Deep Heterogeneous Social Network Alignment

Lin Meng*, Yuxiang Ren*, Jiawei Zhang*, Fanghua Ye†, Philip S. Yu‡
*IFM Lab, Department of Computer Science, Florida State University, FL, USA
†Department of Computer Science, University College London, UK
‡Department of Computer Science, University of Illinois at Chicago, IL, USA
{lin, yuxiang, jiawei}@ifmlab.org, smartyfh@outlook.com, psyu@uic.edu

***Abstract*—The online social network alignment problem aims at inferring the anchor links connecting the shared users across social networks, which are usually subject to the one-to-one cardinality constraint. Several existing social network alignment models have been proposed, many of which are based on the supervised learning setting. Given a set of labeled anchor links, a group of features can be extracted manually for the anchor links to build these models. Meanwhile, such methods may encounter great challenges in the application on real-world social network datasets, since manual feature extraction can be extremely expensive and tedious for the social networks involving heterogeneous information. In this paper, we propose to address the heterogeneous social network alignment problem with a deep learning model, namely DETA (Deep nETwork Alignment). Besides a small number of explicit features, DETA can automatically learn a set of latent features from the heterogeneous information. DETA models the anchor link *one-to-one* cardinality constraint as a mathematical constraint on the node degrees. Extensive experiments have been done on real-world aligned heterogeneous social network datasets, and the experimental results have demonstrated the effectiveness of the proposed model compared against the existing state-of-the-art baseline methods.**

***Index Terms*—Social Network Alignment; Information Fusion; Deep Learning; Data Mining**

I. Introduction

Formally, social network alignment denotes the problem of inferring the anchor links connecting the shared user accounts across different online social networks [11]. In recent years, numerous online social networks have appeared, which can provide various featured services for the users. For instance, Facebook allows users to establish connections with their friends, family members and classmates; Twitter provides users with the latest news information from the public; LinkedIn helps people to establish their professional profiles for job hunting; Foursquare enables people to keep track of their visited locations in the offline world. To enjoy such diverse social network services simultaneously, users nowadays are usually involved in multiple online social networks at the same time. However, in the real world, these online social networks are typically separated from each other without any correspondence relationships [11].

Social network alignment can be the prerequisite task for information fusion across multiple online social platforms. By integrating the information about the shared users from multiple social sites together, it will provide researchers and practitioners with the opportunity to achieve a more comprehensive knowledge about the users' social activities. Meanwhile, for the social network service providers, social network alignment also allows them to improve their services greatly by retrieving useful information from other external social network platforms. Various application services can benefit from the alignment results, e.g., friend recommendation, community detection and information diffusion, which renders the social network alignment to be a crucial learning task.

Many existing social network alignment models are mainly proposed based on the supervised algorithms. Given a set of labeled anchor links, the anchor link inference problem can be modeled as a binary classification task [11], where the existing and non-existing anchor links are labeled as the positive and negative instances, respectively. Based on the social network information, a set of features will be extracted manually for the anchor links. These features are usually designed for the specific input social networks, which can hardly be generalized to other regular social networks. Such a problem will be much more severe when it comes to the real-world heterogeneous social networks, involving multiple types of nodes and complex links, since manual feature design and extraction across these heterogeneous networks will become extremely expensive and tedious.

Problem Studied: In this paper, we propose to study the heterogeneous social network alignment problem, which aims at finding common users among multiple heterogeneous networks. Besides a small number of simple explicit features extracted from user pairs, we further propose to learn a group of latent features automatically based on deep learning models from the heterogeneous social networks, which will be fused together to infer the anchor links across networks.

The heterogeneous social network alignment problem is hard to address due to the following challenges:

- *Network Heterogeneity*: The online social networks usually contain heterogeneous information, involving various categories of nodes and links, which creates great challenges for both explicit feature extraction and the automatic latent feature representation learning for the anchor links.
- *Information Integration*: There exists a two-phase information integration in the studied problem: (1) explicit and latent feature integration for each information category, and (2) cross-category information integration. A new information fusion model which can achieve both of these objectives will be desired and necessary.

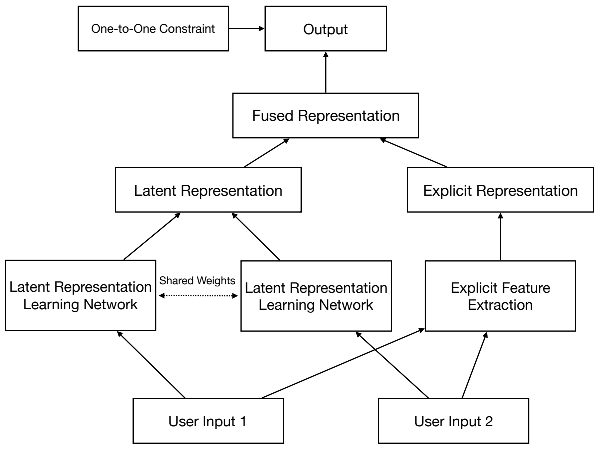


Fig. 1. The DETA Framework.

- *One-to-One Constraint*: The anchor links to be inferred across networks are subject to the *one-to-one* cardinality constraint [11], where each user should be connected by at most one anchor link across networks. Such a strict constraint will pose great challenges on both problem formulation and the model learning.

To resolve the aforementioned challenges, in this paper, we will propose a novel deep heterogeneous social network alignment model, namely DETA (Deep nETwork Alignment). DETA extracts a very small number of general and explicit features for the anchor links across the networks, and learns a group of latent features from each information category with specific deep learning models (including deep autoencoder for social connections, LSTM for trajectory records, and CNN for textual words). Furthermore, DETA fuses these diverse features across information categories with a deep model. The *one-to-one* cardinality constraint on anchor links is modeled as the mathematical constraint on the node degrees in DETA.

II. Terminology and Problem Definition

In this section, we will first define the terminologies used in this paper and then provide the formulation of the heterogeneous social network alignment problem.

A. Notations

In the following sections, we will use the lower case letters (e.g., x) to represent scalars, lower case bold letters (e.g., $\mathbf{x}$) to denote column vectors, bold-face upper case letters (e.g., $\mathbf{X}$) to denote matrices, and upper case calligraphic letters (e.g., $\mathcal{X}$) to denote sets. Given a matrix $\mathbf{X}$, we denote $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ as the i_{th} row and j_{th} column of $\mathbf{X}$. The (i_{th}, j_{th}) entry of matrix $\mathbf{X}$ can be denoted as either $X(i,j)$ or $X_{i,j}$. We use $\mathbf{X}^\top$ to represent the transpose of matrix $\mathbf{X}$. The F-norm of matrix $\mathbf{X}$ can be represented as $\|\mathbf{X}\|_F = (\sum_{i,j} |X_{i,j}|^2)^{\frac{1}{2}}$. The element-wise product of vectors $\mathbf{x}$ and $\mathbf{y}$ of the same dimension is represented as $\mathbf{x} \otimes \mathbf{y}$. The concatenation of vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted as $\mathbf{x} \sqcup \mathbf{y}$ and $Tr(\mathbf{X})$ denotes the trace of matrix $\mathbf{X}$.

B. Terminology Definition

To study the social network alignment problem, we need to defined several terminology first.

Definition 1 (Heterogeneous Social Network): Formally, the heterogeneous social network studied in this paper includes multiple types of entities $\mathcal{V}$ and multiple types of links $\mathcal{E}$. We will use $\mathcal{U}$, $\mathcal{T}$, and $\mathcal{D}$ to represent the sets of users, trajectories and textual words involved in the network respectively and $\mathcal{E}_{u,u}$, $\mathcal{E}_{u,t}$ and $\mathcal{E}_{u,d}$ to denote those different types of links among the entities. Hence, a heterogeneous social network can be represented as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{T} \cup \mathcal{D}$ and $\mathcal{E} = \mathcal{E}_{u,u} \cup \mathcal{E}_{u,t} \cup \mathcal{E}_{u,d}$.

For each user $u_i \in \mathcal{U}$, via the connections, we can easily retrieve his/her neighbors, historical trajectory and used words from the network. If the data types of a user are incomplete, we will discard the component corresponding to the missing data type.

