

Word Relation-based Graph Neural Network for Short Text Similarity Measurement

Hechuan Shen

Sichuan University

Xu Wang

Sichuan University

Xuedong Li

Sichuan University

Dezhong Peng

Sichuan University

Haixiao Huang (✉ huanghaixiao.cd@gmail.com)

Sichuan Provincial Commission of Politics and Law

Research Article

Keywords:

Posted Date: August 1st, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1880638/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Word Relation-based Graph Neural Network for Short Text Similarity Measurement

Hechuan Shen¹, Xu Wang¹, Xuedong Li¹, Dezhong Peng¹, and Haixiao Huang^{2,*}

¹Sichuan University, Chengdu, 610065, China

²Sichuan Provincial Commission of Politics and Law, Chengdu, 61000, China

*huanghaixiao.cd@gmail.com

ABSTRACT

Short text similarity measurement is an important task, which is broadly applied to many fields. Due to the lack of contextual information, it is not easy to accurately compare the similarity between short texts. The relations between words are used to enhance the information. Since Graph Neural Networks (GNN) perform well in handling data with complex relations, the potential of GNN in this task has been explored recently. Unfortunately, the relationship between words is not adequately acquired by existing GNN-based methods. Furthermore, they ignore the deep relationships between words and do not take them into account. Typically, these methods use a fixed graph structure for a sentence and do not update the graph structure in the network. Based on the situation, we explore utilizing the deeper relations between words to improve the applicability of GNN for this task. In this letter, we present a framework called Word Relation-based Graph Neural Network (WRGNN). WRGNN can automatically discover and construct relations between words and reconstruct those relations in the processing. Two benchmark datasets are evaluated extensively, with WRGNN obtaining an average of 71.2 % on STS-B and 82.9 % on SICK.

1 Introduction

Recently, short texts are being used more frequently in our daily lives. Tweets, comments, and messages are all presented in the form of short text. Short text similarity measurement is a fundamental task of text classification, text sentiment analysis, and other text applications. However, due to the lack of contextual information, traditional methods fail to yield good results in measuring the similarity of short text. To solve this problem, some methods have been proposed to increase the amount of contextual information. These methods can be divided into knowledge-based and sequence-based. The knowledge-based approaches (^{1,2,3,4}) use external information such as corpus, semantic web, and pre-trained models to discover and exploit the relations between words. These methods are less effective while external knowledge is different from the domain of the task. Sequence-based approaches (^{5,6,7,8}) employ the arrangement of words to improve the performance of the model, and in addition, attention mechanism is used to associate distant words. By these methods, the relations between the words that are far away from each other are reduced. However, we believe that simply treating sentences as sequences of tokens is not a good representation of the knowledge embedded in the sentences.

Graph Neural Networks (GNN)⁹ has been tried to measure the similarity of short texts. Although the graph structure is well suited for data containing complex relationships, existing GNN-based methods have not achieved satisfactory results. We believe that these methods do not adequately capture and exploit the relationships between words in a sentence. These methods use co-occurrence, syntactic, and other relation parsing tools to construct the graph structure of a sentence. This kind of graph structure constructs relationships between words only from a single perspective. However, the relationships between words are complex and multiple, and errors in the construction of graphs are harmful to the performance of GNN. Additionally, existing GNN-based approaches utilize a single graph filter for convolution. Reusing a single graph filter cannot fully exploit the complex relationships between words. Moreover, some approaches tend to build graphs only based on the original word embedding, without updating the graph structure in the network. As a result of graph convolution, each word embedding collects information about its neighbors. Due to the fact that the embedding contains more complex information than the original one, the relationships between original words are no longer applicable to the current embedding.

