mathematics

MDPI

Article

Sentence Embedding Generation Framework Based on Kullback–Leibler Divergence Optimization and RoBERTa Knowledge Distillation

Jin Han [1,*] and Liang Yang [2]

[1] School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2] School of Computer Science, Nanjing University of Information Science & Technology, Nanjing 210044, China; yl20141344045@163.com
* Correspondence: hjhaohj@126.com

Abstract: In natural language processing (NLP) tasks, computing semantic textual similarity (STS) is crucial for capturing nuanced semantic differences in text. Traditional word vector methods, such as Word2Vec and GloVe, as well as deep learning models like BERT, face limitations in handling context dependency and polysemy and present challenges in computational resources and real-time processing. To address these issues, this paper introduces two novel methods. First, a sentence embedding generation method based on Kullback–Leibler Divergence (KLD) optimization is proposed, which enhances semantic differentiation between sentence vectors, thereby improving the accuracy of textual similarity computation. Second, this study proposes a framework incorporating RoBERTa knowledge distillation, which integrates the deep semantic insights of the RoBERTa model with prior methodologies to enhance sentence embeddings while preserving computational efficiency. Additionally, the study extends its contributions to sentiment analysis tasks by leveraging the enhanced embeddings for classification. The sentiment analysis experiments, conducted using a Stochastic Gradient Descent (SGD) classifier on the ACL IMDB dataset, demonstrate the effectiveness of the proposed methods, achieving high precision, recall, and F1 score metrics. To further augment model accuracy and efficacy, a feature selection approach is introduced, specifically through the Dynamic Principal Component Selection (DPCS) algorithm. The DPCS method autonomously identifies and prioritizes critical features, thus enriching the expressive capacity of sentence vectors and significantly advancing the accuracy of similarity computations. Experimental results demonstrate that our method outperforms existing methods in semantic similarity computation on the SemEval-2016 dataset. When evaluated using cosine similarity of average vectors, our model achieved a Pearson correlation coefficient (τ) of 0.470, a Spearman correlation coefficient (ρ) of 0.481, and a mean absolute error (MAE) of 2.100. Compared to traditional methods such as Word2Vec, GloVe, and FastText, our method significantly enhances similarity computation accuracy. Using TF-IDF-weighted cosine similarity evaluation, our model achieved a τ of 0.528, ρ of 0.518, and an MAE of 1.343. Additionally, in the cosine similarity assessment leveraging the Dynamic Principal Component Smoothing (DPCS) algorithm, our model achieved a τ of 0.530, ρ of 0.518, and an MAE of 1.320, further demonstrating the method's effectiveness and precision in handling semantic similarity. These results indicate that our proposed method has high relevance and low error in semantic textual similarity tasks, thereby better capturing subtle semantic differences between texts.

Keywords: semantic textual similarity (STS); Kullback–Leibler divergence (KLD); knowledge distillation; feature selection; similarity evaluation; sentence embedding

MSC: 68T50

Citation: Han, J.; Yang, L. Sentence Embedding Generation Framework Based on Kullback–Leibler Divergence Optimization and RoBERTa Knowledge Distillation. *Mathematics* **2024**, *12*, 3990. https://doi.org/10.3390/math12243990

Academic Editor: Simone Faro

Received: 18 November 2024
Revised: 17 December 2024
Accepted: 18 December 2024
Published: 18 December 2024

1. Introduction

Semantic textual similarity (STS) [1] is a fundamental task within natural language processing (NLP) [2] that involves evaluating the semantic similarity between text segments, typically sentences. The task requires assigning a similarity score, ranging from 0 to 5, with higher scores indicating greater semantic similarity between the sentence pairs. STS has wide-ranging applications across various domains, such as information retrieval, machine translation, text summarization, dialog systems, and educational assessments. For example, in information retrieval, STS enables search engines to assess the semantic similarity between user queries and document content, thereby enhancing the relevance of search results. In education, STS can be employed to automate the evaluation of student responses, ensuring consistency and fairness by comparing student essays with predefined standard answers.

Historically, early methods for computing STS relied on techniques like bag-of-words models, syntactic analysis, and word vector-based similarity. However, these methods often fell short in capturing nuanced semantic relationships and contextual dependencies. The introduction of deep learning-based approaches and pre-trained language models, such as BERT and RoBERTa, has significantly advanced the field by better capturing complex semantic relationships, incorporating contextual information, and improving model generalization through transfer learning. These advancements have led to a notable increase in the precision and robustness of STS evaluations.

Nevertheless, deep learning models, particularly those using pre-trained language models, are resource-intensive, often requiring extensive computational power and time for training and inference. These models, with their hundreds of millions to billions of parameters, demand high-performance hardware such as GPUs or TPUs, presenting a significant challenge for real-time processing and large-scale deployment.

In parallel, traditional methods for sentence representation, such as simple and weighted averaging of word vectors, often fail to capture the subtle semantic differences between sentences, particularly in cases involving polysemy or context-dependent language. To address these limitations, we propose a novel method that optimizes sentence vectors using Kullback–Leibler Divergence (KLD). This approach enhances the semantic differentiation between sentences, allowing for more accurate representations of semantic changes and contextual nuances within and between sentences.

Moreover, feature selection plays a crucial role in improving the performance of natural language processing tasks by identifying the most informative features from a vast set of potential inputs. In our framework, we introduce Dynamic Principal Component Smoothing (DPCS) as a feature selection technique that dynamically adapts the composition of sentence representations. By selecting and emphasizing the most discriminative components, DPCS optimizes sentence vectors, enabling better alignment with different semantic levels and enhancing performance across various NLP tasks. This integration of KLD optimization and DPCS feature selection provides a powerful solution for capturing subtle semantic differences and improving model generalization, particularly for complex tasks such as text classification and sentiment analysis.

In summary, our proposed method addresses the following challenges in STS and broader NLP tasks: (1) the limitation of static word vector-based sentence representations in capturing nuanced semantic differences, (2) the computational inefficiency of deep learning models for real-time applications, and (3) the need for adaptive, context-sensitive feature selection to improve model performance and generalization.

The primary contributions of this paper are as follows:

1. We propose a novel KLD-enhanced word vector method that uses KLD as a metric tool and iterative optimization to generate vectors with better semantic differentiation. This approach improves semantic feature representation in practical contexts. However, it has limitations in handling context dependency and polysemy, which are better addressed by deep learning methods. This motivated the development of a second method to overcome these challenges.

