8-28-2020

Find me if You Can: Aligning Users in Different Social Networks

Priyanka Kasbekar
San Jose State University

Katerina Potika
San Jose State University, katerina.potika@sjsu.edu

Chris Pollett
San Jose State University, chris.pollett@sjsu.edu

Recommended Citation

Priyanka Kasbekar, Katerina Potika, and Chris Pollett. "Find me if You Can: Aligning Users in Different Social Networks" *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)* (2020): 46-53. https://doi.org/10.1109/BigDataService49289.2020.00015

Find me if you can: aligning users in different social networks

1st Priyanka Kasbekar
Department of Computer Science
San Jose State University
San Jose, USA
priyanka.kasbekar@sjsu.edu

2nd Katerina Potika
Department of Computer Science
San Jose State University
San Jose, USA
katerina.potika@sjsu.edu

3rd Chris Pollett
Department of Computer Science
San Jose State University
San Jose, USA
chris@pollett.org

***Abstract*—Online Social Networks allow users to share experiences with friends and relatives, make announcements, find news and jobs, and more. Several have user bases that number in the hundred of millions and even billions. Very often, many users belong to multiple social networks at the same time under possibly different user names. Identifying a user from one social network on another social network gives information about a user's behavior on each platform, which in turn can help companies perform graph mining tasks, such as community detection and link prediction. The process of identifying or aligning users in multiple networks is called network alignment. These similar (or same) users on different networks are called anchor nodes and the edges between them are called anchor links. The network alignment problem aims at finding these anchor links. In this work we propose two supervised algorithms and one unsupervised algorithm using thresholds. All these algorithms use local structural graph features of users and some of them use additional information about the users. We present the performance of our models in various settings using experiments based on Foursquare-Twitter and Facebook-Twitter data [1]. We show that our approaches perform well even when we use the neighborhood of the users only, and the accuracy improves even more given additional information about a user, such as the username and the profile image. We further show that our best approaches perform better at the HR@1 task than unsupervised and semi-supervised factoid embedding approaches considered earlier for these datasets.**

***Keywords*-Online Social Networks, Big Data, Graph structure, Network Alignment, Anchor links, Supervised learning, Unsupervised learning, Embeddings.**

I. Introduction

The availability and ease of access to the internet have made Online Social Networks (OSNs) an integral part of our lives. OSNs are used for many purposes ranging from sharing media, reviews, news, and opinions to finding job opportunities, cabs, dates, and much more. With so many roles for OSNs, it is natural that many of them have acquired large user bases. Users are often members of multiple such social networks rather than a single social network as many of these networks have strengths and weaknesses as to the kind of sharing that can be done on them. For example, Facebook might be used to share images, Yelp might be used to review restaurants, and LinkedIn might be used to search for jobs.

Given a user (node) in one network, the problem of identifying the same user (node) or a similar user (node) in another network is called *network alignment*. This graph mining process when applied to social networks helps identify the same (physical) user on different social networks. Users that have been identified through the process of network alignment are called *anchor nodes* and the edges between them across the networks are called *anchor links* or *anchor edges*. If all of the nodes of one network are aligned with all of the nodes of another network, then the networks are said to be *fully aligned*. Fully aligned social networks are unlikely. For example, it is not mandatory for a Facebook user to be a Twitter user too. Social networks are *partially aligned* if either user has a node which is not an anchor node, and some users are unaligned.

Due to the diverse services OSNs offer, they may contain nodes other than user nodes. For example, Facebook contains locations and posts as nodes and has edges between users and users, users and a posts, users and locations, etc. These other nodes are called *information entities* [2]. A social network of only one type of node is called *homogeneous*, otherwise, the network is said to be *heterogeneous*.

The problem of network alignment in social networks has generated a lot of interest, see [3]–[7], as it helps researchers study the social behavioral patterns of the same user in different social contexts. Some applications of aligning social networks

are [2]: community detection, information sharing, viral marketing, and information diffusion.

To solve the problem of network alignment, we use various features based on the graph structure of the networks like friendships (edges), or on usernames and profile images.

