Within-Dataset Graph Enhancement for Short Text Classification

Wei Pang, Ruixue Duan[1], and Ning Li[1]

Beijing Information Science and Technology University, Beijing, China
pangweitf@bupt.cn, duanruixue@bistu.edu.cn

Abstract. Prior work on short text classification has mainly focused on augmenting short texts with features obtained from external sources, which often leads to topic drift and ambiguity. Instead, this paper explores internal knowledge within dataset to enhance short text classification. In particular, we construct a single directed graph on the dataset where nodes denote words and edges represent the order between words. We treat a short text as a path and then propose a novel graph-based model, which aggregates graph topological features of each word into themselves. Compared with previous work, we focus on enhancing the representation of short texts based on geometry-based neighbors, regarded as internal knowledge from the dataset. Furthermore, we construct two new Chinese short text datasets and develop a simple method for short text classification. Experimental results on nine benchmark datasets validate the effectiveness of the proposed method and show improvements in classification accuracy.

Keywords: Short Text Classification · Directed Graph · Topological Feature· Graph Attention Network.

1 Introduction

Short text classification is the task of classifying short texts into predefined class labels. Recently, short texts have been produced explosively in the form of various domains, such as user comments on e-commerce platforms, user utterances in many conversational systems [1], search queries in information retrieval (IR)[2–4], and rapidly growing bullet screen (danmu messages) on short video platform [5, 6]. Classifying these short texts into predefined labels has become a significant problem in many applications, e.g., utterance-level emotion analysis in dialogues [7] and query intent classification in IR[8, 9]. Unlike long texts such as blogs, news articles, and web pages, short texts pose a new challenge for contextual understanding due to the limitation of the input length [3, 10, 1]. In general, short sentences lack enough textual features, making their understanding more challenging. Instead of adding features obtained from external sources, this paper focuses on in-dataset features for short text classification.

As mentioned above, it is challenging to adapt traditional long text classification methods to short texts with good performance[11–13] because shortness

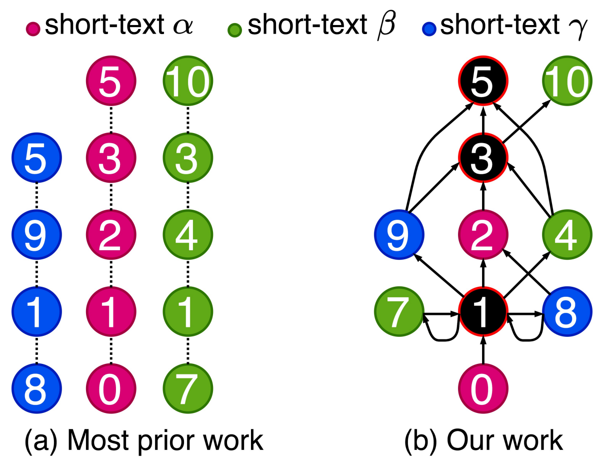


Fig. 1. Most prior work on short text classification considers one short text in isolation, e.g., the left part (a): three short texts tagged with α, β, and γ are shown in parallel. By contrast, in our work, the right part (b) displays the three short texts in a graph, where they interleave via overlapped words. Contextual features of a short text are not only from itself (within-sentence) but also from other related short texts (cross-sentence). For conciseness, we denote a word with a number in this paper.

and sparsity of the input text. Previous work assumes that the missing context in short text can be obtained from external sources [14–16]. Therefore, a major concern is shift to increase textual features with external knowledge from out-of-dataset sources. For example, they collect similar titles and snippets returned by search engines [17, 18] to expand the raw short text or fetch attributes of the mentioned entity in short text from databases [19, 14, 10], such as Wikipedia and YAGO. Specifically, in [10], they retrieve conceptual information from YAGO and Freebase as background knowledge. In [20], they leverage textual category information from Wikipedia based on explicit mentions of entities in short texts.

Most current work on short text classification is limited to classifying each short text separately. As shown in the left part of figure 1, three short texts tagged with α, β, and γ are represented in isolation, ignores intra-relations across them.

Intuitively, we observe rich adjacent relationships across short texts in a dataset where related short texts often share certain words. For example, the right side of figure 1 shows that the three short texts share the same words. Although the words labeled 9 and 4 do not appear in the short text α, they may provide language prompts to help classify it, which should be worth considering for short text classification.

With these observations, we think the interconnected short texts might provide an essential piece of contextual prompts, which can also be a source of internal knowledge. Specially, we construct a single directed graph (digraph briefly) on the dataset. As seen in the right part in figure 1, a node represents a word (or a Chinese character), such as nodes marked as v_0, v_1, v_2 that denote three different words; a directed edge represents orders between pairs of nodes, which is convenient for encoding word order-related semantic information. Compared to prior work, we regard a short text as a path in the digraph, and each short text is represented as a sequence of nodes and edges, i.e, the short text α can be written as a set of $\alpha = \{v_0, e_{0,1}, v_1, e_{1,2}, v_2, e_{2,3}, v_3, e_{3,5}, v_5\}$, naturally.

