




GIMM: A graph convolutional network-based paraphrase identification model to detecting duplicate questions in QA communities

KunPeng Du^{1,2} · Xuan Zhang^{1,3,4}  · Chen Gao¹ · Rui Zhu^{1,3,4} · Qiong Nong¹ · XianYu Yang¹ · ChunLin Yin⁵

Received: 18 May 2022 / Revised: 2 May 2023 / Accepted: 21 August 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Paraphrase Identification (PI) is an important task in Natural Language Processing (NLP), which aims to detect whether two sentences expressed in various forms are semantically consistent. It can be used to solve the problem of duplicate detection in QA Communities (eg: Quora and Stack Overflow). There have many studies that applied Convolutional Neural Networks to capture rich matching information between sentence pairs layer by layer. However, only a limited number of studies have explored the more flexible Graph Convolutional Networks (GCNs) for this task. GCN operates directly on the graph, and learns the representation of the node according to the neighborhood information of nodes. Thus, the interactive information between two sentences can be effectively integrated based on the local graph structure. In this paper, a Graph-based Interaction Matching model (GIMM) for PI is proposed. GIMM takes each word as a node, the word co-occurrence relations between sentence pairs, and the phrase relations within a single sentence as the relations between nodes to build the interaction graph. Then, the GCN are applied to learn the richer word representations based on the local structure of the graph. Finally, the node representations are aligned by the Attention mechanism to obtain the matching vector, and the results of PI are obtained by the Fully Connected Layer. We conduct experiments to compare the performance of GIMM with the current baselines on the Quora and Stack Overflow datasets. Experimental results demonstrate that the proposed model achieves excellent performance on both of these datasets.

Keywords Paraphrase identification · Graph convolutional network · Graph-based interaction · Text matching

✉ Xuan Zhang
zhxuan@ynu.edu.cn

¹ School of Software, Yunnan University, Kunming, Yunnan, China

² School of Electro-Mechanical and Information, Yiwu Industrial & Commercial College, Yiwu, Zhejiang, China

³ Key Laboratory of Software Engineering of Yunnan Province, Kunming, Yunnan, China

⁴ Engineering Research Center of Cyberspace, Kunming, Yunnan, China

⁵ Electric Power Research Institute, Yunnan Power Grid Co., Ltd., Kunming, Yunnan, China

1 Introduction

Due to the rich variability of natural language, Paraphrase Identification (PI) plays an important role in the field of Natural Language Processing (NLP). PI aims to detect whether two sentences expressed in various forms are semantically consistent [1]. It can be used to improve many related tasks, such as machine translation evaluation [2], information retrieval [3], question and answer systems [4], plagiarism detection [5], etc. In addition, analyzing data from Community-based Question Answering (CQA) sites (eg: Quora and Stack Overflow) is becoming an increasingly popular direction. It can be very interesting to look at paraphrasing issues on CQA because the CQA often encounters a problem where there are a large number of questions within the community that are expressed in different ways but actually have the same intent. Multiple questions with the same intent may cause the questioner to spend more time trying to find the best answer. It also takes more time for the responder to answer multiple versions of the same question over and over again. The large number of duplicate questions make it more difficult to maintain these Communities and severely impacts the user experience, while reducing the number of duplicate issues in the Communities by manually analyzing and flagging duplicate questions manually take a lot of time and effort. PI is one of the methods to solve the duplication questions in CQA. Recently, many studies for this task have emerged, such as DupPredictor [6], Dupe [7], CQADupStack [8] and SemEval-2016 Task 3[9]. Although these methods detect duplicate questions in CQA automatically, they only use the similarity aspect of the text, matching scores for PI are then generated. Such methods are literal-level text comparisons that do not adequately consider the semantic relevance between texts. Recent work has shown that deep learning models have proven to be effective for PI tasks [10–12]. For PI approaches, there are two different ways to train paraphrase models. The first is the representation-based method [13–16], which seeks to encode each sentence in an independent manner to obtain a high-quality sentence vector. Then the similarity between two vectors is calculated as the match between texts in this type of method, two sentences are encoded independently and they cannot make use of interactive information during encoding. The other is the interaction-based method [10, 11, 17, 18]. Based on the recognition that making a good matching decision requires considering the rich interaction structure in the text matching process [1], the interaction between sentences at all levels is considered to capture rich matching features. In our work, we use an interaction-based approach that captures as much interactive information as possible to improve performance. However, only a limited number of studies have explored the more flexible graph convolutional neural networks for the task. Inspired by the application of Graph Neural Networks (GNNs) in NLP tasks, such as text classification [19, 20], semantic role labeling [21] and relation extraction [22], recommendation systems [3], machine translation [2] as well as summary generation [23]. Compared to the other Deep Neural Network approaches, GNNs learn the richer word representations based on graph's local structures and capture the non-consecutive phrases and long-distance word dependency semantics [24]. For example, in a text classification task, all text and words can be considered as nodes and then a GCN can be applied to complete the classification task and outperform traditional CNN models [19, 20]. In this paper, we propose a new PI model, Graph-based Interaction Matching model (GIMM) based on graph structure. Firstly, in order to obtain multi-level text features, we use three concatenations of word embedding, character embedding and syntactic features as embedding representation. Meanwhile, to effectively integrate the interaction information between sentences, GIMM takes each word as nodes, phrase relations within a single sentence, and word co-occurrence relations between sentence pairs as inter-node relations to construct interaction

graphs. Then GCN is used to learn the richer word representations based on graph's local structures, Finally, the node representations are then aligned by the attention mechanism to obtain the matching vector, and the probability distribution of PI results is calculated based on matching features through the fully connected layer. To demonstrate the effectiveness of our model, we conducted experiments on the Quora dataset with the Stack Overflow dataset to evaluate the performance of our model. Experimental results on these two datasets show that our method can obtain better performance than previous methods.

The main contributions of this paper can be summarized as follows:

- (1) The paper introduces a novel approach for generating interaction graphs that capture rich interaction information between non-consecutive phrases and long-distance words in sentence pairs, which leads to better matching decisions.
- (2) The proposed PI model uses GCNs to learn more comprehensive word representations based on the local structure of interaction graphs. This approach aggregates information from neighboring nodes and interaction information in the graph, enabling the model to fully capture word semantics and interactions.
- (3) The experimental results on publicly available datasets demonstrate that the proposed model significantly outperforms the other baselines and achieves an impressive F1 value improvement of 7.25% on the Stack Overflow dataset. This indicates that the proposed approach has great potential for improving the accuracy and effectiveness of PI tasks.

This paper is structured as follows: we present related work in Section 2, and introduce our proposed GIMM framework in Section 3. We describe the dataset, experiments, and analysis in Section 4. Finally, we conclude and discuss future work in Section 5.

2 Related work

One of the cores of our proposed GIMM is the use of GNNs to learn richer node representations based on the local structure of the interaction graph, other GNN-based models that have guided our work. Also, PI is one of the subtasks of text matching and most of the Text Matching (TM) models are inspiring for the PI task. In this section, we will briefly review two necessary tasks related to our work: Graph Neural Networks and text matching models.

2.1 Graph neural network

GNNs have shown important insights in representation learning, and research on analyzing graphs with machine learning has received increasing attention [25]. Unlike data with regular grid structures (e.g., images and sequences), an increasing number of applications represent data as arbitrary structured graphs [19], including social networks, Knowledge Graphs (KG) [26], protein structures, etc. Typically, nodes in the graph will have variable-sized neighbors, and traditional operations in deep learning (e.g., Convolution and Pooling) will be ineffective [19]. To solve these problems, scholars have proposed GNN [27, 28]. GNNs usually follow the principle of neighborhood aggregation, where nodes compute their representation vectors by iteratively aggregating and transforming their neighborhood representations, i.e., aggregating node information using edge information to generate a new node representation [28]. Inspired by

spectral graph theory, Kipf et al [29] proposed the GCN and achieved state-of-the-art results on several semi-supervised graph classification tasks. Graph Neural Networks (GNNs) typically require graph modeling, which can be referred to as "graph-based solutions." These models first transform text into a word graph before applying graph convolution operations to the word graph. When using GNNs for classification tasks, the way to construct the graph structure is not the same depending on the application scenario. Generally, these models use words or text as nodes of the graph, and edges are constructed in a variety of ways, such as based on word co-occurrence relations [30, 31], document word relations [19], word similarity relations [24], word frequency-inverse document frequency (TF-IDF) relations in documents [19], point-wise mutual information (PMI) relations [32], semantic-aware inverted index structure constituted by inverted index and semantic networks (SemIndex) [33, 34] etc. Some models [24, 30, 31] generally set a fixed size sliding window to collect co-occurrence information in order to take advantage of global word co-occurrence information, while reducing memory consumption during training. The graph structure constructed by these graph-based solutions has the advantage of capturing non-sequential and long-distance semantics, which has achieved excellent performance in text classification tasks. However, GNN-based models are not currently used to handle PI tasks or sentence pair modeling related tasks. The main reason is that there is no efficient way to represent the sequenced text as a graph and to construct the interaction between two sentences. It is not clear whether the GNN-based model is valid for this type of task. Therefore, we try to propose a new model of PI based on graph structure, and through experiments we evaluate the effectiveness of the proposed approach.

