Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2019.DOI

Diting: An Author Disambiguation method based on Network Representation Learning

LIWEN PENG[1], SIQI SHEN[2], JUN XU[3], YONGQUAN FU[4], DONGSHENG LI[5], AND ADELE LU JIA[6].

[1]School of Computer, National University of Defense Technology, Changsha, Hunan, China (e-mail: pengliwen13@nudt.edu.cn)
[2]School of Computer, National University of Defense Technology, Changsha, Hunan, China (e-mail: shensiqi@nudt.edu.cn)
[3]Ant Financial Services Group, Hangzhou, Zhejiang, China (e-mail: xj169860@antfin.com)
[4]School of Computer, National University of Defense Technology, Changsha, Hunan, China (e-mail: yongquanf@nudt.edu.cn)
[5]School of Computer, National University of Defense Technology, Changsha, Hunan, China (e-mail: dsli@nudt.edu.cn)
[6]College of Information and Electrical Engineering, China Agricultural University, Beijing, China (e-mail: ljia@cau.edu.cn)

Corresponding author: Siqi Shen, Yongquan Fu (e-mail: shensiqi@nudt.edu.cn, yongquanf@nudt.edu.cn).

ABSTRACT It is important to disambiguate names among persons in many scenarios. In this work, we propose an unsupervised method Diting and a semi-supervised method Diting++ for author disambiguation. In Diting, we learn a low-dimensional vector to represent each paper in networks, which are formed by connecting papers with multiple types of relationship (such as co-author). During representation learning, we focus on maximizing the gap between positive edges and negative edges. Further, we propose a clustering algorithm which associates papers to their real-life authors. To make full use of the authorship information, which is easy to obtain from the authors' homepages, we design Diting++ to improve the performance for name disambiguation. Diting++ uses the authorship information listed on the authors' homepages to construct label networks and uses a network representation learning method to learn paper representations based on label networks and other networks. Further, Diting++ uses a semi-supervised clustering method to partition learned paper representations into disjoint groups. Each group belongs to a distinct author. By making use of the label information, the clustering method partitions papers written by the same author in the same group, whereas papers written by different authors locate in different groups. Through extensive experiments, we show that our methods are significantly better than the state-of-the-art author disambiguation methods.

INDEX TERMS network representation learning, network embedding, author disambiguation

I. INTRODUCTION

MANY authors share the same names. When we search for documents about one particular author in the field of literature search, we may get many results (e.g., papers, web pages) containing the author's name. However, even those documents share the same name we search for, they can be different peoples. For example, a search query for the name "Mark Newman" could obtain a physicist who works in the University of Michigan, a computer scientist who works in the same university, and so on. It is hard to determine which the person we care about is. Apart from these, the ambiguous name problem also appears in many other fields, such as law enforcement and bibliometrics science. Table 1 shows a list of names, the number of documents, and the number of actual persons associated with the names. We can see that there are many ambiguous names, and the average number of documents written by one distinct author varies, from 1.1 to 77.0, which makes the ambiguous name problem more challenging.

In this work, we focus on author disambiguation that associates documents (e.g., web pages, papers) to different persons who share an identical name. Author disambiguation, also known as name disambiguation, entity resolution [1], name identification, has long been viewed as a challenging problem by many scholars in many domains [2], [3]. Researchers attempt to explore efficient methods to disambiguate ambiguous names. They collect ambiguous names and related information associated with the names. That

TABLE 1. Examples of name disambiguation in Arnetminer dataset.

Name	#Document	#Person	#Doc/Person	Name	#Document	#Person	#Doc/Person
Ajay Gupta	36	9	4.0	Fan Wang	56	14	4.0
Alok Gupta	57	2	28.5	Fei Su	37	4	9.3
Bin Li	181	60	3.0	Feng Liu	149	32	4.7
Bin Yu	105	17	6.2	Feng Pan	73	15	4.9
Bin Zhu	46	15	3.1	Frank Mueller	101	3	33.7
Bing Liu	182	18	10.1	Gang Chen	177	46	3.8
Bo Liu	124	47	2.6	Gang Luo	47	9	5.2
Bob Johnson	11	7	1.6	Hao Wang	178	48	3.7
Charles Smith	7	4	1.8	Hiroshi Tanaka	40	7	5.7
Cheng Chang	27	5	5.4	Hong Xie	12	7	1.7
Daniel Massey	43	2	21.5	Hui Fang	42	8	5.3
David Brown	61	25	2.4	Hui Yu	32	21	1.5
David C. Wilson	65	5	13.0	J. Guo	13	10	1.3
David Cooper	18	7	2.6	J. Yin	18	7	2.6
David E. Goldberg	231	3	77.0	Jeffrey Parsons	31	2	15.5
David Jensen	53	4	13.3	Ji Zhang	64	16	4.0
David Levine	48	18	2.7	Jianping Wang	37	5	7.4
David Nelson	20	11	1.8	Jie Tang	66	6	11.0
Eric Martin	85	5	17.0	Jie Yu	32	9	3.6
F. Wang	19	17	1.1	Jim Gray	192	6	32.0

information can help distinguish ambiguous names. Albeit author disambiguation has received much attention from the research community, three significant challenges remain.

The first challenge is how to represent papers effectively and flexibly by making use of biographical information such as address and organization, and networking information such as collaboration and citation. Albeit researchers have proposed various feature construction methods [2], most of them [4], [5] use only user-defined heuristics to learn the graph properties from networking information. Moreover, as the online data are becoming more complex and dynamic, feature representation methods are required to be more flexible and extensible for scenarios such as the privacy-preserving scenario in which some information is not available.

The second challenge is how to determine the assignment of papers to its authors. Previously, researchers use supervised methods [6] or unsupervised methods [3], [4] to address this challenge. However, the supervised methods require data labeling, and the unsupervised methods require manual input of the number of unique authors (denoted as K) with an identical name. For example, [2] uses the Bayesian Information Criterion (BIC) to determine K. But this approach may underestimate the actual number [7]. Author disambiguation methods with little human labor are needed.

The third challenge is how to make use of the available information of ambiguous names to assist disambiguation. For some authors, their published papers are listed on their homepages, which can be used to assist the author disambiguation. With such information, we can identify the true authors of the papers in a more effective way. However, it is unclear how to effectively encode such label information and make use of it to improve the performance of author disambiguation.

To address the first two challenges, we propose Diting[1]. Diting uses a network representation learning[2] method which models the relationships between papers as a set of undirected graphs and learns paper representations from these networks jointly. Network representation learning can transform complex information into different paper relationships (the edges in networks), which are easy to be processed through network representation learning methods. And this method can make full use of the information regardless of its completeness, as networks formed based on relationships can be easily added or removed based on data availability. To learn representations effectively, for an ambiguous name, we build the networks among papers associated with the name, rather than a network consisting of all the articles. In Diting, each paper is treated as a vertex in a network, and we take into account the paper relationships such as co-author, co-title, co-summary, and co–organization. Further, we optimize the gap between positive and negative edges and coarsen networks to capture global structure information. The proposed method can learn better network properties for disambiguation than state-of-the-art methods.

To address the second challenge, Diting uses an unsupervised clustering algorithm, which can adaptively determine the assignment of papers to its real-life authors. At the same time, our algorithm does not need to input the number of clusters (K) manually. It can automatically determine the cluster numbers during the algorithm process. We conduct our clustering algorithm on paper representation vectors learned by the network embedding method. Then these papers will be divided into different parts, and the papers in the same

[1]Diting is a magical creature which is good at distinguishing objects in Chinese mythology.

[2]In this work, we interchangeably use network representation learning and network embedding.

part represent that they are written by the truly same author. Our algorithm chooses based on a density metric between the results of two clustering algorithms HDBSCAN [8] and affinity propagation clustering (AP) [9].

To address the third challenge, we propose Diting++, which can make use of the label information to assist disambiguation. For the obtained authorship information (e.g., from authors' homepage), Diting++ constructs a co-label network. In the network, if two papers are written by the same unique author, there exists an edge between them. Otherwise, they are not connected. The co-label network models the authorship information. Diting++ encodes such information into paper representations by jointly learning embeddings on the co-label network and other networks constructed according to paper relationships. Further, besides encoding such information into the learned paper representations, we design a semi-supervised clustering method to partition papers into distinct groups by using the authorship information.

The contributions of this work are listed as follows.

- We propose a novel network representation learning method for author disambiguation, which models multiple types of paper relationships to paper representations.
- We propose an unsupervised clustering algorithm for author disambiguation, which does not require to input the number of authors sharing the same name.
- We improve the network representation learning method via incorporating label information to obtain better paper representations.
- We propose a semi-supervised clustering algorithm to partition papers according to constraints which model authorship information.
- We show through experiments that our methods Diting and Diting++ obtain better results than state-of-the-art methods. The Marco-F1 score of Diting++ method obtained on three widely used datasets is at most 28.8% higher than others and is about 10.9% higher than the best alternative.

The rest of this work is organized as follows. Section II discusses related work, and Section III formulates the author disambiguation problem. Section IV describes our unsupervised approach Diting consisting of an unsupervised network representation learning procedure and an unsupervised clustering procedure. Section V depicts the semi-supervised approach Diting++, which improves the unsupervised approach by taking the label information into account. Section VI displays the experiment results compared with other state-of-the-art methods on three datasets. Section VII concludes this work.

II. RELATED WORK

In this section, we discuss related work regarding author disambiguation and network representation learning.

A. AUTHOR DISAMBIGUATION

Author disambiguation has received much attention from researchers. In general, the author disambiguation task consists of two parts: the feature construction part which prepares features for disambiguation and the assigning part which assigns papers to their real-life authors. Based on the information used for name disambiguation, there are mainly three genres of name disambiguation methods: supervised-based, unsupervised-based, and active-learning-based.

For supervised-based approaches, Han *et al.* [6] learn a model for each name based on supervised data and then use the model to tell whether a paper is written by an author, thus determining the assignment of each paper. Supervised-based approaches require labeled data, so these methods are not widely adopted for name disambiguation.