In this paper, we compare supervised and heuristic approaches to network alignment using features based mostly on the structure of the network. Additionally, we use other features such as the user name embedding similarity, and the profile image feature vector similarity to train our model. We have performed experiments on two pairs of social networks, namely the Facebook-Twitter and the Foursquare-Twitter [1]. For these networks we have extracted as features some widely available similarity metrics that use the structure of the networks, but extended to two networks, such as that of the common neighbors, the Jaccard coefficient, and the Adamic-Adar. First, each of these structural based features were independently used for the heuristic method. Then we train our supervised models by combining all of the above features. The effects of using these features separately and in a combined manner were observed.

II. Preliminaries and Methodology

Let us start by formally defining our problem before we describe in detail the features we use. The network alignment problem takes as input two graphs G_s and G_t, as well as a set of anchor links A between these two graphs. In more detail, the inputs consist of graph $G_s = (V_s, E_s)$ and graph $G_t = (V_t, E_t)$, where the vertex sets V_s, V_t represent users and the edge sets E_s, E_t represent friendships between users of G_s and G_t respectively. Let us denote with A the set of known anchor links, i.e., $A = \{(v,u), v \in V_s, u \in V_t\}$, A is a small set of matches that matches one user from graph G_s to a user to other graph G_t. Given these inputs the network alignment problem is the task of determining whether a new link $(x,y) \notin A$ of users x and y, where $x \in V_s$ and $y \in V_t$, is an anchor link. In our approach to the problem we divide it into two phases: (i) Feature extraction, and (ii) Anchor link prediction. This problem can be considered an extension to the link prediction task on a single graph [8]. In this work, the solution assumes the presence of some already known anchor links between the source and target network, i.e., A is non-empty.

Feature Description: As we will be using local structural features, we choose commonly used measures for link prediction [8], [9], [10]. For the anchor link prediction task, we have adapted these features from their usual single network to a two network setting. Apart from structural features, we use two other features, namely the username embedding similarity and the profile image embedding similarity.

1) Extended Common Neighbors (ECN): For a pair of users (v,u) from two different networks $v \in V_s$ and $u \in V_t$, their ECN score is the number of their neighbors, that contribute to an anchor link [11]. If $\Gamma(v)$ represent the set of neighbors of v in G_s (or G_t), then the Extended Common Neighbors score is defined by the equation
$ECN(v,u) = |(x,y) \in A, x \in \Gamma(v), y \in \Gamma(u)|$

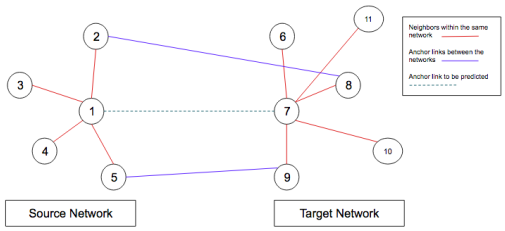

Figure 1: Example of two networks with anchor links

As an example, consider Figure 1, the red lines are edges from G_s and G_t and blue lines are pairs from A. The number of neighbors of User 1 in G_s is 4 and number of neighbors of User 7 in G_t is 5. Out of these neighbors, the number of anchor users is four with two anchor links between them. As these four users are essentially two users, we see that $ECN(1,7) = 2$.

2) Extended Jaccard Co-Efficient ($EJCE$): The Extended Jaccard Co-Efficient of two users is their ECN score divided by the number of distinct neighbors of these users [11]. It can be formally defined as:

$$EJCE(v,u) = \frac{|ECN(v,u)|}{|\Gamma(v) \cup \Gamma(u)|}$$

where $|\Gamma(v) \cup \Gamma(u)| = |\Gamma(v)| + |\Gamma(u)| - |\Gamma(v) \cap \Gamma(u)|$. Here the term $|\Gamma(v) \cap \Gamma(u)|$ counts aligned users.

For example in Figure 1, the $EJCE$ of User 1 and User 7 is obtained by dividing the ECN by the distinct neighbors of the user pair, which is 5, thus $EJCE = 2/5$.

