Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 4186750, 12 pages
https://doi.org/10.1155/2021/4186750

Research Article

Exploiting Syntactic and Semantic Information for Textual Similarity Estimation

Jiajia Luo,[1] **Hongtao Shan**,[1] **Gaoyu Zhang**,[2] **George Yuan**,[3] **Shuyi Zhang**,[4] **Fengting Yan**,[1] **and Zhiwei Li**[1]

[1]*School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China*
[2]*School of Information Management, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China*
[3]*School of Financial Technology, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China*
[4]*Lixin Research Institute, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China*

Correspondence should be addressed to Hongtao Shan; shanhongtao@sues.edu.cn

Received 23 June 2020; Revised 19 December 2020; Accepted 13 January 2021; Published 23 January 2021

Academic Editor: Massimiliano Ferrara

The textual similarity task, which measures the similarity between two text pieces, has recently received much attention in the natural language processing (NLP) domain. However, due to the vagueness and diversity of language expression, only considering semantic or syntactic features, respectively, may cause the loss of critical textual knowledge. This paper proposes a new type of structure tree for sentence representation, which exploits both syntactic (structural) and semantic information known as the weight vector dependency tree (WVD-tree). WVD-tree comprises structure trees with syntactic information along with word vectors representing semantic information of the sentences. Further, Gaussian attention weight is proposed for better capturing important semantic features of sentences. Meanwhile, we design an enhanced tree kernel to calculate the common parts between two structures for similarity judgment. Finally, WVD-tree is tested on widely used semantic textual similarity tasks. The experimental results prove that WVD-tree can effectively improve the accuracy of sentence similarity judgments.

1. Introduction

The measurement of the similarity between a pair of sentences is a fundamental and essential task in a wide range of NLP applications, such as question and answer system (Q&A system) [1, 2], information retrieval [3], and text classification [4]. Because of the vagueness and diversity of language expression, the measurement faces a certain amount of technical challenges. For instance, a pair of sentences with many same words may represent different meanings, respectively.

Recently, the emergence of word embedding techniques [5], which map a word into a numerical vector, results in many methods achieving success via sentence embeddings in textual similarity tasks [6, 7]. For example, Tien et al. [8] encoded many features from various sets of word embeddings into one embedding and secondly learned similarity between sentences via the new embedding. Xing et al. [9] made use of a word embedding method to find high-frequency phenotypes used as input in a sentence embedding method. The core idea behind these works is to identify semantically related terms in both sentences and to bring these similarities together to aggregate an overall similarity. But the disadvantage is that the process of calculating sentence similarity is superficial and does not delve into the relationships between words in sentence. Gradually, works have turned to model with distributed representations and neural network architectures. Yao et al. [10] took advantage of a convolutional neural network (CNN) and word embedding to achieve sentence embeddings. Palangi et al. [3] designed a model utilizing recurrent neural networks (RNN) with long short-term memory (LSTM) cells to perform document retrieval. These methods of neural network have truly improved performance in the textual similarity task. However, the training processes of these methods are

often time-consuming, and the syntactic features of sentences often cannot be fully exploited. Our research follows some typical features with theirs: (1) it also employs word embedding technology to capture semantic information and (2) the method also performs well on the issues related to sentence similarity.

Moreover, some studies have added the attention weight mechanism to further improve performance [11, 12]. Since there are many successful applications of attention mechanism in machine translation [13, 14], attention mechanism has been widely applied in the natural language processing domain. Chen et al. [15] proposed a new network to pay attention to the production of the hidden state of one sentence with the help of the other sentence's hidden states and attention information. In this paper, we design a novel attention weight that is more sensitive to "distance" to improve the performance of the attention mechanism.

Another line of works that discussed tree kernels is also associated with our work because our model encodes syntactic-semantic information represented by tree structures. Some attempts have proven the advantages of tree structures that capture syntactic features of sentences [16, 17]. Li et al. [18] proposed an algorithm based on a syntactic structure for textual similarity, which analyzed the sentence element and then transformed sentence similarity into word similarity through analysis. Gao et al. [19] presented a tree-to-tree method of Chinese machine translation based on subtree alignment. However, these proposed tree structures lacked semantic features consideration. As for the study of the tree kernel, Aiolli et al. [20] described an efficient algorithm for injecting positional information into a tree kernel and presented ways to enlarge its feature space. Rieck et al. [21] proposed an effective approximation technique for parse tree kernels. Nevertheless, few tree kernel methods are applied to textual similarity tasks.

Based on the inheritance of existing tree structure, tree kernel calculation method, and attention mechanism, this paper develops a novel sentence modeling approach that incorporates semantic and syntactic information. To obtain a sentence representation with multiple features, we propose a structure called WVD-tree, which treats the sentence pairs as input objects. WVD-tree integrates the word vector information into the tree structure. Moreover, Gaussian attention weight is proposed to assign a weight value to each word for better highlighting the important semantic features of sentences. To estimate the similarity between structure trees, we design an enhanced tree kernel tailored for WVD-tree. Our method is tested on STS tasks and gets satisfactory performances. Our method's advantages are the following: (1) it can be used as a general architecture because more powerful techniques can replace the integration techniques; (2) it avoids time-consuming training, different from some neural network methods.

In summary, this paper's main contributions are as follows: (1) In this paper, the WVD-tree is proposed to connect semantic and syntactic information, which can be viewed as sentence representation. (2) Considering that the features of different words in a sentence should be treated differently, we design a novel Gaussian attention weight for better capturing semantic information. (3) For similarity judgment, the enhanced tree kernel is proposed based on the traditional tree kernel and calculates the tree structures' common fragments.

In the remainder of this paper, Section 2 describes several main techniques used in the architecture. Section 3 introduces the proposed model. Section 4 shows the experimental results. Section 5 analyzes the experiments. Finally, Section 6 summarizes the conclusions.

2. Background

This section briefly reviews several main techniques to facilitate understanding of this paper's model, including word embedding, attention mechanism, dependency tree, and tree kernel.

2.1. Word Embedding. At present, continuous vectors are often used to capture the hidden semantic features of sentences. Generally, this technique is known as word embedding [22]. We can map a word into a numerical vector called word vector. If words have a similar meaning, they will be mapped to a similar vector space position. For example, "good" and "great" are close to each other, whereas "good" and "apple" are distant. The above phenomenon means that the word embedding technique can be directly applied to measure the semantic similarity for word pairs or find the word that is most similar to the target word. However, people focus more on studies of the similarity between two sentences or passages. The sentences are embedded via word embedding technique to determine the similarity between texts. Precisely, we can compute the cosine similarity between numerical vectors of words. Then we add up the results and take the average. Finally, we get the similarity between the sentences. The process of semantic similarity computation is shown in Figure 1. In the correlation matrix graph, a word pair has a high correlation if the color between them is dark. It means the words are closer to each other. For example, the similarity score between "boy" and "child" is 0.6, while that between "boy" and "hitting" is 0.083. This phenomenon illustrates that "boy" and "child" have a relatively similar vector space position. The final similarity score can be calculated as follows:

$$\mathrm{SIM}s_1, s_2 = \frac{1}{l(s_1) + l(s_2)} \sum_{i=1}^{l(s_1)} \sum_{j=1}^{l(s_2)} \frac{w_{1i} \cdot w_{2j.}}{w_{1i} w_{2j}}, \quad (1)$$

where s_1 and s_2 are sentence one and sentence two, respectively; w_{1i} and w_{2j} are the i^{th} and j^{th} word vectors of s_1 and s_2, respectively; and $l(s_1)$ and $l(s_2)$ are lengths of s_1 and s_2, respectively.

