Meta Diagram based Active Social Networks Alignment

Yuxiang Ren
IFM Lab
Florida State University
Tallahassee, USA
yuxiang@ifmlab.org

Charu C. Aggarwal
IBM Research AI
New York, USA
charu@us.ibm.com

Jiawei Zhang
IFM Lab
Florida State University
Tallahassee, USA
jiawei@ifmlab.org

***Abstract*—Network alignment aims at inferring a set of anchor links matching the shared entities between different information networks, which has become a prerequisite step for effective fusion of multiple information networks. In this paper, we will study the network alignment problem to fuse online social networks specifically. Social network alignment is extremely challenging to address due to several reasons, i.e., *lack of training data*, *network heterogeneity* and *one-to-one constraint*. Existing network alignment works usually require a large number of training instances, but such a demand can hardly be met in applications, as manual anchor link labeling is extremely expensive. Significantly different from other homogeneous network alignment works, information in online social networks is usually of heterogeneous categories, the incorporation of which in model building is not an easy task. Furthermore, the *one-to-one* cardinality constraint on anchor links renders their inference process intertwistingly correlated. To resolve these three challenges, a novel network alignment model, namely *ActiveIter*(Active Iterative Alignment), is introduced in this paper. The model *ActiveIter* defines a set of *inter-network meta diagrams* for anchor link feature extraction, adopts *active learning* for effective label query and uses *greedy link selection* for anchor link cardinality filtering. Extensive experiments were performed on a real-world aligned networks dataset, and the experimental results have demonstrated the effectiveness of *ActiveIter* compared with other state-of-the-art baseline methods.**

***Index Terms*—Heterogeneous Network, Network Alignment, Active Learning, Data Mining**

I. Introduction

Formally, the network alignment problem [4], [26] denotes the task of inferring the set of anchor links [7] between the shared information entities in different networks, where the anchor links are usually assumed to be subject to the *one-to-one* cardinality constraint [21]. Network alignment has concrete applications in the real world, which can be applied to discover the set of shared users between different online social networks [7], [26], identify the common protein molecules between different protein-protein-interaction (PPI) networks [4], [15], [16], and find the mappings of POIs (points of interest) across different traffic networks [26]. In this paper, we will use online social networks as an example of a real world setting of the network alignment problem and also use this setting to elucidate the proposed model.

Online social networks usually have very complex structures, involving different categories of nodes and links. For instance, in online social networks, like Twitter and Foursquare as shown in Figure 1, users can perform various kinds of social activities, e.g., following other users, writing posts. Viewed in such a perspective, their network structures will contain multiple types of nodes and links, i.e., "User", "Post" (node types), and "Follow", "Write" (link types). Users' personal preference may steer their online social activities, and the network structure can provide insightful information for differentiating users between networks. Furthermore, the nodes in online social networks can be also attached with various types of attributes. For example, the written post nodes can contain words, location check-ins and timestamps (attribute types), which will provide complementary information for inferring users' language usage, spatial and temporal activity patterns respectively. Based on such an intuition, both the network structure and attribute information should be incorporated in the network alignment model building.

Most of the existing network alignment models are based on supervised learning [7], which aim at building classification/regression models with a large set of pre-labeled anchor links to infer the remaining unlabeled ones (where the existing and non-existing anchor links are labeled as the positive and negative instance respectively). For the network alignment task, pre-labeled anchor links can provide necessary information for understanding the patterns of aligned user pairs in their information distribution, especially compared with the unsupervised alignment models [4], [26]. However, for the real-world online social networks, cross-network anchor link labeling is not an easy task, since it requires tedious user-account pairing and manual user-background checking, which can be very time-consuming and expensive. Therefore, a large training data set as required by existing network alignment models [7] is rarely available in the real world.

Problem Studied: In this paper, we propose to study the heterogeneous network alignment problem based on the active learning setting, which is formally referred to the *Active heterogeNeous Network Alignment* (ANNA) problem. Subject to the pre-specified query budget (i.e., the label query times), ANNA allows the models to selectively query for extra labels of the unlabeled anchor links in the learning process. In Figure 1, we shown an example of the ANNA problem between the Foursquare and Twitter social networks.

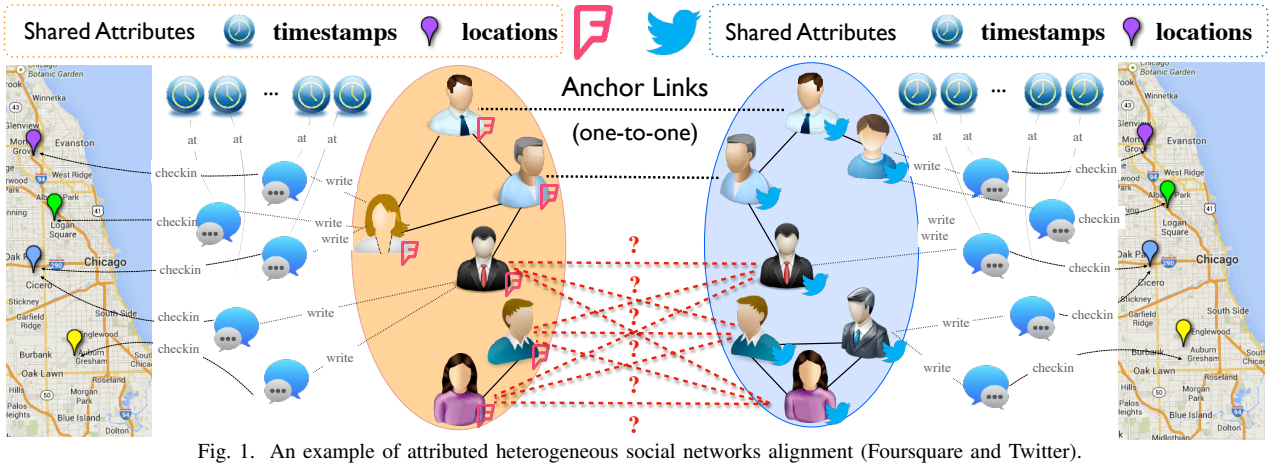


Fig. 1. An example of attributed heterogeneous social networks alignment (Foursquare and Twitter).

The current research has not studied the heterogeneous network alignment problem based on active learning setting yet. The ANNA problem is a novel yet difficult task, and the challenges mainly come from three perspectives, e.g., *network heterogeneity*, *lack of training data*, and *one-to-one constraint*.

- *Network Heterogeneity*: According to the aforementioned descriptions, both the complex network structure and the diverse attributes have concrete physical meanings and can be useful for the social network alignment task. To incorporate such heterogeneous information in model building, a unified approach is required to handle the network structure and attribute information in a unified analytic.
- *Paucity of Training Data*: To overcome problems caused by paucity of training data, besides the labeled anchor links, active learning also allows models to query for extra labels of unlabeled instances. In this context, active learning application in network alignment still remains unexplored.
- *One-to-One Cardinality Constraint*: Last but not the least, the anchor links to be inferred are not independent in the networked data scenario. The *one-to-one* cardinality constraint on anchor links will limit the number of anchor links incident to the user nodes [7], [21], which renders the information of positive and negative anchor links to be imbalanced. For each user, if one incident anchor link is identified to be positive, the remaining incident anchor links will all be negative by default. Viewed from such a perspective, positive anchor links contribute far more information compared with the negative ones. Effectively maintaining and utilizing such a constraint on anchor links in the active label query and model building is a challenging problem.

To address these challenges, we will introduce a new network alignment model, namely *Active Iterative Alignment* (*ActiveIter*), in this paper. To model the diverse information available in social networks, *ActiveIter* adopts the *attributed heterogeneous social network* concept to represent the complex network structure and the diverse attributes on nodes and links. Furthermore, a unified feature extraction method will be introduced in *ActiveIter*, based on a novel concept namely *meta diagram*, for anchor links between attributed heterogeneous social networks. *ActiveIter* accepts coupled user pairs as the input, and outputs the inference results of the anchor links between them utilizing information about both the labeled and unlabeled anchor links. To deal with the paucity of training data, active learning will be adopted in *ActiveIter* to utilize the unlabeled anchor links in model building by querying for extra anchor link labels based on a designated stratedy within certain pre-specified query budget. Due to the *one-to-one* constraint, the unlabeled anchor links no longer bears equal information, and querying for labels of potential positive anchor links will be more "informative" compared with negative anchor links. Among the unlabeled links, *ActiveIter* aims at selecting a set of mis-classified false-negative anchor links as the potential candidates. Using such an approach contributes to not only these queried labels but also other potential extra label corrections of the conflicting negative links. An innovative query strategy is proposed to make sure that *ActiveIter* can select mis-classified false-negative anchor links more precisely. *ActiveIter* can outperform other non-active models with less than 10% of extra training instances which has the additional benefits of reducing the time and space complexity.

The remaining parts of this paper will be organized as follows. In Section II, we will introduce the definitions of several important terminologies and the formal problem statement. Detailed information about the proposed model will be provided in Section III, whose effectiveness and efficiency will be tested in Section IV. Related works will be talked about in Section V, and finally in Section VI we will conclude this paper.

II. CONCEPT AND PROBLEM DEFINITION

In this section, we will define several important concepts used in this paper, and provide the formulation of the ANNA problem.

A. Terminology Definition

Definition 1 (Attributed Heterogeneous Social Networks): The *attributed heterogeneous social network* studied in this paper can be represented as $G = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \bigcup_i \mathcal{V}_i$ and $\mathcal{E} = \bigcup_i \mathcal{E}_i$ represent the sets of diverse nodes and complex links in the network. The set of attributes associated with nodes

in $\mathcal{V}$ can be represented as set $\mathcal{T} = \bigcup_i \mathcal{T}_i$ ($\mathcal{T}_i$ denotes the i_{th}-type of attributes).

Meanwhile, for the *attributed heterogeneous social networks* with shared users, they can be represented as the multiple *aligned attributed heterogeneous social networks* (or *aligned social networks* for short).

Definition 2 (Multiple Aligned Social Networks): Given online social networks $G^{(1)}$, $G^{(2)}$, $\cdots$, $G^{(n)}$ sharing common users, they can be represented as the *multiple aligned social networks* $\mathcal{G} = \big((G^{(1)}, G^{(2)}, \cdots, G^{(n)}), (\mathcal{A}^{(1,2)}, \mathcal{A}^{(1,3)}, \cdots, \mathcal{A}^{(n-1,n)})\big)$, where $\mathcal{A}^{(i,j)}$ represents the set of undirected anchor links connecting the common users between networks $G^{(i)}$ and $G^{(j)}$.

