 Random predictor

Let U, E denote the sets of users and edges in the social network respectively. Let T denote the set of edges in the training set and $\bar{T}$ (i.e. $E - T$) denote the set of edges in the test set. Let p be the number of predictions made by the random predictor. The expected number of correct predictions made by the random predictor is given by $\frac{|\bar{T}|*p}{\binom{|U|}{2}-|T|}$.

3.2.2 Katz

We implemented link prediction using *katz* similarity measure on the social network. *katz* is a measure of proximity between two nodes that directly sums over the collection of paths of all lengths between the two nodes, exponentially dampened by the length. If A is the UserUser matrix and β is the *damping factor*, the matrix of *katz* scores for all pairs of nodes in the network is given by,

$$K = \sum_{i=0}^{\infty} \beta^i A^i$$

Method	*Precision*	*Completeness*
Katz	0.1873	0.1873
Common Neighbours [5]	0.1080	0.1080
Random	0.0013	0.0013

Table 3.1: Baseline evaluation.

In our experiments, we computed an approximate value of *katz* by truncating the series at the 6th power. When the number of predicted links was equal to the number of test edges, *Completeness* measure was low. However, we found, not surprisingly, that *Completeness* increases with the increase in the ratio of number of predicted links to the number of test edges.

We can observe the values of *Precision* and *Completeness* measures for *katz* and Random predictor in Table 3.1. We also see that *katz* significantly outperforms random predictor. We see that the two measures are equal in both cases, as exactly $|\bar{T}|$ number of predictions were made. [1]

3.3 Is there information in the affiliation network for link prediction?

We now report two experiments which indicate that the affiliation network contains information useful in link prediction on the social network.

3.3.1 Using the affiliation network directly

Using the *katz* measure on $fold(UserGroup)$, up to the second power, performs significantly better than random prediction. Note that this is exactly the common neighbors technique. We see that the prediction is 12 times more precise than random (See Table 3.4).

3.3.2 Clusters in the affiliation network

In this experiment, we address the question as to whether there is information obtained from clusters in the affiliation network. One way of clustering the affiliation network is to derive $fold(UserGroup)$ and then cluster the resulting network.

The relevance of the affiliation network clusters with respect to the social network can be analyzed. Let $\{C_i\}, i = 1, ..., K$ denote the set of K clusters identified by some clustering algorithm on $fold(UserGroup)$. We then define,

$$edgeCount(C_i, C_j) \quad = \quad \sum_{x \in C_i} \sum_{y \in C_j} UserUser_{x,y}$$

[1]This is the case for all our experiments, and hence we show only *Precision* in the rest of the report.

Network clustered	Cluster	Fraction of intra-cluster edges in Social Network
$fold(UserGroup)$	1	.988
	2	.99
UserGroup	1	.57
	2	.83

Table 3.2: What can affiliation network clusters tell us about the social network?

Number of clusters	Overall fraction of intra-cluster edges in Social Network
2	.98
3	.82
10	.42

Table 3.3: How many clusters to identify in the affiliation network for link prediction on the social network? As the number of clusters increases, the fraction of cross-cluster edges in the social network decreases.

Note that, even though the clusters are derived from the affiliation network, the edgeCount matrix is calculated using $UserUser$, not $fold(UserGroup)$.

From the edgeCount matrix, we can infer the fraction of edges in UserUser which have one end-point in C_i, and another end-point in C_j. It is natural to suppose that edges to be predicted follow the same pattern. So, we call this fraction $edgeProbability(i, j)$.

We present the $edgeProbability(i, i)$ for the clusters identified by *graclus* [2] on $fold(UserGroup)$ in Table 3.2.

As we increased the number of clusters, the $edgeProbability(i, i)$ of the identified clusters decreased significantly. This is shown in the Table 3.3.

We now propose a way of improving over random prediction by exploiting *edgeProbability*. Having clustered the nodes in $fold(UserGroup)$, and having calculated the edgeProbability matrix the method picks an edge spanning the clusters i and j with probability $edgeProbability(i, j)$.

As in the case of random prediction, we theoretically calculated the expected number of correct predictions. We see that our method performs significantly better than random prediction. We summarize our results in Table 3.4.

3.4 Attempts at surpassing Katz

Having confirmed that there is information in affiliation networks and in affiliation network clusters, which may be useful in link prediction on social networks, we attempted to beat *katz*, our baseline for performance in link prediction. However, our simplistic attempts haven't succeeded. So, it appears that more sophisticated techniques are needed to incorporate affiliation network information

Experiment	Increase in *Precision* over Random prediction
$katz$ (Common neighbors) on $fold(UserGroup)$	12 x
Random prediction biased by $UserGroup$ clustering	2.43 x

Table 3.4: Relevance of affiliation network in link prediction.

in link prediction on the social network. We propose a few such methods later in this report.

