Spring 5-22-2019

Network Alignment In Heterogeneous Social Networks

Priyanka Kasbekar
San Jose State University

Network Alignment In Heterogeneous Social Networks

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for

CS298

by

Priyanka Kasbekar

Spring 2019

The Designated Project Committee Approves the Project Titled

Network Alignment In Heterogeneous Social Networks

by

Priyanka Kasbekar

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

Spring 2019

Dr. Katerina Potika	Department of Computer Science
Dr. Christopher Pollett	Department of Computer Science
Prof. Rula Khayrallah	Department of Computer Science

ABSTRACT

Network Alignment In Heterogeneous Social Networks

by Priyanka Kasbekar

Online Social Networks (OSN) have numerous applications and an ever growing user base. This has led to users being a part of multiple social networks at the same time. Identifying a similar user from one social network on another social network will give information about a user's behavior on different platforms. It further helps in community detection and link prediction tasks. The process of identifying or aligning users in multiple networks is called Network Alignment. More the information we have about the nodes / users better the results of Network Alignment. Unlike other related work in this field that use features like location, timestamp, bag of words, our proposed solution to the Network Alignment problem primarily uses information that is easily available which is the topology of the given network. We look to improve the alignment results by using more information on users like username and profile image features. Experiments are performed to compare the proposed solution in both unsupervised and supervised setting.

Keywords: Online Social Networks, Network Alignment, Supervised learning, Unsupervised learning

ACKNOWLEDGMENTS

I would like to thank my project advisor Dr.Katerina Potika for her constant guidance, motivation and support. She helped me through her valuable inputs and guided my research in the right direction. I would also like to thank my committee members Dr. Christopher Pollett and Prof. Rula Khayrallah for their valuable inputs and feedback on the project. A special thanks to the authors of the paper "Unsupervised User Identity Linkage via Factoid Embedding" [1] Wei Xie, Xin Mu, Roy Ka-Wei Lee, Feida Zhu and Ee-Peng Lim from the Singapore Management University, Singapore for their dataset and their helpful insights on it which immensely helped my research. Last but not the least, I would like to thank my family, peers and friends for their encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The availability and ease of access to the internet have made Online Social Networks (OSN) an integral part of our lives. OSN are used for many purposes ranging from sharing media, reviews, news, opinions to finding job opportunities, cabs, dates, asking questions and more applications added every day. The multifaceted role of OSN have led them to have a huge user base. Users are often members of multiple social networks as a single social network may not offer everything a user wants. For example, Facebook is used to share media, make new friends, Yelp is used to review restaurants, LinkedIn is used to search jobs, Twitter is used to voice opinions [2].

1.1 Network Alignment

Given a node in one network, the problem of identifying the same node or a similar node in another network is called Network Alignment. The same process when applied to social networks will help identify a user on different social networks. The users identified in the process of Network Alignment are called anchor nodes and the edges between them across the networks are called anchor links/edges. If all the nodes/users of one network are aligned with nodes/users of other network, then the networks are said to be fully aligned. Fully aligned social networks are unlikely. For example, it is not mandatory that a person who is a Facebook user will be a Twitter user too. Social networks are partially aligned where every user from one network is not necessarily an anchor node. Some users can be unaligned [3].

1.2 Heterogeneous Social Networks

Due to the diverse services OSN offer, they contain different types of nodes apart from user nodes. For example, Facebook, apart from user nodes, contains locations, posts nodes. Edges are formed between user node and user node, user node and a post node, user node and a location node etc. These other type of nodes are also called information entities [4]. Hence, each of such OSN can be considered as a Heterogeneous network.

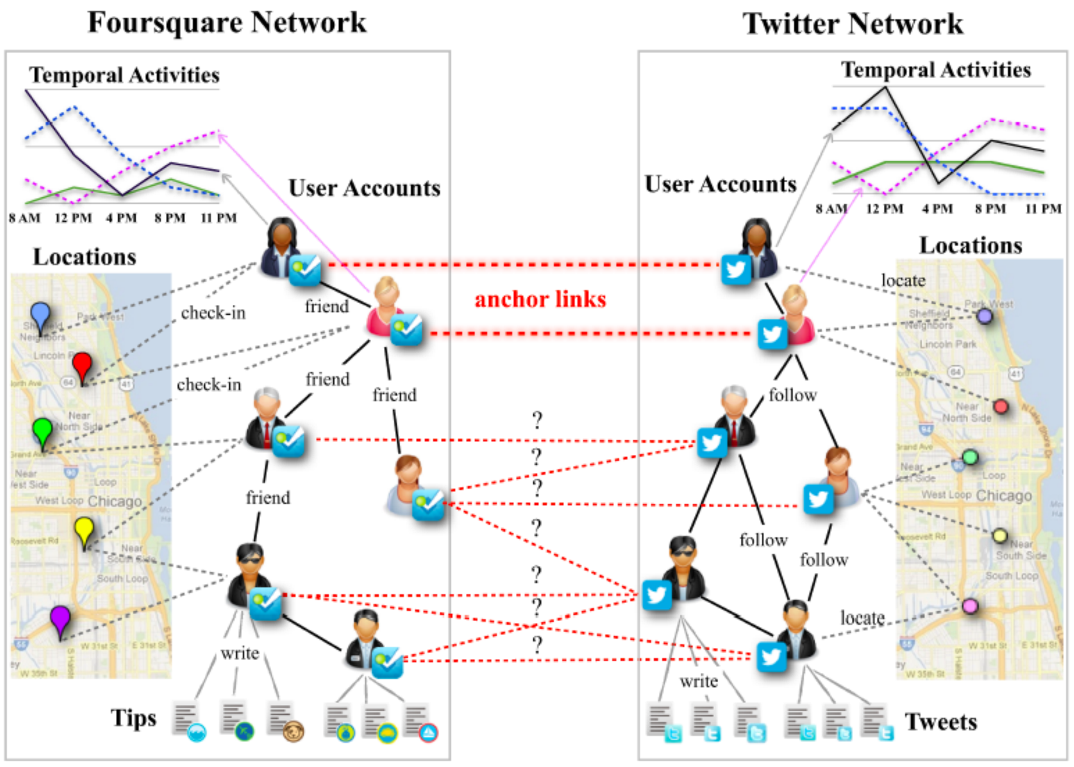


Figure 1: Depiction Of Heterogeneous Social Networks And Anchor Links [2].

The question marks in Figure 1 represent the anchor links that are to be determined and the dashed lines represent the already known anchor links. Each user node in a network is connected to different kinds of nodes. For example, as seen in Figure 1, each user node in a foursquare network is connected to another user node, a tip node or a location node. A tip in foursquare network is a small note that a user can leave on a place describing their experience at that location. Hence a user node can be connected to many tips that he/she have left on many locations. Location nodes are the foursquare check-ins made by a user node.

1.3 Applications of the Network Alignment Problem

The problem of Network Alignment in social networks has generated a lot of interest as it helps researchers study the social behavioral pattern of the same user in different social networks [4]. Aligning social networks have more applications like

- Community Detection - When a well established social network is aligned with a fairly new social network, the communities from the first network can be used to detect communities on the new social network [4].
- Information from multiple aligned networks can give better indication of the social relations and choices of users which can be used to enhance link prediction effectiveness [4].
- Viral Marketing - Learning preferences of a user from one network can be used to display related, useful ads for the similar user/users in another network [4].
- Information Diffusion - Network Alignment enables the researchers to study the information diffusion patterns in both the networks [4]. This will enhance the information required for identifying influential nodes in a network.
- Friend Recommendations - If a given user uses the source network frequently and is fairly new on the target network, Network Alignment can give insights for friend recommendation for the user on the target network. This is possible only if Network Alignment of the source and target network is done on the basis of features which are not dependent on friendship/topology information.

1.4 Challenges in extracting Information Entities

To solve the problem of Network Alignment, generally topology based features like friendships as well as temporal activities like check-ins, contents of tweets/tips are used. For most OSN, accessing user specific information like check-ins, tips through their publicly available APIs is challenging. This is due to the restrictive nature of the APIs. Most of the methods provided by the APIs require private authentication tokens which is not available. Hence, to solve the problem of Network Alignment, we can't completely rely on these features. In this project, we compare various supervised and unsupervised approaches to Network Alignment using features based on the topology of the networks. We additionally use some features like username embedding similarity, profile image feature vector similarity to train our models. We perform experiments on two pairs of social networks, Facebook-Twitter, Foursquare-Twitter from which we extract features like common neighbors, jaccard coefficient, adamic/adar measure. These topology based features are independently used in unsupervised training. The same features combined are used to train a supervised model. We observe the effects of using these features separately and in a combined manner.

CHAPTER 2

Related Work

The Network Alignment problem bears a lot of similarity with the maximum sub graph or the bipartite graph matching problem [5] [6]. Apart from the field of OSN, network alignment has been applied to other fields of study like Bio-informatics on networks like protein-protein interaction network which are mostly homogeneous networks [7] [8]. Two main approaches have been followed in aligning social networks.

