RA-GCN: Mitigating the Over-smoothing Phenomenon in Text Classification (Postprint)

Authors: Fanjun Su, Ma Mingxu, Tong Guoxiang

Date: 2022-04-07T15:01:56+00:00

Abstract

Most existing algorithms utilizing Graph Neural Networks for text classification have overlooked the over-smoothing problem inherent in GNNs and the discrepancies arising from text graph topological differences, leading to suboptimal performance in text classification. To address this issue, we propose WACD, a method to measure the smoothness of multiple text graph representations, and RWACD, a regularization term to mitigate over-smoothing. Subsequently, we introduce ARS, an attention- and residual-based network architecture to compensate for the loss of text information caused by graph topological differences. Finally, we propose RA-GCN, a Graph Convolutional Neural Network-based text classification algorithm. RA-GCN employs ARS to fuse text representations in the graph representation learning layer and uses RWACD to mitigate over-smoothing in the readout layer. Experimental results on six Chinese and English datasets demonstrate the classification performance of RA-GCN, and multiple comparative experiments validate the effectiveness of RWACD and ARS.

Full Text

Preamble

Vol. 39 No. 8
Application Research of Computers
ChinaXiv Cooperative Journal

RA-GCN: A Text Classification Algorithm for Suppressing Over-smoothing

Su Fanjun, Ma Mingxu†, Tong Guoxiang
(School of Optical-Electrical and Computer Engineering, University of Shanghai

for Science and Technology, Shanghai 200093, China)

Abstract: Most existing text classification algorithms based on graph neural networks overlook the over-smoothing problem inherent in GNNs and the errors introduced by variations in text graph topology, resulting in suboptimal classification performance. To address these issues, we propose WACD, a method for measuring the smoothness of multiple text graph representations, and RWACD, a regularization term designed to suppress over-smoothing. We then introduce ARS, an attention-based residual network structure that compensates for text information loss caused by graph topology differences. Finally, we present RA-GCN, a graph convolutional neural network text classification algorithm. RA-GCN employs ARS to fuse text representations in the graph representation learning layer and utilizes RWACD in the readout layer to suppress over-smoothing. Experiments on six Chinese and English datasets demonstrate the classification performance of RA-GCN, with multiple comparative experiments validating the effectiveness of RWACD and ARS.

Keywords: text classification; graph convolutional neural network; over-smoothing; attention mechanism

---

0 Introduction

Text classification serves as a fundamental problem in natural language processing and has been applied to numerous real-world scenarios such as spam detection, news categorization, and sentiment recognition. The performance of text classification models largely depends on the quality of text representations. Deep learning-based methods avoid manual rule and feature engineering by automatically learning semantically meaningful representations [**?**]. While CNN and RNN-based deep learning approaches effectively capture semantic and syntactic features in local contiguous sequences, they still face limitations in extracting non-contiguous words and long-distance semantic information [**?**].

In recent years, graph neural networks have mitigated these limitations. Yao [**?**] constructed a single corpus-wide text-word heterogeneous graph and employed GCN [**?**] to learn word co-occurrence information, updating text and word representations for classification. Wu [**?**] simplified GCN into SGC by removing nonlinear activation functions and collapsing weight matrices between consecutive layers, achieving promising results on single heterogeneous graph-based data. However, single heterogeneous graph methods are unsuitable for testing new texts and consume substantial memory space. Consequently, Huang [**?**] constructed individual graph data for each text while sharing global word representations and edge weights to better capture local features and reduce memory consumption. Zhang [**?**] built unique graph data for each text to improve the inductive learning capability of graph methods, using GGNN [**?**] to update word features and obtain text representations and categories. Nevertheless, these approaches ignore the over-smoothing problem in GNNs. This paper focuses on the

graph classification direction based on per-text graph representations [?, ?, ?] to mitigate over-smoothing and enhance text classification performance.