3.4.1 Using affiliation network clusters

Boosting intra-cluster Katz scores

For a given clustering of nodes in the affiliation network, we introduced the notion of $edgeCount$ earlier. The *predictivePower* of a cluster of nodes is defined as follows.

$$predictivePower(C_i) = \frac{edgeCount(C_i, C_i)}{\sum_{j:j \neq i} edgeCount(C_i, C_j)}$$

We tried to use *predictivePower* of the clusters in order to incorporate node clustering information in calculating a $katz$-based proximity measure among nodes. We tried to boost the proximity measure of node pairs (i, j) belonging to some common cluster C_k as follows. Having calculated the $katz$ proximity measure $k(i,j)$ for all node pairs, we computed a new proximity measure $k'(i,j)$ for within-cluster node pairs: $k'(i,j) = k(i,j) * predictivePower(C_k)$.

Katz on social network weighted using clusters

We also tried to transform the social network by assigning higher weights to intra-cluster edges, relative to cross-cluster edges, and then tried computing the truncated Katz similarity measure on this weighted graph to make our prediction. But, this too did not yield better performance than $katz$.

The results

We used graclus to cluster nodes in $fold(UserGroup)$. Despite high values of *predictivePower* of the clusters as observed from Table 3.2, we found the performance of this method dropped but by a small margin compared to the performance of the baseline $katz$ (See Table 3.5).

In another experiment, we clustered the actual affiliation network, without folding it. The clustering it yielded turned out to be less favorable for link

Method	*Change in Precision*
Addition of *katz* Scores	1 x
katz scores boosted using $fold(UserGroup)$ clusters	0.99 x
katz scores boosted using $UserGroup$ clusters	0.06 x

Table 3.5: Comparison of performance against *katz*.

prediction on the social network than the clustering on $fold(UserGroup)$. This is expected as the fraction of intra-cluster edges in the social network were very low (See Table 3.2).

3.4.2 A naive attempt: Adding Katz scores

One simple way of incorporating information from the affiliation network in link prediction could be the following:

1. $K_1 \leftarrow katz(UserUser)$.
2. $K_2 \leftarrow katz(fold(UserGroup))$.
3. Predict links with the score matrix $K_1 + K_2$.

We found that this method did not yield any improvement in *Precision* compared to the predictions made with just K_1 (See Table 3.5).

3.4.3 Using Supervised Katz

The idea

Savas et al[7] have proposed that using a supervised variant of the truncated Katz measure outperforms link prediction with the truncated Katz measure alone. Furthermore, they have extended this method to incorporate information from multiple sources. This method finds natural application in our case, as we are trying to incorporate information from two networks: the social and the affiliation network. So, we tried to see if, using a similar approach, we can outperform Katz.

Information sources and their weights

Let A and B be the adjacency matrix corresponding to the primary and secondary networks respectively. Then, the pure powers of A are A, A^2, A^3... Hybrid powers of A and B are matrices like AB, BA, ABA, A^2B, B^2A... The score matrix used for link prediction is similar to the Katz score:

$$K = \sum_{i=1}^{n} w_i M_i$$

In the above equation, M_i is a pure power of A, or a pure power of B, or a hybrid power of A and B. w_i is the scalar weight assigned to M_i.

	Sources	*Change in Precision*
1	$A^2, ..., A^5$ w = (47.30, 97.17, -53.12, -52.55)	2.177x
2	$A^2, ..., A^5, B$ w = (47.06, 96.43, -53.30, -51.84, 136e-6)	2.179x
3	$A^2, ..., A^5, C$	2.18x
4	$A^2, ..., A^5, C, C^2, (AC + CA)$	2.177x
5	A^2, A^3	1.458x
6	A^2, A^3, B	1.461x

Table 3.6: Supervised katz method for link prediction in the social network. Comparison against *katz*. We represent our primary source of information, the social network, by A, and $fold(UserGroup)$ by B. We indicate the sparsified $fold(UserGroup)$ by C.

Creating a validation edge set

For the purpose of our experiments, we had already split the edges in a known network into training and test edge sets. Here, we further constitute the validation edge set by setting aside 30% of the edges in the training edge set. Let V be the adjacency matrix corresponding to the graph formed by the validation edge set. We then try to find the best weights to make a prediction.