2. We present a RoBERTa-based knowledge distillation [3] framework enhanced with Dynamic Principal Component Smoothing (DPCS). This framework combines RoBERTa's deep semantic insights with traditional embedding techniques, transferring contextual knowledge to improve semantic fidelity and computational efficiency. DPCS further refines sentence representations by adapting principal components to contextual nuances, enhancing adaptability and precision across multiple semantic layers. This method excels in textual similarity tasks, demonstrating superior robustness and generalization in complex NLP applications.

We compare our proposed method with Word2Vec, FastText [4], GloVe, BERT, SBERT [5], and SimCSE [6] using cosine similarity with average vectors, TF-IDF-weighted [7] cosine similarity [8], and DPCS-weighted cosine similarity. Experimental results on the SemEval-2016 dataset show that our approach achieves a Pearson correlation coefficient (τ) [9] and Spearman correlation coefficient (ρ) [10] of 0.470 and 0.481, respectively, with a mean absolute error (MAE) of 2.100 for average vectors. For TF-IDF-weighted cosine similarity, τ is 0.528, ρ is 0.518, and MAE is 1.343. With DPCS, τ is 0.518, ρ is 0.517, and MAE is 1.287, demonstrating the high relevance and low-error performance of our method.

Additionally, sentiment analysis experiments were conducted on the ACL IMDB dataset [11] to further evaluate the performance of our proposed method. The results show that our approach outperforms existing models, achieving a precision of 0.75, recall of 0.88, and an F1 score of 0.81. Compared to other methods such as Word2Vec (precision: 0.66; recall: 0.02; F1: 0.04), GloVe (precision: 0.73; recall: 0.77; F1: 0.75), and BERT (precision: 0.71; recall: 0.82; F1: 0.76), our model demonstrated significantly higher precision and recall, indicating its effectiveness in capturing sentiment-related information. These results further validate the robustness and superior performance of our method in sentiment classification tasks.

2. Related Work

This section reviews related work on sentence embeddings and semantic textual similarity (STS). It discusses the limitations of traditional embedding methods like Word2Vec, GloVe, and FastText in handling contextual dependencies and polysemy, as well as recent advances in pre-trained language models such as BERT and RoBERTa, highlighting challenges in computational efficiency and interpretability. Additionally, it examines techniques like Smoothing Inverse Frequency (SIF) and Principal Component Analysis (PCA), which enhance embedding quality by reducing noise and focusing on task-relevant features. These studies provide the foundation for the proposed KLD-enhanced sentence embeddings and the RoBERTa knowledge distillation framework with DPCS.

2.1. Word Vector Models

Word vector models are pivotal in natural language processing (NLP) and have undergone significant evolution through various stages and methodologies. Initial efforts in the late 1980s and early 1990s concentrated on statistical methods such as word frequency statistics and co-occurrence matrices. Although rudimentary, these approaches laid the groundwork for the development of subsequent word vector models.

With advancements in computational power and the advent of deep learning technologies, word vector models transitioned into a new developmental phase. In 2013, Tomas Mikolov et al. [12] introduced the Word2Vec model, which included two primary training methods: Continuous Bag of Words (CBOW) [13] and Skip-gram [14]. These techniques leverage neural network models to learn distributed representations, or word vectors, from large-scale text corpora. The introduction of Word2Vec markedly enhanced the quality and applicability of word vectors, thereby improving the ability of computers to understand and process semantic information in natural language.

In 2014, Pennington et al. [15] developed the GloVe (Global Vectors for Word Representation) model. GloVe combines global word frequency statistics with local context window information, learning word vectors by minimizing a loss function. This approach

improved the performance of word vector models in handling polysemous and rare words. The GloVe model further diversified the design methodologies of word vector models and found extensive applications across various NLP domains.

In 2016, Mikolov et al. [16] introduced the FastText model, which enhances the representation capabilities of word vectors by incorporating subword information, making it particularly suitable for languages with rich morphology and rare words. In 2018, Peters et al. introduced the ELMo (Embeddings from Language Models) [17] model, which learns context-sensitive word vectors through bidirectional language models, thereby further enhancing the expressive power and semantic understanding of word vectors. In the same year, Devlin et al. [18] proposed the BERT (Bidirectional Encoder Representations from Transformers) model, which introduced the Transformer architecture and the Masked Language Modeling task. BERT represented a significant milestone in NLP tasks and has been widely applied to various text processing tasks. In 2019, Facebook AI released RoBERTa (Robustly optimized BERT approach) [19], an enhanced version of BERT. RoBERTa optimized the pre-training phase by utilizing a larger training dataset, removing the next sentence prediction task, and increasing the batch size and learning rate, among other enhancements. These improvements enabled RoBERTa to achieve superior performance across various NLP tasks.

Building upon the BERT framework, Reimers et al. introduced SBERT (Sentence-BERT) in 2019 as a method to generate more effective sentence embeddings, particularly in tasks like semantic textual similarity. SBERT uses Siamese networks for training sentence pairs, enabling the model to generate fixed-length sentence embeddings that capture complex sentence-level semantics more efficiently. SBERT has demonstrated significant improvements over BERT-based methods in tasks such as clustering and retrieval, and its computational efficiency makes it more suitable for real-time applications. Another important approach in this line is SimCSE, introduced by Gao and Chen in 2021, leveraging contrastive learning to optimize sentence embeddings. By leveraging self-supervised learning through positive and negative samples, SimCSE enhances the quality of sentence embeddings, especially for similarity-based tasks. SimCSE's simplicity and effectiveness have led to its widespread use in various NLP benchmarks.

In recent years, large model-based approaches have further advanced the capabilities of NLP. Notable models such as OpenAI's GPT-3 (Generative Pre-trained Transformer 3), released in 2020, have demonstrated remarkable performance across a variety of tasks, including text generation, translation, and summarization. With 175 billion parameters, GPT-3 showcases the potential of massive models to handle highly complex linguistic patterns and generate contextually appropriate responses. However, the immense scale of these models also brings challenges related to computational resources and deployment efficiency. These models, built on the Transformer architecture, benefit from vast pre-training corpora and advanced fine-tuning techniques, yet their large size makes them resource-intensive, often requiring specialized hardware for training and inference.

Other significant advancements include Google's T5 (Text-to-Text Transfer Transformer), introduced in 2020, which frames all NLP tasks as a text-to-text problem, enabling its application across a wide variety of tasks. T5 has set new benchmarks in multiple NLP domains, including text classification and summarization. The use of such large pre-trained models reflects a broader trend in NLP research towards models that integrate vast amounts of linguistic knowledge and are adaptable to diverse tasks.