2.2 Text matching

Text matching is a central problem in natural language understanding. The study of Text Matching (TM) can be applied to a large number of known NLP tasks, such as information retrieval [3], Question Answering [4], Semantic Textual Similarity [35, 36], machine translation [2], dialogue systems [37], Paraphrase Identification [10–14], and so on. These NLP tasks can all be abstracted to some extent into TM problems, such as information retrieval which can be reduced to matching query terms and documents. Question Answering can be reduced to matching questions and candidate answers, machine translation can be boiled down to matching between two languages, dialogue systems can be boiled down to matching the previous sentence of dialogue and the response, and PI can be boiled down to matching two synonymous words and phrases. Recently, the research on TM has gradually shifted from statistical-based methods to deep neural network-based methods. Usually statistical-based methods need to be based on a large number of manually defined and extracted features with relatively few learnable parameters, and they consider that the more the number of occurrences of the same word in two texts and the closer the ordering of the word sequences, the more the two express the same semantics. Examples include the traditional TFIDF [38], BM25[39], LD (Levenshtein Distance) [40] algorithms based on lexical overlap. Using deep learning methods, features can be automatically extracted from the original data, eliminating the need for a lot of manual feature design overhead. In this research direction, the feature extraction process is part of the model and can be easily adapted to different tasks depending on the training data. The first is the semantics-based solutions, which attempt to extract semantic information from multiple perspectives, use rich semantic information to model

matching models, and then calculate the similarity between two vectors. For text matching, this involves extracting multiple perspectives of semantic information from text, followed by matching. Classic works in this category include the DSSM model [41], Siam-CNN [42], Siam-LSTM [15], and InferSent [43]. These methods encode the two sentences independently, without utilizing interactive information during encoding, but have a simple model structure, strong interpretability, and are easy to implement. Subsequent methods have emerged to enrich semantic information from different perspectives. For example, Wang et al. [44] used external unstructured Wikipedia knowledge combined with the semantic information of two sentences to make predictions, DPIM-ISS [45] considered the interaction between semantics and syntax, Mohamed et al. [46] separated the calculation of named entity tagged semantic similarity from other parts of sentence text, and the SNMA [47] framework separated the comparison and interaction modules, distinguishing semantic differences and extracting interactive information. Additionally, MatchACNN [48] learning architecture extracted features at multiple granularities such as words, phrases, and even sentences, and used a two-dimensional convolutional neural network attention mechanism for matching. These semantics-based solutions have achieved good results in specific scenarios. The other is the interaction-based approach, which captures richer matching features based on the recognition that making a good matching decision requires considering the rich interaction structure in the text matching process [49], the interaction between sentences at all levels is considered to capture rich matching features. The classical models include an enhanced sequential inference model (ESIM) [11], a bilateral multi-view matching model (BiMPM) [12], the RE2 model [17], and the dense interactive inference network (DIIN) [10], etc. Such methods can capture the interactions between two sentences at various levels and thus improve the performance of the model, but the models are generally more complex. In this paper, we propose GIMM for PI using an interaction-based approach. Drawing on previous state-of-the-art work, we construct an interaction graph of two sentences and use GCNs, combined with local structure learning of the graph to obtain a richer fine-grained node representation that aggregates the information of its neighboring nodes, as well as the interaction information in the interaction graph, to fully capture the semantics of word information and its interaction.

3 Method

In this section, we first provide a brief definition of the PI task in Section 3.1, a high-level overview of our model in Section 3.2, and then a detailed description of the Graph Construction and Graph-based Word Interaction in Section 3.3.

3.1 Task definition

In general, we can consider each instance of the PI task as a triple (S_a, S_b, lab) , where $S_a = \{w_1, w_2, \dots, w_m\}$ is a sentence of length m , $S_b = \{v_1, v_2, \dots, v_n\}$ is a sentence of length n . $lab \in Y$ is a label that represents the relation between S_a and S_b , and Y is a set of task-specific labels. The PI algorithm is to learn a classification function $F(S_a, S_b) \rightarrow lab$ for determining whether each instance S_a, S_b is a paraphrased pair or not. Usually, $lab = 1$, means the two sentences have the same semantic meaning, which is a paraphrased pair. $lab = 0$, which means the semantics of the two sentences do not agree, and it is a non-paraphrased pair.

3.2 Model overview

In this section we introduce our proposed Graph-based Interaction Matching model (GIMM). Fig. 1 shows the overall structure, which is divided into Embedding layer, Contextual Encoding layer, Graph Interaction layer, Alignment layer, and Output layer.

Embedding layer The goal of the Embedding layer is to represent each word in S_a and S_b with a d -dimensional vector and construct a representation matrix of the sentences. After pre-processing the sentences S_a and S_b by removing stop words, word stemming, and Lemmatization. We use word embedding, character embedding [50] and word co-occurrence features, an Indicator Function (IF) [49], and the concatenation of these three different vectors as the output of the embedding layer. Word embedding is obtained by mapping words to a high-dimensional vector space using a pre-trained word vector GloVe²[51]. Character embedding¹ follows the work of Wang et al [50], where all characters of a word are first vectorized and then the character sequence is fed into the Bi-directional Long Short-Term Memory Network (Bi-LSTM), and then the forward final output of the Bi-LSTM and the reverse final output is concatenated, where the character embedding is randomly initialized and learned along with other network parameters in the model. Character embedding provides additional information for some out-of-vocabulary (OOV) words. Word co-occurrence features are used to mark two sentence pairs with the same stem between them, and this simple exact matching helps the model to understand the sentences better [10], considering that there are generally a certain number of co-occurring words in the paraphrase pairs. The output of this layer is the embedding vectors $E_a : [e_1^a, \dots, e_m^a], E_b : [e_1^b, \dots, e_n^b]$. The embedding vector e_i^a is the concatenation of word embedding, character embedding and word Co-occurrence features, and so is e_j^b .

Contextual encoding layer In this layer, Bi-LSTM is used to merge contextual information into the representation of each word. The Bi-LSTM consists of forward and backward LSTMs [52]. With its three-gate structure, the LSTM can solve the long-term dependence problem well. Using Bi-LSTM, the bidirectional semantic dependencies within neural units can be well captured. This step can be represented by the following equation.

$$\vec{h}_i = \overline{LSTM}(\vec{h}_{i-1}, e_i) \quad i = 1, \dots, N \quad (1)$$

$$\overleftarrow{h}_i = \overline{LSTM}(\overleftarrow{h}_{i+1}, e_i) \quad i = N, \dots, 1 \quad (2)$$

$$h_i = [\vec{h}_i; \overleftarrow{h}_i] \quad (3)$$

where N is the length of the input sequence and e_i is the embedding vector of the embedding layer. \vec{h}_i is the hidden layer state of the forward LSTM in time step i , \overleftarrow{h}_i is the hidden layer state of the backward LSTM, and h_i is the input of the Bi-LSTM in time step i , denoted as the contextual embedding of the word w_i .

¹ Our pre-trained word embeddings are openly available in <https://nlp.stanford.edu/projects>

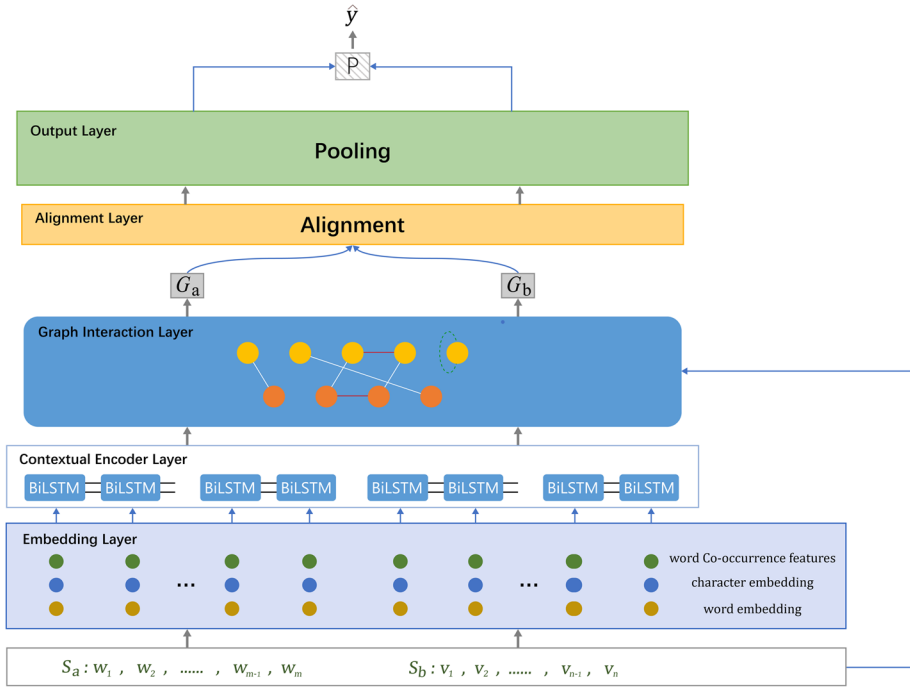


Fig. 1 The overall structure of the Graph-based Interaction Matching model (GIMM), where the Graph Interaction Layer constructs a graph of interactions between two sentences by treating each word as a node of the graph. The contextual embedding of the word is used as the feature of this node, and the word co-occurrence relations between sentence pairs, and the phrase relations within a single sentence is used as the inter-node relations. using its node features and graph structure information, the representation of the node is learned through GCN, and the output is G_a, G_b representation sequence. p is the value of the sentence pair representation vector pooling after doing the concatenation

Graph interaction layer This is the core layer of our model. We take each word w_i as a node of the graph, the contextual embedding of the word as a feature of that node, the word co-occurrence relations between sentence pairs, and the phrase relations within a single sentence as the inter-node relations are used to construct the interaction graph of two sentences. Then, according to the node features and the structure information of the graph, the embeddings of word nodes are learned through GNNs, where a node can aggregate information from neighboring nodes and update its representation based on its original representation and the information aggregated to obtain a graph interaction representation. The output of this layer is a sequence of two interaction features $G_a : [g_1^a, \dots, g_m^a], G_b : [g_1^b, \dots, g_n^b]$. The specific graph structure and graph-based interaction operations are described in detail in Section 3.2.