3) Extended Adamic/Adar Measure ($EAAM$): For a pair of nodes v, u from two different networks, their $EAAM$ score is the sum of the inverse log of the average degree of pairs of neighbors of these two users [11]. It is defined by the equation:

$$AA(v,u) = \sum \frac{1}{\log\left(\frac{|\Gamma(v_s)|+|\Gamma(u_t)|}{2}\right)}$$

where the measure is calculated over every pair $(v_s, u_t) \in |\Gamma(v) \cap \Gamma(u)|$.

4) Username Embedding Similarity (UES): This measures the closeness of username embeddings for the user pair under consideration. This representation is called the *factoid embedding of the user and the object* [1]. A given user in one network might have several username objects in that network. Jaro-Winkler distance, which measures edit distance, is used to measure the similarity of each username object of a user from a network to each username object of either user giving a vector of scores. The list of username objects from either user can be sorted, so the coordinates of this vector are nicely ordered. The cosine distance between embedding vectors of two users is then used as a feature in our model. If two users have similar embedding vectors, the cosine distance between them should be low. Here the cosine distance between two vectors x, y is defined as $1-$Cosine Similarity.

5) Profile Image Embedding Similarity ($PIES$): 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. The deep learning framework $VGG16$ is used to extract features from profile images of users, with pre-trained weights from ImageNet [1].

III. Algorithms

As we described earlier, our network alignment method first extracts the structural features from the two networks and then uses various unsupervised or supervised learning models to predict anchor links between user pairs. The unsupervised learning can be done using the feature scores described above along with a thresholding approach. However, in order to do the supervised learning we need to create labeled data consisting of rows of feature values for pairs of users together with a label for whether or not they are aligned. We next present two algorithms to carry these steps out based on what type of features have been extracted.

A. Algorithm 1 - With structural features

In the first step of the Procedure *NetAlignStr* of our Algorithm 1, we go through each known anchor link pair $(v, u) \in A$, where $v \in G_s$ and $u \in G_t$, and we create a dictionary, named *AnchorLink*, that stores v as the key and the corresponding anchor user u as the value. Additionally, we construct a list that includes all anchor users from G_s that are in A and similarly construct a list of anchor users from G_t that are in A. After that, we call the Procedure *getStructuralFeatures* given in Algorithm 1 to extract the structural features for each possible pair of users of the type (v, u), where $v \in G_s$ and $u \in G_t$. The extracted features are used to construct the training and the test set. A training (test) sample of a user pair is labelled as positive (1) if the user pair is present in the ground truth file otherwise it is labelled as negative (0). Note, that we denote by $deg(v)$ the degree of node v. After that step we can use a binary classifier to predict anchor links.

B. Algorithm 2 - With structural features, username and profile image embeddings

Given inputs G_s, G_t, and A as well as the user name and the profile embeddings, we run Algorithm 1 but in Step 10, we also look up the username and image embeddings for the users in the pair and compute the their UES and $PIES$ score. We then add these values as additional features in our labeled training data rows.

C. Algorithm 3 - Structural features independently

Every user pair (v, u), where $v \in G_s$ and $u \in G_t$, that belongs to the test set, is treated as a missing link. For each missing link the ECN, $EJCE$ and $EAAM$ are calculated. The scores obtained from each of the methods are added to their respective lists of scores, i.e., ECN to scoresECN, $EJCE$ to scoresEJCE and $EAAM$ to scoresEAAM. In the next step, the scores are 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.