- Matrix operations based methods
- Supervised Learning methods

One of the related work in this field takes an Unsupervised approach to Network Alignment using a special kind of embedding called factoid embedding.

2.1 Matrix based approaches

This approach involves utilizing the topology based information through adjacency matrices and degree matrices of the networks. This information combined with some simple matrix operations like dot product, transpose, normalization, Kronecker product, matrix addition or subtraction is used to perform Network Alignment. Following are some related work using this approach

- **Fast Attributed Network Alignment** - FINAL (Authors - Si Zhang,Hanghang Tong) [9] - In this paper, Network Alignment is done using topology information and network and edge attributes from the network. In a network like DBLP co-authorship, node attributes can be the most recent conference attended by an author and an edge attribute can be the papers in which both authors have co-authored presented in the same conference. Network Alignment is done using Alignment Consistency Principle which states that the alignment between two pairs of nodes across networks should be similar if the pairs of nodes are similar to each other [9]. For Alignment Consistency Principle to hold the following three principles should hold good.

  - Topology Consistency - If two nodes are close neighbors in one network, the respective aligned nodes should also be close neighbors in the other network [9].
  - Node Attribute Consistency - Two nodes aligned with each other should have similar node attributes [9].
  - Edge Attribute Consistency - If two nodes in the given network have an edge attribute, the respective aligned nodes in the second network should have similar edge attribute between them [9].

The Network Alignment principle begins with a initial similarity matrix H of size $n_2 * n_1$, where n_2 is the number of nodes in the second network and n_1 is the number of nodes in the first network. After performing operations like matrix normalization, element wise multiplication, transpose operations for few iterations on the similarity matrix, adjacency matrices, node attribute matrices and edge attribute matrices, the similarity matrix will converge with values where each value $H_{i,j}$ represents the similarity between node i in network 1 and node j in network 2.

- **Multiple Anonymized Social Networks Alignment** [10] - (Authors - Jiawei Zhang, Philip S. Yu) - This paper also uses topology of the input networks to perform Network Alignment. The method used in this paper assumes there are no existing known anchor links. The input networks have no information about user attributes and hence the Network Alignment is completely based on topology. Start with a binary transitional matrix T where rows correspond to users from input network 1 and columns correspond to users from input network 2. Use an iterative approach to finally arrive at a point where $T^{i,j} == T^{j,i}$. This represents the matrix that has anchor links between the input graphs.

This paper uses the concept of social links. A social links exists between two users in a given network if they are connected to each other. To perform Network Alignment the social links from one network are projected on to another network. The rule of friendship consistency is used to measure the correctness of Network Alignment. Friendship consistency is the number of shared social links between the mapped links from 1st input network onto the second input network and the number of social links originally in the second input network [10].

The method used in this paper can be used to align multiple networks at once. At the end of Network Alignment, transitive property should hold. Transitive property states that when anchor links are identified between network 1 and network 2 and network 2 and network 3, the same links can be identified as anchor links between network 1 and network 3 [10].

- **Network Similarity Decomposition (NSD)** [11]: A Fast and Scalable Approach to Network Alignment - (Authors - Giorgos Kollias, Ananth Y. Grama, Shahin Mohammadi) - This paper uses the concept of page rank to determine the anchor links between two networks. In a single network setting, to find the most important node a page rank algorithm is used. A node is important if it is linked by other important nodes. In a multiple network setting, a tensor product of two input adjacency matrices is calculated. Page rank algorithm is applied on this new matrix. This process works with the principle that a node with a high page rank will be aligned with another node with a high page rank. Hence, obtaining a high page rank on the tensor product of two input graphs would mean that the respective nodes are aligned. This would also mean that similarity should also be preserved over the neighbors.

The tensor product G_c between two matrices G_a and G_b is calculated such that an edge between two nodes c_i and c_j exists in G_c only if there is a edge between a_i and a_j in G_a and there is an edge between b_i and b_j in G_b [11].

Two approaches are used to calculate the page rank of the nodes:

– HITS algorithm - Start with a initial similarity matrix X containing all 1s. Iteratively use the following equation until X converges to a fixed set of values.Normalize X after every step.

$$X = BXA^T + B^TXA$$

– PageRank algorithm - Start with an initial similarity matrix X and iteratively apply the following formula until X converges.

$$x = \alpha \tilde{A} \otimes \tilde{B} x + (1 - \alpha) h$$

. Here x = vec(X) and h = vec(H). The vec operation stacks columns of the matrix. Here the matrix H can be considered as the elemental similarity matrix. This idea is based on the IsoRank algorithm used to align protein - protein interaction networks [7].In the IsoRank algorithm, H is a matrix with blast scores that represent elemental similarity between proteins. In this paper, Single Value Decomposition(SVD) is used to construct H.

2.2 Supervised Learning approaches

These approaches use feature vectors constructed from node attributes. The feature vectors are used to train models and obtain similarity scores. Based on the similarity scores the anchor links can be inferred using different matching techniques.

- **Inferring anchor links across multiple heterogeneous social networks** (Authors - X. Kong, J. Zhang, P. Yu) [2] - This paper first assumes that there are some existing anchor links between the input networks which are used for training. Social features like common neighbors, jaccard coefficient, adamic measure are extracted. Users usually tend to post at the same time and at similar locations. The similarity between location vectors can be used as features. Users tend to post similar contents on multiple social networks. The similarity between content vectors can be used as features. The similarity between time slots of when the user's post their content can also be used as features [2].

  These extracted features are used to train a binary classifier to get similarity scores between nodes. The paper uses stable matching to infer anchor links from the similarity scores. Each node in the given graph is matched to the best possible node in the second graph [2]. As stable matching is used, we can assume that the alignment is performed on input networks where for each node in the first graph, there is a matching node in the second graph. This means that the input graphs are fully aligned.

- **Partial Network Alignment with Generic Stable Matching** [3] (Authors - Jiawei Zhang, Weixiang Shao, Senzhang Wang, Xiangnan Kong, Philip S. Yu) This paper uses the concept of Anchor Meta Paths for Network Alignment. It assumes that the input networks are partially aligned which means that for every node in the given network the matching node in the other network may or may not be present. As we are working with heterogeneous networks, different types of nodes like user, location, timestamp are present and various types of relation exists between nodes like follow, create, checkin. Using the nodes and relations, anchor meta paths are created. For example, intra network meta paths can be of the form

$$\Phi = T_1 \xrightarrow{R_1} T_2 \xrightarrow{R_2} ...T_k \xrightarrow{R_k - 1}$$

where $T_i \in T_G$ and $i \in \{1, 2, 3...k\}$ and $R_i \in R_G$ and $i \in \{1, 2, 3...k-1\}$. $T_G and R_G$ are set of node types and link types respectively [3]. Anchor meta paths between two different networks is of the form

$$User^i \xrightarrow{follows} User^i \leftrightarrow anchorUser^j \xrightarrow{follows} User^j$$

The anchor links needn't be only between user node types but they always begin and end with user node types. For example,

$$User^i \xrightarrow{writes} Post^i \xrightarrow{checkinat} Location \xleftarrow{check-inat} Post^j \xleftarrow{writes} User^j$$

Anchor adjacency scores is the number of anchor meta paths between different set of users. Using these scores a supervised link prediction model is built. The link prediction can be +1 when anchor link is present and -1 otherwise [3]. To infer the anchor links between the networks, generic stable matching is used. This method is different from the traditional stable matching as it doesn't require every unmatched node to be matched for the matching process to be complete. This is useful as the input networks used in the paper are partially aligned. With this process, a node with no matching node in the other network is matched to itself(self matched with a high score) [3].

- **Exploring Identical Users on GitHub and Stack Overflow** [12] (Authors - Takahiro Komamizu, Yasuhiro Hayase, Toshiyuki Amagasa, Hiroyuki Kitagawa) This paper illustrates a method of identifying common users between two of the technical and programming related platforms of Github and Stackoverflow. To find the known common users, they use the email addresses of the users. These are considered as known anchor links. Similar attributes like username, words from project description and questions, account creation date are extracted from both the networks. Two types of similarity functions are used one to measure the text features and other to measure the date and time features. Words are considered to follow a bag of words and cosine similarity function is applied on them. Date and time are compared using inverse of difference between the dates [12]. The paper also experiments with various learning mechanisms like linear regression, k-nearest neighbors, random forests and logistic regression.

2.3 Unsupervised Approach

- **Unsupervised User Identity Linkage via Factoid Embedding** [1](Authors - Wei Xie, Xin Mu, Roy Ka-Wei Lee, Feida Zhu and Ee-Peng Lim) This paper takes the unsupervised approach to Network Alignment. The idea of the paper is to use every piece of information available on the users which can help in identifying the users. The available information like username, profile image is represented using factoids. Each factoid is in the form of a triplet which can take two forms.