For unsupervised-based approaches, [2], [4], [10]–[12] model the relationships among papers using various methods and then employ unsupervised clustering to partition and group all the papers. Then each disjoint set of papers is assigned to a distinct person. DISTINCT [10] uses neighborhood relationships and the random walk probability to measure object similarity and then uses an agglomerative hierarchical clustering algorithm to cluster objects into disjoint sets. Tang *et al.* [2] model the relationships between papers using Hidden Markov Random Field and then use Bayesian Information Criterion to determine the number of distinct authors. Finally, they use Xmeans and Kmeans to cluster papers. Zhang *et al.* 2016 [11] propose a Bayesian non-exhaustive classification method for the name disambiguation problem. Lin *et al.* [4] calculate metric confidences and use the most confident metric to cluster papers. Different from them, in this work, we use network representation learning to capture the relationships of papers.

For active-learning-based approaches, an initial name disambiguation result is calculated first, then pairs of objects which need human intervention are presented for labeling. Then the user inputs are used to guide clustering towards better name disambiguation results. For example, [7] models the name disambiguation problem using a pairwise factor graph (PFG) model and an active learning approach. Then it selects top pairs of documents that are most uncertain to improve the results. Zhang *et al.* [13] use user inputs as constraints to guide their name disambiguation method. Imran *et al.* [14] propose a distance-based proximity method and use hierarchical clustering to cluster papers. The active-learning-based methods require human intervention thus are not scalable. Different from these methods, our method Diting++ makes use of the existing labeled information (such as the authors' lists of publications labeled by themselves) without any additional intervention.

Our work uses a network representation learning technique to learn representation for each paper and uses a clustering technique to partition papers into disjoint parts. For the representation learning and clustering procedure, we use both unsupervised and semi-supervised methods.

B. NETWORK REPRESENTATION LEARNING

Network representation learning, also called network embedding, has attracted much interest of researchers [15],

[16]. DeepWalk [17] performs random walks on a network and uses the word2vec [18] method to learn representations. LINE [19] models the first- and second-order proximity of nodes. GraRep [20] and [21] model node proximity via factorizing multiple high-order adjacent matrixes. Node2Vec [22] improves DeepWalk via combining depth-first and breadth-first search. MMDW [23] proposes the max-margin DeepWalk. Feng *et al.* [24] propose an approach to embed scale-free networks. Yang *et al.* [25] compute high-order matrixes via approximation. HARP [26] iteratively merges nodes into higher-level nodes and then uses the high-level representations as the initialization value. GraphGAN [27] and ANE [28] learn graph representations through adversarial training. GCN [29] uses a variant of convolutional neural networks to learn the hidden layer representations. GAT [30] leverages masked self-attention layers to address the shortcoming in prior methods based on graph convolutions. GraphSage [31] presents an inductive framework that can learn representations for previously unseen nodes in the network. For heterogeneous networks, PTE [32] constructs word-word, word-document, and word-label networks and learns them jointly. PathSim [33] studies the similarity search in heterogeneous information networks. Mepapath2vec [34] learns graphs based on meta-path. Tu *et al.* [35] propose a method to learn hyper-graph. Different from all the above methods, we learn network representations through jointly analyzing multiple relationship networks to solve the author disambiguation problem.

This paper is an extension of our previous work [12]. In this work, besides considering more types of information, we construct a co-label network using the authorship information of papers to learn better representations for disambiguation. Further, we propose a semi-supervised clustering algorithm with the authorship information to obtain better disambiguation results.

III. PROBLEM FORMULATION

In this section, we formulate the author disambiguation problem by using the network representation learning and a clustering algorithm. We model the related information of papers for disambiguation by constructing multiple networks. Then we get the low-dimensional paper representations through network representation learning. Thus we can distinguish different persons sharing the same ambiguous name by dividing the learned paper representations into disjoint parts.

For an ambiguous name n, we denote the papers containing the name n as $\mathcal{P}^n = \{p_1^n, p_2^n, ..., p_m^n\}$, where p_i^n is the ith paper in $\mathcal{P}^n$. There can be more or fewer paper properties depending on the availability of information. The properties of publications considered in this work are described in Table 2. This information of papers can be obtained from online libraries, e.g., IEEE. For each ambiguous name n, we seek to distinguish the actual real-life persons who share the name n and assign true author label to each paper.

Then we construct our networks $G = \{G_a, G_t, G_s, ...\}$ based on the name n and the papers in $\mathcal{P}^n$. We denote the network as $G_T = (V, E^T)$, in which E^T represents the edge set, and V represents the vertex set in the network. The variable T means only vertices that have the T (co-author, co-title, etc.) type relationship can form an edge $e \in E^T$ in network G_T.

TABLE 2. Properties of publication p_i considered in this work.

Attribute	Description
$p_i.authors$	the authors' names for p_i
$p_i.title$	the title of p_i
$p_i.venue$	the published conference/journal of p_i
$p_i.summary$	the summary of p_i
$p_i.organization$	the affiliation of the author of p_i
$p_i.year$	the publication year of p_i
$p_i.label$	the true author label for p_i

In this work, we treat each paper $p_i \in \mathcal{P}^n$ as a vertex $v_i \in V$ in the network and use different relationships between papers to construct the edges in different networks. Then we use network embedding method to obtain paper representations via learning the low-dimensional latent representation vector for each vertex $v_i \in V$ in all the networks (each paper corresponds to one vertex). After the network embedding, we convert all vertices in the networks into low-dimensional vectors. It can be considered that these vectors significantly preserve the edge properties and connection relationships of the vertices. And then we use a clustering algorithm to divide the vertices (papers) into different groups according to the characteristics learned by the vectors. We reasonably regard the papers that are clustered into the same group are written by the same actual person.

IV. DITING: AN UNSUPERVISED AUTHOR DISAMBIGUATION APPROACH

The proposed method Diting in our paper consists of three parts: network construction, network representation learning, and a clustering algorithm. For papers in $\mathcal{P}^n$ sharing the ambiguous name n, we construct networks based on $\mathcal{P}^n$ and their relationships (Section IV-A). Then we use network representation learning to learn paper embeddings (Section IV-B). After the clustering procedure, all papers sharing the ambiguous name n in $\mathcal{P}^n$ are partitioned into disjoint sets. Each set is viewed as all the papers written by a distinct author (Section IV-C). The overview of the Diting approach is depicted in Fig. 1.

A. NETWORK CONSTRUCTION

Given all names $N = \{n_1, n_2, ..., n_m\}$ and all papers $\mathcal{P}$ published by the authors in a dataset, we can construct a big network based on the whole $\mathcal{P}$. However, the set $\mathcal{P}$ is often huge, then the set of vertices in the big network is large, which causes the network representation learning procedure to be ineffective. Therefore, we construct multiple paper-paper networks for each name n based on $\mathcal{P}^n$, instead of $\mathcal{P}$. We show in Section VI-G that learning representations on $\mathcal{P}^n$ is better than on $\mathcal{P}$.

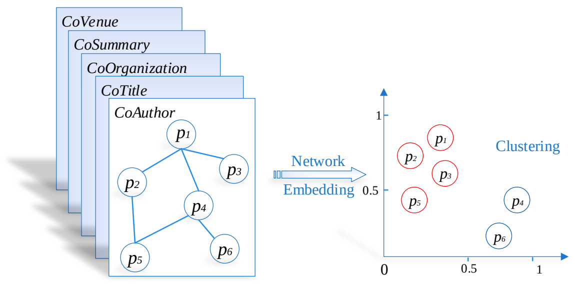


FIGURE 1. The overview of the Diting approach.

For name n, we create multiple networks connecting papers in $\mathcal{P}^n$ based on different paper relationships, as specified in Section III. The following paragraphs will describe in detail the construction process of six types of network in our work: co-author, co-title, co-summary, co-venue, co-organization, and co-year. In these networks, V is the vertex set, and each vertex v_i represents a paper p_i.

The *co-author network* is $G_a = (V, E^a)$, where E^a is the edge set. Given two papers p_i and p_j, and their authors, the edge weight w^a_{ij} between p_i and p_j is the number of overlapping authors of these two papers (not counting n). For example, if paper p_i has authors A_1, A_2, A_3, A_4, and paper p_j has authors A_1, A_2, A_3, A_5, and the name A_1 is the ambiguous name, then the overlapping authors for p_i and p_j are A_2, A_3. So the edge weight w^a_{ij} is 2.

The authors may write their names in the abbreviation form. For example, "Mark Twain" may be written as "M. Twain." Sometimes, the authors may write their names in a different order. For example, "Frank Mueller" may be written as "Mueller Frank." A partially shortened name is also extraordinarily ordinary, like "C. J. Lin" and "C. J. Lint." To deal with those problems, we adopt the method proposed in [5] to assist the calculation of w^a_{ij}.

The intuition behind the co-author network is that, if two papers have many co-authors, the probability that the same person writes them is high. Even if they do not have a co-author, but their neighboring papers have many co-authors, these two papers are likely to be written by the same person.

The *co-title network* $G_t = (V, E^t)$ models the relationship between paper titles. The word set used in the title of p_i is denoted as t_i (stop words are removed). We use NLTK [36] to transform each word in t_i to its word origin (e.g., networking to network). Then, the edge weight w^t_{ij} between p_i and p_j is the number of words that overlap between t_i and t_j. That is, $w^t_{ij} = |t_i \bigcap t_j|$. We use NTEE [37] to get title words' embeddings. If the similarity between two words' vectors is higher than a threshold, these two words are counted as the same.

The *co-summary network* $G_s = (V, E^s)$ models the relationship between paper summaries. For a paper p_i, we use the method proposed in [4] to extract a fixed number of words from the abstract of p_i as its summary s_i. Then, w^s_{ij} is calculated in the same way as w^t_{ij} by using s_i and s_j. For the co-title and co-summary networks, we assume that if two papers have similar usage of words in their titles and summaries, they may be written by the same person.

The *co-venue network* $G_v = (V, E^v)$ models the relationship among paper venues (e.g., conference, journal). To deal with acronym and abbreviation, we collect about 20,000 venues' full names from the Internet and obtain the full name of each paper' venue. After that, we use the same procedure as the co-title network to calculate w^v_{ij}. For the co-venue network, if two papers are published in similar research areas, their authors may be the same one.