Alignment layer The attention mechanism was first used in neural machine translation models [53]. The attention mechanism can efficiently extract the interaction information between sentences by inter-sentence alignment and obtain better performance. In this paper, the attention mechanism is applied to capture semantic interactions between pairs of sentences. The alignment layer takes the features G_a, G_b of two sequences as input and computes the aligned representation as output. The output Graph Interaction Layer has a

sequence of graph interaction features $G_a G_b$ where the attention weight value e_{ij} between g_i^a and g_j^b ($i \in [1, m], j \in [1, n]$) is computed as the dot product of the two vectors.

$$e_{ij} = g_i^{aT} g_j^b \tag{4}$$

The aligned output vectors a' and b' are computed by a weighted summation of the representations of the other sequence. This summation is obtained by weighting the attention weights between the current position and the corresponding position in the other sequence and is calculated as follows:

$$a_i' = \sum_{j=1}^n \frac{\exp(e_{ij})}{\sum_{k=1}^n \exp(e_{ik})} g_j^b \tag{5}$$

$$b_j' = \sum_{i=1}^m \frac{\exp(e_{ij})}{\sum_{k=1}^m \exp(e_{kj})} g_i^a \tag{6}$$

Intuitively, a_i' is the weighted sum of $\{g_j^b\}_{j=1}^n$, and the content associated with a_i' in $\{g_j^b\}_{j=1}^n$ will be selected and represented as a_i' . The alignment vectors a' and b' are obtained by performing the same operation on each word in the sequence. As with other advanced models [10, 11, 17], to further compare the graph-interaction (local) representation and the aligned representation, we concatenated the graph-interaction representation of the words with the aligned features to merge the aligned features. Comparing the local and aligned representations from three perspectives, the formula is as follows:

$$\bar{a}_i^1 = f_1([g_i^a; a_i']) \tag{7}$$

$$\bar{a}_i^2 = f_2([g_i^a; a_i - a_i']) \tag{8}$$

$$\bar{a}_i^3 = f_3([g_i^a; a_i \cdot a_i']) \tag{9}$$

$$\bar{a}_i = [\bar{a}_i^1; \bar{a}_i^2; \bar{a}_i^3] \tag{10}$$

where $[\cdot]$ is the concatenation operation, $[-]$ represents the subtraction operation highlights the difference between two vectors, and $[\cdot]$ indicates the dot product operation highlights the similarity between vectors. f_1, f_2, f_3 are three single-layer feedforward networks with independent parameters respectively. Finally, we concatenate the results obtained from the above three fusion methods and input them into another single-layer feedforward network F to get \bar{a}_i , and do the same for the other sentence to get \bar{b}_j .

Output layer According to Eq. (3), the output \bar{a} and \bar{b} of the Bi-LSTM is obtained, and the maximum pooling and average pooling are performed, and the values after pooling are concatenated together again. The specific formula is as follows:

$$\bar{h}_{a,ave} = \sum_{i=1}^m \frac{h_{a,i}}{m} \tag{11}$$

$$\bar{h}_{a,max} = \max_{i \in [1,n]} h_{a,i} \quad (12)$$

$$\bar{h}_{b,ave} = \sum_{j=1}^n \frac{h_{b,j}}{n} \quad (13)$$

$$\bar{h}_{b,max} = \max_{j \in [1,n]} h_{b,j} \quad (14)$$

$$p = [\bar{h}_{a,ave}; \bar{h}_{a,max}; \bar{h}_{b,ave}; \bar{h}_{b,max}] \quad (15)$$

p is the value from pooling. In this step, p is as the input for GIMM. Therefore, the final probability distribution is output according to the experimental setup of Mou et al [54]. The specific formula is as follows:

$$\hat{y} = H(p) \quad (16)$$

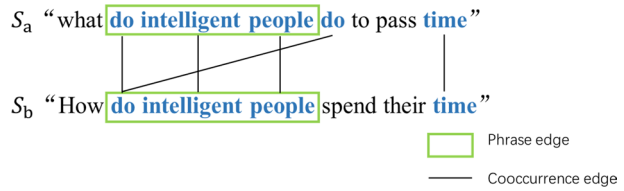
H is a multi-Layer Perceptron (MLP) classifier. In the experiment, the MLP has a hidden layer with tanh activation and a softmax output layer. where $\hat{y} \in \mathbb{R}^C$, is the probability distribution of each class, where C is the number of classes.

3.3 Graph-based interaction

The graph-based interactions are implemented in the Graph Interaction Layer, which we introduce in two parts: Graph Construction, Graph-based Word Interaction.

Graph construction In this section, we describe how to represent the sentence pairs S_a and S_b as a graph G . Suppose that the graph $G = (V, E)$, where V is the set of vertices with node characteristics and E is the set of edges as topology. We denote each word in a sentence pair as a node of size $m + n$, the sum of the two sentence lengths, then $V = \{w_1, w_2, \dots, w_m, v_1, v_2, \dots, v_n\}$. The output of the Contextual Encoding Layer, i.e., the contextual embedding of the word, is used as the features of this node. In addition to the node feature matrix, the adjacency matrix describing the topology also forms the graph. This structure usually describes the connections between nodes and reveals their relations, i.e., edges. Pang et al [49] argue that two semantically identical sentences have many identical or similar counterparts at the word level. The same holds true at the phrase level, where more information can be obtained by doing interactions at a finer granularity. Pang et al [49] constructed a matching matrix when constructing the relations between words, considering three different perspectives: Indicator Function (whether two words are the same), cosine (cosine similarity), and dot (inner product). They construct a word order retention graph and then select one of the top N nodes from each node's ranking of proximity centrality features to establish a relation. Similarly, edges can be used to represent different types of relations between any nodes, such as lexical or semantic relations [55]. We consider two types of relations between sentences when constructing edges: word co-occurrence relations between pairs of sentences and phrase relations within a single sentence. The construction of these two types of relations is shown in Fig. 2:

Fig. 2 An example of Word co-occurrence relations between sentence pairs, phrase relations within a single sentence



The word co-occurrence relations between two pairs of sentences are used to describe a relation between two pairs of sentences that have the same word. We use NLTK² to do Lemmatization and Stemming for all words. After the above process, if a word in S_a is same as another word in S_b , then these two nodes have a Cooccurrence edge. Considering that it is easy to have long sentences in which a word in S_a has more than one co-occurrence in S_b , for example, S_a is "what do intelligent people do to pass time?" and S_b is "how do intelligent people spend their time". The "do" in S_b corresponds to the two "do" in S_a . Obviously, the semantics of the expression is more similar to the first "do" in S_b . We set up a sliding window, and the final co-occurrence relation describes the relationship between words that appear within a fixed-size sliding window, where each word is connected to neighboring words that may share related contextual meanings. The sliding window is set to avoid dense connections of graphs. If the connections of a graph are too dense, the structural information is blurred during message passing in the GNN [30]. The effectiveness of sliding windows was demonstrated by the work of Nikolentzos et al [56] and Zhang et al [30]. After getting the word co-occurrence relations between sentence pairs, we consider the phrase-level relations within a single sentence, and we connect the words that make up the phrases under the co-occurrence relations as phrase relations. As in the above example S_a and the set of co-occurring words of S_b is: {do, intelligent, people, time}, where {do, intelligent, people} is sequentially connected according to the sequence order in the sentence, then "do intelligent people" is a phrase, then they are connected using phrasal edges.

Then the edges can be represented by the adjacency matrix $A \in [0, 1]^{|n+m| \times |n+m|}$, where the formula is as follows:

$$A_{i,j} = \begin{cases} Cooccurrence(i,j) & (i \in [1, m], j \in [1, n]) \\ Phrase(i,j) & (i \in [1, m], j \in [1, n]) \\ 0 & other \end{cases} \quad (17)$$

where $A_{i,j} = 0$ means node i and node j are not related and are not connected in the graph, $Cooccurrence(i,j)$ means node i and j have co-occurrence relations. $Phrase(i,j)$ means node i and j have phrase relations, and the values of $Cooccurrence(i,j)$ and $Phrase(i,j)$ are hyperparameters and each node is connected to itself.

Graph-based word interaction GCN is a multilayer neural network that operates directly on the graph structure [29], it aggregates the current node features and first-order neighboring node features to represent the new features of the current node, by which it facilitates the introduction of contextual information of graph nodes, making each node representation influenced by neighboring nodes. In Graph Construction we constructed graphs for

² <https://www.nltk.org/>

sentence pairs, and we formed all node features into a feature matrix $X \in \mathbb{R}^{|m+n| \times d}$, where $|m+n|$ is the total number of nodes and d is the feature dimension. Then the message passing rules of multilayer GCN can be expressed as the following equation, and the message passing of GCN is shown in Fig. 3:

$$H^{(k+1)} = f(H^{(k)}, A) \tag{18}$$

Here k refers to the number of network layers, and $H^{(k)}$ is the feature of the k th layer of the network, where $H^{(0)} = X$. where A is the adjacency matrix obtained from Equation (17).

Following the graph convolutional operator given by Kipf et al [29], Eq. (18) becomes

$$H^{(k+1)} = f\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)}\right) \tag{19}$$

where $\tilde{A} = A + I$ is the adjacency matrix of the undirected graph G with added node self-connections. I is the unit matrix, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$, $W^{(k)}$ is the trainable weight matrix, and then $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the normalized symmetric neighbor matrix. $f(\bullet)$ denotes the activation function, and $H^{(k+1)} \in \mathbb{R}^{|m+n| \times d}$ is the activation matrix of the $k+1$ th layer.