Learning the weights

In our experiments, time and memory constraints limit the number of sources (M_i) we use in our experiments. Let M be the matrix we get by vectorizing the matrices (M_i) and arranging them as columns of a matrix. Let w be the weight vector ($w_1..w_n$). We learn weights w by solving the following least squares problem $V \approx Mw$.

This problem can be solved using the normal equations (NE). Or, one can try to impose the constraint that w be non-negative, in which case we arrive at w by using a non negative least squares (NNLS) solver. We found solving normal equations much faster in comparison to computing the non negative least squares (NNLS) solution. Furthermore, using NNLS led to decrease in prediction performance, when compared with using the normal equations solution. Hence, we used the normal equations approach to finding the weights.

Results

We present the results in table 3.6.

Significance of negative weights Consider experiment 1 in table 3.6. We observe that negative weights are learned for A^4, A^5. This leads to the question:

Does this suggest that these matrices do not contribute any new information towards the prediction? This is not the case: Upon using only A^2, A^3 as sources, we obtain a much lower, 45.8%, improvement over truncated Katz.

Exploiting information in the affiliation network Comparing experiments 1 and 2, we observe that the inclusion of $fold(UserGroup)$ as a source in making the prediction leads to a 0.6% improvement in precision. This is a comparatively minor improvement. We also observe that the weight assigned to B is three orders of magnitude smaller than weights assigned to powers of A. This indicates that, while there is some information present in the affiliation network useful in link prediction on the social network, this information is rather insignificant compared to information present in the social network itself.

Similarly, from experiment 4, we observe that higher powers of B and hybrid powers such as (AB + BA) do not significantly help in prediction either.

Using a sparsified affiliation network As noted earlier, we found a large number of groups with a membership size over 20. We hypothesize that large groups do not yield significant information about the closeness of their members. For example, just because two people are among a 500 members of a certain group, we cannot deduce that friendship between them is more likely than friendship with someone not in that group.

Furthermore, we wanted to reduce the density of $fold(UserGroup)$. The matrix $fold(UserGroup)$ originally had a density of 0.25. This limited our ability to compute the higher powers of this matrix.

So, we removed groups with size over 20, to make the $fold(UserGroup)$ sparser (density 0.01), while retaining as much information as possible. From experiment 3, we can see that the sparsification successfully achieved this aim.

In similar spirit, following the intuition that larger groups possess lower user-user similarity information, we tried normalizing rows of $UserGroup$ before calculating $fold(UserGroup)$. However, that was not sufficient to alter link prediction performance.

The overfitting problem Compare experiments 3 and 4. Even though we use more number of sources in the latter experiment, we find that the performance decreases upon adding these sources. One might ask the question: Why did we not end up with the same weights as in experiment 3, with 0 weights assigned to the extra sources? That would have resulted in better precision. The reason that this did not happen, however, is that the weights are overtuned to perform optimally while predicting against the validation edge set. We encounter a similar problem while investigating the use of supervised Katz in affiliation network link prediction. However, in this project, we did not pursue a solution to this problem.

3.4.4 SVD Latent factors approach

Consider the following merged network.

$$MergedNet = [\lambda * UserGroup \; UserUser]$$

We want to know what new links will be formed in this network, particularly in the $UserUser$ portion of the network. One can view this as a matrix completion problem.

A simple way to do this is to try to find a low rank matrix J which is close to $MergedNet$. One can look upon this low rank matrix J as a low dimensional representation of $MergedNet$, and one can hypothesize that a future version of $MergedNet$ will have a very similar low dimensional representation. So, we can consider J to be the matrix which assigns a score to each possible vertex pair: the higher the score, the more likely the edge between those vertices.

Here, we use the rank k SVD approximation $MergedNet \approx J = U_k S_k V_k^T$. Here, all three matrices have rank at most k.

We see that the method is parameterized by λ and k. It is easy to see that setting $\lambda = 0$ gives a low rank approximation which considers $UserUser$ alone.

We considered various combinations of $f \in [10, 200]$ and $\lambda \in [0, 1]$, but we always found that $\lambda = 0$ led to the optimal prediction. In other words, this approach to link prediction on the social network worked best when it entirely ignored information from the affiliation network. However we will see that a similar approach proves useful in the case of link prediction in the affiliation network.

3.5 Summary

In this chapter, we reported on our attempts at using information from the affiliation network in link prediction in the social network. Initially, by comparison against the random predictor, we confirmed that there is some information in the affiliation network and in affiliation network clusters, which can be used in social network link prediction. We then reported various unsuccessful attempts at exploiting information in affiliation network clusters. We then described our application of the supervised Katz method for incorporating information from both the networks in link prediction.

