Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6654349, 10 pages
https://doi.org/10.1155/2021/6654349

Research Article

Fusing Node Embeddings and Incomplete Attributes by Complement-Based Concatenation

Zheng Wang,[1,2] Yuexin Wu,[3] Yang Bao,[3] Jing Yu,[3] and Xiaohui Wang[4]

[1]*Department of Computer Science and Technology, University of Science and Technology Beijing, Beijing, China*
[2]*Department of Computer and Information Science, University of Macau, Macao, China*
[3]*School of Software, Tsinghua University, Beijing, China*
[4]*School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, China*

Correspondence should be addressed to Xiaohui Wang; wangxh14@ustb.edu.cn

Received 27 December 2020; Revised 22 January 2021; Accepted 8 February 2021; Published 25 February 2021

Academic Editor: Wei Wang

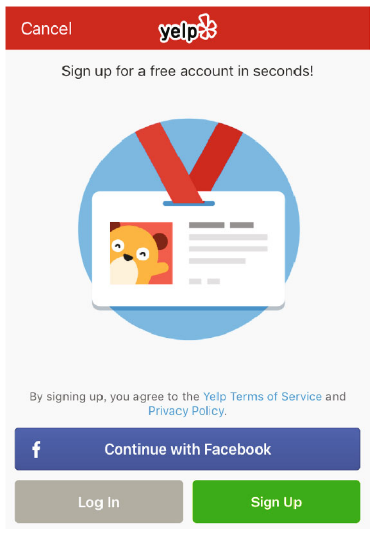
Network embedding that learns representations of network nodes plays a critical role in network analysis, since it enables many downstream learning tasks. Although various network embedding methods have been proposed, they are mainly designed for a single network scenario. This paper considers a "multiple network" scenario by studying the problem of fusing the node embeddings and incomplete attributes from two different networks. To address this problem, we propose to complement the incomplete attributes, so as to conduct data fusion via concatenation. Specifically, we first propose a simple inductive method, in which attributes are defined as a parametric function of the given node embedding vectors. We then propose its transductive variant by adaptively learning an adjacency graph to approximate the original network structure. Additionally, we also provide a light version of this transductive variant. Experimental results on four datasets demonstrate the superiority of our methods.

1. Introduction

Social network sites (SNSs, also commonly referred as social networking services) are online platforms which provide users with various features to facilitate digital social interaction and information sharing [1, 2]. Over three billion users are currently active on various SNSs (like Facebook, Twitter, and QQ), spending on average two hours daily. These wide and active SNSs naturally form an important part of the digital economy, making social network analysis [3, 4] become a hot research topic over the years.

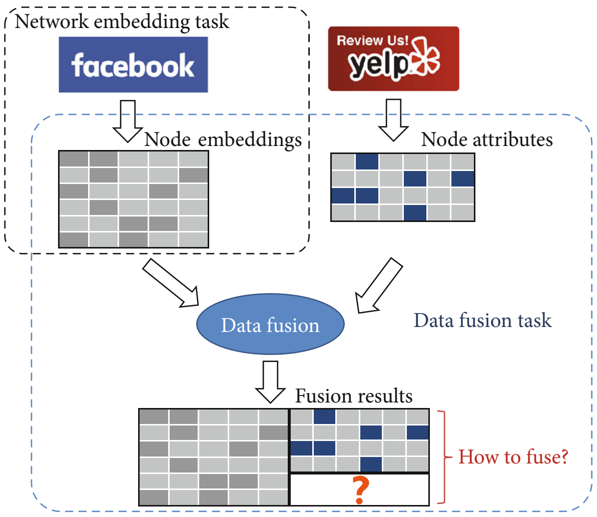
Recently, network embedding [5], as a fundamental problem in network analysis, has aroused considerable research interest. Network embedding learns low-dimensional vector representations for network nodes. The learned vectorized representations, which preserve certain structural and content information of networks, can be easily combined with off-the-shelf learning algorithms for many social network analysis tasks such as node classification [6], link prediction [7], and diffusion prediction [8].

1.1. Problem. Although various network embedding methods have been proposed, they mainly focus on a single network scenario. In the era of big data, the related information from different networks should be fused together to facilitate applications. In this paper, we consider a "multiple network"


Cancel
yelp
Sign up for a free account in seconds!
By signing up, you agree to the Yelp Terms of Service and Privacy Policy.
Continue with Facebook
Log In
Sign Up

Facebook login on Yelp

(a)



Fusing node embeddings and incomplete attributes

(b)

FIGURE 1: Illustration of the studied problem.

scenario by studying the problem of fusing the node embeddings and incompleted attributes provided by two different networks.

As illustrated in Figure 1, this problem has practical importance. Imagine that you use Yelp (see Figure 1(a)), a popular review app, and try to get in your account there. Yelp allows you to sign in using your Facebook account. In addition, as the node (user) embeddings not only preserve certain characteristics of networks but also protect users' privacy [9], Facebook may provide these embeddings to Yelp to facilitate its applications, e.g., cold-start recommendation. More importantly, as some Yelp users begin to write reviews, a very practical problem would arise: is it possible to fuse the original node embeddings provided by Facebook and the reviews provided by Yelp to get new user embeddings (illustrated in Figure 1(b))?

1.2. Challenge and Solution. Certainly, one fundamental challenge is the incompleteness of attributes, i.e., only a small part of nodes are further provided with attributes. This challenge is very common. As reported [10], the distribution of user activity tends to be long-tailed, suggesting most social media contents (like the reviews on Yelp) are actually written by a few active users. To address this, we propose to complement the incomplete attributes by defining attributes as a parametric function of the given node embedding vectors. This complement enables us to conduct data fusion via concatenation (illustrated in the bottom right corner of Figure 1(b)).

To obtain high-quality fusion results, we further propose a transductive method by adaptively learning an adjacency graph to approximate the original network structure. In particular, the adjacency graph is learned by jointly considering the given node embeddings and attribute knowledge. Additionally, we also provide a light version of the proposed transductive method. Specifically, for each node, this light version reduces its neighbor candidate set for efficient adjacency graph learning. We then conduct extensive experiments to verify the effectiveness of our methods.