To address the above issues, we propose a GNN-based end-to-end structure for the measurement of the similarity of short texts called Word Relation-based Graph Neural Network (WRGNN). We design a relation-seeking item, through which the WRGNN is able to discover relations between words adaptively. The relation-seeking item can calculate the likelihood of relations between words based on the embedding of the words. Unlike the single-faceted perspective of relationships, relations learned through relationship-seeking item are more comprehensive. Considering the relationships between words change in response to the information contained in word embeddings, we develop a relation progressive learning mechanism based on the

relationship-seeking item. After each change in the embedding, the relation progressive learning mechanism re-establishes the relationship between words, allowing the graph filters to take advantage of deeper relationships. Since the relationship between words obtained by WRGNN is complex and multiperspective, a single graph filter can not fully exploit this relationship, so we propose a multi-dimensional graph filter based on Graph Attention Network (GAT)¹⁰ and Graph Sample and aggregate (GraphSAGE)¹¹. Using the multi-dimensional graph filter, the information from GAT and GraphSAGE is fused, which enhances WRGNN's ability to leverage the relationship between words. Additionally, since the structure of the siamese network¹² is suitable for similarity measurement, we adopt it as the basic framework.

The main contributions of this letter can be summarized as:

- We design an end-to-end siamese network based on GNN called WRGNN for the measurement of short text similarity. We test the performance of WRGNN on SICK and STS-B. Experimental results validate the effectiveness of WRGNN.
- We propose a relation progressive learning mechanism based on a relationship-seeking item which can learn the relationship between words from different perspectives adaptively. The mechanism can reconstruct the word-to-word relationships as the words converge in the graph to make them more adaptive to the updated embedding.
- We design a multi-dimensional graph filter to update node information from different angles, which can make better use of the relationships between words.

2 Related Work

The methods of short text similarity can be divided into three categories: corpus-based methods, knowledge-based methods, and sequence-based methods.

Corpus-based: The corpus uses regularized texts from documents, papers, and books, which contain a large amount of semantic information as well as words association information. Corpus-based approaches usually use semantic information and lexical information in the corpus to derive the correlations between words in short texts. For example,¹ uses highly organized texts from Wikipedia as the corpus. The authors represent the input texts as high-dimensional weighted vectors via a semantic interpreter which is constructed from corpus. Then the relevance between the texts is calculated by comparing the vectors of them via distance metrics such as cosine coefficient. However, the information and words relations contained in the corpus are fixed and not well adapted to downstream tasks. Additionally, the corpus-based methods do not take the near-synonyms and the semantic information between words into account, so the performance of these methods is limited.

Knowledge-based: Knowledge-based approaches usually maintain a bag of words or semantic network constructed based on external knowledge. In the bag of words or semantic network, words are associated through co-occurrence, proxemics, and other methods. In², the authors use word co-occurrence information from an external corpus to convert Bag-of-Words into the form of Bag-of-Word-Pairs in which each element consists of two co-occurring words, and then calculate the similarity between the short texts based on the obtained word pairs. In³, a probabilistic topic model is constructed by external knowledge based on the assumption that document is composed of multiple topics. The authors treat each topic as a probability distribution of words. Similarity is calculated based on the probability distribution of each word in the sentence pair across topics. In¹³, the authors use Masked-Language Modeling (MLM)¹⁴ for pre-training and a deep bi-directional Transformer to build the BERT model based on an external corpus. BERT is able to learn rich word relations, syntactic features, and semantic features through the corpus. In calculating the similarity, the BERT is first fine-tuned on the training data, and then the similarity score of the sentence pair is obtained by applying the distance calculation method to the sentence embedding of each sentence. Although knowledge-based approaches can achieve good results in some cases. However, these methods are highly dependent on external knowledge. These methods are less effective if the external knowledge is different from the domain of the task or if the task contains words that are not available in the knowledge base.

Sequence-based: Sequence-based models treat text as a collection of words combined in a sequence. In this way, Long Short Term Memory (LSTM)⁵, Recurrent Neural Networks (RNN)⁶, and other methods that are suitable for processing temporal problems can be used for NLP tasks. The sequence-based methods are able to learn the words embedding based on the sequential of the words in sentences, thus the related information between words and semantic information are contained in each word embedding. In⁷, the authors combine two LSTMs, one of which accepts sequential inputs and the other accepts inverse-order inputs of the original inputs. Different from LSTM, BiLSTM acquire information from front and back directions. In this way, BiLSTM performs better than LSTM by the more adequate contextual information. In⁸, the authors used co-occurrence and semantic parsing to construct a tree structure of the words in the sentences. Tree-LSTM is able to selectively obtain information from the child nodes via the constructed tree structure. By this way, Tree-LSTM can enhance the expression of key information in sentences. However, the sequence-based methods can only directly utilize the relationship between adjacent words in the text. The information utilization between the words that are far away from each other in the text is reduced.