Moreover, a basic idea is that the two or more interlinked sentences could form some topological geometry structures in the digraph, e.g., a simple triangle consists of three nodes v_1, v_8, v_2, or other polygons. By looking closer at these geometry-based features, we explore how to integrate them into a short text. In Figure 1, we combine nodes v_1, v_2, v_3 of α with a neighbor node v_4 to form a quadrangle, which are closely related to the path of short text α that can serve as candidate topological features. As a result, an augmented representation for α can be rewritten as $\hat{\alpha} = \alpha \cup \{v_4, e_{1,4}, e_{4,3}, v_9, e_{1,9}, e_{9,5}, \cdots\}$.

We developed a classification method called PathWalk. For each word in a text, we first fuse neighbors' topological features into node (i.e. word) representation via a graph attention mechanism. Then the list of representations of nodes can be fed into an encoder to obtain a final short text representation used for classification. At last, the final representation is passed through a softmax classifier for prediction, resulting in a probability distribution over predefined labels. In addition, we present 2 new large short datasets: DanMu with size of 100K used for binary sentiment classification, and CVQD consists of 150K user queries categorized by 26 topics. Extensive experiments show that the proposed PathWalk method obtains results competitive with all compared methods on most benchmark datasets. Significantly on the DanMu dataset, the proposed method achieves state-of-the-art accuracy compared to all the baselines, including the Bert-based ERNIE model. To summarize, our contributions are mainly three-fold:

- We propose a novel topological graph-based model for short text classification, thinking of in-dataset geometric structures as a source of internal knowledge and constructing a single directed graph on the dataset. It aims to aggregate neighbours' topological features without requiring external sources.
- We present a simple yet effective graph-based method PathWalk, that enhances short text representation using geometry-based topological features through graph attention. It considers neighboring features and helps alleviate the problem of shortness in a short text.
- We construct two large-scale Chinese datasets for short text classification. Experiments on nine benchmark datasets demonstrate that our method achieves state-of-the-art results on six of nine datasets. Especially our method obtains start-of-the-art on the new DanMu dataset.

2 Related Work

Short text classification is a fundamental problem in natural language processing (NLP), mainly based on textual representation learning [10, 13]. This section briefly categorizes existing text classification methods into three groups from a perspective of textual representation.

Bag-of-Words model In traditional machine learning, most previous work on text classification methods heavily rely on human-designed features [21, 13], and they view a text as a bag of words [22, 23]. One shortcoming is that it disregards orders between words in the text. These methods, such as support vector machine [23] and Logistic Regression [23], are widespread approaches for traditional text classification and achieve good performance when the right bag of features is designed. However, due to the limited content of a short-text, bag-of-words-based methods usually fail to attain good performance for short text classification.

Neural network-based methods In NLP, three types of neural network-based methods have obtained great progress on text classification, including convolutional neural networks (CNN) [24–26], recurrent neural networks (RNN) model [27, 28] and various recent transformer-based models[29, 30].

CNN-based methods commonly convert a sentence into a feature matrix of words based on pre-trained word vector [25, 24, 31], e.g., word2vec or GloVe vectors. Suppose sentence length is n, the dimension of word vector is d, then the dimensionality of the sentence matrix is $n \times d$. CNN is good at capturing key-phrases [32] or informative ngrams [21] in text. In [25], TextCNN is presented for sentence-level classification, using a simple CNN with parameter tuning, and achieves strong results on sentiment analysis and question classification tasks.

RNN-based methods view a text as a sequence of words [27, 32] that is good at extracting long-range semantic dependency. In [27, 33, 10], a simple bidirectional LSTM (BiLSTM) model is used for text classification called TextRNN. It obtains competitive or new state-of-the-art results on sentiment analysis and topic classification tasks.

Transformer-based models have become a dominant neural architecture in the field of NLP. Transformer adopts a self-attention mechanism that can model all the interactions between every word in the text, improving state-of-the-art performance on various NLP tasks.

Graph-based methods The closest work to ours is in [34–37]. Ding et al. [34] built a hypergraph for each text document to capture word interactions, using an attention over node and edge respectively. Jian Tang et al. [35] proposed to build a large-scale heterogeneous graph on a text corpus. The graph consists of three types of text networks: word-word network, word-document network and word-label network. Liang Yao et al. [36] presented to use of graph convolutional neural networks (GCN) for text classification, building a single text graph for a corpus and converting text classification into document-node classification.