During GNN development, Li [?] first drew attention to over-smoothing [?], demonstrating that graph convolution is a special form of Laplacian smoothing and concluding that smoothing operations are the key mechanism enabling GCN to work. However, after multiple rounds of Laplacian smoothing, node features converge to similar values—a phenomenon known as over-smoothing. Over-smoothing renders nodes indistinguishable and degrades network performance. Chen [?] verified that smoothing is essential to GNNs, proposed MAD (Mean Average Distance) to measure smoothness, analyzed over-smoothing causes from a graph topology perspective, identified excessive mixing of information and noise as a key factor, and introduced the MADreg regularization term and AdaGraph iterative training algorithm. Other researchers have proposed optimizing artificially constructed graph topology through model refinement to improve performance and suppress over-smoothing. Wang [?] expanded node receptive fields through multi-hop attention mechanisms, enabling remote interactions between nodes not directly connected but within multiple hops, thereby filtering high-frequency noise information. Yang [?] utilized pointer networks [?] to identify relevant nodes in multi-order neighborhoods and employed one-dimensional convolutions to extract high-level features, filtering noise information to alleviate over-smoothing. From a network structure perspective, literature [?] stacked deep GCNs using residual connections, dense connections, and dilated convolutions, significantly improving GCN performance on point cloud semantic segmentation tasks while mitigating over-smoothing. From a data perspective, Rong [?] randomly dropped a certain proportion of edges during each training period to serve as both a data augmenter and message passing decelerator, reducing the convergence speed of over-smoothing.

Based on literature [?, ?] and our experimental observations, when using GCN for text graph representation learning, smoothing causes word features to converge to similar values, inevitably making word representations alike and damaging text classification performance. Therefore, to better measure and analyze the smoothness of word nodes for text classification, we propose WACD (Weighted Average Cosine Distance), a method for measuring the smoothness of multiple text graph representations. Unlike MAD [?], which applies to single graphs, WACD operates on multiple graphs, making it more suitable for our focused graph classification direction. Drawing from over-smoothing suppression methods in node classification, we propose RWACD (Regularization based on Weighted Average Cosine Distance) as a regularization term to suppress over-smoothing. Subsequently, we introduce ARS (Attention-based Residual Network Structure) to compensate for text information loss caused by graph topology differences. Unlike [?], ARS requires neither iterative training nor searching for important relevant nodes, using only attention mechanisms and residual structures to accelerate training speed. Finally, we present RA-GCN (RWACD-ARS based Graph Convolutional Neural Network Text Classification Algorithm). RA-GCN employs ARS to fuse text representations in the graph

representation learning layer and utilizes RWACD in the readout layer to suppress over-smoothing. Experiments on six Chinese and English datasets demonstrate RA-GCN' s performance, with multiple comparative experiments validating the roles of RWACD and ARS.

Overall, this paper makes the following contributions:

a) We propose WACD for measuring the smoothness of multiple text graph representations and introduce RWACD to suppress over-smoothing.

b) We propose ARS, an attention and residual-based network structure that compensates for text information loss caused by graph topology differences while suppressing over-smoothing.

c) We present RA-GCN, a graph convolutional neural network text classification algorithm based on RWACD and ARS, whose performance is demonstrated on six Chinese and English datasets.

d) We conduct comprehensive comparative experiments validating that both RWACD and ARS can suppress over-smoothing and improve model performance, confirm the correctness of compensating for text information loss from a graph topology perspective, and analyze and discuss the over-smoothing phenomenon in the per-text graph representation-based graph classification direction.

1 Related Research

Our proposed algorithm addresses the over-smoothing problem in text graph classification, representing an improvement and refinement of the method in literature [**?**]. Therefore, this section focuses on introducing literature [**?**], which primarily targets over-smoothing in node classification by proposing MAD for measuring graph representation smoothness, the MADreg regularization term for suppressing over-smoothing, and the AdaGraph iterative training algorithm.

1.1 MAD and MADreg

MAD measures graph representation smoothness based on cosine distance. Given a graph representation matrix $H \in \mathbb{R}^{n \times d}$, where n is the number of nodes and d is the feature dimension, the distance matrix D is calculated through cosine distance. The distance between each node pair is computed as:

$$D_{i,k} = 1 - \frac{H_{i,:} \cdot H_{k,:}}{\|H_{i,:}\| \|H_{k,:}\|}, \quad i, k \in [1, 2, ..., n]$$

where $H_{i,:}$ represents the i-th row of graph representation H. Cosine distance is used because it is unaffected by the absolute values of node vectors, better reflecting graph representation smoothness [**?**].

