Influence based Link Prediction using Supervised Learning

Neeral Beladia
beladia@stanford.edu

Neera Vats
nvats@stanford.edu

1 Introduction

In recent years, there has been a lot of interest in understanding how nodes get influenced in networks, and how does their influence propagate to their neighbors. In online social networks, users receive feeds about their friend's online activities and when a user sees their social contacts performing an action such as joining an online community, that user may be influenced to perform the action themselves. In this project, we want to study how a node influences other nodes, and how a nodes' influence can be used to predict future links in the network. We propose an influence based supervised learning task and use it to predict new friend relationship links in a network.

2 Data Set

last.fm data: We collected data from a music website last.fm. It was founded in the United Kingdom in 2002. It has claimed over 40 million active users based in more than 190 countries. The API is pretty rich and we can get user information such as user's name, age and address, play history and their friend list. A user, once logged in on last.fm, can view in real-time his/her friend's music activity. The last.fm API allows us to call methods that respond in REST style xml. API documentation is available at: http://www.last.fm/api/intro. We collected datasets for users in Spain, United Kingdom and US.

3 Data Set Statistics

We collected data for about 10,000 users each for Spain and United Kingdom. The data distributions for number of friends, activity and number tracks for users from to Spain and UK look very similar. Figure:1 shows the distributions for UK users.

We could not collect the list of tracks that a user listened for all the users as some of them have set their track-list as private. However we found that this percentage was relatively small for both Spain and UK data sets. 3.6 % users from UK, and 5% of users from Spain as they have kept their track-list private.

Since the API limits the maximum size of track-list in the method call to 200, some users can have very high activity while others stay inactive. If a user is highly active, the maximum number of tracks in our dataset for this user can be a result of only one or two days of his activity. According to our algorithm, the influence score for such users will be high, even though we have less data about them in terms of number of days of activity. For the less active users the track listing might be the result of several days of activity and thus might be lower than the highly active users. However from the distribution shown below we see that most of the users have activity of between 100 to 400 days. One outlier in the tracks distribution graph represents the users who have kept their track-list private.

Both the datasets have very similar distribution for number of friends, user activity and number of tracks listened to by a user. The average number of friends for Spain data set is 130, and that for the UK data set is 100. Very few users have more than 400 friends. Graph shows that 40 users have 1000 friends. 1000 was our maximum limit on number of friends that can be fetched. These might be some

hub/celebrity nodes connected to lots of users.