The growth of large models has also extended into the multimodal domain, as demonstrated by OpenAI's DALL·E, released in 2021, which generates high-quality images from textual descriptions. This approach exemplifies the potential of combining NLP with computer vision, further expanding the boundaries of large models. Despite the successes, these models face significant challenges, particularly in terms of energy consumption, hardware requirements, and scalability in real-time applications.

While pre-trained language models excel at capturing complex semantic relationships and contextual information, they require high-performance hardware for practical

implementation. As model complexity increases, real-time processing and large-scale deployment become increasingly challenging.

2.2. Methods for Semantic Text Similarity Calculation

Semantic text similarity calculation is a fundamental task in natural language processing, involving the measurement of semantic or structural closeness between text fragments or words. With the advancement of word vector models, methods for similarity calculation have evolved and improved.

Euclidean Distance [20], introduced by Euclid in the 3rd century BC, measures the straight-line distance between two points and is often employed in vector space calculations for tasks such as image processing and clustering analysis. Cosine similarity, an early method based on the vector space model, measures the cosine of the angle between two vectors to assess their similarity. This method is simple and efficient, making it suitable for basic semantic matching and text retrieval tasks. Jaccard Similarity [21], proposed by Paul Jaccard in 1901, compares the similarity and difference between finite sample sets by calculating the ratio of their intersection to their union, and is useful for document similarity and set data analysis. Edit Distance [22], introduced by Vladimir Levenshtein in 1965, calculates the minimum number of edit operations (insertions, deletions, or substitutions) needed to transform one string into another. Term frequency–inverse document frequency (TF-IDF), introduced by Salton and Buckley in 1972, is widely used in information retrieval and text mining to assess the importance of words in documents. TF-IDF-weighted cosine similarity integrates TF-IDF weighting with cosine similarity for text similarity calculation.

Tree Kernel-based similarity [23], proposed by Collins and Duffy in 2003, applies support vector machines to calculate the similarity between the syntactic trees of sentences, which is useful for tasks such as semantic role labeling. Word Mover's Distance (WMD), introduced by Matt Kusner et al. [24] in 2016, measures the semantic similarity between documents based on word embeddings, considering the distance and flow path between words to capture semantic relationships more accurately.

Smooth inverse frequency (SIF) cosine similarity, introduced by Arora et al. [25] in 2017, combines global information of word vectors with local context information. By reducing the weight of common words and increasing the weight of rare words, SIF enhances the quality of word vector representations. It first calculates each word's weight and then generates the document or sentence vector representation through weighted averaging. This method excels in reducing the interference of common words while enhancing the contributions of rare words, making it suitable for document similarity calculation and information retrieval tasks.

However, the SIF method has certain limitations, particularly in handling semantically complex texts. It often fails to capture subtle semantic differences within sentences, especially in cases involving polysemy or strong contextual dependencies. In contrast, the Dynamic Principal Component Smoothing (DPCS) method proposed in this study overcomes these limitations by dynamically selecting and weighting the most relevant principal components in the context to generate sentence vectors. This approach allows for greater flexibility in adapting to varying contexts and semantic levels, addressing the shortcomings of SIF in fine-grained semantic capture. DPCS not only retains computational efficiency but also avoids the drawback of excessive reliance on rare words inherent in SIF. Furthermore, it automatically adjusts the feature selection strategy according to different contexts, thereby enhancing the robustness and adaptability of the model. Consequently, when dealing with semantically complex and context-dependent texts, DPCS demonstrates superior advantages over SIF, particularly in tasks such as textual similarity computation and other complex natural language processing applications.

3. Introduction of the Algorithm

The objective of this paper is to develop a novel word vector space for text similarity computation. The process of generating the vector space and calculating text similarity is depicted in Figure 1.

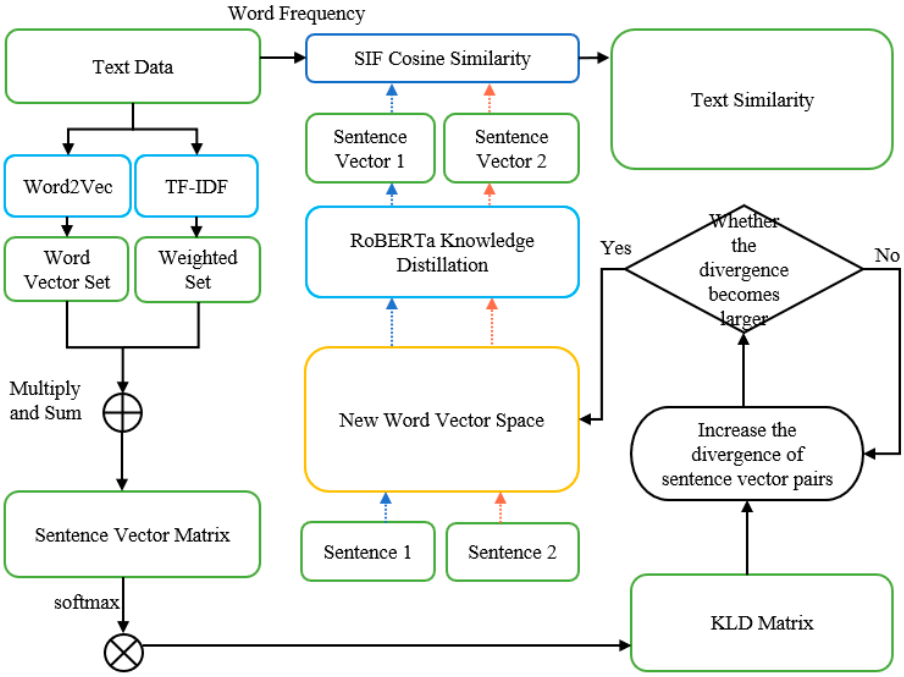


Figure 1. Construction of word vector space and text similarity calculation.

3.1. Sentence Embedding Generation Method Based on KLD Optimization

Word2Vec employs two principal training methods: Continuous Bag of Words (CBOW) and Skip-gram. These methods utilize neural networks to learn distributed representations, or word vectors, from large-scale text corpora. This study employs the Skip-gram model from Word2Vec as the primary embedding technique, selected for its demonstrated capability in capturing both syntactic and semantic relationships across large-scale text corpora. The Skip-gram approach is particularly advantageous due to its ability to generate high-quality vector representations for rare and low-frequency words, a critical factor for optimizing embeddings through KLD [26]. Conversely, CBOW predicts the center word for a given context, performing well for frequent words but lacking representation for rare words. Alternative embedding methods, such as GloVe and FastText, were not utilized within the proposed workflow due to specific limitations. GloVe's reliance on global co-occurrence statistics lacks the dynamic adaptability required for KLD-driven optimization, while FastText's incorporation of subword information introduces complexities that are extraneous to the objectives of the current framework.