In Figure 1, we show an example of two *aligned social networks*, Foursquare and Twitter, which can be represented as $\mathcal{G} = ((G^{(1)}, G^{(2)}), \mathcal{A}^{(1,2)})$. Formally, the Twitter network can be represented as $G^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)}, \mathcal{T}^{(1)})$, where $\mathcal{V}^{(1)} = \mathcal{U}^{(1)} \cup \mathcal{P}^{(1)}$ denotes the set of nodes in the network including users and posts, and $\mathcal{E}^{(1)} = \mathcal{E}^{(1)}_{u,u} \cup \mathcal{E}^{(1)}_{u,p}$ involves the sets of social links among users as well as write links between users and posts. For the posts, a set of attributes can be extracted, which can be represented as $\mathcal{T}^{(1)} = \mathcal{T}^{(1)}_w \cup \mathcal{T}^{(1)}_l \cup \mathcal{T}^{(1)}_t$ denoting the words, location checkins and timestamps attached to the posts in $\mathcal{P}^{(1)}$ respectively. The Foursquare network has a similar structure as Twitter, which can be represented as $G^{(2)} = (\mathcal{V}^{(2)}, \mathcal{E}^{(2)}, \mathcal{T}^{(2)})$. Twitter and Foursquare are aligned together by the user anchor links connecting the shared users, and they also share some common attributes at the same time.

In this paper, we will use these two *aligned social networks* $\mathcal{G} = ((G^{(1)}, G^{(2)}), \mathcal{A}^{(1,2)})$ as an example to illustrate the problem setting and proposed model, but simple extensions of the model can be applied to multiple (more than two) aligned social networks as well.

B. Problem Definition

Problem Definition: Given a pair of partially *aligned social networks* $\mathcal{G} = ((G^{(1)}, G^{(2)}), \mathcal{A}^{(1,2)})$, we can represent all the potential anchor links between networks $G^{(1)}$ and $G^{(2)}$ as set $\mathcal{H} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$, where $\mathcal{U}^{(1)}$ and $\mathcal{U}^{(2)}$ denote the user sets in $G^{(1)}$ and $G^{(2)}$ respectively. For the known links between networks, we can group them as a labeled set $\mathcal{L} = \mathcal{A}^{(1,2)}$. The remaining anchor links with unknown labels are those to be inferred, and they can be formally denoted as the unlabeled set $\mathcal{U} = \mathcal{H} \setminus \mathcal{L}$. In the ANNA problem, based on both labeled anchor links in $\mathcal{L}$ and unlabeled anchor links in $\mathcal{U}$, we aim at building a mapping function $f : \mathcal{H} \to \mathcal{Y}$ to infer anchor link labels in $\mathcal{Y} = \{0, +1\}$ subject to the *one-to-one* constraint, where class labels $+1$ and 0 denote the existing and non-existing anchor links respectively. Besides these known links, in the ANNA problem, we are also allowed to query for the label of links in set $\mathcal{U}$ with a pre-specified budget b, i.e., the number of allowed queries. Besides learning the optimal variables in the mapping function $f(\cdot)$, we also aim at selecting an optimal query set $\mathcal{U}_q$ to improve the performance of the learned mapping function $f(\cdot)$ as much as possible.

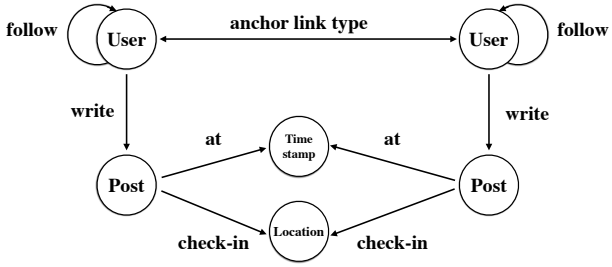


Fig. 2. Schema of aligned networks.

III. PROPOSED METHOD

In this section, we will introduce the proposed model *ActiveIter* in detail. At the very beginning, we will introduce the notations used in this paper. After that, the formal definition of *Meta Diagram* will be provided, based on which a set of meta diagram based features will be extracted. Finally, we will introduce the active network alignment model for anchor link inference.

A. Notations

In the sequel, we will use lower case letters (e.g., x) to denote 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. The i_{th} entry of vector $\mathbf{x}$ is denoted as $x(i)$. Given a matrix $\mathbf{X}$, we denote $\mathbf{X}(i,:)$ (and $\mathbf{X}(:,j)$) as the i_{th} row (and the j_{th} column) of $\mathbf{X}$, and the (i_{th}, j_{th}) entry of matrix $\mathbf{X}$ can be denoted as $X(i,j)$ or $X_{i,j}$ (which are interchangeable). We use $\mathbf{X}^\top$ (and $\mathbf{x}^\top$) to denote the transpose of matrix $\mathbf{X}$ (and vector $\mathbf{x}$). For vector $\mathbf{x}$, we denote its L_p-norm as $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$, and the L_p-norm of matrix $\mathbf{X}$ can be represented as $\|\mathbf{X}\|_p = (\sum_{i,j} |X(i,j)|^p)^{\frac{1}{p}}$. Given two vectors $\mathbf{x}$, $\mathbf{y}$ of the same dimension, we use notation $\mathbf{x} \le \mathbf{y}$ to denote that entries in $\mathbf{x}$ are no greater than the corresponding entries in $\mathbf{y}$.

B. Meta Diagram based Proximity Features

The *attributed heterogeneous social network* introduced in Section II provides a unified representation for most of the popular online social networks, like Facebook, Twitter and Foursquare.

1) Network Schema and Inter-Network Meta Path: To effectively categorize the diverse information in the *aligned social networks*, we introduce the *aligned network schema* concept as follows.

Definition 3 (Aligned Social Network Schema): Formally, the schema of the given aligned social networks $\mathcal{G} = ((G^{(1)}, G^{(2)}), \mathcal{A}^{(1,2)})$ can be represented as $S_{\mathcal{G}} = ((S_{G^{(1)}}, S_{G^{(2)}}), \{\text{anchor}\})$. Here, $S_{G^{(1)}} = (\mathcal{N}^{(1)}_{\mathcal{V}} \cup \mathcal{N}_{\mathcal{T}}, \mathcal{R}_{\mathcal{E}} \cup \mathcal{R}_{\mathcal{A}})$, where $\mathcal{N}^{(1)}_{\mathcal{V}}$ and $\mathcal{N}_{\mathcal{T}}$ denote the set of node types and attribute types in the network, while $\mathcal{R}_{\mathcal{E}}$ represents the set of link types in the network, and $\mathcal{R}_{\mathcal{A}}$ denotes the set of association types between nodes and attributes. In a similar way, we can represent the schema of $G^{(2)}$ as $S_{G^{(2)}} = (\mathcal{N}^{(2)}_{\mathcal{V}} \cup \mathcal{N}_{\mathcal{T}}, \mathcal{R}_{\mathcal{E}} \cup \mathcal{R}_{\mathcal{A}})$.

TABLE I
SUMMARY OF INTER-NETWORK META DIAGRAMS.

ID	Notation	Meta Diagram	Semantics
P_1	$\mathrm{U} \to \mathrm{U} \leftrightarrow \mathrm{U} \leftarrow \mathrm{U}$	User $\xrightarrow{follow}$ User $\xleftarrow{anchor}$ User $\xleftarrow{follow}$ User	Common Anchored Followee
P_2	$\mathrm{U} \leftarrow \mathrm{U} \leftrightarrow \mathrm{U} \to \mathrm{U}$	User $\xleftarrow{follow}$ User $\xleftarrow{anchor}$ User $\xrightarrow{follow}$ User	Common Anchored Follower
P_3	$\mathrm{U} \to \mathrm{U} \leftrightarrow \mathrm{U} \to \mathrm{U}$	User $\xrightarrow{follow}$ User $\xleftarrow{anchor}$ User $\xrightarrow{follow}$ User	Common Anchored Followee-Follower
P_4	$\mathrm{U} \leftarrow \mathrm{U} \leftrightarrow \mathrm{U} \leftarrow \mathrm{U}$	User $\xleftarrow{follow}$ User $\xleftarrow{anchor}$ User $\xleftarrow{follow}$ User	Common Anchored Follower-Followee
P_5	$\mathrm{U} \to \mathrm{P} \to \mathrm{T} \leftarrow \mathrm{P} \leftarrow \mathrm{U}$	User $\xrightarrow{write}$ Post $\xrightarrow{at}$ Timestamp $\xleftarrow{at}$ Post $\xleftarrow{write}$ User	Common Timestamp
P_6	$\mathrm{U} \to \mathrm{P} \to \mathrm{L} \leftarrow \mathrm{P} \leftarrow \mathrm{U}$	User $\xrightarrow{write}$ Post $\xrightarrow{checkin}$ Location $\xleftarrow{checkin}$ Post $\xleftarrow{write}$ User	Common Checkin
$\Psi_1(\mathrm{P}_1 \times \mathrm{P}_2)$	$\mathrm{U} \leftrightarrow \mathrm{U} \xleftrightarrow{anchor} \mathrm{U} \leftrightarrow \mathrm{U}$	User $\rightleftarrows$ User $\xleftrightarrow{anchor}$ User $\rightleftarrows$ User	Common Aligned Neighbors
$\Psi_2(\mathrm{P}_5 \times \mathrm{P}_6)$	$\mathrm{U} \to \mathrm{P} \begin{smallmatrix}\to \mathrm{L} \leftarrow \\ \to \mathrm{T} \leftarrow\end{smallmatrix} \mathrm{P} \leftarrow \mathrm{U}$	User $\xrightarrow{write}$ Post $\begin{smallmatrix}\xrightarrow{checkin} \mathrm{Location} \xleftarrow{checkin} \\ \xrightarrow[at]{} \mathrm{Timestamp} \xleftarrow[at]{}\end{smallmatrix}$ Post $\xleftarrow{write}$ User	Common Attributes
$\Psi_3(\mathrm{P}_1 \times \mathrm{P}_5 \times \mathrm{P}_6)$	$\begin{smallmatrix}\mathrm{U} & & & & \mathrm{U} \\ \uparrow & & & & \uparrow \\ \mathrm{U} \to \mathrm{P} & \begin{smallmatrix}\to \mathrm{L} \leftarrow \\ \to \mathrm{T} \leftarrow\end{smallmatrix} & & \mathrm{P} \leftarrow \mathrm{U}\end{smallmatrix}$	User $\rightleftarrows$ User $\xleftrightarrow{anchor}$ User $\rightleftarrows$ User $\xrightarrow{write}$ Post $\begin{smallmatrix}\xrightarrow{checkin} \mathrm{Location} \xleftarrow{checkin} \\ \xrightarrow[at]{} \mathrm{Timestamp} \xleftarrow[at]{}\end{smallmatrix}$ Post $\xleftarrow{write}$ User	Common Aligned Neighbor & Attributes