  1. user_identity – predicate – user_identity
  2. user_identity – predicate – object

  The objects are properties associated with the users like user name, screen name, profile image. Every user is associated with a unique id. An example of a $\mathsf{user\_identity} - \mathsf{predicate} - \mathsf{object}$ factoid is $\mathsf{1} - \mathsf{has\_name} - \mathsf{Ann}$. This means that a user with unique_id 1 has name Ann. An example of $\mathsf{user\_identity} - \mathsf{predicate} - \mathsf{user\_identity}$ factoid is $\mathsf{1} - \mathsf{follows} - \mathsf{3}$. This means that a user with unique_id 1 follows another user with unique_id 3. The embedding of user objects involves finding the similarity between the objects. For every predicate, a similarity matrix is created. For example, for the predicate 'has_name', a similarity matrix is created where the similarity of each user with other users is measured. For name predicate, Jaro-Winkler distance is used to measure similarity. For image predicate, deep learning techniques are used to generate image embedding vectors. The users who are similar will be close in the embedding space. To learn the object embeddings, following objective function is used.

$error_{pred} = \sum_{i,j}((v_{o_i})^T(v_{o_j}) - S_{i,j}^{pred})^2$

where v_o is object embedding vector like username embedding vector. The object embedding vector is learnt through minimizing the $\mathsf{error}_{\mathsf{pred}}$

CHAPTER 3

Methodology

3.1 Problem Definition

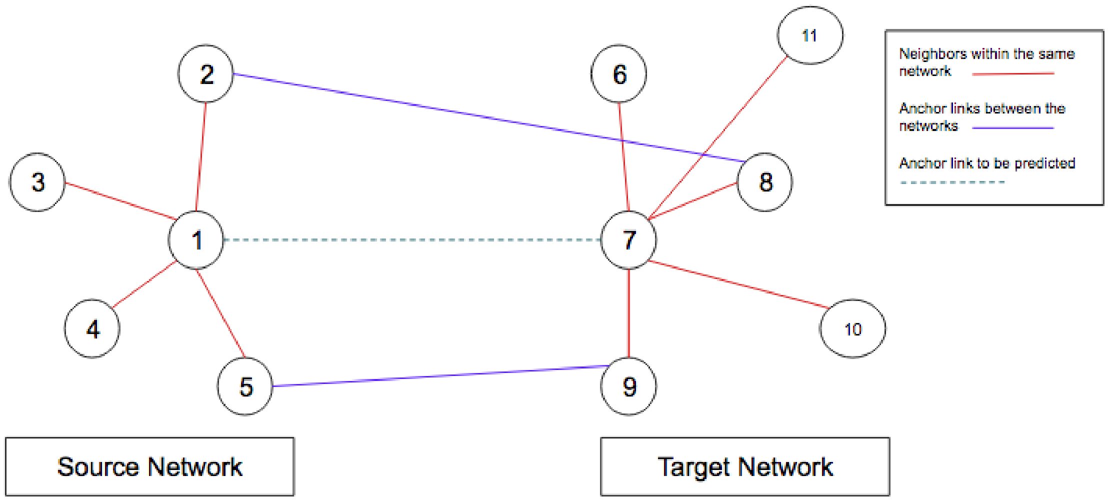
Given two heterogeneous social networks, source network $G_s = (V_s, E_s)$ and target network $G_t = (V_t, E_t)$ where V_s, V_t are users/nodes and E_s, E_t are friendships/edges of G_s and G_t respectively. A small proportion of anchor links $A = \{(u_i^s, v_j^t), u_i^s \in V_s, v_j^t \in V_t\}$ are given. The Network Alignment problem is to predict if a pair of users u_i^s and v_j^t where $u_i^s \in V_s$ and $v_j^t \in V_t$ have an anchor link between them. There is a one-one matching between the user accounts. The solution to the problem can be divided into two phases - Feature extraction and Anchor link prediction. This problem is similar to the link prediction task where we predict if there is a possibility of link formation between two users of a network based on the information from the existing edges. The difference in the Network Alignment problem is that we predict links between users of two different networks instead of a single network. In this project, the solution assumes presence of some already known anchor links between the source and target network.

3.1.1 Feature Extraction

As we will be using topological features, we choose the commonly used measures for link prediction such as Common Neighbors, Jaccard Coefficient, Adamic-Adar measure. For the anchor link prediction task, these features have to be adapted to a two network setting. Apart from the topology based features, we use two other features namely the Username Embedding Similarity and the Profile Image Embedding Similarity.

- **Extended Common Neighbors** In a single network, the common neighbors are the number of neighbors/friends shared by a pair of users. This measure has to be adapted for the Network Alignment task and hence the name Extended Common Neighbors. Extended Common Neighbors between a pair of users from two different networks is the number of neighbors/pairs of neighbors who already have anchor links between them [2]. If $\Gamma(u_i^s)$ and $\Gamma(v_j^t)$ represent the set of neighbors of u_i^s and v_j^t respectively, then Extended Common Neighbors is defined by the equation

$$ECN(u_i^s, v_j^t) = |(u_m^s, v_n^t) \in A, u_m^s \in \Gamma(u_i^s), v_n^t \in \Gamma(v_j^t)|$$



2
6
11
3
1
7
8
4
5
9
10
Neighbors within the same network
Anchor links between the networks
Anchor link to be predicted
Source Network
Target Network

Figure 2: Illustration - Common Neighbors

Consider the network in Figure 2, there are four neighbors of $user1$ in source network and there are five neighbors of $user7$ in target network. Out of these neighbors, there are four anchor users with two anchor links between them. As these four users are essentially two unique users, the number of extended common neighbors between $user1$ and $user7$ is measured as 2.

- **Extended Jaccard Coefficient** - Jaccard Coefficient in terms of a single network is the normalized form of common neighbors. When we calculate the Jaccard Coefficient between users of two networks, we divide the extended common neighbors by distinct neighbors of the user pair. When distinct neighbors are considered we have to leave out the anchor users [2]. Jaccard Coefficient is defined by the equation.
  $JC(u_i^s, v_j^t) = \frac{|ECN(u_i^s, v_j^t)|}{|\Gamma(u_i^s) \cup \Gamma(v_j^t)|}$

  where $|\Gamma(u_i^s) \cup \Gamma(v_j^t)| = |\Gamma(u_i^s)| + |\Gamma(v_j^t)| - |\Gamma(u_i^s) \cap \Gamma(v_j^t)|$

  From Figure 2, the Extended Jaccard Coefficient of $user1$ from source network and $user7$ from target network is obtained by dividing the extended common neighbors by distinct neighbors of the user pair. In this case the distinct neighbors would be $user3$, $user4$, $user6$, $user11$, $user10$. Hence the Extended Jaccard Coefficient will be calculated as $2/5$ which is 0.2.

- **Extended Adamic/Adar Measure** - This measure considers the average degree of the common neighbors in the network. We calculate this measure in the two network setting as the average degree of the extended common neighbors and taking the inverse log of it [2]. Adamic/Adar measure can be defined by the equation
  $AA(u_i^s, u_j^t) = \sum \log^{-1} \frac{|\Gamma(u_m^s)|+|\Gamma(v_n^t)|}{2}$
  where the measure is calculated over every pair $(u_m^s, v_n^t) \in |\Gamma(u_i^s) \cap \Gamma(u_j^t)|$

- **Username Embedding Similarity** - This feature measures the closeness of username embedding of the user pair under consideration. In a heterogeneous social network, information like username, profile image can be treated as separate entities called objects/ information nodes [1]. These objects are associated with users through predicates. For example, {user_123, has_name, Andrew}. This representation is called factoid embedding of user and object. For a predicate like has_name, let $\mathsf{O}_{\mathsf{has\_name}} = \{\mathsf{Andrew}, \mathsf{Yang}, \mathsf{Martha}, \mathsf{Andrew}_{\mathsf{g}}, \mathsf{Yang}_{\mathsf{l}}, \mathsf{MarthaS....}\}$ be a set of objects. This set has users from both the networks. A username embedding is generated as a vector containing similarity of a given username to every other username object in both source and target network. The embedding vectors of users with similar names will be close to each other in the embedding space. Jaro-Winkler distance is used to measure the similarity of each username to other username objects. Jaro-Winkler distance is similar to measuring edit distance between two sequences. Xie, et al in their work on Network Alignment through factoid embeddings generate these username embeddings [1]. In this project, we use the dataset created by Xie, et al and hence we didn't calculate the username embeddings. We directly use the username embeddings to calculate the cosine distance between these embedding vectors and use it as a feature in our model. If two users have similar embedding vectors, the cosine distance between them should be low which provides an indication of similarity. Cosine similarity between two vectors x and y is defined as

$CosineSimilarity = \frac{\sum_{i=1,n} x_i y_i}{\sqrt{\sum_{i=1,n} x_i^2}\sqrt{\sum_{i=1,n} y_i^2}}$

$CosineDistance = 1 - CosineSimilarity$

- **Profile Image Embedding Similarity** - A user may have similar profile pictures on two different social networks. Extracting the facial features of a person's image can help in finding similarity between users. A Deep Learning framework VGG16 is used to extract features from profile images of users. The VGG16 is used with pre-trained weights on ImageNet [1]. The VGG16 emits a feature vector for each profile image. This is used as the profile image embedding vector. These embeddings are already present in the dataset. We use the cosine distance to measure the distance between the profile image embedding vectors. This distance is an indicator of user similarity too.