2.2. Attention Mechanism. The aim of introducing the attention mechanisms [11] for the sentence similarity measurement is to pay more attention to some critical words instead of each word being treated equally in the processing of sentence features. So far, much research has proven that the attributes of words affect human reading efficiency [23]. Therefore, researchers believe that different words should be

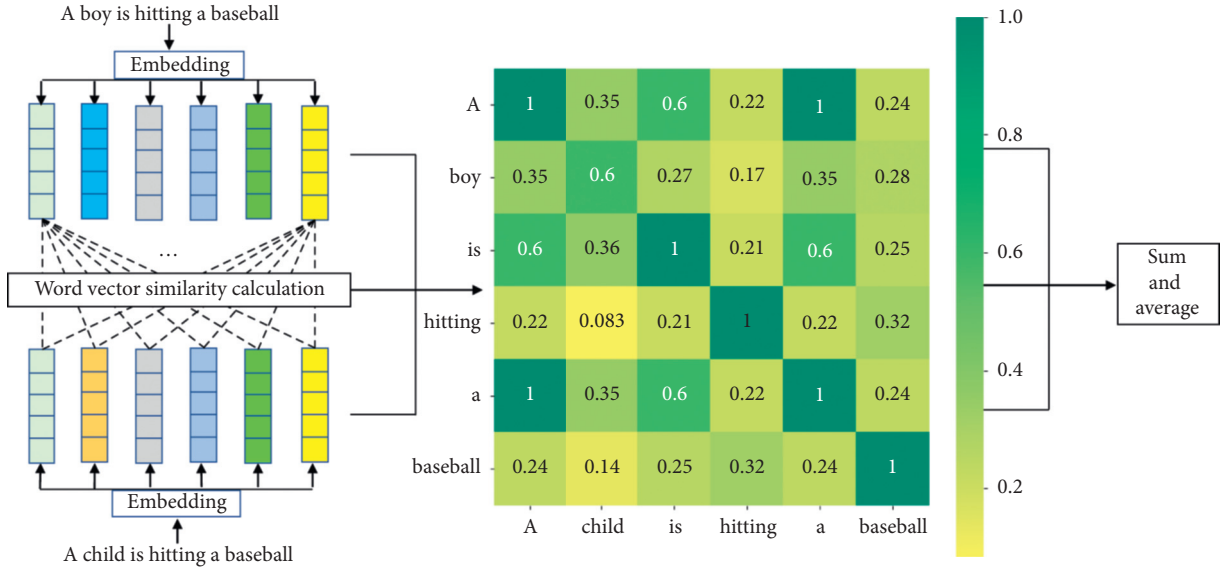

Figure 1: The process of computing semantic similarity between sentences.

assigned different weights. To propose a new attention weight based on traditional attention weight value, we introduce TF-IDF [24] to obtain the original attention weights of words. The original weight value can be calculated as follows:

$$\text{TF} - \text{IDF} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right), \tag{2}$$

where $tf_{i,j}$ means the number of occurrences of the word i in document j; df_i means the number of documents containing the word i; N means the total number of documents in a corpus.

2.3. Dependency Tree. The foundational feature of the dependency tree [25, 26] is that we view sentence structure in terms of the dependency relation. A sentence is represented as a tree with words at leaf nodes and their dependency-relation tags at the nonterminal nodes. Internal nodes are labeled by the nonterminal category, while leaf nodes are labeled by the terminal category. A simple explanation is that a sentence can be represented as a dependency tree. Each leaf node represents a word in a sentence, while each nonterminal node represents a dependency relation between two words. Every node in the dependency tree can be classified as a root node, a branch node, a nonbranch node, or a leaf node. Within a dependency tree, there is only one root node. A branch node links two or more other nodes. A nonbranch node links only one node. A leaf node is a terminal node that does not control other nodes. For example, in the sentence "A boy is hitting a baseball," firstly, Figure 2(a) displays the dependency-relation of a sentence; subsequently, Figure 2(b) shows the traditional dependency tree constructed by the Stanford parser tool. Some abbreviations are listed in Table 1.

2.4. Tree Kernel. A representative tree kernel, known as a partial tree kernel (PTK) [27], is used to fully exploit dependency trees. The kernel that can be considered represents trees in terms of their substructures (fragments). Tree kernel aims to measure the number of common substructures between a pair of trees T_1 and T_2 without taking into account the whole fragment space. The tree kernel can effectively and automatically extract meaningful fragments. A general tree kernel function framework has been designed to compute the common fragments as follows:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \tag{3}$$

where N_{T_1} and N_{T_2} are the sets of nodes in the tree T_1 and tree T_2; n_1 and n_2 are two nodes from two trees, respectively. $\Delta(n_1, n_2)$ function enriches the diversity of kernel space because it can be designed to achieve different performance.

The kernel detects whether a tree subset (common to both trees) belongs to the feature space that PTK intends to produce. For this purpose, the desired substructures need to be described. For syntactic parse trees, each node with its children is related to a grammar production rule. The tree kernel in turn generates a tree subset composed of each node and its child nodes according to grammar production rule. Then tree kernel function requires to calculate the similarity between two subsets from different trees. The evaluation of the common substructures rooted in nodes n_1 and n_2 needs the selection of the shared child subsets of the two nodes; for example, [VP [V DT NP]] and [VP [DT NP NP]] have [VP [NP]] (2 times) and [VP [DT NP]] in common.

The PTK computes common fragments that are shared between the two trees. The considered fragments are called partial trees (PTs) in the calculation of structure tree similarity. PTs refer to a node with its partial children (subtree structure can be incomplete, meaning that partial children

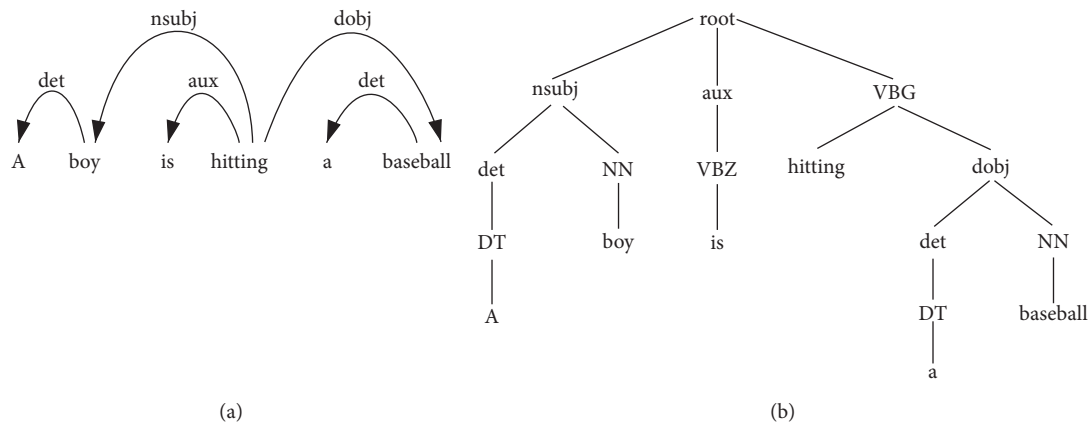

Figure 2: The dependency-relation and the dependency tree (DT). (a) Dependency-relation of a sentence. (b) An example of a dependency tree.