Definition 2 (Aligned Heterogeneous Social Networks): Given n heterogeneous social networks $G^{(1)}, G^{(2)}, ..., G^{(n)}$ sharing common users, we can represent them as the multiple *aligned heterogeneous social networks* $\mathcal{G} = \big((G^{(1)}, \cdots, G^{(n)}), (\mathcal{A}^{(1,2)}, \mathcal{A}^{(1,3)}, \cdots, \mathcal{A}^{(n-1,n)})\big)$. In the notation, $\mathcal{A}^{(i,j)}$ denotes the set of undirected anchor links between networks $G^{(i)}$ and $G^{(j)}$, which can be represented as follows formally:

$$\mathcal{A}^{(i,j)} = \{(u^{(i)}, v^{(j)}) | u^{(i)} \in \mathcal{U}^{(i)}, v^j \in \mathcal{U}^{(j)}, \\ u^{(i)} \text{ and } v^{(j)} \text{ are the same user.}\}$$

where $\mathcal{U}^{(i)}$ and $\mathcal{U}^{(j)}$ denote the user sets of networks $G^{(i)}$ and $G^{(j)}$, respectively. Furthermore, as indicated in the existing works [11], anchor links between social networks are subject to the *one-to-one* cardinality constraint, which means each user in $G^{(i)}$ can only be connected to at most one user in $G^{(j)}$.

C. Problem Formulation

To simplify the settings, in this paper, we will use pairwise network alignment as an example to introduce the problem definition and the proposed models. An easy extension of the proposed model can be applied to the alignment problem among more than two heterogeneous social networks as well.

Problem Definition: Formally, given a pair of partially aligned social networks $\mathcal{G} = \big((G^{(1)}, G^{(2)}), (\mathcal{A}^{(1,2)})\big)$ with a very small number of observed anchor links $\mathcal{A}^{(1,2)}$, we can represent all the potential anchor links between networks $G^{(1)}$ and $G^{(2)}$ as $\mathcal{C} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$. Based on the known anchor links $\mathcal{A}^{(1,2)}$ together with the information in networks $G^{(1)}$ and $G^{(2)}$, we aim at building a mapping $f : \mathcal{C} \to \mathcal{Y}$ to project the potential anchor links to a pre-defined label space $\mathcal{Y} = \{+1, 0\}$, where the existing anchor links will be assigned with label $+1$ while the non-anchor links are assigned with label 0. Significantly different from the classification tasks of regular links [15], [7], as mentioned above, the anchor links studied in this paper are subject to the *one-to-one* cardinality constraint, which poses a strict constraint on the prediction task.

III. Proposed Method

In this section, we will introduce our method in detail. As indicated in Figure 1, given a user pair from two different social networks, the proposed model DETA is capable of computing the users' feature representations based on their heterogeneous information, which will be further fused to generate the output. Among the functional modules in the framework, the main components include the "latent representation learning network" and "explicit feature extraction", and their detailed architecture is illustrated in Figure 2. In this section, we will first introduce the representation learning network model with the heterogeneous social information, including social connections, trajectories and textual words in Sections III-A - III-C, respectively, and then provide the overall framework and learning algorithm in Section III-D.

A. Social Connection Representation Learning

Users' social connections can clearly indicate their social preferences, which provide important signals for inferring the shared users across networks. As indicated by the central part in Figure 2, the DETA can extract a set of *explicit features* and *latent features* from the social network for the anchor links. In this part, we will introduce these latent and explicit features extracted from the social network information in detail.

1) Latent Feature Extraction: In DETA, the social connection latent feature vectors are learned with an extended deep autoencoder model. Traditional deep autoencoder [25] is an unsupervised model, which aims at learning a dense, low-dimensional feature representation for the sparse and high-dimensional input. The autoencoder model includes two parts: encoder and decoder.The encoder maps the input vector into a target low-dimensional space, then the decoder restores the mapped vector to the input vector in a reconstructed space (the same dimension as the original input). The deep autoencoder model aims at minimizing the differences of the input vector against the reconstructed vector, so as to learn the model variables.

When applying the deep autoencoder model to learn the latent features for social connections, we regard users' social neighborhood information as the input. Formally, given the user set $\mathcal{U}$ and the social connection set $\mathcal{E}_{u,u}$ in network G (which can be either $G^{(1)}$ or $G^{(2)}$), we can represent the social links available in network G as the adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{U}|}$, where $\mathbf{A}(i,j) = 1$ iff $(u_i, u_j) \in \mathcal{E}_{u,u}$ and $u_i, u_j \in \mathcal{U}$. User u_i's social neighborhood can be indicated by his/her corresponding row in $\mathbf{A}$ (i.e., $\mathbf{x}_i = \mathbf{A}(i,:)$). Because of the nature of $\mathbf{A}$, the original objective function is capable to capture the *second-order proximity* of nodes in the network, i.e., nodes with similar neighbors (i.e., the input vectors to the model) will have similar latent representations (i.e., the output vectors).

However, slightly different from the classic learning settings with independent data instances, the user nodes in social networks are highly connected, whose learning results will be strongly correlated with each other. For instance, given two connected user nodes (even if they have totally different neighbors), their learned latent feature vectors should be close to each other in the objective low-dimensional space. Based on such an intuition, we extend the deep autoencoder model by incorporating such a *first-order proximity* [26] into consideration. Formally, given users u_i and u_j, based on their input feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, we can represent their learned latent feature vectors as $\mathbf{z}_i$ and $\mathbf{z}_j$, respectively. To preserve the *first-order proximity*, we introduce a loss term as follows:

$$\mathcal{L}_1 = \sum_{u_i,u_j \in \mathcal{U}} S_{i,j} \cdot \parallel \mathbf{z}_i - \mathbf{z}_j \parallel_2^2 = Tr(\mathbf{Z}^\top \mathbf{L}\mathbf{Z}),$$

where $\mathbf{L}$ is the laplacian matrix, which can be compute by $\mathbf{L} = \mathbf{D} - \mathbf{S}$, and $\mathbf{D}$ is the diagonal matrix corresponding to $\mathbf{S}$. Matrix entry $S_{i,j}$ denotes whether u_i and u_j are connected or not. Here we give its mathematical represention as follows:

$$S_{i,j} = \begin{cases} 1, & \text{if } u_i \text{ and } u_j \text{ are connected;} \\ -1, & \text{if } u_i \text{ and } u_j \text{ are not connected.} \end{cases}$$

Instead of $\mathbf{A}$, $\mathbf{S}$ can capture difference between users better. Normally, the social network connections are very sparse and most of the elements in $\mathbf{A}$ are zeros. In such a setting, simply adopting the traditional deep autoencoder may lead to trivial solutions, and the social network information may get lost in the learned user representation vectors. Hence, we further extend loss function with the *second-order proximity* of the deep autoencoder model by adding a vector $\mathbf{b}_i$ to emphasize the importance of the non-zero entries and the embedding loss term can be changed into:

$$\mathcal{L}_2 = \sum_{u_i \in \mathcal{U}} \parallel (\mathbf{x}_i - \hat{\mathbf{x}}_i) \otimes \mathbf{b}_i \parallel_2^2,$$

where the $\hat{\mathbf{x}}_i$ denotes the recovered input vector from $\mathbf{z}_i$ and the elements of $\mathbf{b}_i$ is defined based on the input vector $\mathbf{x}_i$ as follows:

$$b_i(j) = \begin{cases} \alpha, & \text{if } x_i(j) = 1; \\ 1, & \text{if } x_i(j) = 0. \end{cases}$$

where $\alpha > 1$ is the larger weight assigned for the non-zero elements. In the experiments, α is usually determined by the ratio of the numbers about the zeros and ones in vector $\mathbf{x}_i$.

By adding up the loss terms of the *first-order proximity* and the *second-order proximity*, we can represent the latent social connection feature extraction objective function as follows:

$$\mathcal{L}_{SNE} = \beta \cdot \mathcal{L}_1 + \eta \cdot \mathcal{L}_2.$$

Parameters β and η denote the corresponding loss term weights. According to the descriptions, we can represent the extracted latent feature vectors for users $u_i^{(1)}$ and $u_j^{(2)}$ as vectors $\mathbf{x}_{\mathrm{S},i}^{\mathrm{LAT}}$ and $\mathbf{x}_{\mathrm{S},j}^{\mathrm{LAT}}$, respectively.

