mathematics

MDPI

Article

Attributed Graph Embedding with Random Walk Regularization and Centrality-Based Attention

Yuxuan Yang, Beibei Han, Zanxi Ran, Min Gao and Yingmei Wei *

School of System Engineering, National University of Defense Technology, Changsha 410073, China; yangyuxuan20@nudt.edu.cn (Y.Y.)
* Correspondence: weiyingmei@nudt.edu.cn

Abstract: Graph-embedding learning is the foundation of complex information network analysis, aiming to represent nodes in a graph network as low-dimensional dense real-valued vectors for the application in practical analysis tasks. In recent years, the study of graph network representation learning has received increasing attention from researchers, and, among them, graph neural networks (GNNs) based on deep learning are playing an increasingly important role in this field. However, the fact that higher-order neighborhood information cannot be used effectively is a problem of most existing graph neural networks. Moreover, it tends to ignore the influence of latent representation and structural properties on graph embedding. In hopes of solving these issues, we introduce centrality encoding to learn the node properties, add an attention mechanism consideration to better distinguish the significance of neighboring nodes, and introduce random walk regularization to make sample neighbors that consistently satisfy predetermined criteria. This allows us to learn a representation of a potential node. We tested the performance of our model on node-clustering and link prediction tasks using three widely recognized benchmark datasets. The outcomes of our experiments demonstrate that our model significantly surpasses the baseline method in both tasks, indicating that the graph embedding it generates is highly expressive.

Keywords: attributed graph embedding; attributed network; graph representation learning; graph neural networks

MSC: 05C75; 05C62; 68-04

Citation: Yang, Y.; Han, B.; Ran, Z.; Gao, M.; Wei, Y. Attributed Graph Embedding with Random Walk Regularization and Centrality-Based Attention. *Mathematics* **2023**, *11*, 1830. https://doi.org/10.3390/math11081830

Academic Editors: Dawei Cheng, Zhibin Niu and Yiyi Zhang

Received: 23 March 2023
Revised: 9 April 2023
Accepted: 11 April 2023
Published: 12 April 2023

1. Introduction

As information technology has advanced, graph networks have become commonplace in daily life. Since graph networks are frequently used to describe connections among items, they can be further mined and analyzed, allowing for a deeper understanding of the networks through data mining and network analysis. Graph data exist widely in various scenarios, and they often feature a large scale, complex structure, and multiple information [1]. The diverse entities and inter-entity associations in these data constitute a series of different information networks [2–4]. For example, in social-networking platforms, social networks are formed by friends or followers among users; citation networks are formed between papers in academic websites; and the World Wide Web is formed by Web pages. In addition, the network model has also contributed to the epidemiological study of the global pandemic of COVID-19 [5]. Common network analysis tasks include social recommendation, anomaly detection, node classification, node clustering, and community discovery [6].

In the era of fast-paced information development, the scale of real-life networks is often very large, with numerous nodes exhibiting complex attributes. Traditional network analysis algorithms are therefore inadequate for deployment and application in such colossal networks. Consequently, efficiently mining crucial knowledge from these information

networks has emerged as a recent research hotspot and critical research direction within the artificial intelligence and data-mining domains, delivering significant societal worth. The majority of deep-learning-based techniques are currently being utilized to learn potential graph representations by fusing node attribute and graph topology data. For example, the GNN-based model [4], which has excelled in graph embedding, is able to fuse topological and feature information better. Gated graph sequence neural networks (GGNN) [7] optimizes the previously proposed graph neural network, and the researchers introduce the gated cycle GRU for neural network coding. Since GNNs are by nature vulnerable to hostile attacks, or small intentional perturbations on inputs, there are also many studies that introduce adversarial attack methods into graph data learning [8], using adversarial training based on generators and discriminators to improve the model capabilities. The graph autoencoder [9] is widely used in unsupervised network representation learning. The basic idea is to learn low-dimensional network node representation by taking the adjacency matrix or its variant as the original features of the nodes and using the autoencoder to achieve dimension reduction. Both the encoder and decoder in the model are multi-layer perceptron structures with multiple hidden layers; that is, they try to compress the graph structure information into low-dimensional vectors and then reconstruct its original structural features. Deepening the number of layers in a deep-learning network can allow us to learn multi-order neighbor information, but it often leads to over-smoothing [10], making features less distinguishable between nodes, which results in the opposite effect.

To more effectively address the issues listed above, we introduce centrality encoding in the node attributes and add its consideration to the attention mechanism to better distinguish the importance of neighboring nodes, while adding random walk regularization makes it possible to sample neighbors that satisfy specific conditions each time in order to learn a potential representation of the node. These improvements help to address these problems more effectively. This will enable us to better capture neighborhood information in the attribute network and learn a stronger graph-embedding representation. We employ the features acquired by the model for node-clustering and link prediction tasks in order to further demonstrate the efficacy of our proposed methodology. Experimental results demonstrate that our technique performs better than other baseline methods, which supports both our hypothesis and the validity of the model. The following three main points sum up our contribution in this work:

- We consider an attention-based convolutional layer with centrality encoding. In order to effectively aggregate and identify the significance and influence of various neighbor nodes, we apply a novel attention technique to integrate multi-hop neighborhood information;
- We propose a novel attributed graph-embedding approach called RCAGE. An attention mechanism based on centrality encoding is employed for node attributes and graph structure information, while random walk regularization is introduced to learn the latent representation;
- With various datasets, we perform node-clustering and link prediction tasks while utilizing the characteristics discovered by RCAGE. The experimental results indicate that the model achieves good performance in the corresponding tasks, proving its effectiveness and plausibility.

2. Related Works

The ubiquitous attribute graph network data are usually nonlinear, sparse, dynamic, and heterogeneous, which brings many challenges to the problem of attribute-network-related analysis. The aim of network representation learning is to obtain a reduced-dimensional vector representation, which enables nodes with analogous structures in the network to acquire comparable representations. Due to the impact of deep-learning techniques on the excellent capability of low-dimensional representation learning from data, the representation learning of attribute networks has recently attracted fresh attention in the field of study.

