TRAP: Two-level Regularized Autoencoder-based Embedding for Power-law Distributed Data

Dongmin Park, Hwanjun Song, Minseok Kim, Jae-Gil Lee*
Graduate School of Knowledge Service Engineering, KAIST
{dongminpark,songhwanjun,minseokkim,jaegil}@kaist.ac.kr

ABSTRACT

Recently, autoencoder (AE)-based embedding approaches have achieved state-of-the-art performance in many tasks, especially in top-k recommendation with user embedding or node classification with node embedding. However, we find that many real-world data follow the power-law distribution with respect to the data object sparsity. When learning AE-based embeddings of these data, dense inputs move away from sparse inputs in an embedding space even when they are highly correlated. This phenomenon, which we call *polarization*, obviously distorts the embedding. In this paper, we propose ***TRAP*** that leverages two-level regularizers to effectively alleviate the polarization problem. The *macroscopic* regularizer generally prevents dense input objects from being distant from other sparse input objects, and the *microscopic* regularizer individually attracts each object to correlated neighbor objects rather than uncorrelated ones. Importantly, *TRAP* is a *meta-algorithm* that can be easily coupled with existing AE-based embedding methods with a simple modification. In extensive experiments on two representative embedding tasks using six-real world datasets, *TRAP* boosted the performance of the state-of-the-art algorithms by up to 31.53% and 94.99% respectively.

CCS CONCEPTS

• **Computing methodologies** → **Regularization**; **Neural networks**; • **Information systems** → **Data mining**.

KEYWORDS

Power-law Distribution, Recommender System, Graph Embedding, Autoencoder

ACM Reference Format:
Dongmin Park, Hwanjun Song, Minseok Kim, Jae-Gil Lee. 2020. TRAP: Two-level Regularized Autoencoder-based Embedding for Power-law Distributed Data. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3366423.3380233

1 INTRODUCTION

With the rapid growth of online services including e-commerce and social media, a variety of high-dimensional datasets have become available, e.g., user-item transaction matrices and social relation matrices [17]. However, owing to their extremely high dimensionality, most of existing machine learning algorithms have been reported to suffer from high computational and space complexity [1]. Thus, many researchers have employed a *data embedding* technique, which maps high-dimensional data to lower-dimensional data, in order to directly apply the existing algorithms on the lower-dimensional data [8, 15, 29].

Recently, numerous autoencoder (AE)-based embedding approaches have been studied actively, and they are empirically proven to achieve not only low-dimensional but also highly informative embeddings because of the strong representation power of neural networks. Hence, it becomes feasible to capture a data manifold smoothly even in the high-dimensional data [4]. This family of AE-based approaches is well known to reach the state-of-the-art performance in numerous machine learning tasks, especially in top-k recommendation tasks with user embedding [13, 29, 30] or link prediction and node classification (or clustering) tasks with node embedding [4, 5, 8].

Despite their great success, we claim that a *skewed* sparsity distribution of input vectors (e.g., movie ratings) in real-world data severely hurts the performance of embedding methods. Specifically, as shown in Figure 1a, the sparsity distribution of input vectors is strongly skewed toward being sparse. This observation is *natural* in considering that various phenomena in the real world approximately follow a *power law* over a wide range of magnitudes [3]. However, as shown in Figure 1b, this skewed distribution entails a *polarization* problem that causes a dense input to move away from sparse inputs in an embedding space (See Section 3.2 for the details), even when the two inputs are highly correlated. Evidently, these polarized latent representations degrade the performance of the embedding methods.

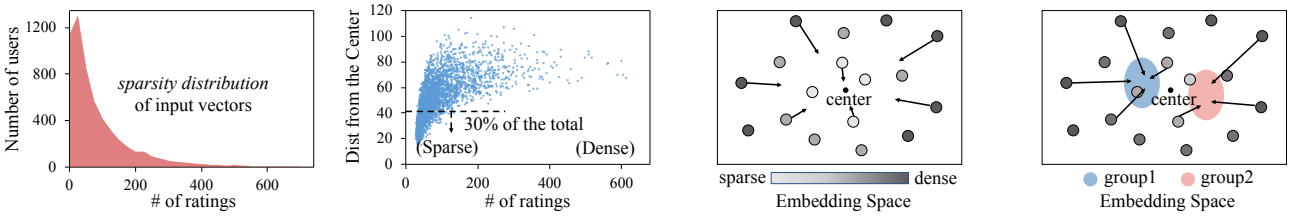
Intuitively, the deleterious effect of polarization is very common in real-world scenarios. Let's consider a recommendation task where its data follows the power-law distribution [28]. The goal of the task is to suggest unexperienced but interesting items to users based on their estimated preference under the assumption that people's tastes are highly correlated with each other. However, the polarization problem hinders dense inputs from being closely located to sparse inputs in the latent space regardless of the correlation between them, and thus, the existing embedding methods may inaccurately estimate the users' preference. This limitation calls for a new approach to alleviate the polarization problem.

To handle the polarization problem, we propose a novel *meta-algorithm* called *TRAP (Two-level Regularized Autoencoder based embedding for Power-law distributed data)* that can be easily combined with any existing autoencoder-based embedding methods leveraging *two-level* regularizers. *(i)* The "macroscopic regularizer" *generally* prevents dense input objects from being distant from

*Jae-Gil Lee is the corresponding author.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380233



(a) Sparsity Distribution. (b) Polarization. (c) Macroscopic Regularizer. (d) Microscopic Regularizer.

Figure 1: Key idea of *TRAP*: (a) shows the sparsity distributions of the number of movie ratings per user in the MovieLens-1M dataset. The distribution was severely skewed in the sparse direction (i.e., small number of ratings) and tended to follow the *power-law* distribution because the number of users drastically decreased in the dense direction (i.e., large number of ratings); (b) shows a polarization problem when using the *AutoRec* embedding method on MovieLens-1M, where each point indicates a user. In the embedding space, the users with dense ratings tended to be far from the center whereas those with sparse ratings tended to be near the center; (c) and (d) show the effect of the *macroscopic regularizer* and *microscopic regularzier*, respectively.

other sparse input objects, and *(ii)* the "microscopic regularizer" *individually* enhances each object's freedom of movement to place it closely to correlated objects rather than uncorrelated ones.

In detail, as shown in Figure 1c, the macroscopic regularizer adds a regularization term in the loss function to pull all data objects to a specific center (e.g., origin) such that they become closer to each other. Here, the dense input objects located far from the center move toward the center more aggressively whereas the sparse ones less aggressively. On the other hand, as shown in Figure 1d, the microscopic regularizer introduces a new object-wise parameter to make each object close to its correlated objects, not simply reducing the overall distance between all data objects. Notably, our proposed merger of the two-level regularizers indeed overcomes the polarization problem, thereby significantly improving the performance of popular embedding tasks. Our main contributions are summarized as follows:

- We clarify that the *power-law* distribution of input vectors' sparsity in real-world data causes the *polarization* problem, which severely hurts the performance of embedding methods. To the best of our knowledge, our work is the first to shed light on the polarization problem.
- We propose a novel approach *TRAP* equipped with two-level regularizers to alleviate the polarization problem. Most importantly, *TRAP* is a *meta-algorithm* that can be easily coupled with existing AE-based embedding methods by simply adding a regularization term and a new parameter.
- We conduct extensive experiments on two representative embedding tasks using *six* real-world datasets. Combining *TRAP* with widely-used embedding methods, we significantly boosted the performance by up to 31.53% and 94.99% in user-item and graph embedding tasks, respectively.