2) Explicit Feature Extraction: Besides the latent features learned by the model automatically, to get more concrete information of social networks, we also propose to extract several explicit features across social platforms to improve performance of the model. These extracted features include the product of user pair's degree, the extended common neighbors, the extended Jaccard's coefficient and the extended Adamic/Adar measure, respectively. Given a user pair $(u_i^{(1)}, u_j^{(2)})$, the extracted product of node degrees can be represented as:

$$Degree(u_i^{(1)}, u_j^{(2)}) = \left\| \mathbf{x}_i^{(1)} \right\|_1 \times \left\| \mathbf{x}_j^{(2)} \right\|_1,$$

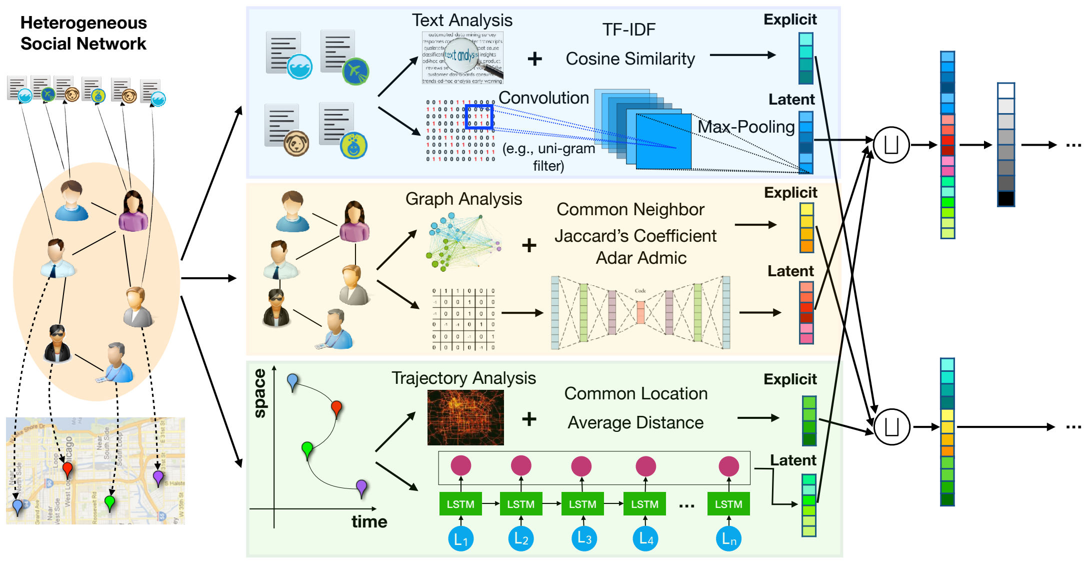


Fig. 2. Deep Representation Learning Model.

where $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_j^{(2)}$ denote the social neighbor vectors of $u_i^{(1)}$ and $u_j^{(2)}$, respectively. As for extended common neighbors, extended Jaccard's coefficient and extended Adamic/Adar measure, they are defined based on the known anchor links in the training set. The key idea of calculating the three features is based on the observation that users in different networks can be linked by known anchor links. $CN(u_i^{(1)}, u_j^{(2)})$ represents the number of extended common neighbors between $u_i^{(1)}$ in network $G^{(1)}$ and $u_j^{(2)}$ in network $G^{(2)}$ and it can be formally defined as follows:

$$\begin{aligned} CN(u_i^{(1)}, u_j^{(2)}) &= \left|\Gamma(u_i^{(1)}) \textstyle\bigcap_{\mathcal{A}} \Gamma(u_j^{(2)})\right| \\ &= |\{(u_p^{(1)}, u_q^{(2)}) \in \mathcal{A}, u_p^{(1)} \in \Gamma(u_i^{(1)}), u_q^{(2)} \in \Gamma(u_j^{(2)})\}|, \end{aligned}$$

where $\Gamma(u_i^{(1)})$ denotes the neighbors of $u_i^{(2)}$ in network $G^{(1)}$ and $\Gamma(u_j^{(2)})$ denotes the neighbors of $u_j^{(2)}$ in network $G^{(2)}$. For the extended Jaccard's coefficient, it rescales the "extended common neighbor" by considering the anchor links between users:

$$JC(u_i^{(1)}, v_j^{(2)}) = \frac{|\Gamma(u_i^{(1)}) \bigcap_{\mathcal{A}} \Gamma(v_j^{(2)})|}{|\Gamma(u_i^{(1)}) \bigcup_{\mathcal{A}} \Gamma(v_j^{(2)})|},$$

$$\left|\Gamma(u_i^{(1)}) \bigcup_{\mathcal{A}} \Gamma(u_j^{(2)})\right| = |\Gamma(u_i^{(1)})| + |\Gamma(u_j^{(2)})| - \left|\Gamma(u_i^{(1)}) \bigcap_{\mathcal{A}} \Gamma(u_j^{(2)})\right|.$$

For the extended Adamic/Adar measure, it further assigns each "extended common neighbor" a unique weight, which is determined by their degrees:

$$AA(u_i^{(1)}, u_j^{(2)}) = \sum_{\forall (u_p^{(1)}, u_q^{(2)}) \in \Gamma(u_i^{(1)}) \bigcap_{\mathcal{A}} \Gamma(u_j^{(2)})} \log^{-1}\left(\frac{|\Gamma(u_p^{(1)})| + |\Gamma(u_q^{(2)})|}{2}\right).$$

These four extracted features compose the explicit social features $\mathbf{x}_{\mathrm{S}(i,j)}^{\mathrm{EXP}}$ for the user pair $(u_i^{(1)}, u_j^{(2)})$ across networks.

B. Trajectory Representation Learning

Nowadays, online social networks provide the services to allow users having check-ins at various locations, which has become a significant feature in heterogeneous social networks. From users' trajectory information, we can capture users' spatial social behavior patterns, which will be useful for inferring the anchor links connecting the shared users. In order to make a good use of trajectory information, besides several simple explicit features, we propose to use Long Short-Term Memory (LSTM) [8] to extract a set of latent trajectory features for users.

1) Latent Feature Extraction: The trajectory latent features vector is learned with a one-layer LSTM. For the sequence input, LSTM has a wonderful advantage in maintaining an effective "memory" of the information in a sequence. Given the check-in records of one user in an online social network, we will first divide the activity regions into a set of blocks, where each block is assigned with a unique ID. For the locations within the same block, they will be assgined with the same block ID by default. This is reasonable since the social network check-in coordinate pairs are not precise. Also, our goal is to predict the anchor links, the exact check-in location matching can hardly get more useful information for identifying the common users. Therefore, grid blocks can provide a fuzzy representation of users' check-in activities. Formally, given a user's check-in records, we will represent them as a sorted sequence according to the timestamps, where the entries denote the corresponding block IDs.

Given a user set $\mathcal{U}$, the complete timestamp set $\mathcal{Q}$, and user-trajectory set $\mathcal{E}_{u,t}$ in network G (which can be either $G^{(1)}$ or $G^{(2)}$), we first divide the activity regions into n blocks and represent the user's trajectory records as a decimal matrix $\mathbf{T}_{dec} \in \{0, 1, 2, ..., n\}^{|\mathcal{U}| \times |\mathcal{Q}|}$. Matrix entry $T_{dec}(i, j) \in \{1, 2, ..., n\}$ denotes the block that user $u_i \in \mathcal{U}$ visited at time $\tau_j \in \mathcal{Q}$ in his historical trajectory. In model learning, binary representations may take fewer rounds to achieve convergence than decimal representations. Therefore, for each entry in $\mathbf{T}_{dec}$, we can replace it with the binary code, which will lead to a binary trajectory matrix $\mathbf{T}_{bin} \in \{0,1\}^{|\mathcal{U}| \times (\log n \cdot |\mathcal{Q}|)}$ and it can be simply denoted as $\mathbf{T}$ by default in this paper.

In order to ensure similar trajectories will have similar latent feature representations, we make some extensions to the traditional LSTM model. First, we add one fully-connected layer to project $\mathbf{T}(i,:)$ to a vector $\hat{\mathbf{x}}_i$ as follows:

$$\hat{\mathbf{x}}_i = \sigma(\mathbf{W}_1 \mathbf{T}(i,:) + \mathbf{b}_1).$$

where $\mathbf{W}_1$ and $\mathbf{b}_1$ denote the weight and bias variables. These projected vectors, e.g., $\hat{\mathbf{x}}_i$, will be fed to a RNN model with k LSTM cells, whose outputs can be denoted as vectors $\mathbf{y}_{i,1}$, $\mathbf{y}_{i,2}$, $\cdots$, $\mathbf{y}_{i,k}$ respectively. To make sure all information can be maintained, we will combine all the outputs of LSTM cells via another fully-connected layer with sigmoid activation function as indicated by the following equation:

$$\begin{cases} \hat{\mathbf{y}}_i = & \mathbf{y}_{i,1} \sqcup \mathbf{y}_{i,2} \sqcup ... \sqcup \mathbf{y}_{i,k}, \\ \mathbf{z}_i = & \sigma(\mathbf{W}_2 \hat{\mathbf{y}}_i + \mathbf{b}_2). \end{cases}$$

where $\mathbf{z}_i$ will be the final latent feature vector learned for user u_i based on the trajectory. In the equation, $\mathbf{W}_2$ and $\mathbf{b}_2$ denote the weight and bias variables involved in the model. According to the descriptions, we can represent the extracted latent feature vectors based on the trajectory for users $u_i^{(1)}$ and $u_j^{(2)}$ as vectors $\mathbf{x}_{\text{T},i}^{\text{LAT}}$ and $\mathbf{x}_{\text{T},j}^{\text{LAT}}$, respectively.