Early network-representation-learning techniques concentrate on dimensionality reduction by calculating the eigenvectors of the network connection matrix, such as the adjacency matrix and Laplacian matrix [11]. The typical techniques for spectral clustering are Laplacian eigenmaps (LE) [12], locally linear embedding (LLE) [13], etc. It is challenging to apply such techniques to bigger networks because the feature vector's computational cost is nonlinear compared to the eigenvectors of the matrix. Then, Perozzi et al. [14] proposed the DeepWalk algorithm, which analogizes the sequence of nodes obtained by random wandering to sentences in natural language processing, and then performs the representation learning of nodes in the network by applying the SkipGram [15–17] model. Subsequent researchers discovered that employing various random walk algorithms may result in various node representations, which also led to the creation of classical models for learning graph structure information, such as Node2vec [18] and Struc2vec [19]. LINE [20] compensates for the sparse first-order proximity problem by defining the first-order and second-order proximity among the nodes and modeling the probability separately. DNGR [6] learns the low-dimensional vector representation of nodes with stacked denoising self-encoders. SDNE [21] uses a deep self-encoder to model the similarity between nodes. The approaches described above only take into account the network's structural information, while actual networks typically also contain a significant quantity of attribute information. To more effectively maintain the information in the network, which is a hot topic for future study, both the attribute information and the structure information of nodes are required to be learned.

In recent years, with the development of deep learning, the emergence of graph neural networks especially has efficiently solved the above problems. Initially, some researchers [22,23] applied the CNN to analyze graph structure data, then employed the Fourier transform to decompose the graph's Laplacian matrix before using graph convolution to extract features. Subsequently, Kipf et al. [4] simplified the prior approach by proposing the graph convolutional network (GCN) algorithm. The graph attention network (GAT) [24] extends the GCN with the introduction of the attention mechanism [25,26] and it utilizes a masked self-attentive layer to assign different weights to different nodes based on the features of their neighborhoods. SANE [27] uses the attention mechanism and CBOW [16] model to weight the interaction strength between nodes while capturing the similarity of the network topology and attribute information. DANE [28] employs two deep models to capture and maintain a high level of nonlinearity as well as numerous similarities in the topology and node attributes. There is also the more common graph adversarial learning, whose various methods are still based on generators and discriminators that improve the ability of the model by means of adversarial training. The discriminator in GraphGAN [29] makes the node pairs in the original network graph more similar, and lets the node pairs generated by the generator have less similarity. ANE [30] applies generative adversarial networks as an additional regularization to existing network-representation-learning methods by treating prior distributions as real data and node vectors as generative samples. GraphSGAN [31] designs a new optimization objective with multiple complex loss terms by means of semi-supervised learning to ensure that samples are generated in the density difference when the generator is at equilibrium. NetGAN [8] instead treats graph generation as learning a distribution with biased random wandering and proposes a generative adversarial network framework for generating and distinguishing random wandering using LSTM.

The autoencoder [32] is an unsupervised neural network model with two stages, decoding and encoding, and it typically employs a deep neural network. The graph autoencoder (GAE) [9] invokes the idea of the autoencoder, using the GCN in the encoding phase and using the form of inner product in the decoding part, which is suitable for unsupervised learning. The graph auto-encoder aims to learn a condensed graph representation by minimizing the difference between the reconstructed adjacency matrix and the original matrix, which serves as the loss function to train the model and learn node features. The graph variational autoencoder (VGAE) [9] introduces a Gaussian distribution to constrain the distribution of low-dimensional vectors based on the GAE, and by sam-

pling in the low-dimensional vector distribution, it can obtain approximately real samples. DNENC [33] employed a neighbor-aware GAE and an end-to-end learning approach to gather neighbor information. Building on these models, numerous following models are developed for encoders by incorporating regularization, higher-order neighbor information, and so on. ARGA [34], DAEGC [35], AGC [36], GEC-CSD [37], and other common approaches are listed below. In addition, several optimization methods such as the reconfiguration loss optimizer [38] and the modularity optimizer [39] are given. Motivated by these methodologies, we propose our approach in this paper.

3. Methodology

In this section, we focus on the model framework designed for attributed graph embedding. The overall architecture of RCAGE is shown in Figure 1.