Using Page Rank as a feature Related work in the field of Network Alignment such as S.Zhang, et al [9], Y.Zhang, et al [13] suggest that page rank can be used to measure similarity between users on two different networks. The basis for the argument here is that if a user has a high page rank(user is a leader) in one network, he/she should have a similar page rank in another network too. This may not be true always. For example, there might be a user who is very active on Facebook but not very active on Twitter. We experimented with the page rank values by extracting page rank values for each user in both the networks. We labelled the top 1% of users as opinion leaders, next 10% as medium users and remaining users as others [9] [13]. For each user pair we measured the number of neighbors who are anchor users and fall in the same category as the user in the user pair who is being evaluated. For example, based on the page rank score, if u_i^s from the source network is categorized as opinion leader, we look for the neighbors of u_i^s who are anchor users and characterized as opinion leaders too. The same is done for v_j^t in the target network. This measure didn't prove to be beneficial for our results. From our experiments, page rank didn't prove to be a good measure of user similarity.

3.1.2 Algorithms for the Network Alignment problem

The integral part of the whole methodology is the method we use to extract the features and then use them in supervised learning models in order to predict anchor links between user pairs. Algorithms use procedures like getFeatures, compareUsernameEmbeddings, and compareImageEmbeddings to fetch the features required for the training. Below the algorithms for extracting individual features followed by construction of the dataset is described.

3.1.2.1 Algorithm 1 - Network Alignment - Using topology based features.

In the first step of the Procedure NETWORK_ALIGN_TOPOL of Algorithm 1 as we go through every known anchor link pair we create a dictionary that stores a user from the source network G_s as key and the corresponding anchor user from the target network G_t as value, i.e., $\text{ground\_truth}[\text{source\_user}] = \text{target\_user}$. In the next step we construct a list, source_anchors of users who are anchor users from G_s. Similarly construct a list, target_anchors of anchor users from G_t. After that, call the GETTOPOLOGYFEATURES procedure to extract the required topology based features. The extracted features are used to construct the training and test set. A training and test sample of a user pair (u_i^s, v_j^t) where $u_i^s \in G_s$ and $v_j^t \in G_t$ is labelled as positive(1) if the user pair is present in ground truth file otherwise it is labelled negative(0).

Algorithm 1 Network Alignment with Topology Based Features

```
1:  procedure GETTOPOLOGYFEATURES(source_user, target_user)
2:      extended_common_neighbor $\leftarrow 0$
3:      extended_jaccard_coefficient $\leftarrow 0$
4:      extended_adamic_adar $\leftarrow 0$
5:      for each edge $e = (v^s, u^s) \in G_s$, where $v^s$ = source_user do
6:          neighbor $\leftarrow u^s$
7:          if neighbor in source_anchors then
8:              Add neighbor into possible_src_neighbors
9:          else
10:             Add neighbor into non_anchor_users
11:         end if
12:     end for
13:     Repeat steps of lines $5 - 12$ for the target_user using target_anchors and add the
    possible common neighbors to possible_target_neighbors
14:     for each source_anchor in possible_src_neighbors do
15:         if ground_truth[source_anchor] is present in possible_target_neighbors then
16:             Increment extended_common_neighbor
17:             extended_adamic_adar = extended_adamic_adar +
    (degree(source_anchor) + degree(ground_truth[source_anchor]))/2
18:         end if
19:     end for
20:     extended_jaccard_coefficient = (extended_common_neighbor)/(non_anchor_users)
21:     return extended_common_neighbor, extended_jaccard_coefficient, extended_adamic_adar
22: end procedure
23:
24: procedure NETWORK_ALIGN_TOPOL($G_s$, $G_t$)
25:     for (source_user, target_user) in ground_truth file do
26:         ground_truth[source_user] = target_user
27:         add source_user into source_anchors
28:         add target_user into target_anchors
29:     end for
30:     for each each $v \in G_s$ and each $u \in G_t$ of the training set do
31:         call Procedure GETTOPOLOGYFEATURES($v, u$)
32:     end for
33: end procedure
```

3.1.2.2 Algorithm 2 - Network Alignment - Using topology based features, username and profile image embedding

This algorithm outlines the process of Network Alignment using embedding similarity as features. It uses the topology based features from Algortihm 1 as well. In the first step of the Procedure Network_Align_Topo_Emb we create the ground_truth dictionary similar to Algorithm 1. In the next step we create more dictionaries username_source_embedding, username_target_embedding, image_source_embedding, image_target_embedding. username_source_embedding contains users from G_s as key and corresponding username embedding as value. i.e. username_source_embedding[source_user] = source_user_name_embedding. Similarly, image_source_embedding contains users from G_s as key and corresponding image embedding as value. i.e. image_source_embedding[source_user] = source_user_image_embedding. We fill the username_target_embedding and image_target_embedding dictionary in the same way for users in G_t

In the next step, for every pair u_i^s, v_j^t in the training set, we call the procedure GET-TOPOLOGYFEATURES to get the topology based features. Next we call the procedure compareUsernameEmbeddings and procedure compareProfileimageEmbeddings to get username and profile image embedding similarity. In the next step training and test sample of every pair of users (u_i^s, v_j^t) where $u_i^s \in G_s$ and $v_j^t \in G_t$, are labelled positive(1) if the user pair is found in the ground truth file otherwise labelled negative(0).

3.1.2.3 Algorithm 3 - Unsupervised Network Alignment - Using topology based features independently

Procedure NETWORK_ALIGN_UNSUP first creates the dictionary ground_truth as described in Algorithm 1 for the user pairs in training set along with the source_anchors

Algorithm 2 Network Alignment with Topology Based Features + User name Embedding and Profile Image Embedding

```
procedure compareUsernameEmbeddings($u_i^s$,$v_j^t$)
    Fetch the embedding of $u_i^s$ from username_source_embedding
    Fetch the embedding of $v_j^t$ from username_target_embedding
    Compare the embeddings using scipy.spatial.distance.cosine method
    return the cosinedistance
end procedure

procedure compareProfileimageEmbeddings($u_i^s$,$v_j^t$)
    Fetch the embedding of $u_i^s$ from image_source_embedding
    Fetch the embedding of $v_j^t$ from image_target_embedding
    Compare the embeddings using scipy.spatial.distance.cosine method
    return the cosinedistance
end procedure

Procedure Network_Align_Topo_Emb($G_s$,$G_t$)
    Initialize ground_truth_dict, username_source_embedding, username_target_embedding,
image_source_embedding, image_target_embedding to empty dictionaries/hashmap
    for each anchor_link in the ground_truth_file do
        Repeat Step 27-30 from Algorithm 1
    end for
    for each user in source_anchors do
        fetch the username embedding of the user and add user as key and username
embedding as value into username_source_embedding
        fetch the profile image embedding of the user and add user as key and image
embedding as value into image_source_embedding
    end for
    Repeat step 22 - 25 for every user in target_anchors
    for each user pair in training set do
        call Procedure (getTopologyFeatures)
        call Procedure (compareUsernameEmbeddings)
        call Procedure (compareProfileimageEmbeddings)
    end for
end Procedure
```

and target_anchors list. Three other lists scores_ecn, scores_jaccard, scores_adam are also initialized.For every user pair (u_i^s, v_j^t) where $u_i^s \in G_s$ and $v_j^t \in G_t$,in the test set which are treated as missing links, the following measures are calculated - Extended Common

Neighbors, Extended Jaccard Coefficient and Extended Adamic/Adar measure. The scores obtained from each of the methods are added to their respective lists. i.e Extended Common Neighbors are added to source_ecn, Extended Jaccard Coefficient is added to source$_j$accard and Extended Adamic/Adar measure is added to source_adam. In the next step, the scores are then sorted in descending order along with the predictions. The true and false positives are calculated at various thresholds and an AUC_Score for every method is calculated using sklearn.metrics.auc.

Algorithm 3 Unsupervised Network Alignment with Topology Based Features one at a time

```
1: procedure GETAUCSCORE($scores$,$targets$)
2:     Use the sklearn.metrics.auc method to calculate the AUC scores.
3: end procedure
4:
5: Procedure NETWORK_ALIGN_UNSUP($missing\_links$)
6:     Initialize scores_ecn, scores_jaccard, scores_adam to empty arrays
7:     for each anchor_link in training set do
8:         Repeat Step 27-30 from Algorithm 1
9:     end for
10:    for each user_pair in missing_links do
11:        call 1 (getTopologyFeatures)
12:        call Add the features to scores_ecn, scores_jaccard, scores_adam respectively.
13:    end for
14:    Sort the scores in scores_ecn, scores_jaccard, scores_adam along with the predic-
       tions in descending order
15:    Call 1 (getAUCScore) with scores_ecn and sorted predictions
16:    Repeat the step 15 for scores_jaccard, scores_adam
17:    return AUC_Score for each method.
18: end Procedure
```

CHAPTER 4

Experimental Evaluation

4.1 Datasets

For our experiments, we are using the User Identity Linkage dataset used in [1]. It contains information on two pairs of social networks: Foursquare-Twitter and Facebook-Twitter. Table 1 describes the number of users and the number of edges in each sub-network.