To obtain cosine distances between target node pairs, a target mask matrix $M_{tgt} \in \{0,1\}^{n\times n}$ is constructed to derive the distance matrix for target node pairs:

$$D_{tgt} = M_{tgt} \odot D$$

where $\odot$ denotes element-wise multiplication. The average of non-zero values per row is then calculated:

$$D_{tgt,i} = \frac{1}{\sum_{k=1}^{n} M_{tgt,ik}} \sum_{k=1}^{n} D_{tgt,ik}$$

where $x = 0$ if $x = 0$, and $x = 1$ otherwise. By averaging non-zero values in D_{tgt}, MAD is obtained for given target nodes:

$$MAD = \frac{1}{n} \sum_{i=1}^{n} D_{tgt,i}$$

Literature [**?**] observed that in node classification, two nodes with small topological distances are more likely to belong to the same category. Therefore, they proposed using graph topology to approximate node categories and calculated the difference between remote and neighbor node MAD values to estimate graph representation over-smoothing, termed MADGap:

$$MADGap = MAD_{rmt} - MAD_{neb}$$

where MAD_{rmt} is the MAD value for remote nodes and MAD_{neb} is the MAD value for neighbor nodes in the graph topology. Introducing coefficient λ yields the over-smoothing suppression regularization term MADreg:

$$MADreg = \lambda \times MADGap$$

1.2 AdaGraph

Literature [**?**] observed that optimizing graph topology using true labels mitigated over-smoothing and improved node classification performance, leading to the proposed iterative training algorithm AdaGraph for optimizing graph topology. The process involves first training a GNN, then optimizing the graph topology by deleting inter-class edges and adding intra-class edges based on prediction results. Repeating this process multiple times reduces graph topology differences, suppresses over-smoothing, and enhances node classification performance.

2 Proposed Algorithm

MAD, MADreg, and AdaGraph are suitable for single graph representation learning, whereas our focus is on graph classification based on multiple text graph representations. Consequently, literature [**?**] cannot be directly applied to our direction. Moreover, MADreg requires finding optimal orders to calculate MADGap, and AdaGraph needs iterative training to optimize graph topology, increasing training time and differing significantly from our focus.

Therefore, we propose WACD (Weighted Average Cosine Distance) for measuring smoothness across multiple text graph representations and RWACD for suppressing over-smoothing. We also introduce ARS, an attention and residual-based network structure that compensates for information loss caused by text graph topology differences while suppressing over-smoothing. Finally, we present RA-GCN, a graph convolutional neural network text classification algorithm.

2.1 WACD and RWACD

For a single text graph representation $H \in \mathbb{R}^{m \times d}$, where m is the number of word nodes and d is the word embedding dimension, WACD measures smoothness across multiple text graphs. Higher values indicate lower smoothness and lower over-smoothing probability, while lower values indicate higher smoothness and greater over-smoothing probability.

Treating all word pairs as target nodes, we calculate the Average Cosine Distance (ACD) for each text graph using equations (1)-(5). Weighted coefficients μ_i for each text' s ACD are computed based on text length to better estimate smoothness across multiple text graph representations:

$$\mu_i = \frac{l_i}{\sum_{i=1}^{b} l_i}$$

where b represents the number of texts and l_i is the length of the i-th text. The WACD calculation process is:

$$WACD = \sum_{i=1}^{b} \mu_i \times ACD_i$$

The regularization term RWACD is calculated as:

$$RWACD = 1 - WACD$$

WACD better measures smoothness across multiple text graphs through text length-weighted averaging of ACD. RWACD reduces over-smoothing probability by decreasing text graph representation smoothness. Compared to MADreg [**?**],

RWACD requires no optimal order search and is more suitable for our focused text graph classification direction.

2.2 ARS

Drawing from graph topology discussions in node classification [**?**], we argue that artificially constructed text graph topology deviates from the underlying true text topology, causing text information loss in graph representation learning. Therefore, we propose ARS, an attention and residual network structure for each network layer to mitigate this phenomenon while suppressing over-smoothing. Unlike [**?**], ARS requires neither iterative training nor searching for important relevant nodes, using only attention and residual mechanisms to accelerate training speed, making it more suitable for our focus. ARS will be detailed in Section 2.3.2.

2.3 RA-GCN

Figure 1 illustrates the RA-GCN algorithm framework. To enhance clarity, the framework uses red, blue, and green colors to highlight computational flows: red indicates GCN forward computation, blue shows ARS forward computation, and green represents RWACD forward computation. Overall, RA-GCN comprises three components: a text processing layer, a graph representation learning layer, and a readout layer. The text processing layer converts texts into inputs for graph representation learning. The graph representation learning layer learns text representations, primarily consisting of GCN and ARS components—GCN learns graph-level text representations while ARS compensates for information loss due to text graph topology differences. The readout layer obtains text categories, uses cross-entropy loss, and employs RWACD to suppress over-smoothing.