Our account of the application of the supervised Katz approach to this problem is interesting in its own right, because it describes the subtle factors that one encounters in the process, such as the sparsification of $fold(UserGroup)$ and the overfitting problem. Even though the supervised Katz approach was unable to significantly exploit information in the affiliation network, its application on the powers of $UserUser$ alone emphatically demonstrated its effectiveness in link prediction. Finally, we described the SVD latent factor approach to prediction, which, while again surpassing $katz$ in its performance, was again unable to effectively exploit the information present in the affiliation network for link prediction in the social network.

Chapter 4

Link prediction in the affiliation network

4.1 Introduction

Link prediction in the affiliation network is closely related to the problem of link prediction in the social network. It is quite intuitive to consider the problem of predicting the new users who will join a given community (group). Equivalently, one could ask the question of what new communities a given user will join.

Formally stated, the problem is to predict new links that will appear in the bipartite graph between set of users and communities which is the affiliation network. It is natural to apply to this case the techniques similar to those that we used in the social network link prediction problem.

However, the adjacency matrix of an affiliation network is rectangular. It is therefore essential to modify the techniques used for link prediction in social networks suitably. We emphasize here that we try to do link prediction in the affiliation network while incorporating information from the social network.

Recall that the affiliation network is represented by the $UserGroup$ adjacency matrix and the indirect social network is obtained by folding the affiliation network and is represented as $fold(UserGroup)$.

Let $C = \left[\begin{smallmatrix} B & A \\ A^T & 0 \end{smallmatrix}\right]$.

4.2 Baseline

We compute a similarity measure $AffiliationSimilarity$ among users by applying truncated $katz$ on $fold(UserGroup)$ (up to the third power), as already explained in the context of link prediction in social networks. Each user-group link is then assigned a score, which is given by the matrix $AffiliationSimilarity * UserGroup$. We use these scores to make link predictions, as elaborated in the previous experiments.

Note that link prediction using the above score matrix is algebraically identical to the following prediction process.

1. Form the adjacency matrix A corresponding to the affiliation network [1].
2. Compute the truncated *katz* similarity matrix S using A.
3. Predict user-group links using the relevant portion of S.

Comparing the performance with table 3.1, we realize that the problem of link prediction in the affiliation network could be harder than link prediction in the social network. Henceforth, we will refer to this method as simply "baseline".

4.2.1 Information in B

Can the information contained in social network proximity help us with link prediction on the affiliation network?

Precision of random prediction is bounded above by .002 for orkut.

B^2A for Orkut dataset yields 3% precision, and for youtube yields 4.5%.

BA for Orkut dataset yields 4.2% precision, and for youtube yields 6.1%.

Common neighbors (AA^TA) yields $7.442067e-02$.

4.2.2 Katz and SVD baselines

On the orkut dataset, the best precision obtained using this baseline method is 7.5% (beta = 10e-6). Katz on C yields 7.5% (beta = -1, l =0.2).

On orkut dataset, the best precision obtained using SVD on A alone is 8.9% (rank 10 approximation).

On the youtube dataset, best precision obtained using this baseline method is 12% (beta = 10e-12: common neighbors, effectively). Katz on C yields 12.2% (beta = 0, l =0.2).

On youtube dataset, the best precision obtained using SVD on A alone is 11.8% (rank 20 approximation).

4.3 Merging social and affiliation networks

One simple way of using information from the social network to do link prediction in affiliation network is to merge the two networks and perform *katz* on the resulting network. This is equivalent to computing the following summation. We represent our primary source of information, the affiliation network, by A, and $UserUser$ by B.

$$\begin{aligned} UserSimilarity &= \beta B + \beta^2(B^2 + AA^T) + \beta^3(B^3 + BAA^T + AA^TB)^2 \\ Score(i,j) &= (UserSimilarity * UserUser)_{ij} \end{aligned}$$

[1]Note that this is different from $UserGroup$.

[2]AA^T is a very dense matrix (25% nonzeros in socialNet10133). Computing higher powers of this matrix takes long time, and so we stop at the third power.

	Sources	*Change in Precision*
1	$A, A^2, A^3, B, B^2, B^3, AB, BA$	0.276x
3	A^2, A^3 $w = [1.76 * 10^{-4}\ \ 3.08 * 10^{-6}]$	0.036x

Table 4.1: Supervised Katz method for link prediction in the affiliation network. Comparison is against $katz$. We represent our primary source of information, the common-groups matrix, by A, and $UserUser$ by B. Performance is hit by the overfitting problem.