3 Models

This section introduces a digraph-based method PathWalk for short text classification. An overview of our model architecture is depicted in figure 2. There are four modules: Dataset-level Graph Construction (DGC), Geometry-based

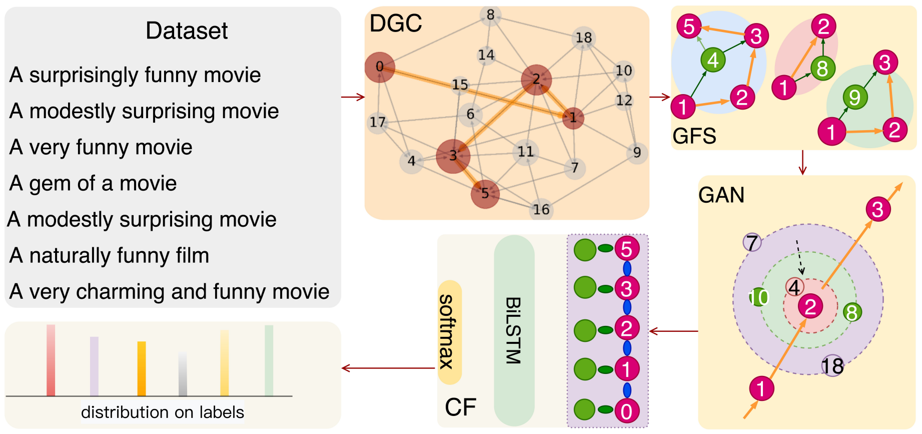

Dataset
A surprisingly funny movie
A modestly surprising movie
A very funny movie
A gem of a movie
A modestly surprising movie
A naturally funny film
A very charming and funny movie
DGC
GFS
GAN
softmax
BiLSTM
CF
distribution on labels

Fig. 2. Overall structure of the proposed model.

Feature Selection (GFS), Graph Attention Network (GAN), and Classification Framework (CF). Where the DGC module first constructs a dataset-level digraph from a dataset. GFS module selects some geometry-based features in the digraph for use in the GAN. GAN module aggregates the features of neighboring nodes or edges via the attention mechanism. Finally, the CF module employs a basic classification method based on augmented features for prediction. In the remainder of this section, we go into more detail on each module, respectively.

Dataset-level Graph Construction (DGC)

Given a dataset, we construct a directed graph $G = (V, E)$ (digraph in brief), in which V is a set of nodes $\{v_i\}_{i=0}^{N}$ that correspond to the dataset's vocabulary of size N. E is a set of directed edges as written in $\{e_{i,j} \mid e_{i,j} = (v_i, v_j), i, j \in [0, \cdots, N]\}$, i.e., $e_{i,j}$ is a directed edge from node v_i to node v_j. For each short text, we add all the words into the digraph one by one, connecting them with a directed edge. The right part of figure 1 shows an example, short text α contains nodes $\{v_0, v_1, v_2, v_3, v_5\}$, there are four directed edges linking them, i.e., $\{e_{0,1}, e_{1,2}, e_{2,3}, e_{3,5}\}$.

We have three notes on DGC as follows. Firstly, different from previous dataset-level graphs [36], DGC includes directed edges representing the order of words, which is used to explicitly capture order-related semantic information within a text sequence. Secondly, any word in digraph has lots of edges linked with other words. Two or more related sentences may form some geometrical structures, such as a loop or a triangle. Borrowing an item from point-set topology, we call such structures simplex seen later in GFS. Thirdly, we assume that the dataset's internal knowledge might come from these geometrical structures. It is the major insight of this work, which may provide a new perspective to explore the usage of knowledge within dataset.

Geometry-based Feature Selection (GFS)

Unlike previous work, we think of a short text as a path in the digraph, our goal is to explore within-dataset geometry and capture word associations from them. It is crucial to define topological structure along the path. For example, given a short text α in figure 3(a) that highlighted in red, we refer to the nodes v_1, v_2, v_3, v_5 in α as in-text nodes (or words), other nodes as neighbor nodes. Let an n-simplex be $v_0, v_1, \ldots, v_n$ consists of $n+1$ nodes, e.g. a quadrangle consists of v_1, v_2, v_3, v_9 that is an 3-simplex, which might provide contextual prompts for α. We define three types as optional neighbor features in the following:

(a) 1-simplex-based features, which are represented by one neighbor node that pointed to one in-text node plus with the directed edge between them, such as neighbor node v_9 and edge $e_{9,3}$ in figure 3(a), which can augment textual feature to in-text node v_3 of α.

(b) 2-simplex-based features, which are defined by a triangle, include the directed edges between them. For example, the triangle denoted by $\{v_1, v_2, v_8\}$ in figure 3(b), the in-text node v_1 can reach the follow-up v_2 through a neighbor node v_8. Both of $e_{1,8}$ and $e_{8,2}$ can be as an optional path between the two in-text nodes v_1 and v_2, which may capture complementary short-range contextual information for α.

(c) 3-simplex or higher-simplex based features, a 3-simplex shape is represented by a quadrangle such as $\{v_1, v_2, v_3, v_4\}$ in figure 3(c), a 4-simplex shape like a pentagon $\{v_1, v_2, v_3, v_5, v_4\}$. Intuitively, the neighbor node v_4 and edge $e_{4,3}$ are considered to have the ability to capture long-distance interactions between in-text nodes v_1 and v_3 of α, bypassing in-between node v_2.