3 Methods

In this section, we present the specific framework and the details of WRGNN.

3.1 Overview

The framework of our model is illustrated in Figure 1. As shown in the figure, the WRGNN consists of four modules: Word Embedding Layer, Relation Deeping Layer, Pooling Layer, and Similarity Calculating Layer. G_n represents the graph build based on sentence S_n . As for a sentence S_n , we generate the vectors of node embedding via the Word Embedding Layer. Based on the vectors, Relation Deeping Layer creates the word relation graph of a sentence and then learns a new embedding of nodes based on the graph. To implement the relation progressive learning mechanism, we use two structurally identical Relation Deeping Layers that do not share any parameters. Graphs with different numbers of nodes make it difficult to judge the similarity. To solve this problem, we use the Graph Pooling Layer to convert the graph containing different nodes into a single vector. In order to extract the features using the same way as possible, we use the siamese structure with shared parameters to learn the short texts in a pair separately. Finally, the similarity of the two sentences is obtained based on the vectors of them via the Similarity Calculating Layer.

3.2 Details of WRGNN

In this section, we describe the components of WRGNN and how it is organized. We introduce the implementation details of Word Embedding Layer, Relation Deeping Layer, Graph Pooling Layer, Similarity Calculating Layer and some necessary mathematical derivations.

3.2.1 Word Embedding Layer

As for the Word Embedding Layer, we first segment sentence S_n into a collection of tokens $\{t_1, t_2, \dots, t_n\}$. Then we collect the tokens of all the sentences in the training data to generate the vocabulary. Based on the vocabulary, we use the Global Vectors for Word Representation (Glove)¹⁵ to generate the embedding vectors for each token in the sentence.

3.2.2 Relation Deeping Layer

In each Relation Deeping Layer, relationships between words are sought and graphs constructed based on a sentence are filtered. The detailed structure of the Relation Deeping Layer is shown in Figure 1. As seen, we propose the relationship-seeking item to learn the relationships between words based on their embedding vectors. The relationship-seeking item constructs a word relation graph for each sentence based on the vectors. As each Relation Deeping Layer contains a relationship-seeking item, WRGNN reconstructs the relationship of words before the filtering of each graph. By reusing the relationship-seeking item, we implement the relation progressive learning mechanism.

According to¹⁰, we find that compared to other graph filters, GAT assigns different weights to nodes in the same neighborhood by attention mechanism. In this way, GAT is able to filter the neighboring nodes and thus improve the representation capability in the model. However, according to the experimental results in¹⁶, it can be seen that GAT lacks the learning of the graph structure. Graph structure is an important feature of graphs and plays an important role in the step of learning representations of nodes in the graph. Based on the consideration, we add SAGE to the adoption of GAT. SAGE learns the features of each node by sampling neighbors. By this method SAGE is able to achieve an inductive representation of the local structure instead of a single local structure. In addition, GAT can compensate for the shortcomings of SAGE in terms of loss of important local information of some nodes caused by the sampling mechanism and instability of node features. As a result of this, we utilize an integration item to aggregate information from GAT and SAGE in the multi-dimensional graph filter.

The first part of the Relation Deeping Layer is to convert the node embedding into graph structure data via a relationship-seeking item. We define $G(v, e)$ as an undirected graph and e represents the edges of graph G , v represents the node embedding of graph G and $\|v\| = n$. Building a graph means building the node embedding v and edges e in the graph. We take the embedding vectors $\{v_1, v_2, \dots, v_n\}$ as the node embedding of G . The edges of the graph are determined by the relationships between the word embedding vectors. In order to discover the relationship between words, we adopt the learning-based method in¹⁷.

Firstly, we compute the relationship score $R_{i,j}$ between node i and j by using a learnable vector W_s . The parameters of W_s are updated as the model is trained. The calculation process is formulated as follows:

$$R_{i,j} = \text{ReLU}(W_s \odot v_j) \times \text{ReLU}(W_s \odot v_i)^T \quad (1)$$

where $v_i \in R^m$ and $v_j \in R^m$ represent the embedding vectors of node i and j , respectively. $W_s \in R^m$ is a weight vector to weight each dimension of the vectors, m is the dimension of the vectors in node embedding and $R_{i,j}$ is the relation score between node i and node j .