The *co-organization network* $G_o = (V, E^o)$ models the relationship among author affiliations. Given a paper p_i, the word set used in the authors' organization name is denoted as o_i. The weight w^o_{ij} measures the co-organization similarity and is calculated in the same way as w^t_{ij} by using o_i and o_j. For the co-organization network, if the two same-name authors are from the same organization, they may be the same person.

The *co-year network* $G_y = (V, E^y)$ models the relationship between the papers' publication date. If two papers are published in the same year, then we add an edge between them in the co-year network, and their weight w^y_{ij} is set to 1.

Our method is flexible and extensible because a paper relationship can be added or removed easily as a network. For example, in a privacy-preserving context, only author and title information could be available, then two networks based on the two relationships can be created. As another example, if the citation relationships of papers are available, a citation network could be built.

There could be more ways to capture the relationships between papers. For example, a paper-word network could be created based on the abstract of papers. As another example, a network could be created among names and papers based on research topics. We plan to explore those networks in the future.

B. NETWORK REPRESENTATION LEARNING

Generally, a network representation learning problem is defined as: given a graph $G = (V, E)$ wherein V represents the node set, and E represents the edge set, the aim is to find a function $V \rightarrow R^l$ that maps each node to an l-dimensional ($l \ll |V|$) vector which captures its structural properties.

Given the six networks created in Section IV-A, the network embedding algorithm generates for each paper p_i ($v_i \in V$) an l-dimensional vector $d_i \in R^l$ through jointly learning on the networks. The basic idea is that vertices which are close in networks are similar to each other and should be encoded closely in R^l. Because the goal of the embedding procedure is to obtain network representations which are useful for author disambiguation, we do not seek to reconstruct the whole network by minimizing the KL-divergent between the probability and weight of edges. Instead of using softmax function [17], [22] to calculate the score (e.g., probability) of an edge, we define the score of an edge e_{ij} connecting vertex v_i and v_j as follows (d_i and d_j are corresponding vectors):

$$s(v_i, v_j) = \frac{d_i^T d_j}{\|d_i\| \cdot \|d_j\|}. \tag{1}$$

For $e_{ij} \in E$, $s(v_i, v_j)$ is defined as the cosine similarity between the two vertices. As the networks in this work are undirected, the cosine similarity ensures that $s(v_i, v_j) = s(v_j, v_i)$. Given a vertex k that is not connected to i, we denote the non-existing edge as ne_{ik} and the set of all non-existing edges as NE. The score of ne_{ik} is defined as the cosine similarity $s(v_i, v_k)$ between v_i and v_k. Then, we model the loss for (i, j, k) triple as:

$$L(i,j,k) = max(\epsilon, s(v_i,v_j) - s(v_i,v_k)). \tag{2}$$

We set $\epsilon = 0.01$ to avoid the case of divided by zero. The probability can be viewed as the score gap between e_{ij} and ne_{ik}. The intuition for (2) is that the probability of preserving the score order between e_{ij} and ne_{ik} should be large to reflect the edge status. Assuming that all the probabilities of score orders are independent, we aim to maximize the objective function defined in (3). For convenience, we transform (3) into (4) by minimizing the negative log objective $O = -lnF$.

$$F = \prod_{\substack{(v_i,v_j)\in E \\ (v_i,v_k)\in NE}} L(i,j,k) \tag{3}$$

$$\begin{aligned} O &= -ln(\prod_{\substack{(v_i,v_j)\in E \\ (v_i,v_k)\in NE}} L(i,j,k)) \\ &= \sum_{\substack{(v_i,v_j)\in E \\ (v_i,v_k)\in NE}} -ln(max(\epsilon, s(v_i,v_j) - s(v_i,v_k))) \end{aligned} \tag{4}$$

The calculation of the loss function $L(i, j, k)$ requires sampling one existing edge e_{ij} and one non-existing edge ne_{ik}. When training, for each network with $|E|$ edges, we sample $T \times |E|$ triples. In Section VI-G, we evaluate the performance of different T values. For each (i, j, k) triple, we denote the existing edge e_{ij} as positive edge and the non-existing edge ne_{ik} as negative edge. Our sampling methods for positive and negative edges are described as follows.

- Positive sampling: the probability of sampling e_{ij} is proportional to the edge weight w_{ij}. The edge having a larger weight connecting the current node is more likely to be chosen as the positive sample.
- Negative sampling: for node i's non-neighbor nodes, the probability of sampling ne_{ik} is proportional to the value of an exponential function e^{-s}, where $s = s(v_i, v_k)$. Through negative sampling, the more similar between i and its non-connecting node k, the fewer sampling probability of the non-existing edge ne_{ik}.

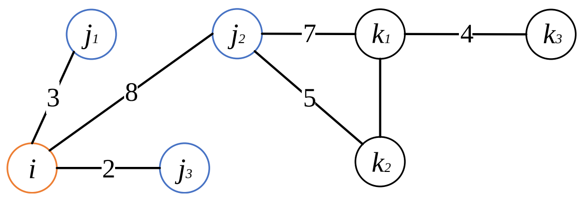


FIGURE 2. An example of positive and negative edge sampling.

An example of edge sampling is depicted in Fig. 2. As in Fig. 2, supposing vertex i is the current node, we aim to sample one positive edge and one negative edge for i. Vertices j_1, j_2, j_3 have existing edges with i, and e_{ij_2} has the largest weight among them. Vertices k_1, k_2, k_3 have non-existing edges with i, and k_3 is most dissimilar with i. According to the above sampling methods, the probability of sampling e_{ij_2} as the positive edge is the largest and the probability of sampling ne_{ik_3} as the negative edge is the largest. So we sample the triple (i, j_2, k_3). Our sampling method is different from DeepWalk, which samples negative vertices according to their weights. We focus on keeping the vertices that are least likely to have connected edges away from each other as far as possible.

We denote the objective functions of the co-author, co-title, co-venue, co-summary, co-organization, and co-year networks as $O_a, O_t, O_v, O_s, O_o, O_y$, respectively, and the negative edge set of them can be denoted as $NE^a, NE^t, NE^v, NE^s, NE^o, NE^y$, respectively. These different objective functions $O_{\mathcal{N}}, \mathcal{N} \in \{a, t, v, s, o, y\}$ are described in (5). The final objective function O_{Diting} is defined in (6), where $w_i, i \in \{a, t, v, s, o, y\}$ are the weights of the corresponding networks. L^2 is the l^2 regulation of the learned parameters to avoid overfitting. For model training, we use stochastic gradient descent (SGD) to optimize the objective function.

$$O_{\mathcal{N}} = \sum_{\substack{(v_i,v_j)\in E^{\mathcal{N}} \\ (v_i,v_k)\in NE^{\mathcal{N}}}} -ln(max(\epsilon, s(v_i,v_j) - s(v_i,v_k)))$$

$$\mathcal{N} \in \{a,t,v,s,o,y\} \tag{5}$$

$$O_{Diting} = \sum_{i\in\{a,t,v,s,o,y\}} w_i O_i + \lambda L^2 \tag{6}$$

To better capture the global properties of networks, we coarsen our networks to get the pre-trained vectors of all vertices. The procedure is depicted in Fig. 3. We coarsen the network by merging low-degree vertices. We randomly select a low-degree vertex v_i and merge it with its high-degree neighbor v_j. For vertices having the same degree, vertices with a high-weight edge are preferred. During the process, the edge weights of vertices which are not involved in the merging remain unchanging. The merging procedure ends when the number of nodes is 1/3 of the original network.

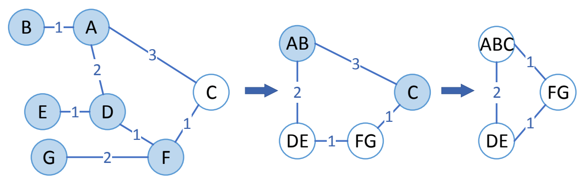


FIGURE 3. Illustration of the coarsening procedure (shaded neighboring nodes are merged).

This idea is inspired by [26]. Different from its edge collapsing, we coarsen nodes according to node degree instead of randomly, and [26] deals with only one network instead of multiple. We first coarsen the most important network (i.e., co-author), then other networks are coarsened in the same way. After that, the node representations in the coarsened networks are learned. Then, these representations are used as the initial value for the vertices in all the networks. Finally, network representation learning is performed on the original graphs to obtain the final representations.

C. CLUSTERING ALGORITHM

In Diting, for each ambiguous name, the clustering algorithm divides the papers into non-overlapping clusters, and each cluster belongs to a distinct author. Thus we can distinguish the different persons sharing the same ambiguous name. To reduce the human labors regarding labeling the true authorship information, Diting adopts an unsupervised clustering method.

The problem of clustering papers in Diting has two special features. Firstly, the number of clusters before clustering is completely unknown to us. As shown in Table 1, the number of publications per person is skewed. Some could publish more than 70 papers, while some publish several. This indicates that the size of cluster belonging to each author is skewed. For skewed clusters, the most well-known clustering algorithms are K-means [38], AP [9], and DBSCAN [39]. We do not adopt K-means in the unsupervised method Diting as it requires the input of the number of authors. Secondly, isolated papers, which actually represent different single authors, may be merged into one cluster. In the networks constructed by Diting, we find that there are many isolated paper nodes in the dataset. Therefore, compared to other agglomerative clustering methods, the agglomerative hierarchical clustering algorithm is more suitable for our problem. It first treats each paper as a cluster center, and then repeatedly merges the most similar clusters until convergence. Thus it can reduce the possibility of merging isolated papers into one cluster comparing to other agglomerative clustering methods such as DBSCAN.

In Diting, we combine two methods HDBSCAN [8] and AP to obtain our clustering results. HDBSCAN is an agglomerative hierarchical clustering algorithm. It creates hierarchical clusters and then condenses the small clusters to larger ones. Albeit HDBSCAN is better than DBSCAN in our scenario, we find that using HDBSCAN alone is not enough to partition papers well. HDBSCAN prefers to cluster data into a small number of clusters, as it condenses clusters into larger ones. Thus a person with a small publication number may be merged with others. To address this problem, we use AP to cluster papers as well. AP initially treats all nodes as cluster centers and exchanges responsibility and availability messages between node pairs until convergence. AP tends to produce more clusters, which may treat a person as multiple persons via dividing a person's publications into multiple clusters. As the two methods (HDBSCAN and AP) both have advantages and disadvantages, we use the results of them together to mitigate their negative effects.