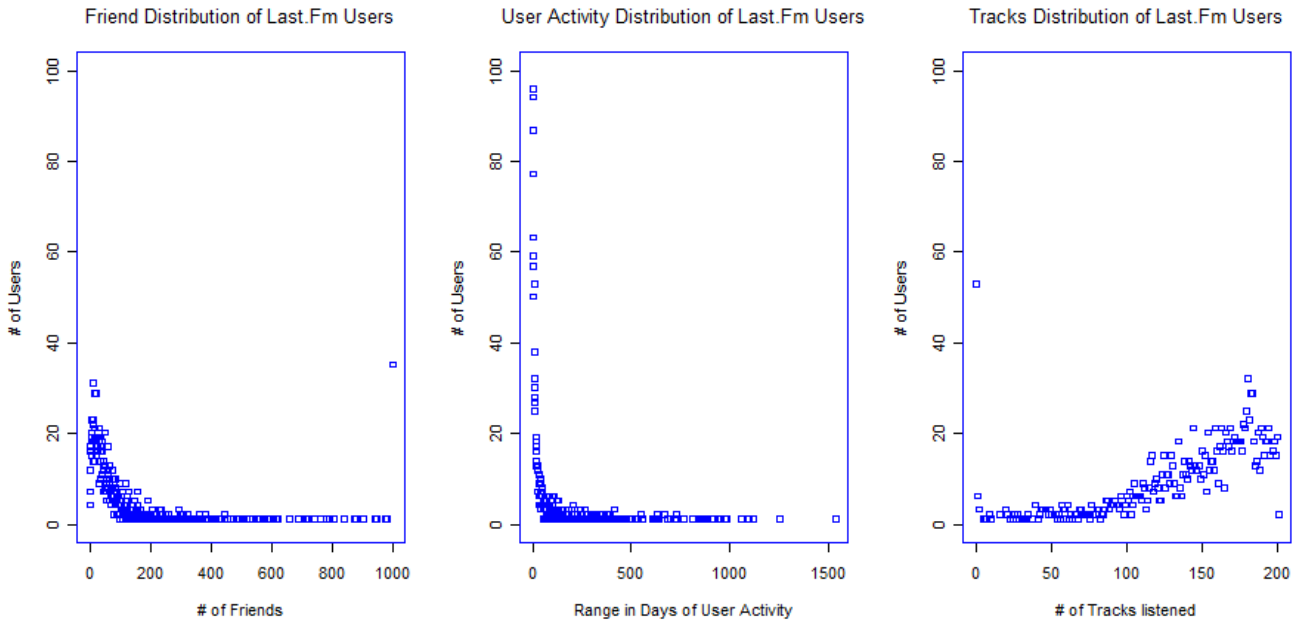

Figure 1 : UK dataset Distributions

4 Algorithm

How influence scores are calculated?

Consider a network G (V, E), where V represents set of users and E represents set on links between these users, a user *A* connects to user *B* at time $tc_{A,B}$. We say that node *B* is influenced by node *A*, if *B* performs same action *a*, that *A* performed *soon enough*. If user *A* performed action at $t_{A,a}$ and *B* performed the same action at time $t_{B,,a,,}$ we define influence of node A on node B as follows:

$$Inf_{A,B} = \frac{1}{|\{action\ a \mid a \in actions_B \ and\ t_{B,a} > tc_{A,B}\}|} * \sum_{a \in Act_{A,B}} f(t_{A,a}, t_{B,a}) \qquad [1]$$

$t_{A,a}$= *time action a was taken by node A*

$tc_{A,B}$= *time nodes A and B got connected*

$Act_{A,B} = \{action\ a \mid a \in actions_A \cap actions_B \ and\ tc_{A,B} \le t_{A,a} < t_{B,a}\}$

$f(t_1, t_2)$= *function measuring time decay of* t_2 *w.r.to* t_1

We choose $f(t_1, t_2)$ as $e^{(-(tB,a - t\,A,a))}$.

In our dataset user action is defined as listening to a track. We assume that influence scores by individual neighbors of a user are independent. Thus, if we have a model for capturing individual influences, we can compute the aggregated influence. We then define the aggregated influence of node A as:

$$Inf_A = \frac{\sum_{C \in neigh(A)} w_C * Inf_{A,C}}{\sum_{C \in neigh(A)} w_C} \qquad [2]$$

To normalize the level of activity of A's neighbors, we take a weighted average for all its neighbors. The intuition for this normalization is that node A is highly influential if node A can influence its highly active neighbor(s). In our dataset, nodes are listeners and we define influence of listener A on listener B as how often and soon enough, user B listens to same songs that A has recently listened to.

5 Feature Engineering

Choosing an appropriate feature set is the most critical task of any learning algorithm. Our training and test data will consist of observations representing edge/link between a pair of nodes. We chose the following features for our predictor:

5.1 Aggregated Features

- ***Aggregated influence score of user A, Inf_A:*** We calculate this score as defined in Equation [2].
- ***Aggregated influence score of user B, Inf_B:*** We calculate this score as defined in Equation [2].
- ***Average aggregated influence scores of common neighbors:*** This feature represents the connectivity of user A and user B**.** It is computed by taking average of scores to common neighbors of User A and User B.
- ***Influence of Users at 2 hops and 3 hops distance:*** User A and user B's 2 hop distance is calculated via their common neighbor C (A->C->B). We calculate influence of nodes at 2-hop distance as:

$$\frac{1}{|neigh(A) \cap neigh(B)|} \sum_{c \in neigh(A) \cap neigh(B)} Inf_{A,C} + Inf_{C,B}$$

  The above formula can be extended to calculate influence of nodes which are 3 hops away.

- ***Influence concentration on common neighbors:*** We calculate this as a ratio of A's influence on common neighbors of A and B, with A's influence on its own neighbors. It is defined as:

$$\frac{\sum_{C \in neigh(A) \cap neigh(B)} Inf_{A,C}}{\sum_{C \in neigh(A)} Inf_{A,C}}$$

  Similarly, we calculate the concentration of B's influence on common neighbors.

- In order to deal with the cold-start problem, i.e., if two nodes are the new nodes, with no or very little activity data, we will use the attributes and similarity features of the nodes and edges along with the influence based features.