In summary, our main contributions are as follows:

(i) We study the problem of fusing node embeddings and incomplete attributes from two different networks. To our best knowledge, little work has addressed this problem

(ii) We propose a very simple and effective inductive method based on the idea of attribute complement

(iii) We further propose a transductive method POINTS and its light version POINTS*, both of which could obtain superior performance

(iv) The remainder of the paper is organized as follows. We review related work in Sect. 2 and formalize the problem in Sect. 3. We present our method in Sect. 4 and then provide some discussion in Sect. 6. We conduct experiments in Sect. 7. Finally, we end with a conclusion in Sect. 8

2. Related Work

2.1. Network Embedding. Over the past few years, there has been a lot of interest in learning useful node embeddings (i.e., features) from large-scale networks automatically [5]. A representative work is DeepWalk [6] which performs random walk on a network to generate node sequences and then perform the skip-gram algorithm [11] on those sequences to achieve the embedding. Another well-known work is LINE which preserves both first-order proximity (i.e., the similarity between linked nodes) and second-order proximity (i.e., the similarity between the nodes with shared neighbors) of a network. In addition, researchers have also proposed some deep learning-based embedding models, such as SDNE [12] and

GraphGAN [13]. Recently, lots of studies consider the network embedding with side information, such as node attributes. For example, by proving DeepWalk is equivalent to matrix factorization, the work in [14] presents text-associated DeepWalk (TADW). GraphSAGE [15] employs graph convolutional networks [16] to aggregate features among node neighborhood for network embedding. RSDNE [17] and RECT [18] further consider the problem of zero-shot graph embedding, i.e., the completely imbalanced label setting.

2.2. Data Fusion. Data fusion is the study of efficient methods for automatically transforming information from different sources and different points in time into a representation that provides effective support for human or intelligent systems. Data fusion has proved useful in many disciplines, as discussed in [19, 20]. For example, in bioinformatics, jointly analyzing multiple datasets describing different organisms improves the understanding of biological processes [21]. In information retrieval, fusing the retrieval results from multiple search engines would significantly improve the retrieval performance [22]. In biometric recognition systems, feature fusion could greatly improve the recognition performance [23]. We refer to [24, 25] for a comprehensive survey.

However, little previous work considers the fusion of incomplete data or network embedding data. Our work fills this gap.

3. Problem Statement

The studied problem is defined as follows. We are given the node embeddings of a network $U \in \mathbb{R}^{n\times d}$, where n is the node number and the i-th row of U (denoted as u_i) is a d-dimensional embedding vector of node i. On the other hand, another network further provides the attributes of l ($l < n$) nodes: $\mathscr{L}_A = \{(u_1, a_1), \cdots, (u_l, a_l)\}$, where $a_i \in \mathbb{R}^{1\times m}$ is the attribute vector, and m is the attribute feature number. Our goal is to fuse the given node embeddings and those incomplete attributes, so as to get the updated embeddings for all nodes. Note that different from existing network embedding methods, the original network structure is unknown in our problem.

4. The Proposed Method

4.1. Fusion via Attribute Complement. Since only a small part of nodes are further provided with attributes, we cannot directly fuse node embeddings and attributes. To address this problem, we adopt a very simple complement strategy: predicting the nonexist attributes. In particular, for each node i which is further provided with attributes, we assume that its node embedding u_i should have the ability to generate its attribute vector a_i. The optimal generation function f can be obtained by solving the following minimization problem:

$$\min_f \sum_{i\in\mathscr{L}_A} \ell(f(u_i), a_i), \tag{1}$$

where ℓ is a loss function that measures the reconstruction error, such as squared loss or hinge loss.

By solving the problem in Eq. (1), we can obtain the generation function f. Then, for a node i with no attributes, we can predict its attributes by applying $f(u_i)$. This complement enables us to conduct data fusion via concatenation. More details and discussion about the concatenation strategy can be found in Sect. 6.2.

4.2. Transductive Attribute Prediction. The method formulated in Eq. (1) is inductive. In this section, we present a transductive method. Generally, transductive methods, which leverage the test data for model training, perform better than inductive methods [26]. For network embedding, classical transductive methods exploit all network nodes by preserving the inherent network structure in the embedding space, i.e., connected nodes tend to have similar embeddings [27, 28]. Although the original network structure is unknown, one can simply build a sparse adjacency graph (We use the term "graph" to describe the recovered network structure, as to avoid ambiguity with the original network structure.) S to approximate it, i.e., $S_{ij} = 1$ when node j is the k-nearest neighbors of node i in the given node embedding space, otherwise $S_{ij} = 0$. This approximation can capture the intuition of transductive learning by the following cost term:

$$\begin{gathered}\min_Y \sum_{i,j} S_{ij} Dist\left(y_i, y_j\right), \\ \text{s.t.} \forall i \in \mathscr{L}_A, y_i = a_i,\end{gathered} \tag{2}$$

where $Dist(\cdot,\cdot)$ is a distance function, and $y_i \in \mathbb{R}^{1\times m}$ (the i-th row of matrix $Y \in \mathbb{R}^{n\times m}$) is the predicted attribute vector of node i. The imposed constraint ensures the predicted attributes to be consistent with the known attributes.

The adjacency graph plays a crucial role in this kind of graph-based transductive learning methods [29, 30]. However, the matrix S in Eq. (2) might not be the optimal adjacency graph. On the one hand, the original network information is only approximately described by the given node embeddings (i.e., $u_{i=1,\cdots,n}$) which S is built from. On the other hand, the construction of S ignores the attribute information, i.e., similar (dissimilar) attributes indicate similarity (dissimilarity) between different nodes. In this paper, we solve this problem in an adaptive way. Specifically, we propose to learn S by jointly considering the given node embeddings and attribute knowledge. This yields the following cost term:

$$\begin{aligned}\min_{Y,S}\ & \frac{\alpha}{2}\sum_{i,j} S_{ij} Dist\left(y_i, y_j\right) + \frac{\beta}{2}\sum_{i,j} S_{ij} Dist\left(u_i, u_j\right), \\ \text{s.t.}\ & \quad \forall i \in \mathscr{L}_A, y_i = a_i \\ & \quad \forall i,\ \left\|s_i^{\prime}\right\|_0 = s_i^{\prime}\mathbf{1} = k,\ S_{ii} = 0\end{aligned}, \tag{3}$$

where $s_i \in \mathbb{R}^{n\times 1}$ is a vector with the j-th element as S_{ij} (i.e., $s_i^{\prime}$ is the row vector of matrix S), $\mathbf{1}$ denotes a column vector with all entries equal to one, and α and β are two adjustable parameters. Intuitively, the first and second term of Eq. (3) measure how well the adjacency graph fits the attributes and the given node embeddings, respectively.