Below we detail each component and process.

2.3.1 Text Processing Layer As shown in Figure 1, for text $T = \{w_1, w_2, ..., w_n\}$ where n is the word count and w_i is a word, the text graph is represented as $G = (V, E, X)$, where $V = \{v_1, v_2, ..., v_m\}$ is the set of unique word nodes ($|V| = m \leq n$), E is the edge set, and $X \in \mathbb{R}^{m \times d}$ is the initial word feature matrix. Sliding windows construct the word node set V and edge set E, where E indicates connections between nodes v_i and v_k (1 if connected, 0 otherwise). The adjacency matrix $A \in \mathbb{R}^{m \times m}$ is constructed, and the degree matrix $D = \text{diag}(d_1, d_2, ..., d_m)$ is derived, where d_i is the degree of node v_i. The normalized adjacency matrix is:

$$\hat{A} = D^{-1/2} A D^{-1/2}$$

The initial word feature matrix X is built using pre-trained word embeddings, where d is the word embedding dimension.

2.3.2 Graph Representation Learning Layer As shown in Figure 1, the graph representation learning layer consists of GCN and ARS components. GCN learns word co-occurrence information to obtain text graph representations, while ARS uses attention and residual structures to produce the current layer' s text representation output.

1) **GCN**: For the l-th layer' s text graph representation:

$$H_{gcn}^{l+1} = \hat{A} H^{l} W_{gcn}^{l+1}$$

where $H^l \in \mathbb{R}^{m \times d}$ is the l-th layer' s text representation output, W_{gcn}^{l+1} is the learnable parameter matrix, and ρ is the activation function (LeakyReLU).

2) **ARS**: First, attention scores are assigned to all previous layers' text representation outputs and the current layer' s text graph representation:

$$\lambda_i = \sigma\left(\frac{1}{d_{mean}}(W_\lambda H_{total,i} + b_\lambda)\right), \quad i \in [0, 1, ..., l]$$

where $H_{total} = [H^0, H^1, ..., H^l, H_{gcn}^{l+1}]$, $H_{total,i}$ represents different dimensional text representations, λ_i are attention scores for each representation, W_λ and b_λ are learnable parameters, and σ is the sigmoid function.

Subsequently, the current layer' s text representation output H^{l+1} is obtained using attention scores and residual structure:

$$H^{l+1} = \sum_{i=0}^{l} \lambda_i H_{total,i}$$

2.3.3 Readout Layer As shown in Figure 1, the readout layer aggregates word features using attention mechanisms to obtain the final text representation and predict text categories. The final text representation G is calculated as:

$$h_i = \tanh(W_h H_i^L + b_h)$$
$$\alpha_i = \sigma(W_\alpha h_i + b_\alpha)$$
$$G = \sum_{i=1}^{m} \alpha_i h_i$$

where σ is the sigmoid function, α_i represents importance coefficients assigned to words, tanh further transforms word features, and $W_h, b_h, W_\alpha, b_\alpha$ are learnable parameter matrices. Additionally, to leverage both all words and important words, average and important features are extracted:

$$G_{avg} = \frac{1}{m}\sum_{i=1}^{m} h_i$$

$$G_{max} = \max_{i=1}^{m} h_i$$

$$G = [G_{avg}; G_{max}]$$

Finally, the softmax function predicts text categories. The objective function is cross-entropy loss with the RWACD regularization term:

$$\hat{y} = \text{softmax}(W_y G + b_y)$$

$$\mathcal{L} = -\sum_i y_i \log(\hat{y}_i) + \xi \times RWACD$$

where $\hat{y}$ is the predicted text category, W_y and b_y are learnable parameters, y is the true text category, and ξ is the RWACD coefficient.

3 Experiments

3.1 Experimental Environment

The experimental environment for this algorithm is shown in Table 1.

Table 1. Experimental Environment

Component	Configuration
OS	Ubuntu 20.04.3
Python	3.7.11
GPU	Nvidia GTX 2060S
IDE	Pycharm
CUDA Version	-
Deep Learning Framework	Tensorflow 2.4.1

3.2 Datasets

We evaluate RA-GCN' s performance on six datasets. Table 2 presents dataset statistics, where * indicates no validation set provided.