5.2 Similarity Based Features: For link prediction, features that represent some form of similarity between the pair of users have shown to be very effective [3]. We chose the following similarity based features:

- ***Track Genre Similarity:*** We measure the proximity between two users by similarities between

the genre of tracks they have listened to. From our entire set of tracks we compute top 10 genre that are most listened to. Based on the track-list of each user we calculate their genre scores as a normalized vector of frequencies of top genre. We calculate the track similarity between two users as the *Euclidian distance* between genre score vectors.

- ***Track Similarity:*** Two users have similar music interests if their track lists have a good amount of overlap.
  We define track similarities as follows:
  - Overall track similarity between A's and B's track-list $\left| \frac{tracklistA \cap tracklistB}{|tracklistA \cup tracklistB|} \right|$
  - Track similarity for user A $\left| \frac{tracklistA \cap tracklistB}{|tracklistA|} \right|$
  - Track similarity for user B $\left| \frac{tracklistA \cap tracklistB}{|tracklistB|} \right|$
- ***Friend-list Similarity, Album Similarity and Artist Similarity:*** These scores are calculated by using formulae similar to those for track similarity.

5.3 Topological Features

- ***Shortest Path Length:*** This feature is one of the most significant in link prediction as we found in our research. [1] and [5] showed that in an online social network most of the new connections are made between close neighbors. We used the smallest hop count as the shortest path distance between two nodes. We also considered calculating weighted influence score based path length between two nodes. This can be a directed path length in which each edge has a weight equal to the influence score between the two nodes connected by this edge. Since this is very expensive to compute we did not include it in our current set of experiments.

6 Classification Algorithms

There are numerous machine learning algorithms for classification like Decision Trees, SVM, ANN, Naive Bayes, etc. Each of these has its own characteristics and underlying assumptions. For this report we have used kernel SVM, Random Forests, Gradient Boosted Machines, Logistic Regression and adaBoost to predict if a pair of users are connected or not.

In our results the accuracy corresponds to the proportion of test examples correctly classified, and this is our primary goal in the task of supervised learning. This results in a relatively higher cost of misclassifying a positive example (i.e. connected pair of users) compared to that of misclassifying a negative example. To compensate for this, we have performed a weighted bagging while performing Gradient Boosted Machines, so that we have relatively higher weight in training set for positive examples. We compared the performance of the above classification algorithms using different performance metrics like Area Under Curve(AUC), accuracy, precision-recall, sensitivity-specificity. We used 5-fold cross validation for the results reported. We used F-measure such that it takes both the precision and recall into account to understand the true accuracy of the model on the test data.

We also aggregated the predictions from the 5 classifiers to serve as an ensemble e1. By aggregating the predictions from different classifiers the resulting model will not be susceptible to variance errors.

7 Results and Findings

From the data of about 10,000 users each for Spain and UK, we randomly sampled 50,000 pairs of connected and disconnected pairs of nodes. This served as our base data. We randomly sampled about 2/3rd of this base data to serve as the training data for the model and the rest as the test data. In both the datasets, we kept the counts of positive classes and the negative classes the same. Therefore, a baseline classifier would have accuracy around 50% by classifying all the testing data points to be equal to 1 or 0. We used R packages for modeling and evaluation.

Authors in [3] show promising results with similarity based features for their link prediction problem. We implemented most of their similarity features which we were able to apply to our data set. To compare how influence based feature performed in comparison with similarity based features, we did experiments with similarity and topological features, followed by experiments with influence and topological features. The performance comparison of each of the learned model on the unseen pairs of nodes is shown below. Table 1 shows performance metrics with similarity and topological features and Table 2 shows metrics with influence and topological features.

All our models achieved accuracy above 80%, which indicates that our features have good discriminating ability. Results with influence based features are very close to the results reported using similarity based features. The ensemble method gives a slight improvement in recall measure for influenced based features. Recall measures are of high importance in link prediction because they represent the proportion of actual links which are correctly identified.

All the models have similar precision-recall behavior. Their precision value is higher than recall value for predicted links. This means that the models have more false negatives than false positives i.e. the models are missing actual links more than they are predicting false links. For LastFm it might make sense, because some users might be connected due to reasons that are not captured by any of our features, such as they are acquaintance or classmates etc.