Table 1: Dataset Overview

Dataset	Foursquare-Twitter		Facebook-Twitter	
Network	Foursquare	Twitter	Facebook	Twitter
Users	21668	25772	17359	20024
Links	312740	405590	224762	165406

The dataset also comes with a set of known anchor links.

Foursquare - Twitter has 3602 known anchor links.

Facebook - Twitter has 1998 known anchor links.

Apart from this topological information, the username and profile image embedding vectors for users are available.

The experiments involves three phases- data preparation, training using supervised learning and unsupervised learning and comparing the supervised and unsupervised approaches to network alignment. Some of the unsupervised methods are Extended Jaccard coefficient, Extended Common Neighbors, Extended Adamic-Adar measure. We intend to use the scores obtained from the unsupervised methods as features for supervised learning. The output of the supervised learning model is binary where the output of 0 indicates that anchor link cannot exist between the user pair and an output of 1 indicates the presence of anchor link between the user pair.

4.2 Data Preparation

For anchor link prediction, we have to label the data so that it can be used for supervised learning. To prepare the training and test data, we use the ground truth file. Every user pair that appears in the ground truth file are labelled as positive. For negative training samples, a user from the source network is paired with any other user from the target network apart from the one with which it is paired in the ground truth file. We can mark this pair as negative as we definitely know that there can be no anchor link between this pair as they have been paired with their respective anchor pair in the ground truth file. For example, consider a user pair `foursquare_user_14628` and `twitter_user_15915` in the ground truth file. The training sample with this pair is labelled as positive(1) as we definitely know they are anchor users. We pair `foursquare_user_14628` with any other twitter user for example `twitter_user_2841` and label the corresponding sample as negative(0) as we definitely know there is no anchor link between them. We can be sure of that due to the fact that anchor links have a one-one constraint. This implies that a given user pair can belong to only one user.

Table 2: Supervised Learning Training Sample.

User-Pair	ECN	JC	Adam	Usr_Emb_Sim	Img_Emb_Sim	label
$fq14628\#tw15915$	6	0.090909	-33.7733	0.9016	1	1
$fq14628\#tw2841$	0	0.0	0	1	0.916	0

Table 2 shows a positive and negative training sample. Each column other than the User-Pair is a feature. The ground truth user pairs are divided into two separate folders called training and testing to ensure that the data used for training is never used for prediction. For the user pair in the test set, the features are constructed in the same way but the label field is stripped off during prediction. We experimented with varied number of training samples. We used two approaches to create the training samples. One approach creates the positive and negative samples in a balanced manner. Another approach creates more negative samples and then we use Upsampling to balance out the data.

4.2.1 Upsampling and Downsampling

To increase the number of training samples, for every positive labelled training data sample, we created 20 negative labelled training data. This causes the training data to be skewed towards the negative side [12]. To balance out the data we use the technique of upsampling to increase the positive samples by randomly duplicating the positive samples until we balance the positive and negative samples for training. We can also perform the downsampling of the negative samples but this will lead to data loss also the training samples decrease in number.

4.2.2 No Upsampling

For every positive labelled training data sample, we created one negative labelled training data. This will create a balanced training set with equal number of positive and negative samples. With this approach we have lesser but balanced training set.

4.3 Training

We used only the topology based features for unsupervised learning. The same features were combined for supervised learning. The experiments were conducted with training samples that were unbalanced which were later balanced out through upsampling as well as a balanced training set.

4.3.1 Unsupervised Learning Methods

Link prediction tasks lack labelled data. In such scenarios, we can use the unsupervised approach to predict links. Jaccard coefficient, Common Neighbors, Adamic/Adar scores are some of the generally used unsupervised link prediction methods on a single network. These measures have been appropriately modified to suit the link prediction on a two network setting as described in section 3.1.1 and [2]. They give out scores purely based on topology. We measure the performance of each of these measures separately.

To perform the training, the user pairs in the test set are treated as missing links. The user pairs in the training set are treated as existing anchor links of the network and the Extended common neighbors, Jaccard coefficient, Adamic/adar measure is calculated for the missing anchor links which is equivalent to predicting anchor links between user pairs in the test set . The scores are then sorted in descending order. A certain threshold is set for the scores on the basis of which true positives and false positives are calculated. For example, we set a threshold of 20 for extended common neighbors. The following rules are followed to calculate true positives, false negatives and false positives.

- Any user pair that has extended common neighbors greater than 20 and appears in the ground truth links, we classify it as true positive.
- Any user pair which is not in the ground truth links but has extended common neighbors greater than 20 is treated as false positive.
- Any user pair which is not in the ground truth links and has extended common neighbors lesser than 20 is treated as true negative.
- Any user pair which is in the ground truth links and has extended common neighbors lesser than 20 is treated as false negative.

We use the AUC (Area Under the ROC Curve) score to measure the accuracy. The AUC score for a single threshold point is calculated as

$\mathsf{AUC} = (\mathsf{truepositives} + (0.5) \cdot \mathsf{falsepositives})/\mathsf{total\_sample}$

We plot a curve with the sorted scores and the targets/predictions. The scores are sorted in a descending order. To calculate the AUC score of the method, the true positive, false positives have to be considered at different thresholds. The sklearn.metrics.auc calculates the false positive rate and true positive rates at different thresholds and gives out an AUC Score for the prediction method.

4.3.2 Supervised Learning Methods:

We experiment with two models. One with only topology based features and another with additional features username embedding similarity and profile image embedding similarity. The details of these features are discussed in section 3.1.1 and [1]. We experiment with Logistic Regression, K Nearest Neighbors(KNN) and a simple neural network as training methods. We experiment with two configurations of the KNN and the neural network. Both the models are trained on both upsampled and un-sampled data.

- Logistic Regression - This is one of the most commonly used method for link prediction. The model trains on the data in the training set and emits a 0 or 1 during prediction on the test set. The accuracy of the predictions is calculated through comparing the predictions against the labels on the test set. The sklearn.metrics module is used to calculate the accuracy.

- K Nearest Neighbors - This is a simple machine learning algorithm used for classification and regression. Every point is classified based on the K-nearest neighbors. For example, when k = 3 and if two nearest samples are labelled positive and another sample is negative, the current sample is classified as positive. We experimented with two configuration, K = 3 and K = 5. We have only used odd numbers for K as it can break ties during classification.

- Neural Network - The number of neurons in the input layer depends on the number of features. If only topological features are used, then the input layer will have 3 neurons. If username and profile image embedding similarity is used, the input layer will have 5 neurons. We experimented with two types of neural networks. The first one has a single dense layer of 7 neurons between the input and the output layer. The second one has two dense layers with 7 and 5 neurons respectively. Sigmoid and Relu activation functions between the layers. On the output layer, softmax function followed by categorical crossentropy function is used to measure the loss and accuracy. Using categorical crossentropy classifies a given training sample to close to 0 or close to 1. 100 epochs or iterations were used to train the model.

CHAPTER 5

Results

5.1 Experimental Setup

Python is used as the implementation language to fetch features. Python libraries such as pandas, numpy sklearn are used to preprocess data and train the models. Keras library is used for the neural network model. All the experiments were conducted on a MacBook with OS X version 10.11.6, 4GB internal memory.

5.2 Results

We have performed various network alignment experiments on the two pairs of social networks Foursquare - Twitter and Facebook - Twitter. The results show the performance of various models with and without upsampling of the training set.

Table 3: Number Of Training And Testing Samples Without Upsampling.

	Foursquare - Twitter	Facebook - Twitter
Training	5763	3197
Testing	1439	798

Table 3 shows the number of training and test samples used to train and test the models without performing any upsampling on the training set to balance the positive and negative samples in the training set.

Table 4: Number Of Training And Testing Samples With Upsampling.

	Foursquare - Twitter	Facebook - Twitter
Training	54568	30191
Testing	1439	798

Table 4 shows the number of training and test samples used to train and test the models after performing upsampling on the training set to balance the positive and negative samples in the training set. This will lead to increase in positive samples to balance the negative samples.

5.2.1 Results - Algorithm 1 -(Models - Topological Features)

Table 5 shows the anchor link prediction accuracy of the model that uses only three topology based / social link based features - Extended common neighbors, Extended Jaccard Coefficient and Extended Adamic/Adar measure. No upsampling/downsampling was performed on the training set. As seen from the Table 5, for the Foursquare-Twitter dataset, the K-Nearest-Neighbors(KNN) with K=5 did the best. The same KNN did not perform very well with the Facebook-Twitter dataset. The neural network with a single dense layer gave the best accuracy for the Facebook-Twitter dataset.

Table 5: Prediction Accuracy With Topological Features And No Upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	0.8700	0.71016
K-Nearest-Neighbors(K=3)	0.8721	0.5006
K-Nearest-Neighbors(K=5)	**0.8769**	0.5006
Neural Network(1 Dense Layer)	0.8601	**0.7163**
Neural Network(2 Dense Layers)	0.8601	0.7160

Figures 3,4,5,6,7 and 8 show the results in the form of ROC curves for the models and results discussed above. Figure 7 clearly shows that the KNN model couldn't do a good job at prediction on the Facebook - Twitter dataset and hence the curve is almost superimposed on the red line. The KNN model with K=3 and 5 had similar performance on Foursquare - Twitter dataset and hence the blue(K=3) and green(K=5) line are superimposing on each other.