Skip-gram predicts context words for a given word, which is effective for rare words and long-tail distributed data but less effective for frequently occurring words. By introducing KLD, the semantic distance between different sentence or word vectors can be expanded, enhancing the differentiation capability of word vectors by making similar words closer and dissimilar words farther apart. We propose a novel method for generating sentence vectors, termed KLD-enhanced word vectors.

KLD is an asymmetric measure used to quantify the difference between two probability distributions, P and Q. As shown in Equation (1), it describes the information loss of distribution Q relative to distribution P.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \quad (1)$$

First, a pre-trained Word2Vec model generates word vectors, followed by sentence vector generation using the average method and weighted average method. The softmax function then converts sentence vectors into probability distributions. By calculating the KLD between different sentence vectors, the semantic difference in the space can be assessed. KLD measures the difference between two probability distributions and, in this context, assesses semantic differences between sentence vectors.

Based on the KLD results, the training process of the Word2Vec model can be adjusted or optimized to make sentence vector differences more distinct or reasonable. Iterating this process generates new word or sentence vectors that better capture semantic differences and subtle changes, thereby improving performance and accuracy in tasks such as text similarity comparison, information retrieval, and semantic understanding. The advantage of this method lies in using KLD as a measurement tool combined with iterative optimization, resulting in word or sentence vectors that align more closely with semantic differences in practical applications, accurately expressing the semantic features and information content of texts. The KLD optimization process is illustrated in Algorithm 1.

Algorithm 1. KL divergence matrix computation.

Require: stop words, word vectors, sentences, TF-IDF weights
Ensure: Updated word vectors saved to a file
Objective: Minimize Loss

Initialize optimizer with word to vec.values()
Set learning_rate = 1×10^{-4}
Set N = 50
Set batch_size = 100
optimizer = initialize SGD_optimizer with parameters(word_to_vec.values(), learning_rate)
for iteration = 1 to N **do**
total loss ← 0
for each sentence pair (i,j) in batches of size batch_size **do**
Zero gradients
vi ← compute sentence vector(sentences[i],tfidf weights,word to vec)
vj ← compute sentence vector(sentences[j],tfidf weights,word to vec)
if vi = 0 or vj = 0 **then**
continue
end if
pi ← softmax vector(vi)
pj ← softmax vector(vj)
kl ← kl divergence(pi, pj)
loss ← −kl
Backpropagate loss
Update optimizer
end for
end for
Save updated word vectors to file

The key parameters required to fine-tune the optimization process in the proposed KLD optimization method include the learning rate, the number of training epochs, and the batch size. In our experiments, the learning rate is set to 0.0001, a value that strikes a balance between fast convergence and stability, ensuring that the optimization process neither overshoots the optimal solution nor converges too slowly. The number of training epochs is set to 50, providing enough iterations for the model to converge effectively without overfitting. The batch size is set to 100, which offers a good balance between the efficiency of training and the stability of gradient estimates. Smaller batch sizes may increase variance

but improve convergence speed, while larger batch sizes can smooth out the gradient but slow down the optimization. These parameters are critical to achieving an effective KLD optimization that refines sentence embeddings to capture semantic similarities accurately.

The objective of Kullback–Leibler Divergence (KLD) optimization is to enhance the representation of sentence embeddings by maximizing the semantic differentiation between sentences. The primary goal is to refine the sentence embeddings to more accurately capture subtle semantic variations, especially in cases where sentences are contextually similar but differ in meaning. By optimizing the KLD, we ensure that the resulting sentence embeddings are more sensitive to fine-grained semantic distinctions, thereby improving the performance in tasks like semantic textual similarity (STS) and other NLP applications. The KLD-based optimization method addresses the limitations of static word vectors, enabling more dynamic and context-aware representations.

KLD is employed as the loss function in this study due to its unique ability to capture the asymmetrical dissimilarities between probability distributions. While inverse cosine similarity is a widely used metric for measuring vector similarity, it assumes symmetry, making it less suitable for tasks where directional relationships are critical. KLD, in contrast, inherently accounts for the directionality of information flow, which is crucial in semantic textual similarity tasks where one sentence may contribute more information than the other. Additionally, KLD operates on probability distributions, aligning well with the probabilistic outputs of language models and enabling finer granularity in capturing semantic nuances. While inverse cosine similarity could theoretically be used, it may not capture the subtle asymmetrical relationships inherent in the task, thereby limiting its effectiveness. The choice of KLD as the loss function reflects its robustness in handling the probabilistic and asymmetrical nature of semantic tasks, ensuring a more accurate and context-sensitive optimization. This approach is particularly beneficial in applications where capturing subtle directional dissimilarities significantly impacts performance.

In this paper, we first selected one pair of related sentences and one pair of unrelated sentences, then calculated and plotted their similarity heatmaps before and after optimization. The pair of sentences with high similarity is "A man puts three pieces of meat into a pan" and "A man is putting meat in a pan" The pair with low similarity is "Yes, there is a rule against this" and "There's no rule against it". The heatmaps display the cosine similarity between words in each sentence, with darker colors indicating higher similarity and lighter colors indicating lower similarity. Table 1 illustrates the impact of KLD optimization on sentence similarity scores.

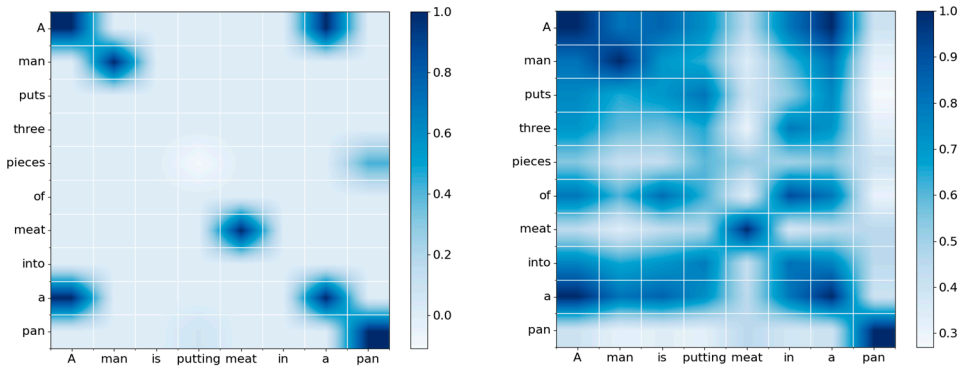
Table 1. Similarity scores of sentence pairs before and after optimization.