IV. Experimental Results

Our network alignment experiments were conducted using the User Identity Linkage dataset [1]. It consists of data from two pairs of networks Foursquare-Twitter and Facebook-Twitter. Table I describes the number of users and the number of

Algorithm 1 Network Alignment with Structural Based Features

procedure NETALIGNSTR(G_s, G_t, A)
  for each (v, u) in A **do**
    AnchorLink$[v] = u$
    Append(ListAnchorSource, v)
    Append(ListAnchorTarget, u)
  for each $v \in V_s$ of the training set **do**
    for each $u \in V_t$ of the training set **do**
      create vector: (v, u),
      getStructuralFeatures(v, u),
      if $(v, u) \in A$ **then**
        $label = 1$
      else $label = 0$
procedure GETSTRUCTURALFEATURES(v, u)
  ECN $\leftarrow 0$, EJCE $\leftarrow 0$, EAAM $\leftarrow 0$
  for each edge $(v^s, x^s) \in E_s$, and $v^s = v$ **do**
    if x^s in ListAnchorSource **then**
      Append(SourceNeigh, x^s)
    else
      Append(NonAnchor, x^s)
  for each edge $(u^t, y^t) \in E_t$, and $u^t = u$ **do**
    if y^t in ListAnchorTarget **then**
      Append(TargetNeigh, y^t)
    else
      Append(NonAnchor, y^t)
  for each $w \in$ SourceNeigh **do**
    if AnchorLink[w] $\in$ TargetNeigh **then**
      ECN $++$
      EAAM$+= \frac{1}{\log^{-1} \frac{\mathsf{deg(w)+deg(AnchorLink[w])}}{2}}$
  EJCE $=$ (ECN)/cardinality(NonAnchor)
  return ECN, EJCE, EAAM

edges in each network in this dataset. This dataset also comes with a set of known anchor links. The Foursquare - Twitter has 3602 known anchor links. The Facebook - Twitter has 1998 known anchor links. Apart from the structural information, the username and profile image embedding vectors for users are included in the datasets.

Table I: Dataset Overview

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

We break down the descriptions of each of our experiments into the following parts: data preparation, training, and results. In the data preparation part we detail which pairs of the social networks are used and which of the algorithms are used. The training part describes which unsupervised or supervised training methods are used. Finally, the results part details how these performed using the specific previous parts. We next give some of the nuances of each of these parts before actually going into our experimental results.

A. Data Preparation

As we have described in Section III for the anchor link prediction, we have to label the data so that it can be used by a supervised learning algorithm. To prepare the training and test data, we use the ground truth file. Every user pair that appears in the ground truth file is labelled as a positive sample. For negative training samples, a user from G_s is paired with all other users from G_t, except the one with which it is paired in the ground truth file. We can mark this pair as a negative sample, as we definitely know that there can be no anchor link between this pair. For example see Table II, the user pair ($f14$, $t159$) in the ground truth file is labelled positive (1), and for any other Twitter user, e.g., $t28$, we make a pair, ($f14$, $t28$), and label the corresponding sample negative (0).

Table II: Supervised Learning Training Sample.

Pair	ECN	EJ CE	EA AM	UES	PI ES	label
$f14, t159$	6	0.09	-33.77	0.90	1	1
$f14, t28$	0	0.0	0	1	0.92	0

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 experiment with different numbers of training samples. We use two approaches to create these 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.

1) Upsampling and (no Downsampling): To increase the number of training samples, for every positively labelled training sample, empirically we observe that we create on average 20 negatively

labelled training samples. This causes the training data to be skewed towards negative samples [12]. To balance out the data we use an upsampling technique to increase the positive samples by randomly duplicating existing positive sample rows in the training file until we balance the positive and negative samples for training. Alternatively, we could have performed downsampling of the negative samples, but this would lead to data loss.

2) No upsampling: For every positively labelled sample, we create one negatively labelled training item, thus getting a balanced training set containing an equal number of positive and negative samples. We have a smaller but more balanced set.

B. Training

For our experiments, when we use Algorithm 3 as a component of our network alignment algorithm, we only extract the structural features from the data set. For our experiments involving Algorithm 1 and 2, we considered both structural features as well as the UES and the $PIES$. In the latter case, the training rows were just augmented with additional columns for the added features.

1) Unsupervised Learning Methods: Link prediction tasks lack labelled data. In such scenarios, we can use an unsupervised approach to predict links, see Algorithm 3. Some of them include the Jaccard Co-Efficient, the Extended common neighbors, and the Adamic/Adar scores that form the basis of some of the generally used unsupervised link prediction methods. As we described, we have appropriately modified these scores following [11] to suit the link prediction problem in a two network setting. These scores are purely based on the local structure of the networks under consideration.