Since our edges are provided with weight values to distinguish between classes of edges, the nodes are updated with the following formula:

$$H_i^{(k+1)} = W^{(k)T} \sum_j \frac{\alpha_{j,i}}{\sqrt{d_j d_i}} H_j^{(k+1)} \tag{20}$$

where $d_i = 1 + \sum_j \alpha_{j,i}$ is the edge weight i from source node j to target node. We capture information about direct neighbors by a layer of graph convolution to obtain two sequences of interaction features $G_a : [g_1^a, \dots, g_m^a]$, $G_b : [g_1^b, \dots, g_n^b]$.

4 Experiments

In this section, we evaluate our model on two PI datasets. We first introduce the datasets in Section 4.1, and the experiments setup and evaluation metrics of the GIMM model are also briefly described. Then, we present the baseline of the experiments and compare our model with state-of-the-art models in Section 4.2. Finally, in Sections 4.3, 4.4, and 4.5 we perform some case studies to fully evaluate the properties of our model.

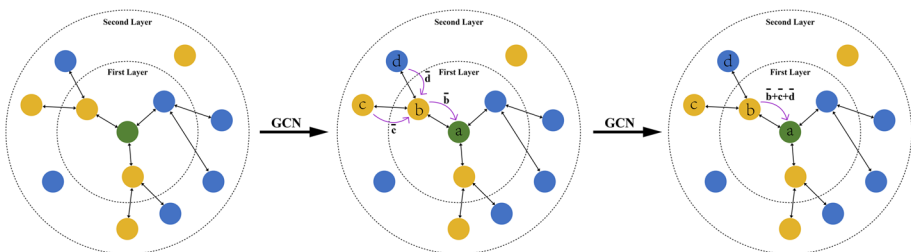


Fig. 3 GCN's message passing schematic

4.1 Dataset and experiment settings

In this section, we briefly describe the datasets used in the experiments and the experiment setups and evaluation metrics.

Quora Question Pairs³ [57] is a dataset containing 400,000 question pairs collected from the Quora website. Each question pair is labeled with binary values indicating whether two questions are paraphrased from each other. We randomly select 5,000 paraphrases and 5,000 non-paraphrases as the validation set, and extract another 5,000 paraphrases and 5,000 non-paraphrases as the test set. What's more, the remaining instances are as the training set.

SKU⁴ (Stack Overflow Knowledge Unit dataset) [58] is constructed based on a question-and-answer data dump from the Stack Overflow website and contains 347,372 knowledge unit pairs. In this dataset, there are four association classes (Duplicate, Direct, Indirect, Isolated) for Knowledge Unit (KU) pairs according to the degree of association between two knowledge units from high to low. Duplicate association classes represent two KUs discuss the same question in different ways and can be answered by the same answer. Direct association classes represent one KU can help solve the problem in the other KU. For example, by explaining certain concepts, providing examples, or covering a sub-step for solving a complex problem. Indirect association classes represent one KU provides related information, but it does not directly answer the question in the other. Isolated association classes represent the two KUs are not semantically related [58]. The number of each association class is 1/4. 60% of the KU pairs in the dataset are used as the training set, 10% as the validation set, and 30% as the test set.

Taking a sample of KU pairs as an example, each instance of the dataset can be represented as $\langle KU_a, KU_b, lab \rangle$, where $lab \in \{Duplicate, Direct, Indirect, Isolated\}$ is the label indicating the relationship between KU_a and KU_b . For each $KU = \{Title, Body, Answer\}$. *Title* represents the title of the question in Stack Overflow, *Body* is the detailed description of the question (Exclude Code Snippets), and the "Answer" indicates the text of question's Accepted Answer (Exclude Code Snippets). The structure of the SKU data set is shown in Table 1.

Both Quora Question Pairs and SKU are datasets built by taking data from QA communities. The difference is that Stack Overflow is a domain-specific QA community for software developers, focusing on programming questions, most of which have terminology or special characters, and the dataset does not simply detect whether knowledge units are duplicated, but requires the model to consider the relevance of knowledge units. Quora, on the other hand, is a popular Q&A community where people ask more varied questions that cover a wide range of topics, including technology, entertainment, politics, culture, and philosophy, and some questions include special characters, such as mathematical symbols and foreign language characters. The average length of questions in Quora is 59. SKU has different text lengths depending on the different parts of the KU, where the average length of Title is 9, Body is 67, and Answer is 68. choosing different sentences of the dataset allows us to evaluate the effectiveness of our model for texts of different sizes. The statistics of the two datasets are shown in Table 2. In summary, evaluating our proposed model on these two highly different datasets, which are both based on community QA, provides an objective demonstration of the performance of the proposed model on the task of detecting duplicate questions.

³ <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>

⁴ <https://anonymoussaaai2019.github.io/>

Table 1 The structure of the SKU dataset.

label	KU_a	KU_a
duplicate	<p>Title: "Convert epoch time to dd MM yyyy using java" Body: "I need to convert a String which is a epoch Unix time format to a Date class an after a String formatted dd MM yyyy Thank you for your help!" Answer: "Unix time is the number of seconds since 1 January 1970 so this should work BTW SimpleDateFormat accepts millis as argument too, so it is possible to get the same result as "</p>	<p>Title: "Converting Integer time stamp into java date " Body: "I have following time stamp in Integer form I can convert it using SQL. How to convert it in java So that it would return value e" Answer: " Assuming its the time since 1 1 1970 in seconds you can try"</p>

Table 2 Dataset Statistics. where the data in parentheses indicate the percentage of the whole dataset

Dataset	Quora Question Pairs [57]	SKU [58]
Sources	https://data.quora.com	https://stackoverflow.com
Train Set Size	380k (95%)	208k (60%)
Valid Set Size	10k (2.5%)	34k (10%)
Test Set Size	10k (2.5%)	104k (30%)
All	400,000	347,372
Average Length of Data	59	Title:9 Body:97 Answer:68
Vocabulary Size	743K	234k
Number of Categories	2	4
Categories	Paraphrased (37%) Non-Paraphrased (63%)	Duplicate (25%) Direct (25%) Indirect (25%) Isolated (25%)

Finally, we implement our model using Pytorch and trained it on an NVIDIA GeForce RTX 3060. First, pre-processing all sentences, converting them to lowercase letters and removing all punctuation and stop words. The word embeddings are initialized using 840B-300-dimensional GloVe word vectors and fixed during training. The embeddings of the out-of-vocabulary words (OOV) are also randomly initialized and fixed. We use the Adam optimizer with the learning rate set to 0.001. For better training and to force the network to find different activation paths for better generalization, a dropout layer with a value of 0.2 is used. We do not limit the maximum sequence length, all sequences in the batch are populated to the batch maximum, the batch size is from 128 to 32, and all measures reported by the model for the tests correspond to the best values obtained on the validation set.

4.2 Comparison

In this section, we compare our model with state-of-the-art models on PI tasks.

Experiment I: Results on quora The selected baselines are as follows.

Siamese-LSTM [15]: The classical Siamese Neural Network model proposed by Mueller et al, which uses a neural network encoder to encode two input sentences into sentence vectors and makes decisions based on the cosine similarity between the two sentence vectors.

L.D.C[16]: An attention-based model proposed by Wang et al, which decomposes the hidden representation into similar and distinct parts and then processes each part separately to generate the result.

ESIM [11]: Chen et al proposed an enhanced sequential inference model that achieved competitive performance on an eight-sentence pair modeling task. It uses a Bi-LSTM to encode each sentence, connects the encoded representations of each pair of sentences, and classifies them by a multilayer perceptron (MLP).

BiMPM [12]: A bilateral multi-perspective matching model proposed by Wang et al, which first encodes sentence pairs using a Bi-LSTM encoder. and matches the two encoded sentences in both directions. Then, the matching results are aggregated into a fixed-length matching vector using another Bi-LSTM layer. Finally, based on the matching vector, a decision is made by a fully connected layer.

RE2[17]: This model proposed by Yang et al exploits three key features of inter-sequence comparison: original point-wise features, previous aligned features, and contextual features while simplifying all the remaining components. The model takes embedding as input and stacks layers consisting of encoding, alignment, and fusion. The pooling layer aggregates the sequence representations into vectors, which are finally processed by the prediction layer for the final prediction.

DIIN [10]: Gong et al. proposed the densely interactive inference network adopts DenseNet, a two-dimensional convolutional structure, to extract higher-order verbatim interactions between n-gram pairs. The model has achieved state-of-the-art performance.

BERT-base [13]: Arase proposed transfer fine-tuning using phrasal paraphrases to allow BERT's representations to be suitable for semantic equivalence assessment between sentences.

PPBERT-base [13]: Arase et al. proposed a model called PPBERT-base by using transfer fine-tuning and paraphrase classification training on the BERT-base model to solve PI tasks.

DITM [18]: Yu et al proposed the Deep Interaction Text Matching (DITM) model. It uses a matching-aggregation framework and integrates the encoder layer, co-attention layer, and fusion layer as an interaction module to achieve deep interaction. The interaction process is iterated multiple times to obtain in-depth interaction information, and the relationship between text pairs is extracted through a collection of multiple perspectives.

Unified Approach [14]: Palivela et al. proposed a lightweight unified model aimed at solving paraphrase identification and generation problems by using carefully selected data and fine-tuning the T5 model. This lightweight model can be trained to achieve the goal of paraphrase generation and can also be used to solve the task of paraphrase identification.