Table 2. Dataset Information

Dataset	Train	Val	Test	Classes	Avg Length
MR	-	-	-	2	-
Tnews	-	-	-	15	-
Ohsumed*	-	-	-	23	-

Dataset	Train	Val	Test	Classes	Avg Length
R8	-	-	-	8	-
SST-5	-	-	-	5	-
SST-2	-	-	-	2	-

a) **MR Dataset**: A 2-class English sentiment dataset with positive/negative polarity.

b) **Tnews Dataset** [**?**]: A Chinese news classification dataset with 15 categories.

c) **Ohsumed Dataset**: An English medical abstracts classification dataset for cardiovascular diseases with 23 categories.

d) **R8 Dataset**: An English Reuters news classification dataset with 8 categories.

e) **SST-2 and SST-5 Datasets**: English sentiment classification datasets with 2 and 5 classes, respectively.

3.3 Baselines

Since literature [**?**] focuses on node classification rather than our targeted text graph classification direction, we compare only with the following baselines:

a) Traditional deep learning text classification methods: TextCNN [**?**] and TextRNN [**?**].

b) Single text-word heterogeneous graph-based methods: TextGCN [**?**] and TextSGC [**?**].

c) Per-text graph representation-based methods: Huang [**?**], P-GCN and P-SGC (RA-GCN without RWACD and ARS), and RA-GCN.

3.4 Parameter Settings

For datasets without validation sets, we randomly split the training set into a 9:1 ratio for actual training and validation. For initial word features, English texts use 200-dimensional pre-trained GloVe [**?**] vectors, while Chinese texts use 300-dimensional vectors trained on Sogou news [**?**]. Out-of-vocabulary (OOV) words are randomly sampled from a uniform distribution [-0.01, 0.01]. The algorithm uses the Adam [**?**] optimizer with a learning rate of 0.001; other parameters are adjusted per dataset. Model performance is measured using Accuracy.

3.5 Experimental Results

Table 3 shows the accuracy of each model across six datasets, with results averaged over five training runs. RA-GCN achieves the best results on all datasets.

Table 3. Experimental Results

Model	MR	Ohsumed	SST-5	SST-2	Tnews	R8
TextCNN	0.7775	-	-	-	-	-
TextRNN	0.7768	-	-	-	-	-
TextGCN	0.7674	-	-	-	-	-
TextSGC	-	-	-	-	-	-
Huang	-	-	-	-	-	-
P-GCN	-	-	-	-	-	-
P-SGC	-	-	-	-	-	-
RA-GCN	**Best**	**Best**	**Best**	**Best**	**Best**	**Best**

Compared to traditional methods, CNN and RNN underperform graph-based methods on most datasets, proving graph models benefit text classification. Per-text graph classification models (Huang, P-GCN, P-SGC, RA-GCN) outperform single heterogeneous graph models (TextGCN, TextSGC), particularly on short-text datasets like MR and SST-2, validating the effectiveness of per-text graph representation methods.

RA-GCN shows significant improvements on short-text datasets (MR, SST-2, SST-5, Tnews) but smaller gains on long-text datasets. Since constructed text graph topology differs from the true underlying topology, short texts have smaller graph scales where word information propagates extensively and rapidly under GCN' s message-passing mechanism. RWACD and ARS effectively suppress over-smoothing and compensate for topology-induced information loss, enabling RA-GCN to learn more accurate representations. However, long texts have larger graph scales where topology differences impede smooth information propagation, making it harder for models to learn accurate representations and limiting RWACD and ARS' s effectiveness, resulting in less significant performance improvements.

3.6 Comparative Experiments and Over-smoothing Analysis

This subsection uses GCN and SGC as bases to validate RWACD and ARS' s roles in improving performance and suppressing over-smoothing, analyzing the over-smoothing phenomenon. Experiments are conducted on MR and SST-5 datasets, with four samples from the MR test set selected for visualization and analysis (sample descriptions in Table 4).

Table 4. Sample Description

Sample	Class
a magnificent drama well worth tracking down	1
an awkwardly contrived exercise in magic realism	0
i' ll put it this way if you' re in the mood for a melodrama narrated by talking fish, this is the movie for you	1
children and adults enamored of all things pokemon won' t be disappointed	0

We construct models containing RWACD or ARS individually (RW-GCN, RW-SGC, ARS-GCN, ARS-SGC) and models containing both or neither (P-GCN, P-SGC, RA-GCN, RA-SGC). We examine their performance on MR and SST-5 to explore RWACD and ARS' s impact. Different symbols denote different models (Table 5).