$Score(i, j)$ is the score value assigned to the possible link between user i and group j and β is the $dampingfactor$ as discussed earlier. However, this method performed worse than the baseline. The precision was only 0.35 times that of the baseline. This method is rather naive, where a more sophisticated method would try and assign weight λ for the edges in the social network. However, we did not pursue this idea in our project.

4.4 Using supervised Katz

We applied the supervised approach as previously discussed in the context of link prediction in social networks. We have A and B as the adjacency matrices of the primary(affiliation) and secondary(social) networks. As before, we used a) pure powers of A and b) hybrid powers of A and B, as information sources in our experiments. We followed the same procedure as described in the earlier section for creating test, validation and training sets. As in the case of the baseline, we multiply the resulting user-user similarity matrix by $UserGroup$ to arrive at the final score matrix.

4.4.1 The overfitting problem

In the table 4.2, we report on our attempts to use supervised Katz for link prediction on the affiliation network.

Consider experiment 3. Observe that prediction using truncated $katz$ is equivalent to using weight vector $v = [\beta^2\ \ \beta^3]$. We observed that though w performs slightly better (1.006x) than v on the validation set, it performs much worse (0.036x) on the training set. This is clearly due to overfitting to the validation set. Also, note that $w(A^2)/w(A^3)$ is much higher than $v(A^2)/v(A^3)$, suggesting that the parameters are highly tuned to perform well on the validation set. [3]

We faced the problem of overfitting earlier, when we used supervised katz method in social network link prediction. But, unlike that case, we see that any benefit of using supervised katz is lost by overfitting to validation edge set.

[3]We used $\beta = 0.5$ for our experiments

Sources	Regularizer	*Precision*
$A, A^2, A^3, B, B^2, B^3, AB, BA$	Lasso with $l = 10^3$	0.026

Table 4.2: Supervised Katz method for link prediction in the affiliation network. We represent our primary source of information, the common-groups matrix, by A, and $UserUser$ by B. Performance is hit by the overfitting problem.

Youtube dataset

4.5 Linked Matrix Factorization

Tang et al have proposed an effective clustering technique called *Linked Matrix Factorization*[8](LMF) to use information from different graphs. Their method is based on approximating each graph by matrix factorization with graph-specific factors and a factor common to all graphs.

4.5.1 Factorization of social and affiliation networks

We use LMF, albeit modified, for the problem of link prediction in affiliation networks. Firstly, LMF finds a factor matrix that is common to all graphs. Since affiliation network has 2 types of nodes, we need to find one factor that captures the user-specific information, which is common to both the graphs and other factor that captures the group-specific information.

Let us write the adjacency matrix of primary network as A and that of the secondary network as B. Let N be the number of users and G be the number of groups in the networks.

Similar to the SVD latent factor approach, we try to arrive at low rank representations of A and B in terms of the common and graph-specific low-rank factor matrices. Specifically, we want $A \approx PL_AQ^T$ and $B \approx PL_BP^T$, where the matrices involved are described below.

1. $L_A, L_B \in \mathbb{R}^{d,d}$: symmetric matrices that capture the information specific to primary and secondary networks respectively.
2. $P \in \mathbb{R}^{N,d}$: Factor matrix that captures user-specific information common to both the networks.
3. $Q \in \mathbb{R}^{d,G}$: Factor matrix that captures group-specific information.

Note that, as $d << G$, $d << N$, this amounts to finding the best rank-d linked approximations for A and B. Let us now turn to the specifics of the objective function and implementation details.

4.5.2 Objective function

We use the following objective function.

$$J = \min_{P,Q,L_A,L_B} \{ \left\| A - PL_AQ^T \right\|_F^2 + m \left\| B - PL_BP^T \right\|_F^2 + \lambda(\|P\|_F^2 + \|Q\|_F^2 + \|L_A\|_F^2 + \|L_B\|_F^2)\}$$

The parameter m controls the weight associated with fitting the secondary network. The regularization parameter λ helps avoid overfitting.

It follows that,

$$\{P, Q, L_A, L_B\} = \underset{P,Q,L_A,L_B}{\text{argmin}}\ J$$

Using arguments similar to those made in the context of SVD latent factors method, we consider the low-rank representation of the affiliation network i.e. PL_AQ^T as the score matrix, and make predictions based on the same.