In the above definition, to simplify the representations, (1) the attribute types have no superscript, since lots of attribute types can be shared across networks; and (2) the relation types also have no superscript, and the network they belong to can be easily differentiated according to the superscript of user/post node types connected to them. According to the definition, as shown in Figure 2, we can represent the Twitter network schema as $S_{G^{(1)}} = (\mathcal{N}^{(1)}, \mathcal{R})$, $\mathcal{N}^{(1)} = \{\text{User}^{(1)}, \text{Post}^{(1)}, \text{Word}, \text{Location}, \text{Timestamp}\}$ (or $\mathcal{N}^{(1)} = \{\mathrm{U}^{(1)}, \mathrm{P}^{(1)}, \mathrm{W}, \mathrm{L}, \mathrm{T}\}$ for short) and $\mathcal{R} = \{\text{follow}, \text{write}, \text{at}, \text{check-in}\}$. The Foursquare network schema has exactly the same representation, and it can be denoted as $S_{G^{(2)}} = (\mathcal{N}^{(2)}, \mathcal{R})$, where $\mathcal{N}^{(2)} = \{\mathrm{U}^{(2)}, \mathrm{P}^{(2)}, \mathrm{W}, \mathrm{L}, \mathrm{T}\}$ and $\mathcal{R} = \{\text{follow}, \text{write}, \text{at}, \text{check-in}\}$. Nodes between Twitter and Foursquare can be connected with each other via connections consisting of various types of links. To categorize all these possible connections across networks, we define the concept of *inter-network meta path* based on the schema as follows:

Definition 4 (Inter-Network Meta Path): Based on an aligned attributed network schema, $S_{\mathcal{G}} = ((S_{G^{(1)}}, S_{G^{(2)}}), \{\text{anchor}\})$, path $\mathrm{P} = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{n-1}} N_n$ is defined to be an *inter-network meta path* of length $n-1$ between networks $G^{(1)}$ and $G^{(1)}$, where $N_i \in \mathcal{N}^{(1)} \cup \mathcal{N}^{(2)}, i \in \{1, 2, \cdots, n\}$ and $R_i \in \mathcal{R} \cup \{anchor\}, i \in \{1, 2, \cdots, n-1\}$. In this paper, we are only concerned about *inter-network meta paths* connecting users across networks, in which $N_1, N_n \in \{\mathrm{U}^{(1)}, \mathrm{U}^{(2)}\} \wedge N_1 \neq N_n$.

Based on the aligned network schema shown in Figure 2, several *anchor meta paths* $\{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_6\}$ can be defined, whose physical meanings and notations are summarized in the top part of Table I.

2) Inter-Network Meta Diagram: For the applications on real-world online social networks, these meta paths extracted in the pervious subsection may suffer from two major disadvantages. Firstly, meta path cannot characterize rich semantics. For instance, given two users $u_i^{(1)}$ and $u_j^{(2)}$ with check-in records "$u_i^{(1)}$: (Chicago, Aug. 2016), (New York, Jan. 2017), (Los Angeles, May 2017)", and "$u_j^{(2)}$: (Los Angeles, Aug. 2016), (Chicago, Jan. 2017), (New York, May 2017)" respectively, based on meta path P_5 and P_6, user pair $u_i^{(1)}$, $u_j^{(2)}$ have a lot in common and are highly likely to be the same user, since they have either checked-in the same locations (for 3 times) or at the same time (for 3 times). However, according to their check-in records, we observe that their activities are totally "dislocated" as they have never been at the same place for the same moments. Secondly, different meta paths denote different types of connections among users, and assembling them in an effective way is another problem. Actually, the meta paths can not only been concatenated but also stacked. Based on such an intuition, to solve these two challenges, we introduce a new concept *Inter-Network Meta Diagram*, which is a meta subgraph that fuses diverse relationships together for capturing richer semantic information across aligned attributed heterogeneous networks specifically. *Inter-network meta diagram* is different from the intra-network *meta graph* [28] and *meta structure* [5] concepts proposed in the existing works, since it mainly exists across multiple heterogeneous networks. More detailed information about these concepts and their differences will be provided in Section V

Definition 5 (Anchor Meta Diagram): Give a network schema as $S_{\mathcal{G}} = ((S_{G^{(1)}}, S_{G^{(2)}}), \{\text{anchor}\})$, an *inter-network meta diagram* can be formally represented as a directed acyclic subgraph $\Psi = (\mathcal{N}_\Psi, \mathcal{R}_\Psi, N_s, N_t)$, where $\mathcal{N}_\Psi \subset \mathcal{N}^{(1)} \cup \mathcal{N}^{(2)}$ and $\mathcal{R}_\Psi \subset \mathcal{R} \cup \{anchor\}$ represents the node, attribute and link types involved, while $N_s, N_t \in \{\mathrm{U}^{(1)}, \mathrm{U}^{(2)}\} \wedge N_s \neq N_t$ denote the source and sink user node types from network $G^{(1)}$ and $G^{(2)}$ respectively.

Inter-network meta diagram proposed for the *aligned attributed heterogeneous networks* involves not only regular node types but also attribute types and it connects user node types across networks, which renders it different from the recent intra-network *meta structure* [5] or *meta graph* [28] concepts proposed for single non-attributed networks. Several *meta diagram* examples have been extracted from the networks as shown at the bottom part of Table I which can be represented as $\{\Psi_1, \Psi_2, \Psi_3\}$. Here, the *meta diagrams* Ψ_1 and Ψ_2 are composed of 2 meta paths based on social relationship and anchor (i.e., P_1 and P_2), as well as attributes (i.e., P_5 and P_6) respectively; Ψ_3 is composed of 3 meta paths P_1,

P_5 and P_6 respectively. Besides these listed meta diagram examples shown in Table I, several other *meta diagrams* are also extracted. Formally, we can use $\mathrm{P}_f = \{\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, \mathrm{P}_4\}$ and $\mathrm{P}_a = \{\mathrm{P}_5, \mathrm{P}_6\}$ represent the sets of meta paths composed of the social relationships and the attributes respectively. The complete list of inter-network meta diagrams extracted in this paper are listed as follows:

- Ψ_{f^2} ($\mathrm{P}_f \times \mathrm{P}_f$): Common Aligned Neighbors.
- Ψ_{a^2} ($\mathrm{P}_a \times \mathrm{P}_a$): Common Attributes.
- $\Psi_{f,a}$ ($\mathrm{P}_f \times \mathrm{P}_a$): Common Aligned Neighbor & Attribute.
- Ψ_{f,a^2} ($\mathrm{P}_f \times \mathrm{P}_a \times \mathrm{P}_a$): Common Aligned Neighbor & Attributes.
- Ψ_{f^2,a^2} ($\mathrm{P}_f \times \mathrm{P}_f \times \mathrm{P}_a \times \mathrm{P}_a$): Common Aligned Neighbors & Attributes.

Here, $\Psi_{f^2} = \mathrm{P}_f \times \mathrm{P}_f = \{\mathrm{P}_i \times \mathrm{P}_j\}_{\mathrm{P}_i \in \mathrm{P}_f, \mathrm{P}_j \in \mathrm{P}_f}$, and $\Psi_{f,a} = \mathrm{P}_f \times \mathrm{P}_a = \{\mathrm{P}_i \times \mathrm{P}_j\}_{\mathrm{P}_i \in \mathrm{P}_f, \mathrm{P}_j \in \mathrm{P}_a}$, and similar for the remaining notations. The operator $\mathrm{P}_i \times \mathrm{P}_j$ denotes the stacking of meta paths P_i and P_j via the common node types shared by them. For instance, Ψ_1 is an anchor meta diagram composed by stacking two anchor meta paths of social relationships, i.e., $\Psi_1 \in \Psi_{f^2}$. Actually, *meta path* is also a special type of *meta diagram* in the shape of path. To unify the terms, we will misuse meta diagram to refer to both *meta path* and *meta diagram* in this paper. Formally, all the *meta diagrams* extracted from the social networks can be represented as $\Phi = \mathrm{P} \cup \Psi_{f^2} \cup \Psi_{a^2} \cup \Psi_{f,a} \cup \Psi_{f,a^2} \cup \Psi_{f^2,a^2}$.

3) Proximity Feature Extraction with Meta Diagram: Given a pair of users, e.g., $u_i^{(1)}$ and $u_j^{(2)}$, based on meta diagram $\Phi_k \in \Phi$, we can represent the set of meta diagram instances connecting $u_i^{(1)}$ and $u_j^{(2)}$ as $\mathcal{P}_{\Phi_k}(u_i^{(1)}, u_j^{(2)})$. Users $u_i^{(1)}$ and $u_j^{(2)}$ can have multiple meta diagram instances going into/out from them. Formally, we can represent all the meta diagram instances going out from user $u_i^{(1)}$ (or going into $u_j^{(2)}$), based on meta diagram Φ_k, as set $\mathcal{P}_{\Phi_k}(u_i^{(1)}, \cdot)$ (or $\mathcal{P}_{\Phi_k}(\cdot, u_j^{(2)})$). The proximity score between $u_i^{(1)}$ and $u_j^{(2)}$ based on meta diagram Φ_k can be represented as the following *meta proximity* concept formally.