Table 5. Model Description

Model	Symbol
P-GCN, P-SGC	
RW-GCN, RW-SGC (RW-models)	
ARS-GCN, ARS-SGC (ARS-models)	
RA-GCN, RA-SGC (RA-models)	

3.6.1 Roles of RWACD and ARS 1) Impact of RWACD and ARS on Classification Performance

Table 6 shows the text classification accuracy of eight models on MR and SST-5 test sets, averaged over three training runs.

Table 6. Comparative Experimental Results

Model	MR	SST-5
P-GCN	-	-
P-SGC	-	-
RW-GCN	-	-
RW-SGC	-	-
ARS-GCN	-	-
ARS-SGC	-	-
RA-GCN	**Best**	**Best**
RA-SGC	-	-

From the text graph construction perspective, all eight GCN/SGC-based models outperform TextGCN and TextSGC, highlighting the advantages of per-text graph classification. Regarding RWACD and ARS inclusion, P-models (without both) perform worst. RW-models show slight improvements over P-models, proving RWACD enhances performance. ARS-models demonstrate superior performance on MR and SST-5, outperforming both P- and RW-models, confirming the correctness of optimizing performance from a graph topology perspective. RA-models achieve optimal performance, proving RWACD and ARS jointly improve classification performance.

2) Impact of RWACD and ARS on Different Layer Counts

Figure 2 shows accuracy and WACD curves on the MR test set across layers. P-models perform worst in both metrics. As layers increase, P-models' WACD gradually decreases while accuracy first increases then continuously declines, indicating that moderate smoothing benefits performance but excessive smoothing harms it. RW-models show slight improvements, demonstrating RWACD reduces over-smoothing. ARS-models exhibit significant improvements, showing that compensating for topology-induced information loss substantially enhances performance and suppresses over-smoothing. RA-models achieve the best results, confirming that RWACD and ARS jointly improve performance and suppress over-smoothing.

3) Analysis of ARS' s Role

ARS-models' outstanding performance stems from compensating for text information loss caused by graph topology differences. To further validate ARS, we designed experiments to test whether models can achieve performance comparable to intact graph data when trained on corrupted graphs. Using 2-layer P-GCN and ARS-GCN as baselines, we removed the readout layer' s attention mechanism, randomly deleted edges to corrupt graph topology, and gradually increased deletion ratios. Unlike [**?**], we applied this to all texts including the test set, maintaining topology during training. To highlight results, P-GCN' s maximum deletion ratio was 20% while ARS-GCN' s was 50%. Results on MR are shown in Figure 3.

On intact graphs, ARS-GCN significantly outperforms P-GCN, confirming that artificially constructed topology deviates from true underlying topology, validating ARS' s motivation. As deletion ratios increase, P-GCN' s performance drops sharply, while ARS-GCN maintains performance comparable to or exceeding P-GCN' s even at ~30% deletion, proving ARS compensates for topology-induced information loss and validating the topology optimization approach.

The ARS-GCN curve alone shows sharp performance decline beyond 20% deletion because high deletion ratios create isolated nodes without connections, preventing information exchange and hindering the model from capturing word co-occurrence information. However, at 0%-20% deletion, isolated nodes are rare, and ARS successfully compensates for topology differences, maintaining performance near the original ARS-GCN level.

To observe ARS' s behavior more clearly, we examine classification performance on Table 4 samples. With ARS-GCN using 30% deletion ratio and P-GCN using intact data, results (averaged over three runs) are shown in Table 7, where √ indicates correct prediction and × indicates incorrect prediction.

Table 7. Model Prediction Results for 4 Samples

Sample	Class	P-GCN (Intact)	ARS-GCN (30% deletion)
1	1	0.000/1.000 (√)	0.996/0.004 (√)
2	0	0.239/0.761 (√)	0.358/0.642 (×)
3	1	0.022/0.978 (√)	0.973/0.027 (√)
4	0	0.384/0.616 (√)	0.810/0.190 (√)

ARS-GCN' s predictions on the first three samples approach P-GCN' s results, while on the fourth sample, ARS-GCN predicts correctly where P-GCN fails. This demonstrates that ARS-GCN with 30% deletion performs comparably or even superiorly to P-GCN on intact data, validating Figure 3' s results from a real sample perspective.