Table 1: Abbreviation explanations in the dependency tree.

Dependency-relation abbreviations	Explanations	Part-of-speech tags abbreviations	Explanations
nsubj	Nominal subject	NN/N	Noun
dobj	Direct object	DT/*D*	Determiner
aux	Auxiliary	VBZ	Verb, past tense
det	Determiner	VBG	Verb, gerund or present participle

are allowed). For example, Figure 3 displays a tree with its 10 PTs. PTs can be generated by the partial production rule of the grammar. Consequently, [NP [D]] and [NP [N]] are valid PTs. If we follow the production rule that the chain of fragments structure cannot be broken, sequentially, [NP[D [a]], NP[N[cat]]] is the whole valid fragment. Owing to the partial production rule, PTK processes more fragments than other tree kernel methods when calculating similarity. Specifically, some common fragments rooted at the n_1 and n_2 nodes, and the $\Delta(n_1, n_2)$ function on the partial tree can be calculated as follows.

If the node labels of n_1 and n_2 are different, then $\Delta(n_1, n_2) = 0$.

Else,

$$\Delta(n_1, n_2) = \mu\left(\lambda^2 + \sum_{p=1}^{l_m} \Delta_p(c_{n1}, c_{n2})\right), \tag{4}$$

where μ and λ are two decay factors; c_{n_1} and c_{n_2} refer to list of children nodes of n_1 and n_2; $l_m = \min\{\text{length}(c_{n_1}), \text{length}(c_{n_2})\}$; $\Delta_p(\cdot)$ refers to the number of common subsequences whose length is p; and $n_1 = n_2$ means that the labels of n_1 and n_2 are the same.

3. Model Description

The WVD-tree of a sentence is constructed based on knowledge integration of word vector, attention mechanism, and dependency-relations. This type of tree can be viewed as semantic-syntactic sentence representation, which contains information about the weight distribution of words in a sentence. For a sentence pair, they are firstly transformed into two WVD-trees. Subsequently, an enhanced tree kernel (improves based on the existing partial tree kernel) is designed to calculate the similarity between two structure trees. The process of the WVD-tree similarity calculation is shown in Figure 4.

3.1. Gaussian Attention Weight. The traditional attention mechanism calculates the original attention weight value. However, it cannot capture the local structure of texts. This problem is improved with a Gaussian probability. Our method is inspired by the following observation: the neighboring words tend to have a higher semantic contribution to the central word than distant words. However, the traditional attention mechanism is not very sensitive to "distance." This phenomenon means that the same words at different distances are treated almost equally. As shown in Figure 5, there are two "new" words in the sentence "I bought a new book with a new friend," but only the first "new" is meaningful for the current word "book." However, the traditional attention mechanism assigns the same weight to the two "new" words, as shown in Figure 5(a). This paper's idea is that the attention mechanism should be encouraged to provide greater weight to neighboring words around the central word. For example, in Figure 5, the central word is "book" because it has the highest weight value in a sentence. Therefore, the original weight value calculated by the traditional attention mechanism should multiply Gaussian distribution according to the adjacent position, as shown in Figure 5(b). The weight distribution of the original attention weight is changed to capture the sentence's semantic information more effectively, as

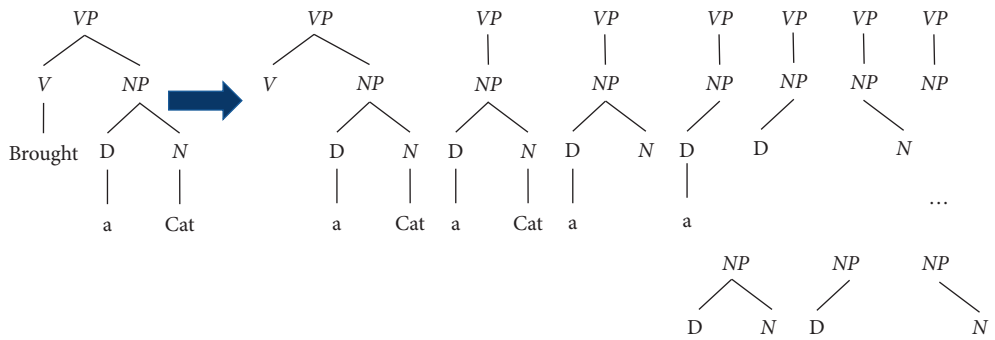

Figure 3: A tree with some of its partial trees (PTs).

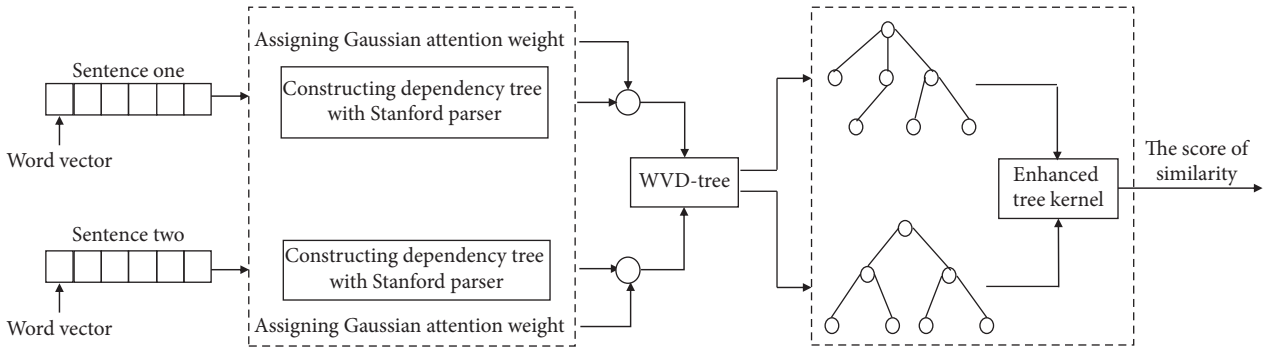

Figure 4: The process of WVD-tree similarity calculation.

shown in Figure 5(c). This new attention mechanism is called Gaussian attention weight. Experiments prove that Gaussian attention weight works better than the traditional attention mechanism.

For simplicity, this paper uses the standard normal distribution whose probability density function is $G(d) = e^{-\pi d^2}$, where d is the random variable. d is 0 when the word has the highest weight value. This paper introduces a scalar variable ω to Gaussian probability function to loosen the restriction, calculated as equation (5). This paper also presents a parameter b to the weight of the central word attending itself, which can be calculated as in equation (6). Thus, the corrected Gaussian attention weight value G_{weight} can be calculated as in equation (7):

$$G(d) = e^{-|\omega d^2|}, \tag{5}$$

$$G(d) = e^{-|\omega d^2 + b|}, \tag{6}$$

$$G_{\text{weight}} = \text{TF} - \text{IDF} \times \text{softmax}(G(d)), \tag{7}$$

where $|\cdot|$ represents the absolute value; $\omega > 0$ and $b \leq 0$ are scalar parameters; TF-IDF can refer to equation (2); $\text{softmax}(\cdot)$ is a normalized function.