Definition 6 (Meta Diagram Proximity): Based on meta diagram Φ_k, the meta diagram proximity between users $u_i^{(1)}$ and $u_j^{(2)}$ in G can be represented as

$$s_{\Phi_k}(u_i^{(1)}, u_j^{(2)}) = \frac{2|\mathcal{P}_{\Phi_k}(u_i^{(1)}, u_j^{(2)})|}{|\mathcal{P}_{\Phi_k}(u_i^{(1)}, \cdot)| + |\mathcal{P}_{\Phi_k}(\cdot, u_j^{(2)})|}.$$

Meta diagram proximity considers not only the meta diagram instances between $u_i^{(1)}$ and $u_j^{(2)}$ but also penalizes those going out from and into $u_i^{(1)}$ and $u_j^{(2)}$, respectively, at the same time. Since the meta diagrams span the whole network, both the local and global network structure can be captured by the the meta diagrams. With the above meta proximity definition, we can represent the meta proximity scores among all users in the network G based on meta diagram Φ_k as matrix $\mathbf{S}_{\Phi_k} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$, where entry $S_{\Phi_k}(i,j) = s_{\Phi_k}(u_i^{(1)}, u_j^{(2)})$. All the meta proximity matrices defined for network G can be represented as $\{\mathbf{S}_{\Phi_k}\}_{\Phi_k \in \Phi}$.

Meanwhile, according to the *meta proximity* definition, to compute the proximity scores among users, we need to count the number of *meta diagram* instances connecting users. However, different from the meta path instance counting (which can be done in polynomial time), counting the number of *meta diagram* instances among users is never an easy task. It involves the graph isomorphism step to match subnetworks with the *meta diagram* structure and node/link types. To lower down the computational time costs, we propose the *minimum meta diagram covering set* concept, which will be applied to shrink the search space of nodes in the networks.

Definition 7 (Meta Diagram Covering Set): Give a *anchor meta diagram* Ψ starting and ending with node types n_s and n_t, Ψ will contain multiple paths connecting n_s and n_t. Formally, these covered paths connecting n_s and n_t can be represented as the covering set of Ψ, i.e., $\mathcal{C}(\Psi) = \{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$, where $\mathrm{P}_i \in \mathcal{C}(\Psi)$ denotes a *meta path* from n_s to n_t. Anchor meta diagram Ψ can be decomposed in different ways, and we are only interested in the *minimum meta diagram covering set* with the smallest size $|\mathcal{C}(\Psi)|$. The the anchor meta diagram covering set recovers the set of meta paths composing the diagrams as introduced before, which can clearly indicate the relationship between meta path and meta diagram.

LEMMA 1: Given a *meta diagram* Ψ, a pair of nodes $u_i^{(1)}, u_j^{(2)} \subset \mathcal{V}$ are connected by instances of *meta diagram* Ψ iff $u_i^{(1)}, u_j^{(2)}$ can be connected by instances of all meta paths in its covering set $\mathcal{C}(\Psi)$.

PROOF: The lemma can be proved by contradiction. Let's assume the lemma doesn't hold, and $\exists \mathrm{P}_k \in \mathcal{C}(\Psi)$ that cannot connect $u_i^{(1)}, u_j^{(2)}$ in the network, given that Ψ has an instance connecting $u_i^{(1)}, u_j^{(2)}$. Since P_k is one part of Ψ, and we can identify the corresponding parts of P_k from Ψ's instance, which will create a path connecting $u_i^{(1)}$ with $u_j^{(2)}$. It contradicts the assumption. Therefore, the Lemma should hold.

Furthermore, based on the above Lemma 1, we can also derive the relationship between the covering sets of meta diagrams.

LEMMA 2: Given two *meta diagrams* Ψ_i and Ψ_j, where $\mathcal{C}(\Psi_i) \subseteq \mathcal{C}(\Psi_j)$, if a pair of nodes $u_i^{(1)}, u_j^{(2)} \subset \mathcal{V}$ can be connected by instances of *meta diagram* Ψ_j, there will also be an instance of *meta diagram* Ψ_i connecting $u_i^{(1)}, u_j^{(2)}$ in the network as well.

The above lemma can be proved in a similar way as the proof of Lemma 1, which will not be introduced here due to the limited space. Based on the above lemmas, we propose to apply the *meta diagram covering set* to help shrink the search space. First of all, we can compute the set of meta path instances connecting users across networks. Formally, given a *meta diagram* Ψ_k, we can obtain its minimum covering set $\mathcal{C}(\Psi_k)$. For each meta path in $\mathcal{C}(\Psi)$, a set of meta path instances connecting the input node pairs can be extracted. By combining these meta path instances together and checking their existence in the network, we will extract instances of Ψ.

Furthermore, in the case that there exist a prior computation result of meta diagram $\Psi_{k'}$ with covering set $\mathcal{C}(\Psi_{k'}) \subset \mathcal{C}(\Psi_k)$, instead of recompute the diagram instances based on meta paths in $\mathcal{C}(\Psi)$, we can just combine the meta diagram instances of $\Psi_{k'}$ and the instances of meta paths in $\mathcal{C}(\Psi_k) \setminus \mathcal{C}(\Psi_{k'})$ to get the meta diagram instance for Ψ_k.

C. Active Network Alignment Model

In this part, we will introduce the *active network alignment* model *ActiveIter* for the anchor link prediction across networks, which involves 4 main components: (1) *discriminative function* for labeled instances, (2) *generative function* for unlabeled instance, (3) *one-to-one constraint* modeling, and (4) *active query* component.

1) Labeled Data Discriminative Loss Function: For all the potential anchor links in set $\mathcal{H}$ (involving both the labeled and unlabeled anchor link instances), a set of features will be extracted based on the meta diagrams introduced before. Formally, the feature vector extracted for anchor link $l \in \mathcal{H}$ can be represented as vector $\mathbf{x}_l \in \mathbb{R}^d$ (parameter d denotes the feature size). Meanwhile, we can denote the label of link $l \in \mathcal{L}$ as $y_l \in \mathcal{Y}$ ($\mathcal{Y} = \{0, +1\}$), which denotes the existence of anchor link l between the networks. For the existing anchor links in set $\mathcal{L}_+$, they will be assigned with $+1$ label; while the labels of anchor links in $\mathcal{U}$ are unknown. All the labeled anchor links in set $\mathcal{L}_+$ can be represented as a tuple set $\{(\mathbf{x}_l, y_l)\}_{l \in \mathcal{L}_+}$. Depending on whether the anchor link instances are linearly separable or not, the extracted anchor link feature vectors can be projected to different feature spaces with various kernel functions $g : \mathbb{R}^d \to \mathbb{R}^k$. For instance, given the feature vector $\mathbf{x}_l \in \mathbb{R}^d$ of anchor link l, we can represent its projected feature vector as $g(\mathbf{x}_l) \in \mathbb{R}^k$. In this paper, the linear kernel function will be used for simplicity, and we have $g(\mathbf{x}_l) = \mathbf{x}_l$ for all the links l.

In the *active network alignment* model, the *discriminative* component can effectively differentiate the positive instances from the non-existing ones, which can be denoted as mapping $f(\cdot; \theta_f) : \mathbb{R}^d \to \{+1, 0\}$ parameterized by θ_f. In this paper, we will use a linear model to fit the link instances, and the *discriminative* model to be learned can be represented as $f(\mathbf{x}_l; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}_l + b$, where $\theta_f = [\mathbf{w}, b]$. By adding a dummy feature 1 for all the anchor link feature vectors, we can incorporate bias term b into the weight vector $\mathbf{w}$ and the parameter vector can be denoted as $\theta_f = \mathbf{w}$ for simplicity. Based on the above descriptions, we can represent the introduced *discriminative* loss function on the labeled set $\mathcal{L}_+$ as

$$L(f, \mathcal{L}_+; \mathbf{w}) = \sum_{l \in \mathcal{L}_+} \big(f(\mathbf{x}_l; \mathbf{w}) - y_l\big)^2 = \sum_{l \in \mathcal{L}_+} (\mathbf{w}^\top \mathbf{x}_l - y_l)^2.$$

2) Unlabeled Data Generative Loss Function: Meanwhile, to alleviate the insufficiency of labeled data, we also propose to utilize the unlabeled anchor links to encourage the learned model can capture the salient structures of all the anchor link instances. Based on the above discriminative model function $f(\cdot; \mathbf{w})$, for a unlabeled anchor link $l \in \mathcal{U}$, we can represent its inferred "label" as $y_l = f(\mathbf{x}_l; \mathbf{w})$. Considering that the result of $f(\cdot; \mathbf{w})$ may not necessary the exact label values in $\mathcal{Y}$, in the generative component, we can represent the generated anchor link label as $sign\big(f(\mathbf{x}_l; \mathbf{w})\big) \in \{+1, 0\}$. How to determine its value will be introduced later in Section III-D. Based on it, the loss function introduced in the generative component based on the unlabeled anchor links can be denoted as

$$L(f, \mathcal{U}; \mathbf{w}) = \sum_{l \in \mathcal{U}} \Big(\mathbf{w}^\top \mathbf{x}_l - sign\big(f(\mathbf{x}_l; \mathbf{w})\big)\Big)^2.$$

3) Query Component and Query Loss Function: Furthermore, besides the labeled links, a subset of the anchor links in $\mathcal{U}$ will be selected to query for the labels from the oracle, which can be denoted as set $\mathcal{U}_q$ formally. The true label of anchor link $l \in \mathcal{U}_q$ after query can be represented as $\tilde{y}_l \in \{+1, 0\}$. The remaining anchor links in set $\mathcal{U}$ can be represented as $\mathcal{U} \setminus \mathcal{U}_q$. Based on the loss functions introduced before, depending on whether the labels of links are queried or not, we can further specify the loss function for set $\mathcal{U}$ as

$$\begin{aligned} &L(f, \mathcal{U}; \mathbf{w}) = L(f, \mathcal{U}_q; \mathbf{w}) + L(f, \mathcal{U} \setminus \mathcal{U}_q; \mathbf{w}) \\ &= \sum_{l \in \mathcal{U}_q} (\mathbf{w}^\top \mathbf{x}_l - \tilde{y}_l)^2 + \sum_{l \in \mathcal{U} \setminus \mathcal{U}_q} \Big(\mathbf{w}^\top \mathbf{x}_l - sign\big(f(\mathbf{x}_l; \mathbf{w})\big)\Big)^2. \end{aligned}$$

Here, we need to add more remarks that notation $\tilde{y}_l$ denotes the queried label of anchor link $l \in \mathcal{U}_q$ which will be a known value, while the labels for the remaining anchor link $l \in \mathcal{U} \setminus \mathcal{U}_q$ will to be inferred in the model.