2) Explicit Feature Extraction: We extract several explicit features from the trajectory data as well. The first one coming into our mind is the locations shared by users $u_i^{(1)}$ and $u_j^{(2)}$. Formally, we will use $\mathcal{M}_i^{(1)}$ and $\mathcal{M}_j^{(2)}$ to denote the location sets of $u_i^{(1)}$ and $u_j^{(2)}$ and it can be defined as follows:

$$CL(u_i^{(1)}, u_j^{(2)}) = \left| \mathcal{M}_i^{(1)} \cap \mathcal{M}_j^{(2)} \right|.$$

In online social networks, different users have different activity regions, and the active region distances can be effectively indicated by the average distances of the check-in locations. The second explicit feature extracted for the user pair $(u_i^{(1)}, u_j^{(2)})$ is the average distance between two location sets of $u_i^{(1)}$ and $u_j^{(2)}$, which can be denoted as:

$$AD(u_i^{(1)}, u_j^{(2)}) = \frac{\sum_{l_p \in \mathcal{M}_i^{(1)}} \sum_{l_q \in \mathcal{M}_j^{(2)}} \|l_p - l_q\|_2}{|\mathcal{M}_i^{(1)}| \times |\mathcal{M}_j^{(2)}|},$$

where $l_p \in \mathcal{M}_i^{(1)}$ and $l_q \in \mathcal{M}_j^{(2)}$ denote the (longitude, latitude) pairs of the locations and their distance can be denoted as the L_2 norm $\|l_p - l_q\|_2$ as indicated in the equation. The third extracted explicit feature for user pair $(u_i^{(1)}, u_j^{(2)})$ is based on the cosine similarity score of the user pair's binary trajectory representations in matrices $\mathbf{T}^{(1)}$ and $\mathbf{T}^{(2)}$, respectively, which can be represented as

$$cos(u_i^{(1)}, u_j^{(2)}) = \frac{\mathbf{T}^{(1)}(i,:) \cdot \mathbf{T}^{(2)}(j,:)^\top}{\left\|\mathbf{T}^{(1)}(i,:)\right\|_2 \times \left\|\mathbf{T}^{(2)}(j,:)\right\|_2}$$

Moreover, the check-in number reveals the activeness of one user and it is another important signal for inferring the anchor links across networks. Therefore, we have the last two explicit features denoting the check-in numbers of $u_i^{(1)}$ and $u_j^{(2)}$:

$$act(u_i^{(1)}) = |\mathcal{M}_i^{(1)}|, \quad act(u_j^{(2)}) = |\mathcal{M}_j^{(2)}|.$$

These five features will compose the explicit feature vector $\mathbf{x}_{\text{T}(i,j)}^{\text{EXP}}$ extracted from the trajectory data for the user pair $(u_i^{(1)}, u_j^{(2)})$.

C. Textual Word Representation Learning

Textual data is ubiquitous in online social networks and it contains a large amount of information. From the historical textual data, we can find out the language usage preference of each user. Analyzing their textual words can help us to identify the anchor links across networks. Meanwhile, convolutional neural network (CNN) achieves a big success on relevent areas and it performs better than word2vec [20] in [9]. Therefore, we will extract a set of latent and explicit features for the user pairs, where the latent feature extraction is based on the CNN.

1) Latent Feature Extraction: Convolutional neural network [13], [12], [9] is one famous neural network proposed in recent years. It is usually used in processing image data [13], [12]. With suitable convolutional kernels, CNN can also achieve an excellent performance on textual word processing [9]. When applying CNN to text data, we need to first get a suitable input. One possible method for the textual word representations utilizes the classic bag-of-word model. We will first get the statistical information of the words used by users in online social networks. By tokenizing each tip/post message into unigrams, we count the occurrence of each word, based on which we are able to transform the words of a user into a vector representation as introduced in the following part.

Formally, given the user set $\mathcal{U}$, word set $\mathcal{D}$, and user-word set $\mathcal{E}_{u,d}$ of network G, we can represent the user-word matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{D}|}$, where $\mathbf{D}(i,j)$ denotes the frequency that user $u_i \in \mathcal{U}$ uses word $d_j \in \mathcal{D}$. The occurrence frequency of all the words used by u_i can be indicated by i_{th} row in matrix $\mathbf{D}$. Each row of matrix $\mathbf{D}$ will be reshaped into a matrix of dimensions $\sqrt{|\mathcal{D}|} \times \sqrt{|\mathcal{D}|}$, and be fed as the input to the CNN where it has two convolutional layers and two max-pooling layers. For each convolutional layer, we adopt ReLU function (e.g. $\delta(\cdot)$) as the activation function. After the last pooling, we get the desired feature matrix. we flat the feature matrix first, and then use one fully-connected layer to project it to dense feature representations. Assuming flatten vector is $\mathbf{x}_i \in \mathbb{R}^m$, the output $\mathbf{z}_i$ can be calculated by

$$\mathbf{z}_i = \delta(\mathbf{W}_3 \mathbf{x}_i + \mathbf{b}_3),$$

where $\mathbf{W}_3$ and $\mathbf{b}_3$ are the weight and bias of the fully-connected layer. Therefore, we can obtain our text feature representations $\mathbf{x}_{\text{W},i}^{\text{LAT}}$ and $\mathbf{x}_{\text{W},j}^{\text{LAT}}$ for users $u_i^{(1)}$ and $u_j^{(2)}$, respectively, showing the language usage patterns of users.

2) Explicit Feature Extraction: For textual word information, we extract four explicit features. For each user pair $(u_i^{(1)}, u_j^{(2)})$, we use $\mathbf{D}^{(1)}(i,:)$, $\mathbf{D}^{(2)}(j,:)$ to denote bag-of-word vectors of $u_i^{(1)}$, $u_j^{(2)}$ in network $G^{(1)}$ and network $G^{(2)}$, respectively. Based on these two vectors, the first extracted explicit textual feature is the word vector inner product, which can be computed as:

$$IN(u_i^{(1)}, u_j^{(2)}) = \mathbf{D}^{(1)}(i,:) \cdot \mathbf{D}^{(2)}(j,:).$$

Also, the number of commonly used words of $u_i^{(1)}$ and $u_j^{(2)}$ also illustrates the similarity between users, which can be computed as:

$$CW(u_i^{(1)}, u_j^{(2)}) = |\{k | D^{(1)}(i,k) \in \mathbb{N}_+ \wedge D^{(2)}(j,k) \in \mathbb{N}_+\}|.$$

Based on $CW(u_i^{(1)}, u_j^{(2)})$, we can further extend it to capture more precise features by considering the total words used by the user pairs, which can be denoted as follows:

$$NCW_1(u_i^{(1)}, u_j^{(2)}) = \frac{CW(u_i^{(1)}, u_j^{(2)})}{|\{d|D^{(1)}(i,d) \in \mathbb{N}_+\}| + |\{d|D^{(2)}(j,d) \in \mathbb{N}_+\}|},$$

$$NCW_2(u_i^{(1)}, u_j^{(2)}) = \frac{CW(u_i^{(1)}, u_j^{(2)})}{|\{d|D^{(1)}(i,d) \in \mathbb{N}_+\} \cup \{d|D^{(2)}(j,d) \in \mathbb{N}_+\}|}.$$

We will integrate these four features to a vector $\mathbf{x}^{\text{EXP}}_{\text{W}(i,j)}$ for $(u_i^{(1)}, u_j^{(2)})$, representing the explicit features extracted from users textual word usage.

D. Deep Network Alignment Model Learning

we will introduce DETA and its learning algorithm in this part. DETA focuses on fusing those features and how to add our alignment goal into the objective function. Besides, maintaining the one-to-one constraint is another important objective in the network alignment problem. Thus, our model includes two shared representation learning networks and the joint objective function subject to the one-to-one constraint.

1) Feature Representation Fusion: In DETA, we propose to combine all features in the above sections together to get the representations of user nodes in the social networks. In order to obtain representations of user nodes in $G^{(1)}$ and $G^{(2)}$ simultaneously, we use two same representation learning networks. As indicated in Fig. 1, the two parts have shared weights, which will not only greatly reduce the variable learning costs but also effectively project the data to the same feature space.