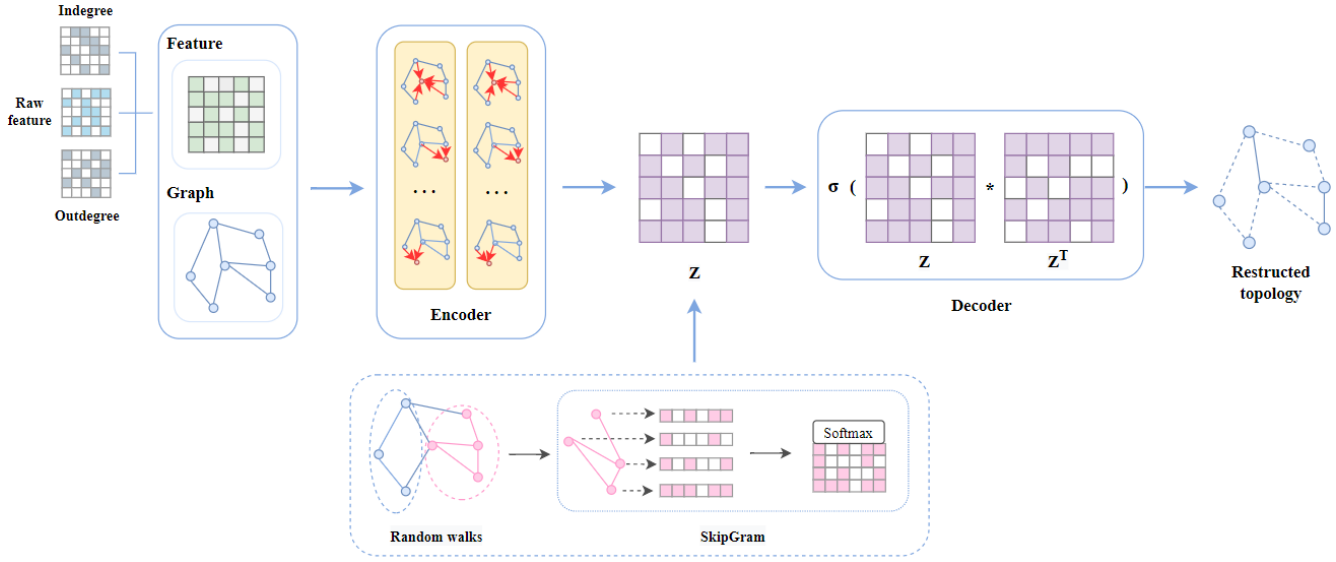


Figure 1. The RCAGE architecture. Our model takes in the graph structure and node attributes as inputs on the left side. The following component is an encoder that utilizes an attention mechanism to produce the embedding matrix Z, which is subject to random walk regularization. The last is the decoder and the loss calculation method.

3.1. Problem Description

We define an attributed graph as $G = (V, E, X)$. V represents the nodes in the graph G, which can be expressed as $V = \{v_1, v_2, \cdots, v_n\}$($n$ is the number of nodes). $E = \{e_{ij}\}$ is a set of edges, where e_{ij} denotes the edge between node i and node j. $X = \{x_1, x_2, \cdots, x_n\}$ is the features of the nodes in the graph G, where $x_i \in \mathrm{R}^m$ represents the feature of node i. We use an adjacency matrix $A \in \mathrm{R}^{n \times n}$ to represent the edges in order to better express the graph topology, where $A_{ij} = 1$ if $e_{ij} \in E$; otherwise, $A_{ij} = 0$.

We want to obtain a d-dimensional vector for each node $\mathrm{v_i}$ in the attributed graph G by training it with a function F. This process can be expressed as $\mathrm{F(A, X)} \rightarrow \mathrm{Z}$, where $\mathrm{Z} \in \mathrm{R}^{\mathrm{d} \times \mathrm{n}}$(d << n) is the final learned embedding matrix. We want Z to retain as comprehensive the information as possible about node attributes and graph topology in order to have better performance in downstream tasks.

In this paper, we pick node clustering and link prediction as the graph downstream tasks. The purpose of the node-clustering task is to partition all nodes into different classes so that the similarity of node features within the same class is as large as possible. The link prediction task determines whether a link exists between two nodes based on their characteristics.

3.2. Graph Autoencoder

3.2.1. Centrality Encoding

Different nodes in a network may have varying degrees of significance. The self-attention module, which primarily uses node semantic properties to determine similarities, however, does not take into account this information. Node centrality, which gauges a node's importance in the network, is often a powerful indicator of graph comprehension [40]. For example, celebrities with enormous followings are a key component in anticipating social-networking trends. Such content should be a useful signal for graph learning, but it is ignored in the present attention computation. As an extra signal to the neural network, we employ the degree centrality, one of the accepted centrality metrics in the paper. We include it in the input node attributes when we apply the centrality encoding to each node.

$$z_i^{(0)} = x_i + z^+_{deg^+(v_i)} + z^-_{deg^-(v_i)} \tag{1}$$

In Equation (1), learnable embedding vectors z^+, z^- are determined by the outdegree $deg^+(v_i)$ and indegree $deg^-(v_i)$. For undirected graphs, the above two can be unified as $deg(v_i)$. In this way, the model can better capture the node importance during training with the attention mechanism.

3.2.2. Graph Attentional Encoder

In this paper, we design a variation of the graph attention network as an encoder to capture both node attributes and graph structure in a consolidated framework. Depending on the node's degree in a real graph network, the neighbors' level of contribution to the central node will vary. By introducing an attention mechanism, we may increase the weights of neighbor nodes that are more pertinent to the central node when learning the node representation in order to gauge the significance of various neighbors. The expression is as follows:

$$z_i^{l+1} = \sigma(\sum_{j \in N_i} \alpha_{ij} W z_j^l) \tag{2}$$

In the Equation (2), z_i^{l+1} depicts the output representation for node i, with N_i representing its neighbor set. The attention coefficient α_{ij} is used to measure the significance of adjacent node j to node i. σ is a nonlinear function.

For the node attributes, a single-layer fully connected network is used to calculate the similarity coefficients, and the weight vector is denoted as α_1.

$$s_{ij} = \alpha_1 \left[Wh_i || Wh_j \right] \tag{3}$$

In terms of graph topology, the impact of various-order neighbor nodes must be considered. We cannot take into consideration merely 1-hop neighbor information as in the GAT model, due to the complexity of the graph structure relationship. Here, by setting a parameter, we give consideration to multi-order neighbor information.

$$\beta^+ = \left(\lambda_1 + \lambda_1^2 + \cdots + \lambda_1^k \right) / k \tag{4}$$

$$\beta^- = \left(\lambda_2 + \lambda_2^2 + \cdots + \lambda_2^k \right) / k \tag{5}$$

where $\lambda_{1ij} = \frac{1}{d^+}$, $\lambda_{2ij} = \frac{1}{d^-}$ if $e_{ij} \in E$, and $\lambda_{1ij} = \lambda_{2ij} = 0$ otherwise. d^+ and d^- mean the outdegree and indegree of node i, respectively. β stands for the topological correlation between nodes j and i up to k-hops. k is a parameter that can be set to a value of your choice for different datasets. For undirected graphs, β^+ and β^- may not be distinguished and are uniformly defined as β.