The unified model: POINTS with learning the attribute generation function (Eq. (1)) and adjacency graph (Eq. (3)), the proposed method is to solve the following optimization problem:

$$\begin{aligned}\min_{f,Y,S} \mathcal{J} = &\sum_{i\in\mathcal{L}_A} \ell(f(u_i), y_i) + \frac{\alpha}{2}\sum_{i,j} S_{ij} Dist\left(y_i, y_j\right) \\ &+ \frac{\beta}{2}\sum_{i,j} S_{ij} Dist\left(u_i, u_j\right), \\ \text{s.t.} \quad &\forall i \in \mathcal{L}_A, y_i = a_i \\ &\forall i,\ \left\|s_i'\right\|_0 = s_i'\mathbf{1} = k,\ S_{ii} = 0\end{aligned} \quad (4)$$

Since the key idea of this method is to learn the adjacency graph adaptively, we term our method as adaPtively netwOrk embeddIng aNd aTtribute fuSion (POINTS).

A Light Version of POINTS: for each node i, to learn its optimal neighbors, POINTS needs to consider all nodes. This is very inefficient, as the network may be extremely large (some theoretical analysis can be found in Sect. 6.3). Therefore, we give a light version of POINTS (denoted as POINTS*). In particular, we propose to build a candidate neighbor set (denoted as $\mathcal{N}_{k^*}(i)$) for each node, where k^* ($k < k^* \ll n$) is the candidate neighbor number. Based on this idea, the light version POINTS* is to solve the following optimization problem:

$$\begin{aligned}\min_{f,Y,S} \mathcal{J}^* = &\sum_{i\in\mathcal{L}_A} \ell(f(u_i), y_i) + \frac{\alpha}{2}\sum_{i,j} S_{ij} Dist\left(y_i, y_j\right) \\ &+ \frac{\beta}{2}\sum_{i,j} S_{ij} Dist\left(u_i, u_j\right), \\ \text{s.t.} \quad &\forall i \in \mathcal{L}_A, y_i = a_i \\ &\forall i,\ \left\|s_i'\right\|_0 = s_i'\mathbf{1} = k,\ S_{ii} = 0\,. \\ &\forall i,\ \text{if } j \notin \mathcal{N}_{k^*}(i),\ S_{ij} = 0\end{aligned} \quad (5)$$

5. Optimization

The objective functions of POINTS (i.e., Eq. (4)) and POINTS* (i.e., Eq. (5)) both contain 0/1 constraints, which might be difficult to solve by the conventional optimization tools. In this section, we develop efficient solutions for these two problems.

5.1. Optimization for POINTS. Before deriving the optimization algorithm, we need to specify the choice of functions in Eq. (4). For simplicity, we choose a linear model for f, i.e., $f(u_i) = u_i W$, where $W \in \mathbb{R}^{d\times m}$ is the model parameter matrix. In addition, we adopt squared loss for ℓ and squared Euclidean distance for $Dist(\cdot,\cdot)$. As such, we can update the variables in Eq. (4) iteratively, as follows.

Update rule of W and Y: by fixing the other variables, the partial derivative of $\mathcal{J}$ w.r.t. W is

$$\frac{\partial \mathcal{J}}{\partial W} = 2\left(U'UW - U'Y\right). \quad (6)$$

Therefore, we can update W as $W \leftarrow W - \eta(\partial\mathcal{J}/\partial U)$, where η is the learning rate.

When the other variables are fixed, we can obtain the partial derivative of $\mathcal{J}$ w.r.t. Y as

$$\frac{\partial \mathcal{J}}{\partial Y} = 2(-UW + Y) + 2\alpha\Delta Y, \quad (7)$$

where $\Delta = D - (S + S')/2$, and D is a diagonal matrix whose i-th diagonal element is $\sum_j (S_{ij} + S_{ji})/2$. Then, we can update Y as $Y \leftarrow Y - \eta(\partial\mathcal{J}/\partial Y)$. After that, for each node i with given attributes a_i, we adjust its predicted attributes as $y_i = a_i$, so as to satisfy the constraint in Eq. (4).

Update rule of S: when the other variables are fixed, the original optimization problem reduces to

$$\begin{aligned}&min_S \alpha\sum_{i,j} S_{ij}\left\|y_i - y_j\right\|_2^2 + \beta\sum_{i,j} S_{ij}\left\|u_i - u_j\right\|_2^2 \\ &\text{s.t.}\forall i,\ \left\|s_i'\right\|_0 = s_i'\mathbf{1} = k,\ S_{ii} = 0.\end{aligned} \quad (8)$$

As problem (8) is independent between different i, we can instead to solve n decoupled subproblems:

$$\begin{aligned}&\min_{s_i,\forall i} \sum_{j=1}^{n} \alpha\left\|y_i - y_j\right\|_F^2 S_{ij} + \beta\left\|u_i - u_j\right\|_F^2 S_{ij} \\ &\text{s.t.}\forall i,\ \left\|s_i'\right\|_0 = s_i'\mathbf{1} = k,\ S_{ii} = 0.\end{aligned} \quad (9)$$

The optimal solution of problem (9) is (proved in Sect. 6.1)

$$S_{ij} = \begin{cases} 1, & \text{if } j \in \mathcal{N}_k^{UA}(i)\,; \\ 0, & \text{otherwise.}\end{cases} \quad (10)$$

where set $\mathcal{N}_k^{UA}(i)$ contains the top-k nearest nodes to i in the network "embedding-attribute" space, where the distance between node i and j is defined as $\alpha\|y_i - y_j\|_2^2 + \beta\|u_i - u_j\|_2^2$..