For clustering results, we expect that the publications of the same person are closely clustered together, while the publications of others are far away. The SD index [40] matches our requirement and is defined as follows:

$$SD = \max_{i,j} d(c_i, c_j)scat + dis$$

$$scat = \frac{1}{M} \sum_i \|\sigma(c_i)\| / \|\sigma(D)\|$$

$$dis = \frac{\max_{i,j} d(c_i, c_j)}{\min_{i,j} d(c_i, c_j)} \sum_i (\sum_j d(c_i, c_j))^{-1}$$

$$\sigma(c_i) = \frac{1}{n_i} \sum_j d(p_j, c_i)^2. \tag{7}$$

In the above formulas, $d(c_i, c_j)$ is the distance of two cluster centers c_i and c_j. M is the number of clusters. $\sigma(c_i)$ and $\sigma(D)$ are the variances of cluster c_i and the whole dataset D. n_i is the number of papers in cluster c_i, and $d(p_j, c_i)$ is the distance between paper p_j and the cluster center c_i. The first term of SD evaluates how closely papers are together based on variance, and the second term measures the difference among distinct authors based on distances between clusters. We denote the SD value of the clustering results for HDBSCAN and AP as SD_H and SD_A, respectively.

SD may lead to inferior performance when paper density varies. To mitigate this problem, for each paper p_i, we obtain the top-k closest papers $\{p_{i_1}, ... p_{i_k}\}$. The density μ_i of p_i is defined as $\sqrt{\sum_{j=1}^{k} d(p_{i_j}, p_i)}$. Given a density threshold m, we count the total number of times $\mu_i < m$ as y_d. If $y_d > n/2$ (n is the number of papers), it indicates that the average paper density is high. In this case, if $SD_H \leq SD_A$, then the result of HDBSCAN is used as the final clustering result. Otherwise, AP is used.

The unsupervised clustering algorithm procedure in this paper is depicted in Algorithm 1.

V. DITING++: A SEMI-SUPERVISED AUTHOR DISAMBIGUATION APPROACH

It is common for researchers to maintain and organize their publication records online by posting them on some websites (e.g., university website). This publication information

Algorithm 1: The unsupervised clustering algorithm in Diting.

```
Input: papers' representation vectors $\{d_1, d_2, ...d_n\}$,
       threshold $m$, $SD_H$, $SD_A$
Output: clustering results
$y_d = 0$
for each $d_i$ do
    $dist = \min \sqrt{\sum_{l=1}^{k} distance(d_i, d_l)}$
    if $dist < m$ then
        $y_d \leftarrow y_d + 1$
    end
end
if $y_d > n/2$ and $SD_H \leq SD_A$ then
    adopt the results of $HDBSCAN$ algorithm
else
    adopt the results of $AP$ algorithm
end
```

maintained by the authors can be utilized to assist the author disambiguation procedure. For example, given a set of papers $\mathcal{P}^n$ associated with an ambiguous name n for disambiguation, we denote the true authors sharing the name n as $t_1^n, t_2^n, ..., t_i^n$. If we can locate the homepage of a true author t_i^n, then we can quickly know that papers $P_{t_i^n}$ listed in the homepage are published by the author with high confidence, while the other papers $\mathcal{P}^n - P_{t_i^n}$ may not be published by the author. In the following section, we call this type of authorship information as label information.

By taking consideration of such authorship information, we design Diting++, a semi-supervised method to solve the disambiguation problem. The same as the Diting method, Diting++ consists of three parts: network construction, network representation learning, and a clustering algorithm. In addition to modeling other paper relationships in networks (Section IV-A), the network construction part in Diting++ also models the authorship information about papers (Section V-A). Then a network representation learning method is used to learn paper embeddings (Section V-B). Further, Diting++ introduces a semi-supervised clustering method to partition papers into disjoint sets (Section V-C). The overview of Diting++ is depicted in Fig. 4. We first construct the multiple networks using different relationships between papers and then get the paper representation vectors by a network representation learning technique. At last, we devise a semi-supervised clustering method to distinguish different authors.

A. NETWORK CONSTRUCTION

Similar to Diting, Diting++ models the relationships among papers using networks. For each ambiguous name n, Diting++ builds networks for papers in $\mathcal{P}^n$. Besides modeling the six relationships (e.g., co-author, co-summary) as that of Diting, Diting++ models the true authorship information using a co-label network. We describe the construction process of the co-label network in the following paragraph.

The *co-label network* $G_l = (V, E^l)$ models the authorship information among papers in $\mathcal{P}^n$ sharing with the ambiguous name n. V is the set of vertices, that is the set of all papers. E^l is the edge set of G_l. For two papers p_i and p_j associated with the name n, if their authors are the same person, an edge e_{ij} is created between the two papers in the co-label network, and the edge weight w_{ij}^l is set to 1. Otherwise, the weight w_{ij}^l is set to 0. Compared to the networks constructed by other paper relationships, the label information is extracted from places with high credibilities, such as the author's homepage. Compared to other forms of paper relationships, the true authorship contains more information for disambiguation. Thus, with the introduction of the co-label network, Diting++ encodes into paper representations with more information for author disambiguation.

B. NETWORK REPRESENTATION LEARNING

The homepage information of authors may be located on many different websites, so there may be only part of the information that we can easily obtain, and there is also other information that we do not obtain. For these reasons, we choose the network representation learning to deal with such information. Our method can model the partial information as edges in the label network and encode this partial information as much as possible. In this section, we use network representation learning to obtain paper representations based on the multiple networks constructed in Section V-A and Section IV-A. For each paper p_i (vertex v_i in network), we generate the low-dimensional vector $d_i \in R^l$ by maximizing the probability of existing edges appearing in the networks.

We call an existing edge between vertices v_i and v_j as a positive edge and a non-existing edge between vertices v_i and v_k as a negative edge. Similar to the other networks such as the co-author network, when learning representations on the co-label network G_l, we aim to make sure that the score $s(v_i, v_j)$ of a positive edge e_{ij}^l is larger than the score $s(v_i, v_k)$ of a negative edge ne_{ik}^l. We formulate the objective function for the co-label network representation learning in (8). In the formula, E^l is the set of positive edges, and NE^l is the set of negative edges in the co-label network.

$$O_l = \sum_{\substack{(v_i,v_j)\in E^l \\ (v_i,v_k)\in NE^l}} -ln(max(\epsilon, s(v_i, v_j) - s(v_i, v_k)) \quad (8)$$

To reduce the computation cost of representation learning, we use positive sampling and negative sampling techniques introduced in Section IV-B. To learn both the unsupervised features (such as co-author relationship) and the supervised feature (such as authorship information) jointly, Diting++ reformulates the objective function defined in (6) as (9), which takes the labeled authorship information into account. The parameters w_i, $i \in \{l, a, t, v, s, o, y\}$ represent the weights of different networks. During training, we use stochastic gradient descent to optimize the objective function.

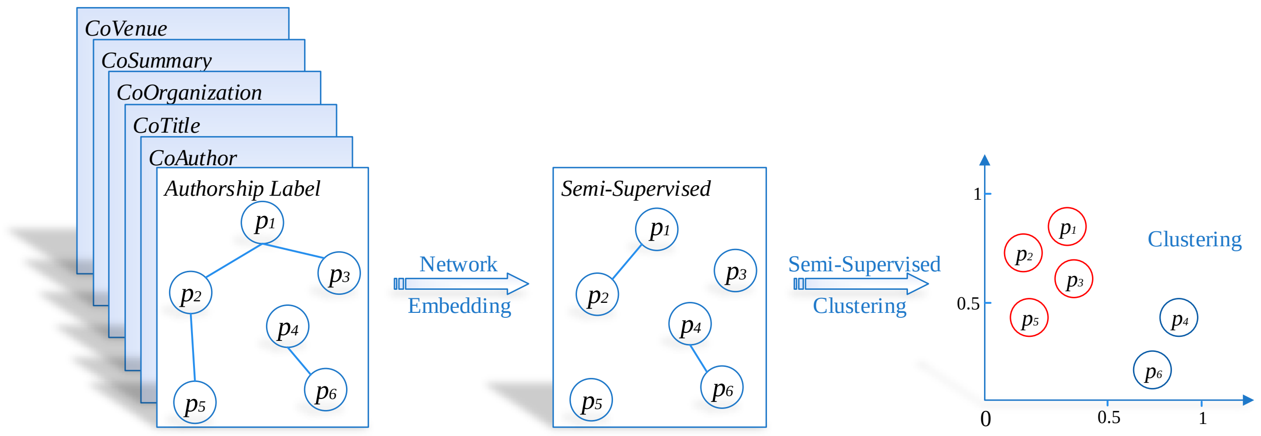


FIGURE 4. The overview of the Diting++ approach.

$$O_{Diting++} = \sum_{i \in \{l,a,t,v,s,o,y\}} w_i O_i + \lambda L^2 \quad (9)$$

C. SEMI-SUPERVISED CLUSTERING

Diting uses an unsupervised clustering method (see Section IV-C) to group and partition the papers of an ambiguous name into disjoint sets. Each set of papers is viewed as all the publications of a distinct author. This method can identify authors without human labors and the actual number of authors sharing the name. However, the unsupervised clustering method does not take into account the fact that much information useful for disambiguation is easy to be obtained. In this section, Diting++ adopts a semi-supervised clustering algorithm by taking such ground-truth authorship information into account to improve the performance of paper clustering.

Diting++ adopts a Kmeans-based semi-supervised clustering method to improve the results of the author disambiguation problem. In the following, we introduce two types of links used in Diting++ to assist the clustering method, and then we describe the clustering algorithm procedure in detail.

1) Must and Cannot Links

Diting++ improves author disambiguation performance by using Must-Link and Cannot-Link constraints. Must-Link and Cannot-Link are collections of node pairs. In a given dataset $\mathcal{P}^n$, and $p_i, p_j \in \mathcal{P}^n$, if p_i and p_j are written by the same unique author, then we add (p_i, p_j) into the Must-Link collection $\mathcal{ML}$. If p_i and p_j are not written by the same true author, then we add (p_i, p_j) into the Cannot-Link collection $\mathcal{CL}$. For papers in dataset $\mathcal{P}^n$, if the labels of two papers are identical, this paper pair is in the Must-Link collection; otherwise, this pair is in the Cannot-Link collection.