4.5.3 Implementation and Results

Tang[8] et al use a variant of the *L-BFGS* algorithm, which is well-suited for high-dimensional optimization problems. Their implementation uses "Alternating Minimization" technique, wherein one variable is optimized with other variables fixed. This procedure is repeated until convergence. We adopted this implementation to solve our optimization problem.

However, the convergence rate was very slow, and we chose to terminate the minimization using an arbitrary thresholding procedure. The experiment yielded a slight increase of 0.7% in the precision over baseline. We observed that this method is very slow compared to other link prediction methods we tried.

The initial values of P, Q, L_A, L_B affect the convergence of the solution, as is the case with many non-convex optimization problems. The simplest approach is to initialize them to random values. A better way is to initialize $P = U_d, L_A = S_d, Q = V_d$ where $U_dS_dV_d^T = A$, the d-rank SVD approximation of the affiliation network matrix. When the initialization was random, the convergence was poorer and the performance was worse than the baseline.

We tried to learn the best values of m, λ, and d using cross validation. However, they did not seem to affect the outcome. This technique for link prediction yielded over a 25% improvement over the baseline.

4.6 SVD latent factors approach

We extend the method of using SVD latent factors to the affiliation network link prediction. Similar to what we discussed in the context of social network link prediction problem, we form a "merged network" by tacking the UserUser adjacency matrix on the UserGroup matrix. i.e.

$$MergedNet = [UserGroup \quad \lambda * UserUser]$$

We then apply SVD approximation to $MergedNet$ and use the resulting $U_kS_kV_k^T$ as the score matrix. We see that the method is parameterized by λ

and k. It is easy to see that setting $\lambda = 0$ effectively ignores information from the social network.

4.6.1 Successful exploitation of information from the social network

As in the supervised Katz experiments, we set aside a fraction of the edges in the training set to constitute the validation edge set. We then used this validation set to learn $\lambda \in [0, 1]$ and $k \in [10, 160]$. This caused us to use the values (60, 0.8) to make the final predictions on the test set. The table [removed] [**Incomplete**]shows that these parameters generalize well - they are indeed the best parameters we could have chosen in that range.

We see that this method greatly outperforms the baseline. Furthermore, we realize that the SVD latent factors approach applied to the merged network yields a significant improvement in the performance of link prediction than when applied to the affiliation network alone. So, unlike the social network link prediction case, we are able to effectively exploit information from a secondary network.

On average (taken over 6 runs), the precision of SVD method is 10%.

For SVD done on the matrix $C = \left[\begin{smallmatrix} \lambda S & A \\ A^T & 0 \end{smallmatrix}\right]$. on Orkut test set: numFactors: 70, l: 0.600000, precisionSVD: 0.101300: average over 7 runs; std deviation is 0.004.

4.6.2 Youtube dataset

```
beta: 1.000000e-06,useBestKatz: 0,katz precision: 1.172803e-01
on validation set: numFactors: 10,l: 0.000000,precisionSVD: 0.073295
on validation set: numFactors: 10,l: 0.200000,precisionSVD: 0.071938
on validation set: numFactors: 60,l: 0.000000,precisionSVD: 0.074567
on validation set: numFactors: 60,l: 0.200000,precisionSVD: 0.079657
on validation set: numFactors: 60,l: 0.400000,precisionSVD: 0.087122
on validation set: numFactors: 60,l: 0.600000,precisionSVD: 0.087122
on validation set: numFactors: 60,l: 0.800000,precisionSVD: 0.085171
on validation set: numFactors: 110,l: 0.000000,precisionSVD: 0.063709
on validation set: numFactors: 160,l: 0.000000,precisionSVD: 0.055989
on test set: numFactors: 60,l: 0.800000,precisionSVD: 0.128088

useBestKatz: 1,katz precision: 1.141330e-01

best beta: 1.000000e-12. useBestKatz: 1,katz precision: 1.199525e-01
on validation set: numFactors: 50,l: 0.000000,precisionSVD: 0.074992
on validation set: numFactors: 50,l: 0.200000,precisionSVD: 0.077961
on validation set: numFactors: 50,l: 0.400000,precisionSVD: 0.083220
on validation set: numFactors: 50,l: 0.600000,precisionSVD: 0.084662
on validation set: numFactors: 50,l: 0.800000,precisionSVD: 0.086105
on validation set: numFactors: 50,l: 1.000000,precisionSVD: 0.079912
```

Network	Parameters	Precision(%)
J	$iterations = 1$	0.63
J	$iterations = 3$	0.68
J	$iterations = 5$	0.7
C	$iterations = 1$	0.71
C	$iterations = 3$	0.75
C	$iterations = 5$	0.76
$C^\star$	$iterations = 1$	??
$C^\star$	$iterations = 5$	??