4) Cardinality Mathematical Constraint: As introduced before, the anchor links to be inferred between networks are subject to the *one-to-one* cardinality constraint. Such a constraint will control the maximum number of anchor links incident to the user nodes in each networks. Subject to the cardinality constraints, the prediction task of anchor links between networks are no longer independent. For instance, if anchor link $(u^{(1)}, v^{(2)})$ is predicted to be positive, then all the remaining anchor links incident to $u^{(1)}$ and $v^{(2)}$ in the unlabeled set $\mathcal{U}$ will be negative by default. Viewed in such a perspective, the cardinality constraint on anchor links should be effectively incorporated in model building, which will be modeled as the mathematical constraints on node degrees in this paper. To represent the user node-anchor link relationships in networks $G^{(1)}$ and $G^{(2)}$ respectively, we introduce the user node-anchor link incidence matrices $\mathbf{A}^{(1)} \in \{0, 1\}^{|\mathcal{U}^{(1)}| \times |\mathcal{H}|}, \mathbf{A}(2) \in \{0, 1\}^{|\mathcal{U}^{(2)}| \times |\mathcal{H}|}$ in this paper. Entry $A^{(1)}(i, j) = 1$ iff anchor link $l_j \in \mathcal{H}$ is connected with user node $u_i^{(1)}$ in $G^{(1)}$, and it is similar for $A^{(2)}$.

According to the analysis provided before, we can represent the labels of links in $\mathcal{H}$ as vector $\mathbf{y} \in \{+1, 0\}^{|\mathcal{H}|}$, where entry $y(i)$ represents the label of link $l_i \in \mathcal{L}$. Depending on which group l_i belongs to, its value has different representations as introduced before $y(i) = +1$, if $l_i \in \mathcal{L}_+$; $y(i) \tilde{y}_{l_i}$, if $l_i \in \mathcal{U}_q$, and $y(i)$ is unknown if $l_i \in \mathcal{U} \setminus \mathcal{U}_q$. Furthermore, based on the anchor link label vector $\mathbf{y}$, user node-anchor link incidence matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$, we can represent the user node

degrees in networks $G^{(1)}$ and $G^{(2)}$ as vectors $\mathbf{d}^{(1)} \in \mathbb{N}^{|\mathcal{H}|}$ and $\mathbf{d}^{(2)} \in \mathbb{N}^{|\mathcal{H}|}$ respectively.

$$\mathbf{d}^{(1)} = \mathbf{A}^{(1)}\mathbf{y}, \text{ and } \mathbf{d}^{(2)} = \mathbf{A}^{(2)}\mathbf{y}.$$

Therefore, the *one-to-one* constraint on anchor links can be denoted as the constraints on node degrees in $G^{(1)}$ and $G^{(2)}$ as follows:

$$\mathbf{0} \leq \mathbf{A}^{(1)}\mathbf{y} \leq \mathbf{1}, \text{ and } \mathbf{0} \leq \mathbf{A}^{(2)}\mathbf{y} \leq \mathbf{1}.$$

D. Joint Optimization Objective Function

Based on the introduction in the previous subsection, by combining the loss terms introduced by the labeled, queried and remaining unlabeled anchor links together with the cardinality constraint, we can represent the joint optimization objective function as

$$\begin{aligned}\min_{\mathbf{w},\mathbf{y},\mathcal{U}_q} \; & L(f,\mathcal{L}_+;\mathbf{w}) + \alpha \cdot L(f,\mathcal{U}_q;\mathbf{w}) \\ & + \beta \cdot L(f,\mathcal{U}\setminus\mathcal{U}_q;\mathbf{w}) + \gamma \cdot \|\mathbf{w}\|_2^2 \\ s.t. \; & |\mathcal{U}_q| \leq b, \text{ and } y_l = \tilde{y}_l, \forall l \in \mathcal{U}_q, \\ & y_l \in \{+1,0\}, \forall l \in \mathcal{U}\setminus\mathcal{U}_q, \text{ and } y_l = +1, \forall l \in \mathcal{L}_+, \\ & \mathbf{0} \leq \mathbf{A}^{(1)}\mathbf{y} \leq \mathbf{1}, \text{ and } \mathbf{0} \leq \mathbf{A}^{(2)}\mathbf{y} \leq \mathbf{1}.\end{aligned}$$

Here, we set the weight scalar α and β with the value 1, because we assume that each link is equally important for training, if no other external knowledge exists, regardless of whether it belongs to $\mathcal{U}_q$ or $\mathcal{U}\setminus\mathcal{U}_q$. In this way, the new loss term of all the links in sets $\mathcal{L}_+, \mathcal{U}_q$ and $\mathcal{U}\setminus\mathcal{U}_q$ can be simplified as

$$\begin{aligned}& L(f,\mathcal{L}_+;\mathbf{w}) + \alpha \cdot L(f,\mathcal{U}_q;\mathbf{w}) + \beta \cdot L(f,\mathcal{U}\setminus\mathcal{U}_q;\mathbf{w}) \\ & = L(f,\mathcal{H};\mathbf{w}) = \|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2,\end{aligned}$$

where matrix $\mathbf{X} = [\mathbf{x}_{l_1}^\top, \mathbf{x}_{l_2}^\top, \cdots, \mathbf{x}_{l_{|\mathcal{H}|}}^\top]^T$ denotes the feature matrix of all the links in set $\mathcal{H}$.

Here, we can see the objective function involve multiple variables, i.e., variable $\mathbf{w}$, label $\mathbf{y}$, and the query set $\mathcal{U}_q$, and the objective is not jointly convex with regarding these variables. What's more, the inference of the label variable $\mathbf{y}$ and the query set $\mathcal{U}_q$ are both combinatorial problems, and obtaining their optimal solution will be NP-hard. In this paper, we design an hierarchical alternative variable updating process for solving the problem instead:

1) fix $\mathcal{U}_q$, and update $\mathbf{y}$ and $\mathbf{w}$,
(1-1) with fixed $\mathcal{U}_q$, fix $\mathbf{y}$, update $\mathbf{w}$,
(1-2) with fixed $\mathcal{U}_q$, fix $\mathbf{w}$, update $\mathbf{y}$,
2) fix $\mathbf{y}$ and $\mathbf{w}$, and update $\mathcal{U}_q$.

A remark to be added here: we can see that variable $\mathcal{U}_q$ is different from the remaining two, which involves the label query process with the oracle subject to the specified budget. To differentiate these two iterations, we call the iterations (1) and (2) as the *external iteration*, while call (1-1) and (1-2) *internal iteration*. Next, we will illustrate the detailed alternative learning algorithm as follows.

• **External Iteration Step (1)**: Fix $\mathcal{U}_q$, Update $\mathbf{y}$, $\mathbf{w}$.

▪ **Internal Iteration Step (1-1)**: Fix $\mathcal{U}_q$, $\mathbf{y}$, Update $\mathbf{w}$.

With $\mathbf{y}$, $\mathcal{U}_q$ fixed, we can represent the objective function involving variable $\mathbf{w}$ as

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

The objective function is a quadratic convex function, and its optimal solution can be represented as

$$\mathbf{w} = \mathbf{H}\mathbf{y} = c(\mathbf{I} + c\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y},$$

where $\mathbf{H} = c(\mathbf{I} + c\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is a constant matrix. Therefore, the weight vector $\mathbf{w}$ depends only on the $\mathbf{y}$ variable.

▪ **Internal Iteration Step (1-2)**: Fix $\mathcal{U}_q$, $\mathbf{w}$, Update $\mathbf{y}$.

With $\mathcal{U}_q$, $\mathbf{w}$ fixed, together with the constraint, we know that terms $L(f,\mathcal{L}_+;\mathbf{w})$, $L(f,\mathcal{U}_q;\mathbf{w})$ and $\|\mathbf{w}\|_2^2$ are all constant. And the objective function will be

$$\begin{aligned}\min_{\mathbf{y}} \; & \|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2 \\ s.t. \; & y_l \in \{+1,0\}, \forall l \in \mathcal{U}\setminus\mathcal{U}_q, \\ & y_l = \tilde{y}_l, \forall l \in \mathcal{U}_q \text{ and } y_l = +1, \forall l \in \mathcal{L}_+, \\ & \mathbf{0} \leq \mathbf{A}^{(1)}\mathbf{y} \leq \mathbf{1}, \text{ and } \mathbf{0} \leq \mathbf{A}^{(2)}\mathbf{y} \leq \mathbf{1}.\end{aligned}$$

It is an integer programming problem, which has been shown to be NP-hard and no efficiently algorithm exists that lead to the optimal solution. In this paper, we will use the greedy link selection algorithm proposed in [21] based on values $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, which has been proven to achieve $\frac{1}{2}$-approximation of the optimal solution. The time complexity of this step is $O(|\tilde{L}|)$, where $\tilde{L} = \{l | l \in \mathcal{U}\setminus\mathcal{U}_q\}$.

• **External Iteration Step (2)**: Fix $\mathbf{w}$, $\mathbf{y}$, Update $\mathcal{U}_q$.

Selecting the optimal set $\mathcal{U}_q$ at one time involves the search of all the potential b link instance combinations from the unlabeled set $\mathcal{U}$, whose search space is $\binom{|\mathcal{U}|}{b}$, and there is no known efficient approach for solving the problem in polynomial time. Therefore, instead of selecting them all at one time, we propose to choose several link instances greedily in each iterations. Due to the one-to-one constraint, the unlabeled anchor links no longer bears equal information, and querying for labels of potential positive anchor links will be more "informative" compared with negative anchor links. Among the unlabeled links, *ActiveIter* selects a set of mis-classified false-negative anchor links (but with a large positive score) as the potential candidates, benefits introduced by whose label queries includes both their own label corrections and other extra label gains of their conflicting negative links at the same time. Formally, among all the unlabeled links in $\mathcal{U}$, we can represent the set of links classified to be positive/negative instances in the previous iteration step as $\mathcal{U}^+ = \{l | l \in \mathcal{U}, y_l = +1\}$ and $\mathcal{U}^- = \{l | l \in \mathcal{U}, y_l = 0\}$. Based on these two sets, the group of potentially mis-classified false-negative anchor link candidates as set

$$\begin{aligned}\mathcal{C} = \{l | l \in \mathcal{U}^-, \exists l', l'' \in \mathcal{U}^+ \text{ that conflicts with } l, \\ \hat{y}_{l'} \sim \hat{y}_l \gg \hat{y}_{l''} > 0\},\end{aligned}$$

TABLE II
PROPERTIES OF THE HETEROGENEOUS NETWORKS

		network	
	property	**Twitter**	**Foursquare**
# node	user	5,223	5,392
	tweet/tip	9,490,707	48,756
	location	297,182	38,921
# link	friend/follow	164,920	76,972
	write	9,490,707	48,756

where statement "l'/l'' conflicts with l" denotes l'/l'' and l are incident to the same nodes respectively. Operator $\hat{y}_{l'} \sim \hat{y}_l$ represents $\hat{y}_{l'}$ is close to $\hat{y}_l$ (whose difference threshold is set as 0.05 in the experiments). All the links in set $\mathcal{C}$ will be sorted according to value $\hat{y}_l - \hat{y}_{l''}$, and, instead of adding one by one, the top k candidates will be added to $\mathcal{U}_q$ in this iteration (Here, k denotes the query batch size, which is assigned with value 5 in the experiments). Because *ActiveIter* has to select the top k candidates from all potential candidates, where the potential candidates we defined as $\tilde{L}^- = \{l | l \in \mathcal{U} \setminus \mathcal{U}_q, \tilde{y}_l = 0\}$, the time complexity of External Iteration Step (2) is $O(|\tilde{L}^-|)$.