In summary, artificially constructed text graph topology differs from true underlying topology, and ARS compensates for this difference-induced information loss, improving model performance.

4) Case Study

Building on Table 4 samples, we visualize average distances between words within the first two samples (Figure 4), ACD value curves across layers (Figure 5), and predictions on samples 3-4 by different-layer P-GCN and RA-GCN models (Table 8).

Table 8. Model Predictions for Sample 3 and Sample 4

Model (Sample)	Layer 1	Layer 2	Layer 3	Layer 4
P-GCN (Sample 3)	-	-	-	-
ARS-GCN (Sample 3)	-	-	-	-
P-GCN (Sample 4)	-	-	-	-
ARS-GCN (Sample 4)	-	-	-	-

Figure 4 shows RA-GCN significantly increases average distances between words (e.g., "worth"rises from 0.12 to 0.61). In Figure 5, P-GCN's ACD values approach 0 by the third layer, making words similar—consistent with literature [**?**]' s over-smoothing description. RA-GCN' s ACD values show clear improvement, demonstrating RWACD and ARS suppress over-smoothing. In Table 8, P-GCN predicts correctly 3 times while RA-GCN predicts correctly all times, showing RWACD and ARS improve sample classification performance.

In summary, using P-models as baseline, over-smoothing in our focus direction occurs per-text-graph, emerging in shallow networks and increasing with network stacking, harming performance. RWACD and ARS both reduce over-smoothed samples, suppress over-smoothing, and improve classification performance.

3.6.2 Over-smoothing Phenomenon Analysis Figure 5 in Section 3.6.1 shows that 3-layer P-GCN already drives sample ACD toward 0, exhibiting over-smoothing as described in literature [**?**]. Figure 2 shows P-models' WACD decreasing with layers while accuracy first rises then falls, indicating moderate smoothing benefits classification but excessive smoothing harms it. We hypothesize that in per-text graph representation-based classification, over-smoothing emerges in some samples as layers stack, with over-smoothed texts increasing and affecting performance.

To verify this, we analyze how the number of texts with ACD below a threshold varies with network layers using our eight models on MR and SST-5 test sets. Results are shown in Figure 6, with different thresholds per model (annotated). Figures (a)-(b) show GCN/SGC results on MR, (c)-(d) on SST-5.

Combining Figure 2 and Figure 6' s P-model results, texts with ACD < 0.3 increase with layers, WACD decreases, and accuracy first rises then falls. Combined with Figure 4-5's P-GCN visualizations, this confirms that over-smoothing in per-text graph classification manifests per-text-graph, emerging in shallow networks and intensifying with layer stacking, though performance degradation begins at 2-3 layers.

RW- and ARS-model curves show both RWACD and ARS reduce over-smoothed texts and suppress over-smoothing. RA-models achieve the best results, demonstrating that RWACD and ARS jointly reduce over-smoothed samples, suppress over-smoothing, and improve classification performance.

4 Conclusion

This paper proposes WACD, a weighted average cosine distance metric for measuring smoothness across multiple text graph representations, and RWACD, a regularization term for suppressing over-smoothing. We introduce ARS, an attention and residual network structure that compensates for text information loss caused by text graph topology differences while suppressing over-smoothing. Finally, we present RA-GCN, a graph convolutional neural network text classification algorithm based on RWACD and ARS. Experiments on six datasets demonstrate RA-GCN' s performance, with multiple comparative experiments validating RWACD and ARS' s effectiveness.

References

[1] Li Qian, Peng Hao, Li Jianxin, et al. A survey on text classification: from shallow to deep learning [J/OL]. ACM Trans on Interactive Intelligent Systems, 2021, 37. (2021-04) [2021-12-11]. https://arxiv.org/pdf/2008.00364.pdf.

[2] Kowsari K, Jafari M K, Heidarysafa M, et al. Text classification algorithms: a survey [J]. Information, 2019, 10(4): 150.

[3] Chiu B, Sahu S K, Sengupta N, et al. Attending to inter-sentential features in neural text classification [C]// Proc of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 2020: 1685-1688.

[4] He Li, Zheng Zaoxian, Xiang Fengtao, et al. Research progress of text classification technology based on deep learning [J]. Computer Engineering, 2021, 47(2): 1-11.