3.2. WVD-Tree. The new way of sentence representation is similar to the traditional dependency tree in some ways; for example, (1) it also has only one root node; (2) the internal nodes are labeled by some abbreviations, for instance, "DT" and "NN"; and (3) every leaf node includes a word from the sentence. The change is that all leaf nodes in the WVD-tree also include two other factors: one is a word vector and the other is the Gaussian attention weight value. The process of constructing a WVD-tree has several essential steps. Firstly, every word in the sentence is assigned a "part of speech"; for example, the word "boy" is classified as "noun," and the word "hitting" is classified as "verb." Secondly, the word vector of the corresponding word is obtained from the pretrained word embedding; meanwhile, each word is given a Gaussian attention weight to distinguish its importance with others. Thirdly, it is also essential to analyze the relationships among nodes. For example, in the sentence "A boy is hitting a baseball," "direct object" is the relationship between "hitting" and "baseball"; "determiner" is the relationship between "a" and "baseball." Finally, all relations among nodes are connected to form a complete structure. As such, we can obtain the WVD-tree, as shown in Figure 6.

3.3. Enhanced Tree Kernel. As mentioned before, the tree kernel is used to compute the similarity between structured trees. The newly designed tree kernel is inspired by the traditional partial tree kernel. Like the previous tree kernel, a generic function framework is used (refer to equation (1)). The change is that vec_1 and vec_2 (word vectors of nodes n_1 and n_2) and wt_1 and wt_2 (Gaussian attention weights of nodes n_1 and n_2) are added to the $\Delta(n_1, n_2)$ function of tree kernel. The traditional tree kernel computation method considers the syntactic information of nodes but does not explore semantic information. Unlike the traditional tree

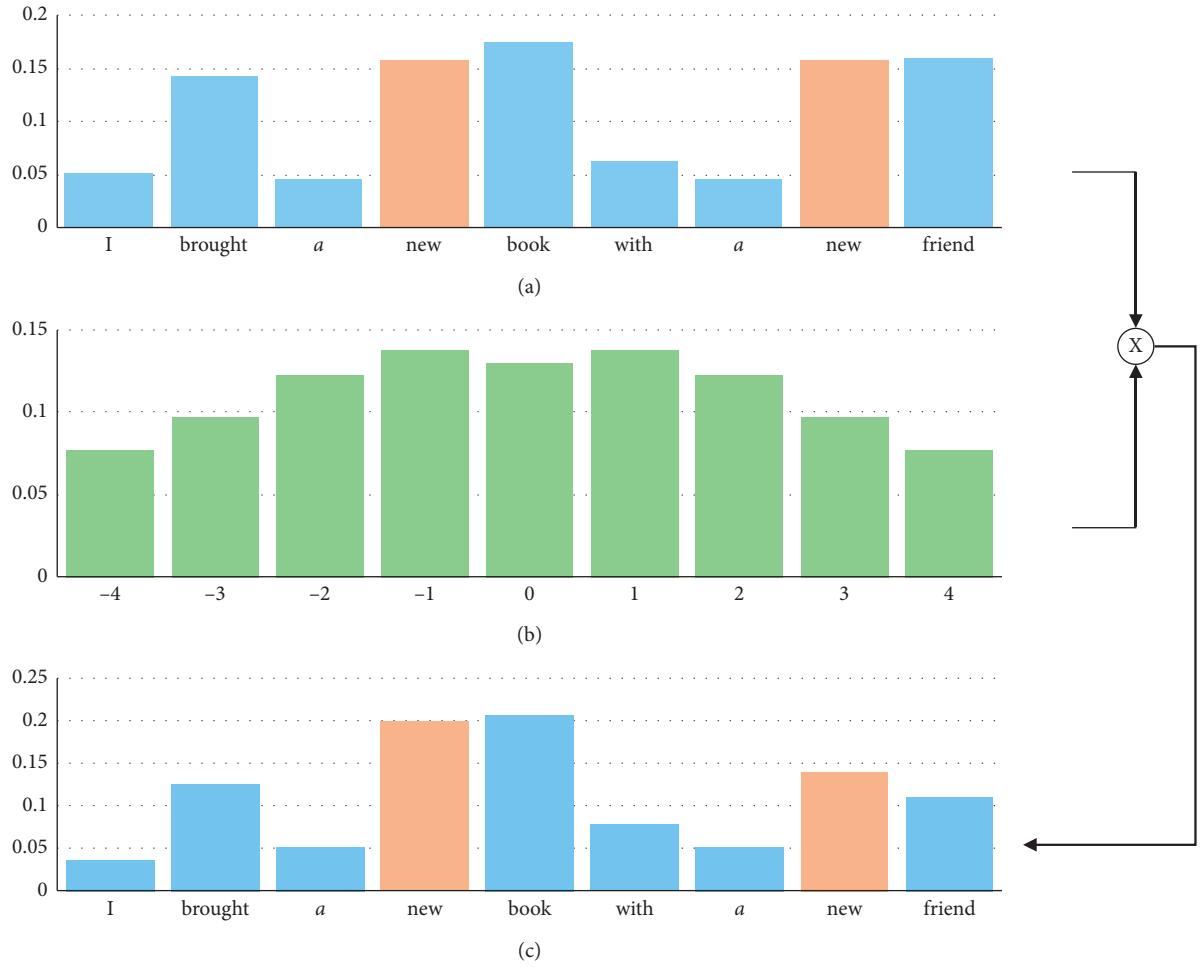

Figure 5: Attention weight values are corrected by Gaussian probabilities. (a) presents a traditional attention mechanism. The word "new" that appeared in different positions acquired the same contribution to sentence, which is inconsistent with our experience that adjacent words should be more critical. (b) describes the Gaussian distribution of the x-axis. (c) shows the attention value corrected by the Gaussian distribution, where the first "new" is more critical compared to the second "new."

kernel, we calculate the semantic similarity of leave nodes separately, instead of using the same function for all nodes, just like equation (4). The recursive function $\Delta_{\mathrm{ETK}}(n_1, n_2)$ of enhanced partial tree kernel is defined as follows:

$$\Delta_{\mathrm{ETK}}(n_1, n_2) = \begin{cases} 0, & n_1 \text{ or } n_2 \text{ is not leaf node, and } n_1 \neq n_2, \\ \mathrm{Att}_{\mathrm{weight}} \times \mathrm{SIM}(\mathrm{vec}_1, \mathrm{vec}_2), & n_1, n_2 \text{ are leaf nodes}, \\ \mu\left(\lambda^2 + \sum_{p=1}^{l_m} \Delta_p(c_{n1}, c_{n2})\right), & \text{otherwise}, \end{cases} \tag{8}$$

$$Att_{\mathrm{weight}} = w_{t1} \times w_{t2}, \tag{9}$$

where c_{n_1}, $c_{n_2}l(\mathrm{m})$, μ, and λ have the same meaning mentioned in equation (4); $\mathrm{SIM}(\cdot)$ function is designed to measure the cosine similarity between vectors; $n_1 \neq n_2$ means that the labels of n_1 and n_2 are different. For example, a label with a node of "NN" and a label with a "VP" do not match.