To make the attention coefficients easily comparable across nodes, they are normalized in the set N_i with the softmax function. The formula is as follows, where $j \in N_i$:

$$\alpha_{ij} = softmax_j(s_{ij}) = \frac{exp(s_{ij})}{\sum_{m \in N_i} exp(s_{im})} \tag{6}$$

The following equation can be used to express the attention coefficient once the activation function LeakyReLU and centrality parameter have been introduced to this base.

$$\alpha_{ij} = \frac{exp\Big(LeakyReLU\Big(\beta^+{}_{ij}\beta^-{}_{ij}\big(\alpha_1\big[Wh_i||Wh_j\big]\big)\Big)\Big)}{\sum_{m \in N_i} exp(LeakyReLU(\beta^+{}_{im}\beta^-{}_{im}(\alpha_1[Wh_i||Wh_m])))} \tag{7}$$

where $h_i = z_i^{(0)}$ is input to the model. It is then trained in two stacked graph attention layers to integrate node attributes and graph structure, and finally output the embedding results $z_i = z_i^{(2)}$.

$$z_i^{(1)} = \sigma(\sum_{j \in N_I} \alpha_{ij} W^{(0)} h_j) \tag{8}$$

$$z_i^{(2)} = \sigma(\sum_{j \in N_I} \alpha_{ij} W^{(1)} z_j^{(1)}) \tag{9}$$

3.3. Random Walk Regularization

In this section, we apply random walk regularization, drawing inspiration from DeepWalk and Word2vec, to improve the learning of potential node representation information. We use random walk with restarts to sample and its neighbor nodes satisfying certain conditions as a combination, and apply the SkipGram idea to learn the potential representation of nodes.

Random walk with restarts [41] algorithm is an enhancement to the random walk algorithm. Beginning with node v_i in the graph, our approach presents two possibilities at each step: selecting a neighboring node at random or going back to the origin. The parameter p governs the likelihood of resuming from the original node, whereas $1 - p$ controls the chance of shifting to an adjacent node. According to this method, we can obtain a set of context nodes W_{v_i}, which can capture the multifaceted relationship between two nodes and the overall structural information of the graph.

Analogous to the NLP tasks, we consider the sampled set W_{v_i} as a sentence, and we aim to maximize the co-occurrence probability of node v_i with other nodes in this window. This can be expressed as the following equation:

$$L_{rw} = \log p(\mu_i | Z(v_i)) \tag{10}$$

In Equation (10), $\mu_i \in W_{v_i}$ and $Z(v_i)$ denote the potential representation of the node v_i with encoder.

3.4. Decoder

Now, the graph decoder mainly includes three types: reconstructing attributes, reconstructing graph topology, or both of the above. In this paper, the embedding matrix we finally obtain already includes both node attributes and graph topology information, so we directly adopt the form of inner product decoder:

$$\hat{A} = sigmoid\Big(ZZ^T\Big) \tag{11}$$

3.5. Reconstruction Loss

We use the loss of decoder reconstructing attributes and graph topology as reconstruction loss, which is a flexible and efficient method. The specific formula can be expressed as follows:

$$L_r = E_{q(Z|X,A)}\left[\log_p(A|Z)\right] \tag{12}$$

4. Experiments

4.1. Datasets

We performed node-clustering and link prediction tasks on the Cora, Citeseer, and Pubmed datasets, which are three commonly used citation network datasets. The link indicates the citation relationship of the paper and the attribute is the word band model representation of the corresponding paper [42]. Table 1 displays the specifics of the three datasets.

- Cora (https://paperswithcode.com/dataset/cora, accessed on 2 March 2023) consists of 2708 papers in the field of machine learning. Case based, genetic algorithms, neural networks, probabilistic techniques, reinforcement learning, rule learning, and theory are the seven categories in which these studies fall [43].
- Citeseer (https://paperswithcode.com/dataset/citeseer, accessed on 2 March 2023) comprises 3312 scientific and technical papers in the Citeseer network database, divided into six areas: Agents, AI (Artificial Intelligence), DB (Database), IR (Information Retrieval), ML (Machine Language), and HCI (Human Computer Interaction).
- Pubmed (https://paperswithcode.com/dataset/pubmed, accessed on 2 March 2023) consists of 19,717 scientific publications on diabetes from the Pubmed database, grouped into three categories. The three categories are "Diabetes Mellitus Experimental", "Diabetes Mellitus Type 1", and "Diabetes Mellitus Type 2".

Table 1. The details of datasets.

Datasets	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
Citeseer	3327	4723	3703	6
Pubmed	19,717	44,338	500	3

4.2. Baseline Methods

We compare the method proposed in this paper with 11 other benchmark models. These approaches are classified into three main groups:

(1) Methods depending on features only (F):
- K-means [44] automatically builds clusters according to node feature characteristics;
- Spectral-F [45] computes the cosine similarity between node attributes as an input.

(2) Methods depending on graph structure only (G):
- Spectral-G [45] is spectral clustering that uses the adjacency matrix as the similarity matrix for calculation;
- DNGR [6] generates a low-dimensional vector representation of each node by capturing the graph structure information and employing a stacked denoising;
- DeepWalk [14] learns the vector representation of nodes by exploiting node-to-node co-occurrence relationships in the graph.

(3) Methods making use of both features and graph structure (F&G):
- GAE [9] is an unsupervised learning framework using an auto-encoder based on node attributes and graph structure;
- VGAE [9] replaces the autoencoder with a variational graph autoencoder;
- ARGA & ARVGA [27] are adversarially regularized on the basis of GAE and VGAE;
- AGC [36] obtains node features by utilizing higher-order graph convolution;

- DAEGC [35] introduces a graph attention network to aggregate the features of different hops neighbor nodes, and then it combines the loss of both the graph reconstruction and clustering for model optimization.

4.3. Evaluation Metrics

For the node-clustering task, we use four metrics to measure the results. They are Accuracy (ACC), Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and macro F1-score (F1). For the link prediction task, we choose Area Under Curve (AUC) and Average Precision (AP) to judge the performance.