Our comparative baselines include two different modeling methods for the PI tasks. One-class method is a representation-based model, such as Siam-LSTM or L.D.C, which independently encode two sentences without utilizing interaction information during encoding. Additionally, with the development of large-scale language models, Researchers can use these models to obtain better sentence representations and achieve better results for the PI task through simple fine-tuning. such as BERT-base, PPBERT-base, and Unified Approach. Although representation-based models have a simple and interpretable structure, they fail to fully capture the interaction between sentences and generally perform poorly. Another kind of method is an interaction-based model, such as ESIM, BiMPM, DIIN, RE2, and DITM. These methods capture interactions at different levels between two sentences, thereby improving the model's performance, but the models tend to be more complex. The above two methods, the former of which solves PI tasks based on semantic similarity technology by obtaining better sentence representations, and the latter of which uses state-of-the-art deep learning algorithms such as convolutional neural networks (CNN), recursive neural networks (RNN), and long short-term memory (LSTM) to capture rich matching information between sentence pairs layer by layer. However, only a few researchers have explored more flexible GCNs for the PI tasks. GCNs operate directly on the graph and learns node representations based on adjacency information, thereby

effectively integrate interaction information between the two sentences, and model them based on local graph structure.

Compared with the previous state-of-the-art work, we propose a graph-based interactive matching model (GIMM) for PI tasks. GIMM constructs an interactive graph for two sentences and uses GCNs to learn rich and fine-grained node representation by integrating local structural information of the graph. In addition, it aggregates information from adjacent nodes and interaction information in the graph to fully capture the semantic information and interaction of words.

The corresponding comparative experimental results are shown in Table 3:

We can see from Table 3 that the accuracy of Siamese-LSTM, L.D.C, BiMPM, and ESIM on the Quora dataset are: 0.826, 0.848, 0.882, and 0.853. The accuracy of BERT-base, PPBERT-base, and Unified Approach on the Quora dataset are: 0.883, 0.880, and 0.872. The accuracy of DIIN, RE2, DITM and our proposed GIMM are: 0.891, 0.892, 0.892 and 0.892, respectively. Through analysis, we find that Quora's duplicate question pairs accounted for only 37% of the training data. This indicates a significant imbalance in the dataset, with many more non-paraphrased questions than paraphrased pairs, and 20% of the questions appearing multiple times in different question pairs. In addition, we further analyze the reasons affecting the performance of GIMM, First the training dataset is noisy, i.e., there are pairs of questions labeled as non-paraphrased, where two questions share almost all words and the exact same meaning. For example, a question pair consisting of sentences ["What is solution for this question?", "What is solution to this question?"] is labeled as non-paraphrased, although the question pair differ only in an insignificant deactivation word. On the other hand, the number of co-occurring word pairs has an impact on the identification of repetition problems. If there is a phenomenon that two questions share almost all words, but the difference in a keyword causes the semantics of the two sentences to be different. For example, the question consisting of the sentence ["How can you visualize a perfect 2-dimensional space", " How can one visualize 4-dimensional space"] is labeled as non-paraphrased. The mismatch in the keywords " you/one" and "2-dimensional/4-dimensional" leads to a semantic inconsistency between the two sentences. This type of phenomenon also has an impact on our model. Since GIMM constructs an interaction graph in which the two sentences are densely connected. Structural information is ambiguous during message passing in GNN. The representations of the nodes learned in this way will be relatively similar in vector space. This is not conducive to our

Table 3 Experimental results on the QQP dataset. Bolded values represent the optimal or best-performing results among the selected baselines on the QQP dataset

Model	Accuracy
Siamese-LSTM [15]	0.826
L.D.C [16]	0.848
BERT-base [13]	0.883
PPBERT-base [13]	0.880
Unified Approach [14]	0.872
BiMPM [12]	0.882
ESIM [11]	0.853
DIIN [10]	0.891
RE2[17]	0.892
DITM [18]	0.892
GIMM (our)	0.892

Table 4 Experimental results on SKU

Model	Overall: Micro-F1	Precision	Recall
SOFTSVM [58]	0.59	0.58	0.59
DOTBILSTM [58]	0.75	0.75	0.75
GIMM (our)	0.8225	0.8290	0.8225

subsequent alignment and leads to degradation of the model performance. How to avoid such phenomena and build high-quality interaction diagrams is one of the directions of our future work.

In addition, we have conducted experiments on the SKU dataset. This dataset requires the model to detect whether the questions are duplicated or not, but it also to considering the correlation between questions.

Experiment II: Results on stack overflow The experimental results for the SKU dataset are presented in Table 4. These include two models proposed by Shirani et al [58], SOFTSVM, an SVM model for questions relevance tasks. SOFTSVM investigated the impact of different features and different data choices on the final results. Another is DOTBILSTM, a BiLSTM-based model. DOTBILSTM used dot product to calculate the matching degree of KUs after encoding by BiLSTM, which progressively learns and compares the semantic representation of different parts of two KUs.

Through Table 4 we can see that the F1 values of SOFTSVM, DOTBILSTM on SKU dataset are: 0.59, 0.75 respectively, and the F1 value of our GIMM is 0.8225, which is 7.25% improvement compared to DOTBILSTM. In addition, it can be seen by Fig4 that the F1 value of GIMM exceeds 0.8. Table 5 shows the predicted Micro-F1 scores for individual classes, comparing the results for each. GIMM performed better than the baselines in predicting the four categories duplication, direct, indirect, and isolated. The improvement is 13% in predicting the Direct class, 8% in predicting the Indirect class, and 5% in predicting the Isolated class. In particular, on the duplicate problem detection, GIMM achieves 0.9333 and 0.9206 for Micro-F1 in predicting the duplicate and indirect classes, respectively. It can also be seen intuitively in Fig 5 that the F1 value of GIMM on the duplicate and indirect classes exceeds 0.9. This, to some extent, indicates that our model has a great advantage on a specific domain. SKU provides a richer text description, which facilitates the model to more fully capture the semantic meaning of what the sentence is trying to express. Therefore, the domain specificity

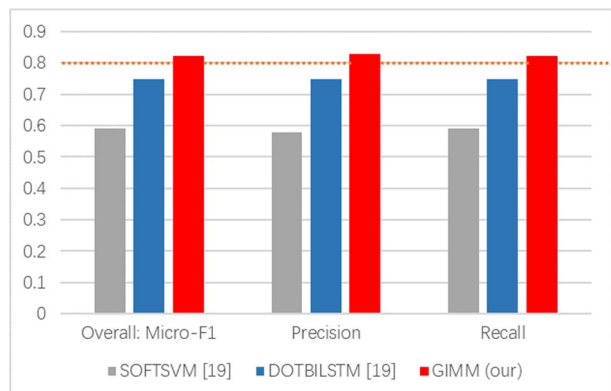
Fig. 4 Results of the model on the SKU dataset

Table 5 Micro-F1 values for predicting individual classes

Model	Overall: Micro-F1	duplicate	direct	indirect	isolated
SOFTSVM [58]	0.59	0.53	0.57	0.44	0.79
DOTBILSTM [58]	0.75	0.92	0.55	0.67	0.87
GIMM (our)	0.8225	0.9333	0.6785	0.7536	0.9206

of the SKU, the co-occurrence of words between the two KUs is mainly in Terminology and some Humping Variables (e.g., "IndexOutOfBoundsException"). To a certain extent, the quality of the interaction graph of these two KUs is guaranteed. This combined with the local structure of the graph to learn the node representation is more helpful for the model to make good matching decisions. In the subsequent sections we will discuss these important components that have an impact on the model to check the validity of each component Figs. 4 and 5.

4.3 Ablation study

To fully evaluate the performance of our model, we further analyze the main components of GIMM to check the impact of these components on the model by removing a portion of the components.

First, we analyze on the SKU dataset which part of the KU has the most impact on the performance of the final model. The comparative experimental results are shown in Table 6.

We can see that using only the Body section of KU, the Precision, Recall, and Micro-F1 achieved the best results with an F1 value of 0.758, while the Answers section performed poorly with an F1 value of 0.5887. Using only the data from the Title section of KU, its F1 value is 0.758. The performance of different parts of the KU varies widely in the model. Different parts of the KU have different data sizes, Title having the smallest data size with an average length of 9, Body with an average length of 97, and Answer with an average length of 68. The obvious difference in their data size is one of the reasons for the vast difference in performance. Secondly, they have different functions in KUs. Title and Body are descriptions of the questions, and their relevance in analyzing KUs plays a decisive role. Answer is a response to a question, which is useful for question duplication detection.

Fig. 5 Micro-F1 values of different models in each class of SKU dataset

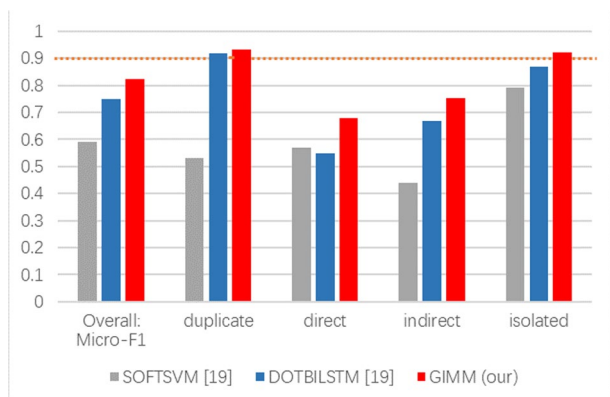


Table 6 Results of choosing different text selections

Model	Overall: Micro-F1	Precision	Recall
Title	0.7419	0.7478	0.7419
Body	0.7580	0.7629	0.7580
Answers	0.5887	0.6043	0.5887

Then, based on the model architecture, we further analyze the components that are important to help GIMM achieves good performance. We are more interested in the effect of the Title part of the question on the detection of duplicate questions. Because the general problem is not described specifically, and also Title is a short text, the performance of GIMM on the short text can be evaluated. The experimental results are shown in Table 7.