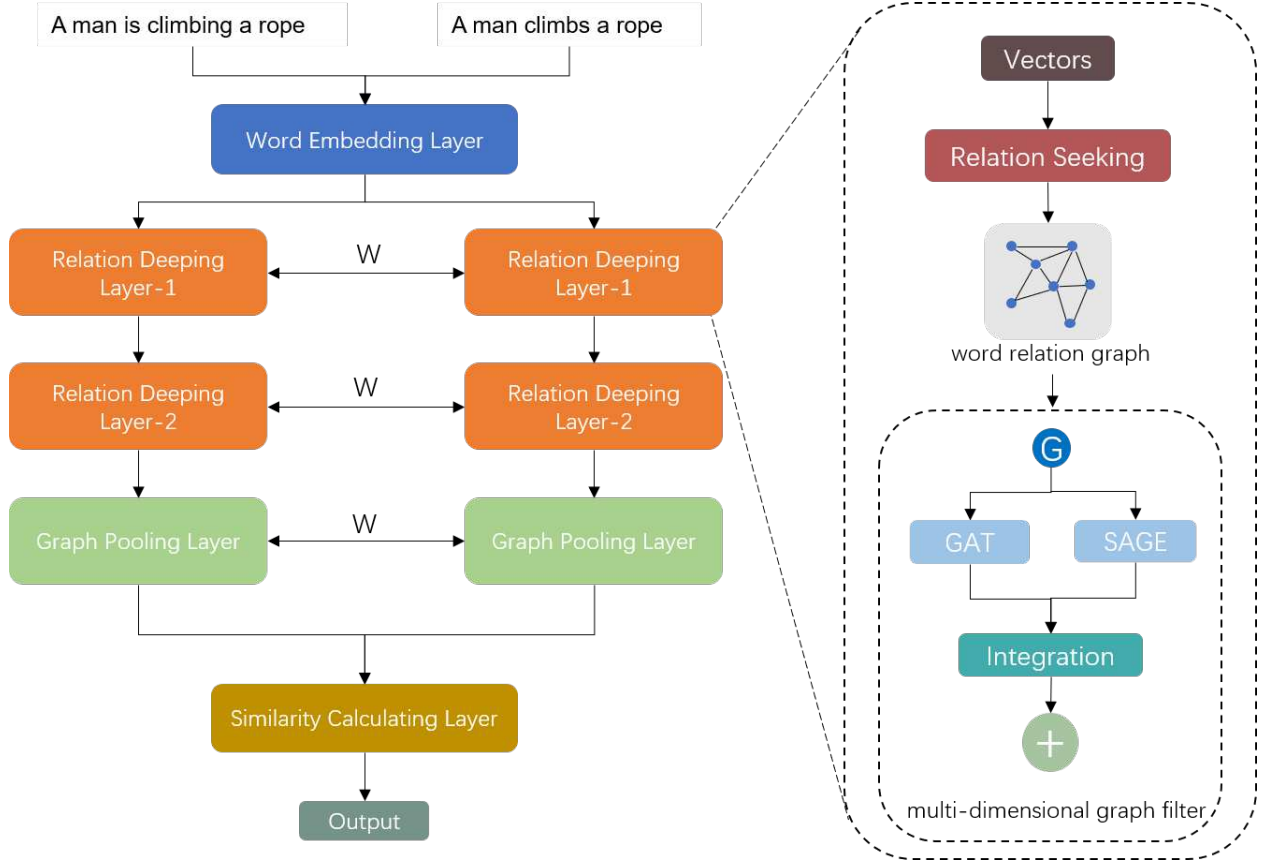


Figure 1. The figure shows the overall framework of WRGNN. Among them, Relation Deeping Layer-1 and Relation Deeping Layer-2 have the same structure but do not share parameters. Each Relation Deeping Layer is consist of a Relation seeking item and a multi-dimensional graph filter. Identical parts of the siamese network share parameters. W denotes the parameter they share. Vectors are the embedding of all tokens in a single sentence.