4.4. Parameter Settings

We set the number of walks to 45, the window size to {25, 20}, and the walk length to {25, 20} for the hyper-parameters associated with the random walk regularization network. Our experiments reveal that, depending on the dataset, the best-performing model employs 45 walks with a window size and walk length of either 25 or 20. The initial learning rate for the random walk regularization is 0.001. For the attention part, we consider two-hops neighbor nodes on Cora and Citeseer and set $k = 2$. On Pubmed, we set $k = 3$. The number of units in the hidden layer is set to 256 for all datasets. We use a 16-neuron embedding layer for Cora and Citeseer and a 32-neuron embedding layer for Pubmed.

4.5. Experimental Results

4.5.1. Node Clustering

The clustering results of the three datasets are summarized in Table 2, Table 3, Table 4, respectively. The greatest outcomes are shown by bold numbers, while the second-best results are represented by underlined numbers. Overall, the results of our proposal method generally yield the best in the three datasets in this paper. In comparison to the best baseline, the clustering accuracy of our approach on Cora, Citeseer, and Pubmed increased by 0.4%, 2.5% and 3.2%, respectively. Among the three datasets, the NMI rose by 2%, 2.1% and 6.2%, and the ARI improved by 4.4%, 3.0% and 5.7%, accordingly. The F1 improved on Citeseer and Pubmed by 0.2% and 3.2%, respectively, but still scored 2.4% below the top baseline results on Cora.

By integrating the properties of the various datasets and comparing the experimental results on various datasets, we discover that the proposed approach performs better on Pubmed. We speculate that the reason for this may be that datasets with a larger data density are more sensitive to the node indegree and outdegree. This makes them more sensitive to attention parameters, so we finally obtain a correspondingly higher ultimate effect improvement.

Table 2. Node-clustering performance on Cora.

Methods	Input	ACC	NMI	ARI	F1
K-means	F	0.347	0.167	0.239	0.254
Spectral-F	F	0.363	0.151	0.071	0.256
Spectral-G	G	0.342	0.195	0.045	0.302
DNGR	G	0.492	0.373	0.142	0.373
DeepWalk	G	0.467	0.318	0.291	0.381
GAE	F&G	0.533	0.407	0.302	0.420
VGAE	F&G	0.560	0.385	0.347	0.415
ARGA	F&G	0.640	0.449	0.352	0.619
ARVGA	F&G	0.638	0.450	0.374	0.627
AGC	F&G	0.689	<u>0.537</u>	0.486	0.656
DAEGC	F&G	<u>0.704</u>	0.528	<u>0.496</u>	**0.682**
RCAGE	F&G	**0.708**	**0.557**	**0.540**	<u>0.658</u>

Table 3. Node-clustering performance on Citeseer.

Methods	Input	ACC	NMI	ARI	F1
K-means	F	0.385	0.170	0.285	0.305
Spectral-F	F	0.462	0.212	0.183	0.337
Spectral-G	G	0.259	0.118	0.013	0.295
DNGR	G	0.326	0.180	0.043	0.442
DeepWalk	G	0.362	0.097	0.137	0.267
GAE	F&G	0.413	0.183	0.191	0.291
VGAE	F&G	0.444	0.227	0.206	0.319
ARGA	F&G	0.573	0.350	0.341	0.546
ARVGA	F&G	0.544	0.261	0.245	0.529
AGC	F&G	0.670	0.411	0.419	0.625
DAEGC	F&G	0.672	0.397	0.410	0.636
RCAGE	F&G	**0.697**	**0.432**	**0.449**	**0.638**

Table 4. Node-clustering performance on Pubmed.

Methods	Input	ACC	NMI	ARI	F1
K-means	F	0.573	0.291	0.246	0.574
Spectral-F	F	0.599	0.326	0.098	0.586
Spectral-G	G	0.397	0.035	0.057	0.520
DNGR	G	0.454	0.154	0.059	0.179
DeepWalk	G	0.619	0.167	0.255	0.471
GAE	F&G	0.641	0.230	0.246	0.493
VGAE	F&G	0.655	0.251	0.201	0.510
ARGA	F&G	0.591	0.232	0.217	0.584
ARVGA	F&G	0.582	0.206	0.183	0.230
AGC	F&G	0.698	0.316	0.282	0.687
DAEGC	F&G	0.671	0.266	0.278	0.659
RCAGE	F&G	**0.730**	**0.328**	**0.335**	**0.719**

4.5.2. Link Prediction

The link prediction results of the three datasets are summarized in Table 5. The greatest outcomes are represented by bold numbers, while the second-best results are represented by italicized numbers. The results of the experiments indicate that our technique also performs well on the link prediction job. According to the best baseline results, AUC improved by 3.9% and 4.8% on Cora and Citeseer, respectively. AP increased by 3.0% and 4.0% on Cora and Citeseer, respectively, in comparison to the best baseline results.

Table 5. Link prediction results.

Methods	Input	Cora		Citeseer	
		AUC	AP	AUC	AP
Spectral-G	G	0.844	0.886	0.803	0.849
DeepWalk	G	0.833	0.851	0.804	0.835
GAE	F&G	0.913	0.921	0.893	0.898
VGAE	F&G	0.915	0.927	0.908	0.920
ARGA	F&G	0.923	0.930	0.917	0.931
ARVGA	F&G	0.924	0.926	0.924	0.930
RCAGE	F&G	**0.963**	**0.962**	**0.972**	**0.973**

4.5.3. Ablation Study

To further clarify how each module in the overall model functions, we ran ablation experiments. Here are the specifics of the three models that were created by eliminating the random walk with restarts, centrality encoding, and attention mechanism sections, respectively. As shown in Figure 2, the outcomes of the node clustering using these three models are compared to our method in this paper.

- RCAGE/rwr: Our proposed method does not use the random walk with restarts regularization strategy to learn node embedding;
- RCAGE/att: Our proposed method does not utilize the attention mechanism;
- RCAGE/ce: The model input does not contain node centrality encoding, only the raw feature information.