Table 4.3: Performance of iterative SVP algorithm on the combined graph and affiliation network, studied using the mask matrix M_α at $\alpha = 5$, on an *Artificial dataset* constructed using randomly selected user and group factors. Average performance over 2 trials: the variance is very low. The prediction accuracy as given by column 3 compares against 16.7%(1/6) when links are predicted at random.

```
on validation set: numFactors: 60,l: 0.000000,precisionSVD: 0.073040
on validation set: numFactors: 70,l: 0.000000,precisionSVD: 0.070580
on validation set: numFactors: 80,l: 0.000000,precisionSVD: 0.069817
on validation set: numFactors: 90,l: 0.000000,precisionSVD: 0.066339
on validation set: numFactors: 100,l: 0.000000,precisionSVD: 0.063115
on test set: numFactors: 50,l: 0.800000,precisionSVD: 0.130523
```

This is a 9.4% improvement [**Incomplete**]. We are yet to verify if this is the best Katz (beta, i.e.).

On the matrix C: on test set: numFactors: 100, l: 0.600000, precisionSVD: 0.134.

If we use $0.9B+0.1B^2$: numFactors: 50, l: 0.400000, precisionSVD: 0.131829; with sparsified B. With unsparsified B: numFactors: 80, l: 0.600000, precisionSVD: 0.132779.

4.6.3 Artificial dataset

4.7 SVD separate

Take reduced rank-k SVD decompositions of A and B separately to get $A = U_A S_A V_A^T$ and $B = U_B S_B V_B^T$. The idea is to find $\min_S \left\| A - [U_A U_B] S V_A^T \right\|$, and then use $[U_A U_B] B V_A^T$, the low rank approximation of A, as the score matrix. Using a validation set, we determined that the optimum number of factors, k, is 10. This yielded a precision of 0.09.

On the youtube data-set it yielded 0.10, for 5 factors - worse than katz.

Predictor	Time Test(s)	Time Validate(s)
SVD	28,27	642,625
$SVD2$		
$SVDSymmetric$	218	7899
$SVDJoint3$	661	
tKatz(A)	626 (sparse), 29(full)	1372(sparse)
tKatz(C)	183	2290

Table 4.4: Orkut: Running time statistics for various predictors.

4.8 Spatial approach

It took 46500 seconds for a single iteration: very slow compared to other methods. For 60 factors, $\lambda = 10^4, traceWt = 10^2$, it yielded 0.065 precision: worse than katz.

4.9 SVP approach

Using a Mask matrix generated by considering the links corresponding to the top katz scores as unknown, we get 8.8% for 1 iteration - same as find the SVD low rank approximation (no validation done): worse than using SVD. For 5 iterations, we got 86.29%.

However, given a metric which is better at identifying the top candidates, would using the svp approach to then pick some links among them yield results better than SVD on the training matrix alone? To do this, we constructed mask matrix to consider all test links plus 5 times as many random zeros as missing values. We use the low rank matrix which is the output of svp as a score matrix and then consider the top links as our prediction. For 1 iterations, we get: 68.3% precision: this is same as SVD. For 5 iterations, we get: 68.5%.

Note that using random prediciton would give us $1/6 \approx 16\%$ accuracy.

4.10 Time taken

4.10.1 Score matrix computation

4.10.2 Evaluation

Orkut: Evaluating a dense score matrix: 1075 s = 19 min.

Youtube: Evaluating a dense score matrix: 448s = min.

Predictor	Time Test(s)	Time Validate(s)
SVD		
$SVD2$	175	6408
$SVD3$	93	4503
$SVDSymmetric$	204	10996
tKatz(A)	14s	2028s (full)
tKatz(C)	59s	

Table 4.5: Youtube: Running time statistics for various predictors.

Predictor	Orkut	Youtube
SVDSymmetric	numFactors: 50, l: 0.800000	numFactors: 90, l: 1.000000
SVD4	numFactors: 60, l: 0.6	
SVD3	numFactors: 60, l: 0.6	numFactors: 70, l: 1
SVD2	numFactors: 60, l: 0.6	numFactors: 70, l: 1
tKatz(A)	$\beta = 10^{-12}$	$\beta = 10^{-12}$
tKatz(C)	$\beta = 10^{-2}, l = 0.2$	$\beta = 10^{-1}, l = 0.4$

Table 4.6: Best parameters for various predictors.