E. Time Complexity Analysis

Here, we start to analyze the time complexity of *ActiveIter* from a holistic perspective based on the analysis of each step in section III-D. As we set the query batch size as k and the budget as b, the whole hierarchical alternative variable updating process has to be executed b/k rounds. The iteration step (1-1) is a matrix multiplication which has he time complexity $O(d * |\mathcal{H}|)$. The time complexity the iteration step (1-2) is $O(|\tilde{L}|)$. Besides, the time complexity of the iteration step (2) is $O(|\tilde{L}^-|)$.We can find *ActiveIter* is scalable, with near linear runtime in the data size $|\mathcal{H}|$.

IV. EXPERIMENTS

To demonstrate the effectiveness of *ActiveIter* and the meta diagram based features, extensive experiments have been done on real-world heterogeneous social networks. In the following part, we will describe the dataset we use in experiments at first. Then we will introduce the experimental settings, including different comparison methods and evaluation metrics used in the experiments. At last, we will show the experimental results together with the convergence analysis and parameter sensitivity analysis.

A. Dataset Description

Our dataset used in experiments consists of two heterogeneous networks: Foursquare and Twitter. Both of them are famous online social networks. The key statistical data describing these two networks can be found in Table II. About the method and strategy of crawling this dataset, you can get detailed information in [7], [22].

- **Twitter**: Twitter is a popular online social network that provides a platform for users to share their life with their online friends. Lots of the tweets written by users in Twitter are location-related along with timestamp. Our dataset includes $4,893$ users and $9,490,707$ tweets. $257,248$ locations appears along with tweets. Besides, the number of follow links between these users is $164,920$ in total.
- **Foursquare**: Foursquare is another famous social network allowing users to interact with friends online through multiple location-related services. Our dataset has $5,392$ users in Foursquare and $76,972$ friendship relationship among them. All these users have checked-in at $38,921$ different locations via $48,756$ tips. There are $3,282$ anchor links between Twitter and Foursquare in the dataset.

B. Experimental Settings

1) Experimental Setup: In the experiments, we are able to acquire the set of anchor links between Foursquare and Twitter. The size of the set is $3,282$ which can be represented as $\mathcal{L}_+$. Based on the problem definition introduced in Section II-B, between the Foursquare and Twitter network, all the remaining non-existing anchor links can be represented as set $\mathcal{H}$. A proportion of non-anchor links are sampled randomly from $\mathcal{H} \setminus \mathcal{L}_+$ as negative set based on different negative-positive (NP) ratios θ. NP-ratio θ in experiments ranges from 5 to 50 with the step length 5. The positive and negative link sets are divided into 10 folds. Among them, 1 fold will be used as the training set and the remaining 9 folds as the test set. In order to simulate the problem setting without enough labeled data, we further sample a small proportion of labeled instances from the 1-fold training set as the final training set. The sampling process is controlled by parameter sample-ratio γ, which takes values in $\{10\%, 20\%, \cdots, 100\%\}$. Here, $\gamma = 10\%$ denotes only 10% of the 1-fold training set (i.e., only 1% of the complete labeled data) is sampled in the final training set; while $\gamma = 100\%$ means all the instances in the 1-fold training set (i.e., 10% of the labeled data) are used for training the model. In order to prevent unexpected impacts caused by data partitioning, we take 10 folds in turns to act as train set and the average metrics of 10 experiments are taken as the final results. We run the experiments on a Dell PowerEdge T630 Server with 2 20-core Intel CPUs and 256GB memory. The operating system is Ubuntu 16.04.3, and all codes are implemented in Python.

2) Comparison Methods: The methods used in experiments are listed as following, we use them to verify 2 aspects of conclusions. One is the effectiveness of meta diagram based feature vector, and the other is the advantage of *ActiveIter*.

- ***ActiveIter***: *ActiveIter* is the model proposed in this paper which implements the learning process described in Section 3.4. Through a limited budget, we aim at selecting a good query set with the objective to improve the performance of *ActiveIter*. Two different versions of *ActiveIter* with budgets 50 and 100 are compared in the experiments.
- ***ActiveIter-Rand***: In this method, we select the query set $\mathcal{U}_q$ in a random way in this method. The method is

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT METHODS FOR NETWORK ALIGNMENT. WE USE DIFFERENT NP-RATIOS WITH $\gamma = 60\%$.

metrics	methods	Negative Positive Ratio θ 5	10	15	20	25	30	35	40	45	50
F1	*ActiveIter-100*	**0.631±0.01**	**0.575±0.01**	**0.524±0.01**	**0.484±0.02**	**0.455±0.02**	**0.436±0.02**	**0.413±0.01**	**0.402±0.02**	**0.384±0.01**	**0.363±0.01**
	ActiveIter-50	0.625±0.01	0.571±0.01	0.514±0.01	0.482±0.02	0.454±0.02	0.429±0.02	0.404±0.01	0.392±0.02	0.374±0.02	0.361±0.01
	ActiveIter-Rand-50	0.616±0.01	0.553±0.01	0.501±0.01	0.463±0.01	0.437±0.01	0.413±0.01	0.392±0.02	0.381±0.02	0.368±0.02	0.352±0.01
	Iter-MPMD	0.616±0.01	0.556±0.01	0.507±0.01	0.469±0.02	0.441±0.01	0.414±0.02	0.396±0.01	0.380±0.03	0.365±0.01	0.350±0.01
	SVM-MPMD	0.387±0.05	0.300±0.04	0.247±0.04	0.165±0.06	0.159±0.06	0.150±0.03	0.152±0.04	0.102±0.06	0.091±0.07	0.049±0.06
	SVM-MP	0.476±0.11	0.093±0.08	0.055±0.05	0.004±0.01	0.002±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Precision	*ActiveIter-100*	**0.856±0.01**	**0.767±0.01**	**0.693±0.01**	**0.632±0.02**	**0.591±0.02**	**0.559±0.02**	**0.526±0.02**	**0.509±0.02**	**0.486±0.02**	**0.457±0.02**
	ActiveIter-50	0.848±0.01	0.762±0.01	0.676±0.02	0.626±0.02	0.587±0.02	0.551±0.02	0.515±0.02	0.496±0.03	0.473±0.02	0.454±0.02
	ActiveIter-Rand-50	0.836±0.01	0.735±0.01	0.657±0.01	0.600±0.02	0.563±0.02	0.528±0.02	0.498±0.03	0.481±0.02	0.462±0.02	0.440±0.02
	Iter-MPMD	0.835±0.01	0.738±0.01	0.665±0.01	0.609±0.02	0.569±0.02	0.530±0.02	0.504±0.02	0.4809±0.02	0.459±0.02	0.439±0.02
	SVM-MPMD	0.743±0.06	0.703±0.04	0.652±0.06	0.587±0.20	0.585±0.09	0.520±0.05	0.519±0.06	0.487±0.25	0.331±0.27	0.311±0.31
	SVM-MP	0.571±0.02	0.338±0.28	0.323±0.27	0.057±0.17	0.018±0.05	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Recall	*ActiveIter-100*	**0.499±0.01**	**0.460±0.01**	**0.422±0.01**	**0.392±0.01**	**0.371±0.01**	**0.357±0.01**	**0.339±0.01**	**0.332±0.01**	**0.318±0.01**	**0.301±0.01**
	ActiveIter-50	0.495±0.01	0.457±0.01	0.414±0.01	0.392±0.01	0.371±0.01	0.352±0.02	0.333±0.01	0.324±0.01	0.310±0.01	0.300±0.01
	ActiveIter-Rand-50	0.488±0.01	0.443±0.01	0.404±0.01	0.376±0.01	0.357±0.01	0.340±0.01	0.323±0.01	0.315±0.01	0.305±0.01	0.293±0.01
	Iter-MPMD	0.488±0.01	0.446±0.01	0.410±0.01	0.381±0.02	0.360±0.01	0.340±0.01	0.327±0.01	0.314±0.01	0.302±0.01	0.290±0.01
	SVM-MPMD	0.271±0.07	0.194±0.04	0.155±0.03	0.097±0.03	0.094±0.03	0.086±0.02	0.088±0.02	0.059±0.04	0.053±0.04	0.027±0.03
	SVM-MP	0.439±0.14	0.055±0.05	0.031±0.03	0.002±0.00	0.001±0.01	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Accuracy	*ActiveIter-100*	**0.902±0.00**	**0.938±0.00**	**0.952±0.01**	**0.960±0.00**	**0.966±0.00**	**0.970±0.00**	**0.973±0.00**	**0.976±0.00**	0.978±0.00	0.979±0.00
	ActiveIter-50	0.901±0.00	0.938±0.00	0.951±0.00	0.960±0.00	0.966±0.00	0.970±0.00	0.972±0.00	0.975±0.00	0.977±0.00	0.979±0.00
	ActiveIter-Rand-50	0.898±0.00	0.934±0.00	0.949±0.00	0.958±0.00	0.964±0.00	0.968±0.00	0.972±0.00	0.975±0.00	0.977±0.00	0.978±0.00
	Iter-MPMD	0.898±0.00	0.935±0.00	0.950±0.00	0.958±0.00	0.964±0.00	0.969±0.00	0.972±0.00	0.975±0.00	0.977±0.00	0.978±0.00
	SVM-MPMD	0.860±0.00	0.918±0.00	0.941±0.00	0.954±0.00	0.962±0.00	0.968±0.00	0.972±0.00	0.976±0.00	**0.978±0.00**	**0.980±0.00**
	SVM-MP	0.850±0.00	0.909±0.00	0.937±0.00	0.952±0.00	0.961±0.00	0.967±0.00	0.972±0.00	0.975±0.00	0.978±0.00	0.980±0.00

used to verify the effectiveness of the query set selection criteria used in *ActiveIter*.