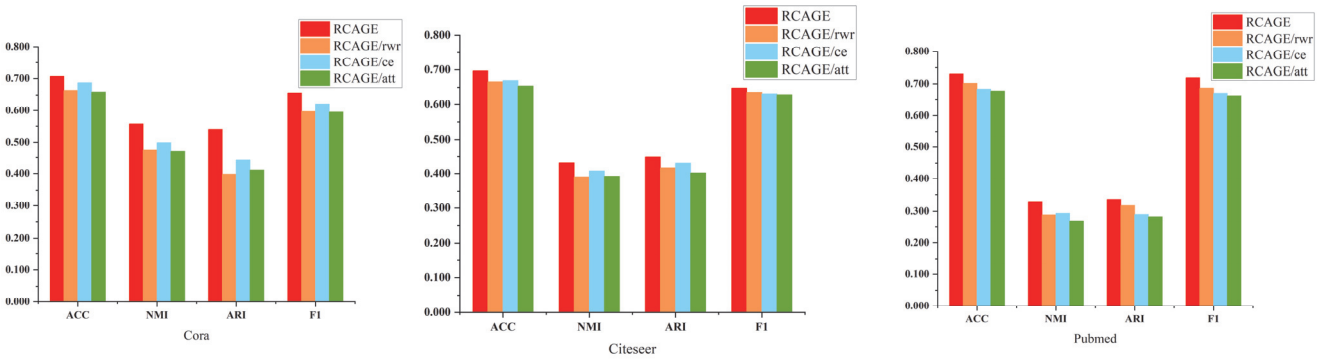


Figure 2. Comparison of different variants of the model with our method.

From Figure 2, we can obviously see that each module contributes significantly to the overall model. Moreover, we can draw some conclusions from the figure. Of these three modules, the attention mechanism applied to this work has the most impact on the experiment results, indicating that it is important for learning the feature of graph topology. For Cora and Citeseer, random walk regularization has a stronger impact than the centrality encoding module, but the converse is true for Pubmed. Combined with the previous experimental results, we assume that the centrality encoding and the attention mechanism with the effect of the indegree and outdegree discussed in this research can better acquire complicated and larger graph structure features. This aids us in obtaining superior node-embedding representations.

4.5.4. Variant Analyses

Analysis for the variants of random walk with restarts: There are three main variables involved in this module; they are the number of walks, walk length, and window size. Number of walks refers to the total number of random node-to-node journeys. The duration of the random walk that began at each node is known as the walk length. Window size indicates the co-occurrence window size of the SkipGram model when sampling the neighbors of each node. Figure 3 shows that the results of node clustering, as an example, are relatively better when the walk length and window size are set to values of 25 or 20. At the same time, when the number of walks is set to 45, the experimental results are noticeably better than the cases with other values.

Analysis for k: To allow the model to explore the effect of the node neighborhood on its feature learning, we control the amount of the order for which neighborhood information is utilized by setting different values of k. We set the neighbor order to {1, 2, 3, 4}, and Table 6 displays the experimental results on the three datasets with different neighbor orders. Table 6 indicates that the model's experimental results on three datasets gradually increases as the number of neighbor orders increases from 1 to 2, and the highest clustering accuracy is achieved when $k = 2$. The model may capture more interference information from less important neighbors when the neighbor order is raised above the optimum, failing to acquire a more discriminative node representation as a result.

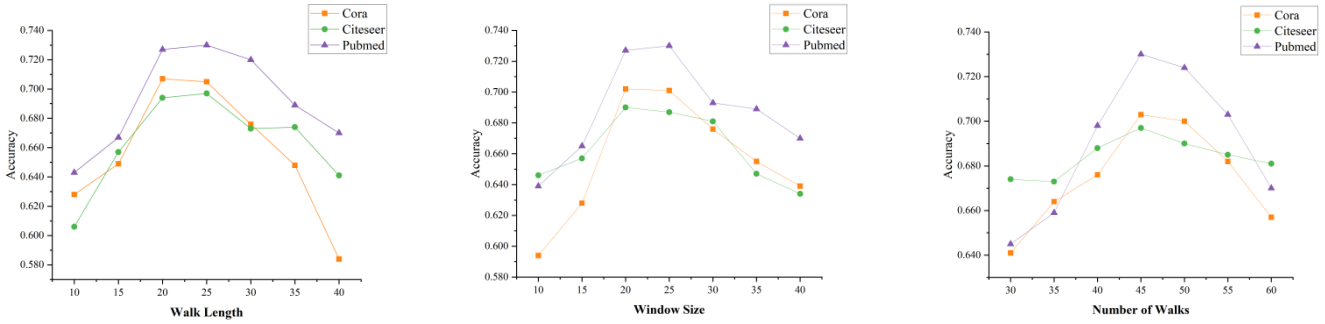


Figure 3. Analysis for walk length, window size, and number of walks.

Table 6. Differences in the accuracy of node clustering with different neighbor orders.

k	Cora				Citeseer				Pubmed			
	ACC	**NMI**	**ARI**	**F1**	**ACC**	**NMI**	**ARI**	**F1**	**ACC**	**NMI**	**ARI**	**F1**
$k = 1$	0.673	0.476	0.398	0.597	0.665	0.434	0.435	0.629	0.694	0.299	0.328	**0.724**
$k = 2$	**0.707**	**0.557**	**0.540**	**0.658**	**0.697**	**0.432**	**0.449**	0.638	**0.730**	**0.328**	**0.335**	0.719
$k = 3$	0.687	0.487	0.451	0.606	0.671	0.423	0.442	**0.641**	0.712	0.315	0.320	0.701
$k = 4$	0.684	0.479	0.422	0.652	0.658	0.407	0.421	0.633	0.678	0.270	0.283	0.671

Analysis for embedding size: Taking node clustering as an example, we set the embedding size to {4, 16, 32, 64, 256} to investigate its effect on the experiment. The details are summarized in Figure 4. As can be observed, the model performs best on Cora and Citeseer when the value of the embedding size is taken as 16, while it achieves the best performance on Pubmed when the value of the embedding size is taken as 32. For graph networks with a different feature richness, it is necessary to experiment repeatedly with the value of the embedding dimension to ensure that as much feature information as possible is preserved.