4.11 Best parameters

4.12 Summary

We have shown that the quality of link prediction in affiliation network can be considerably enhanced when the information from social network is properly exploited. We adapted many of the proposed methods for link prediction in social networks to link prediction in affiliation networks. Our application of supervised *katz* was, unlike in the social network link prediction case, unsuccessful. We formulated an optimization problem based on the linked matrix factorization approach and achieved a significant improvement in performance over the baseline; but that algorithm was slow. Using the method of obtaining latent factors from SVD approximation, we obtained the best performance. This best substantiates the claim that social network can be effectively exploited to do link prediction in affiliation networks.

Chapter 5

Conclusion

5.1 Findings

In this project, we investigated how information in the affiliation networks can be used in link prediction on the social network. Conversely, we also investigated ways of using information in the social network to perform link prediction in the affiliation network.

In this report, we detailed efforts at preparing suitable data for our experiments. We reported on some attempts at exploiting information in affiliation networks for link prediction in the social network. However, our experiments suggested that, while there is useful information for this task in the affiliation network beyond what may be determined from the social network itself, the benefit from using this information is insignificant. Similarly, we also reported on our attempts to use information from the social network for link prediction in the affiliation network. For this task, we were able to effectively exploit information from the social network.

5.1.1 Successful exploitation of a secondary network

In our experiments, we have been far more successful in exploiting information in the social network to do link prediction on the affiliation network, than in using information in the affiliation network in doing link prediction on the social network. How do we account for this?

Intuitively, one can argue that, given the evidence that two people belong to the same group, the probability of their being friends is not significantly higher. For example, two people being students in UTCS, or living in Austin does not increase the probability of their being friends significantly. On the other hand, given that two people are friends, the probability that they share some tastes and affiliations increases significantly. For example, given that two people are friends, then it is likely that they were co-located in some city at some point in the past.

We will now try to see if the numbers are in favor of this intuition.

In general, how useful a feature is friendship in the social network in defining a user's affiliations? To see this, we determined the following expectation: E[(friends with whom a user shares an affiliation)/(all people with whom a user shares an affiliation)]. In other words, among the people with whom a user shares his affiliations, what fraction are his friends? For the orkut dataset, this turns out to be 0.12, and for the youtube dataset, this turns out to be 0.085.

How useful a feature is the set of a user's affiliations in defining his circle of friends in the social network? To see this, we determined E[(affiliations a user shares with friends)/(affiliations of all his friends)]. In other words, among all the people who are a user's friends, what fraction of affiliations is shared? For the orkut dataset, this turns out to be 0.023, and for youtube, it is 0.03.

Contrasting the numbers presented above, we see that the intuition stated earlier was well founded.

Another way to look at the utility of the affiliation network (A) in social network (S) link prediction is to see how good 'common groups' (AA^T) is as a 'score matrix'. For orkut, it turns out to be 12 times better than random prediction and around 6.6 times worse than 'common neighbors'. We can contrast this with a similar measure of the utility of the social network in link prediction on the affiliation network. SA as a score matrix turns out to be around 20 times better than random prediction and only 1.76 times worse than AA^TA.

5.2 Future work

Our results indicate that, unlike in the social network link prediction case, information in a social network can be exploited in performing link prediction in the affiliation network. The results suggest that matrix completion methods, which have been so successful in the case of recommendation systems like Netflix can be easily extended to exploit a social network among users. This avenue of research should be explored further.

The overfitting problem we faced in our use of supervised Katz is also very intriguing. Exploring ways of avoiding this problem promises to be a good challenge.

5.3 Acknowledgements

We thank Dr. Alan Mislove [6] for giving us the Orkut data. We referred to, and used code by Wei Tang and Berkant Savas. We also thank Prof. Inderjit Dhillon, Zhengdong Lu and Prateek Jain for their insights.

Bibliography

[1] Wen-Yen Chen, Jon-Chyuan Chu, Junyi Luan, Hongjie Bai, Yi Wang, and Edward Y. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. pages 681–690, 2009.

[2] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, November 2007.

[3] Silvio Lattanzi and D. Sivakumar. Affiliation networks. 2009.

[4] Jure Leskovec. *Dynamics of large networks.* PhD thesis, CMU.

[5] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management.*

[6] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. October 2007.

[7] Berkant Savas, Zhengdong Lu, Wei Tang, and Inderjit S. Dhillon. Link prediction for social networks: A supervised approach. October 2009.

[8] Wei Tang, Zhengdong Lu, and Inderjit S. Dhillon. Clustering with multiple graphs. October 2009.