It remains to account for the way of calculating $\Delta_p(\cdot)$ according to the enhanced partial tree kernel. For the sake of understanding, consider $c_{n_1} = s_1 a$ and $c_{n_2} = s_2 b$ (a and b are the last children; s_1 and s_2 are subsequences of c_{n_1} and c_{n_2}, respectively); $\Delta_p(c_{n_1}, c_{n_2})$ can be solved by developing a "recursive" function as follows:

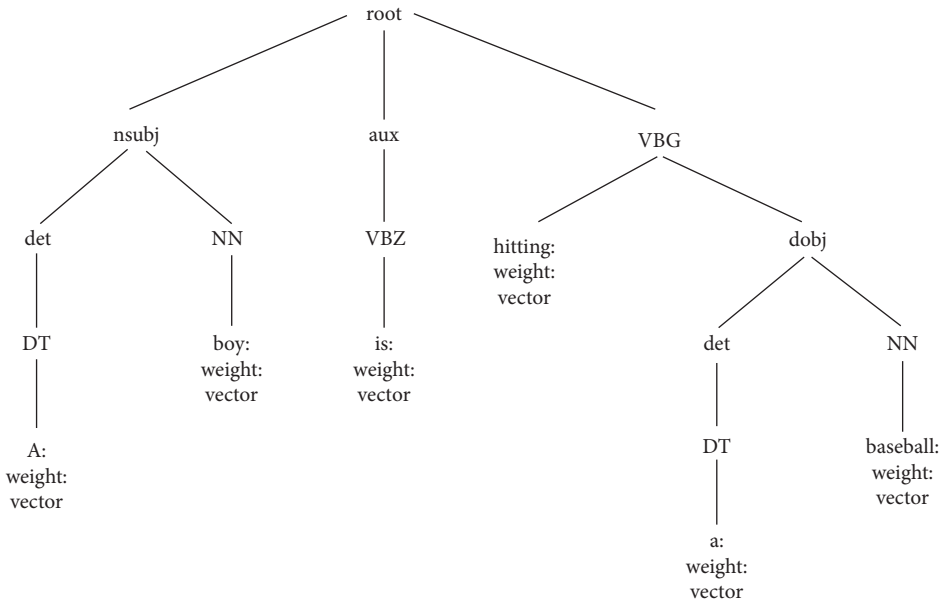

Figure 6: An example of the WVD-tree.

$$\Delta_p(s_1 a, s_2 b) = \Delta(a,b)\sum_{i=1}^{|s_1|}\sum_{j=1}^{|s_2|}\left(\lambda^{|s_1|-i+|s_2|-j}\times\left(\Delta_{p-1}(s_1[1{:}\ i], s_2[1{:}\ j])\right)\right), \tag{10}$$

where $s_1[1{:}\ i]$ (resp., $s_2[1{:}\ j]$) is the subsequence of s_1 (resp., s_2) ranging from 1 to i (resp., from 1 to j); $|s_1|$ (resp., $|s_2|$) is the length of s_1 (resp., s_2). Note that, here, $\Delta(a,b)$ is calculated using equation (8), while $\Delta_{p-1}(\cdot)$ is recursively calculated using equation (10), and the recursive process stops when it reaches the leaf node.

4. Results

4.1. Datasets. Following prior studies, the experiments are also conducted on semantic textual similarity (STS) datasets (2012–2015). These datasets cover various domains, for example, news, web forums, images, glosses, and Twitter. Statistics of STS datasets are sorted by year as shown in Table 2 (datasets with the same name have different data in different years).

4.2. Experimental Settings

4.2.1. Word Embedding Setting. This paper compares the most commonly used word embedding techniques, word2vec [21] and glove [27], for choosing high-quality word vectors. Here, WVD-tree uses the simple version of the enhanced tree kernel by setting $Att_{\text{weight}} = 1$, $\mu = 0.1$, and $\lambda = 0.1$ in equation (8). Considering that the numerical vector's key parameter is the vector dimension, which is set to 50, 100, 200, and 300, the compared results tested on the answers-forums dataset are shown in Figure 7. It can be found that dimension 300 gains the best performance. Based on this analysis, the number of dimensions in word embeddings is set as 300 throughout the experiment process. Meanwhile, it can be clearly seen that WVD-tree$_{\text{word2vec}}$ (use Gaussian weight word2vec-embedded dependency tree) performs better than WVD-tree$_{\text{glove}}$ (use Gaussian weight glove-embedded dependency tree). Thus, WVD-tree$_{\text{word2vec}}$ (with its vector dimension being 300) is selected as the better method to conduct experiments on all STS datasets (2012–2015).

4.2.2. Impact of μ and λ. The enhanced tree kernel function is related to two parameters, λ and μ. This section studies the impact of the two parameters on the accuracy of WVD-tree. Here, we use the simple version of enhanced tree kernel by setting Att_{weight} in equation (8). Firstly, we set $\lambda = 0.1$ by default while studying the parameter μ. As shown in Figure 8(a), $\mu = 0.3$ gets the best performance. Then, we set $\mu = 0.3$ while studying the parameter λ. As shown in Figure 8(b)), $\lambda = 0.2$ gets the best performance.

4.2.3. Impact of ω and b. The enhanced tree kernel function is also related to two other parameters, ω and b. The two parameters are derived from equation (6). They play an essential role in assigning Gaussian attention weight to each word. Firstly, we set $b = -0.1$ by default while studying the parameter ω. As shown in Figure 9(a), $\omega = 0.06$ gets the best performance. Then we set $\omega = 0.06$ while studying the parameter b. As shown in Figure 9(b), $b = -0.3$ gets the best performance.

4.2.4. Other Settings. Following previous research, the term frequency-inverse document frequency (TF-IDF) is introduced to produce original attention weights before being corrected by Gaussian probabilities. For calculation, each

Table 2: Statistics of the provided datasets for the SemEval semantic textual similarity tasks (2012–2015).

STS′12		STS′13		STS′14		STS′15	
Dataset	Sentence pairs	Dataset	Sentence pairs	Dataset	Sentence pairs	Dataset	Sentence pairs
MSRpar	750	Headlines	750	Deft-forum	450	Answers-forums	375
MSRvid	750	OnWN	561	Deft-news	300	Belief	375
SMTeuroparl	459	FNWN	189	Headlines	750	Images	750
OnWN	750			Images	750	Headlines	750
SMTnews	399			OnWN	750		
				Tweet-news	750		

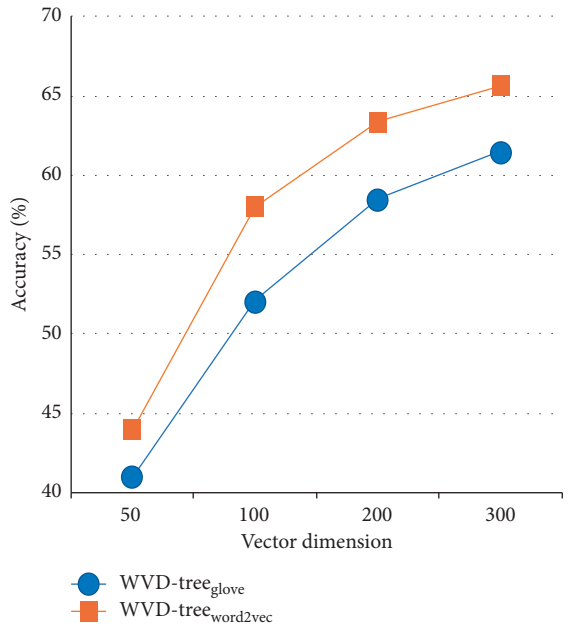

Figure 7: The performance of varying word embedding.