2 RELATED WORK

Numerous studies have been conducted to learn the low-dimensional embeddings from sparse data. Here, we briefly review the studies for two representative tasks: *(i)* user embedding in recommender systems and *(ii)* node embedding in graph mining. Note that most of the existing work has overlooked the polarization problem resulted from the inherent nature that the density of real-world inputs follows the power-law distribution. As far as we know, *TRAP* is the first method to overcome the *polarization* problem in the embedding task.

2.1 User-Item Embedding

Learning a low-dimensional embedding of users and items has been reported to make a considerable performance improvement in the top-k recommendation task [13, 30]. A popular approach is *matrix factorization* (MF) [12] that decomposes the sparse user-item rating matrix into the product of two lower-dimensional but dense matrices based on the singular value decomposition (SVD). To further improve the typical MF, Koren [11] integrated the neighbor-based method, and Mnih and Salakhutdinov [16] employed the probabilistic regularization technique. Nevertheless, their performances are limited by the *linearity* of the SVD [13].

By virtue of the *non-linearity* in neural networks, many recent researchers have attempted to design *AE-based* embedding approaches [29]. *AutoRec* [21] reconstructs only the observed ratings in a sparse user-item rating matrix based on the point-wise loss. *CDAE* [27] adopts the denoising AE [25] to force the hidden layer to discover more robust embeddings from the sparse input. *Mult-VAE* [13] adopts the variational AE (VAE) [10] and uses a multinomial log-likelihood loss, which is known to suit the top-k recommendation. *JCA* [30] introduces a joint learning paradigm with their pair-wise loss such that the AE model captures the correlation between users and items.

2.2 Node Embedding

Mapping nodes in a graph to the low-dimensional embeddings is another essential problem in numerous tasks such as link prediction and node classification. Many research efforts have been devoted to encoding the nodes in the embedding space. *DeepWalk* [19] feeds the truncated random walks of the nodes into the SkipGram model [15] to preserve the high-order proximity between the nodes. As opposed to *DeepWalk*, *LINE* [24] focuses on preserving the first (and second) order proximity. *Node2Vec* [7] introduces both breadth-first and depth-first node sampling methods to construct the context of nodes. *Struc2Vec* [20] generates a series of the weighted auxiliary graphs and then uses the biased random walks as an input of *Node2Vec*. However, their general principle based on SkipGram fails to capture the high *non-linearity* in the graph [5].

Similar to the user-item embedding, recent studies [2, 4, 5, 26] have proposed *AE*-based embedding approaches to capture the *non-linearity* in the graph. *SDNE* [26] exploits both local and global graph structures to generate the node embeddings based on the first-order and second-order proximity. *DANE* [4] maintains an additional AE to reconstruct the sparse attribute of each node. *ProGAN* [5] exploits more complex underlying proximities between the nodes generated by a generative adversarial network (GAN) [6].

3 POLARIZATION PROBLEM

3.1 Preliminary

Let $\mathbb{X} = \{X_1, X_2, \cdots, X_M\} \in \mathbb{R}^{M \times N}$ be a matrix of M objects (i.e., row vectors in $\mathbb{X}$), where the i-th object is $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,N})$ with N properties (i.e., values in a vector X_i). Then, a *low-dimensional embedding* is formally defined by Definition 3.1.

Definition 3.1. A *low-dimensional embedding* is learning a function $\boldsymbol{f}: \mathbb{X} \rightarrow \mathbb{X}'$, where $\mathbb{X} \in \mathbb{R}^{M \times N}$ is an input matrix, $\mathbb{X}' \in \mathbb{R}^{M \times N'}$ is an embedding matrix, and $N > N'$. □

For the notion of sparse embedding, the *sparsity* of the i-th object X_i is defined by Definition 3.2, where $[\cdot]$ is the Iverson bracket[1].

Definition 3.2. The *sparsity* of the i-th object X_i is formulated as in Eq. (1).

$$\mathcal{S}(X_i) = \frac{\sum_{x_{i,j} \in X_i} [x_{i,j} = 0]}{dim(X_i)} \quad \square \tag{1}$$

Then, we regard that the matrix $\mathbb{X}$ is a *sparse binary matrix* by Definition 3.3, which is a common problem setting in the recent literature for both recommender systems and graph embedding [4, 13, 21, 26, 27, 30].

Definition 3.3. A matrix $\mathbb{X} \in \mathbb{R}^{M \times N}$ is a *sparse binary matrix* if Eq. (2) holds.

$$(\forall x_{i,j} \in \mathbb{X} : x_{i,j} \in \{0, 1\}) \wedge (1/M \sum_{i=1}^{M} \mathcal{S}(X_i) \approx 1) \quad \square \tag{2}$$

3.2 Theoretical Analysis on Polarization

Let $X = (x_1, x_2, \cdots, x_N)$ be a random vector where each element $x_j \in X$ is a binary random variable such that $P(x_j = 1) = p_X$ and $P(x_j = 0) = (1 - p_X)$. Besides, let $f(X_s)$ and $f(X_d)$ be the embedding vectors of two random vectors X_s and X_d with different sparsity such that $E[S(X_s)] > E[S(X_d)]$. Then, the notion of *polarization* is formalized as Definition 3.4, in which the *total variance* [18, 23] in Definition 3.5 is used as a measure of the dispersion from a center $E[f(X)]$ to a given embedding vector $f(X)$. That is, the higher the total variance is, the more $f(X)$ is disperse from the center.

Definition 3.4. Polarization is a phenomenon that the embedding vector of dense inputs tends to be located farther from the center than that of sparse inputs, which is proven by Theorem 3.9. □

Definition 3.5. The *total variance* of an embedding vector $f(X)$ is the sum of all eigenvalues λ of the cov$(f(X))$ as in Eq. (3).

$$\text{totvar}(f(X)) = \sum_{i=1}^{N'} \lambda_i = tr(\text{cov}(f(X))) \quad \square \tag{3}$$

[1]The Iverson bracket $[P]$ returns 1 if P is true; 0 otherwise.

We theoretically prove the existence of polarization, assuming more realistic network architectures step by step: a 1-layer linear AE (Lemma 3.6), a 1-layer *non*-linear AE (Lemma 3.7), and a *multi*-layer *non*-linear AE (Lemma 3.8).