We can iteratively update these three variables until convergence to obtain the final solution. After that, as discussed in Sect. 6.2, we can get the final fusion results by concatenation. For clarity, we summarize the complete fusion procedure in Alg. 1.

5.2. Optimization for POINTS*. The optimization approach of POINTS* is very similar to that of POINTS in Sect. 5.1. The only difference is that when updating S as other variables are fixed, we only need to sort the nodes in (its neighbor candidate set) $\mathcal{N}_{k^*}(i)$ to get the top-k nearest neighbors in the

```
Input: The given node embeddings $U$, the attribute information, learning rate $\eta$, and parameters $\alpha$ and $\beta$;
Output: The final fusion results;
1: Initialize $W$, $Y$ and $S$;
2: repeat
3:   Update $W$ as $W \leftarrow W - \eta(\partial \mathcal{J}/\partial U)$.
4:   Update $Y$ as $Y \leftarrow Y - \eta(\partial \mathcal{J}/\partial Y)$;
5:   Set $y_i = a_i$ ; $\forall i \in \mathcal{L}_A$;
6:   Update $S$ by solving problem (8);
7: until Convergence or a certain iteration;
8: Obtain the final fusion results via concatenation, as discussed in sect. 6.2
9: return The final fusion results.
```

Algorithm 1. POINTS.

network embedding attribute space, so as to get the optimal solution of S.

6. Algorithm Analysis

6.1. Optimization Algorithm Solving Problem (9)

Theorem 1. *The optimal solution of problem (9) is Eq. (10).*

Proof. By contradiction, suppose node i has gotten its optimal neighbor set $\mathcal{N}_k^{UA}$ which contains a node p not in i's top-k nearest nodes in the "node-attribute" space. For convenience, we use $\Psi(i, j)$ to denote the distance between nodes i and j in this space, i.e., $\Psi(i, j) = \alpha\|y_i - y_j\|_2^2 + \beta\|u_i - u_j\|_2^2$. As such, there must exist a node $q \notin \mathcal{N}_k^{UA}$ which is one of i's top-k nearest nodes in this space. Then, we get $\Psi(i, p) > \Psi(i, q)$. Considering our minimization problem (i.e., Eq. (9)), this inequation leads

$$\sum_{j \in \mathcal{N}_k} \Psi(i, j) > \sum_{j \in \{\mathcal{N}_k + q\} \setminus p} \Psi(i, j). \quad (11)$$

This indicates that $\{\mathcal{N}_k^{UA} + q\} \setminus p$ is a better optimal solution than $\mathcal{N}_k^{UA}$, a contradiction.

Actually, we can generalize the above proof to a more general case.

Theorem 2. *Suppose node i is close to j than node p. If the chosen distance function $Dist(\cdot, \cdot)$ satisfies $Dist(i, j) < Dist(i, p)$, the optimal solution of problem (9) is Eq. (10) (which adopts the distance function $Dist(\cdot, \cdot)$).*

Proof. This conclusion can be proved by replacing the squared Euclid distance function in the proof of Theorem 1 by $Dist(\cdot, \cdot)$.

6.2. Fusion Strategy. In this part, we will discuss how to conduct data fusion, based on the proposed attribute complement methods. The inductive method (described in Sect. 4.1) would learn a generation function f. Then, for each node i, we can predict its attribute vector as $y_i = f(u_i)$. For those two transductive methods (described in Sect. 4.2), we will directly obtain the predicted attribute vectors $y_{i=1,\cdots,n}$. As such, the attributes are completed for fusion. Specifically, we adopt a "trick" concatenation strategy: (1) if node i has no attributes, we obtain its final fusion vector by concatenating u_i and the predicted attribute vector y_i; (2) if node i has attributes, we obtain its final fusion vector by concatenating u_i and a_i. The principle of this trick is that the given attributes are always more stable and accurate than the predicted attributes for node description.

Table 1: The statistics of datasets.

Name	Citeseer	Cora	Wiki	Pubmed
#nodes	3,312	2708	2,405	19,717
#edges	4,732	5429	17,981	44,338
#classes	6	7	17	3
#attributes	3,703	1433	4,973	500

6.3. Time Complexity. The time complexity of Alg. 1 is as below. The complexity for updating W is $O(d^2n + d^2m + d\,mn)$. The complexity for updating Y is $O(nnz(\Delta)m + dmn)$, where $nnz(\cdot)$ is the number of nonzeros of a matrix. The complexity for updating S is $O(n^2 \log n)$, because for each node, we have to calculate its top-k nearest neighbors. As d, $m \ll n$ is linear with n and $nnz(\Delta)$ is linear with n, the overall complexity of POINTS is $O(\tau(n^2 \log n))$, where τ is the number of iterations to converge.

For the light version, i.e., POINTS*, the complexity of updating S becomes $O(nk^* \log k^*)$, and all others remain the same. Hence, since $k^* \ll n$, the overall complexity becomes $O(\tau(d^2n + dmn + nk^* \log k^*))$. As our method usually converges fast ($\tau \leq 20$ in our experiments) and $d, m \ll n$, the complexity of POINTS* is linear to the number of nodes.

7. Experiments

Datasets: we conduct our experiments on four widely used citation network datasets: Citeseer [31], Cora [31], Wiki [32], and Pubmed [32]. In these networks, nodes are documents, and edges denote the citation relationship between them. Node attributes (i.e., features) are the bag-of-word representations of documents. The statistic of these networks is shown in Table 1.