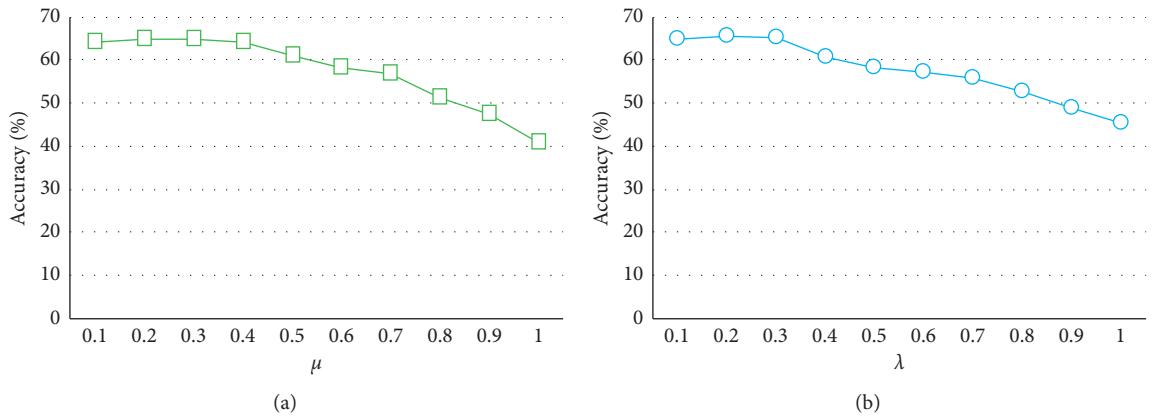

Figure 8: Varying μ and λ on the answers-forums dataset.

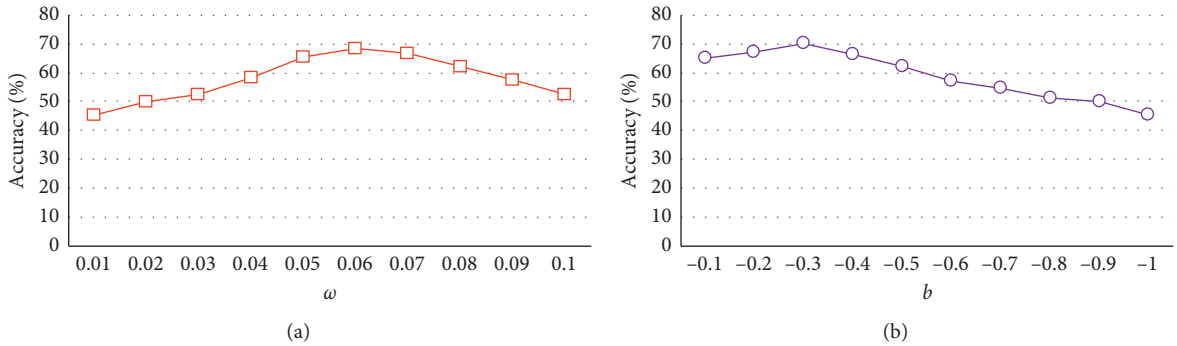

Figure 9: Varying ω and b on the answers-forums dataset.

Table 3: Comparison results of the proposed method and other state-of-the-art methods on the STS tasks (2012–2015).

Year	Dataset	Compared methods					The method
		Glove	Word2vec	RNN	LSTM	iRNN	WVD-tree
2012	MSRpar	47.8	49.2	18.6	16.1	43.5	**64.3**
	MSRvid	63.8	65.6	66.5	71.3	73.3	**80.6**
	SMTeuroparl	46.2	45.1	40.9	41.8	47.2	**51.7**
	OnWN	55.3	55.9	63.1	65.2	**70.2**	68.8
	SMTnews	50.3	52.3	51.3	51.0	**58.3**	53.2
2013	Headlines	63.9	65.5	59.5	57.4	72.7	**78.1**
	OnWN	49.1	52.4	54.6	50.4	69.3	**80.6**
	FNWN	34.3	37.9	30.9	38.4	45.2	**50.3**
2014	Deft-forum	27.2	38.7	41.5	46.1	49.1	**49.4**
	Deft-news	68.3	69.3	53.7	39.1	72.3	**74.1**
	Headlines	59.8	63.8	57.5	50.9	70.3	**71.7**
	Images	61.2	61.7	67.6	62.8	**78.3**	74.2
	OnWN	58.3	57.4	67.7	61.5	78.7	**85.5**
	Tweet-news	51.1	53.4	58.0	48.3	**76.7**	73.5
2015	Answers-forums	30.7	32.3	32.8	51.4	67.2	**70.2**
	Belief	40.7	42.3	51.9	52.7	**75.8**	73.4
	Headlines	61.9	63.2	65.3	56.8	75.0	**80.8**
	Images	67.6	68.5	71.4	64.3	81.2	**83.4**

Bold values denote the optimal results among the horizontal experimental results.

sentence is viewed as a document, and all sentences in the dataset are used to compute the value of IDF.

4.3. Comparison with Other Methods. Comparison results of the proposed method and other state-of-the-art methods for each task are provided in Table 3. It is easy to see that the proposed method obtains better or comparable performance. The methods that utilize word embedding/sentence embedding techniques include glove [28], word2vec [29], RNN [30], LSTM [31], and iRNN [32].

Glove denotes global vectors, which is is a word representation tool based on global word frequency statistics. The glove can map a word into a numerical vector that captures some semantic properties between words.

Word2vec denotes word to vector. Word2vec trains layers of neural networks to map one-hot word vectors to distributed word vectors.

RNN is the classical recurrent neural network that can solve the problem that the training sample is a continuous sequence with different lengths.

iRNN is an independently recurrent neural network that is improved based on RNN. iRNN's neurons in the same layer are independent of each other, and they are connected across layers. Multiple iRNNs can be stacked to construct a network that is deeper than the existing RNNs.

LSTM denotes long short-term memory, which is is a special kind of RNN. For remembering the long text's vital information during the training, the neural network cells will forget some information and remember some information.

Compared with these methods, our method accomplishes the best results on 13 out of 18 datasets. Even if the proposed method is not the best one on some datasets, it can also compete with the best results, such as 12′OnWn datasets and 15′belief datasets. It is just as that the result of the method on the 12′OnWn datasets is 68.8%, and the result of the best method is 70.2%.

5. Analysis

5.1. Analysis of Results. Table 3 shows the results of the accuracy of the state-of-the-art methods on the same datasets. The methods that use word embedding techniques include glove and word2vec. The performance of word2vec is better than that of glove. The reason is that word2vec is more suitable for processing sequential data than glove and the most typical sequential data is the sequence of text. Besides, word vectors trained by word2vec have stronger semantic correlation than glove. Because our method integrates word2vec into the dependency tree, meanwhile, Gaussian attention weight is added to highlight important features. The WVD-tree method, which contains semantic and syntactic information, is better than the word2vec method.