Lemma 3.6. *Let $f(X) = WX + b$ be a 1-layer AE with N' hidden units and an identity activation function, where $W \in \mathbb{R}^{N' \times N}$ and $b \in \mathbb{R}^{N'}$. Then, Eq. (4) holds.*

$$E[\mathcal{S}(X_s)] > E[\mathcal{S}(X_d)] \rightarrow \text{totvar}(f(X_s)) < \text{totvar}(f(X_d)) \tag{4}$$

Proof. By Definition 3.5, the total variance of the embedding vector $f(X)$ is derived by Eq. (5).

$$\begin{aligned} \text{totvar}(f(X)) &= tr(\text{cov}(WX + b)) = tr(\text{cov}(WX)) \\ &= tr(W\text{cov}(X)W^T) = tr(\text{cov}(X)W^T W) \end{aligned} \tag{5}$$

Since each element $x_j \in X$ is a binary random variable, x_j follows a Bernoulli distribution $\mathcal{B}(1, p_X)$, where $p_X = (1 - E[\mathcal{S}(X)])$. Then, by $E[\mathcal{S}(X_s)] > E[\mathcal{S}(X_d)]$, $0 < p_{X_s} < p_{X_d} < 1/2$, and both (cov$(X_d)$ − cov(X_s)) and $W^T W$ are positive semidefinite. Therefore, Eq. (6) holds because $tr(AB) > 0$ if A and B are positive semidefinite. This concludes the proof.

$$\begin{aligned} &\text{totvar}(f(X_d)) - \text{totvar}(f(X_s)) \\ &= tr(\text{cov}(X_d)W^T W) - tr(\text{cov}(X_s)W^T W) \\ &= tr((\text{cov}(X_d) - \text{cov}(X_s))W^T W) > 0 \quad \square \end{aligned} \tag{6}$$

Lemma 3.7. *Let $ReLU(Z) = \max(0, Z)$ be a rectified linear unit (ReLU) activation function, where $Z = f(X)$ is a 1-layer embedding. Then, Eq. (7) holds.*

$$\begin{aligned} &\text{totvar}(Z_s) < \text{totvar}(Z_d) \rightarrow \\ &\quad \text{totvar}(ReLU(Z_s)) < \text{totvar}(ReLU(Z_d)) \end{aligned} \tag{7}$$

Proof. Let's assume that Z_s and Z_d follow the multivariate normal distribution, where $E[Z_s] = E[Z_d]$ and $\text{cov}_{i,i}(Z_s) < \text{cov}_{i,i}(Z_d)$ for all i-th diagonal elements of the covariance matrices.

(i) $E(Z_s) \geq 0$: Let Y be $\min(Z, 0)$, and U be Z if $Z \geq 2E[Z_s]$, $E[Z_s]$ if $2E[Z_s] > Z \geq 0$, and Y otherwise. Then, $\text{cov}_{i,i}(ReLU(Z)) = \text{cov}_{i,i}(Z) - \text{cov}_{i,i}(Y)$. Since $E[U] = E[Z_s] = E[Z_d]$, by the definition of the covariance, Eq. (8) holds.

$$\begin{aligned} &\text{cov}_{i,i}(ReLU(Z_d)) - \text{cov}_{i,i}(ReLU(Z_s)) \\ &= \text{cov}_{i,i}(Z_d) - \text{cov}_{i,i}(Z_s) - (\text{cov}_{i,i}(Y_d) - \text{cov}_{i,i}(Y_s)) \\ &> \text{cov}_{i,i}(Z_d) - \text{cov}_{i,i}(Z_s) - (\text{cov}_{i,i}(U_d) - \text{cov}_{i,i}(U_s)) > 0 \end{aligned} \tag{8}$$

(ii) $E(Z_s) < 0$: The proof is similar to the above case.

Therefore, $\text{cov}_{i,i}(ReLU(Z_d)) > \text{cov}_{i,i}(ReLU(Z_s))$ for every i-th diagonal element. This concludes the proof. □

Lemma 3.8. *Let's consider $f'(Z') = W'Z' + b'$, where $W' \in \mathbb{R}^{N' \times N''}$, $b' \in \mathbb{R}^{N''}$, and Z' is the embedding passing through $ReLU(\cdot)$ in the previous layer. Then, Eq. (9) holds.*

$$\begin{aligned} &\forall i, \text{cov}_{i,i}(Z'_s) < \text{cov}_{i,i}(Z'_d) \rightarrow \\ &\quad \text{totvar}(f'(Z'_s)) < \text{totvar}(f'(Z'_d)) \end{aligned} \tag{9}$$

Proof. By Eq. (6) and the condition of Eq. (9), Eq. (10) holds.

$$\begin{aligned} &\text{totvar}(f'(Z'_d)) - \text{totvar}(f'(Z'_s)) \\ &= tr((\text{cov}(Z'_d) - \text{cov}(Z'_s))W'^T W') > 0 \quad \square \end{aligned} \tag{10}$$

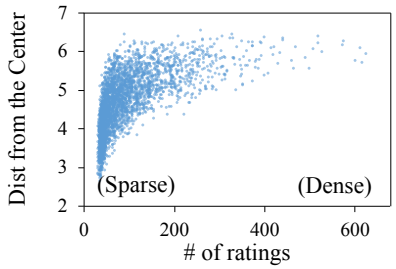


Figure 2: Polarization problem when using *AutoRec* with two layers on the MovieLens-1M dataset.

Finally, the overall proof of the polarization on a multi-layer non-linear AE is concluded by Theorem 3.9.

Theorem 3.9. *Let $f(X)$ be a multi-layer AE with the ReLU activation function. Then,* $\text{totvar}(f(X_d)) > \text{totvar}(f(X_s))$.

Proof. By the mathematical induction in which Lemma 3.6 is the base step and Lemmas 3.7 and 3.8 are the consecutive inductive steps, the polarization still holds. □

Figure 2 shows the distance from the center to the embedding vector according to the number of a user's ratings in the MovieLens-1M dataset when using *AutoRec* with *two* layers. Similar to the result of *AutoRec* with one layer in Figure 1b, the users with dense ratings tended to be located farther than those with sparse ratings. That is, we empirically confirm Theorem 3.9.

4 TWO REGULARIZERS OF *TRAP*

The key idea of *TRAP* is to constrain the distance between sparse and dense objects so that they are not too far from each other, which is achieved by two simple and effective regularizers: *(i)* the *macroscopic regularizer* that generally restricts dense objects being away from sparse objects, and *(ii)* the *microscopic regularizer* that individually adjusts each object's movement to neighbor with correlated objects.

4.1 Macroscopic Regularizer

To handle the polarization problem, an intuitive method is to constrict the dispersion of data objects by pulling them into the center of their correlated group in the embedding space. However, this process is not straightforward because the clustering task may induce several difficulties including high computation cost and low quality of the result [22].

In this regard, we introduce a *macroscopic regularizer* that simply pulls all objects into the *origin* (i.e., zero vector) in the embedding space. Specifically, as shown in Eq. (11), this regularizer adds the total $L2$-norm of all embeddings z as a penalty term with its scaling hyperparameter η in the objective function, where AE′ is any existing AE-based method, $\mathcal{L}'$ is its reconstruction loss function, and L is the number of layers of the encoder in AE′.

$$\mathcal{L} = \sum_{i=1}^{M} \left(\mathcal{L}'(X_i, AE'(X_i)) + \eta \sum_{l=1}^{L} \| z_i^{[l]} \|_2^2 \right) \tag{11}$$

Since the dense objects which tend to be far from the origin have a higher $l2$-norm than the sparse objects which tend to be close from the origin, the dense ones are pulled strongly whereas the sparse ones are pulled weakly. Consequently, the macroscopic regularizer generally restricts the dense objects being distant from other sparse objects.