Sentence Pair	Before Optimization	After Optimization
A man puts three pieces of meat into a pan. A man is putting meat in a pan.	0.13	0.72
Yes, there is a rule against this. There's no rule against it.	0.74	0.20

Before optimization, the sentence pair "A man puts three pieces of meat into a pan" and "A man is putting meat in a pan" shows a low similarity score of 0.13, suggesting that the model initially struggles to capture the semantic equivalence between these sentences. In contrast, the pair "Yes, there is a rule against this" and "There's no rule against it" demonstrates a higher similarity score of 0.74, reflecting a closer alignment in meaning between the two sentences. After applying KLD optimization, the scores adjust significantly. The first pair's similarity score increases to 0.72, indicating that KLD optimization successfully enhanced the model's ability to recognize the semantic similarity between the sentences. However, the second pair's score drops to 0.20, reflecting the optimization's ability to more accurately capture the subtle opposition in meaning between the phrases, which initially shared a high similarity score. These results underscore the efficacy of KLD optimization in

refining sentence embeddings, improving both the detection of semantic equivalence and the differentiation of nuanced differences.

In Figure 2a, although the sentences as a whole show some similarity, certain word pairs exhibit lower similarity, revealing semantic differences in the details of the sentences.



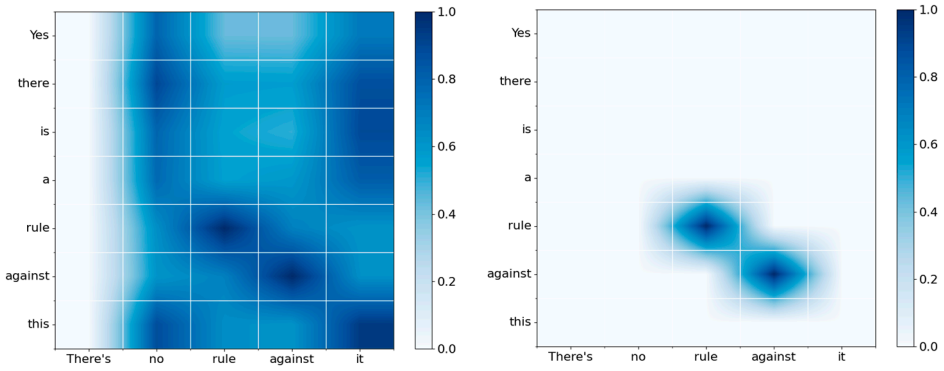
(**a**) (**b**)

Figure 2. The similarity between two relevant sentences. (**a**) Before KLD optimization; (**b**) after KLD optimization.

After KLD optimization, Figure 2b demonstrates more consistent regions of high similarity, indicating that the optimization successfully reduced semantic differences between the sentences, leading to more concentrated representations of the related sentence pair in the semantic space. As a result, the similarity of the related sentence pair was significantly improved through KLD optimization. By optimizing the word vector space, the related sentence pair showed greater semantic consistency, and the cosine similarity between the sentences increased significantly after optimization. This demonstrates that the optimization process effectively captured the underlying semantic relationships between the sentences and enhanced their semantic aggregation in the high-dimensional vector space, thereby improving the model's ability to capture subtle semantic differences.

In Figure 3a, the word pair similarities of the unrelated sentence pair are relatively high, which may reflect that the word vector space failed to effectively differentiate the semantics of the unrelated sentences. After KLD optimization, Figure 3b gradually shows a more dispersed similarity distribution, particularly with significantly reduced similarity for most word pairs. This indicates that the optimization effectively increased the semantic distance between the unrelated sentence pair. Therefore, for unrelated sentence pairs, KLD optimization effectively widened their distance in the vector space. After optimization, the cosine similarity of the unrelated sentence pair significantly decreased, indicating that the optimization process successfully distinguished semantically unrelated or opposing sentence pairs. In this way, the model further improved its ability to distinguish between related and unrelated sentence pairs, thereby enhancing its precision in determining textual semantic similarity.

The optimization of KLD is essential for improving the alignment of sentence embeddings in the vector space. KLD optimization aims to ensure that semantically similar sentence pairs are closer together in the embedding space, while dissimilar pairs are pushed farther apart. This enhances the ability of the model to effectively capture nuanced semantic relationships between sentences. By optimizing the KLD, we focus on adjusting the word vector distributions such that they better represent the actual semantic similarity between sentences. This optimization is particularly important because it improves the quality of the sentence embeddings, which is essential for tasks like semantic textual similarity (STS), where the goal is to evaluate how similar two pieces of text are.



(**a**) (**b**)

Figure 3. The similarity between two irrelevant sentences. (**a**) Before KLD optimization; (**b**) after KLD optimization.

However, this method has limitations in handling contextual dependencies and polysemy, potentially resulting in less effective semantic representation at the sentence level compared to deep learning-based methods. To address these limitations, we propose a second method.

3.2. Sentence Embedding Generation Framework Based on RoBERTa Knowledge Distillation

As illustrated in Figure 4, knowledge distillation is a model compression technique that enhances the performance of a smaller model (student model [27]) by learning from a larger model (teacher model [28]). Specifically, the teacher model produces soft labels (i.e., probability distributions) to guide the training of the student model.

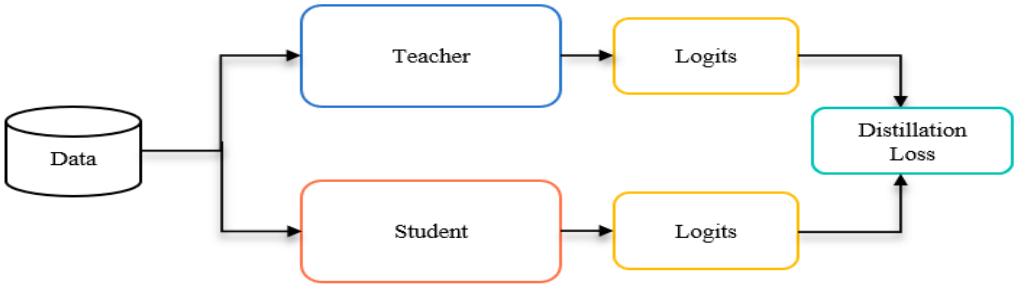


Figure 4. Knowledge distillation model.