Based on the above descriptions, we get three types of latent features extracted for user $u_i^{(1)}$ and $u_j^{(2)}$ for the heterogeneous social networks as $\mathbf{x}^{\text{LAT}}_{\text{S},i}$,$\mathbf{x}^{\text{LAT}}_{\text{T},i}$, $\mathbf{x}^{\text{LAT}}_{\text{W},i}$ and $\mathbf{x}^{\text{LAT}}_{\text{S},j}$, $\mathbf{x}^{\text{LAT}}_{\text{T},j}$, $\mathbf{x}^{\text{LAT}}_{\text{W},j}$, respectively. By concatenating these features together, we can further project the latent features of users $u_i^{(1)}$ and $u_j^{(2)}$ to more dense representations according to the following equation:

$$\begin{cases} \hat{\mathbf{x}}_i &= \sigma(\mathbf{W}_{s_1}\mathbf{x}_i + \mathbf{b}_{s_1}), \text{ where } \mathbf{x}_i = \mathbf{x}^{\text{LAT}}_{\text{S},i} \sqcup \mathbf{x}^{\text{LAT}}_{\text{T},i} \sqcup \mathbf{x}^{\text{LAT}}_{\text{W},i}, \\ \hat{\mathbf{x}}_j &= \sigma(\mathbf{W}_{s_1}\mathbf{x}_j + \mathbf{b}_{s_1}), \text{ where } \mathbf{x}_j = \mathbf{x}^{\text{LAT}}_{\text{S},j} \sqcup \mathbf{x}^{\text{LAT}}_{\text{T},j} \sqcup \mathbf{x}^{\text{LAT}}_{\text{W},j}. \end{cases}$$

In the above equation, $\mathbf{W}_{s_1}$ and $\mathbf{b}_{s_1}$ are the shared weight and bias in this layer. Meanwhile, to effectively integrate the latent and explicit features together, we adopt a deep fusion layer as follows:

$$\begin{cases} \mathbf{x}_{i,j} &= \mathbf{x}^{\text{LAT}}_{i,j} \sqcup \mathbf{x}^{\text{EXP}}_{i,j}, \\ \mathbf{x}^{\text{LAT}}_{i,j} &= \sigma\left(\mathbf{W}_{s_2}(\hat{\mathbf{x}}_i \sqcup \hat{\mathbf{x}}_j) + \mathbf{b}_{s_2}\right), \\ \mathbf{x}^{\text{EXP}}_{i,j} &= \mathbf{x}^{\text{EXP}}_{\text{S}(i,j)} \sqcup \mathbf{x}^{\text{EXP}}_{\text{T}(i,j)} \sqcup \mathbf{x}^{\text{EXP}}_{\text{W}(i,j)}, \end{cases}$$

where $\mathbf{W}_{s_2}$ and $\mathbf{b}_{s_2}$ denote the weight and bias in the fusion layer and the $\mathbf{x}^{\text{EXP}}_{i,j}$ represents the concatenation of all explicit features between user pair $(u_i^{(1)}, u_j^{(2)})$.

2) Joint Objective Function: According to the representation fusion step aforementioned, we can represent the complete feature vectors of all the potential anchor links in set $\mathcal{C}$ as $\mathbf{X} \in \mathbb{R}^{|\mathcal{C}| \times d'}$ (where d' denotes the fused feature vector length). Formally, for all the anchor links in set $\mathcal{C}$, we can denote their labels as vector $\mathbf{y} \in \{0,1\}^{|\mathcal{C}|}$. Based on the fused feature representations of the anchor link $(u_i^{(1)}, u_j^{(2)})$ (i.e. $\mathbf{X}(i,j)$), we can represent the introduced loss term on all the links in $\mathcal{C}$ as follows:

$$\mathcal{L}_{\text{DETA}} = ||\mathbf{X}\mathbf{r} - \mathbf{y}||_2^2.$$

where $\mathbf{r}$ denotes the mapping function for transforming $\mathbf{X}$ to $\mathbf{y}$. Considering the *one-to-one* cardinality constraint, which can be modeled as a constraint on node degrees. Formally, we can represent the joint objective function of the DETA as

$$\begin{aligned} \min_{\mathbf{r},\mathbf{y}} \quad & \frac{1}{2} \cdot ||\mathbf{r}||_2^2 + \frac{\gamma}{2} \cdot ||\mathbf{X}\mathbf{r} - \mathbf{y}||_2^2 \\ s.t. \quad & \mathbf{y} \in \{0,1\}^{|\mathcal{C}|},\ y_{i,j} = 1, \forall (u_i^{(1)}, u_j^{(2)}) \in \mathcal{A}^{(1,2)}, \\ & 0 \le \sum_{u_j^{(2)} \in G^{(2)}} y_{i,j} \le 1, \forall u_i^{(1)} \in G^{(1)}, \\ & 0 \le \sum_{u_i^{(1)} \in G^{(1)}} y_{i,j} \le 1, \forall u_j^{(2)} \in G^{(2)}, \end{aligned}$$

where $||\mathbf{r}||_2^2$ denotes the regularization term on the model variable. Parameters γ is the scalar used to adjust the weight of the regularization term. The objective function is a combinatorial optimization problem, which is an NP-hard problem [28]. Since this problem cannot be trained by the existing learning algorithms directly, we will introduce the learning algorithm to solve the problem in the following section.

3) Optimization Algorithm: In this part, we will introduce a feasible optimization algorithm to solve the objective function. **Representation Learning Component Pre-Training**: Instead of learning the complete model DETA together, we propose to pre-train each representation learning component prior to the prediction component. Formally, according to the descriptions of the three feature representation learning components, we propose to feed their learned feature representations to an output layer with two neurons representing the $+1/0$ class labels, respectively. For instance, given the learned representations from the historical trajectory data of user pair $(u_i^{(1)}, u_j^{(2)})$, i.e., $\mathbf{x}^{\text{LAT}}_{\text{T},i}$, $\mathbf{x}^{\text{LAT}}_{\text{T},j}$ and $\mathbf{x}^{\text{EXP}}_{\text{T}(i,j)}$, we can denote the inferred labels as

$$\bar{\mathbf{y}}_{i,j} = \sigma(\mathbf{W}_{pre}(\mathbf{x}^{\text{LAT}}_{\text{T},i} \sqcup \mathbf{x}^{\text{LAT}}_{\text{T},j} \sqcup \mathbf{x}^{\text{EXP}}_{\text{T}(i,j)}) + \mathbf{b}_{pre}).$$

Compared against the ground truth label of the links, we can represent the complete pre-training loss on the training set $\mathcal{T}_{train} \subset \mathcal{L}$ as:

$$\mathcal{L}_{pre} = -\sum_{(u_i^{(1)}, u_j^{(2)}) \in \mathcal{T}_{train}} \sum_p y_{i,j}(p) \log \bar{y}_{i,j}(p).$$

TABLE I
INFERRING ANCHOR LINK RESULT (PARAMETER λ CHANGES IN $\{1, 2, \cdots, 10\}$, $\beta = 0.01, \eta = 1, \theta = 1000, \gamma = 0.01$).