- ***Iter-MPMD***: *Iter-MPMD* extends the cardinality constrained link prediction model proposed in [21] by incorporating the meta diagrams for feature extraction from aligned heterogeneous networks. ITER-MPMD is based on a PU (positive unlabeled) learning setting, without active query step.
- **SVM-MP**: SVM is a classic supervised learning model. The feature vector used for building the SVM-MP model are extracted merely based on the meta paths.
- **SVM-MPMD**: SVM-MPMD is identical to SVM-MP excepts it is built based on the features extracted with both meta paths and meta diagrams. Results comparison between SVM-MPMD and SVM-MP can verify the effectiveness of the meta diagram based features proposed in this paper. Meanwhile, comparison of SVM-MPMD and *Iter-MPMD* can also show that PU learning setting adopted in *Iter-MPMD* is suitable for the network alignment problem.

3) Evaluation Metrics: We choose to use conventional evaluation metrics to measure the performance of different methods in experiments. The methods we test in experiments, including SVM-MP, SVM-MPMD, *Iter-MPMD*, *ActiveIter-Rand*, and *ActiveIter*, can all output link prediction labels, and we will use F1, Recall, Precision and Accuracy as evaluation metrics. It should be noted that we need to query some labels in *ActiveIter-Rand* and *ActiveIter*. In other words, for the active-learning based methods, labels of these queried links are known already. In evaluation, we will remove these queried links from test set to maintain evaluation fairness between different comparison methods.

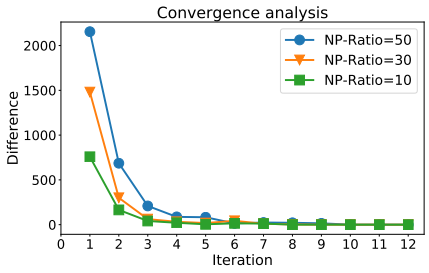


Fig. 3. Convergence analysis when sample-ratio=100%.

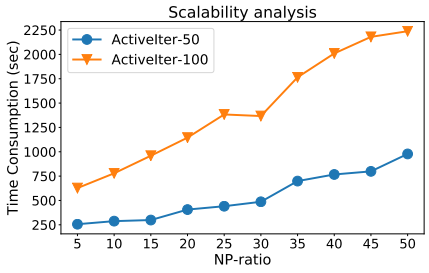


Fig. 4. Scalability analysis when sample-ratio=100%.

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT METHODS FOR NETWORK ALIGNMENT. WE USE DIFFERENT SAMPLE-RATIOS WITH $\theta = 50$.

metrics	methods	Sample Ratio γ 10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
F1	*ActiveIter-100*	**0.235±0.00**	**0.265±0.02**	**0.291±0.02**	**0.309±0.01**	**0.333±0.01**	**0.363±0.01**	**0.369±0.02**	**0.397±0.01**	**0.404±0.00**	**0.422±0.01**
	ActiveIter-50	0.230±0.01	0.247±0.01	0.289±0.02	0.300±0.01	0.323±0.02	0.361±0.01	0.362±0.02	0.396±0.01	0.399±0.00	0.410±0.01
	ActiveIter-Rand-50	0.219±0.01	0.234±0.01	0.284±0.02	0.289±0.01	0.316±0.01	0.352±0.01	0.360±0.01	0.383±0.01	0.391±0.00	0.402±0.01
	Iter-MPMD	0.217±0.01	0.233±0.01	0.280±0.02	0.293±0.01	0.316±0.02	0.350±0.01	0.361±0.02	0.385±0.01	0.387±0.00	0.400±0.01
	SVM-MPMD	0.005±0.01	0.006±0.01	0.065±0.04	0.043±0.05	0.042±0.06	0.049±0.06	0.082±0.06	0.09±0.06	0.092±0.07	0.131±0.06
	SVM-MP	0.005±0.01	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Precision	*ActiveIter-100*	**0.318±0.01**	**0.352±0.02**	0.379±0.02	**0.396±0.01**	**0.424±0.02**	**0.457±0.02**	**0.460±0.03**	**0.491±0.01**	**0.499±0.01**	**0.518±0.02**
	ActiveIter-50	0.310±0.01	0.327±0.02	0.375±0.02	0.384±0.015	0.410±0.02	0.45±0.02	0.450±0.03	0.489±0.02	0.492±0.01	0.503±0.02
	ActiveIter-Rand-50	0.295±0.01	0.310±0.01	0.369±0.02	0.370±0.01	0.400±0.02	0.440±0.02	0.447±0.02	0.471±0.02	0.480±0.01	0.493±0.01
	Iter-MPMD	0.292±0.01	0.308±0.01	0.364±0.02	0.374±0.01	0.399±0.02	0.439±0.02	0.448±0.02	0.474±0.01	0.475±0.01	0.489±0.01
	SVM-MPMD	0.050±0.15	0.078±0.19	**0.395±0.27**	0.236±0.29	0.180±0.27	0.311±0.31	0.343±0.28	0.424±0.27	0.361±0.29	0.449±0.22
	SVM-MP	0.044±0.13	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Recall	*ActiveIter-100*	**0.186±0.01**	**0.213±0.01**	**0.236±0.01**	**0.253±0.01**	**0.274±0.01**	**0.301±0.01**	**0.308±0.02**	**0.334±0.01**	**0.339±0.00**	**0.356±0.01**
	ActiveIter-50	0.183±0.01	0.198±0.01	0.235±0.01	0.246±0.01	0.267±0.01	0.300±0.01	0.303±0.02	0.333±0.01	0.336±0.01	0.347±0.01
	ActiveIter-Rand-50	0.174±0.01	0.188±0.01	0.231±0.01	0.237±0.01	0.261±0.01	0.293±0.01	0.302±0.01	0.322±0.01	0.330±0.00	0.340±0.01
	Iter-MPMD	0.173±0.01	0.188±0.01	0.228±0.01	0.241±0.01	0.261±0.01	0.290±0.01	0.302±0.01	0.324±0.01	0.327±0.00	0.338±0.00
	SVM-MPMD	0.002±0.01	0.003±0.01	0.036±0.02	0.024±0.03	0.024±0.03	0.027±0.03	0.047±0.03	0.056±0.03	0.053±0.04	0.077±0.04
	SVM-MP	0.003±0.01	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00	0.000±0.00
Accuracy	*ActiveIter-100*	0.976±0.00	0.977±0.00	0.977±0.00	0.978±0.00	0.978±0.00	0.979±0.00	0.979±0.00	0.980±0.00	0.980±0.00	**0.981±0.00**
	ActiveIter-50	0.976±0.00	0.976±0.00	0.977±0.00	0.977±0.00	0.978±0.00	0.979±0.00	0.979±0.00	0.980±0.00	0.980±0.00	0.980±0.00
	ActiveIter-Rand-50	0.975±0.00	0.975±0.00	0.977±0.00	0.977±0.00	0.977±0.00	0.978±0.00	0.979±0.00	0.979±0.00	0.979±0.00	0.980±0.00
	Iter-MPMD	0.975±0.00	0.975±0.00	0.977±0.00	0.977±0.00	0.977±0.00	0.978±0.00	0.979±0.00	0.979±0.00	0.979±0.00	0.980±0.00
	SVM-MPMD	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00	0.980±0.00
	SVM-MP	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	**0.980±0.00**	0.980±0.00

C. Convergence and Scalability Analysis

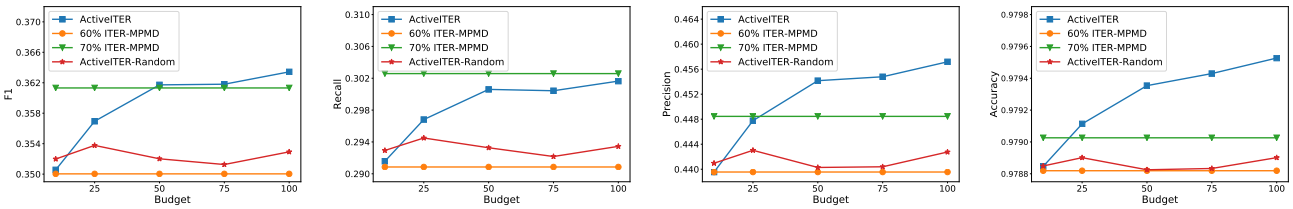
In building the model *ActiveIter*, we propose to use the External Iteration Step (1) in the Section 3.4 essentially to learn both the model variable vector $\mathbf{w}$ and predict the anchor link label vector $\mathbf{y}$. In order to to show such an iteration step can convergence, in Figure 3, we show the label vector changes in each iteration. Here, the x axis denotes the iterations, and the y axis denotes the changes of vector $\mathbf{y}$ in sequential iterations i and $i-1$, i.e., $\Delta\mathbf{y} = \left\|\mathbf{y}^i - \mathbf{y}^{i-1}\right\|_1$. According to Figure 3, we observe that the label vector of *ActiveIter* in the external iteration step can converge in less than 5 iterations for different NP-ratios.

Figure 4 shows the near-linear scaling of *ActiveIter*'s running time in the data size. Here the X axis is the NP-ratio θ, where the value of θ can represent the number of total links as we set before. The slopes indicate linear growth which shows the scalability of *ActiveIter*.