This paper introduces a sentence embedding generation framework based on RoBERTa knowledge distillation, wherein RoBERTa generates soft labels to train the embedding model. This framework aims to leverage the robust representational power of the pre-trained language model RoBERTa. It not only inherits RoBERTa's deep semantic representation capabilities but also significantly improves the text similarity measurement accuracy of the smaller sentence embedding model through knowledge distillation. Algorithm 2 delineates the training process:

RoBERTa knowledge distillation is adopted in this study due to its ability to address limitations inherent in other pre-trained language models during distillation. For instance, while models like DistilBERT are effective in reducing the size and computational cost of BERT, they inherit BERT's architectural constraints, such as sensitivity to masking strategies and reliance on the next sentence prediction (NSP) task, which has been shown to contribute minimally to downstream performance. In contrast, RoBERTa eliminates NSP, employs dynamic masking, and utilizes larger, more diverse training datasets, resulting in enhanced contextual understanding and semantic representation. These advantages make RoBERTa a more robust foundation for knowledge distillation. Additionally, DistilRoBERTa, while also a distilled version of RoBERTa, focuses on general-purpose tasks and lacks task-

specific optimizations. Our approach customizes the distillation process for the specific task of semantic textual similarity (STS), ensuring an optimal balance between efficiency and accuracy. Furthermore, the necessity of this approach lies in achieving lightweight yet high-performing models for resource-constrained applications, which generalized models like DistilBERT and DistilRoBERTa may not sufficiently address. By leveraging RoBERTa's strengths, the proposed framework overcomes these limitations, while its model-agnostic design enables broader applicability beyond RoBERTa to other advanced pre-trained language models.

Algorithm 2. Model to train and save embeddings.

Input: Sentences S, soft labels L, save path p
Output: Trained Embeddings model saved to p

```
Function train and save Embeddings model(S, L, p)
embedding dim ← Embeddings model.vector size
num classes ← L.shape[−1]
model ← EmbeddingsModel(embedding dim, num classes)
optimizer ← Adam(model.parameters(), lr = 0.001)
criterion ← MSELoss()
dataset ← TensorDataset(S, L)
dataloader ← DataLoader(dataset, batch size = 32, shuffle = True)
for e ← 0 to N do
    model.train()
    total loss ← 0
    for (Xb, Yb) in dataloader do
        Zero gradients
        predictions ← model(Xb)
        losscriterion(predictions, Yb)
        Backpropagate loss
        Update optimizer
    end for
end for
save(model.state dict(), p)
```

The main steps of this framework include the following:

1. Obtaining Soft Labels for Sentences from the Pre-trained Language Model: Use the pre-trained language model RoBERTa to generate hidden layer representations of sentences and calculate the soft labels. Given an input sentence S, tokenize it into word IDs, and then process it through the pre-trained model to obtain hidden layer outputs H. In Equation (2), h_i represents the hidden layer representation of the i-th word:

$$H = \{h_1, h_2, \cdots, h_n\}. \tag{2}$$

To obtain the sentence's embedding representation, compute the average of hidden layer outputs. In Equation (3), SoftLabel(S) is the soft label of the sentence, representing the vector representation of the sentence:

$$\text{SoftLabel}(S) = \frac{1}{n}\sum_{i=1}^{n} h_i. \tag{3}$$

2. Preprocessing Sentences to Match the Input Requirements of the Above Model: Preprocess input sentences by removing stop words and tokenizing. Then, average or weighted-average the word vector representations. For a preprocessed sentence $S = \{w_1, w_2, \ldots, w_m\}$, its vector can be calculated as shown in Equation (4), where m

is the number of words in the sentence, and MyVectorModel(w_i) is the word vector of w_i generated by the KLD expansion algorithm:

$$\text{Embedding}(S) = \frac{1}{m} \sum_{i=1}^{m} \text{MyVectorModel}(w_i). \quad (4)$$

3. Training the Above Embedding Model Using Soft Labels: Match the preprocessed representation of the sentence with the soft labels and use the mean square error loss function to train the embedding model. The goal is to minimize the mean square error between the outputs of the embedding model, as shown in Equation (5), where N is the number of training samples, Embedding(S_i) is the student model's predicted output for the sentence S_i, and SoftLabel(S) is the soft label generated by the teacher model:

$$L = \frac{1}{N} \sum_{i=1}^{N} |\text{Embedding}(S_i) - \text{SoftLabel}(S_i)|^2. \quad (5)$$

4. Generating New Sentence Embeddings Using the Trained Embedding Model: In Equation (6), Preprocess(S_{new}) is the vector representation obtained after preprocessing the new sentence S_{new}, and f is the trained embedding model.

$$\text{SentenceVector}(S_{new}) = f(\text{Preprocess}(S_{new})). \quad (6)$$

Finally, the Dynamic Principal Component Smoothing (DPCS) algorithm is employed to compute the vector representation of text. This method, by combining the inverse frequency of words and weighted averaging, first calculates the weight of each word and generates a weighted average vector for the sentence. Unlike the traditional approach of simply averaging the word embeddings in a sentence, DPCS dynamically selects and retains the principal components with higher contribution rates, effectively removing low-variance components that are more likely to be noisy. As a result, DPCS preserves the main semantic information of the sentence while reducing the interference from completely irrelevant or noisy words, thereby enhancing the quality of the text representation. Ultimately, the similarity between sentences is computed using cosine similarity.

Compared to the traditional smoothing inverse frequency (SIF) method, DPCS offers significant advantages when handling high-dimensional word embeddings, particularly in its dynamic removal of low-variance components. This approach enables a more precise focus on the core semantic structure of the sentence, avoiding the over-weighting of irrelevant words as seen with simple averaging. The main steps of DPCS include calculating word weights, computing the weighted average vector of the sentence, dynamic principal component removal, and calculating cosine similarity.