Based on the relation score, we set up the adjacency matrix A of G that demonstrates the edges between nodes. We set a threshold α for the relation score. If $R_{i,j}$ is bigger than α , we set $A_{i,j}$ as 1 which means there is an edge between node i and node j , otherwise, we set $A_{i,j}$ as 0. The process is formulated as follows:

$$A_{i,j} = \begin{cases} 1 & \text{if } R_{i,j} \geq \alpha \\ 0 & \text{if } R_{i,j} < \alpha \end{cases} \quad (2)$$

where $A_{i,j}$ is the element of matrix $A \in R^{n \times n}$, n is the number of vectors in node embedding.

After obtaining the complete graph G , we use the multi-dimensional graph filter to learn the node embedding based on the structure of G . In the multi-dimensional graph filter, we use F_{gat} and F_{sage} to aggregate the embedding of the neighbors. F_{gat} is a graph filter build based on multi-headed Graph Attention Network (GAT)¹⁰ and F_{sage} is the other graph filter build based on Graph Sample and aggreGatE (GraphSAGE)¹¹. The relationship between words can be utilized from different perspectives via F_{gat} and F_{sage} . Thus, WRGNN is able to take the most advantage of the relationships between words. Since F_{gat} has three head, we can get four vectors a_i^1, a_i^2, a_i^3, s_i for each node i via the multi-dimensional graph filter, which is formulated as:

$$\begin{aligned} a_i^1, a_i^2, a_i^3 &= F_{gat}(G, i) \\ s_i &= F_{sage}(G, i) \end{aligned} \quad (3)$$

where $a_i^1 \in R^m, a_i^2 \in R^m$ and $a_i^3 \in R^m$ are the vectors of node i obtained by F_{gat} based on graph G , $s_i \in R^m$ is obtained by F_{sage} in the same way.

After this, we use the integration item to fuse the obtained vectors to form a new word embedding. As for the integration item, we use a learning-based approach that allows the integration item to discover and use key information in the vectors. Take node i as an example. We first set up a learnable weight matrix $M_w \in R^m$ for computing the weights of each vector in node i .

Then we concatenate the vectors of node i into a matrix $I \in R^{4 \times m}$. Furthermore, we adopt *Sigmoid* to strengthen the expressive of the item. At the end of the integration item, we multiply each dimension of I by the corresponding weight and accumulate it to get a new vector $v'_i \in R^m$ which is the new embedding vector of node i . This is formulated as follows:

$$\begin{aligned} I_i &= s_i || a_i^1 || a_i^2 || a_i^3 \\ v'_i &= \sum (Sigmoid(IM_w^T) \otimes I) \end{aligned} \quad (4)$$

where $||$ denotes the concatenate operation. $M_w \in R^m$ is a learnable weight vector, v'_i is the final embedding of node i .

3.2.3 Graph Pooling Layer

Since the number of tokens contained in sentences may be different, the two graphs generated by the network may contain different numbers of nodes. We propose Graph Pooling Layer to align the graphs. Take graph G_n as an example. We use the weight and sum method⁹ to calculate a weight for each node embedding in the graph by using the weight vector $W_p \in R^m$, and then perform the weighted summation G_n as the graph embedding. The process is formulated as follows:

$$P_n = \sum_{i \in G_n(v)} W_p v_i^T v_i \quad (5)$$

where $v_i \in R^m$ is the embedding vector of node i , $W_p \in R^m$ is the learnable weight vector, $P_n \in R^m$ is the graph embedding vector, $G_n(v)$ is the nodes set of G_n .

3.2.4 Similarity Calculating Layer

At the end of the model, we use the Similarity Calculating Layer to calculate the similarity between the pair of sentences based on the graph embedding vectors. The similarities between sentences can be reflected by the degree of correlation between the vectors of them. We adopt the Pearson product-moment correlation coefficient¹⁸ since it can be used to measure the degree of correlation between two variables. Take the sentence pair x as an example which is consist of S_n and S_m , we use the absolute value of Pearson product-moment correlation coefficient of P_n and P_m to represent the similarity of S_n and S_m . The result value is between 0 and 1, which is formulated as follows:

$$Sim_x = \frac{|E[(P_n - \mu_{P_n})(P_m - \mu_{P_m})]|}{\sqrt{\sum_{n=1}^m (P_n - \mu_{P_n})^2} \sqrt{\sum_{n=1}^m (P_m - \mu_{P_m})^2}} \quad (6)$$

where P_n, P_m are the graph embedding vectors of the pair of sentences S_n and S_m , Sim_x is the similarity of sentence S_n and S_m , P_n^n represents the n -th element in P_n .