The Area Under Curve (AUC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. All the models have high score for AUC. Since it is a binary classification problem, Recall and Sensitivity measure have the same values.

Method	AUC	Accuracy	F-measure	Precision	Recall	Sensitivity	Specificity
Kernel SVM	0.89	88.54	88.26	91.04	85.65	85.65	91.47
Random Forest	0.89	88.72	88.39	91.68	85.32	85.32	92.17
Gradient Boosted Machines	0.89	88.82	88.52	91.55	85.68	85.68	92.00
Logistic Regression	0.89	88.61	88.32	91.19	85.62	85.62	91.63
AdaBoost	0.89	88.71	88.33	91.98	84.95	84.95	92.50
Ensemble e1	0.89	88.77	88.44	85.38	91.72	91.72	96.17
Ensemble e1 = argmax (kernel SVM, Random Forest, GBM, Logistic Regr, AdaBoost)							

Table 1: Classifier performance with similarity and topological features

Method	AUC	Accuracy	F-measure	Precision	Recall	Sensitivity	Specificity
Kernel SVM	0.89	88.49	88.20	91.09	85.48	85.48	91.53
Random Forest	0.89	88.81	88.41	92.27	84.85	84.85	92.80
Gradient Boosted Machines	0.89	88.79	88.43	91.93	85.19	85.19	92.44
Logistic Regression	0.89	88.67	88.29	91.97	84.89	84.89	92.50
AdaBoost	0.89	88.74	88.33	92.29	84.49	84.49	92.84
Ensemble e1	0.89	88.81	88.41	84.92	92.21	92.21	85.87
Ensemble e1 = argmax (kernel SVM, Random Forest, GBM, Logistic Regr, AdaBoost)							

Table 2: Classifier performance with influence and topological features

7.1 Feature Selection

In order to understand our dataset better, in terms of the importance and relevancy of each of the features, we used the Forward Stepwise Rank, Backward Stepwise and All Subset methods for feature selection. Forward stepwise selection starts with an empty list of predictors and fills up the list one predictor at a time, choosing the predictor (not already in the list) that minimizes some objective error function for the base model. We chose Ordinary Least Square as our base model and SSE as the objective error function. Backward stepwise selection works in a manner quite similar to the Forward Stepwise method, in which it starts with a list consisting of all predictors and removes one at time, the predictor that when removed minimizes the SSE the most. All subset selection chooses top k subsets from each of the 2p - 1 combinations (p = # of predictors) of predictor selections. Table 3 shows the result of these methods along with feature type. Least Squared Error of these methods were: forward selection: 0.098, Backward selection: 0.0987; All-subset : 0.0989.

We see that the shortest path feature is the most significant among topological features. Influence of common neighbors of A and B is more significant among influence features. We can infer the reason behind these rankings by looking at the distribution of the features. Figure 2 shows the distribution of some of the features on log-log scale. The blue curves in the graphs show value for the connected nodes, and the red curves show values for the disconnected nodes. From these graphs we see that most of the features follow power law distribution.

The reason behind the strength of the shortest path feature can be seen in its distribution plot. For connected users the mean distance between the user pair is 0.5 on the log scale, whereas the same for the non-connected users is 1.2. Due to this noticeable difference in distribution, classifiers are able to discern that connected nodes are concentrated towards low feature value and non-connected users have high feature value. Similarly we can explain why friend similarity, genre similarity and influence of common neighbors of A and B are ranked higher.

The average influence on nodes at 3 hop distance *(avg3hopAB)* is ranked higher than the average influence on nodes at 2 hop distance *(avg2hopAB)*. Intuitively this looks incorrect, because most of the

new links are made between friends of friends. Distributions in Figure 2 also do not explain this. To verify this further, in the training data, we counted the number of non-zero values of influence on 2 hop neighbors and found that all the values up to 75th percentile are zero. Whereas, values for influence on 3 hop neighbors have non-zero scores for 50% of the training data. We can improve this by either collecting more friend data for each user or collecting data more strategically depending on user connections.