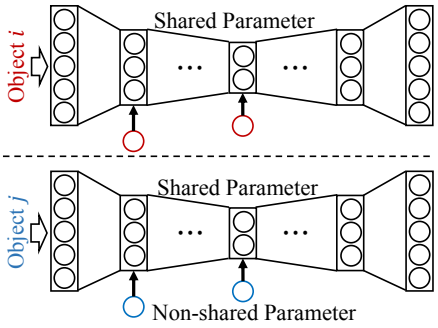


Figure 3: Structure of *TRAP*. The red (or blue) neuron means a "non-shared" object-wise scaling parameter which is assigned to each object.

4.2 Microscopic Regularizer

One limitation of the macroscopic regularizer lies in that it is applied invariably to the input data as parameter update is shared by all objects, while each object is not equally distant from the rest correlated objects in the embedding space. To achieve the goal of embedding which is to locate correlated objects nearby, we additionally introduce a *microscopic regularizer* that adjusts each object's sparsity by adding *non-shared* parameters v called *object-wise scaling parameters*, as shown in Figure 3. More specifically, as shown in Eq. (12), the object-wise scaling parameter v_i^l of the object X_i is multiplied with the output of neurons in the l-th layer before the non-liner activation σ,[2] where $\circ$ is the element-wise product.

$$z_i^{[l]} = \begin{cases} \sigma((W^{[l]}X_i + b^{[l]}) \circ v_i^{[l]}), & if\ l = 1 \\ \sigma((W^{[l]}z_i^{[l-1]} + b^{[l]}) \circ v_i^{[l]}), & otherwise \end{cases} \tag{12}$$

That is, when applied with the macroscopic regularizer, the microscopic regularizer provides additional capacity to let an input object become closer to its correlated objects in an object-wise manner.

4.3 Quick Analysis on *TRAP*

We contend that the polarization problem in Definition 3.4 can be alleviated by *TRAP* based on the following explanation. Let's recall Lemma 3.6, but the AE function is now defined as $f'(X) = (Wx + b) \circ v$ with the v of the microscopic regularizer. Then, Eq. (6) is changed to Eq. (13).

$$\begin{aligned} &\text{totvar}(f'(X_d)) - \text{totvar}(f'(X_s)) \\ &= tr(\text{cov}(X_d)(W \circ v_d)^T(W \circ v_d)) - tr(\text{cov}(X_s)(W \circ v_s)^T(W \circ v_s)) \end{aligned} \tag{13}$$

In this case, note that Lemma 3.6 does not always hold because the total variance of the embedding vector $f'(X_d)$ can be rather smaller than that of the embedding vector $f'(X_s)$ if v_d of the dense object X_d is much smaller than v_s of the sparse object X_s. With the

[2]We used the *tanh* function as the activation in all experiments.

Table 1: Summary statistics of the user feedback datasets.

Dataset	# Users	# Items	# Feedbacks	Sparsity	Skewness
ML1M	6,027	3,062	574,026	0.969	2.888
Yelp	12,705	9,245	318,314	0.997	6.310
Video	19,056	9,073	184,609	0.999	22.073

macroscopic regularizer, v_d was generally much smaller than v_s according to our ablation study in Section 5.1.6. Therefore, *TRAP* can indeed overcome the polarization problem.

5 EXPERIMENTS

To validate the superiority of *TRAP*, we performed extensive experiments on *two* independent tasks: *(i)* user-item embedding and *(ii)* node embedding. *TRAP* as well as the other algorithms were implemented using TensorFlow 1.8.0 and executed using a single NVIDIA Titan Volta GPU. For reproducibility, we provide the source code at https://github.com/kaist-dmlab/TRAP.

5.1 User-Item Embedding

5.1.1 Datasets. We performed a user-item embedding task on *three* user feedback datasets: MovieLens-1M (ML1M)[3], Yelp[4], and VideoGame (Video)[5]. Here, the rating or purchase history of the user i for the item j corresponds to the value of $x_{i,j} \in \mathbb{X}$. For the embedding task, we converted all users' ratings or histories into binary values $\in \{0, 1\}$ following the problem setting in Section 3.1, where 0 is negative feedback and 1 is positive feedback. Specifically, in ML1M and Yelp, a user-item rating was converted to 1 if it is greater than or equal to 4 and to 0 otherwise. In VideoGame, all items purchased less than 5 times and all users who purchased less than 5 items were excluded by the pre-processing step, and then a user-item history was binarized to 1 if the user purchased the item and to 0 otherwise. In addition, for each dataset, we randomly selected 70% of all user-item values as the training set, and 10% of them as the validation set, and the rest 20% of them as the test set. Table 1 summarizes the statistics of each dataset. Note that our sparsity assumption in Definition 3.3 is likely to hold in real-world user feedback data as the sparsity of data is almost 1 in Table 1. Also, the positive skewness[6] in Table 1 implies that the sparsity of each object approximately follows the power law.

5.1.2 Algorithms.

- *MF* [12]: A traditional matrix factorization model with mean square error (MSE) loss and alternating least squares (ALS) for the optimization.
- *NCF* [9]: A combination model of an extended neural network-based MF model and a multi-layer perceptron (MLP) model.
- *AutoRec* [21]: A 1-layer AE model using user rating vectors as inputs. Because the loss of *AutoRec* is devised for the explicit feedback, we converted it into the pairwise loss in *JCA*, which is proven to be more effective in top-k rocommendation with implicit feedback.
- *CDAE* [27]: A 1-layer denoising AE model. We chose the hinge-based pairwise loss introduced in *CDAE*.
- *MultiVAE* [13]: A multi-layer VAE model. We used the multinomial probabilistic loss presented in *MultiVAE*.
- *JCA* [30]: A joint model of user-based *AutoRec* and item-based *AutoRec* that fully takes advantage of user-item correlation. We used the pairwise loss proposed in *JCA*.

We combined *TRAP* with *AutoRec*, *CDAE*, *MultiVAE*, and *JCA*, each of which is denoted as $TRAP_{Auto}$, $TRAP_{CDAE}$, $TRAP_{Multi}$, and $TRAP_{JCA}$, respectively. We validated the performance improvement of the combined ones compared with the original ones.

5.1.3 Experiment Setting. The hyperparameters of all compared algorithms were favorably set to be the best values reported in the original papers [9, 12, 13, 21, 27, 30]. Regarding *TRAP*, the weight η for the macroscopic regularizer was set to be the best value found by a grid $\eta \in [3\times10^{-3}, 1\times10^{-4}, 3\times10^{-3}, 1\times10^{-3}, 1\times10^{-2}]$, and the object-wise weight v for the microscopic regularizer was initialized as the values randomly drawn from a uniform distribution $\mathcal{U}(0, 1)$. As for the training configuration, we used an Adam optimizer, a mini-batch size of 1500, and a constant learning rate of 0.003.