To avoid introducing noises, we constraint a topological structure that starts with an in-text node of α pointing to one neighbor node and then re-pointing to another α's in-text node. Taking the 4-simplex in figure 3(c) as an example, although the in-text word v_1 is far away from the last in-text word v_5, v_1 has extra short path that connects to v_5 via neighbor node v_4 that not in α, bypassing the two in-text nodes v_2 and v_3. These shortcut connections, such as through v_4, which may efficiently represent long-range associations in a text, would quickly transfer the complementary long-range context of v_1 to v_5.

Graph Attention Network (GAN)

Following the output of the GFS module, GAN aims to selectively aggregate topological features into the original features of the each word in text. Three steps are proposed as below.

Subset of n-simplexes features: We first adopt a random strategy to sample or choose a subset of candidate neighbour features of size K, which is used for attention calculation in equation 4:

$$Subset_K = random\_choose(\{n{-}simplexes\}_{n=0}^{4}). \tag{1}$$

Edge-aware node represention: An edge-aware node represention is to fuse features of directed edge into node represention. Let $h_{e_{i,j}} \in R^d$ denote represention of the directed edge $e_{i,j}$ from node v_i to v_j, $h_{v_i}, h_{v_j} \in R^d$ denote represention of nodes. Here let the two nodes v_i and v_j be in-text nodes that belong to α.

The edge-aware node representionare h_{v_i} is computed as follows:

$$\begin{aligned} \widehat{h_{e_{i,j}}} &= gelu(W_e[h_{e_{i,j}}, (h_{v_j} - h_{v_i})] + b_e), \\ h_{v_i} &= gelu(W_v[h_{v_i}, \widehat{h_{e_{i,j}}}] + b_v), \\ h_{v_i} &= LayerNorm(h_{v_i}), \end{aligned} \tag{2}$$

where the two map matrixes W_e, W_v and their corresponding bias b_e, b_v are trainable parameters, $[\cdot, \cdot]$ denotes concatenation, d is the dimension of the node or edge represention. We use a difference operator as $h_{v_j} - h_{v_i}$, which means the representaion of the directed edge $e_{i,j}$ is closely related to both of the head and tail node.

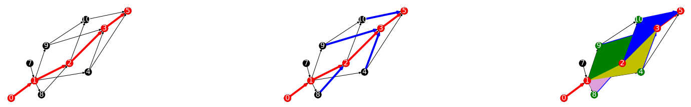

(a) 1-simplex-based features (b) 2-simplex-based features (c) 3/4-simplex based features

Fig. 3. Geometry-based topological feature selection for short text $\alpha = \{v_0, v_1, v_2, v_3, v_5\}$. Figure best viewed in color.

More specially, we assume that v_k is a neighbor node that does not belong to α. There are two directed edges $e_{i,k}$ and $e_{k,j}$ that links the three nodes. The representation of v_k can be calculated as follows:

$$\begin{aligned} h_{v_k} &= gelu(W_o[\widehat{h_{e_{i,k}}}, h_{v_k}, \widehat{h_{e_{k,j}}}] + b_o), \\ h_{v_k} &= LayerNorm(h_{v_k}), \end{aligned} \tag{3}$$

where the projection matrix W_o and the bias b_o are trainable parameters.

For each in-text node in short text α, such as $v_i \in \alpha$, there exists a subset of neighbour nodes of size K, $Subset_K$, where each neighbour node v_k is not belong to text α. To selectively focus on some informative neighors for the specific in-text node, we employ a graph attention model to learn the importance on the subset, and assign different attention scores to them. Formally,

$$\begin{aligned} h_{i,k} &= gelu(W_2[W_1 h_{v_i} + b_1, W_1 h_{v_k} + b_1] + b_2), \\ \varphi_{k,i} &= \frac{exp(h_{i,k})}{\sum_{n \in Subset_K}^{K} exp(h_{i,n})}, \\ \widetilde{h}_{v_i} &= gelu(h_{v_i} + \sum_{k=1}^{K} \varphi_{k,i} h_{v_k}), \end{aligned} \tag{4}$$

where $W_{1,2}$ and $b_{1,2}$ are learnable parameters, $\varphi_{k,i}$ denotes the weight of attention of neighbor node v_k to the in-text node vi. $\widetilde{h}_{v_i}$ is output representation for the in-text node $v_i \in \alpha$ that can fuse these neighbour context features to obtain the final embedding for v_i.