D. Experimental Results with Analysis

The experimental results acquired by different comparison methods are shown in Table III and Table IV mainly. In Table III, Sample-ratio γ is fixed as 60%, and NP-ratio θ changes within $\{5, 10, \cdots, 50\}$. The experimental results of these comparison methods are evaluated by the F1, Recall, Precision and Accuracy metrics respectively. Here, *ActiveIter-50* denotes *ActiveIter* with 50 query budget, and *ActiveIter-100* has a query budget of value 100. At first, we focus on the comparison between SVM-MP and SVM-MPMD. We can find SVM-MPMD has a distinct advantage over SVM-MP with $\theta \in \{5, 10, \cdots, 50\}$. Especially when θ is over 25, the Recall of SVM-MP goes down to 0, and it denotes SVM-MP becomes ineffective in identifying the positive anchor links. However, SVM-MPMD can still work in such a class imbalance scenario. There is only one exception in the table: when $\theta = 5$, the recall of SVM-MP is better than SVM-MPMD. We believe it is caused by very limited positive links and then conduct the supplementary experiment which samples the dataset another time and verifies the recall in $\theta = 5$ is just an accident finally. Therefore, we can verify the effectiveness of the feature vector based on meta diagrams by the comparison of this set of experiments. Besides, the comparison between SVM-MPMD and *Iter-MPMD* demonstrates that *Iter-MPMD* based on a PU learning setting provides a much better modeling for network alignment. However, we can find that the Accuracy of SVM-MPMD is the highest when θ is over 45. Here, we should remind when θ is high enough, SVM-MPMD can not predict positive links correctly which can be found from its Recall. Therefore, in such a class-imbalance setting, Accuracy cannot work well in evaluating the comparison methods performance any more.

Meanwhile, by comparing *Iter-MPMD* with *ActiveIter-Rand-50*, we can discover the metrics obtained by *ActiveIter-Rand-50* can even be worse than *Iter-MPMD* in some cases. In other words, querying labels in a random way will not contribute to the improvement of the prediction



(a) F1 (b) Recall (c) Precision (d) Accuracy

Fig. 5. Performance Analysis under different budgets when $\theta = 50$ and $\gamma = 60\%$ for *ActiveIter* and *ActiveIter-Rand*.

result. From the results, we are also able to observe that *ActiveIter-50* outperforms *ActiveIter-Rand-50* consistently for $\theta \in \{5, 10, \cdots, 50\}$. In addition, the comparison between *ActiveIter-50* and *ActiveIter-100* shows the budget value may have an impact on the performance of *ActiveIter*, whose sensitivity analysis is available in Section IV-E.

In Table IV, we fix θ as 50 and change the sample-ratio γ with values in $\{10\%, 20\%, \cdots, 100\%\}$. From Table IV, we can confirm conclusions verified from Table III are still valid firstly. Furthermore, we can make comparison between *ActiveIter-100* with certain γ and *Iter-MPMD* with $\gamma + 10\%$, When $\theta = 50$, the size of training set will increase by $1,670$, if γ increases by 10%. Between these two methods, besides the γ percentage of training instances shared by both methods, ITER-MPMD uses additional $1,670$ training instances, while *ActiveIter-100* merely queries for additional 100 instances. According to the results, in most of the cases, *ActiveIter-100* with far less training data can still outperform ITER-MPMD with great advantages. For example, when $\gamma = 80\%$, *ActiveIter-100* has metrics that $F1 = 0.3978$, $Precision = 0.4913$, $Recall = 0.3343$ and $Accuracy = 0.9804$. We use *Iter-MPMD* which $\gamma = 90\%$ as a comparison. $F1$, $Precision$, $Recall$ and $Accuracy$ achieved by *Iter-MPMD* are 0.3875, 0.4755, 0.3270 and 0.9797 respectively. In other words, *ActiveIter* can get better performance with around 5% cost in labeling links compared with *Iter-MPMD*.

E. Parameter Analysis

The effects of the parameter budget b on the performance of *ActiveIter* will be analyzed in this part. From Figure 5, we can observe that *ActiveIter* can achieve better prediction results consistently along with querying critical labels continuously, but *ActiveIter-Rand* can not improve prediction output with random labels. This result shows that when b rises, *ActiveIter* is accompanied by better results in all metrics including F1, Precision. Recall and Accuracy. Meanwhile, this performance improvement is continuous and significant because when the b changes within $\{10, 25, 50, 75, 100\}$, the improvement of performance does not slow down. After b exceeds 50, three key metrics including F1, Precision and Accuracy have been higher than *Iter-MPMD* which has $1,670$ more labeled links in the training set. According to the analysis results, with far less (less than 100 additional) training instances, method *ActiveIter* proposed in this paper based on active learning can achieve comparable and even better results than the non-active method *Iter-MPMD* with 1,670 extra training instances.

V. Related Work

Network alignment problem is an important research problem, which has been studied in various areas, e.g., protein-protein-interaction network alignment in bioinformatics [6], [8], [15], chemical compound matching in chemistry [17], data schemas matching data warehouse [11], ontology alignment web semantics [3], graph matching in combinatorial mathematics [10], and figure matching and merging in computer vision [1], [2]. Network alignment is an important problem for bioinformatics. By studying the cross-species variations of biological networks, network alignment problem can be applied to predict conserved functional modules [13] and infer the functions of proteins [12]. Graemlin [4] conducts pairwise network alignment by maximizing an objective function based on a set of learned parameters. Some works have been done on aligning multiple network in bioinformatics. IsoRank proposed in [16] can align multiple networks greedily based on the pairwise node similarity scores calculated with spectral graph theory. IsoRankN [8] further extends IsoRank by exploiting a spectral clustering scheme.

Similarity measure based on heterogeneous networks has been widely studied. Sun introduces the concept of *meta path-based similarity* in *PathSim* [18], where a *meta path* is a path consisting of a sequence of relations. However, the *meta path* suffers from two disadvantages. On one hand, *meta path* cannot describe rich semantics effectively. On the other hand, once numerious *meta paths* are defined, it's challenging to assemble them. Some methods to resolve these deficiencies are proposed later. *Meta structure* [5] applys meta-graph to similarity measure problem, but entities are constrained to be of the same type. Zhao [28] proposes the concept of *meta graph* and extends the idea to recommendation problems which require that entities belong to different types. However, *meta structure* and *meta graph* are proposed for single non-attribute networks. In our *inter-network meta diagram* definition, not only regular node types but also attribute types are involved, and it can be applied to the similarity measure across networks.

For online social networks, network alignment provides an effective way for information fusion across multiple information sources. In the social network alignment model building, the anchor links are very expensive to label manually, and achieving a large-sized anchor link training set can be extremely challenging. In the case when no training data is available, via inferring the potential anchor user mappings across networks, Zhang et al. have introduced an unsupervised

network alignment models for multiple social networks in [26] and an unsupervised network concurrent alignment model via multiple shared information entities simultaneously in [27]. However, pre-labeled anchor links can provide necessary information for understanding the patterns of aligned user pairs in their information distribution, which lead to the better performance than the unsupervised alignment models. Therefore, in [21], [25], Zhang et al. propose to study the network alignment problem based on the PU learning setting.

Active learning is an effective method for network alignment in the face of lacking labeled links which has been previous studied by [9], [19]. The query strategies proposed by Cortés and Serratosa [19] return a probability matrix for different alignment choices which makes the quantification of network alignment straightforward. However, this kind of strategies totally ignore the *one-to-one cardinality constraint* existing in online social networks. Therefore, we provide an innovative query strategy considering *one-to-one cardinality constraint* in *ActiveIter*. Malmi [9] proposes two relative-query strategies TOPMATCHING and GIBBSMATCHING instead of focusing on absolute-query. However, it may not be less challenging for experts to make comparative judgements in online social networks, because the quantity of cantidates corresponding to one node will be huge.

Across the aligned networks, various application problems have been studied. Cross-site heterogeneous link prediction problems are studied by Zhang et al. [23] by transferring links across partially aligned networks. Besides link prediction problems, Jin and Zhang et al. proposes to partition multiple large-scale social networks simultaneously in [24]. The problem of information diffusion across partially aligned networks is studied by Zhan et al. in [20], where the traditional LT diffusion model is extended to the multiple heterogeneous information setting. Shi et al. give a comprehensive survey about the existing works on heterogeneous information networks in [14], which includes a section talking about network information fusion works and related application problems in detail.

VI. CONCLUSION

In this paper, we study the ANNA problem and propose an active learning model *ActiveIter* based on meta diagrams to solve this problem. Meta diagrams can be extracted from the network to constitute heterogeneous features. In our experiments, we verify the effectiveness of meta diagram based feature vectors at first. In the active learning model *ActiveIter*, we propose an innovative query strategy in the selection process to in order to query for the optimal unlabeled links. Extensive experiments conducted on two real-world networks *Foursquare* and *Twitter* demonstrate that *ActiveIter* has very outstanding performance compared with the state-of-the-art baseline methods. *ActiveIter* only needs a small-size training set to build up initially and can outperform the other non-active models with much less training instances.

REFERENCES

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

[2] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 2004.

[3] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies*. 2004.

[4] J. Flannick, A. Novak, B. Srinivasan, H. McAdams, and S. Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome research*, 2006.

[5] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta structure: Computing relevance in large heterogeneous information networks. In *KDD*, 2016.

[6] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. In *RECOMB*. 2008.

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

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

[9] E. Malmi, A. Gionis, and E. Terzi. Active network alignment: A matching-based approach. In *CIKM*, pages 1687–1696, 2017.

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

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

[12] D. Park, R. Singh, M. Baym, C. Liao, and B. Berger. Isobase: a database of functionally related proteins across ppi networks. *Nucleic Acids Research*, 2011.

[13] R. Sharan, S. Suthram, R. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. 2005.

[14] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. Yu. A survey of heterogeneous information network analysis. *CoRR*, abs/1511.04854, 2015.

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

[16] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 2008.

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

[18] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011.

[19] Cortés X and Serratosa F. Active-learning query strategies applied to select a graph node given a graph labelling. In *Graph-Based Representations in Pattern Recognition*, pages 61–70, 2013.

[20] Q. Zhan, S. Wang J. Zhang, P. Yu, and J. Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD*, 2015.

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

[22] J. Zhang, X. Kong, and P. Yu. Predicting social links for new users across aligned heterogeneous social networks. In *ICDM*, 2013.

[23] J. Zhang, X. Kong, and P. Yu. Transfer heterogeneous links across location-based social networks. In *WSDM*, 2014.

[24] J. Zhang and P. Yu. Community detection for emerging networks. In *SDM*, 2015.

[25] J. Zhang and P. Yu. Integrated anchor and social link predictions across partially aligned social networks. In *IJCAI*, 2015.

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

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

[28] H. Zhao, Q. Yao, J. Li, Y. Song, and D. Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *KDD*, 2017.