Table 3: Feature Ranking

Feature	Type	Forward Stepwise Rank	Backward Stepwise	All-subset	Feature	Type	Forward Stepwise Rank	Backward Stepwise	All-subset
shortPath	Topological	1	Y	Y	infA	Influence	11	Y	Y
frndSimB	Similarity	2	Y		avg3hopBA	Influence	12	Y	Y
frndSimA	Similarity	3	Y		Avg2hopAB	Influence	13	Y	Y
genreSim	Similarity	4	Y	Y	infAonB	Influence	14	Y	Y
infCmnNbhAB	Influence	5			trackSimB	Similarity	15	Y	Y
infB	Influence	6	Y	Y	trackSimAll	Similarity	16	Y	Y
artistSimB	Similarity	7	Y		tagSimALL	Similarity	17	Y	Y
frndSimALL	Similarity	8	Y	Y	trackSimA	Similarity	18	Y	Y
infConcAonCmnNbh	Influence	9		Y	tagSimB	Similarity	19	Y	Y
infConcBonCmnNbh	Influence	10	Y	Y					

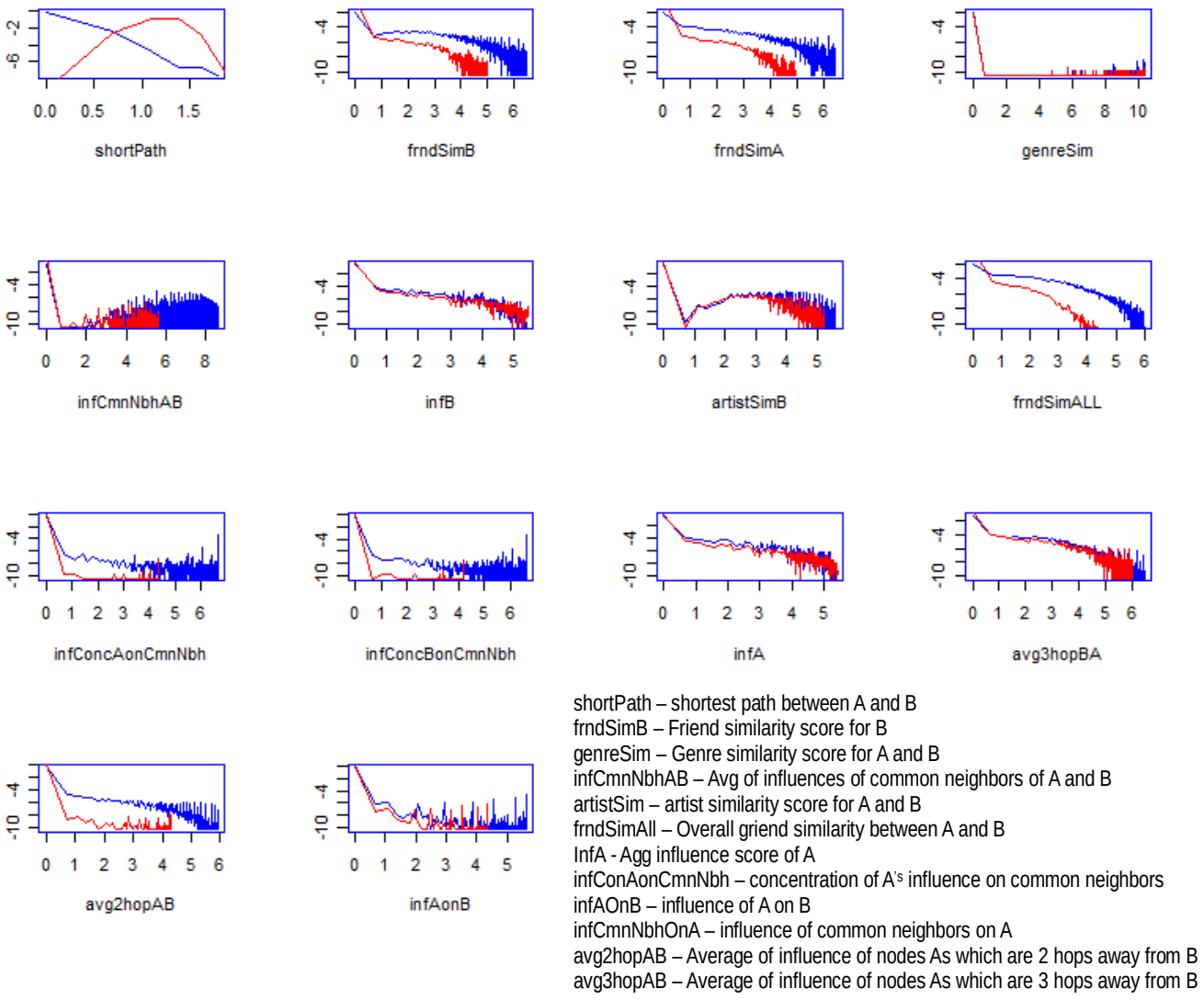

Figure 2 : Distribution plots for feature values

7.2 Correlation Chart

Correlation chart of the features is shown in Figure 4. Red color represents non-connected users and blue color represents connected users. The darkness of both the colors is proportional to the correlation of the feature. From the graph we see that shortest path, friend similarity, genre similarity and aggregate influence of common neighbors(*InfCmnNbhAB*) are highly correlated with the response feature "connected".