Classification Framework (CF)

Figure 2 shows a classification framework. It contains a two-layer bidirectional long short-term memory (BiLSTM) network as text encoder as [10] does, a softmax classifier layer to yield a probability distribution over the predifined labels. Formally,

$$\begin{aligned} \hbar_{\alpha}^{(f)} &= LSTM^{(f)}(\{\widetilde{h}_{v_i}\}_1^l), \\ \hbar_{\alpha}^{(b)} &= LSTM^{(b)}(\{\widetilde{h}_{v_i}\}_1^l), \\ p(\hat{y} \mid \alpha) &= softmax(W_c[\hbar_{\alpha}^{(f)}, \hbar_{\alpha}^{(b)}] + b_c), \\ \hat{y} &= \arg\max_{\hat{y}} p(\hat{y} \mid \alpha), \end{aligned} \tag{5}$$

where W_c and b_c are trained parameters, l is the sentence length of α. $\hbar_{\alpha}^{(f)}$ and $\hbar_{\alpha}^{(b)}$ represent the last hidden state for forward and backward pass respectively, which are concatenated to feed to the softmax layer. $p(\hat{y} \mid \alpha)$ denotes the probability distribution produced on the predefined labels, the higher one $\hat{y}$ is choosed as the predicted result,

Model Learning With the guide of labels, we can optimize the proposed model via back propagation and learn the embeddings of the graph, including the node and directed edge. For supervised text classification task, we can minimize the categorical Cross-Entropy loss between the prediction and the ground-truth and the loss can be optimized as: $\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i)$, where N is the batch size, y is the ground truth label, and θ are trainable model parameters. In our work, we use dropout and gradient value clip to prevent overfitting in gradient descent step.

4 Experiment and Evaluation

Table 1. Summary statistics of the datasets.

Dataset	Class	AvgLen	Size	Train	Dev	Test	Nodes	Edges
DanMu	2	6	100K	70K	15K	15K	2,913	118,14
CVQD	26	8	150K	105K	22.5K	22.5K	3,601	232,387
MR	2	20	10.6K	7464	1,599	1,599	4,479	71,832
SST1	5	18	11.8K	8,544	2,210	1,101	4,615	73,825
SST2	2	18	9.6K	6,920	872	1,821	3,929	60,487
SUBJ	2	23	10K	7,001	1,495	1,499	5,061	79,591
TREC	6	11	6.4K	5,452	500	500	1,402	13,739
CR	2	16	3.7K	2,639	566	565	1,485	23,588
MPQA	2	3	10.6K	7,423	1,590	1,590	1,157	8,185

Datasets We evaluate experiments on seven widely used benchmark datasets, including Movie Review (MR)[38], TREC [39], Customer Reviews (CR) [40], MPQA [41], SST-1/2[42], and SUBJ. Due to there being no standard split for MR, TREC, CR and MPQA, we randomly shuffle their datasets and split them into the training (70%), dev (15%), and test (15%) set, respectively. We summarize the detailed statistics in Table 1. However, on the one hand, the average sentence length of most above datasets is quite long, such as the average length of SUBJ is 23, MR is 20, and SST-1/2 18. On the other hand, our NLP community lacks large-scale Chinese datasets for short text classification. Therefore, we built 2 large short datasets: DanMu and CVQD. We crawled both with a spider from a wide range of sources, such as the open web or video search engines, short-video websites, and Weibo websites. We annotated each sample automatically according to our hand-crafted features, such as their domain names in URL or taking the majority vote in the returned snippets of search engines.

Chinese Video Query dataset (CVQD) We collected a query dataset of size 150K from some popular Chinese video search engines, which involves classifying a query into 26 query types, whether the query is about Movie, Cartoon, Travel, Fashion, Culture, Music, and Game, etc. We split the dataset into the training (70%), validation (15%), and test (15%) set, respectively.

Chinese DanMu dataset (DanMu) We gathered user reviews of size 100K from some famous Chinese short-video websites and assigned binary emotion tagging, i.e., positive or negative, for the task of binary sentiment classification. We follow the standard partition as the CVQD did.

Compared Methods Following the compared methods used in [34], we adopt the four classes of popular baselines. (1) Word embedding-based methods, such as fastText[43, 23]; (2) Sequence-based methods, including a family of CNN's-based methods and LSTMs-based methods[27, 10, 33]; (3) Transformer-based models[44]; (4) ERNIE[45], that is an extension of the pre-trained Bert model.

Implementation Details We fix the maximum length to 20 for the Chinese and English datasets. We set the batch size to 64, embedding size of 300-dimension for node and edge, encoder hidden state to 300, hyperparameter K to 45 for all datasets, random seed 4257 for reproducibility. We trained our model for 50 epochs using Adam with a learning rate of 5e-3 and do not perform dataset-specific tuning, performed early stopping on the dev set if the validation accuracy did not increase for ten consecutive epochs. To alleviate overfitting, we employ dropout before each MLP layer with a rate of 0.2, in BiLSTM encoder with rate 0.2. We save the punctuations in CVQD and DanMu datasets, because we think they may have semantic information in language, which is corroborated as in [46]. We build directed graphs by NetworkX and implement our model with PyTorch. For measure, we follow [25, 34] to use classification performance by the accuracy metric on the test set.

Comparison with strong baselines Due to the variance of the sampled subset in GAN module can be quite high, we repeat the classification process 10 times and report the averaged accuracy in Table 2. Compared with baseline methods on

Table 2. Test accuracy on test set of the 9 benchmark datasets.