First, we verify the effect of the character embedding and word co-occurrence features of the embedding layer on the model, and compare the F1 scores of the modified model with those of the original model in the title section. Specifically, we remove both character embedding and word co-occurrence features (-Char -IF). This setup demonstrates the importance of adding other embedding features compared to traditional word embedding. In addition, we also perform the following ablation studies, including (1) removing only character embedding(-Char); (2) removing only word co-occurrence features (-IF). We measure the extent to which these isolated modifications affect the model performance and analyze their impact on the model.

Then the effect of Graph Interaction Layer (GIL) on the model is verified. Specifically first remove the GIL(-GIL), which is obviously set up to highlight the impact of that layer on the model. Importantly the main focus of this layer on the impact of the model is the graph structure. Therefore, the influence of the choice of edges on the model when constructing the graph is further verified. This is done as follows: (4) remove the co-occurrence edge (- Co-occurrence edge) and keep only the phrase edge, so that the interaction between the two sentences of the layer will be lost. (5) remove the phrase edge (- phrase edge), where only the match between co-occurring words between sentences is considered. Finally, we remove both character embedding, word co-occurrence features, and GIL, leaving the model structure as a traditional attention-based interaction model.

Through Table 7, we can see that Micro-F1 decreases by 0.91% after removing the character embedding (-char). We find a large number of Humped Variables in the SKU dataset. The Humped Variables here refer to words like "IndexOutOfBoundsException", "StrutsPrepareAndExecuteFilter", etc. Word embedding is not effective in providing additional information for these out-of-vocabulary words (OOV). When the word co-occurrence feature (- IF) is removed, we find that the model performance decreases by 0.77%. As observed

Table 7 Ablation experimental results on SKU

Model	Micro-F1
GIMM(Title)	0.7419
(1)-Char	0.7328
(2)- IF	0.7342
(3)-Char - IF	0.7306
(4)- Co-occurrence edge	0.7282
(5)- phrase edge	0.7332
(6)- GIL	0.7250
(7)-Char - IF -GIL	0.7126

by Gong et al [10] and Chen et al [59], the simple exact alignment function does help the model to understand the sentences better. Removing both the character embedding and word co-occurrence features decreases the model performance by 1.13%. It shows that adding richer static embedding information to the embedding has a boosting effect on the PI model. Because of the inclusion of rich and effective static features, the embedding vector contains more semantic information and better sentence representations can be obtained.

Secondly, removing the Graph Interaction Layer decreases the F1 value by 1.7%. GIL incorporates the local structure of the graph. Aggregate the contextual information of graph nodes such that each node feature representation is influenced by neighbor nodes. Fine-grained feature representations can be learned. There is a positive effect on the PI task. Considering the impact of edge relations on the model during graph construction. After removing co-occurring edges, the model performance decreases by 1.37%. GIMM considers the interaction between identical words. The sentence vectors of two pairs of sentences will be close to each other in the vector space when they meet pairs of sentences with essentially the same expression form. At the same time, the model performance decreases by 0.87% after we remove the phrase edges. As Pang et al [49] argue that different levels of interaction allow more information to be obtained. For example, "bread and milk" and "milk and bread", which semantically express the same meaning and show that considering the basic features of the phrase can help with the PI task. Finally, we remove both character embedding, word co-occurrence features, and GIL, the model performance decreases by 2.93%.

In addition, we also consider the impact of the number of layers of GNN in the Graph Interaction Layer, and it can be seen that increasing the number of convolution layers has little effect on the model. The experimental results are shown in Table 8. When the number of layers of the GNN is 2, it allows nodes to learn to get information about their 2nd order neighbors. Since there are only two different edges in our interaction graph, and the graph structure is sparse. The general single-layer convolution can already learn the contextual information including the information of the nodes connected by these two edges. The increase in the number of layers of GNN does not capture more information and it also makes model training more difficult.

4.4 Model analysis

In this section, we discuss the computational complexity of GIMM and compared it with several classic algorithms. Specifically, we analyze the computational complexity of each algorithm and provide some insights on the trade-off between efficiency and accuracy.

In Table 9, complexity refers to computational complexity, which we represent using big O notation. We select the following benchmark models for comparison: Siamese-LSTM, BERT-base, ESIM, RE2, and DITM. Siamese-LSTM and BERT-base are classic representation-based models, while ESIM, RE2, and DITM are currently the best performing interaction-based models. The specific comparison results are shown in Table 9. Complexity is the theoretical decoding computational complexity relative to sequence length

Table 8 Impact of the number of GCN network layers on the SKU-Title results

Model	Micro-F1
GCN Number of layers = 1	0.7419
GCN Number of layers =2	0.7412

Table 9 Complexity of GIMM and comparing it with several classic algorithms

Model	Parameters	Complexity
Siamese-LSTM [15]	0.3M	$O(n)$
BERT-base [13]	110M	$O(n)$
ESIM [11]	4.3M	$O(n^2)$
RE2[17]	2.8M	$O(n^m)$
DITM [18]	5.9M	$O(n^m)$
GIMM	2.6M	$O(e^2)$

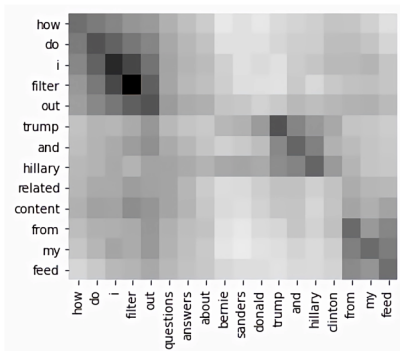
n and interaction module size. Siamese-LSTM and BERT-base do not consider interaction between sentences, so their computational complexity is $O(n)$. In contrast, ESIM, RE2, and DITM require the design of interaction modules. The interaction modules of RE2 and DITM require multiple iterations to better capture the depth of interaction information between text pairs, resulting in computational complexity of $O(n^m)$, where m is the number of iterations. The local inference modeling module of ESIM is accomplished through the attention weight matrix and vector concatenation of the inference composition module, resulting in computational complexity of $O(n^2)$. GIMM selects the information that needs to interact through sliding windows, resulting in computational complexity of $O(e^2)$ where $e = n/w$ and w is the size of the sliding window.

In short, from the table above, it can be observed that the representation-based models have fewer parameters and lower algorithmic complexity. These models have a simple and interpretable structure, but they fail to fully capture the interaction between sentences and generally perform poorly. With the introduction of large-scale language models, the representation-based models can better represent the semantic information of sentences and improve model performance. However, these language models require a huge number of parameters and corresponding computing resources, which have high device requirements. Interaction-based models need to design interaction modules to capture interactions at different levels between two sentences, thereby improving model performance. However, these models tend to be more complex, and their computational complexity is generally $O(n^2)$ or higher.

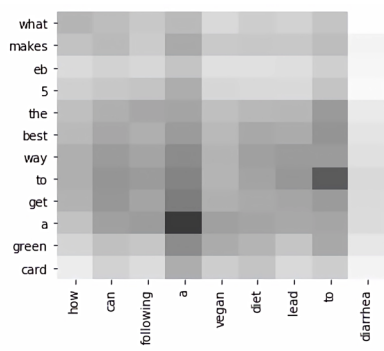
In addition, from the table we can also see that GIMM is superior to state-of-the-art methods in terms of model complexity and parameter quantity due to its carefully designed interaction module. GIMM builds an interactive graph and uses sliding windows to reduce the amount of node information needed for comparison, thereby learning rich and fine-grained node representations while integrating local structure information of the graph to fully capture the semantic information and interaction of words. Therefore, GIMM can reduce the number of model parameters.

4.5 Case study

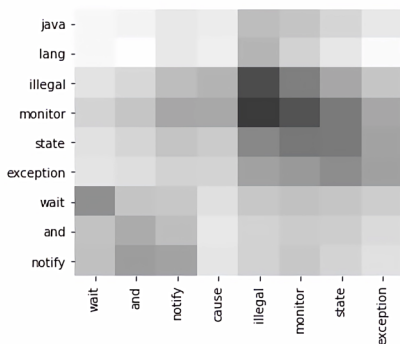
To visually demonstrate the validity of the model, we conduct a case study in the experimental setting of Quora and SKU. Fig. 6 shows the effect of Graph-based Word Interaction on the model, which makes each node representation influenced by neighbor nodes. The attention weight matrix of the output results of Graph Interaction Layer is visualized by using a heat map. It is worth noting that in this case study, the darker the color in the heat map, the higher the attention weight.