We describe the construction of the Must-Link and Cannot-Link collections using an example shown in Fig. 5. For an ambiguous name A (the true persons sharing with the

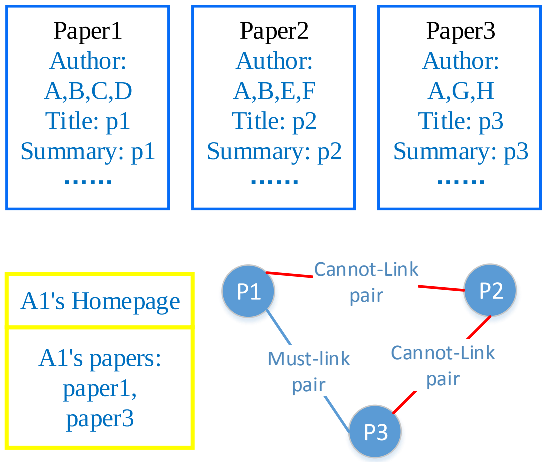


FIGURE 5. An example of Must-Link and Cannot-Link constraints.

name may be $A_1, A_2, ..., A_i$) and all the papers p_1, p_2, p_3 associated with name A in the dataset, we get the authors, titles, summaries, and other information about papers. Moreover, we can locate the homepage of person A_1, and the papers written by A_1 are listed on the homepage. By analyzing such information, we can construct the Must-Link collection $\mathcal{ML}$ and the Cannot-Link collection $\mathcal{CL}$. From the homepage of A_1, we find that p_1 and p_3 are truly written by the same person, so we add (p_1, p_3) in $\mathcal{ML}$. And we also know that p_2 is not written by A_1, so we add $(p_1, p_2), (p_2, p_3)$ in $\mathcal{CL}$. Through these two constraints, we can make better use of the label information to improve our clustering procedure.

2) Semi-supervised clustering for disambiguation

In this section, we propose a semi-supervised clustering method by using the Must-Link and Cannot-Link constraints. Through the network representation learning proposed in Section V-B, the representation of each paper is learned.

Then the semi-supervised clustering method aims to partition the learned representations into non-overlapping parts, each disjoint set of papers corresponds to a unique author.

Diting++ uses a Kmeans-based method to cluster papers. Firstly, it selects K papers in the dataset as cluster centers. The selection of papers as cluster centers is based on their distance to other cluster centers. Moreover, for any paper pair p_i and p_j, if they are written by the same author, that is $(p_i, p_j) \in \mathcal{ML}$, then at most one of the papers can be selected as cluster center. Secondly, during each iteration of the clustering algorithm, the papers are assigned to its closest cluster, which does not violate the Cannot-Link constraints. At the end of each iteration, the cluster centers are updated. The above iterations are repeated until convergence, and the clustering results are returned. The main flow of the algorithm is shown in Algorithm 2. In the following paragraphs, we describe the selection of cluster centers and the distance metric in detail.

When selecting the cluster centers, we must ensure the Must-Link constraints are satisfied. Besides, Diting++ enhances the selection of cluster centers by using an improved KMeans++ algorithm.

The main process of selecting cluster centers by KMeans++ algorithm is to select a cluster center c_1 from the dataset $\mathcal{P}^n$ randomly and then calculate the distance between c_1 and all the other vertices. The vertex having the largest distance with c_1 will be selected as the next cluster center. When the current cluster center is c_{i-1}, the probability for selecting v_i as the next center is calculated as follows.

$$p(v_i, c_{i-1}) = \frac{d(v_i, c_{i-1})^2}{\sum_{v' \in \mathcal{P}^n} d(v', c_{i-1})^2} \tag{10}$$

In the above formula, $d(v_i, c_{i-1})$ calculates the distance between vertex v_i and the cluster center c_{i-1}. In Diting++, we use the cosine similarity as a measure of the distance between two paper vertices. The distance measurement method calculating the similarity between any two papers v_i and v_j is defined as (d_i and d_j are corresponding paper vectors):

$$d(v_i, v_j) = \frac{1}{2} + \frac{d_i \cdot d_j}{2 \cdot \|d_i\| \cdot \|d_j\|}. \tag{11}$$

However, the KMeans++ algorithm is not suitable for our disambiguation problem. In these datasets, there exist multiple isolated papers associated with an ambiguous name. This means the isolated papers which are not far enough from the center of a cluster are very likely to be merged into the cluster rather than forming a new cluster. For example, as shown in Fig. 6, the black dots in the graph represent the real cluster centers. If we use the KMeans++ algorithm to select the cluster centers, supposing A is chosen as the first cluster center, then the algorithm will choose B as the second cluster center rather than E. Because the distance between A and B is larger than the distance between A and E. This may lead to the situation that the publications of a unique author are partitioned into two disjoint clusters.

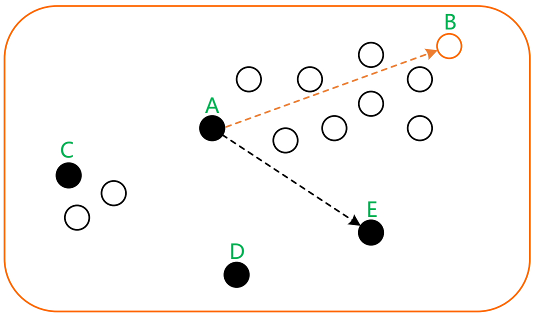


FIGURE 6. The problem of KMeans++ algorithm in selecting cluster centers.

Because of the above reason, we improve the selection of cluster centers by assigning more probability to select isolated papers as cluster centers. Diting++ uses the following formula to calculate the probability of v_i to be selected as the next cluster center when the current cluster center is c_{i-1}. The measure of distance is the same as (11).

$$\begin{aligned} q(v_i, c_{i-1}) &= \frac{1}{2} \cdot \frac{d(v_i, c_{i-1})}{\sum_{v' \in \mathcal{P}^n} d(v', c_{i-1})} \\ &+ \frac{1}{2} \cdot \frac{\sum_{j=1}^{min(5,|\mathcal{P}^n|)} D_j^{v_i}}{min(5, |\mathcal{P}^n|)} \end{aligned} \tag{12}$$

The first term in (12) is to calculate the basic probability of selecting v_i as the next cluster center among all the paper vertices. The second term is to increase the probability that isolated papers are selected as cluster centers, which can effectively avoid the situation that a big cluster is divided into multiple clusters. $D_j^{v_i}$ is the distance between v_i and its jth closest node. We use the average distance of its top-5 closest neighbors to obtain a robust estimation. If v_i is an isolated paper, its second term in (12) is larger than vertices that are not isolated, so the probability of selecting the isolated papers will be higher.

This clustering method is inspired by semi-supervised clustering methods such as COP-Kmeans [41]. However, different from COP-Kmeans, we propose a new distance metric (11) according to characteristics of the data, while COP-Kmeans uses Euclidean Distance. Besides, COP-KMeans chooses the K initial clustering centers randomly, while our clustering algorithm chooses the K initial centers according to (12). On the other hand, our algorithm uses the must-link constraints in the center choosing process and uses the cannot-link constraints in the node assigning process, while COP-KMeans uses both constraints in the node assigning process.

VI. EXPERIMENTAL RESULTS

In this section, we compare our methods with other disambiguation methods and evaluate the impact of different network embedding methods and different clustering methods. We find that our unsupervised method Diting can obtain at least 5.7% better Marco-F1 result than the other author disambiguation methods, and our semi-supervised

Algorithm 2: The semi-supervised clustering algorithm in Diting++.

```
Input: set of papers $\mathcal{P}^n = \{v_i\}_{i=1}^N$, set of Must-Link constraints $\mathcal{ML} = \{(v_i, v_j)\}$, set of Cannot-Link constraints
        $\mathcal{CL} = \{(v_i, v_j)\}$, number of clusters $K$
Output: Disjoint $K$-partition of $\mathcal{P}^n$
choose an initial center $c_1$ randomly from $\mathcal{P}^n$.
repeat
    choose the next center $c_{i+1}$, selecting $c_{i+1} = v' \in \mathcal{P}^n$ where $v'$ is the paper having the maximum $q(v', c_i)$ value.
    if $\exists_{j\in(1,i)}(c_{i+1}, c_j) \in \mathcal{ML}$ then
        drop the selected cluster center $v'$, choose another $v'$ which has the second maximum $q(v', c_i)$.
    end
until get a total of $K$ cluster centers $C = \{c_1, c_2, ..., c_K\}$;
repeat
    for each paper $v_i$ in $\mathcal{P}^n$ do
        assign it to the closest cluster $C_j$ if $\forall_{v_j\in C_j}(v_i, v_j) \notin \mathcal{CL}$.
    end
    for each cluster $C_i$ do
        update its center by averaging all of the papers that have been assigned to it.
    end
until convergence;
return the clustering results $C = \{C_1, C_2, ..., C_K\}$
```

method Diting++ can obtain at least 10.9% better Marco-F1 result. Besides, Diting++ can obtain at least 10.9%, at best 23.3% better Macro-F1 result than other network embedding methods (e.g., node2vec, PTE) for disambiguation. In addition, the unsupervised clustering algorithm in Diting and the semi-supervised clustering algorithm in Diting++ can disambiguate papers better than other unsupervised and semi-supervised clustering methods respectively.

A. DATASETS AND EXPERIMENTAL SETUP

We conduct experiments on three datasets: Arnetminer, DBLP, and CiteSeerX, which are depicted in Table 3. In these datasets, each ambiguous name is a distinct dataset where the disambiguation experiment is conducted. For these datasets, we augment the records of each paper with its abstract and authors' organization by fetching them from the IEEE and ACM repositories through web-crawling.