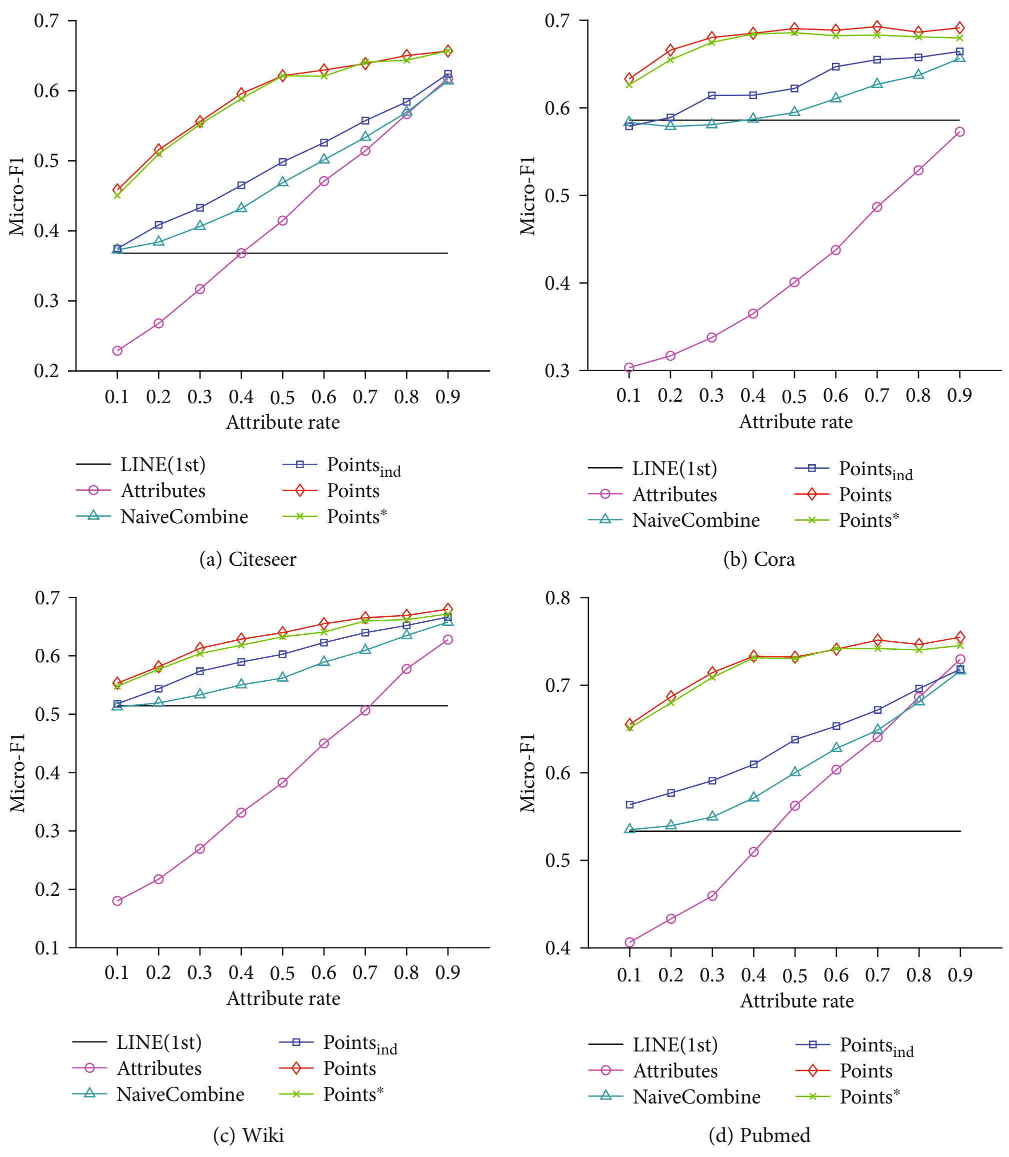


Figure 2: Node classification results (Micro-F1).

Experimental setting: as illustrated in Figure 1, for each dataset, we first obtain the original node embeddings and then provide some nodes with attributes for data fusion, so as to simulate fusing data from two different networks. Specifically, we first obtain the original node embeddings by the famous network embedding method LINE. We adopt its first-order proximity version LINE (1st). Besides, we also try other network embedding methods in Sect. 7.3. After obtaining the original node embeddings, we randomly select some nodes and provide them with attributes. At last, we employ different fusion methods to obtain the final fusion results.

Baseline methods: since this incomplete data fusion problem has not been previously studied, there is no natural baseline to compare with. We thus compare our methods with those methods which directly fuse the original given node embeddings and attributes. We list them as follows:

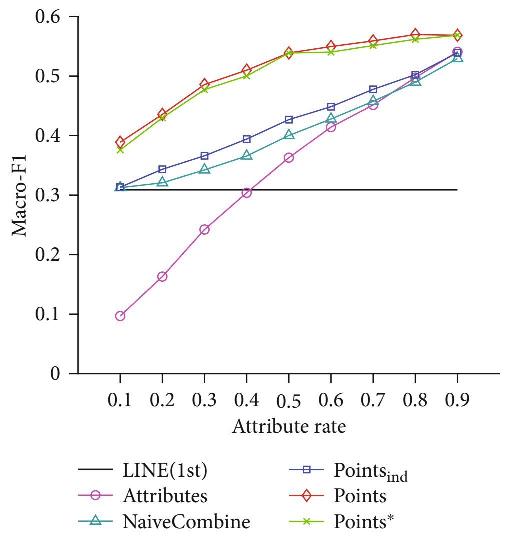
(1) LINE(1st). We adopt LINE(1st) to obtain the original node embeddings. This method neglects the incomplete node attributes. Note: in Sect. 7.3, we also try more network embedding methods

(2) Attributes. We use the zero-padded attributes as fusion results. This method neglects the given node embeddings

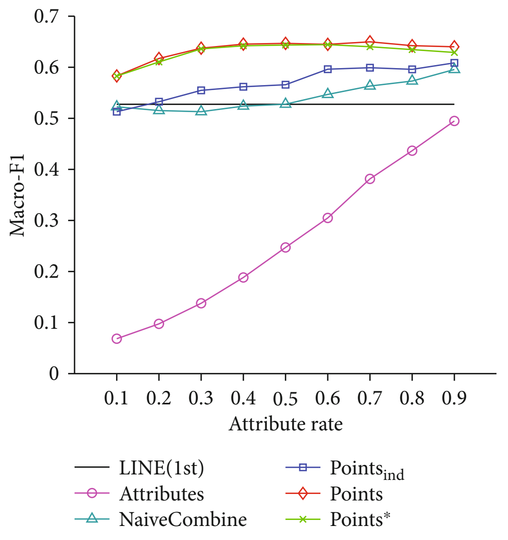
(3) NaiveCombine. We simply concatenate the vectors from the given node embeddings and the zero-padded attributes

For our method, we test its three different versions: POINTS$_{ind}$ (the inductive version formulated in Eq. (1)), POINTS (the full transductive version formulated in Eq. (4)), and POINTS* (the light version formulated in Eq. (5)).