To perform unsupervised training, a single link prediction score mentioned above is calculated for each pair of users from the two networks being aligned. A threshold is then chosen based on the scores of the known anchors links so as to minimize the number of false positives and maximize the number of true positives on this subset of pairs. This threshold is then used to determine the anchor links amongst the test data. For example, suppose we set a threshold of 20 for ECN. The following rules are followed to calculate true positives and false negatives. Any user pair that has $ECN \geq 20$ 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 $ECN \geq 20$ is treated as false positive. Any user pair which is not in the ground truth links and has $ECN < 20$ is treated as true positive. Any user pair which is in the ground truth links and has $ECN < 20$ is treated as false positive.

Supervised Learning Methods: For the supervised learning methods, we experiment with two distinct sets of features for training: One with only local structural features, see Algorithm 1, and another one that these also uses $PIES$ and UES, see Algorithm 2. Using each of these features sets we then examine the effect of using logistic regression, k nearest neighbors (kNN) and neural networks as the training methods. We experimented with two configurations for each of the kNN and the neural network. For kNN, we experimented with two configurations, $k = 3$ (not shown) and $k = 5$. Our neural network, when only the structural features are used, consisted of three input layer neurons. If also the embeddings are used, the input layer has 5 neurons. We experiment with two types of neural networks (NN). 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. A softmax function followed by categorical cross entropy function is used to measure the loss and accuracy. Using categorical cross entropy we classify a given training sample as either close to 0 or close to 1. We used 100 epochs to train our networks.

C. Results

We perform various experiments on the two sets of networks Foursquare-Twitter, and Facebook-Twitter. The results show the performance of various models with no upsampling (balanced) and upsampling of the training set.

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.

Table III: No upsampling (no up) and with upsampling (up).

	Foursquare - Twitter no up ǀ up	Facebook - Twitter no up ǀ up
Training	5763 ǀ 54568	3197 ǀ 30191
Testing	1439	798

Table III shows the number of training and test samples used for the models with and with

no upsampling on the training set to balance the positive and negative samples in the training set.

1) Results using Algorithm 1: Table IV shows the anchor link prediction accuracy of the model that uses only the three local structural features. No upsampling was performed on the training set. For the Foursquare-Twitter dataset kNN with $k = 5$ gives the best results. The neural network with a single dense layer gives the best accuracy for the Facebook-Twitter dataset.

Table IV: Prediction accuracy for Algorithm 1, no upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Log. Regression	0.8700	0.7102
$kNN(k = 5)$	**0.8769**	0.5006
NN(1 DenseLayer)	0.8601	**0.7163**
NN(2 DenseLayers)	0.8601	0.7160

Table V shows the anchor link prediction accuracy for the model that uses the three local structural features along with upsampling on the training set to balance out the positive and negative samples. As can be seen, logistic regression gave the best results for the Foursquare-Twitter dataset. The kNN classifier performed poorly on both the datasets. The neural network did best for the Facebook-Twitter dataset. The logistic regression and neural network curves show that they did better for the Foursquare-Twitter, Facebook-Twitter datasets, respectively.

Table V: Prediction accuracy for Algorithm 1, with upsampling.

Methods	Foursquare -Twitter	Facebook -Twitter
Logistic Regression	**0.8756**	0.7101
$kNN(k = 5)$	0.4808	0.4981
NN(1DenseLayer)	0.8582	**0.7164**
NN(2DenseLayers)	0.8582	**0.7164**

2) Results using Algorithm 2: Table VI shows the anchor link prediction accuracy of the model that uses the three local structural features along with the username and the image embedding similarity features with no upsampling. As can be seen from Table VI, the kNN classifier with $k = 5$ gives the best results on the Foursquare-Twitter dataset. The neural network with two layers did the best for the Facebook-Twitter dataset.

By comparing Tables IV and Table VI, one can see that the results are better when the username and profile image embedding similarity are used as features on top of the structural features. All the models give considerably better predictions compared to the models of Algorithm 1.