5.1.4 Evaluation Metrics. To measure the performance of the user-item embedding, we performed a top-k recommendation task, which is a popular way to validate the quality of obtained user embeddings [29]. The top-k recommendation is to suggest k unseen items to each user. Let $\mathbb{I}_i^*$ and $\hat{\mathbb{I}}_i(k)$ be the ground truth and k recommended items for the i-th user, where $|\mathbb{I}_i^*| \geq k$. Then, given M users and N items, the recommendation performance is typically measured by the following *four* metrics:

$$Precision@k = \frac{1}{M}\sum_{i=1}^{M}\frac{|\mathbb{I}_i^* \cap \hat{\mathbb{I}}_i(k)|}{k}; \tag{14}$$

$$Recall@k = \frac{1}{M}\sum_{i=1}^{M}\frac{|\mathbb{I}_i^* \cap \hat{\mathbb{I}}_i(k)|}{|\mathbb{I}_i^*|}; \tag{15}$$

$$F1\text{-}score@k = \frac{(2 \cdot Precision@k \cdot Recall@k)}{(Precision@k + Recall@k)}; \text{ and} \tag{16}$$

$$NDCG@k = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{IDCG(|\mathbb{I}_i^*|)}\sum_{j^+\in\mathbb{I}_i^*}\frac{1}{log_2(rank(j^+)+2)}, \quad \text{where } IDCG(|\mathbb{I}_i^*|) = \sum_{l=1}^{|\mathbb{I}_i^*|}\frac{1}{log_2(l+2))}. \tag{17}$$

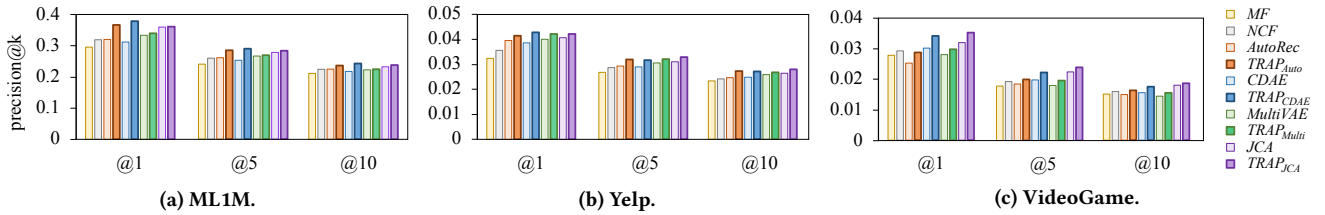
In support of reliable evaluation, we repeated every task *five* times and reported the average of each metric.

5.1.5 Performance Comparison. Table 2 shows $F1\text{-}score@k$ and $NDCG@k$ on three datasets with varying k. Overall, the performance of all existing AE-based embedding methods was *significantly* improved by incorporating *TRAP* into them regardless of k values. Quantitatively, compared with the original method, $F1\text{-}score@k$ was improved by up to 20.38% in ML1M, 24.16% in Yelp, and 31.53% in VideoGame; $NDCG@k$ was improved by up to 21.42% in ML1M, 21.51% in Yelp, and 25.76% in VideoGame. In particular, the best performance improvement was mostly achieved with *CDAE*. However, when combined with *MultiVAE*, the improvement

[3] http://files.grouplens.org/datasets/movielens/ml-1m.zip
[4] http://www.yelp.com/dataset/challenge
[5] http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/ratings_Video_Games.csv
[6] The skewness was measured by Pearson's moment coefficient of skewness.

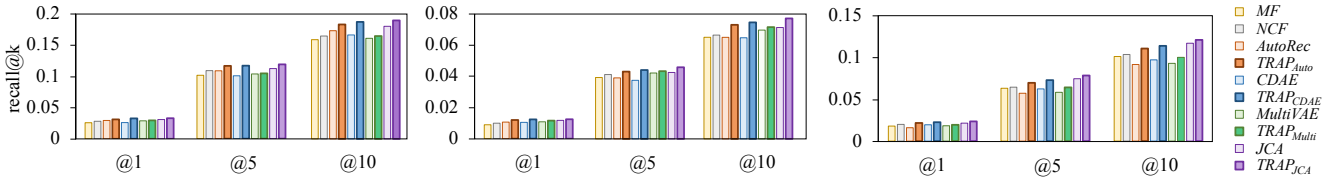
Table 2: *F1-score@k* and *NDCG@k* of all user-item embedding algorithms on three datasets (the best results are marked in bold).

Metrics	F1-score									NDCG								
Datasets	ML1M			Yelp			VideoGame			ML1M			Yelp			VideoGame		
Models	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10
MF	.0469	.1418	.1801	.0140	.0318	.0343	.0214	.0269	.0253	.2778	.2510	.2472	.0351	.0372	.0477	.0252	.0425	.0510
NCF	.0511	.1525	.1887	.0155	.0337	.0354	.0223	.0287	.0266	.2955	.2727	.2709	.0378	.0390	.0496	.0279	.0444	.0561
AutoRec	.0499	.1486	.1956	.0167	.0334	.0357	.0192	.0271	.0248	.3197	.2813	.2735	.0394	.0408	.0493	.0242	.0411	.0524
$TRAP_{Auto}$	.0568	.1644	.2051	.0185	.0366	.0396	.0253	.0316	.0284	.3660	.3126	.2950	.0413	.0449	.0551	.0305	.0504	.0638
%improve	*13.83*	*10.61*	*4.85*	*10.89*	*9.55*	*11.14*	*31.53*	*16.80*	*14.31*	*14.48*	*11.14*	*7.87*	*4.79*	*10.00*	*11.63*	*25.76*	*22.48*	*21.67*
CDAE	.0475	.1431	.1873	.0155	.0323	.0352	.0233	.0291	.0259	.3114	.2721	.2634	.0366	.0396	.0482	.0290	.0457	.0571
$TRAP_{CDAE}$	.0572	.1647	**.2106**	.0193	.0373	**.0402**	.0267	.0330	.0293	**.3781**	**.3182**	**.3025**	.0445	.0458	.0561	.0328	.0527	.0661
%improve	*20.38*	*15.11*	*12.42*	*24.16*	*15.50*	*14.18*	*14.76*	*13.32*	*13.24*	*21.42*	*16.93*	*14.82*	*21.51*	*15.72*	*16.44*	*13.22*	*15.13*	*15.81*
MultiVAE	.0526	.1483	.1857	.0168	.0347	.0376	.0218	.0266	.0241	.3330	.2893	.2726	.0399	.0424	.0518	.0269	.0425	.0539
$TRAP_{Multi}$	.0538	.1496	.1888	.0171	.0367	.0391	.0231	.0291	.0259	.3393	.2919	.2756	.0420	.0451	.0543	.0296	.0462	.0579
%improve	*2.35*	*0.91*	*1.66*	*1.53*	*5.93*	*3.95*	*6.22*	*9.20*	*7.60*	*1.89*	*0.91*	*1.12*	*5.33*	*6.36*	*4.80*	*6.24*	*8.56*	*7.54*
JCA	.0563	.1589	.2018	.0182	.0358	.0385	.0252	.0334	.0301	.3592	.3044	.2892	.0405	.0440	.0537	.0307	.0526	.0667
$TRAP_{JCA}$	**.0599**	**.1665**	.2098	**.0210**	**.0384**	.0401	**.0277**	**.0355**	**.0312**	.3605	.3109	.2973	**.0464**	**.0477**	**.0573**	**.0338**	**.0562**	**.0702**
%improve	*6.28*	*4.81*	*3.96*	*15.82*	*7.20*	*4.30*	*9.97*	*6.26*	*3.51*	*0.37*	*2.11*	*2.80*	*14.56*	*8.39*	*6.60*	*10.26*	*6.82*	*5.19*



(a) ML1M. **(b) Yelp.** **(c) VideoGame.**

Figure 4: *Precision@k* of all user-item embedding algorithms on three datasets.