- Arnetminer[3] contains 110 names, 7022 papers, and 13.8 authors per name for disambiguation. In this dataset, each paper record consists of the following information: author list, title, publication year, authors' affiliation, and publication venue.
- DBLP[4] contains 679 names, 6478 papers, and 2.2 authors per name for disambiguation. In this dataset, each paper record consists: author list, title, venue, keyword, download link, and publication year.
- CiteseerX[5] contains 14 names, 8453 papers, and 33.4 authors per name for disambiguation. In this dataset, each paper record consists of author list, title, and venue information of the papers.

[3]https://aminer.org/disambiguation
[4]https://github.com/yaya213/DBLP-Name-Disambiguation-Dataset
[5]http://clgiles.ist.psu.edu/data/

TABLE 3. The three datasets used in this paper.

	Arnetminer	DBLP	CiteseerX
Papers	7022	6478	8453
Names	110	679	14
Authors	1515	1463	468
Papers/Name	63.8	93.9	603.8
Authors/Name	13.8	2.2	33.4
Papers/Author	4.6	4.4	18.1

In our experiments, we set the representation dimension for all the methods as 40. And the learning rate is $\eta = 0.02$, the regularization coefficient is $\lambda = 0.05$. We obtain parameters w_i in (6) and (9) through hyperparameter tuning in the experiments and find out the most appropriate parameter values which gain the highest Macro-F1 result. The parameter settings for other methods are the same as specified in their publications. In the experiments, we find that the co-year relationship has little effect on our result. Thus we do not use the co-year information for all the methods. We adopt the result using 10% Must-Link and Cannot-Link constraints as the final Macro-F1 for semi-supervised method Diting++, and we also conduct experiments on different percentages of the two constraints. All the experiments in this work are conducted at least five repetitions, and we calculate the average values of Macro-F1 for all methods.

B. PERFORMANCE MEASURES

In this section, we describe the performance measures used in this work. As Diting uses an unsupervised method to cluster papers, we use the precision and recall metrics which are used in unsupervised clustering settings. They are defined as follows.

For a dataset we conduct experiments on, our clustering algorithm aims to divide all papers $\mathcal{P}^n$ associated with the ambiguous name n into disjoint parts, and the clustering result is denoted as $C = \{C_1, C_2, ..., C_r\}$. We get the clustering result that each cluster C_i represents a set of papers written by the same author, but the actual paper distribution maybe not like this. We suppose the number of true distinct authors in $\mathcal{P}^n$ is $|A|$ ($|A|$ may not equal to r). We denote n_i as the number of papers in cluster C_i, and n_{ij} is the number of papers in cluster C_i that are written by true author j. And j_i is the true author id that contains the maximum number of papers in C_i among all the true authors, i.e., $n_{ij_i} = \max_{j=1}^{|A|}\{n_{ij}\}$. The *precision* of a cluster C_i is the same as its purity.

$$prec_i = \frac{1}{n_i}\max_{j=1}^{|A|}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}$$

And the *recall* of cluster C_i is defined as:

$$recall_i = \frac{n_{ij_i}}{|T_{j_i}|} = \frac{n_{ij_i}}{m_{j_i}},$$

where T_{j_i} is the set of papers written by true author j_i and $m_{j_i} = |T_{j_i}|$ is the number of papers in the set. And $recall_i$ measures the fraction of papers in partition T_{j_i} shared in common with cluster C_i. The F1 is the harmonic mean of the precision and recall values for each cluster. The measure for cluster C_i is therefore given as:

$$F1_i = \frac{2}{\frac{1}{prec_i} + \frac{1}{recall_i}} = \frac{2 \cdot prec_i \cdot recall_i}{prec_i + recall_i} = \frac{2n_{ij_i}}{n_i + m_{j_i}}.$$

The Macro-F1 for clustering C is the mean of cluster-wise $F1_i$ values:

$$F1 = \frac{1}{r}\sum_{i=1}^{r} F1_i.$$

The Macro-Precision for clustering C is the mean of cluster-wise $prec_i$ values:

$$Precision = \frac{1}{r}\sum_{i=1}^{r} prec_i.$$

The Macro-Recall for clustering C is the mean of cluster-wise $recall_i$ values:

$$Recall = \frac{1}{r}\sum_{i=1}^{r} recall_i.$$

C. COMPETING DISAMBIGUATION METHODS

We compare our methods Diting and Diting++ with several state-of-the-art disambiguation methods. The parameter setting for each method is the same as specified in their publications. The author disambiguation methods used for disambiguation are listed as follows.

- Khabsa *et al.* 2015 [42] design a pairwise profile similarity function and propose an improved-DBSCAN method to cluster papers. This method first calculates the neighborhood density of each record in the training set. Then for new records, the Euclidean distance is used to calculate their neighborhood densities. If the neighborhood is sparse, the new record is assigned to the new class. Otherwise, it is classified as an existing class of the neighborhood containing the new record.
- Qian *et al.* 2015 [43] use hierarchical agglomerative clustering to cluster papers. This method uses the number of occurrences of each attribute to calculate the class conditional probability of each existing class, assuming that all attributes are independent. The result of the calculation is then compared to a defined threshold to determine whether to assign the newly added record to the current class or a new class.
- Zhang *et al.* 2016 [11] use a Bayesian non-exhaustive classification framework for name disambiguation. This method classifies the incoming sequentially observed stream records into existing classes and emerges classes by using one sweep Gibbs sampler. This model is mainly for solving online name disambiguation task, and the result of name disambiguation has not been greatly improved.
- Zhang *et al.* 2017 [3] use the PTE method to learn paper representations and a hierarchical agglomerative clustering to cluster them. Zhang's main intuition is that neighboring nodes in a graph should have more similar vector representations in the embedding space than non-neighboring nodes. According to three different relation networks, this method maps each node to a low-dimensional vector and then clusters them.

The upper part of Table 4 shows the results of comparing Diting and Diting++ with other disambiguation methods. We find that our methods significantly outperform all the other methods. For some ambiguous names in Arnetminer dataset, we list the Marco-F1 results of different methods in Table 5. We find that our Diting method gets the best result most of the time, and the PTE method gets the best result some times.

D. COMPARING NETWORK EMBEDDING METHODS

We evaluate the performance of different network embedding methods (listed below). For comparison, we use the same network construction method and clustering algorithm. As some methods, like DeepWalk, LINE, and Node2Vec, are designed for a single network rather than multiple networks, we use these methods to learn a vector from a network sequentially for each vertex, and then all the obtained vectors of the vertex are concatenated together as the final representation.

- DeepWalk [17] is a random-walk based method for homogeneous networks. DeepWalk uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences. For each vertex, truncated random walks starting from the vertex are used to obtain the contextual information.
- LINE [19] is a method which models the first-order

TABLE 4. The Macro-Precision(P), Macro-Recall(R), and Macro-F1(F) of different methods for the three datasets.

	Arnetminer			**DBLP**			**CiteSeerX**		
Method	P	R	F	P	R	F	P	R	F
Khabsa *et al.* 2015 [42]	0.633	0.571	0.584	0.674	0.626	0.645	0.421	0.447	0.424
Qian *et al.* 2015 [43]	0.604	0.512	0.547	0.686	0.656	0.681	0.553	0.521	0.547
Zhang *et al.* 2016 [11]	0.523	0.745	0.613	0.741	0.716	0.723	0.547	0.522	0.536
Zhang *et al.* 2017 [3]	0.576	0.693	0.635	0.758	0.706	0.742	0.597	0.575	0.596
DeepWalk 2014 [17]	0.621	0.558	0.582	0.724	0.756	0.734	0.477	0.513	0.482
LINE 2015 [19]	0.654	0.573	0.609	0.723	0.735	0.722	0.536	0.591	0.553
Node2Vec 2016 [22]	0.625	0.541	0.589	0.675	0.705	0.685	0.524	0.465	0.498
PTE 2015 [32]	0.697	0.568	0.632	0.741	0.794	0.762	0.548	0.611	0.578
CANE 2017 [44]	0.588	0.674	0.624	0.682	0.764	0.712	0.499	0.543	0.511
Hin2Vec 2017 [45]	0.655	0.561	0.616	0.714	0.755	0.743	0.589	0.517	0.562
Diting	**0.786**	**0.718**	**0.745**	**0.822**	**0.854**	**0.832**	**0.664**	**0.601**	**0.635**
Diting++	**0.853**	**0.738**	**0.814**	**0.846**	**0.896**	**0.871**	**0.744**	**0.684**	**0.712**

TABLE 5. The Marco-F1 of different methods for some ambiguous names in Arnetminer dataset.

Name	Khabsa	Qian	Zhang16	Zhang17	DeepWalk	LINE	Node2Vec	PTE	CANE	Hin2Vec	Diting
Alok Gupta	0.564	0.571	0.672	0.652	0.618	0.625	0.636	0.745	0.681	0.566	**0.985**
Bin Li	0.615	0.591	0.682	0.676	0.545	0.558	0.579	0.486	0.608	0.582	**0.769**
Bing Liu	0.616	0.644	0.715	0.769	0.742	0.744	0.803	0.701	0.649	0.655	**0.982**
Bo Liu	0.549	0.484	0.664	0.668	0.538	0.641	0.546	0.625	0.626	0.626	**0.814**
Daniel Massey	0.498	0.525	0.556	0.546	0.581	0.517	0.559	0.497	0.743	0.640	**0.846**
David Brown	0.577	0.470	0.603	0.586	0.451	0.545	0.487	0.523	0.581	0.543	**0.854**
David Jensen	0.589	0.640	0.693	0.802	0.782	0.807	0.926	**0.932**	0.559	0.687	0.700
David Nelson	0.501	0.599	0.580	0.569	0.537	0.575	0.600	0.500	0.535	0.649	**0.785**
F. Wang	0.467	0.711	0.778	0.761	0.596	0.636	0.612	0.652	0.587	0.571	**0.912**
Feng Liu	0.563	0.563	0.485	0.551	0.558	0.542	0.477	0.590	0.525	0.529	**0.866**
Hao Wang	0.623	0.573	0.640	0.634	0.522	0.560	0.545	0.650	0.608	0.509	**0.870**
Hui Fang	0.806	0.643	0.685	0.716	0.629	0.688	0.667	0.755	0.512	0.541	**0.862**
J. Guo	0.686	0.497	0.466	0.613	0.465	0.618	0.541	0.645	0.658	0.634	**0.918**
J. Yin	0.454	0.492	0.583	0.584	0.561	0.624	0.488	0.598	0.704	0.556	**0.804**
Jeffrey Parsons	0.771	0.785	0.722	0.768	0.655	0.723	0.744	0.824	0.601	0.533	**0.903**
Ji Zhang	0.492	0.491	0.486	0.513	0.496	0.646	0.492	0.521	0.735	0.638	**0.855**
Jie Yu	0.631	0.698	0.717	0.558	0.713	0.724	0.799	**0.831**	0.558	0.574	0.825
Jim Gray	0.644	0.675	0.789	0.754	0.681	0.832	0.863	0.942	0.613	0.711	**0.966**
Avg Macro-F1	0.591	0.592	0.640	0.651	0.593	0.645	0.631	0.668	0.616	0.597	**0.862**

and second-order proximities. In the first-order, it learns simulations over immediate neighbors of nodes. In the second-order, it learns 2-hop information from the source nodes.