Model	DanMu	CVQD	MR	SST1	SST2	SUBJ	TREC	CR	MPQA
LSTM	94.45	57.49	65.54	33.50	65.18	83.12	85.40	68.14	82.08
BiLSTM[47]	94.20	54.78	64.67	35.43	64.74	82.39	83.00	69.73	83.08
Att-BLSTM[48]	94.62	58.87	65.67	33.78	67.76	84.46	87.20	70.80	82.58
DPCNN[49]	93.75	56.94	66.23	36.11	68.81	87.53	87.20	75.75	79.43
CNN-rand[25]	93.37	60.07	68.86	32.90	71.06	86.32	87.40	74.34	80.50
TextRCNN[21]	94.17	56.28	70.11	32.49	77.59	89.73	86.20	78.94	82.52
fastText[23]	94.51	58.21	60.60	36.88	70.68	88.99	88.00	70.09	85.16
Transformer[44]	93.40	58.30	68.67	30.47	67.93	88.19	86.80	66.02	82.08
ERNIE[45]	95.81	65.42	-	-	-	-	-	-	-
PathWalk (Ours)	**96.68**	57.55	**73.17**	34.39	**77.70**	**89.86**	**88.80**	**79.29**	82.83

the test set, the proposed PathWalk model has achieved state-of-the-art results on six of nine benchmark datasets.

First, compared to LSTM- and CNN-based models in the top half of the table, e.g., BiLSTM, Att-BLSTM, and TextRCNN, PathWalk obtains new state-of-the-art results on six of nine benchmark datasets, including significantly pushing the test accuracy to 96.68% (approximately 3.3% point absolute improvement) on the DanMu dataset, MR accuracy to 73.17% (3% absolute improvement compared to the best TextRCNN 70.11%). PathWalk reaches state-of-the-art accuracy on SST2 77.70%, SUBJ 89.86%, TREC 88.80%, and CR 79.29%, improving over the existing best results.

We note that a shallow CNN-rand model performs better with randomly initialized word embeddings, especially on the CVQD and TREC; it outperforms more complex models like TextRCNN and Transformer. Also, PathWalk is competitive with state-of-the-art LSTM- and CNN-based methods on CVQD, SST1, and MPQA datasets. On the CVQD dataset, CNN-rand attains the best accuracy of 60.07%. Att-BLSTM gets 58.87% and outperforms BiLSTM by 4% absolute improvement, which shows that the attention mechanism provides more important information in text for classification.

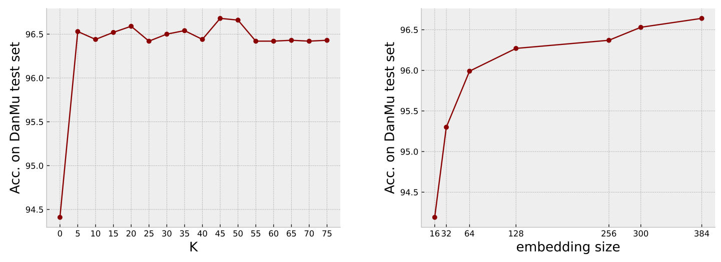
fastText model yields a higher accuracy of 85.16% than other methods on MPQA, and it has a 2.3% absolute improvement compared to the proposed PathWalk-BiLSTM model. It demonstrates that fastText uses the average of word embeddings as sentence representation is helpful for very short text classification.

Second, compared to the Transformer and fastText, the proposed PathWalk-BiLSTM performs the best and significantly outperforms them on most datasets. Especially on the DanMu dataset, our model obtains 3.2% and 2.17% respective accuracy improvement over the Transformer and fastText models. The proposed model obtains a 9% and 13% absolute accuracy improvement on CR.

Finally, the results show that the ERNIE model performs better on the two Chinese datasets, it achieves the best results on the CVQD dataset. ; this is likely because the pre-trained ERNIE model can learn a good representation for

Chinese entities and phrases, which can be fine-tuned for Chinese text classification. Especially on the DanMu dataset, our PathWalk model also achieves new state-of-the-art classification accuracy and performs much better, even if we only adopt the basic BiLSTM as a simple encoder.

We observed that simple models usually achieve comparable results on a specific dataset, such as fastText performs well on CVQD and MPQA. BiLSTM performs better than our complex PathWalk-BiLSTM model on MPQA. Although DanMu and CVQD are all Chinese short text datasets, the PathWalk-BiLSTM model attains new state-of-the-art on DanMu that even surpassed the most sophisticated Transformer models but performs worse on CVQD. Because CVQD tends to be phrase queries at search engines, search users often ignore the word order in their query. These findings emphasize that complex models can not produce consistent performance across all datasets.



(a) (b)

Fig. 4. Ablation studies on the DanMu dataset. (a) Number of neighbors K; (b) Dimension of node/edge embedding.

Ablation Study In this part, we conduct ablation studies to analyze our model further.