The methods that use sentence embedding techniques include RNN, LSTM, and iRNN. Both LSTM and iRNN are improved based on the RNN method. The traditional RNN cannot cope well with the long sequence input due to vanishing gradients or exploding gradients. The vanishing gradients problem of RNN is solved by the LSTM method, but it cannot counter the exploding gradients problem. Another advantage of the LSTM compared to RNN is that it can handle long text. However, it can be seen from Table 3 that the performance of LSTM is not better than that of RNN. The reason is that the STS datasets are full of short sentences, so the advantages of LSTM method cannot be given full play. The performance of iRNN is better than those of RNN and LSTM, because, compared with RNN and LSTM, the gradients of iRNN are relatively stable, and iRNN can be stacked in multiple layers, making it possible to build deeper networks. Our method is generally superior to these neural network methods, mainly because they focus on semantic information training rather than syntactic information, which is different from the WVD-tree method.

Meanwhile, it is necessary to analyze why our method does not perform well on some datasets such as 12′SMTeuroparl and 12′SMTnews. Because of the particularity of the sentences in these datasets, such sentences contain many strings of numbers and special symbols that damage the method's performance, for example, the special string "(a5-0241/2000)" in 12′SMTeuroparl dataset.

5.2. Effectiveness of Syntactic Information. For proving that syntactic information plays a role in the WVD-tree structure, two methods are compared in this section: (1) "Gaussian weight + word2vec," which uses the Gaussian attention weight and word2vec vector together, and (2) "Gaussian weight + word2vec + DT," which uses Gaussian attention weight, word2vec vector, and dependency tree together. Figure 10 displays the results obtained by using four datasets from the STS′15 task. It can find that "Gaussian weight + word2vec + DT" always performs better than the "Gaussian weight + word2vec." The experimental results of the method with "Gaussian weight + word2vec + DT" increased 7.8%, 7.2%, 7.3%, and 2.3%, respectively. The above results essentially prove the effectiveness of syntactic information. It should be noted that the STS′15 task is chosen for the experiment because the sentences of these datasets are clearly expressed and do not contain any special characters, and the improvement of experimental results by syntactic information could be found.

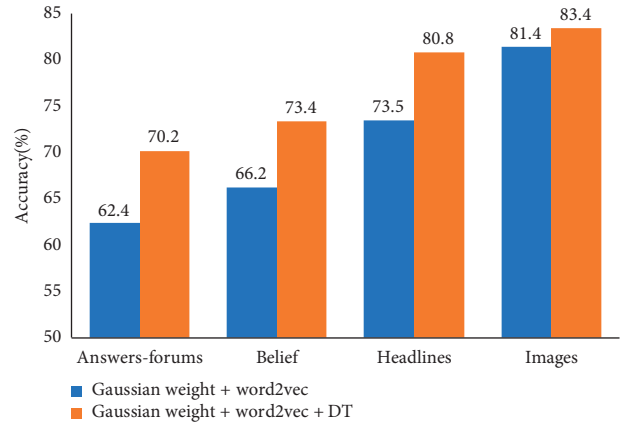

Figure 10: Effectiveness of syntactic information.

5.3. Effectiveness of Gaussian Attention Weight. For proving that the Gaussian attention weight plays a role in WVD-tree structure and works better than traditional attention mechanism, three methods are compared in this section: (1) "word2vec + DT," which uses the word2vec vector and dependency tree together, (2) "traditional weight + word2vec + DT," which uses traditional attention weight, word2vec vector, and dependency tree together, and (3) "Gaussian weight + word2vec + DT," which uses Gaussian attention weight, word2vec vector, and dependency tree together. Figure 11 shows the results obtained by using four datasets from the STS′15 task. It can be found that "Gaussian weight + word2vec + DT" always performs better than "word2vec + DT" and "traditional weight + word2vec + DT." The experimental results of "Gaussian weight + word2vec + DT" increased by 4.5%, 3.8%, 2.7%, and 2.3% compared to "word2vec + DT." The experimental results of "Gaussian weight + word2vec + DT" increased by 1.7%, 2.6%, 2.0%, and 0.8% compared to "traditional weight + word2vec + DT." This phenomenon proves that the Gaussian attention mechanism plays a positive role in performance.

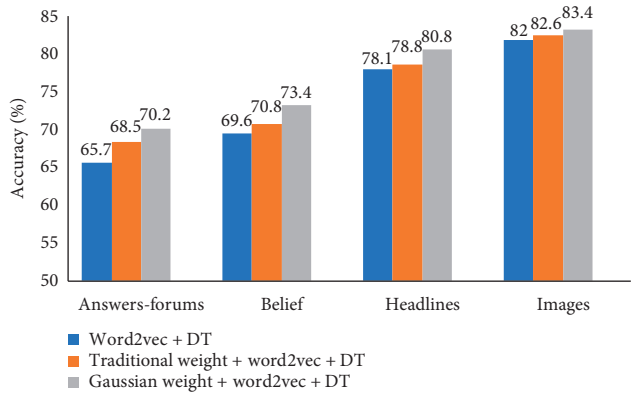

Figure 11: Effectiveness of Gaussian attention weight.

Table 4: The performance of different tree kernels.

Dataset	STK	SSTK	PTK
MSRpar	62.1	62.5	**64.3**
MSRvid	76.8	76.4	**80.6**
SMTeuroparl	47.0	47.5	**51.7**
OnWN	67.1	66.3	**68.8**
SMTnews	50.9	50.1	**53.2**

Bold values denote the optimal results among the horizontal experimental results.

5.4. Varying Tree Kernel. The enhanced tree kernel introduced before (recall Section 3.3) is changed based on the partial tree kernel. However, the commonly used tree kernel includes subtree kernel (STK) [20] and subset tree kernel (SSTK) [27] in addition to partial tree kernel.

5.4.1. Subtree Kernel. A subtree is characterized by containing all descendant nodes of the target node. The STK is a convolution kernel that computes common fragments shared between two trees. The fragment considered is a subtree, which means a node and its full children.

5.4.2. Subset Tree Kernel. Since leaf nodes of subset tree can be nonterminal symbols, the subset tree is a more general structure than a subtree. The SSTK is a convolution kernel that computes common fragments shared between two trees. The fragment considered is a subset tree, that is, a node and some of its children (children may be incomplete in depth, but partial generation is not allowed).

Here we also change subtree kernel and subset tree kernel in the same way (add Gaussian attention weights and word vectors to traditional tree kernel) and then compare their performance. The comparison results tested on the STS′12 task are shown in Table 4. Comparison results of the STK and the SSTK have little change in accuracy rate. The reason is that STK and SSTK do not differ much in the depth of splitting tree fragments. The PTK has better performance than other tree kernels. The reason is that the PTK can get a greater number of tree fragments than other tree kernels so that PTK can compare more common details for computing the similarity between two structure trees.