		Negative/Positive Ratio λ									
metric	method	1	2	3	4	5	6	7	8	9	10
Accuracy	DETA	**0.924±0.016**	**0.904±0.008**	**0.901±0.009**	**0.903±0.008**	**0.902±0.008**	**0.919±0.004**	**0.954±0.020**	**0.965±0.003**	**0.965±0.004**	**0.967±0.003**
	DETA_NO	0.834±0.021	0.865±0.014	0.891±0.004	0.879±0.010	0.897±0.009	0.894±0.022	0.904± 0.021	0.926± 0.009	0.933± 0.007	0.933± 0.013
	PALE	0.517±0.022	0.370±0.016	0.316±0.012	0.290±0.007	0.281±0.011	0.280±0.007	0.285±0.001	0.294±0.005	0.305±0.004	0.315±0.008
	D+T+W	0.547±0.028	0.670±0.017	0.747±0.009	0.678±0.209	0.690±0.261	0.713±0.258	0.722±0.297	0.629±0.340	0.577±0.337	0.662pm0.373
	DEEPWALK	0.487±0.018	0.667±0.013	0750±0.009	0.800±0.009	0.833±0.008	0.857±0.005	0.875±0.005	0.889±0.005	0.900±0.003	0.910±0.005
	WORD2VEC	0.484±0.013	0.667±0.016	0.750±0.009	0.800±0.009	0.833±0.007	0.857±0.005	0.875±0.005	0.889±0.005	0.900±0.003	0.909±0.005
	DIME-SH(social)	0.777±0.018	0.819±0.016	0.849±0.009	0.870±0.014	0.885±0.011	0.896±0.009	0.908±0.008	0.906±0.013	0.916±0.012	0.914±0.009
	TULER(spatial)	0.663±0.051	0.749±0.019	0.768±0.015	0.811±0.012	0.838±0.008	0.858±0.005	0.875±0.005	0.889±0.005	0.900±0.004	0.909±0.005
	CNN(text)	0.572±0.029	0.669±0.014	0.750±0.009	0.800±0.009	0.833±0.007	0.857±0.005	0.875±0.005	0.889±0.005	0.900±0.003	0.909±0.005
	MNA-Social	0.791±0.017	0.844±0.013	0.873±0.014	0.893±0.014	0.908±0.007	0.918±0.010	0.927±0.006	0.935±0.003	0.939±0.004	0.944±0.002
	MNA-Spatial	0.695±0.021	0.781±0.014	0.826±0.009	0.854±0.009	0.874±0.007	0.887±0.004	0.898±0.005	0.908±0.007	0.916±0.003	0.923±0.003
	MNA-Text	0.585±0.021	0.680±0.012	0.754±0.012	0.802±0.010	0.833±0.009	0.857±0.006	0.875±0.005	0.889±0.005	0.900±0.003	0.909±0.005
F1	DETA	**0.928±0.016**	**0.870±0.009**	**0.829±0.014**	**0.797±0.014**	**0.760±0.024**	**0.743±0.008**	**0.813±0.050**	**0.824±0.015**	**0.798±0.024**	**0.791±0.025**
	DETA_NO	0.816±0.025	0.768±0.029	0.752±0.010	0.602±0.035	0.590±0.057	0.421±0.244	0.381±0.254	0.542±0.125	0.532±0.132	0.431±0.236
	PALE	0.667±0.020	0.498±0.019	0.402±0.014	0.334±0.012	0.283±0.012	0.247±0.010	0.223±0.009	0.200±0.012	0.182±0.007	0.166±0.011
	D+T+W	0.518±0.028	0.166±0.042	0.075±0.065	0.114±0.136	0.106±0.116	0.079±0.103	0.065±0.084	0.087±0.094	0.113±0.093	0.062±0.074
	DEEPWALK	0.486±0.032	0.002±0.003	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	WORD2VEC	0.326±0.326	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	DIME-SH(social)	0.743±0.020	0.651±0.040	0.593±0.038	0.538±0.066	0.495±0.066	0.446±0.070	0.429±0.106	0.258±0.191	0.262±0.181	0.102±0.133
	TULER(spatial)	0.547±0.153	0.430±0.048	0.141±0.109	0.109±0.091	0.061±0.060	0.022±0.033	0.007±0.009	0.010±0.014	0.029±0.037	0.007±0.011
	CNN(text)	0.591±0.028	0.012±0.021	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	MNA-Social	0.757±0.021	0.729±0.017	0.707±0.027	0.690±0.036	0.676±0.014	0.665±0.028	0.652±0.026	0.648±0.020	0.633±0.028	0.633±0.024
	MNA-Spatial	0.592±0.028	0.560±0.027	0.529±0.026	0.506±0.028	0.481±0.027	0.446±0.024	0.426±0.024	0.411±0.036	0.395±0.043	0.379±0.035
	MNA-Text	0.517±0.050	0.252±0.032	0.172±0.038	0.126±0.034	0.092±0.039	0.044±0.025	0.032±0.013	0.024±0.015	0.017±0.010	0.007±0.008
Precision	DETA	0.877±0.019	0.793±0.014	0.729±0.018	0.685±0.022	0.643±0.022	0.632±0.014	0.893±0.138	**0.965±0.010**	**0.963±0.010**	**0.962±0.037**
	DETA_NO	**0.913±0.017**	0.891±0.013	0.867±0.019	0.877±0.035	0.874±0.023	0.901±0.069	0.912±0.072	0.866±0.054	0.860±0.051	0.883±0.087
	PALE	0.509±0.021	0.339±0.017	0.258±0.011	0.206±0.009	0.170±0.009	0.145±0.007	0.129±0.006	0.115±0.007	0.103±0.004	0.094±0.007
	D+T+W	0.555±0.030	0.524±0.035	0.459±0.118	0.180±0.167	0.358±0.295	0.092±0.110	0.099±0.108	0.080±0.102	0.070±0.060	0.097±0.148
	DEEPWALK	0.488±0.014	0.350±0.450	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	WORD2VEC	0.242±0.243	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	DIME-SH(social)	0.873±0.024	**0.903±0.021**	0.907±0.016	**0.924±0.038**	**0.924±0.038**	**0.941±0.028**	**0.919±0.042**	0.648±0.426	0.765±0.384	0.477±0.478
	TULER(spatial)	0.784±0.127	0.886±0.028	**0.933±0.069**	0.834±0.288	0.793±0.285	0.307±0.341	0.330±0.424	0.386±0.474	0.387±0.424	0.425±0.448
	CNN(text)	0.572±0.041	0.390±0.478	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	MNA-Social	0.899±0.019	0.867±0.039	0.836±0.042	0.831±0.062	0.822±0.057	0.816±0.077	0.809±0.069	0.818±0.057	0.811±0.077	0.794±0.057
	MNA-Spatial	0.893±0.024	0.850±0.017	0.814±0.020	0.783±0.038	0.768±0.039	0.740±0.032	0.726±0.047	0.715±0.047	0.712±0.040	0.703±0.035
	MNA-Text	0.617±0.034	0.573±0.057	0.553±0.073	0.549±0.106	0.517±0.095	0.477±0.159	0.623±0.219	0.586±0.235	0.544±0.181	0.415±0.390
Recall	DETA	**0.986±0.016**	**0.965±0.014**	**0.961±0.021**	**0.954±0.014**	**0.930±0.035**	**0.901±0.012**	0.764±0.050	0.719±0.025	0.681±0.033	0.673±0.037
	DETA_NO	0.738±0.038	0.676±0.044	0.665±0.023	0.460±0.039	0.448±0.064	0.314±0.190	0.284±0.204	0.409±0.111	0.401±0.113	0.328±0.192
	PALE	0.966±0.009	0.937±0.012	0.922±0.013	0.891±0.014	0.854±0.020	0.829±0.022	**0.822±0.017**	**0.796±0.024**	**0.774±0.022**	**0.752±0.024**
	D+T+W	0.488±0.036	0.100±0.030	0.044±0.045	0.225±0.355	0.240±0.384	0.214±0.364	0.212±0.394	0.348±0.442	0.439±0.422	0.307±0.453
	DEEPWALK	0.489±0.072	0.001±0.001	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	WORD2VEC	0.500±0.500	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	DIME-SH(social)	0.649±0.034	0.511±0.046	0.442±0.045	0.384±0.065	0.342±0.063	0.296±0.062	0.288±0.091	0.166±0.127	0.166±0.120	0.060±0.082
	TULER(spatial)	0.433±0.132	0.286±0.041	0.081±0.070	0.061±0.054	0.033±0.034	0.012±0.017	0.003±0.005	0.005±0.007	0.016±0.020	0.004±0.005
	CNN(text)	0.626±0.116	0.006±0.011	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	MNA-Social	0.656±0.034	0.630±0.029	0.615±0.037	0.592±0.041	0.577±0.030	0.567±0.043	0.554±0.059	0.540±0.042	0.526±0.055	0.530±0.042
	MNA-Spatial	0.444±0.031	0.418±0.028	0.392±0.027	0.375±0.027	0.350±0.025	0.319±0.024	0.302±0.022	0.289±0.032	0.275±0.037	0.260±0.030
	MNA-Text	0.451±0.071	0.163±0.025	0.103±0.026	0.072±0.022	0.052±0.024	0.024±0.014	0.016±0.007	0.012±0.008	0.008±0.005	0.003±0.004

Here, we need to remark that, the pre-training loss term $\mathcal{L}_{pre}$ will be used for learning the representation learning components on both trajectory data and textual data. Meanwhile, for the social connection representation learning part, we merge $\mathcal{L}_{SNE}$ into the $\mathcal{L}_{pre}$. The pre-training can be accomplished by Adam optimization algorithm, and the learned representations will be used for the prediction component.

Prediction Component Training: Since the joint objective function is NP-hard, such an objective function can be effectively learned with an alternative variable updating algorithm.