Effect of the parameter K: K is the number of randomly sampled neighbor nodes. To evaluate its sensitivity to classification accuracy, we grid search for K between 0 and 75, stepped by 5. In particular, K=0 is equivalent to the graph neighbors are not used for classification, which means the proposed model does not introduce neighbor nodes and edges from other texts. Figure 4(a) shows the influence of K on the DanMu test set. From the results, we can find that the model reaches its optimal accuracy at K=45. It shows that aggregating features from neighbours' nodes or edges could benefit classification accuracy. However, the performance does not grow with K because the variance of randomly sampled neighbors might be pretty high.

Dimension of graph embedding: We explore the effect of dimension of the graph embedding with various dimensions. The accuracy on the DanMu test set is shown in figure 4(b). It shows that the accuracy rises with increase of the embedding dimension. Although our model achieves higher accuracy when the dimension is set to 384, we adopt a suitable dimension of 300, a tradeoff between the accuracy and the model size.

Analysis of geometry-based features: In order to check the impact of geometry based features, we evaluate each of them by removing it from the full model. In this work, the full model incorporates neighbours' nodes as well as their edges in-between, we describe the full formulation for short text $\alpha^{(1)}$ as follows: $\boldsymbol{\alpha}_{1:l}^{(1)} = f(\{\boldsymbol{v}_i + \boldsymbol{e}_{i,i+1} + attention_{Subset_K}(\{\boldsymbol{v}_k + \boldsymbol{e}_{k,i}\}_{k=1}^{K})\}_{i=1}^{l})$, where f denotes a generic encoder function, here BiLSTM is employed as an encoder. Next, we remove neighbours' features in $\alpha^{(1)}$, the second formulation of $\alpha^{(2)}$ can be represented as: $\boldsymbol{\alpha}_{1:l}^{(2)} = f(\{\boldsymbol{v}_i + \boldsymbol{e}_{i,i+1})\}_{i=1}^{l})$. Finally, after removing in-text edges of α, our model would degenerate to the traditional method as prior work dose, the third formulation of $\alpha^{(3)}$ can be rewritten as: $\boldsymbol{\alpha}_{1:l}^{(3)} = f(\{\boldsymbol{v}_i\}_{i=1}^{l})$.

Table 3. Effectiveness of geometry-based features on DanMu test set.

Method	Full model $\boldsymbol{\alpha}^{(1)}$	- neighbors $\boldsymbol{\alpha}^{(2)}$	- in-text edges $\boldsymbol{\alpha}^{(3)}$
test accuracy	96.68%	94.41%	94.20%

Table 3 reports the accuracy on the test set. From the results, we can see that the full model of $\boldsymbol{\alpha}_{1:l}^{(1)}$ outperforms $\boldsymbol{\alpha}_{1:l}^{(2)}$ by 2.27% absolute improvement. On the one hand, it demonstrates that augmenting short text with geometry-based neighboring features could improve accuracy significantly. On the other hand, it indicates the effectiveness of the graph attention network (GAN) for selecting important neighboring features. Compared to $\boldsymbol{\alpha}_{1:l}^{(3)}$, $\boldsymbol{\alpha}_{1:l}^{(1)}$ achieves 2.48% absolute improvement, $\boldsymbol{\alpha}_{1:l}^{(2)}$ gets 0.21% absolute improvement, which show directed edges in the text sequence could further increase the features for short text. Overall, it corroborates that internal features within dataset can bring contextual prompts to guide short text classification

5 Conclusion

The paper utilizes within-dataset topological features to improve short text classification, augmenting short text with the internal knowledge from the dataset oneself. We start by transferring the dataset into a single directed graph and regard a short text as a path. Then, we enhance the representation of short text by aggregating topological features from neighboring texts, which alleviates the problem of feature sparsity. Extensive experiments on nine benchmark datasets show that the improvement in classification accuracy comes from topological features, i.e., neighbors nodes and edges. In addition, we build 2 new short-text-specific datasets as a helpful reference for classification. This work may provide a new perspective to data enhancement via introducing knowledge, which can come not only from external resources but also from the dataset oneself.

Acknowledgements

We thank the reviewers for their comments and suggestions. This paper is supported by NSFC (No. 61906018), Huawei Noah's Ark Lab, MoE-CMCC "Artificial Intelligence" Project (No. MCM20190701), Beijing Natural Science Foundation (Grant No. 4204100) and BUPT Excellent Ph.D. Students Foundation (No. CX2020309).