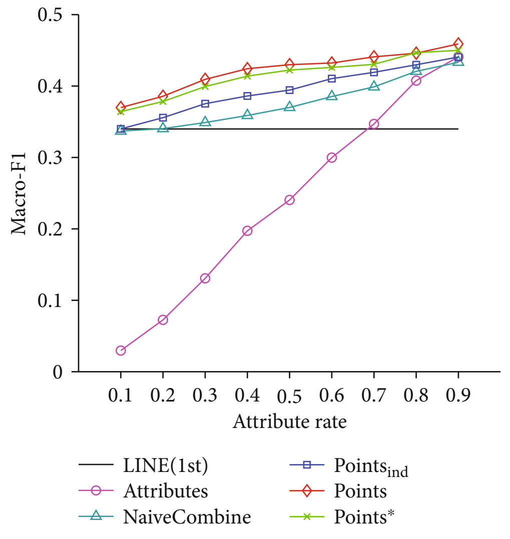
Parameters: we follow the suggestion of LINE to set the embedding dimension to 128. In addition, following [14],



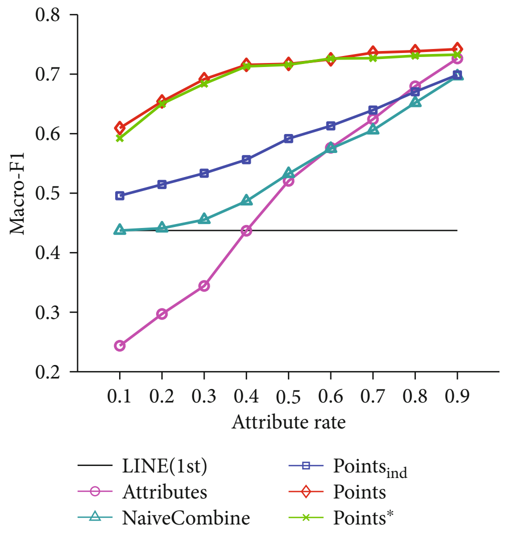
(a) Citeseer



(b) Cora



(c) Wiki



(d) Pubmed

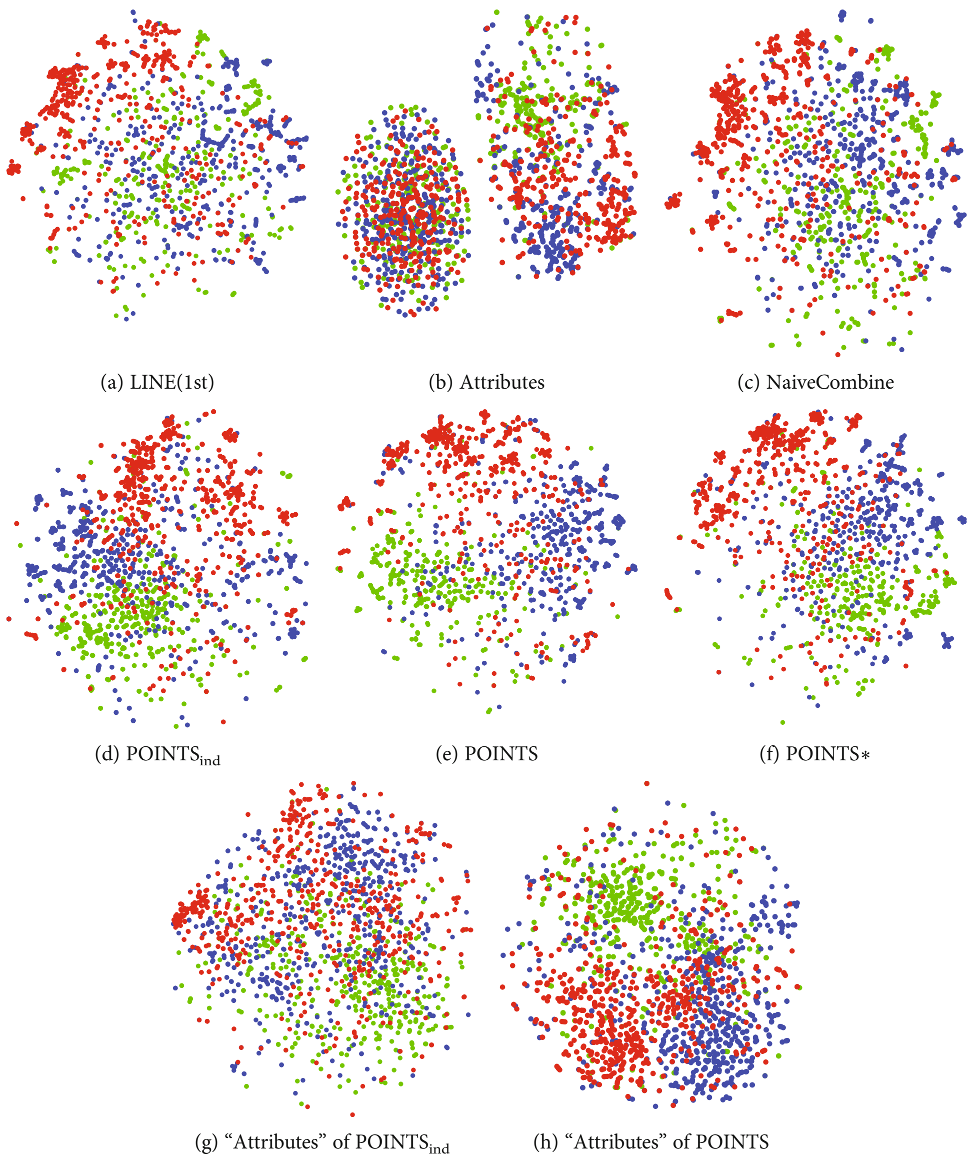
FIGURE 3: Node classification results (Macro-F1).

we reduce the dimension of attributes by applying SVD decomposition on the original text features. For simplicity, we also reduce this dimension to 128. In the proposed methods POINTS and POINTS*, we fix parameters $\alpha = 1$ and $\beta = 1$ throughout our experiments, although adjusting them would yield better results. Besides, we simply set the neighbor number $k = 5$ like most graph-based transductive methods [33] and set the candidate number $m = 20k$ for POINTS$*$.