- Calculating Word Weights: The weight a_i of each word is calculated based on its word frequency P and a smoothing parameter a, as shown in Equation (7). a is a small adjustment parameter, usually set to 0.001.

$$a_i = \frac{a}{a + P(w_i)}. \quad (7)$$

- Calculating the Weighted Average Vector of the Sentence: Compute the weighted average of the word vectors in the sentence S to obtain the representation vector $v(S)$, as shown in Equation (8), where w_i is the word vector of each word, and $|S|$ is the number of words in the sentence S:

$$v(S) = \frac{1}{|S|} \sum_{w_i \in S} a_i w_i. \quad (8)$$

- Dynamic Principal Component Removal: To eliminate common directions in the text representation, Principal Component Analysis (PCA) can be applied. In PCA, we

calculate the eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of each principal component to measure the variance explained by each component. The variance contribution rate of the j-th principal component γ_j is given by Equation (9):

$$\gamma_j = \frac{\lambda_j}{\sum_{i=1}^{n} \lambda_i}, \tag{9}$$

where λ_j is the eigenvalue of the j-th principal component, and the denominator represents the sum of the eigenvalues of all principal components, indicating the total variance. The cumulative variance contribution Γ_k is obtained by summing the variance contributions of the first k principal components, as shown in Equation (10):

$$\Gamma_k = \sum_{j=1}^{k} \gamma_j = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{n} \lambda_i}. \tag{10}$$

To select an appropriate number of principal components, a threshold T, such as 95%, is typically set to ensure that the cumulative variance contribution reaches this threshold, as shown in Equation (11):

$$\Gamma_k \geq T. \tag{11}$$

When the cumulative variance contribution Γ_k meets or exceeds the threshold T, the first k principal components are selected. Based on the variance contributions calculated in the previous steps, we select the first k principal components, ensuring that their cumulative variance contribution Γ_k meets or exceeds the predefined threshold T. These selected components are retained, while the remaining components (which account for low variance or noise) are discarded. Subsequently, after removing the low-variance components, the adjusted sentence vector $v_{adjusted}(S)$ is obtained as shown in Equation (12):

$$v_{adjusted}(S) = v(S) - \sum_{j=k+1}^{n} \left(v(S) \cdot u_j\right) u_j. \tag{12}$$

- Calculating Cosine Similarity: As shown in Equation (13), for two sentences S_1 and S_2, compute the cosine similarity of their vector representations v_1 and v_2. $v_1 \cdot v_2$ represents the dot product of the two vectors, and $\| v_1 \|$ and $\| v_2 \|$ represent the magnitudes of the two vectors (i.e., the Euclidean norms of the vectors):

$$\text{CosineSimilarity}(v_1, v_2) = \frac{v_1 \cdot v_2}{\| v_1 \| \cdot \| v_2 \|}. \tag{13}$$

4. Experiments

This section describes the experimental design, datasets, evaluation metrics, and baseline methods employed to validate the proposed approaches. The experiments were meticulously crafted to assess the efficacy of the KLD-enhanced sentence embedding method and the RoBERTa-based knowledge distillation framework augmented with Dynamic Principal Component Smoothing (DPCS).

Initially, sentiment analysis experiments were conducted to evaluate the classification performance of the proposed method in distinguishing sentiment polarities. These experiments utilized standard datasets and were assessed using precision, recall, and F1 score as evaluation metrics. By comparing against baseline models such as Word2Vec, GloVe, FastText, BERT, SBERT, and SimCSE, the goal was to demonstrate the method's superiority in capturing nuanced emotional contexts, a foundational aspect of natural language processing.

Following this, our experiments focused on semantic textual similarity (STS), aiming to highlight the robustness and generalization capability of the proposed approaches. Correlation metrics, including Pearson and Spearman coefficients, alongside error metrics

like mean absolute error (MAE), were employed for evaluation. The analysis extended to various similarity computation techniques, such as average vectors, TF-IDF-weighted vectors, and DPCS-based cosine similarity. These comprehensive evaluations underline the proposed methods' ability to effectively address complex NLP tasks across diverse contexts.

4.1. Experiment Preparation

The experimental data for this study were obtained through web scraping technology, specifically targeting English texts from the English version of China Daily, collected on 20 April 2023. The collected data underwent a rigorous cleaning process, which included the removal of stop words and special symbols. Given that the data are in English, spaces were utilized to segment the text. Subsequently, the Word2Vec Skip-gram model was employed to convert the segmented words into 50-dimensional word vectors, thereby establishing a word vector space.

For the sentiment analysis task, the ACL IMDB dataset was used as the benchmark dataset. This dataset contains 50,000 movie reviews labeled with binary sentiment polarity (positive or negative). The dataset is evenly split into 25,000 training samples and 25,000 testing samples, ensuring a balanced and unbiased evaluation.

For the semantic textual similarity (STS) task, the test dataset utilized in this study is the SemEval-2016 dataset [29], which comprises 1379 sentence pairs used for the SemEval task. Each pair is annotated with a relatedness score ranging from 0 to 5, reflecting the average relatedness as judged by 10 different individuals.

Table 2 presents the similarity scores for various sentence pairs, evaluated using both the baseline method and the proposed MyModel. The "Baseline Evaluation" column shows the highest similarity values produced by the existing method, except for the proposed method, while the "MyModel Evaluation" column presents the results from the proposed method. The evaluations are based on a scale of similarity between the sentences in each pair, with MyModel consistently yielding results that are either similar to or slightly better than the baseline evaluation. This comparison demonstrates the effectiveness and refinement of MyModel in capturing sentence similarity.

Table 2. Sentence pair similarity evaluation using baseline and proposed model.

Sentence Pair	Similarity	Baseline Evaluation	MyModel Evaluation
1. Suicide attack kills eight in Baghdad 2. Suicide attacks kill 24 people in Baghdad	2.40	2.12	2.41
1. Ukraine to implement unilateral ceasefire 2. Ukraine offers unilateral ceasefire	4.80	4.56	4.78
1. Beaten Florida teen released in Israel 2. Palestinian teen dies of wounds sustained in Israeli shooting	0.4	0.33	0.39
1. Southwest jet hit nose first 2. Southwest Jet's Nose Gear Landed First	3.6	3.45	3.56

4.2. Related Algorithms

This study incorporates a comprehensive suite of algorithms for analyzing text, beginning with a sentiment classification task prior to computing semantic textual similarity (STS). The sentiment classification task utilized the Stochastic Gradient Descent (SGD) algorithm, chosen for its efficiency and scalability in handling large-scale datasets. The SGD algorithm was applied to train a binary classifier on the ACL IMDB dataset, leveraging its ability to iteratively optimize model parameters and minimize classification errors.

For the subsequent STS computation, established methods for generating sentence vectors were employed to measure similarity scores between sentence pairs. Seven approaches were utilized to generate sentence embeddings: (1) the KLD-enhanced sentence embedding

distance amplification algorithm combined with the pre-trained RoBERTa-based sentence embedding framework proposed in this study, (2) Word2Vec, (3) FastText, (4) the GloVe algorithm, (5) the BERT model, (6) the SBERT model, and (7) the SimCSE model.

Similarity calculations encompassed three primary techniques: (1) TF-IDF-weighted cosine similarity, which emphasizes the significance of unique terms in sentences, (2) average vector cosine similarity, which computes similarity based on mean embeddings, and (3) DPCS cosine similarity, which incorporates term frequency weighting to refine similarity scores. These methodologies were applied to assess the robustness of the proposed framework across both sentiment analysis and semantic similarity tasks.