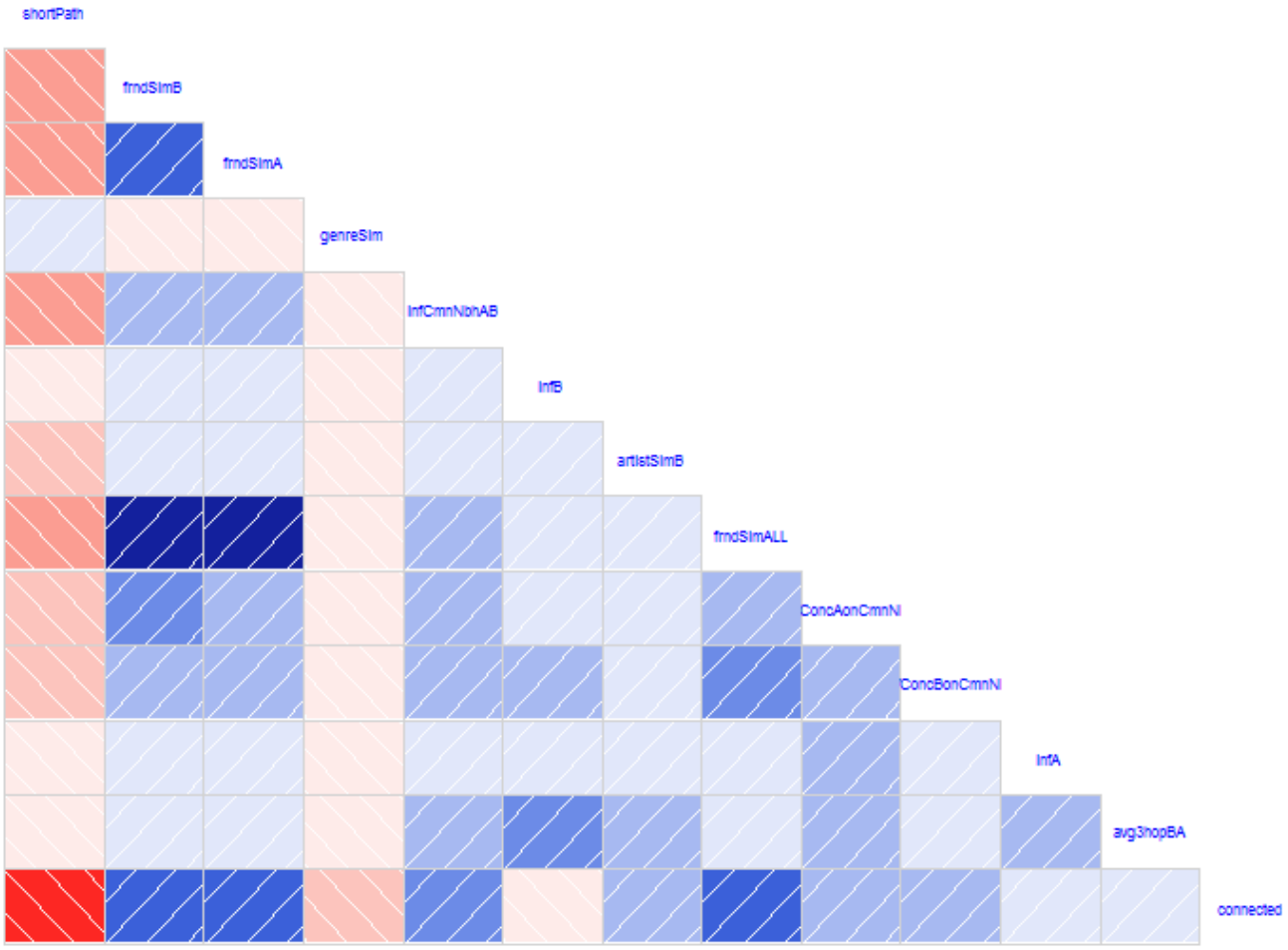

Figure 3: Feature Correlation Chart

8 General difficulties and Assumptions

- In our dataset, we have a timestamp for when a user creates his/her account but it doesn't have information about when two users get connected. Therefore, we are assuming that all the connections for a user are made either at the time the user joined or before the starting time of our sampling. Since our influence based predictors required actions to be timestamped, we were not able to utilize 'becoming friend with other user' as an action.

- By sampling data based on geographical location we make an assumption that it is more probable for a user to connect to a friend located closer physically i.e. the number of short range links for a user are more probable than longer range links. This however can introduce some bias into our dataset. We plan to run more experiments with larger datasets and see how our accuracy changes with datasets from other locations.

- If user A and user B listen to a track T at time T_a and T_b respectively, and $T_a < T_b$. We calculate influence of user A, on user B as a function of (T_b - T_a). However, it is possible that User B had already listened to track T at some other time $T_b' < T_a$, and he already knew about the track T. But on the online social network interface users get real time feeds about their friends activities, so even though they already know about an event, they might still be more interested in repeating it after seeing that one of their friends have done it recently.

9 Future Work

The last.fm API did not provide a way to know when(timestamp) two users were connected. We believe that in deducing true influence of a user on another user, it's useful if we know when 2 users were connected to reduce the weight of actions performed before the connection was established. We plan to experiment on datasets that make this information available.

Our work can be extended to study supervised link prediction in context based influence graphs. i.e. from an undirected graph of friend relationships, we generate a directed weighted graph, where the weight refers to the influence of a node on another node. For example, context can be users who listened to at least one "Rock" song.

We also plan to experiment using influence based link predictors in an unsupervised setting to define a soft neighborhood (cluster) around each user to recommend friend relationships.

10 Related Work