7.1. Node Classification. Following [6], we train one-vs-rest logistic regression classifiers to evaluated the fusion (i.e., the updated embeddings) quality. Specifically, for Citeseer, Cora, and Wiki, we fix the label rate in the classifiers to 10%. Since Pubmed is a much larger dataset with fewer classes, we follow [34] to set the percentage of labeled data to 1%. In addition, we increase the rate of nodes with attributes from 10% to 90% on all datasets. Following [28], before evaluation, we normalize all representation vectors to unit length for a fair comparison. Figures 2 and 3 show the classification performance measured by micro-F1 and macro-F1 [35], respectively. We can draw the following three conclusions from these results.

Firstly, all our methods (including POINTS$_{\text{ind}}$, POINTS, and POINTS*) outperform baseline methods significantly. For example, on Citeseer with 50% attributes, POINTS$_{\text{ind}}$, which performs worst in the proposed three methods, still outperforms LINE(1st) by 13%, attributes by 8%, and NaiveCombine by 3%. Additionally, the improvements of our two transductive methods POINTS and POINTS* are more remarkable. These results clearly demonstrate the effectiveness of our complement strategy.

Secondly, the proposed two transductive methods (i.e., POINTS and POINTS*) consistently outperform our inductive method POINTS$_{\text{ind}}$. Especially on Citeseer, Cora, and Pubmed, these two transductive methods generally


(a) LINE(1st) (b) Attributes (c) NaiveCombine

(d) POINTS$_{\text{ind}}$ (e) POINTS (f) POINTS∗

(g) "Attributes" of POINTS$_{\text{ind}}$ (h) "Attributes" of POINTS

Figure 4: Visualization on Citeseer with 50% attributes. Each point stands for one document. Color of a point indicates the document category. The red indicates the topic of IR, the blue indicates the topic of DB, and the green indicates the topic of HCI. Note: the "Attributes" in (g) and (h) stands for the predicted attributes.

outperform POINTS$_{\text{ind}}$ by 5-12%. On the other hand, we also find that the improvement becomes less significant on Wiki. We conjecture that it may be hard to recover its original network structure from the given node embeddings and attributes. More specifically, this might be because Wiki (whose edge num is eight times greater than node number) is much denser than the other three datasets.

Thirdly, the light version POINTS* is comparable to POINTS on all datasets. This indicates that we can reduce the neighbor candidate set size for efficient transductive learning.

7.2. Visualization. Following [28], we use t-SNE package [36] to visualize the final node representations obtained by different fusion methods. Without loss of generality, we choose the first dataset Citeseer and test the case with 50% attributes. Similar to [28], for a clear comparison, we visualize the nodes from three different research fields: IR, DB, and HCI. Figure 4 shows the visualization results.

As shown in Figures 4(a)–4(c), the visualization results of the compared three baselines are not very meaningful, in which the points belonging to different categories are heavily mixed with each other. This is due to the fact that all these baselines cannot sufficiently utilize the incomplete attributes. In contrast, as shown in Figures 4(d)–4(f), the results of our three methods are much better (nodes with same colors are distributed closer). In addition, compared to our inductive method POINTS$_{\text{ind}}$, our two transductive methods POINTS and POINTS* show more meaningful layout. Specifically, the blue points in POINTS$_{\text{ind}}$ are partly separated by the red points, while these two types of points in POINTS and POINTS* are less mixed with each other. To clarify the reason, we further visualize the predicted attributes of POINTS$_{\text{ind}}$ and

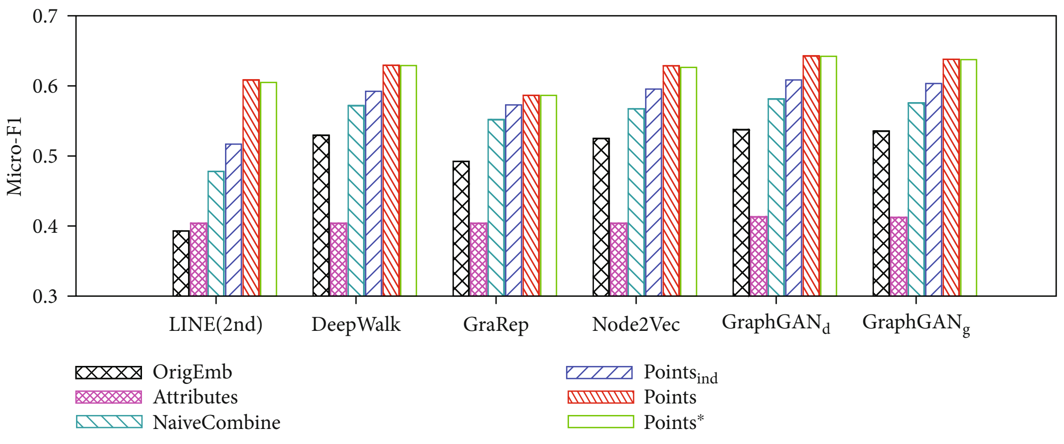


Figure 5: Average node classification performance (micro-F1) of more network embedding methods on Citeseer. LINE(2nd) [28] preserves the second-order proximity of a network to learn node embeddings. DeepWalk [6] learns node embeddings by simulating uniform random walks on networks. GraRep [37] generates node embeddings by computing high-order proximity and uses the SVD to reduce their dimensionality. Node2Vec [7] adopts a biased random walk strategy based on DeepWalk to efficiently explore neighborhood architecture. GraphGAN [13] is a newly proposed deep method which employs adversarial training in a minimax game to learn node embeddings. In this method, its discriminator function (denoted as GraphGAN$_d$) and generator function (denoted as GraphGAN$_g$) could separately obtain network embedding results.

POINTS in Figures 4(g) and 4(h), respectively. We can clearly find that POINTS could obtain high-quality attributes, which explains the superiority of our transductive methods. [21].