Table VI: Prediction Accuracy for Algorithm 2, no upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	0.9478	0.8883
$kNN(k = 5)$	**0.9513**	0.8858
NN(1DenseLayer)	0.9375	0.8827
NN(2DenseLayers)	0.9420	**0.8962**

Table VII shows the anchor link prediction accuracy of the model that uses the three local structural features along with username and image embedding similarity features and with upsampling. As can be seen, logistic regression gave the best results on the Foursquare-Twitter dataset. The neural network with two layers did the best for the Facebook-Twitter dataset. Compared to the no upsampling data of Table VI, the with upsampling data gives slightly higher results for this dataset.

Table VII: Prediction Accuracy for Algorithm 2, with upsampling.

Methods	Foursquare - Twitter	Facebook - Twitter
Logistic Regression	**0.9513**	0.8895
KNN(k=5)	0.9284	0.8732
NN(1DenseLayer)	0.9479	0.8981
NN(2DenseLayers)	0.9481	**0.8986**

From the results in Table VII, the performance of the models are not greatly affected by upsampling of the data during training. All the models were able to give fairly good predictions for the data.

3) Results using Algorithm 3: The results in Table VIII show that when the ECN, $EJCE$, and $EAAM$ measures are used as independent methods, the AUC score is lower as compared to the supervised setting when they are combined together as seen in Tables IV through VII. The AUC Scores for ECN and $EJCE$ are similar, as $EJCE$ 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 $EAAM$ measure performed poorly when used as an independent approach to the problem.

4) Comparison between supervised and unsupervised approaches: For the Foursquare - Twitter dataset when these unsupervised features were

Table VIII: AUC Scores of Unsupervised methods.

Methods	Foursquare - Twitter	Facebook - Twitter
ECN	0.85	0.68
EJCE	0.85	0.68
EAAM	0.15	0.32

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 highest AUC score we obtained was 0.85 with ECN and EJN. For the Facebook - Twitter network, the highest AUC score of 0.71 with logistic regression and neural network models; whereas, the highest AUC score obtained with unsupervised methods was 0.68. Figure IX compares our results against the results of [1] that use of factoid embedding (FE) with $HR@1$ (HitRate@K ($HR@K$, K ranking) for the same datasets.

Table IX: Comparison of our HR@1 versus Factoid Embedding [1].

Methods	Foursquare - Twitter	Facebook - Twitter
Semi-Supervised [1]	0.55	0.68
Unsupervised [1]	0.54	0.68
Our Unsupervised	0.85	0.68
Our Supervised	0.88	0.71

The $HR@1$ score of both semi-supervised and unsupervised approaches in [1] on the Foursquare-Twitter dataset is far below the predictions we could achieve through our approach. The $HR@1$ score on the Facebook-Twitter dataset is comparable to our unsupervised approach. The predictions, though, of our supervised approach are slightly higher.

V. Related Work

The network alignment problem is similar to the maximum sub-graph problem or the bipartite graph matching problem. The problem has application also in bioinformatics, for protein-protein interaction networks [13]. Such networks are mainly homogeneous. Earlier, matrix based computations were used in order to solve network alignment problems [14]. In our setting there are several factors, which make whole graph approaches, such as matrix approaches, less practical. The most important reason is that many social networks are in the big data realm and the associated matrices are huge, and therefore hard/impractical to work with. Additionally, access to the data might be bandwidth limited and thus not available in its entirety for use in a feasibly timely manner. Therefore, we aim to explore the supervised, and unsupervised approaches to network alignment in OSNs using only the local structural setting. Below we review some related work that use similar approaches.

The supervised approaches typically make use of feature vectors constructed from node attributes. At a high level, the feature vectors are used to train models and to obtain similarity scores. Based on these similarity scores, anchor links can be inferred using different matching techniques.

Following this approach is the work by X. Kong et al. [11]. They assume the presence of some existing anchor links between the networks to be aligned as we do. Social features similar to ours are used as unsupervised forms of training. Two additional features, the location vectors of a user's posts on two different social networks and the timestamp of the posts, are added. Additionally, they use stable matching techniques for full alignments.