- *Fix* $\mathbf{y}$*, Update* $\mathbf{r}$: With $\mathbf{y}$ fixed, we can represent the objective function as follows:

$$\min_{\mathbf{r}} \frac{1}{2}||\mathbf{r}||_2^2 + \frac{\gamma}{2}||\mathbf{X}\mathbf{r} - \mathbf{y}||_2^2.$$

  The optimal $\mathbf{r}^*$ which can minimize the objective function can be represented as

$$\mathbf{r}^* = \gamma(\mathbf{I} + \gamma\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y},$$

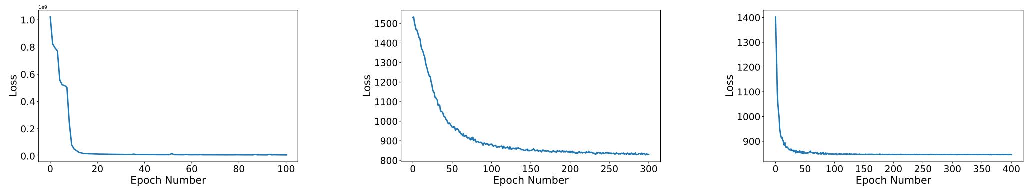
  where entries of the labeled anchor links in vector $\mathbf{y}$ will be assigned with their labels, while the remaining entries are initialized with an unbiased value $\frac{1}{2}$.

- *Fix* $\mathbf{r}$*, Update* $\mathbf{y}$: With $\mathbf{r}$ fixed, we can represent the objective function as follows:

$$\min_{\mathbf{y}} \frac{\gamma}{2}||\hat{\mathbf{y}} - \mathbf{y}||_2^2,$$

  where variable $\mathbf{y}$ is subject to both the binary and the one-to-one constraint as indicated in the original objective function and the scores for being anchor links $\hat{\mathbf{y}} = \mathbf{r}^*\mathbf{X}$. The objective function can be addressed with the greedy link selection algorithm proposed in [28]. For the unknown links, we sort them according to their scores in $\hat{\mathbf{y}}$ and set the elements in $\mathbf{y}$ whose positions corresponding to those with the largest scores label $+1$, if the assignments of the labels will not violate the *one-to-one* cardinality constraint. Such a greedy link selection algorithm can achieve an $\frac{1}{2}$-approximation of the optimal solution.

Such an iterative learning process continues until convergence, and the final prediction labels of all the unknown anchor links in vector $\mathbf{y}$ will be outputted as the results.



(a) Social Network Convergence (b) Trajectory Convergence (c) Text Convergence

Fig. 3. Convergence Analysis

IV. Experiments

To evaluate the performance of the proposed framework, extensive experiments will be done on two real-world datasets. In the following part, we will first talk about the experimental settings, and then discuss the experimental results in detail.

A. Experimental Settings

1) Experimental Setup: The datasets used in the experiments include a pair of aligned social networks: Foursquare and Twitter. There are 3282 anchor links between Foursquare and Twitter. The detailed descriptions of the datasets are available in [11]. We utilize the anchor links as positive links as well as a set of non-anchor links as negative links first, and then extract subset of negative links based on negative/positive ratios. To eliminate variables caused by different training set, we use 10-fold cross validation to partition the links into two subsets according to ratio 9:1, where the 9 folds are treated as training set and the rest 1 fold is treated as testing set.

2) Comparison Methods:

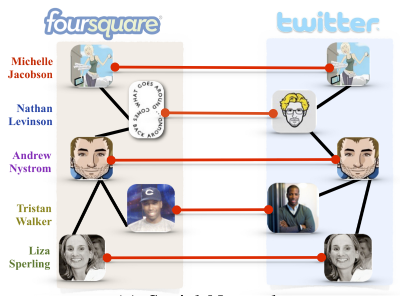
- **Deep Network Aligment** (DETA): DETA can capture the feature representations of the social connections, trajectory and textual words and then do the label inference with the three representations and the onee-to-one constraint.
- **DETA without one-to-one constraint** (DETA_NO): DETA_NO is a variant of the DETA model, which excludes the one-to-one constraint in the label prediction step.
- **DeepWalk**: The DeepWalk [21] uses local information obtained from truncated random walks to learn latent feature representation based on social connections. The parameter settings of DeepWalk involve window size: 5, negative sample: 5, and feature dimension: 64.
- **Word2Vec**: Word2Vec is first proposed in [20] for natural language processing. Based on users' textual data, we use a Google's pre-trained Word2Vec model [1] to get the vector representations of words directly and regard the concatenation of used words' vectors as the user's representation.
- **Single Data Type based Alignment Models**: We also compare DETA with three single data type representation learning models, including social connection representation DIME-SH(social) [29], trajectory representation TULER(spatial) [5] and textual word representation CNN(text) [9], which are all slightly extended by incorporating a small number of explicit features.
- **Explicit Feature based Alignment Models**: Several existing explicit feature based network alignment models, i.e., explicit social features MNA-Social [11], explicit trajectory features MNA-Spatial [11] and explicit textual features MNA-Text [11], are also compared in the experiments.
- **DeepWalk+ TULER + Word2Vec (D+T+W)**: This method uses the combination of DeepWalk for social connections, LSTM for trajectory and Word2Vec for textual words. Here, we adopt the same settings for as indicated in the previous descriptions. We take the combined features of the three types and put it into a SVM classifier.
- **PALE**: The method is proposed in [17], which adds additional links to complement original network according to known anchor links. In our paper, we only use the linear mapping function.

3) Evaluation Metrics: In this paper, we will use accuracy, precision, recall and F1 to measure the experimental results.

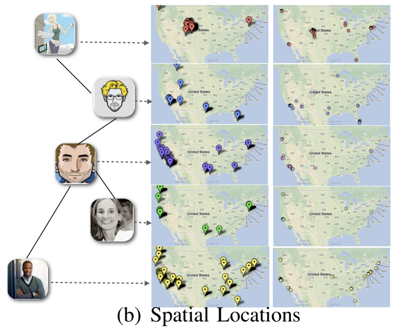
B. Experimental Results

1) Experimental Results Analysis: To demostrate the results more comprehensively, we further have the imbalanced ratio to test DETA. The imbalanced ratio (i.e., λ) denotes the ratio of $\frac{\#\text{negative cases}}{\#\text{positive cases}}$. We set $\lambda \in \{1, 2, \cdots, 10\}$. Besides, we set $\beta = 0.01$, $\gamma = 0.01$, $\eta = 1$, $\theta = 1000$ mentioned in the section III. Now, we can get the result table I. We use four sections to denote four metrics respectively. In each section, each row represents the method and each column represents the negative/positive ratio. Compared with DETA_NO, DETA has a better performance with these four metrics, which demostrates the effectiveness of the one-to-one cardinality constraint. The performances of DETA and DETA_NO are very close on precision, although DETA_NO has a better performance when $\lambda \in \{1, \cdots, 7\}$. DETA also outperforms the PALE on four metrics as well. We notice that PALE outperforms DETA on recall when $\lambda \in \{7, 8, 9, 10\}$. The reason is that the PALE always predicts the positive instances and the recall measures the number of predicted positive instances. Hence, the recall becomes higher when λ rises. DETA also outperforms the hybrid D+T+W, showing the effectiveness of our learned features works better than simply adopting the combination of existing works. In addition, we also show the results by utilizing one of three kinds of

[1]https://code.google.com/archive/p/word2vec/



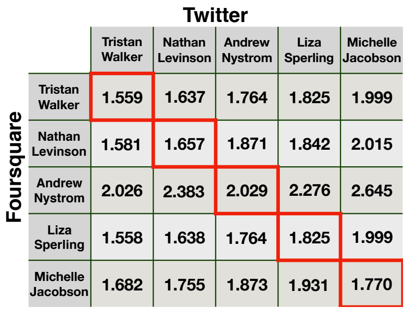
(a) Social Network


(b) Spatial Locations

user	word counts in both networks (Twitter, Foursquare)
Michelle Jacobson	art (65,2), style (16,3) audit (3,2), grill (19,2)
Nathan Levinson	happy (27,5), enjoy (9,4) week (18,4), shows (6,6)
Andrew Nystrom	awsm (2,3), kids (20,3) red (61,3), open (11,4)
Liza Sperling	ask (6,5), coffee (8,3) mochi (1,3), hangout (5,2)
Tristan Walker	win (19,4), amazing (55,5) awesome (51,4), please (9,4)

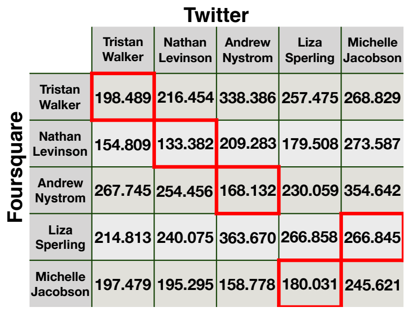
(c) Textual Word Usage

Fig. 4. Case study [11]: five real-world users with their social, spatial and text distributions.