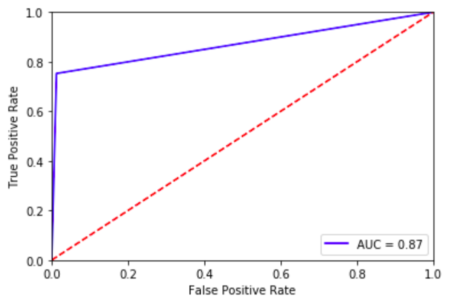


Figure 3: ROC Curve Logistic Regression, Topological Features, No Upsampling [Foursquare-Twitter]

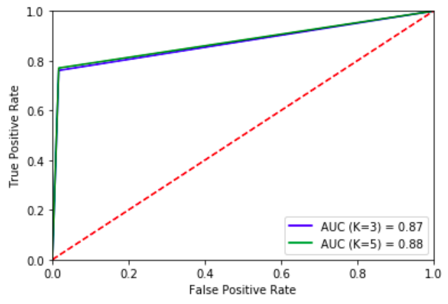


Figure 4: ROC Curve K-Nearest-Neighbors, Topological Features, No Upsampling [Foursquare-Twitter]

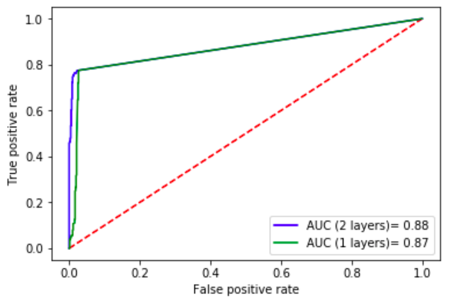


Figure 5: ROC Curve Neural Network, Topological Features, No Upsampling [Foursquare-Twitter]

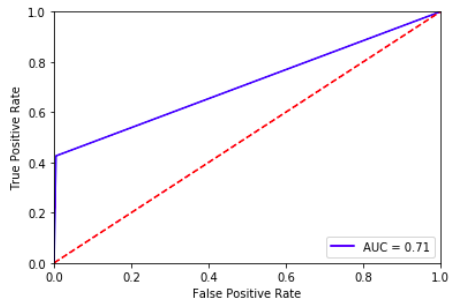


Figure 6: ROC Curve Logistic Regression, Topological Features, No Upsampling [Facebook-Twitter]

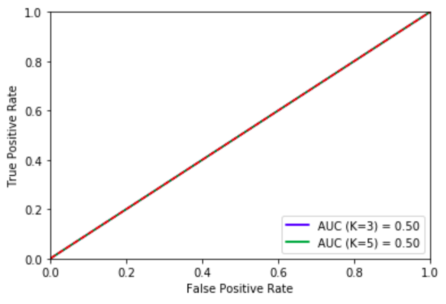


Figure 7: ROC Curve K-Nearest-Neighbors, Topological Features, No Upsampling [Facebook-Twitter]

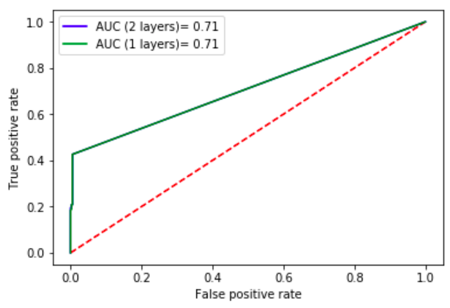


Figure 8: ROC Curve Neural Network, Topological Features, No Upsampling [Facebook-Twitter]

Table 6 shows the anchor link prediction accuracy of the model that uses the three topology based / social link based features along with upsampling on the training set to balance out the positive and negative samples. As seen from table 6, Logistic Regression gave the best results when the upsampled data was used for Foursquare-Twitter dataset. The KNN classifier performed poorly on both the datasets. Neural network did the best for the Facebook-Twitter dataset.

Table 6: Prediction Accuracy With Topological Features And Upsampling .

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	**0.8756**	0.7101
K-Nearest-Neighbors(K=3)	0.4683	0.4943
K-Nearest-Neighbors(K=5)	0.4808	0.4981
Neural Network(1 Dense Layer)	0.8582	**0.7164**
Neural Network(2 Dense Layers)	0.8582	**0.7164**

Figure 9, 10, 11, 12, 13, 14 show the ROC curves for the performance of the models with upsampled data. The curves for KNN classifiers show that it didn't perform well with the upsampled data. The AUC scores for the KNN classifier is in the range of 0.47-0.50 which indicates that it could not predict the links at all and hence almost superimposed on line for AUC score of 0.5 (red line). The logistic regression and neural network curves show that they did better for the Foursquare-Twitter, Facebook - Twitter datasets respectively.

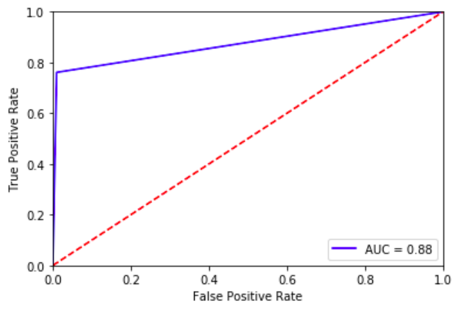


Figure 9: ROC Curve Logistic Regression, Topological Features, With Upsampling [Foursquare-Twitter]

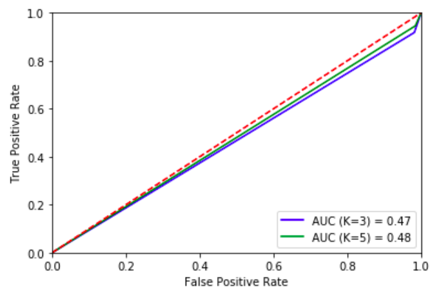


Figure 10: ROC Curve K-Nearest-Neighbors, Topological Features, With Upsampling [Foursquare-Twitter]

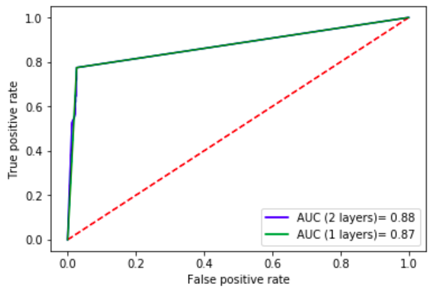


Figure 11: ROC Curve Neural Network, Topological Features, With Upsampling [Foursquare-Twitter]

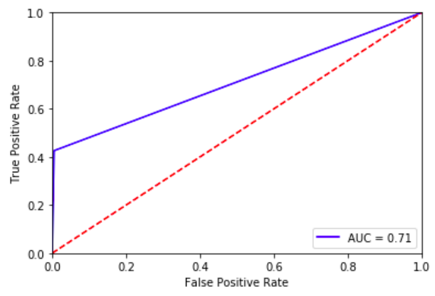


Figure 12: ROC Curve Logistic Regression, Topological Features, With Upsampling [Facebook-Twitter]

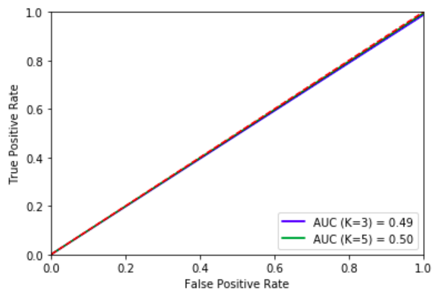


Figure 13: ROC Curve K-Nearest-Neighbors, Topological Features, With Upsampling [Facebook-Twitter]

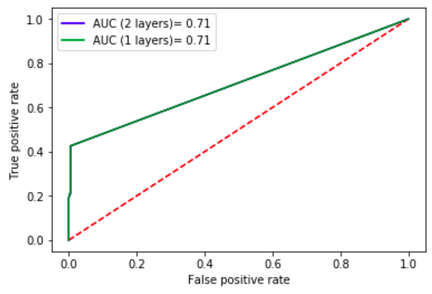


Figure 14: ROC Curve Neural Network, Topological Features, With Upsampling [Facebook-Twitter]

5.2.2 Results - Algorithm 2 - (Model - Topological Features + Username and Profile Image Embedding)

Table 7 shows the anchor link prediction accuracy of the model that uses the three topology based / social link based features along with username, image embedding similarity as features. No upsampling is performed on the training set . As seen from Table 7, KNN classifier with K = 5 gave the best results on Foursquare-Twitter dataset. Neural network with two layers did the best for the Facebook-Twitter dataset.

Table 7: Prediction Accuracy With Topological Features, Username And Image Embedding, No Upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	0.9478	0.8883
K-Nearest-Neighbors(K=3)	0.9444	0.8808
K-Nearest-Neighbors(K=5)	**0.9513**	0.8858
Neural Network(1 Dense Layer)	0.9375	0.8827
Neural Network(2 Dense Layers)	0.9420	**0.8962**

Figures 15, 16, 17, 18, 19, 20 show the ROC curves for the models and the results discussed above. As seen from the curves, and the accuracy from Table 7, the results are better when username and profile image embedding similarity are used as features. All the models gave considerable predictions when compared to the models with only topology based features.