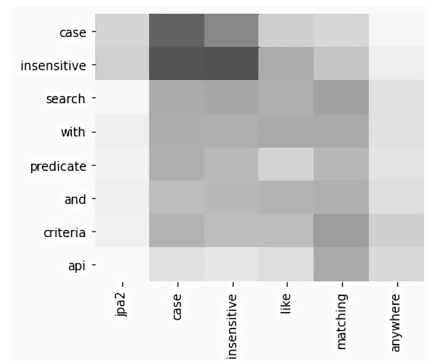
(a)



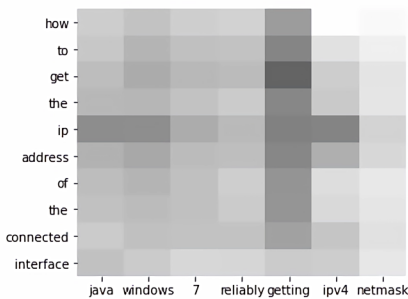
(b)



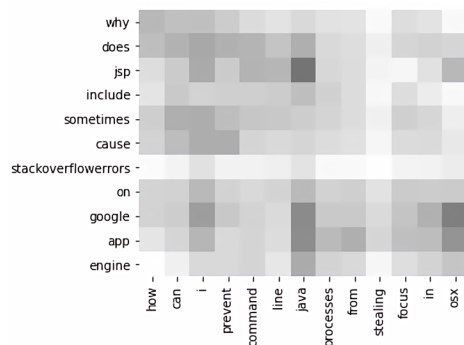
(c)



(d)



(e)



(f)

Fig. 6 Visualization results of the attention weight matrix for the example S_a and S_b . The darker the color, the larger the value. S_a is on the y-axis and S_b is on the x-axis

For the example in Fig. 6. (a), there are two input sentences S_a : "How do I filter out questions answers about bernie sanders trump and hillary clinton form my feed" and S_b : "How do I filter out trump and hillary related form my feed", where the set of co-occurring words is {how, do, I, filter, out, trump, and, hillar, form, my, feed}. It can be seen in the diagram that most of these words are aligned. Since our interaction diagram takes into account

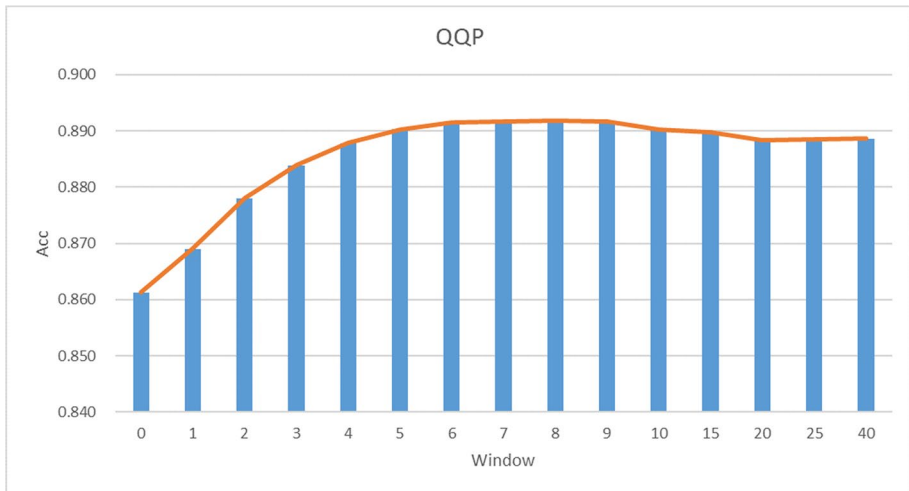
the phrase-level relations, the phrases between the two sentences here include "how do I filter out", "trump and hillary", "form my feed ", which are aligned in a block form in the diagram. Therefore, our method proves that these two sentences are paraphrased. This is consistent with our hypothesis. For the example in Fig. 6 (b), there are two input sentences S_a : "what makes eb 5 the best way to get a green card" and S_b : "how can following a vegan diet lead to diarrhea". where the co-occurring words are {a, to}, which are aligned, but there are no other aligned words or phrases between the two sentences. In addition, examples of each class in the SKU are visualized as follows: duplicate question in Fig. 6. (c); direct question in Fig. 6. (d); indirect question in Fig. 6. (e); isolated question in Fig. 6. (f).

4.6 Parameter sensitivity

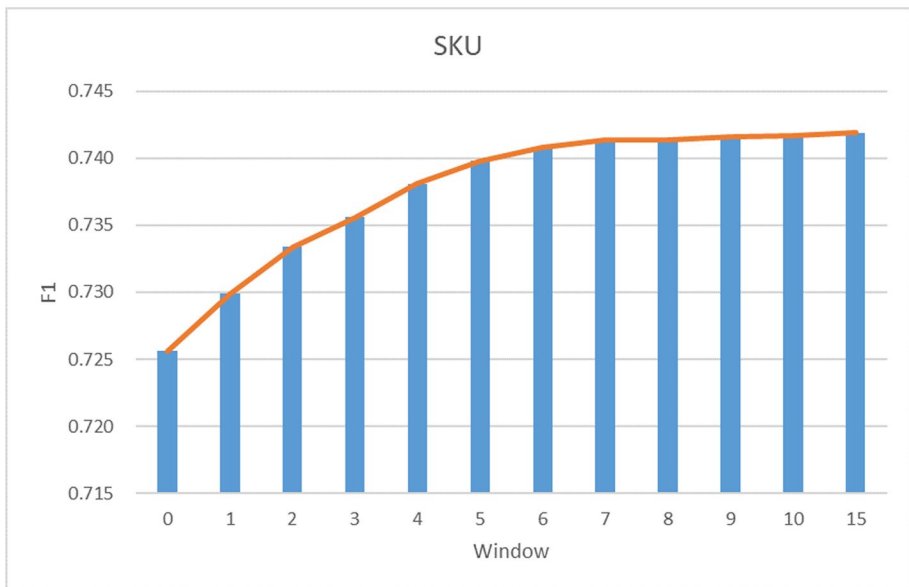
In this section, we evaluate the impact of sliding windows on GIMM. We validate it in the experimental setting of Quora Question Pairs and SKU-Title. The maximum length of the two input sentences is set to 59 after counting all samples of Quora Question Pairs. The maximum length of the two input sentences is set to 15 after counting all samples of Title in SKU data. We evaluate the impact of sliding windows on short texts with the results in Fig. 7(a) and on medium-length texts with the results in Fig. 7(b).

With other parameters set to default values, we first consider whether the sliding window size has an optimal choice when the GCN number of layers is 1. As shown in Fig. 7, on the SKU dataset, the model achieves the best performance when the sliding window size is 15. On the Quora Question Pairs dataset, the model achieves the best performance when the sliding window size is 8. By experimenting with different window lengths, we once again confirmed the positive impact of the graph interaction layer on the model. Setting a fixed-size sliding window to collect co-occurrence information is effective in utilizing global word co-occurrence information. Next, we considered the impact of the GCN number of layers on the model when the sliding window size is set to the optimal value. According to the results in Table 8, the model achieves the best performance when the GNN layer is 1. Increasing the GNN layer cannot capture more information and may even make model training more difficult.

In addition, we conducted a more in-depth analysis of the impact of the sliding window on the model. The sliding window affects our construction of the interaction graph. As can be seen in the Fig.7, when the window is set to 0, our interaction graph will have no edges. It is equivalent to removing the Graph Interaction Layer. SKU-Title's F1 value dropped to 0.7250, down 1.7%, and Quora Question Pairs' ACC value dropped by 3.1%. It is well illustrated that the layer has a positive influence on the PI task. Meanwhile, as the length of the window becomes larger, the performance of the model reaches a high value and decreases slowly after reaching a certain value. This is a very interesting phenomenon. This shows that if the value of the window is set large enough, a word in the sentence can find all words in the sentence that are identical to it. When there is a phenomenon that a word corresponds to multiple co-occurring words, there is a certain impact on the performance of the model. While the F1 value of SKU-Title becomes larger with the length of the window, after reaching a certain value, the performance of the model reaches a high value and then remains stable. This is because the text length of SKU-Title is short, and generally the window is set to an intermediate value to obtain all co-occurring word pairs. Secondly, the characteristics of the data in the SKU dataset mentioned earlier, the co-occurring word pairs between the two KUs are mainly in terminology and some humped variables, which again validates why GIMM achieves good performance on the SKU dataset. SKU has a



(a)



(b)

Fig. 7 (a) Impact of sliding window on Quora Question Pairs (b) Impact of sliding window on SKU-Title

smaller number of co-occurring word pairs compared to Quora, so the F1 value of SKU does not drop as much as Quora as the window becomes larger.

In summary, we have again verified the positive effect of Graph Interaction Layer on the model by experimentally varying the length of the window. To make use of global word co-occurrence information, it is effective to set a fixed size sliding window to collect co-occurrence information. Therefore, setting a proper window can help improve the model performance.

5 Conclusion

In this paper, we explore a deep learning model based on graph interaction matching for PI tasks. The model considers the interaction structure between sentence pairs and at the single-sentence phrase level, combined with the GCN to capture the rich matching information between sentence pairs. The experimental results show that our model can go beyond the baselines. We also discuss the important components that help GIMM achieve good performance and also came to some meaningful conclusions. (1) When there are a large number of co-occurring word pairs in two sentences, the construction of the interaction graph is densely connected, which leads to the structural information being ambiguous during message passing. This has a bad impact on the model performance. (2) Character embedding can provide additional information for domain-specific technical terms or some fixed collocations that word embedding cannot handle out-of-vocabulary words (OOV). It helps the model to understand special words which often determine whether a special domain sentence pair is interpreted or not. (3) Setting an appropriate sliding window allows each word in a sentence to be connected to an adjacent word in another sentence that may share the relevant contextual meaning. As well, the dense connection of graphs can be avoided. It helps to improve the model performance.

In our experimental evaluation of GIMM, we find that GIMM also has few limitations. In the future work, we will make improvements in the direction of interaction graph construction, edge selection, etc. Improve the quality of interaction graphs to face the challenge of GIMM on dense graphs. Also, graph-based integration layers may be applicable to other tasks of sequence matching. This layer can be applied to Machine Translation, Natural Language Inference, Question Answering and other similar scenarios. In the meantime, we will explore more on GCNs. Finally, due to the rich variability of natural language. These phenomena occur when two pairs of sentences share almost all words but (1) differ in a keyword (2) differ in word order, resulting in semantic inconsistency between the pairs. It means that simple methods involving only the word-to-word matching between sentences and neglecting the higher-level abstractions will fail to capture the question semantics and falsely label some question pairs. It is also an interesting work.