3.3 Loss Definition

The error value can be positive or negative in the task. Mean Squared Error Loss (MSELoss) eliminates the effect of positive and negative values by squaring the error values. In addition, the process of taking squares can penalize more for larger errors. So, we use MSELoss as the loss function, the process is formulated as:

$$loss = \frac{1}{M} \sum_{x=0}^M (Sim_x - \frac{T_x}{5})^2 \quad (7)$$

where M indicates the size of a batch, x denotes a sentence pair in the dataset. Sim_x denotes the calculated result based on the sentences in x . T_x denotes the true result. Since the value of T_x in the datasets is between 0 and 5, we shrink it by a factor of 5 for normalization purposes.

4 Experiment

4.1 Dataset

The experiment is carried out on two public datasets, Sentences Involving Compositional Knowledge (SICK)¹⁹ and Semantic Textual Similarity benchmark (STS-B)²⁰. The task for both datasets is semantic relatedness. Specifically, for a given sentence pair, the semantic relatedness task aims to predict human ratings of how semantically related the two sentences are.

SICK: SICK dataset is a dataset for compositional distributional semantics. It consists of 10,000 English sentence pairs that contain rich words, phrases, syntactic, and semantic relations. Each pair of sentences is assigned a relatedness score according to their similarity. The relatedness score ranges from 0 to 5 where 0 means there is no correlation between the two sentences, and 5 means there is a strong correlation between the two sentences. There are 4439 pairs of training data, 4906 pairs of test data, and 495 pairs of trial data in the SICK data set. The sentence pairs are generated from two existing paraphrase data sets:

Table 1. Three example pairs of data in SICK and STS-B.

score	Sentence1	Sentence2
2.167	A man is playing soccer.	A man is playing flute.
0.600	A man is crying.	A woman is dancing.
1.333	The boy is playing the piano.	A band is playing on stage.

one is 8k imageFlickr built and the other is SemEval-2012’s semantic text similarity video description data set. SICK can be obtained from <http://alt.qcri.org/semeval2014/task1/>.

STS-B: STS-B contains sentences obtained from image titles, news headlines, and user forums. STS-B includes a selection of English datasets used in STS tasks organized in the context of SemEval from 2012 to 2017. Similar to SICK, the relatedness score ranges from 0 to 5 with 0 meaning the sentences are completely irrelevant and 5 meaning they are entirely relevant. STS-B contains 8,628 sentence pairs, of which 5,749 are training data, 1,379 are test data, and 1,500 are development data. STS-B can be obtained from <http://alt.qcri.org/semeval2017/index.php?id=tasks>.

In order to present the data set more clearly, we list a portion of the data as an example. Table 1 shows the form of the data.

4.2 Setup

The proposed WRGNN is implemented in Pytorch on 2 GTX-1080ti cards. In this experiment, we adopt the Adagrad algorithm²¹ to optimize the parameters of the model. Adagrad is not sensitive to the selection of learning rate and converges fast. Adagrad adaptively assigns algorithms with different learning rates to each parameter. We set the learning rate at 0.001 and dropout 0.2 in WRGNN.

4.3 Evaluation Metric

In⁸, the authors adopt Pearson’s correlation coefficient¹⁸ and Spearman’s correlation coefficient²² as the evaluation metrics. We adopt the two correlation coefficients between the real similarity array and the predicted similarity array to evaluate the accuracy of the final prediction. In addition, for a more complete picture of the performance of each model, we also calculate the averages of them. Pearson’s correlation coefficient¹⁸ and Spearman’s correlation coefficient²² are used to measure the proximity between two variables.

Set variables X and Y as an example. The outputs of the correlation coefficient are between $(-1, 1)$. If Y tends to increase when X increases, the output is positive. Additionally, if Y tends to decrease when X increases, the output is negative. If the output is zero, it indicates that there is no correlation between the variables.