References

1. Binxia Xu, Siyuan Qiu, Jie Zhang, Yafang Wang, Xiaoyu Shen, and Gerard de Melo. Data augmentation for multiclass utterance classification – a systematic study. In *COLING*, pages 5494–5506, 2020.
2. Zongcheng Ji, Zhengdong Lu, and Hang Li. An information retrieval approach to short text conversation. *arXiv:1408.6988*, 2014.
3. Zhongyuan Wang and Haixun Wang. Understanding short texts. In *ACL*, 2016.
4. Wei Pang and Ruixue Duan. History-aware expansion and fuzzy for query reformulation. In *CICAI*, volume 13070, pages 227–238, 2021.
5. S. Djamasbi, A. Hall-Phillips, Z. Liu, W. Li, and J. Bian. Social viewing, bullet screen, user experience: A first look. In *International Conference on System Sciences*, pages 648–657, 2016.
6. Ming He, Yong Ge, Enhong Chen, Qi Liu, and Xuesong Wang. Exploring the emerging type of comment for online videos: Danmu. *ACM Transaction on the web*, 12(1), 2018.
7. Jiancheng Wang, Jingjing Wang, and et al. Sentiment classification in customer service dialogue with topic-aware multi-task learning. In *AAAI*, pages 9177–9184, 2020.
8. Homa B. Hashemi, Amir Asiaee, and Reiner Kraft. Query intent detection using convolutional neural networks. In *WSDM*, 2016.
9. Xiaowei Liu, Weiwei Guo, and et al. Deep search query intent understanding. In *CIKM*, 2020.
10. Jindong Chen, Yizhou Hu, and et al. Deep short text classification with knowledge powered attention. In *AAAI*, pages 6252–6259, 2019.
11. Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. Topic memory networks for short text classification. In *EMNLP*, pages 3120–3131, 2018.
12. Linmei Hu, Tianchi Yang, Chuan Shi, Houye Ji1, and Xiaoli Li. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP*, pages 4821–4830, 2019.
13. Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *TNNLS*, 2020.
14. Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017.
15. Chenglong Ma, Weiqun Xu, Peijia Li, and Yonghong Yan. Distributional representations of words for short text classification. In *NAACL-HLT*, pages 33–38, 2015.

16. Chenliang Li, Haoran Wang, and et al. Topic modeling for short texts with auxiliary word embeddings. In *SIGIR*, pages 165–174, 2016.
17. Aixin Sun. Short text classification using very few words. In *SIGIR*, pages 1145–1146, 2012.
18. Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, 2011.
19. Xia Hu, Nan Sun, Chao Zhang, and Tat-Seng Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *CIKM*, pages 919–928, 2009.
20. Rima Túrker, Lei Zhang, and et al. Knowledge-based short text categorization using entity and category embedding. In *Hitzler P. et al. (eds) The Semantic Web*, 2019.
21. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273, 2015.
22. Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. Bag-of-embeddings for text classification. In *IJCAI*, 2016.
23. Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *EACL*, pages 427–431, 2017.
24. N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 1746–1751, 2014.
25. Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.
26. Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. In *NAACL*, 2016.
27. Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *IJCAI*, pages 2873–2879, 2016.
28. Sujith Ravi and Zornitsa Kozareva. Self-governing neural networks for on-device short text classification. In *EMNLP*, pages 804–810, 2018.
29. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 13898–13906, 2019.
30. Yinhan Liu, Myle Ott, and et al. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.
31. Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS*, 2015.
32. Shervin Minaee, Nal Kalchbrenner, and et al. Deep learning based text classification: A comprehensive review. *arXiv:2004.03705*, 2020.
33. Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Rethinking complex neural network architectures for document classification. In *NAACL*, pages 4046–4051, 2019.
34. Kaize Ding, Jianling Wang, and Jundong Li. Be more with less: Hypergraph attention networks for inductive text classification. In *EMNLP*, pages 4927–4936, 2020.
35. Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, 2015.
36. Liang Yao, Chengsheng Mao, and Yuan Luoi. Graph convolutional networks for text classification. In *AAAI*, 2019.
37. Wei Pang. Short text classification via term graph. *arXiv.2001.10338*, Jan 2020.
38. Bo Pang and Lillian Lee. Rt-polarity data v1.0, July 2005.
39. Xin Li and Dan Roth. Learning question classifiers. In *COLING*, 2002.

40. M. Hu and B. Liu. Mining and summarizing customer reviews. In *SIGKDD*, 2004.
41. Lingjia Deng and Janyce Wiebe. Mpqa 3.0: Entity/event-level sentiment corpus. In *NAACL-HLT*, 2015.
42. Richard Socher, Alex Perelygin, and et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, 2013.
43. A. Joulin, E. Grave, P. Bojanowski, and et al. Fasttext. zip: Compressing text classification models. *arXiv:1612.03651*, 2016.
44. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Lukasz Kaiser. Attention is all you need. In *NeurIPS*, 2017.
45. Zhengyan Zhang, Xu Han, and et al. Ernie: Enhanced language representation with informative entities. In *ACL*, pages 1441–1451, 2017.
46. Mansooreh Karami, Ahmadreza Mosallanezhad, and et al. "let's eat grandma": Does punctuation matter in sentence representation. In *ECMLPKDD*, 2022.
47. Shu Zhang, Dequan Zheng, and et al. Bidirectional long short-term memory networks for relation classification. In *29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78, 2015.
48. Peng Zhou, Wei Shi, and et al. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL Short Paper*, pages 207–212, 2016.
49. Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *ACL*, pages 562–570, 2017.