6. Conclusions

This paper proposed a new method for textual similarity measurement. The proposed model aims to extract syntactic-semantic information of sentences by combining several techniques, including word embedding, dependency tree, and attention weight mechanism. In this model, the WVD-tree is a semantic-embedded dependency tree structure for sentence representation. Meanwhile, the enhanced tree kernel is an algorithm designed for similarity calculation. Experiment results on widely used STS tasks demonstrate that this model can achieve favourable performance. Moreover, we design the Gaussian attention weight for distinguishing the contribution of different words more effectively. Experiment results also show that the Gaussian attention mechanism outperforms the traditional attention mechanism. In general, the major advantages of our method are the following: (1) it can be used as a general architecture because the integrated techniques in our model can be viewed as building blocks, allowing users to replace them with other more useful techniques; (2) different from most of the neural network methods, our model does not require time-consuming training, once word embeddings are available. In the future, there are still many works to study. For example, (1) other state-of-the-art techniques can be integrated to improve our model further; (2) our model can be compared with some neural network methods (e.g., RNN, LSTM, and iRNN) on larger training datasets to see whether our model can still achieve competitive performance.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the 13th Five-Year Plan Project of National Education Science under Grant no. DGA160245 and in part by the National Natural Science Foundation of China under Grant nos. U181140002 and 71971031.

References

[1] Z. W. Xie, Z. Zeng, G. Y. Zhou, and W. J. Wang, "Topic enhanced deep structured semantic models for knowledge base question answering," *Science China-Information Sciences*, vol. 60, no. 11, 2017.

[2] Y. Chali, S. A. Hasan, and S. R. Joty, "Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels," *Information Processing & Management*, vol. 47, no. 6, pp. 843–855, 2011.

[3] H. Palangi, L. Deng, Y. L. Shen et al., "Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval," *Ieee-Acm Transactions on Audio Speech and Language Processing*, vol. 24, no. 4, pp. 694–707, 2016.

[4] B. Galitsky, "Machine learning of syntactic parse trees for search and classification of text," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 3, pp. 1072–1091, 2013.

[5] J. Li, J. Li, X. Fu, M. A. Masud, and J. Z. Huang, "Learning distributed word representation with multi-contextual mixed embedding," *Knowledge-Based Systems*, vol. 106, pp. 220–230, 2016.

[6] M. Esposito, E. Damiano, A. Minutolo, G. De Pietro, and H. Fujita, "Hybrid query expansion using lexical resources and word embeddings for sentence retrieval in question answering," *Information Sciences*, vol. 514, pp. 88–105, 2020.

[7] H. T. Nguyen, P. H. Duong, and E. Cambria, "Learning short-text semantic similarity with word embeddings and external knowledge sources," *Knowledge-Based Systems*, vol. 182, 2019.

[8] N. H. Tien, N. M. Le, Y. Tomohiro, and I. Tatsuya, "Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity," *Information Processing & Management*, vol. 56, no. 6, p. 11, 2019.

[9] W. H. Xing, X. H. Yuan, L. Li, L. Hu, and J. Peng, "Phenotype extraction extraction based on word embedding to sentence embedding cascaded approach," *Ieee Transactions on Nanobioscience*, vol. 17, no. 3, pp. 172–180, 2018.

[10] H. P. Yao, H. W. Liu, and P. Y. Zhang, "A novel sentence similarity model with word embedding based on convolutional neural network," *Concurrency and Computation-Practice & Experience*, vol. 30, no. 23, p. 12, 2018.

[11] I. Lopez-Gazpio, M. Maritxalar, M. Lapata, and E. Agirre, "Word n-gram attention models for sentence similarity and inference," *Expert Systems with Applications*, vol. 132, pp. 1–11, 2019.

[12] C. Z. Xiong and M. L. Su, "IARNN-based semantic-containing double-level embedding Bi-LSTM for question-and-answer matching," *Computational Intelligence and Neuroscience*, vol. 10, 2019.

[13] H. Choi, K. Cho, and Y. Bengio, "Fine-grained attention mechanism for neural machine translation," *Neurocomputing*, vol. 284, pp. 171–176, 2018.

[14] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *Ieee Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

[15] Q. Chen, Q. Hu, J. X. Huang, L. He, and Aaai, "CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence/Thirtieth Innovative Applications of Artificial Intelligence Conference/Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 265–273, New Orleans, LO, USA, 2018.

[16] Z. Quan, Z. J. Wang, Y. Q. Le, B. Yao, K. L. Li, and J. Yin, "An efficient framework for sentence similarity modeling," *Ieee-Acm Transactions on Audio Speech and Language Processing*, vol. 27, no. 4, pp. 853–865, 2019.

[17] H. Liu, W. Q. Song, M. Li, A. Kudreyko, and E. Zio, "Fractional Levy stable motion: finite difference iterative forecasting model," *Chaos Solitons & Fractals*, vol. 133, p. 11, 2020.

[18] X. Li and Q. S. Li, "Calculation of sentence semantic similarity based on syntactic structure," *Mathematical Problems in Engineering*, vol. 2015, Article ID 203475, 8 pages, 2015.

[19] S. X. Gao, Z. T. Yu, C. Liu, L. Chen, and X. D. Hong, "A Chinese-naxi tree-to-tree machine translation method based on subtree alignment," *Mathematical Problems in Engineering*, vol. 2015, Article ID 414963, 5 pages, 2015.

[20] F. Aiolli, G. Da San Martino, and A. Sperduti, "An efficient topological distance-based tree kernel," *Ieee Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1115–1120, 2015.

[21] K. Rieck, T. Krueger, U. Brefeld, and K. R. Muller, "Approximate tree kernels," *Journal of Machine Learning Research*, vol. 11, pp. 555–580, 2010.

[22] S. Lai, K. Liu, S. He, and J. Zhao, "How to generate a good word embedding," *Ieee Intelligent Systems*, vol. 31, no. 6, pp. 5–14, 2016.

[23] F. H. Lin, Y. F. Liu, H. J. Lee et al., "Differential brain mechanisms during reading human vs. machine translated fiction and news texts," *Scientific Reports*, vol. 9, 2019.

[24] M. Mohammed and N. Omar, "Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec," *Plos One*, vol. 15, no. 3, p. 21, 2020.

[25] W. Chen, M. Zhang, and Y. Zhang, "Distributed feature representations for dependency parsing," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 451–460, 2015.

[26] S.-Y. Yang and V.-W. Soo, "Extract conceptual graphs from plain texts in patent claims," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 4, pp. 874–887, Jun, 2012.

[27] A. Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *Lecture Notes in Computer Science*, T. Scheffer and M. Spiliopoulou, Eds., pp. 318–329, Springer-Verlag, Berlin, Germany, 2006.

[28] F. Sakketou and N. Ampazis, "A constrained optimization algorithm for learning GloVe embeddings with semantic lexicons," *Knowledge-Based Systems*, vol. 195, 2020.

[29] S. Yilmaz and S. Toklu, "A deep learning analysis on question classification task using Word2vec representations," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2909–2928, 2020.

[30] B. Agarwal, H. Ramampiaro, H. Langseth, and M. Ruocco, "A deep network model for paraphrase detection in short text messages," *Information Processing & Management*, vol. 54, no. 6, pp. 922–937, 2018.

[31] J. Lee, S. Seo, and Y. S. Choi, "Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing," *Symmetry-Basel*, vol. 11, no. 6, p. 12, 2019.

[32] X. X. Zhang, Y. H. Liu, X. L. Wu, and Z. W. Niu, "Intelligent pulse analysis of high-speed electrical discharge machining using different RNNs," *Journal of Intelligent Manufacturing*, vol. 31, no. 4, pp. 937–951, 2020.