The Pearson’s correlation coefficient between X and Y is formulated as:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_x \sigma_y} \quad (8)$$

The Spearman’s correlation coefficient between X and Y is formulated as:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (9)$$

4.4 Baselines

We compare our model with Non-GNN-based and GNN-based methods. The Non-GNN-based methods include LSTM⁵, BiLSTM⁷, Tree-LSTM⁸, MP-CNN²³, Tiny-BERT¹³. GNN-based methods include TextSimGNN²⁴, GCSTS²⁵. In addition, we design some variants of WRGNN for the ablation study to demonstrate the effect of the relation progressive learning mechanism, the relationship-seeking item, and the multi-dimensional graph filter.

LSTM: LSTM adds the gating mechanism on RNN which controls the transmission state. Compared to RNN, LSTM can achieve better results in text tasks.

BiLSTM: The BiLSTM consists of two LSTM with different directions: one in the order of the text, the other in the opposite order of the text. In this way BiLSTM can obtain more information about the relationship between words, thus improving the model performance.

Tree-LSTM: Tree-LSTM proposes a variant of LSTM. The authors use grammatical relations to model sentences into dependency trees. LSTM is adopted to learn the tree structure data. Here we use the Constituency Tree-LSTM as the baseline.

MP-CNN: MP-CNN proposes a model for comparing sentences that use a multiplicity of perspectives. It performs better than the traditional CNN²⁶ on the task of short text similarity.

Table 2. This is the results of experiment

	Method	STS-B			SICK		
		Pearson	Spearman	Avg	Pearson	Spearman	Avg
Non-GNN-based methods	LSTM	0.687	0.678	0.683	0.824	0.772	0.798
	BiLSTM	0.696	0.679	0.688	0.834	0.782	0.808
	Tree-LSTM	-	-	-	0.858	0.796	0.827
	MP-CNN	-	-	-	0.847	0.785	0.816
GNN-based methods	TextSimGNN	0.503	-	0.503	0.754	-	0.754
	GCSTS	-	0.693	0.693	-	-	-
	WRGNN _{sage}	0.694	0.682	0.688	0.796	0.744	0.770
	WRGNN _{gat}	0.708	0.700	0.704	0.848	0.800	0.824
	WRGNN _{out}	0.670	0.658	0.664	0.832	0.784	0.808
	WRGNN	0.719	0.705	0.712	0.856	0.802	0.829

Tiny-BERT: Tiny-BERT is a model that uses a large amount of corpus knowledge for pre-training. Here we take the characterization of each sentence by BERT and then use the cosine function to calculate its similarity. In order to control the number of parameters at the same level, we use tiny-BERT with 4layer-312dim from paper⁴.

TextSimGNN: TextSimGNN is a graph neural network method recently proposed to measure semantic similarity. The authors model the text as a graph structure based on grammatical relations. The Simgnn²⁷ is used to calculate the similarity between graphs for representing the similarity between text. Here we only use the word-based network.

GCSTS: GCSTS calculates the similarity between short texts based on graph convolutional network (GCN). The authors use the dependency tree to construct the graph structure of the sentence. The authors modified the GCN to make it more suitable for the task of short text similarity.

5 Results and Analysis

5.1 Experimental Results

The experimental results are shown in Table 2. As shown in the results, WRGNN achieves the best results in five of the six metrics. For instance, when comparing the results of WRGNN and other GNN-based baselines, it can be seen that WRGNN achieves the best performance. This implies that WRGNN is capable of discovering and utilizing the relationship between words. By discovering and exploiting the relationships between words, WRGNN is better suited to the task of short text similarity. Additionally, it can be seen that GCSTS and TextSimGNN perform poor than Non-GNN-based methods on the measurement of the similarity between short texts. While, WRGNN improves the performance of GNN-based methods on this task and outperforms Non-GNN-based methods in most metrics.

5.2 Ablation Study

In order to analyze the contribution of different parts of WRGNN, we incrementally evaluate each of them on STS-B and SICK by constructing 3 variants of our method. WRGNN_{gat} and WRGNN_{sage} are the models that replace the multi-dimensional graph filter in WRGNN with GAT and GraphSAGE. In this way, the effectiveness of the multi-dimensional graph filter can be verified. WRGNN_{out} is the model that removes the relation progressive learning mechanism in WRGNN. In WRGNN_{out}, the relationships between words are only learned in the first Relation Deeping Layer, and the relationships are not altered in subsequent steps. The results are shown in Table 2.