7.3. More Network Embedding Baselines. We evaluate the performance of our methods based on more network embedding methods. In particular, we further test another five network embedding methods as follows:

Without loss of generality, we fix the label rate to 10% and choose 50% nodes with attributes. For convenience, we use "OrigEmb" to denote the original node embeddings obtained by various network embedding methods.

Figure 5 shows the performance on Citeseer. We can clearly find that our methods (including POINTS$_{ind}$, POINTS, and POINTS*) consistently outperform baselines by a large margin. On the other hand, the light version POINTS* could always achieve similar accuracy as its full version POINTS. Taken together, all these observations clearly indicate the effectiveness of our methods.

8. Conclusion

This paper investigates the problem of fusing node embeddings and incompleted attributes provided by two different networks. We develop both inductive and transductive variants of our method. Additionally, we also provide an efficient light version of our transductive variant. Extensive experiments have demonstrated the effectiveness of our methods. In the further, we would extend our method to fuse more types of related information from more different networks and resources.

Data Availability

The datasets used in this paper can be found at https://linqs.soe.ucsc.edu/data.

Conflicts of Interest

The author(s) declare(s) that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 61902020) and Macao Youth Scholars Program (AM201912).

References

[1] X. Kong, S. Tong, H. Gao et al., "Mobile edge cooperation optimization for wearable internet of things: a network representation-based framework," *IEEE Transactions on Industrial Informatics*, p. 1, 2020.

[2] W. Wang, F. Xia, H. Nie et al., "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," in *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[3] C. Wang, C. Wang, Z. Wang, X. Ye, J. X. Yu, and B. Wang, "Deepdirect: learning directions of social ties with edge-based network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2277–2291, 2018.

[4] W. Wang, X. Zhao, Z. Gong, Z. Chen, N. Zhang, and W. Wei, "An attention-based deep learning framework for trip destination prediction of sharing bike," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2020.

[5] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, New York, New York, USA, 2014.

[7] A. Grover and J. Leskovec, "node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD*

international conference on Knowledge discovery and data mining, pp. 855–864, San Francisco California USA, 2016.

[8] S. Bourigault, S. Lamprier, and P. Gallinari, “Representation learning for information diffusion through social networks: an embedded cascade model,” in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 573–582, San Francisco California USA, 2016.

[9] A. Shakimov, H. Lim, R. Cáceres et al., “Visa-vis: privacy-preserving online social networking via virtual individual servers,” in *Proceedings of the 3rd International Conference on Communication Systems and Networks*, pp. 1–10, NW Washington DC, United States, 2011.

[10] X. Cheng, H. Li, and J. Liu, “Video sharing propagation in social networks:measurement, modeling, and analysis,” in *INFOCOM, 2013 Proceedings IEEE*, pp. 45–49, Turin, Italy, 2013.

[11] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *International Conference on Machine Learning*, pp. 1188–1196, Beijing, China, 2014.

[12] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, San Francisco California USA, 2016.

[13] H. Wang, J. Wang, J. Wang et al., “GraphGAN: graph representation learning with generative adversarial nets,” in *In Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2508–2515, New Orleans, Louisiana, USA, 2018.

[14] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, “Network representation learning with rich text information,” in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 2111–2117, Buenos Aires, Argentina, 2015.

[15] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, pp. 1024–1034, Curran Associates Inc., 57 Morehouse Lane, Red Hook, NY, United States, 2017.

[16] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

[17] Z. Wang, X. Ye, C. Wang, Y. Wu, C. Wang, and K. Liang, “RSDNE: Exploring relaxed similarity and dissimilarity from completely-imbalanced labels for network embedding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, New Orleans, Louisiana, USA, 2018.

[18] Z. Wang, X. Ye, C. Wang, J. Cui, and P. Yu, “Network embedding with completely imbalanced labels,” *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2020.

[19] A. Zhang, S. Song, and J. Wang, “Sequential data cleaning: a statistical approach,” in *Proceedings of the 2016 International Conference on Management of Data*, pp. 909–924, San Francisco, California, USA, 2016.

[20] A. Zhang, S. Song, J. Wang, and P. S. Yu, “Time series data cleaning,” *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1046–1057, 2017.

[21] O. Alter, P. O. Brown, and D. Botstein, “Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 6, pp. 3351–3356, 2003.

[22] D. F. Hsu and I. Taksa, “Comparing rank and score combination methods for data fusion in information retrieval,” *Information Retrieval*, vol. 8, no. 3, pp. 449–480, 2005.

[23] Y. Chen, J. Yang, C. Wang, and N. Liu, “Multimodal biometrics recognition based on local fusion visual features and variational bayesian extreme learning machine,” *Expert Systems with Applications*, vol. 64, pp. 93–103, 2016.

[24] J. Bleiholder and F. Naumann, “Data fusion,” *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–41, 2009.

[25] J. J. Clark and A. L. Yuille, *Data fusion for sensory information processing systems*, vol. 105, Springer Science & Business Media, 2013.

[26] T. Joachims, “Transductive inference for text classification using support vector machines,” in *International Conference on Machine Learning*, vol. 99, pp. 200–209, Bled, Slovenia, 1999.

[27] Y. Jacob, L. Denoyer, and P. Gallinari, “Learning latent representations of nodes for classifying in heterogeneous social networks,” in *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 373–382, New York, New York, USA, 2014.

[28] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, Florence, Italy, 2015.

[29] F. Wang and C. Zhang, “Label propagation through linear neighborhoods,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 55–67, 2008.

[30] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Advances in neural information processing systems*, pp. 321–328, Vancouver, British Columbia, Canada, 2004.

[31] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, “Automating the construction of internet portals with machine learning,” *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[32] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[33] X. Zhu, *Semi-supervised learning literature survey*, vol. 2, no. 3, 2006Computer Science, University of Wisconsin-Madison, 2006.

[34] Q. Li, Z. Han, and X. M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” 2018, https://arxiv.org/abs/1801.07606.

[35] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: an artificial intelligence approach*, Springer Science & Business Media, 2013.

[36] L. Van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[37] S. Cao, W. Lu, and Q. Xu, “Grarep: learning graph representations with global structural information,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 891–900, Melbourne, Australia, 2015.