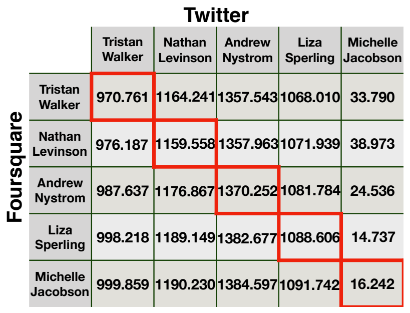
Foursquare \ Twitter	Tristan Walker	Nathan Levinson	Andrew Nystrom	Liza Sperling	Michelle Jacobson
Tristan Walker	1.559	1.637	1.764	1.825	1.999
Nathan Levinson	1.581	1.657	1.871	1.842	2.015
Andrew Nystrom	2.026	2.383	2.029	2.276	2.645
Liza Sperling	1.558	1.638	1.764	1.825	1.999
Michelle Jacobson	1.682	1.755	1.873	1.931	1.770

(a) Latent Social Feature based Distance Matrix



Foursquare \ Twitter	Tristan Walker	Nathan Levinson	Andrew Nystrom	Liza Sperling	Michelle Jacobson
Tristan Walker	198.489	216.454	338.386	257.475	268.829
Nathan Levinson	154.809	133.382	209.283	179.508	273.587
Andrew Nystrom	267.745	254.456	168.132	230.059	354.642
Liza Sperling	214.813	240.075	363.670	266.858	266.845
Michelle Jacobson	197.479	195.295	158.778	180.031	245.621

(b) Latent Trajectory Feature based Distance Matrix



Foursquare \ Twitter	Tristan Walker	Nathan Levinson	Andrew Nystrom	Liza Sperling	Michelle Jacobson
Tristan Walker	970.761	1164.241	1357.543	1068.010	33.790
Nathan Levinson	976.187	1159.558	1357.963	1071.939	38.973
Andrew Nystrom	987.637	1176.867	1370.252	1081.784	24.536
Liza Sperling	998.218	1189.149	1382.677	1088.606	14.737
Michelle Jacobson	999.859	1190.230	1384.597	1091.742	16.242

(c) Latent Text Feature based Distance Matrix

Fig. 5. Case Study Matrices

features. Generally, DETA shows the best performance among DIME-SH(social), CNN(text), TULER(spatial), MNA-Social, MNA-Spatial and MNA-Text. However, the precision scores of DETA are a little bit lower when $\lambda \in \{1, \cdots, 6\}$. It might be that the one-to-one cardinality constraint will assign each user in network 1 with exact one user in network 2. Thus, it would eliminate those truely positive pairs that they provide less information. When the ratio higher than 7, the information for the negative pairs get comprehensive, thus DETA has the better results.

Moreover, we can also illustrate that the importance among three types of features. From table I, compared with TULER(spatial) and the CNN(text), DIME-SH(social) have the best performance among all metrics when the ratio changes from 1 to 10. It demostrates the social connections are the biggest contributor for alignment task compared with trajectories and texts. This is because the checkins are not a must-do in twitter and the texts online is in informal usages. Besides, textural features work when the imbalance ratio λ is not high. By combining these three types of heterogenous information, the DETA achieves a balance among these metrics despite it does not rank first in every measurement.

2) Convergence Analysis: Before showing the experimental results, we will analyze how many epochs are needed to achieve results. In Fig 3, we show the loss introduced in three modules respectively. Fig. 3(a), Fig. 3(b) and Fig 3(c) show the pre-training losses of social connection, trajectory, textual words modules respectively. As indicated in Fig. 3, the x-axis denotes the epoch number and the y-axis represents the corresponding loss. According to the results, each module can converge within a small epoch number, which means DETA can converge in a short time.

3) Case Studies on Latent Features: We further examine the effectiveness of our latent features with a case study since the explict features are based on pairwise caculations. In Figure 4, we show a case of five real-world users who have both Foursquare and Twitter accounts. These five users are socially connected in both networks, as shown in the Figure 4(a). The check-ins of five users are illustrated in the Figure 4(b) and their word usage patterns are provided in the Figure 4(c). Based on these observations, we extract their latent features by three approaches, i.e., DIME-SH(social), TULER(spatial) and CNN(text). To evaluate the effectiveness of latent features, we propose to use their pairwise distances based on these different latent feature representations, aiming to check whether these learned features can help align these users correctly or not. By applying the one-to-one cardinality constraint on each approach, the results are illustrated in the matrices shown in Figure 5, where the rows and columns denote users from Twitter and Foursquare and the matrix entries indicate their pairwise distances. The results demostrate the latent features can predict anchor links alomost correctly.

V. Related Work

Network alignment is a significant research problem, which has been studied in various areas, e.g., protein-protein-interaction network alignment in bioinformatics [14], [22], chemical compound matching in chemistry [23], data schemas matching [19], graph matching in combinatorial mathematics [18], and figure matching and merging in computer vision [1]. For online social media, network alignment provides an effective way for information fusion across multiple social sites. Many research works manually extract features by heuristic methods on different networks [11], [30]. Zhang et al. ustilize transitivity law to align multiple anonmized social networks [30] solely on social network connections. [31] uses the social connections, checkins and posts together tp predict anchor links. Recently, with the great property of network embedding, researchers starts to take advantages of the network embedding on the network alignment task. Matrix factorization based methods([32], [10]) can be one type of network embedding methods to solve network alignment problem, where the matrix can be formed by both homogeneous networks and heterogeneous networks. Man et al. propose to supplement each other networks via anchor links and factorize the matrices on the complementary networks in [17].

In recent years, many research works propose to apply deep learning models on graph-structured data, i.e., the network embedding problems. Network embedding aims at projecting a graph-structured data to feature vector representations. In graphs, the relation can be treated as a translation of the entities, and many translation based embedding models have been proposed, like TransE [2], TransH [27], and TransR [16]. In recent years, many network embedding works based on random walk and deep learning models have been introduced, like Deepwalk [21], LINE [24], node2vec [6], and DNE [26]. Perozzi et al. introduce the Deepwalk algorithm [21]. Chang et al. [3] learn the embedding of networks involving text and image information. Chen et al. [4] introduce a task guided embedding model to learn the representations for the author identification problem. However, most of these embedding models are proposed for homogeneous networks and assume the learned feature vectors can be appropriate to all external tasks. Meanwhile, by this context so far, few research works have been done on studying heterogeneous social network alignment problem with deep learning models yet.

VI. Conclusion

In this paper, we study the heterogeneous social network alignment problem with a deep learning framework, namely DETA. For each information category in the social networks, DETA extracts a small number of explicit features and employs deep learning models to learn a set of latent features for the anchor links. These extracted features will be further integrated together for the anchor link label inference with a deep fusion. Extensive experiments have been done on two real-world heterogeneous social networks, and the experimental results have demonstrated the effectiveness of the proposed deep social network alignment model.

VII. Acknowledgement

This work is partially supported by NSF through grant IIS-1763365 and by FSU. This work is also supported in part by NSF under grants III-1526499, III-1763325, III-1909323, CNS-1930941, and CNS-1626432.

References

[1] M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. In *ICDM*, 2009.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*. 2013.

[3] S. Chang, W. Han, J. Tang, G. Qi, C. Aggarwal, and T. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, 2015.

[4] T. Chen and Y. Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. *CoRR*, abs/1612.02814, 2016.

[5] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. Identifying human mobility via trajectory embeddings. In *IJCAI*, 2017.

[6] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[7] M. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM*, 2006.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.

[9] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[10] Giorgos Kollias, Shahin Mohammadi, and Ananth Grama. Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *TKDE*, 24(12):2232–2243, 2011.

[11] X. Kong, J. Zhang, and P. Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, 2013.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, 2001.

[14] C. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 2009.

[15] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.

[16] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[17] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. Predict anchor links across social networks via an embedding approach. In *IJCAI*, 2016.

[18] F. Manne and M. Halappanavar. New effective multithreaded matching algorithms. In *IPDPS*, 2014.

[19] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[22] R. Singh, J. Xu, and B. Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *RECOMB*, 2007.

[23] A. Smalter, J. Huan, and G. Lushington. Gpm: A graph pattern matching kernel with diffusion for chemical compound classification. In *IEEE BIBE*, 2008.

[24] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

[25] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

[26] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *KDD*, 2016.

[27] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[28] J. Zhang, J. Chen, J. Zhu, Y. Chang, and P. Yu. Link prediction with cardinality constraint. In *WSDM*, 2017.

[29] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and P. S. Yu. Bl-mne: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *ICDM'17*, 2017.

[30] J. Zhang and P. Yu. Multiple anonymized social networks alignment. In *ICDM*, 2015.

[31] J. Zhang and P. Yu. Pct: Partial co-alignment of social networks. In *WWW*, 2016.

[32] Si Zhang and Hanghang Tong. Final: Fast attributed network alignment. In *KDD*, 2016.