Supervised methods for link predictions were extensively evaluated in [2], [3] and [4]. In [2] and [3] classifier uses node similarity, network's topological properties as their feature vectors. These studies show that supervised learning techniques produce impressive results for link prediction. We compared our results with [3] and showed that influence based features show equally good results. Authors in [6] used roughly the same influenced based approach to study the influence diffusion within a network using probabilistic models. Even though their influence scores calculations are very similar to ours, our problem formulation, goals and data sets are different from theirs. The unsupervised methods for link prediction were extensively evaluated by Liben-Nowell and Kleinberg in [6], they found that the Adamic-Adar measure of node similarity [1] performed best. Link prediction in supervised machine learning setting was studied by the relational learning community in [7]. Their approach performs well but, the challenge with these approaches is primarily scalability. Another recent work by Backstrom, et al. [4], although different for our supervised learning method, also used to predicts new links in a social network. They use supervised random walks to learn edge strengths within a graph, these edge strengths are the used for predicting new links.

11 References

[1] Liben-Nowell, D. and Kleinberg, J. 2003. The link prediction problem for social networks. In Proceedings of the *Twelfth international Conference on information and Knowledge Management (New Orleans, LA, USA, November 03 – 08, 2003). CIKM '03. ACM, New York, NY,* 556-559.

[2] Supervised Link Prediction in Weighted Networks, Hially Rodrigues de Sá and Ricardo B. C. Prudêncio, *The 2011 International Joint Conference on Neural Networks (IJCNN)*
[3] Hasan, M., Chaoji, V., Salem, S., and Zaki, M. J., (2006). "Link Prediction using Supervised Learning," *Workshop on Link Analysis, Counter-terrorism and Security (with SIAM Data Mining Conference), Bethesda, MD,* April 2006.
[4] L. Backstrom, J. Leskovec, "Supervised Random Walks: Predicting and Recommending Links in Social Networks", *ACM Internation Conference on Web Search and Data Mining (WSDM),* 2011
[5] A. Goyal, F. Bonchi, L.V.S. Lakshmanan. Learning influence probabilities in social networks. *In Proc. WSDM*, 2010.
[6] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *IKM '03*, pages 556–559, 2003
[7] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *NIPS '03*, 2003.