[5] Yao Liang, Mao Chengsheng, Luo Yuan. Graph convolutional networks for text classification [C]// The 33rd AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2019, 33(1): 7370-7377.

[6] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks [C/OL]// The 5th International Conference on Learning Representations. 2017. (2017-02) [2021-12-11]. https://arxiv.org/pdf/1609.02907.pdf.

[7] Wu F, Zhang Tianyi, Souza A, et al. Simplifying graph convolutional networks [C]// Proc of the 36th International Conference on Machine Learning. [S.l.]: PMLR, 2019: 6861-6871.

[8] Huang Lianzhe, Ma Dehong, Li Sujian, et al. Text level graph neural network for text classification [C]// Proc of Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. Stroudsburg: ACL, 2019: 3440-3450.

[9] Zhang Yufeng, Yu Xueli, Cui Zeyu, et al. Every document owns its structure: inductive text classification via graph neural networks [C]// Proc of the 58th Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2020: 334-339.

[10] Li Yujia, Tarlow D, Brockschmidt M, et al. Gated graph sequence neural networks [C/OL]// The 4th International Conference on Learning Representations. 2016. (2016) [2021-12-11]. https://arxiv.org/pdf/1511.05493.pdf.

[11] Fan Guofeng, Liu Gui, Yao Shaowen, et al. Text classification model with graph network based on semantic dependency parsing [J]. Application Research of Computers, 2020, 37(12): 3594-3598.

[12] Li Qimai, Han Zhichao, Wu Xiaoming. Deeper insights into graph convolutional networks for semi-supervised learning [C]// Proc of the 32nd AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2018: 3538-3545.

[13] Cai Chen, Wang Yusu. A note on over-smoothing for graph neural networks [EB/OL]. (2020-06-23) [2021-12-11]. https://arxiv.org/pdf/2006.13318.pdf.

[14] Chen Deli, Lin Yankai, Li Wei, et al. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view [C]// The 34th AAAI Conference on Artificial Intelligence. Palo Alto: AAAI Press, 2020: 3438-3445.

[15] Wang Guangtao, Ying R, Huang Jing, et al. Multi-hop attention graph neural networks [C]// Proc of the 30th International Joint Conference on Artificial Intelligence. [S.l.]: ijcai.org, 2021: 3089-3096.

[16] Yang Tianmeng, Wang Yujing, Yue Zhihan, et al. Graph pointer neural networks [EB/OL]. (2021) [2022-01-05]. https://arxiv.org/pdf/2110.00973.pdf.

[17] Vinyals O, Fortunato M, Jaitly N. Pointer networks [C]// Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems. 2015: 2692-2700.

[18] Li Guohao, Muller M, Thabet A, et al. DeepGCNs: Can GCNs go as deep as CNNs? [C]// Proc of the IEEE/CVF International Conference on Computer Vision. Piscataway, NJ: IEEE Press, 2019: 9266-9275.

[19] Rong Yu, Huang Wenbing, Xu Tingyang, et al. DropEdge: towards deep graph convolutional networks on node classification [C/OL]// The 8th International Conference on Learning Representations. 2020. (2020-05-12) [2022-01-05]. https://arxiv.org/pdf/1907.10903.pdf.

[20] Xu Liang, Hu Hai, Zhang Xuanwei, et al. CLUE: A Chinese language understanding evaluation benchmark [C]// Proc of the 28th International Conference on Computational Linguistics. [S.l.]: International Committee on Computational Linguistics, 2020: 4762-4770.

[21] Kim Y. Convolutional neural networks for sentence classification [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2014: 1746-1751.

[22] Liu Pengfei, Qiu Xipeng, Huang Xuanjing. Recurrent neural network for text classification with multi-task learning [C]// Proc of the 25th International Joint Conference on Artificial Intelligence. [S.l.]: IJCAI/AAAI Press, 2016: 2873-2879.

[23] Pennington J, Socher R, Manning C D. Glove: Global vectors for word representation [C]// Proc of Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2014: 1532-1543.

[24] Li Shen, Zhao Zhe, Hu Renfen, et al. Analogical reasoning on Chinese morphological and semantic relations [C]// Proc of the 56th Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2018: 138-143.

[25] Kingma D P, Ba J L. Adam: A method for stochastic optimization [C/OL]// The 3rd International Conference on Learning Representations. 2015. (2015) [2022-01-05]. https://arxiv.org/pdf/1412.6980.pdf.

Note: Figure translations are in progress. See original paper for figures.