Contribution of multi-dimensional graph filter: As shown in the results, WRGNN outperforms WRGNN_{gat} by 0.8% and 0.4%, WRGNN_{sage} by 2.4% and 5.9% on STS-B and SICK. By the comparison, we can see that different from a single graph filter, the multi-dimensional graph filter is able to provide a more efficient and comprehensive view of the relationships between words.

Contribution of relation progressive learning mechanism: By removing the graph progressive learning mechanism in WRGNN, WRGNN_{out} underperforms WRGNN by 4.8% and 2.1% on dataset STS-B and SICK. By the comparison, it can be seen that the relation progressive learning mechanism enhances the information acquisition ability of the network. The relation progressive learning mechanism can more fully discover the deep relationships between words, which are particularly important in this task.

6 Discussion

We present WRGNN, a GNN-based method for the measurement of short texts that outperforms other GNN-based methods across several widely used datasets. In WRGNN we use the relationship between words as a complement to the information in the short text. Through our evaluations, we demonstrate that WRGNN is able to learn better representation for short text. According to the experiment, the relationship-seeking item is able to discover more complex and multiple relationships between words than the traditional methods. The relation progressive learning mechanism based on relationship-seeking item can discover deeper relationships between words for improving the performance of WRGNN. Furthermore, the multi-dimensional graph filter can make more complete use of the relationships between words to enhance the graph structure for sentence representation.

7 Conflict of Interest

The authors declare that they have no conflicts of interest regarding the publication of this manuscript.

8 Data availability statement

The datasets generated during or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Gabrilovich, E., Markovitch, S. *et al.* Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, vol. 7, 1606–1611 (2007).
2. Yang, S., Huang, G. & Ofoghi, B. Short text similarity measurement using context from bag of word pairs and word co-occurrence. In *International Conference on Data Service*, 221–231 (Springer, 2019).
3. Quan, X., Liu, G., Lu, Z., Ni, X. & Wenxin, L. Short text similarity based on probabilistic topics. *Knowl. information systems* **25**, 473–491 (2010).
4. Jiao, X. *et al.* Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* (2019).
5. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).
6. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).
7. Graves, A., Jaitly, N. & Mohamed, A.-r. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, 273–278 (IEEE, 2013).
8. Tai, K. S., Socher, R. & Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).
9. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
10. Veličković, P. *et al.* Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
11. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035 (2017).
12. Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Icml* (2010).
13. Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).
14. Taylor, W. L. “cloze procedure”: A new tool for measuring readability. *Journalism quarterly* **30**, 415–433 (1953).
15. Pennington, J., Socher, R. & Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543 (2014).
16. Zhang, K., Zhu, Y., Wang, J. & Zhang, J. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations* (2019).
17. Chen, Y., Wu, L. & Zaki, M. J. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942* (2019).
18. Benesty, J., Chen, J., Huang, Y. & Cohen, I. Pearson correlation coefficient. In *Noise reduction in speech processing*, 1–4 (Springer, 2009).

19. Marelli, M. *et al.* A sick cure for the evaluation of compositional distributional semantic models. In *Lrec*, 216–223 (Reykjavik, 2014).
20. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I. & Specia, L. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055* (2017).
21. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. machine learning research* **12** (2011).
22. Zar, J. H. Significance testing of the spearman rank correlation coefficient. *J. Am. Stat. Assoc.* **67**, 578–580 (1972).
23. He, H., Gimpel, K. & Lin, J. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1576–1586 (2015).
24. Zhou, K., Xu, K., Sun, T. & Zhang, Y. Sentence modeling via graph construction and graph neural networks for semantic textual similarity. In *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 413–418 (IEEE, 2020).
25. Sun, X., Wu, S. & Liu, Y. Gcns-based context-aware short text similarity model. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 1329–1335 (IEEE, 2021).
26. LeCun, Y. *et al.* Handwritten digit recognition with a back-propagation network. *Adv. neural information processing systems* **2** (1989).
27. Bai, Y. *et al.* Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 384–392 (2019).