Z. Jiawei et al [3] propose a solution to network alignment in partially aligned networks. They use the concept of anchor meta paths constructed using the relationship between various heterogeneous components to perform alignments. This work assumes that the networks are partially aligned. As the networks are heterogeneous in nature, different types of nodes like user, location, timestamp are present and various types of relation exists between nodes like follow, create, check-in.

Komamizu et al. [12] study the problem of network alignment between Github and Stack Overflow users. They construct anchor links through the common email addresses of the users between the two networks. Similar attributes like username, terms from the project description, terms from posted questions, and account creation date were extracted from both networks. Then cosine similarity between bag of words vectors are computed. Date and time are also compared using inverse of duration between the dates. A number of classification methods are used.

Several unsupervised approaches to OSN network alignment have also been explored. Wei Xie, et al. [1] consider available information like username and profile image to construct factoids.

Recently, many graph-based machine learning tasks rely on the use of appropriate feature-based learning representations of the graph that produce node embeddings. REGAL [15] is an unsupervised learning representation for the network alignment

problem.

VI. Conclusion

Our approach primarily considers solving the network alignment problem using structural based features, that use only local parts of a network and not the whole network, and thus can be computed efficiently. In the version we are considering we assume that the anchor links are between two nodes, each one from a different network. From the results and related work on the same problem, it is evident that just the topological based features may not be sufficient to make good predictions in a supervised setting. Adding some additional attributes that are widely available, like the username and profile images can improve the results. We anticipate that the availability of more information entities will give better results. The solution we propose and evaluate also makes the assumption that we have some known anchor links. One future direction is to solve the problem in a cold start setting, without any prior knowledge of known anchor links. A further future direction would be to find efficient ways to extract more information related to users, such as location and language, and then evaluate their effect on the performance of our models.

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*, pp. 1338–1343, Nov 2018.

[2] J. Zhang and P. Yu, "Broad learning: An emerging area in social network analysis," *ACM SIGKDD Explorations Newsletter*, vol. 20, pp. 24–50, 2018.

[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*, pp. 166–173, 2015.

[4] M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, and Y. Wang, "Algorithms for large, sparse network alignment problems," in *9th IEEE International Conference on Data Mining*, pp. 705–710, 2009.

[5] D. Koutra, H. Tong, and D. Lubensky, "BIG-ALIGN: fast bipartite graph alignment," in *13th IEEE International Conference on Data Mining*, pp. 389–398, 2013.

[6] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *22nd ACM International Conference on Knowledge Discovery and Data Mining*, pp. 1345–1354, 2016.

[7] H. T. Trung, N. T. Toan, T. Van Vinh, H. T. Dat, D. C. Thang, N. Q. V. Hung, and A. Sattar, "A comparative study on network alignment techniques," *Expert Systems with Applications*, vol. 140, p. 112883, 2020.

[8] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[9] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[10] L. Lü, C.-H. Jin, and T. Zhou, "Similarity index based on local paths for link prediction of complex networks," *Physical Review E*, vol. 80, no. 4, p. 046122, 2009.

[11] X. Kong, J. Zhang, and P. S. Yu, "Inferring anchor links across multiple heterogeneous social networks," in *22nd ACM International Conference on Information & Knowledge Management*, pp. 179–188, 2013.

[12] T. Komamizu, Y. Hayase, T. Amagasa, and H. Kitagawa, "Exploring identical users on github and stack overflow," in *29th International Conference on Software Engineering and Knowledge Engineering*, pp. 584–589, 2017.

[13] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *the National Academy of Sciences of the USA*, vol. 105 35, pp. 12763–12768, 2008.

[14] G. Kollias, S. Mohammadi, and A. Grama, "Network similarity decomposition (NSD): A fast and scalable approach to network alignment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, pp. 2232–2243, Dec 2012.

[15] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *27th ACM International Conference on Information and Knowledge Management*, pp. 117–126, 2018.