- Node2Vec [22] is a broader abstraction of DeepWalk. It adds a biased random walk strategy, which can be either DFS or BFS. These two modes of travel focus on the structural information and the importance of neighborhood nodes respectively. Node2vec has two hyperparameters p and q, which can be adjusted for different networks.
- PTE [32] can obtain the representations for heterogeneous networks. PTE uses networks of Word-Word, Word-Document, and Word-Label to build a heterogeneous network. The loss function of the three networks is superimposed, which is very similar to LINE.
- CANE [44] adds a mutual attention mechanism to fuse the structural information and text information of the node so that the context information of the node can be considered, and it has different representations when interacting with different nodes.
- Hin2Vec [45] aims to learn vector representations of heterogeneous networks. In HIN2Vec, for multiple predictive tasks, each task corresponds to a meta-path.

This multi-task learning method can embed different relationships and network structures into node vectors.

As shown in Table 4, Diting and Diting++ outperform all the other disambiguation methods and network embedding methods. For all the three datasets, our unsupervised method Diting is 3.9% to 21.1% better than all the others, and the semi-supervised method Diting++ is 11.6% to 28.8% better. PTE and Hin2Vec obtain the second- and third-best performance due to their ability to encode multiple networks directly while the other methods (DeepWalk, LINE, Node2Vec, and CANE) cannot. Our methods perform better than PTE and Hin2Vec because we focus on the gap between positive and negative edges, which is suitable for disambiguation task, and our methods capture global graph properties via learning on coarsened networks.

E. COMPARING UNSUPERVISED CLUSTERING METHODS

We evaluate the performance of different unsupervised clustering algorithms together with our unsupervised method Diting and semi-supervised method Diting++. For each algorithm, we use the same parameters of them across the whole experiments. The clustering algorithms used for comparison, which do not require the input of the number of clusters (the number of authors K for each ambiguous name), are HDBSCAN, AP, MeanShift, and Xmeans.

- HDBSCAN [8]: an algorithm which performs DBSCAN over epsilon values and integrates the results to find a cluster that gives the best stability over epsilon.
- AP [9]: an algorithm that uses density clustering. The basic idea is to treat all data points as potential cluster centers. The data points are connected to form a network (similarity matrix), and then the cluster center of each sample is calculated by passing messages through each side of the network.
- MeanShift [46]: an algorithm which uses kernel density estimation. It works by updating centroid candidates to the mean of the points within an area.
- Xmeans [47]: an algorithm based on Kmeans. It iteratively determines K by using the Bayesian Information Criterion.

Fig. 7 summarizes the comparison results. The semi-supervised method Diting++ performs significantly better than all the other unsupervised methods. As for the unsupervised methods, for all the three datasets, our Diting method consistently outperforms MeanShift and Xmeans, and can always obtain a better result between HDBSCAN and AP by using the method proposed in Section IV-C. The Marco-F1 of Diting is about 1.2% to 4.5% better than the second-best unsupervised method (HDBSCAN).

F. COMPARING SEMI-SUPERVISED CLUSTERING METHODS

We also evaluate the performance of different semi-supervised clustering algorithms for disambiguation together with our unsupervised method Diting and semi-supervised method Diting++. For each algorithm, we use the same parameters in the experiments. The semi-supervised clustering algorithms used for comparison, which require the input of the number of clusters, are PCK-Means, COP-KMeans, MK-Means, RCA-KMeans, and MPCK-Means.

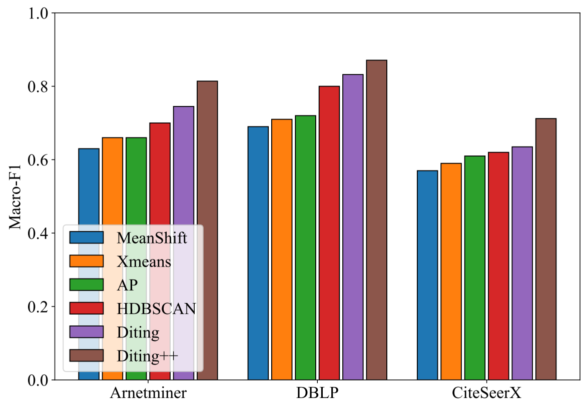


FIGURE 7. The performance of different unsupervised clustering algorithms.

- PCK-Means [48] utilizes constraints for seeding the initial clusters and directs the cluster assignments to respect the constraints without doing any metric learning.
- COP-KMeans [41] proposes the Must-Link and Cannot-Link constraints in the Kmeans algorithm. It returns a partition of the instances in the dataset that satisfies all specified constraints.
- MK-Means [49] has a metric learning component and does not utilize constraints for initialization. A single metric parameterized by a diagonal matrix is used for all clusters.
- RCA-KMeans [50] is accomplished by using side-information in the form of equivalence relations. Equivalence relations provide them with small groups of data points that are known to be similar.
- MPCK-Means [51] involves both seeding and metric learning in the unified framework. A single metric parameterized by a diagonal matrix is used for all clusters.

We only display the results of the Arnetminer and CiteseerX datasets, as the characteristic of DBLP dataset is similar. Fig. 8 and Fig. 9 summarize the comparison results. If there is no constraint or the percentage of constraints is very small, the unsupervised method Diting proposed in our paper is superior to all the other semi-supervised clustering algorithms, including our semi-supervised method Diting++. However, as the percentage increases, the advantages of semi-supervised algorithms reveal. For all the semi-supervised algorithms, the Macro-F1 increases rapidly with the increase of the constraints and finally reaches 100%. And most of the times, the Macro-F1 result of Diting++ is higher than all the other methods. In conclusion, the proposed semi-supervised algorithm in Diting++ outperforms existing semi-supervised clustering algorithms for author disambiguation

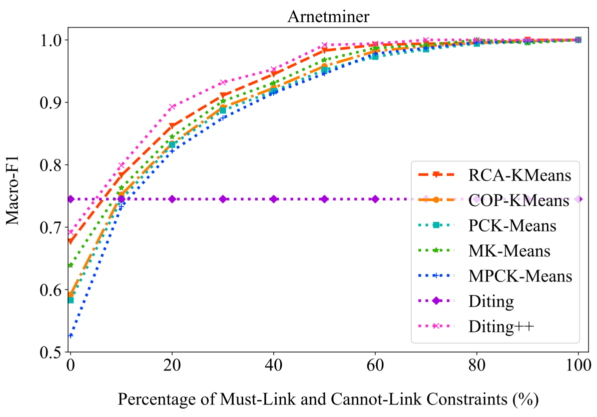


FIGURE 8. The Macro-F1 of semi-supervised algorithms in Arnetminer.

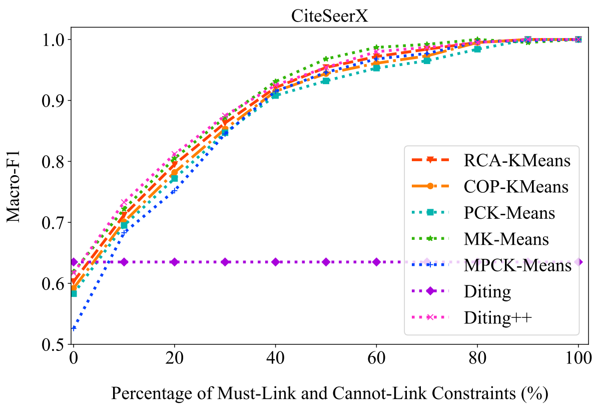


FIGURE 9. The Macro-F1 of semi-supervised algorithms in CiteSeerX.

problem.

G. ALTERNATIVES AND SENSITIVITY RESULTS

In this section, we evaluate some alternatives that are discussed in the previous sections in our work. And we also conduct experiments to analyze the sensitivity results. The results are all shown in Fig. 10, which displays the experiment results on Arnetminer dataset.

Firstly, in our experiments, when learning representations, for a network with $|E|$ edges, $T \times |E|$ triples are sampled. We evaluate the impact of T by varying it from 1 to 4. As we can see from the figure that T does not impact the performance significantly.

Secondly, for each ambiguous name, we create networks based on papers whose author list contains the name. An alternative way is to create a big network based on all the papers in a dataset. The results of the comparison are shown in Fig. 10 with the legend *All network* representing the Macro-F1 result by using the big network and the legend *Diting* and *Diting++* representing the results by using our networks. The Marco-F1 for Diting and Diting++ are about 12% and 18.9% better than the *All network*. Compared to the networks constructed in our work, there are more edges in

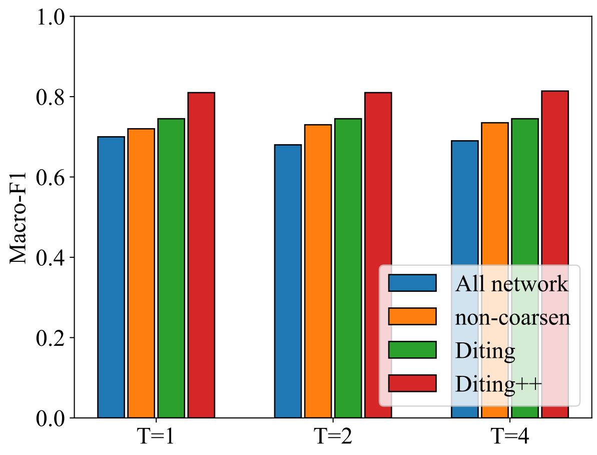


FIGURE 10. The evaluation of some alternatives.

the big network consisting of all papers, which suggests more information. However, most of the edges in the big network are not relevant to the ambiguous name. In other words, they are better treated as negative edges than positive edges.