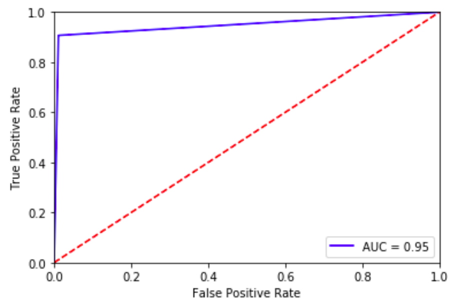


Figure 15: ROC Curve Logistic Regression, Topological Features + Embedding Similarity, No Upsampling [Foursquare-Twitter]

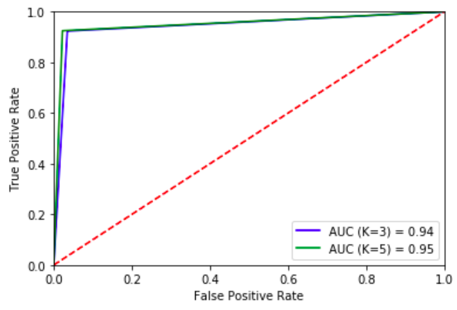


Figure 16: ROC Curve K-Nearest-Neighbors, Topological Features + Embedding Similarity, No Upsampling [Foursquare-Twitter]

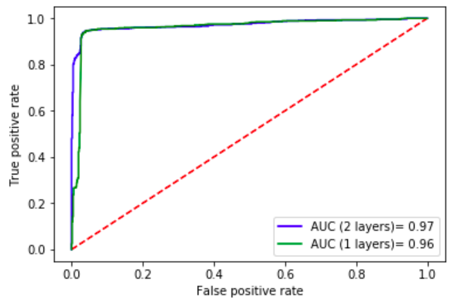


Figure 17: ROC Curve Neural Network, Topological Features + Embedding Similarity, No Upsampling [Foursquare-Twitter]

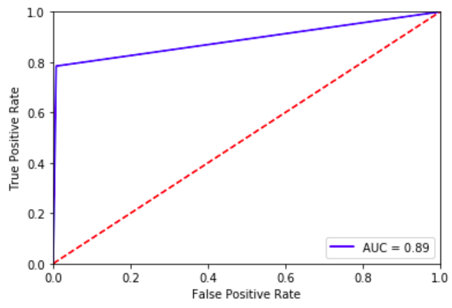


Figure 18: ROC Curve Logistic Regression, Topological Features + Embedding Similarity, No Upsampling [Facebook-Twitter]

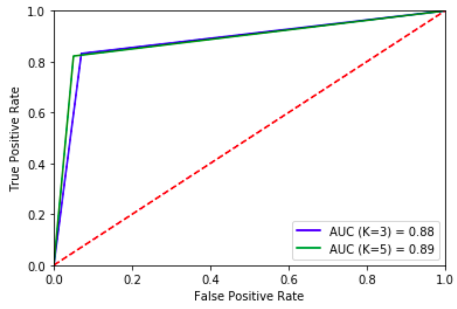


Figure 19: ROC Curve K-Nearest-Neighbors, Topological Features + Embedding Similarity, No Upsampling [Facebook-Twitter]

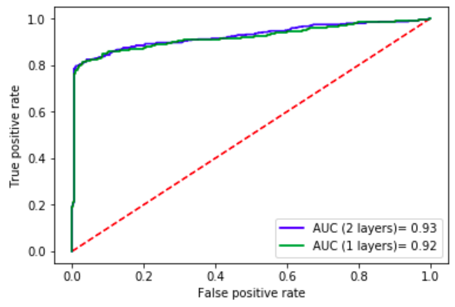


Figure 20: ROC Curve Neural Network,Topological Features + Embedding Similarity, No Upsampling [Facebook-Twitter]

Table 8 shows the anchor link prediction accuracy of the model that uses the three topology based / social link based features along with username,image embedding similarity as features.Upsampling is performed on the training set to balance the positive and negative samples . As seen from Table 8, Logistic Regression gave the best results on Foursquare-Twitter dataset. Neural network with two layers did the best for the Facebook-Twitter dataset. Compared to the unsampled data, the upsampled data gave slightly higher results.

Table 8: Prediction Accuracy With Topological Features, Username And Image Embedding, Upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	**0.9513**	0.8895
K-Nearest-Neighbors(K=3)	0.9263	0.8695
K-Nearest-Neighbors(K=5)	0.9284	0.8732
Neural Network(1 Dense Layer)	0.9479	0.8981
Neural Network(2 Dense Layers)	0.9481	**0.8986**

Figures 21, 22, 23, 24, 25,26 show the ROC curves for the models and results discussed above. As seen from the curves and results in table 8, the performance of the models are not greatly affected by upsampling of the data during training. All the models were able to give fair predictions for the data.

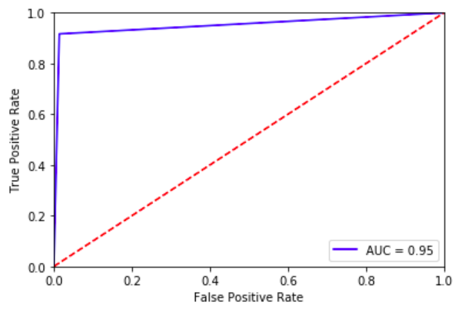


Figure 21: ROC Curve Logistic Regression, Topological Features + Embedding Similarity, With Upsampling [Foursquare-Twitter]

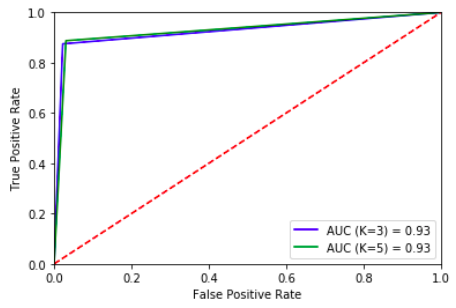


Figure 22: ROC Curve K-Nearest-Neighbors, Topological Features + Embedding Similarity, With Upsampling [Foursquare-Twitter]

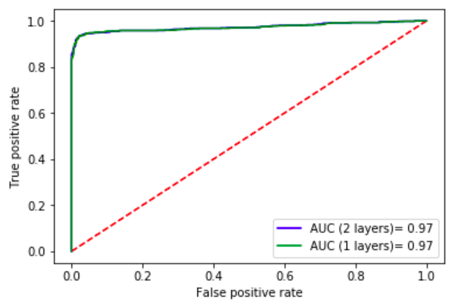


Figure 23: ROC Curve Neural Network, Topological Features + Embedding Similarity, With Upsampling [Foursquare-Twitter]

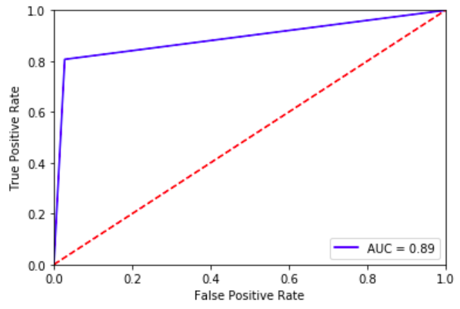


Figure 24: ROC Curve Logistic Regression, Topological Features + Embedding Similarity, With Upsampling [Facebook-Twitter]

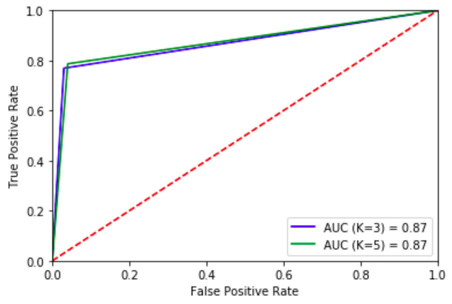


Figure 25: ROC Curve K-Nearest-Neighbors, Topological Features + Embedding Similarity, With Upsampling [Facebook-Twitter]

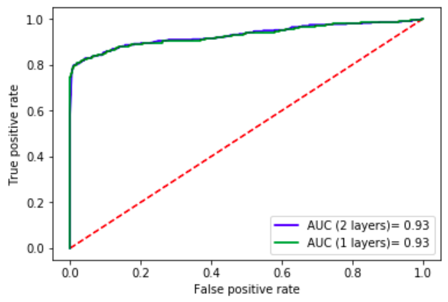


Figure 26: ROC Curve Neural Network,Topological Features + Embedding Similarity, With Upsampling [Facebook-Twitter]

5.2.3 Results - Unsupervised Model

The results in Table 9 show that when Extended Common Neighbors, Extended Jaccard Coefficient, Extended Adamic/Adar measure are used as independent methods to solve the network alignment problem, the AUC score is lower compared to the supervised setting when they are combined together. The AUC Scores for Extended Common Neighbors(ECN) and Extended Jaccard Coefficient (EJC) is similar as EJC is the normalized form of ECN. When the AUC score is calculated over a range of thresholds, the scores for both the methods end up being similar. Another important observation here is that the Extended Adamic/Adar measure performed poorly when used as independent approach to the network alignment problem.