(a) ML1M. **(b) Yelp.** **(c) VideoGame.**

Figure 5: *Recall@k* of all user-item embedding algorithms on three datasets.

was relatively low because the effect of our macroscopic regularizer overlaps with that of the Gaussian prior $N(0, I)$ constraint to the embeddings in the original MultiVAE model. Nevertheless, the synergistic effect with our microscopic regularizer is still effective. Meanwhile, the best performance on the two metrics was achieved by either $TRAP_{CDAE}$ or $TRAP_{JCA}$ in all datasets. Figures 4 and 5 show *precision@k* and *recall@k* of all algorithms on three datasets with varying k. The performance trends with them were similar to those with *F1-score@k* and *NDCG@k*. Most importantly, this consistent dominance of the combined methods empirically proves that mitigating the polarization problem significantly improves the performance of the embedding task.

5.1.6 Ablation Study. To individually examine the effect of the macroscopic and microscopic regularizers, we conducted additional ablation experiments on two methods, *AutoRec* and *CDAE*. We first show the performance of each method's vanilla version and then enable the components of *TRAP* to solely evaluate the effectiveness of each component. We denote the versions combined with

Table 3: Ablation study for the two regularizers of *TRAP* (the best results are marked in bold).

	ML1M			Yelp			VideoGame		
Models	@1	@5	@10	@1	@5	@10	@1	@5	@10
AutoRec	.0499	.1486	.1956	.0167	.0334	.0357	.0192	.0271	.0248
$TRAP^{macro}_{Auto}$	.0566	.1585	.2009	.0173	.0345	.0378	.0223	.0292	.0264
$TRAP^{micro}_{Auto}$	.0563	.1640	.2048	.0177	.0359	.0382	.0241	.0310	.0280
$TRAP_{Auto}$	**.0568**	**.1644**	**.2051**	**.0185**	**.0366**	**.0396**	**.0253**	**.0316**	**.0284**
CDAE	.0475	.1431	.1873	.0155	.0323	.0352	.0233	.0291	.0259
$TRAP^{macro}_{CDAE}$	.0561	.1533	.1929	.0160	.0333	.0354	.0233	.0306	.0273
$TRAP^{micro}_{CDAE}$	.0564	.1561	.2056	.0161	.0338	.0376	.0240	.0313	.0288
$TRAP_{CDAE}$	**.0572**	**.1647**	**.2106**	**.0193**	**.0373**	**.0402**	**.0267**	**.0330**	**.0293**

Table 4: Performance comparison of $TRAP^{macro}$ with two normalization methods (the best results are marked in bold).

	ML1M			Yelp			VideoGame		
Models	@1	@5	@10	@1	@5	@10	@1	@5	@10
AutoRec	.0499	.1486	.1956	.0167	.0334	.0357	.0192	.0271	.0248
Sphere-norm	.0405	.1115	.1431	.0051	.0101	.0111	.0044	.0055	.0057
L1-norm	.0512	.1508	.1936	.0148	.0332	.0360	.0190	.0260	.0242
$TRAP^{macro}_{Auto}$	**.0566**	**.1585**	**.2009**	**.0173**	**.0345**	**.0378**	**.0223**	**.0292**	**.0264**
CDAE	.0475	.1431	.1873	.0155	.0323	.0352	.0233	.0291	.0259
Sphere-norm	.0343	.0992	.1313	.0064	.0159	.0130	.0093	.0193	.0219
L1-norm	.0545	.1530	.1918	.0156	.0330	.0353	.0222	.0300	.0264
$TRAP^{macro}_{CDAE}$	**.0561**	**.1533**	**.1929**	**.0160**	**.0333**	**.0354**	**.0233**	**.0306**	**.0273**

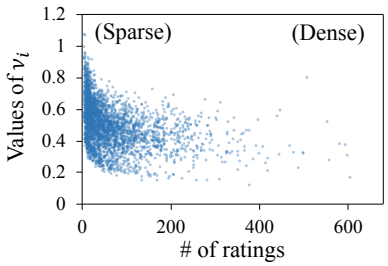
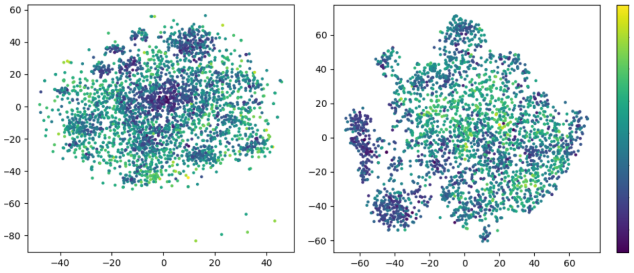


Figure 6: Distribution of the object-wise scaling parameter v_i according to the number of movie ratings of each user i when training $TRAP_{Auto}$ on the ML1M dataset.

both regularizers as $TRAP_{Auto}$ and $TRAP_{CDAE}$, respectively. Combination with either the macroscopic or microscopic regularizer is additionally indicated by the superscript, either *macro* or *micro*, on the name of each combined method. Table 3 reports the $F1\text{-}score@k$ evaluation results of $TRAP_{Auto}$ and $TRAP_{CDAE}$. For all three datasets and k values, each regularizer leads to significant performance improvement. Moreover, the best performance improvement is achieved when both regularizers are combined. This result indicates that our proposed two-level regularizers have the synergistic effect to alleviate the polarization problem.

5.1.7 Macroscopic Regularizer vs. Other Normalizations. To validate the advantage of our macroscopic regularizer (i.e., *l2-normalization*) over other normalization methods, we conducted additional experiments with *l1-normalization* and *hypersphere-normalization* which force all embeddings to be located on the surface of a unit hypersphere. Table 4 shows $F1\text{-}score@k$ of three normalization methods combined with *AutoRec* and *CDAE* on three datasets. Overall, our macroscopic regularizer always showed the highest improvement for both *AutoRec* and *CDAE*. The *l1-normalization* improved the performance, but its effect was weaker than that of our macroscopic regularizer; the *hypersphere-normalization* was shown to even degrade the performance.

5.1.8 Effect of Microscopic Regularizer. Furthermore, we plot the distribution of the object-wise scaling parameter v_i with respect to the object's input sparsity (i.e., the number of movie ratings) on ML1M, as shown in Figure 6. As an input object became denser, the parameter value decreased. By virtue of our design principle in Section 4.2, a large v_i of the sparse objects forces them to move away from the origin, while a small v_i of the dense objects forces them to stay close to the origin. Therefore, v_i properly makes the correlated objects close in the embedding space, regardless of their input sparsity. Figure 7 represents the TSNE visualizations [14] over the learned embeddings of *AutoRec* and $TRAP_{Auto}$. The change from Figure 7a to Figure 7b shows that polarization was effectively resolved by combining with *TRAP*.