Thirdly, in section IV-B, we use graph coarsening to mainly learn the global network structure and get the pre-trained representations for all vertices in our networks. As shown in Fig. 10, compared with the non-coarsen networks, our Diting method gains about 1% better Marco-F1 result and Diting++ gains about 7.9% better result.

VII. CONCLUSION

In this work, we propose an unsupervised method Diting to solve the author disambiguation problem. We first use the network representation learning to get paper vectors through jointly learning the multiple types of networks constructed by different relationships of papers, and then design the clustering algorithm to participate them into distinct parts to distinguish the ambiguous authors. Further, we also propose a semi-supervised method Diting++ to improve the result by taking account of the authorship information of papers. Our method can learn heterogeneous relationships (e.g., co-author, title similarity) between papers and can be easily adapted to many other scenarios. Through extensive experiments, we show that our methods can significantly outperform other state-of-the-art methods.

VIII. ACKNOWLEDGMENTS

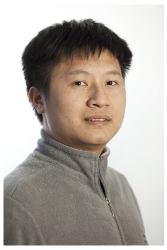
This work was partially supported by the National Key R&D Program of China (2018YFB2101100), the National Natural Science Foundation of China (No.61932001, 61602500, 61972409, 61502500, 61702529, 61702530).

REFERENCES

[1] A. Chen, L. Shu, and X. Ming, "Efficient spectral neighborhood blocking for entity resolution," 2011.
[2] J. Tang, A. C. M. Fong, B. Wang, and J. Zhang, "A unified probabilistic framework for name disambiguation in digital library," TKDE, vol. 24, no. 6, 2012.

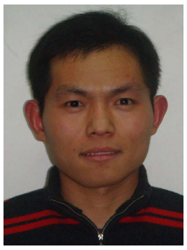
[3] B. Zhang and M. A. Hasan, “Name disambiguation in anonymized graphs using network embedding,” in CIKM, 2017.
[4] X. Lin, F. Zhu, B. Peng, and W. Li, “A novel approach for author name disambiguation using ranking confidence,” in DSFAA, 2017.
[5] W. S. Chin, Y. C. Juan, and W. Cheng, “Effective string processing and matching for author disambiguation,” in KDD Cup, 2013.
[6] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis, “Two supervised learning approaches for name disambiguation in author citations,” in Joint Conference on Digital Libraries (JCDL), 2004.
[7] X. Wang, J. Tang, H. Cheng, and P. S. Yu, “Adana: Active name disambiguation,” in ICDM, 2011.
[8] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-based clustering based on hierarchical density estimates,” in PAKDD, 2013.
[9] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” Science, vol. 315, no. 5814, pp. 972–976, 2007.
[10] X. Yin, J. Han, and P. S. Yu, “Object distinction: Distinguishing objects with identical names,” in Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pp. 1242–1246, 2007.
[11] B. Zhang, M. Dundar, and M. A. Hasan, “Bayesian non-exhaustive classification a case study:online name disambiguation using temporal record streams,” in CIKM, 2016.
[12] J. Xu, S. Shen, D. Li, and Y. Fu, “A network-embedding based method for author disambiguation,” pp. 1735–1738, 2018.
[13] D. Zhang, J. Tang, J. Li, and K. Wang, “A constraint-based probabilistic framework for name disambiguation,” in Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007, pp. 1019–1022, 2007.
[14] M. Imran, S. Gillani, and M. Marchese, “A real-time heuristic-based unsupervised method for name disambiguation in digital libraries,” in JCDL, 2013.
[15] B. Yoshua, C. Aaron, and V. Pascal, “Representation learning: a review and new perspectives,” IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.
[16] D. Yan, T. W. Randolph, J. Zou, and P. Gong, “Incorporating deep features in the analysis of tissue microarray images,” 2018.
[17] B. Perozzi, R. Alrfou, and S. Skiena, “Deepwalk: online learning of social representations,” KDD, 2014.
[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” Computer Science, 2013.
[19] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in KDD, 2015.
[20] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” in CIKM, 2015.
[21] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, “Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec,” in WSDM, 2018.
[22] A. Grover and J. Leskovec, “node2vec: Scalable feature learning,” in KDD, 2016.
[23] C. Tu, W. Zhang, Z. Liu, and M. Sun, “Max-margin deepwalk: discriminative learning of network representation,” in IJCAI, 2016.
[24] R. Feng, Y. Yang, W. Hu, F. Wu, and Y. Zhuang, “Representation learning for scale-free networks,” in AAAI, 2018.
[25] C. Yang, M. Sun, Z. Liu, and C. Tu, “Fast network embedding enhancement via high order proximity approximation,” in IJCAI, 2017.
[26] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, “Harp: Hierarchical representation learning for networks.,” in AAAI, 2018.
[27] H. Wang, jia Wang, jialin Wang, M. ZHAO, W. Zhang, F. Zhang, X. Xing, and M. Guo, “Graphgan: Graph representation learning with generative adversarial nets,” in AAAI, 2018.
[28] Q. Dai, Q. Li, J. Tang, and D. Wang, “Adversarial network embedding,” in AAAI, 2018.
[29] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” CoRR, vol. abs/1609.02907, 2016.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,” International Conference on Learning Representations, 2018. accepted as poster.
[31] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in NIPS, 2017.
[32] J. Tang, M. Qu, and Q. Mei, “Pte: Predictive text embedding through large-scale heterogeneous text networks,” in WWW, 2015.
[33] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” VLDB, vol. 4, no. 11, pp. 992–1003, 2011.
[34] A. Swami, A. Swami, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in KDD, 2017.
[35] K. Tu, P. Cui, X. Wang, fei Wang, and W. Zhu, “Structural deep embedding for hyper-networks,” in AAAI, 2018.
[36] E. L. Bird, Steven and E. Klein, Natural Language Processing with Python. O’Reilly Media Inc., 2009.
[37] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, “Learning distributed representations of texts and entities from knowledge base,” TACL, 2017.
[38] J. Macqueen, “Some methods for classification and analysis of multivariate observations,” in Proceedings of the Fifth Symposium on Math, Statistics, and Probability, pp. 281–297, 1965.
[39] M. Ester, H. P. Kriegel, and X. Xu, “a density-based algorithm for discovering clusters in large spatial databases with noise,” in KDD, pp. 226–231, 1996.
[40] M. Halkidi, M. Vazirgiannis, and Y. Batistakis, “Quality scheme assessment in the clustering process,” in PKDD, 2000.
[41] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, “Constrained k-means clustering with background knowledge,” in Eighteenth International Conference on Machine Learning, 2001.
[42] M. Khabsa, P. Treeratpituk, and C. L. Giles, “Online person name disambiguation with constraints,” in JCDL, 2015.
[43] Y. Qian, J. Zheng, and J. Liu, “Author name disambiguation for growing digital libraries,” Information Retrieval Journal, vol. 18, no. 5, 2015.
[44] C. Tu, H. Liu, Z. Liu, and M. Sun, “Cane: Context-aware network embedding for relation modeling,” in ACL, pp. 1722–1731, 2017.
[45] T. Fu, W. Lee, and Z. Lei, “Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning,” in CIKM, 2017.
[46] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” PAMI, vol. 24, no. 5, pp. 603–619, 2002.
[47] P. Dan and A. W. Moore, “X-means: Extending k-means with efficient estimation of the number of clusters,” in ICML, 2000.
[48] J. Kleinberg and v. Tardos, “Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields,” in Symposium on Foundations of Computer Science, 1999.
[49] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, “Distance metric learning, with application to clustering with side-information,” in International Conference on Neural Information Processing Systems, 2002.
[50] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, “Learning distance functions using equivalence relations,” Icml, pp. 11–18, 2003.
[51] M. Bilenko, S. Basu, and R. J. Mooney, “Integrating constraints and metric learning in semi-supervised clustering,” in International Conference, 2004.

LIWEN PENG received her B.E. degree in computer science from the National University of Defense Technology, Changsha in 2017. Her research interests include social network and network representation learning.



SIQI SHEN received his B.S. and M.S. degree from National University of Defense Technology, China, in 2007 and 2009, respectively, and his Ph.D. degree from the Delft University of Technology, the Netherlands, in 2015. He is currently an assistant professor at National Lab for Parallel and Distributed Processing, National University of Defense Technology, China. His research interests include computer network and data mining.

JUN XU received his B.E. degree in computer science from the University of Chongqing, Chongqing in 2016 and M.S. degree from the National University of Defense Technology in 2019. His research interests include social network and network representation learning. He is working at Ant Financial since 2019.



YONGQUAN FU received the B.E. degree in computer science and technology from Shandong University, Jinan, China in 2005, and the M.S. and Ph.D. degree in computer science and technology from National University of Defense Technology, Changsha, China in 2007 and 2012, respectively. Since 2013, he has been with the Science and Technology Laboratory of Parallel and Distributed Processing, College of Computer, National University of Defense Technology, where he is currently an associate professor. His research interests include network measurement, social network, and distributed systems.

DONGSHENG LI received the B.S. degree (with honors) and Ph.D. degree (with honors) in computer science from College of Computer Science, National University of Defense Technology, Changsha, China, in 1999 and 2005, respectively. He was awarded the prize of National Excellent Doctoral Dissertation of PR China by the Ministry of Education of China in 2008. He is now a full professor at National Lab for Parallel and Distributed Processing, National University of Defense Technology, China. His research interests include distributed computing, cloud computing, computer network, and large-scale data management.

ADELE LU JIA received her B.S. degree from Harbin Institute of Technology, China, in 2007, her M.Phil. degree from The Chinese University of Hong Kong in 2009, and her Ph.D. degree from the Delft University of Technology, the Netherlands, in 2013. She is currently an associate professor in the Computer Science Department at China Agricultural University. Her research interests include complex network analysis and data mining.