Funding This work was supported by the Science Foundation of Young and Middle-aged Academic and Technical Leaders of Yunnan under Grant No. 202205AC160040; Science Foundation of Yunnan Jinzhi Expert Workstation under Grant No. 202205AF150006; Major Project of Yunnan Natural Science Foundation under Grant No. 202302AE09002003; Science and Technology Project of Yunnan Power Grid Co., Ltd. under Grant No. YNKJXM20222254; the Postgraduate Research and Innovation Foundation of Yunnan University under Grant No. 2021Z112; Science Foundation of “Knowledge-driven intelligent software engineering innovation team”.

Data Availability The data associated with our study can be made available upon request. Please contact the corresponding author Xuan Zhang (zhxuan@ynu.edu.cn).

Declarations

Conflicts of interest/Competing interests We confirm that this work is original and has either not been published elsewhere, or is currently under consideration for publication elsewhere. None of the authors have any competing interests in the manuscript.

Consent to participate All the authors are aware of this submission. They have reviewed and consented to participate in this journal submission.

References

1. Xiao Han (2020) Hungarian layer: A novel interpretable neural layer for paraphrase identification. *Neural Netw* 131:172–184
2. Callison-Burch, C, Koehn P, Osborne M (2006) Improved statistical machine translation using paraphrases. In *Proceedings of the human language technology conference of the NAACL, Main Conference*, pp 17–24
3. Wallis P (1993) Information retrieval based on paraphrase. In *Proceedings of pacling conference*, pp 118–126
4. Das Arijit, Saha Diganta (2022) Deep learning based Bengali question answering system using semantic textual similarity. *Multimedia Tools Applic* 81(1):589–613
5. Lukashenko R, Graudina V, Grundspenkis J (2007) Computer-based plagiarism detection methods and tools: an overview. In *Proceedings of the 2007 international conference on computer systems and technologies*, pp 1–6
6. Zhang Yun et al (2015) Multi-factor duplicate question detection in stack overflow. *J Comput Sci Technol* 30(5):981–997
7. Ahasanuzzaman M et al (2016) Mining duplicate questions of stack overflow. In *2016 IEEE/ACM 13th working conference on mining software repositories (MSR)*, IEEE, pp 402–412
8. Hoogeveen D, Verspoor KM, Baldwin T (2015) CQADupStack: A benchmark data set for community question-answering research. In *Proceedings of the 20th australasian document computing symposium*, pp 1–8
9. Nakov P et al (2016) SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pp 525–545
10. Gong Y, Luo H, Zhang J (2018) Natural language inference over interaction space. In *International conference on learning representations*, pp 1–15
11. Chen Q et al (2017) Enhanced LSTM for natural language inference. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long Papers)*, pp 1657–1668
12. Wang Z, Hamza W, Florian R (n.d.) Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th international joint conference on artificial intelligence (IJCAI-17)*, pp 4144–4150
13. Arase Yuki, Tsujii Junichi (2021) Transfer fine-tuning of BERT with phrasal paraphrases. *Comput Speech Lang* 66:101164
14. Palivela Hemant (2021) Optimization of paraphrase generation and identification using language models in natural language processing. *Int J Inform Manag Data Insights* 1(2):100025
15. Mueller J, Thyagarajan A (2016) Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI conference on artificial intelligence* 30(1):2786–2792
16. Wang Z, Mi H, Ittycheriah A (2016) Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pp 1340–1349
17. Yang R, Zhang J, Gao X et al (2019) Simple and effective text matching with richer alignment features[C]. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp 4699–4709
18. Yu Chuanming et al (2021) A simple and efficient text matching model based on deep interaction. *Inform Proc Manag* 58(6):102738
19. Xint Y et al (2021) Label incorporated graph neural networks for text classification. In *2020 25th international conference on pattern recognition (ICPR)*, IEEE, pp 8892–8898
20. Liu Y et al (2021) Deep attention diffusion graph neural networks for text classification. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp 8892–8898
21. Luo Y, Zhao H (2020) Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp 6408–6418
22. Qu M et al (2020) Few-shot relation extraction via bayesian meta-learning on relation graphs. In *International conference on machine learning*, PMLR, pp 7867–7876
23. Wu W et al (2021) BASS: Boosting abstractive summarization with unified semantic graph. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp 6052–6067
24. Peng H et al (2018) Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pp 1063–1072
25. Wu Zonghan et al (2020) A comprehensive survey on graph neural networks.". *IEEE Trans Neural Netw Learn Syst* 32(1):4–24

26. Wang Quan et al (2017) Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans Knowl Data Eng* 29(12):2724–2743
27. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163
28. Xu K et al (2018) How powerful are graph neural networks? In International conference on learning representations, pp 1–17
29. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. In International conference on learning representations, pp 1–14
30. Zhang Y, Yu X, Cui Z et al (2020) Every document owns its structure: Inductive text classification via graph neural networks. In Proceedings of the 58th annual meeting of the association for computational linguistics, pp 334–339
31. Peng H, Li J, Wang S et al (2019) Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification[J]. *IEEE Trans Knowl Data Eng* 33(6):2505–2519
32. Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on artificial intelligence 33(01):7370–7377
33. Tekli Joe et al (2018) Full-fledged semantic indexing and querying model designed for seamless integration in legacy RDBMS.". *Data Knowl Eng* 117:133–173
34. Tekli Joe (2016) An overview on xml semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges. *IEEE Trans Knowl Data Eng* 28(6):1383–1407
35. Viji D, Revathy S (2022) A hybrid approach of Weighted Fine-Tuned BERT extraction with deep Siamese Bi-LSTM model for semantic text similarity identification. *Multimed Tools Appl* 81(5):6131–6157
36. Lai H et al (2020) Bi-directional attention comparison for semantic sentence matching. *Multimedia Tools Applic* 79(21):14609–14624
37. Serban I et al (2016) Building end-to-end dialogue systems using generative hierarchical neural network models. In Proceedings of the AAAI conference on artificial intelligence 30(1):3776–3783
38. Ramos J (2003) Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning vol. 242(1)
39. Robertson S, Zaragoza H (2009) The probabilistic relevance framework: BM25 and beyond. *Now Publishers Inc.* 3(4):333–389
40. Yujian Li, Bo Liu (2007) A normalized Levenshtein distance metric. *IEEE Trans Pattern Anal Mach Intell* 29(6):1091–1095
41. Huang P-S, et al (2013) Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on information & knowledge management, pp 2333–2338
42. Feng M et al (2015) Applying deep learning to answer selection: A study and an open task. In 2015 IEEE Workshop on automatic speech recognition and understanding (ASRU), IEEE, pp 813–820
43. Conneau A et al (2017) Supervised learning of universal sentence representations from natural language inference data. In Proceedings of the 2017 Conference on empirical methods in natural language processing, pp 670–680
44. Wang H et al (2021) Knowledge-guided paraphrase identification. Findings of the association for computational linguistics: EMNLP 2021, pp 843–853
45. K Leilei et al (2020) A deep paraphrase identification model interacting semantics with syntax. *Complexity* 2020:1–14
46. Mohamed Muhidin, Oussalah Mourad (2020) A hybrid approach for paraphrase identification based on knowledge-enriched semantic heuristics. *Lang Resources Eval* 54:457–485
47. Chen Qidong, Sun Jun, Zhao Yuan (2021) A Novel Architecture with Separate Comparison and Interaction Modules for Chinese Semantic Sentence Matching. *Neural Process Lett* 53:3677–3692
48. Chang G, Wang W, Hu S (2022) MatchACNN: A multi-granularity deep matching model. *Neural Process Lett* 1–20
49. Pang L et al (2016) Text matching as image recognition. In Proceedings of the AAAI conference on artificial intelligence, vol. 30(1), pp 2793–2799
50. Ling W et al (2015) Finding function in form: Compositional character models for open vocabulary word representation. In Proceedings of the 2015 Conference on empirical methods in natural language processing, pp 1520–1530
51. Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation. In Conference on empirical methods in natural language processing, pp. 1520–1530
52. Hochreiter Sepp, Schmidhuber Jürgen (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780

53. Parikh AP et al (2016) A decomposable attention model for natural language inference. In Proceedings of the 2016 conference on empirical methods in natural language processing (EMNLP), pp 2249–2255
54. Mou L, Men R, Li G et al (2016) Natural language inference by tree-based convolution and heuristic matching[C]. In Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers), pp 130–136
55. Minaee S, Kalchbrenner N, Cambria E et al (2021) Deep learning–based text classification: a comprehensive review[J]. *ACM Comput Surv (CSUR)* 54(3):1–40
56. Nikolentzos G, Tixier A, Vazirgiannis M (2020) Message passing attention networks for document understanding[C]. *Proc AAAI Conf Artif Intell* 34(05):8544–8551
57. Shankar I, Nikhil D, Kornel C (2017) First quora dataset release: Question Pairs. In <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>
58. Amirreza S et al (2019) Question relatedness on stack overflow: the task, dataset, and corpus-inspired models. In Proceedings of the AAAI reasoning for complex question answering workshop, pp 1–9
59. Chen D, Fisch A, Weston J et al (2017) Reading wikipedia to answer open-domain questions. In 55th annual meeting of the association for computational linguistics, ACL 2017, pp 1870–1879

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.