Table 9: AUC Scores Of Unsupervised Methods For Network Alignment.

Methods	Foursquare - Twitter	Facebook - Twitter
Extended Common Neighbors	0.85	0.68
Extended Jaccard coefficient	0.85	0.68
Adamic/Adar Measure	0.15	0.32

Figures 27, 28, 29, 30, 31, 32 show the performance of the unsupervised methods when used for network alignment independently. The ROC curve suggest that ECN and EJC performed in a similar way for network alignment on both the datasets. The ROC curve for Extended Adamic/Adar measure shows that it performed poorly when used independently.

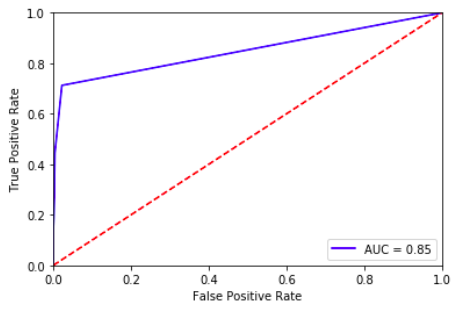


Figure 27: ROC Curve - Extended Common Neighbors [Foursquare-Twitter]

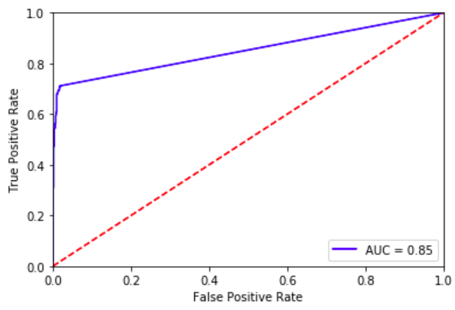


Figure 28: ROC Curve - Extended Jaccard coefficient [Foursquare-Twitter]

5.2.4 A Comparison Between Supervised And Unsupervised Approaches To Network Alignment

For Foursquare - Twitter network when these unsupervised features were combined for training, the highest AUC score of 0.88 was obtained with the neural network model and the KNN model. Among the unsupervised methods, the higest AUC score we obtained was 0.85 with ECN and EJN. For Facebook - Twitter network, the highest AUC score of

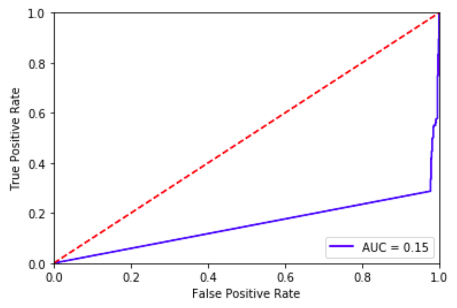


Figure 29: ROC Curve - Extended Adamic/Adar Measure [Foursquare-Twitter]

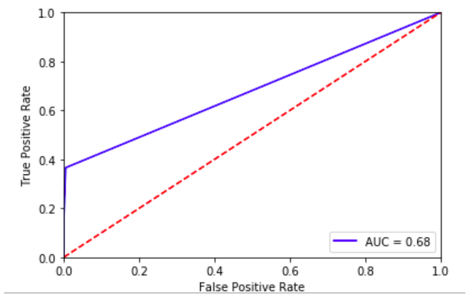


Figure 30: ROC Curve - Extended Common Neighbors [Facebook-Twitter]

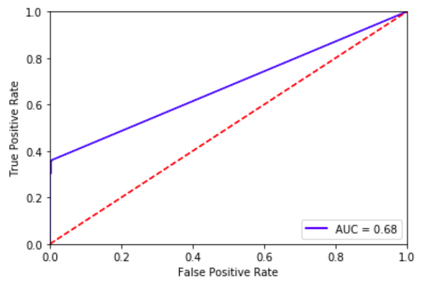


Figure 31: ROC Curve - Extended Jaccard Coefficient [Facebook-Twitter]

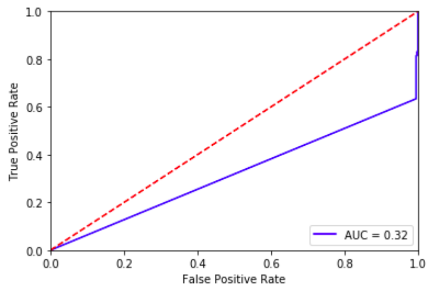


Figure 32: ROC Curve - Extended Adamic/Adar Measure [Facebook-Twitter]

0.71 with Logistic Regression and Neural Network model whereas the highest AUC score obtained with unsupervised methods was 0.68.

Another unsupervised approach to network alignment that we know is used in the paper by W.Xie, et al [1] through the use of factoid embedding. W.Xie, et al tested their approach in the unsupervised setting and a semi supervised setting. They measure the performance using a metric called HitRate@K($HR@K$) where a ranking of matching users is considered to be correct if the correct matched user (v_t^j) from the target network appears in the top K matched users [1]. In our proposed solution we do not have multiple matchings. So, we compare our results with $HR@1$ which would be similar to evaluating if the user appearing in the Top 1 matched users is indeed the correct anchor user from the target network. Table 10 shows the comparison of our approach with the factoid embedding approach used in [1].

Table 10: Comparison With Factoid Embedding Approach(FE).

Methods	Foursquare - Twitter	Facebook - Twitter
FE $HR@1$ Semi-Supervised	0.5541	0.6851
FE $HR@1$ Unsupervised	0.5433	0.6781
Network Alignment Unsupervised	0.85	0.68
Network Alignment Supervised	0.88	0.71

The $HR@1$ score of both Semi-Supervised and Unsupervised approach of factoid embedding approach on the Foursquare-Twitter dataset is far below the predictions we could achieve though our approach. the $HR@1$ score on the Facebook-Twitter dataset is comparable to our unsupervised approach. the predictions through our supervised approach is slightly higher.

CHAPTER 6

Conclusions and Future Work

This project primarily looked at solving the network alignment problem using topology based features. From the results and related work on the same problem, it is evident that just topology based features may not be sufficient to make predictions in a supervised setting. Adding some additional attributes like username similarity and profile image similarity made a huge difference. The availability of more information entities will give better results with training. The solution proposed in the project also assumed that there are some already known anchor links. Future direction of this research is to solve the problem without any prior knowledge of anchor links. Another direction for future work would be to find ways to extract more information entities related to users like location, language (bag of words) can be used to evaluate their effect on the performance of the models.

LIST OF REFERENCES

[1] W. Xie, X. Mu, R. K. Lee, F. Zhu, and E. Lim, “Unsupervised user identity linkage via factoid embedding,” in *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, 2018, pp. 1338–1343.

[2] X. Kong, J. Zhang, and P. S. Yu, “Inferring anchor links across multiple heterogeneous social networks,” in *22nd ACM International Conference on Information and Knowledge Management, CIKM’13, San Francisco, CA, USA, October 27 - November 1, 2013*, 2013, pp. 179–188.

[3] J. Zhang, W. Shao, S. Wang, X. Kong, and P. S. Yu, “PNA: partial network alignment with generic stable matching,” in *2015 IEEE International Conference on Information Reuse and Integration*, 2015, pp. 166–173.

[4] J. Zhang and P. S. Yu, “Broad learning: : An emerging area in social network analysis,” *SIGKDD Explorations*, vol. 20, no. 1, pp. 24–50.

[5] D. Koutra, H. Tong, and D. Lubensky, “BIG-ALIGN: fast bipartite graph alignment,” in *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, 2013, pp. 389–398.

[6] Z. Liang, M. Xu, M. Teng, and L. Niu, “Netalign: a web-based tool for comparison of protein interaction networks,” *Bioinformatics*, vol. 22, no. 17, pp. 2175–2177, 2006.

[7] R. Singh, J. Xu, and B. Berger, “Global alignment of multiple protein interaction networks with application to functional orthology detection.” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105 35, pp. 12 763–8, 2008.

[8] M. Heimann, W. Lee, S. Pan, K. Chen, and D. Koutra, “Hashalign: Hash-based alignment of multiple graphs,” pp. 726–739, 2018.

[9] S. Zhang and H. Tong, “FINAL: fast attributed network alignment,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 1345–1354.

[10] J. Zhang and P. S. Yu, “Multiple anonymized social networks alignment,” in *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, 2015, pp. 599–608.

[11] G. Kollias, S. Mohammadi, and A. Grama, “Network similarity decomposition (NSD): A fast and scalable approach to network alignment,” *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2232–2243, 2012.

[12] T. Komamizu, Y. Hayase, T. Amagasa, and H. Kitagawa, "Exploring identical users on github and stack overflow," in *The 29th International Conference on Software Engineering and Knowledge Engineering, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 5-7, 2017.*, 2017, pp. 584–589.

[13] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "COSNET: connecting heterogeneous social networks with local and global consistency," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 2015, pp. 1485–1494.