(a) *AutoRec*. **(b) $TRAP_{Auto}$.**

Figure 7: TSNE visualizations of the learned embeddings from *AutoRec* and $TRAP_{Auto}$ on the ML1M dataset. The color shows the normalized sparsity of each data object, where yellow indicates being dense and navy blue being sparse.

Table 5: Summary statistics of the graph datasets.

Dataset	# Nodes	# Edges	# Classes	Attribute	Sparsity	Skewness
Cora	2,708	5,429	7	O	0.999	15.271
Cite	3,312	4,732	6	O	0.999	9.989
Blog	10,312	333,983	37	X	0.999	9.823

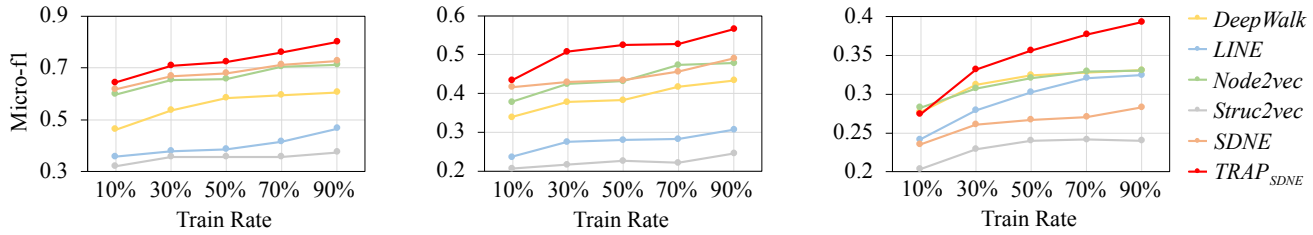
5.2 Node Embedding

5.2.1 Datasets. We performed a node embedding task on *three* benchmark graph datasets: Cora[7], Citeseer (Cite)[8], and BlogCatalog (Blog)[9]. Both Cora and Citeseer are the citation graphs between academic papers, and BlogCatalog is the friendship graph between social network users. Differently to the user feedback dataset, the value of $x_{i,j} \in \mathbb{X}$ was originally 1 if a relationship (e.g., citation and friendship) from the node i to the other node j existed and 0 otherwise. Besides, additional attribute information was available in the Cora and Citeseer datasets, but not in the BlogCatalog dataset. The remaining configuration for data preparation was the same as in Section 5.1.1. Table 5 summarizes the statistics of each dataset. Our assumptions are also likely to hold in the graph datasets because the sparsity of data is again almost 1.

5.2.2 Algorithms.

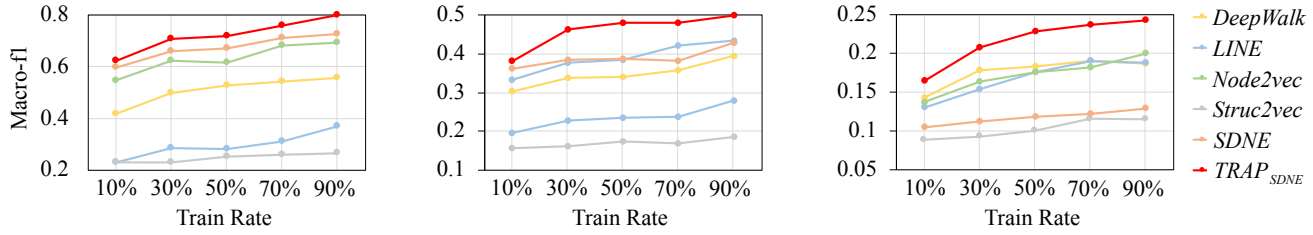
- *DeepWalk*[19]: A basic random walks-based model that uses the SkipGram embedding architecture.
- *LINE* [24]: A random walks-based model that focuses on preserving the first-order and second-order proximities.
- *Node2Vec* [7]: A random walks-based model that uses breadth-first and depth-first sampling instead of random walk proximity sampling.

[7] http://www.cs.umd.edu/~sen/lbc-proj/data/cora.tgz
[8] http://www.cs.umd.edu/~sen/lbc-proj/data/citeseer.tgz
[9] http://socialcomputing.asu.edu/datasets/BlogCatalog3



(a) Cora. (b) Citeseer. (c) BlogCatalog.

Figure 8: *Micro-f1* scores of six graph embedding algorithms on three datasets.



(a) Cora. (b) Citeseer. (c) BlogCatalog.

Figure 9: *Macro-f1* scores of six graph embedding algorithms on three datasets.

- *Struc2Vec* [20]: A model that uses the samples obtained by biased random walks as the input of *Node2Vec*.
- *SDNE* [26]: A multi-layer AE model that exploits the first-order and second-order proximities simultaneously.
- *DANE* [4]: A multi-layer AE model that aggregates additional attribute information of each node.

Again, the two popular AE-based methods, *SDNE* and *DANE*, were combined with *TRAP*, each of which is denoted as $TRAP_{SDNE}$ and $TRAP_{DANE}$ respectively. We validated their performance improvement compared with the original ones.

5.2.3 Experiment Setting. The hyperparameters of all compared algorithms were set to be the best values in the original papers [4, 7, 19, 20, 24, 26]. Regarding *TRAP*, the object-wise weight v was initialized as the values randomly drawn from a uniform distribution $\mathcal{U}(0, 1)$, and the hyperparameter η was set to be the best value found by a grid $\eta \in [3 \times 10^{-5}, 1 \times 10^{-5}, 3 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-3}]$. As for the training configuration, except for the mini-batch size, the configuration was the same as in Section 5.1.2. The mini-batch size was set to be 500 because the graph data was smaller than the user feedback data.

5.2.4 Evaluation. To measure the performance of the node embedding, we performed two popular tasks as follows:

- **Node Classification**: This task aims to provide a high-quality labeling for every node using only a few labeled nodes. Typically, the learned embeddings are used as the input of a simple classifier, and its accuracy is commonly measured by both *micro-f1* and *macro-f1* scores. Refer to Wang et al. [26] for details. For the classification task, we generated five test cases by splitting the entire data into the training and test data with varying ratios of the training data in {10%, 30%, 50%, 70%, 90%}. Then, we performed the task using two simple classifiers: OneVsRestClassifier and LogisticRegressor of scikit-learn[10].
- **Graph Reconstruction**: This task validates how well the learned embeddings preserve the structural information of the graph. For the performance evaluation, we adopted a widely-used metric $precision_{GR}@k$, which indicates how many k-nearest neighbors retrieved using each learned embedding match the true adjacent nodes in the graph. Given the adjacency matrix $\mathbb{X}$ of M users (i.e., the binarized matrix in Section 5.2.1), $precision_{GR}@k$ is defined by Eq. (18), where (i, j) is a pair of users. Since *DANE* uses additional attribute information unlike other embedding methods, it was excluded from the overall comparison for fairness.

$$Precision_{GR}@k = \frac{1}{k} \sum_{i,j} |\{rank(i,j) < k\} \cap \{\mathbb{X}(i,j) = 1\}| \quad (18)$$

In support of reliable evaluation, we repeated every task *five* times and reported the average of each metric.