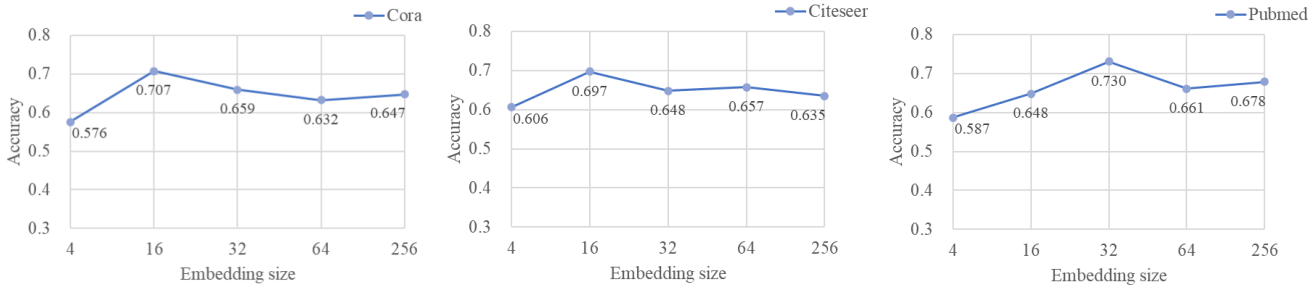
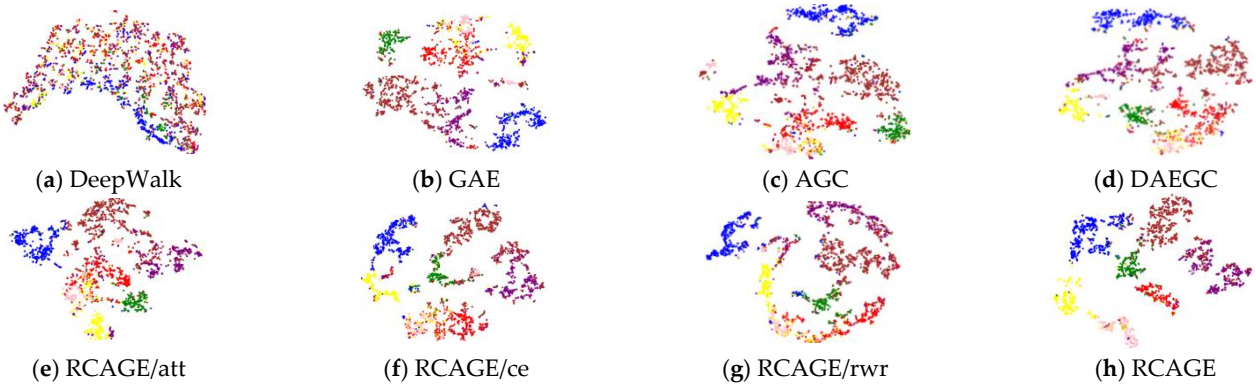


Figure 4. Analysis for embedding size.

4.5.5. Visualization

In order to demonstrate the efficacy of the proposed approach more clearly, we visualize the node-clustering results of several models for comparison. Since the trends of the visualization results are similar on different datasets, we use the Cora and Citeseer datasets here as examples. The output embedding at the last layer, in the previous Softmax operation, was applied to the node-clustering task and the generated node embeddings were plotted using t-SNE [46]. The results are reported in Figure 5 and colored with real labels.



(**a**) DeepWalk (**b**) GAE (**c**) AGC (**d**) DAEGC

(**e**) RCAGE/att (**f**) RCAGE/ce (**g**) RCAGE/rwr (**h**) RCAGE

Figure 5. Visualization of node embeddings learned by different models on Cora.

In this section, we have selected a few typical models to visualize the experimental results. As seen in Figure 5, the other models are noticeably superior to DeepWalk, due to the fact that they consider both the node attributes and graph structure. This further emphasizes that node features play an essential role in the research of mining hidden graph information. The results of the three variant models are displayed in Figure 2 and are also visualized here separately. Comparatively speaking, our method has somewhat more distinct borders between each class, and it particularly clusters well on the pink group. This also illustrates the importance of the joint role of random walk regularization, centrality encoding, and the attention mechanism in the model.

5. Conclusions

In this paper, we propose the RCAGE model and apply it to graph representation learning. In this study, we employ centrality coding to quantify the significance of each node in the network. This information, together with the raw features and graph structure, are given into the model. To better combine node characteristics and graph topology information, we adopt an attention mechanism that takes into account the effect of the node degree. We also use random walk with restarts to sample node neighbors and use it as a regularization to learn potential representations of nodes. The final experimental results show that our model performs well for unsupervised learning tasks such as node clustering and link prediction.

In the future, we plan to investigate expanding the framework to more sophisticated and time-varying graphs, as well as further learning of edge attributes and global location information, in order to allow more accurate graph-embedding representations.

Author Contributions: Conceptualization, Y.Y. and Y.W.; methodology, Y.Y.; software, B.H.; validation, Z.R. and M.G.; formal analysis, Y.Y.; investigation, Z.R. and M.G.; resources, Y.Y.; data curation, B.H.; writing—original draft preparation, Y.Y.; writing—review and editing, Y.Y., Z.R. and M.G.; visualization, Y.Y.; supervision, Y.W.; project administration, Y.W.; funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by both the National Natural Science Foundation of China (NSFC) and the Postgraduate Scientific Research Innovation Project of Hunan Province under number CX20200075.