5.2.5 Performance Comparison.

- **Node Classification**: Figures 8 and 9 show the *micro-f1* and *macro-f1* scores of all algorithms on three datasets with varying ratios of training data. Overall, $TRAP_{SDNE}$ showed the best performance at any ratio of training data in all datasets. Even if *SDNE* was much worse than other existing methods in BlogCatalog, $TRAP_{SDNE}$ was turned to be the best because of the remarkable performance gain obtained by both regularizers. The improvement reached by up to 94.99% when the ratio of training data was 70%. Besides, we combined *TRAP* with the latest embedding method called *DANE*, which aggregates additional

[10] https://scikit-learn.org/stable

Table 6: *Micro-f1* and *macro-f1* of all algorithms on two datasets (the best results are marked in bold).

Ratio of training data			10%	30%	50%	70%	90%
Cora	micro-f1	*DANE*	0.776	0.822	0.835	0.843	0.860
		$TRAP_{DANE}$	**0.793**	**0.831**	**0.838**	**0.848**	**0.867**
	macro-f1	*DANE*	0.754	0.806	0.822	0.827	0.855
		$TRAP_{DANE}$	**0.774**	**0.818**	**0.826**	**0.833**	**0.862**
Cite	micro-f1	*DANE*	0.616	0.685	0.716	0.728	**0.731**
		$TRAP_{DANE}$	**0.630**	**0.712**	**0.722**	**0.731**	**0.731**
	macro-f1	*DANE*	0.566	0.637	0.664	0.660	0.665
		$TRAP_{DANE}$	**0.587**	**0.666**	**0.681**	**0.676**	**0.688**

Table 7: $Precision_{GR}@k$ of six graph embedding algorithms on three datasets (the best results are marked in bold).

$Precision_{GR}@$		10	100	500	1,000	10,000	100,000
Cora	*DeepWalk*	0.6	0.54	0.518	0.504	0.256	0.055
	LINE	**1**	0.760	0.328	0.219	0.052	0.013
	Node2Vec	0.7	0.5	0.45	0.31	0.066	0.026
	Struc2Vec	0.6	0.36	0.182	0.123	0.047	0.012
	SDNE	**1**	0.73	0.638	0.584	0.305	0.069
	$TRAP_{SDNE}$	**1**	**1**	**0.858**	**0.814**	**0.453**	**0.099**
Cite	*DeepWalk*	**1**	0.7	0.19	0.154	0.131	0.044
	LINE	**1**	0.68	0.498	0.349	0.068	0.014
	Node2Vec	0.8	0.76	0.438	0.248	0.047	0.020
	Struc2Vec	0.6	0.29	0.14	0.109	0.037	0.012
	SDNE	0.8	0.82	0.584	0.497	0.227	0.045
	$TRAP_{SDNE}$	**1**	**0.9**	**0.736**	**0.672**	**0.318**	**0.074**
Blog	*DeepWalk*	**1**	**1**	0.92	0.88	0.486	0.152
	LINE	**1**	**1**	**1**	**0.998**	0.669	0.125
	Node2Vec	**1**	0.98	0.984	0.910	0.435	0.145
	Struc2Vec	0.7	0.5	0.272	0.217	0.126	0.098
	SDNE	**1**	0.92	0.868	0.830	0.689	0.477
	$TRAP_{SDNE}$	**1**	**1**	0.98	0.948	**0.826**	**0.545**

attribute information to further improve the embedding performance. Table 6 summarizes the *micro-f1* and *macro-f1* scores of $TRAP_{DANE}$ and *DANE* on two datasets. Similarly, $TRAP_{DANE}$ outperformed *DANE* by alleviating the polarization problem in all cases.

- **Graph Reconstruction**: Table 7 shows $precision_{GR}@k$ of all algorithms on three datasets with varying k. In all datasets, $TRAP_{SDNE}$ generally achieved the best performance. In particular, as k increased, its relative improvement compared with *SDNE* gradually increased by up to 43.48% in Cora, 64.44% in Citeseer, and 14.3% in BlogCatalog. That is, our two regularizers allow us to capture more general structural information from the original graph by handling the polarization problem.

6 CONCLUSION

In this paper, we proposed *TRAP*, a novel meta-approach to address the *polarization* problem that exists in many real-world scenarios. We showed that the sparsity of data objects severely affected the embeddings and, therefore, suggested the two-level regularizers: *(i)* the *macroscopic regularizer* restricts the overall effect of data sparsity, and *(ii)* the *microscopic regularizer* finely tunes objects to become closer to correlated objects in the latent embedding space. *TRAP* can be easily combined with most autoencoder-based approaches by adding the regularizer into the loss function and changing the AE architecture. We validated the effectiveness of our approach on two independent tasks using six datasets, and TRAP always significantly improved the performance when applied to existing embedding methods. Overall, we believe that our work successfully tackled the *polarization* problem to greatly enhance the learning capability of embedding methods.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. 2017R1E1A1A01075927).

REFERENCES

[1] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.
[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. AAAI, 1145–1152.
[3] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law Distributions in Empirical Data. *SIAM Rev.* 51, 4 (2009), 661–703.
[4] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, Vol. 18. IJCAI, 3364–3370.
[5] Hongchang Gao, Jian Pei, and Heng Huang. 2019. ProGAN: Network Embedding via Proximity Generative Adversarial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1308–1316.
[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. NeurIPS Foundation, 2672–2680.
[7] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
[8] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *ArXiv preprint arXiv:1709.05584* (2017).
[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*. IW3C2, 173–182.
[10] Diederik P Kingma and Max Welling. 2013. Auto-encoding Variational Bayes. *ArXiv preprint arXiv:1312.6114* (2013).
[11] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.
[12] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 8 (2009), 30–37.
[13] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the Web Conference 2018*. IW3C2, 689–698.
[14] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. NeurIPS Foundation, 3111–3119.
[16] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NeurIPS Foundation, 1257–1264.
[17] Katarzyna Musiał and Przemysław Kazienko. 2013. Social Networks on the Internet. *World Wide Web* 16, 1 (2013), 31–72.
[18] Vera Pawlowsky-Glahn, Juan José Egozcue, and Raimon Tolosana Delgado. 2015. *Modeling and Analysis of Compositional Data* (1 ed.). Wiley.
[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 701–710.
[20] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. Struc2vec: Learning Node Representations from Structural Identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 385–394.

[21] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web*. IW3C2, 111–112.

[22] Hwanjun Song, Jae-Gil Lee, and Wook-Shin Han. 2017. PAMAE: Parallel k-medoids Clustering with High Accuracy and Efficiency. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1087–1096.

[23] Petre Stoica and Niclas Sandgren. 2006. Total-variance Reduction via Thresholding: Application to Cepstral Analysis. *IEEE Transactions on Signal Processing* 55, 1 (2006), 66–72.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*. IW3C2, 1067–1077.

[25] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*. ICML, 1096–1103.

[26] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1225–1234.

[27] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-encoders for Top-n Recommender Systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[28] Jia-Dong Zhang and Chi-Yin Chow. 2015. Geosoca: Exploiting Geographical, Social and Categorical Correlations for Point-of-interest Recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 443–452.

[29] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 52, 1 (2019), 5.

[30] Ziwei Zhu, Jianling Wang, and James Caverlee. 2019. Improving Top-K Recommendation via Joint Collaborative Autoencoders. In *Proceedings of the Web Conference 2019*. IW3C2, 3483–3489.