Data Availability Statement: Data sharing is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fang, R.; Wen, L.; Kang, Z.; Liu, J. Structure-Preserving Graph Representation Learning. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Orlando, FL, USA, 28 November–1 December 2022.
2. Hastings, M.B. Community detection as an inference problem. *Phys. Rev. E* **2006**, *74*, 035102. [CrossRef] [PubMed]

3. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD-2018), London, UK, 19–23 August 2018; pp. 974–983.
4. Welling, M.; Kipf, T.N. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR-2017), Toulon, France, 24–26 April 2017.
5. Chang, S.; Pierson, E.; Koh, P.W. Mobility network models of COVID-19 explain inequities and inform reopening. *Nature* **2021**, *589*, 82–87. [CrossRef] [PubMed]
6. Cao, S.; Lu, W.; Xu, Q. Deep neural networks for learning graph representations. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
7. Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated Graph Sequence Neural Networks. *arXiv* **2015**, arXiv:1511.05493.
8. Bojchevski, A.; Shchur, O.; Zügner, D.; Günnemann, S. NetGAN: Generating Graphs via Random Walks. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 610–619.
9. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.
10. Henaff, M.; Bruna, J.; Lecun, Y. Deep convolutional networks on graph-structured data. *arXiv* **2015**, arXiv:1506.05163.
11. Chen, F.; Wang, Y.; Wang, B.; Kuo, C. Graph representation learning: A survey. *APSIPA Trans. Signal Inf. Process.* **2020**, *9*, e15. [CrossRef]
12. Newman, M.E. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **2006**, *74*, 036104. [CrossRef] [PubMed]
13. Roweis, S.; Saul, L. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef] [PubMed]
14. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD-2014), New York, NY, USA, 24 August 2014; pp. 701–710.
15. Mikolov, T.; Sutskever, I.; Chen, K. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
16. Mikolov, T.; Chen, K.; Corrado, G. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
17. Mikolov, T.; Karafiát, M.; Burget, L. Recurrent neural network based language model. In Proceedings of the International Speech Communication Association, Makuhari, Chiba, Japan, 26–30 September 2010; pp. 1045–1048.
18. Grover, A.; Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
19. Ribeiro, L.F.R.; Saverese, P.H.P.; Figueiredo, D.R. struc2vec: Learning node representations from structural identity. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13 August 2017; pp. 385–394.
20. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web (WWW-2015), New York, NY, USA, 18–22 May 2015; pp. 1067–1077.
21. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD-2016), San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
22. Bruna, J.; Zaremba, W.; Szlam, A.; Lecun, Y. Spectral networks and locally connected networks on graphs. In Proceedings of the 3rd International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
23. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3844–3852.
24. Velĭckovíc, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
25. Xu, K.; Ba, J.; Kiros, R. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the 32nd International Conference on Machine Learning (ICML 2015), Lille, France, 6–11 July 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 2048–2057.
26. Qin, C.; Zhu, H.; Xu, T. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR 2018), Ann Arbor, MI, USA, 8–12 July 2018; pp. 25–34.
27. Wang, H.; Chen, E.; Liu, Q. A United Approach to Learning Sparse Attributed Network Embedding. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM 2018), Singapore, 17–20 November 2018; pp. 557–566.
28. Gao, H.; Huang, H. Deep Attributed Network Embedding. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018), Stockholm, Sweden, 13–19 July 2018; pp. 3364–3370.
29. Wang, H.; Wang, J.; Jia, W.; Miao, Z.; Guo, M. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
30. Dai, Q.; Li, Q.; Tang, J.; Wang, D. Adversarial Network Embedding. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
31. Ding, M.; Tang, J.; Zhang, J. Semi-supervised Learning on Graphs with Generative Adversarial Nets. In Proceedings of the 27th ACM International Conferenceon Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 913–922.
32. Kingma, D.P.; Welling, M. Auto-encoding variational baye. *arXiv* **2013**, arXiv:1312.6114.

33. Wang, C.; Pan, S.; Celina, P.Y.; Hu, R.; Long, G.; Zhang, C. Deep neighbor-aware embedding for node clustering in attributed graphs. *Pattern Recognit.* **2022**, *122*, 108230. [CrossRef]
34. Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; Zhang, C. Adversarially regularized graph autoencoder for graph embedding. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018.
35. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed graph clustering: A deep attentional embedding approach. In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-2019), Macao, China, 10–16 August 2019.
36. Zhang, X.; Liu, H.; Li, Q.; Wu, X.M. Attributed Graph Clustering via Adaptive Graph Convolution. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
37. Xu, H.; Xia, W.; Gao, Q.; Han, J.; Gao, X. Graph embedding clustering: Graph attention auto-encoder with cluster-specificity distribution. *Neural Netw.* **2021**, *142*, 221–230. [CrossRef] [PubMed]
38. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In Proceedings of the 33rd International conference on machine learning (ICML-2016), New York, NY, USA, 19–24 June 2016; pp. 478–487.
39. Yang, L.; Cao, X.; He, D.; Wang, C.; Wang, X.; Zhang, W. Modularity based community detection with deep learning. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-2016), New York, NY, USA, 9–15 July 2016; pp. 2252–2258.
40. Marshall, P.D. The promotion and presentation of the self: Celebrity as marker of presentational media. *Celebr. Stud.* **2010**, *1*, 35–48. [CrossRef]
41. Pan, J.; Yang, H.; Faloutsos, C.; Duygulu, P. Automatic multimedia cross-modal correlation discovery. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 22 August 2004; pp. 653–658.
42. Cui, G.; Yang, C.; Liu, Z. Adaptive Graph Encoder for Attributed Graph Embedding. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20), New York, NY, USA, 6–10 July 2020; pp. 976–985.
43. Zhang, Z.; Mao, J. Jointly sparse neighborhood graph for multi-view manifold clustering. *Neurocomputing* **2016**, *216*, 28–38. [CrossRef]
44. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
45. Ng, A.; Jordan, M.; Weiss, Y. On spectral clustering: Analysis and an algorithm. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, Vancouver, BC, Canada, 3–8 December 2001; Volume 14.
46. Van Der Maaten, L. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.* **2014**, *15*, 3221–3245.