4.3. Comparative Experiments and Results

For the sentiment analysis task, the experiments were evaluated using three commonly employed metrics in classification tasks: precision, recall, and F1 score. These metrics are essential for assessing the performance of binary classification models, particularly in the context of sentiment analysis.

- Precision measures the proportion of correctly identified positive instances out of all instances predicted as positive. It is particularly useful when the cost of false positives is high, as it ensures that only relevant positive predictions are made.
- Recall, on the other hand, evaluates the proportion of correctly identified positive instances out of all actual positive instances. It is crucial when the cost of false negatives is high, as it ensures that most of the true positives are correctly identified.
- F1 score is the harmonic mean of precision and recall, offering a balanced measure that accounts for both false positives and false negatives. This metric is especially useful when the dataset is imbalanced, as it provides a more comprehensive view of model performance by combining the strengths of precision and recall into a single value.

Table 3 presents the performance of various embedding models evaluated on the ACL IMDB dataset. Word2Vec shows a low F1 score (0.04) due to its low recall (0.02), highlighting limitations in capturing semantic nuances. GloVe performs well with balanced precision (0.73) and recall (0.77), achieving a strong F1 score (0.75). FastText demonstrates high recall (0.85) but moderate precision (0.51), resulting in a lower F1 score (0.64). BERT achieves one of the best F1 scores (0.76) due to its superior recall (0.82). SBERT, while strong in precision (0.67), shows relatively low recall (0.54), leading to a modest F1 score (0.60). SimCSE balances recall (0.81) and precision (0.65), yielding a solid F1 score (0.72). MyModel outperforms all, with the highest precision (0.75), recall (0.88), and F1 score (0.81), showcasing its ability to comprehensively capture textual semantics.

Table 3. Evaluation of embedding models on ACL IMDB dataset (precision, recall, F1 score).

Embedding	Precision	Recall	F1 Score
Word2Vec	0.66	0.02	0.04
GloVe	0.73	0.77	0.75
FastText	0.51	0.85	0.64
BERT	0.71	0.82	0.76
SBERT	0.67	0.54	0.60
SimCSE	0.65	0.81	0.72
MyModel	**0.75**	**0.88**	**0.81**

Bold values indicate the highest results obtained.

For the semantic textual similarity (STS) task, the experiments were evaluated using three commonly employed evaluation metrics in STS tasks: Pearson correlation coefficient (τ), Spearman correlation coefficient (ρ), and mean absolute error (MAE). Pearson and Spearman correlation coefficients are used to evaluate sentence similarity. This approach aims to capture the relational characteristics between sentences comprehensively. The SemEval-2016 dataset measures sentence similarity on a scale of 0 to 5. However, this study adopts a scale of -1 to 1 to align with the computational range of Pearson and Spearman

coefficients. Pearson correlation assesses the linear relationship between sentence similarities. In contrast, Spearman correlation focuses on monotonic relationships, capturing consistent trends in similarity regardless of linearity.

To ensure that these coefficients effectively represent sentence similarity relationships, the -1 to 1 scale was mapped to a 0 to 5 range. This transformation preserves the relative relationships between similarities. It also aligns the scores with the mathematical definitions of Pearson and Spearman coefficients. This ensures consistency and accuracy in model evaluation. By employing this method, this study captures both linear associations and monotonic trends in sentence similarity. This provides more comprehensive and precise evaluation results.

- Pearson correlation coefficient: This statistic measures the strength and direction of the linear relationship between two continuous variables. Its range is between -1 and 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no linear correlation. τ is generally used to measure the correlation between two ordinal variables, as shown in Equation (14), where x_i and y_i are the observed values of variables X and Y, and $\overline{x}$ and $\overline{y}$ are the means of variables X and Y, respectively.

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{14}$$

- Spearman correlation coefficient: This coefficient measures the strength and direction of the monotonic relationship between two variables, not requiring a linear relationship. It is calculated based on the ranks of the variables, and its range is also between -1 and 1. Values close to 1 or -1 indicate strong monotonic positive or negative correlation, respectively, and 0 indicates no monotonic relationship, as shown in Equation (15), where d_i is the difference in ranks of the i-th pair of observations, and n is the number of observations:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{15}$$

- Mean absolute error (MAE): This metric measures the difference between predicted and actual values in a regression model. It calculates the average of the absolute differences between predicted and actual values.

Table 4 presents the results of average vector cosine similarity. The model proposed in this paper outperforms other methods in terms of τ, ρ, and MAE. Specifically, the τ value of this model is 0.470, the ρ value is 0.481, and the MAE value is 2.100, significantly better than the Word2Vec, GloVe, and FastText methods. Compared to the BERT model, this model shows slight improvements in τ and ρ and also demonstrates lower error in MAE. Notably, SBERT achieves a ρ value of 0.482, slightly surpassing the proposed model's ρ of 0.481, while the MAE of SBERT is 2.093, slightly better than that of the proposed model. Similarly, SimCSE delivers a strong performance with a ρ value of 0.490, the highest among all methods, but its τ value of 0.458 and MAE of 2.104 remain slightly behind the proposed model in terms of overall balance.

The average vector of sentences lacks sufficient utilization of statistical information and fails to account for the distribution characteristics of words in different texts. In calculating text similarity, the TF-IDF-weighted method better captures subtle differences between texts. By assigning different weights to different words, it can more accurately measure the similarity of text content, especially in scenarios involving long texts or requiring highly precise matching. Table 5 presents the comparison results of TF-IDF-weighted cosine similarity. The model proposed in this paper significantly outperforms other models in all metrics, with a τ value of 0.528, a ρ value of 0.518, and an MAE value of 1.343. This indicates that the model better captures the semantic similarity between sentences when

considering word importance weights. Therefore, using TF-IDF-weighted averaging to calculate sentence vectors fully accounts for the distribution of words in the corpus. By adjusting weights through IDF, it reduces the influence of common words and increases the distinction between texts. This use of statistical information enhances the model's ability to differentiate texts.

Table 4. Averaging all the word vectors for the SemEval-2016 dataset.

Embedding	τ	ρ	MAE
Word2Vec	0.044	0.379	2.313
GloVe	0.368	0.356	2.256
FastText	0.064	0.472	2.447
BERT	0.451	0.457	2.181
SBERT	0.462	0.482	**2.093**
SimCSE	0.458	**0.490**	2.104
MyModel	**0.470**	0.481	2.100

Bold values indicate the highest results obtained.

Table 5. Averaging all word vectors weighting them with TF-IDF for SemEval-2016 dataset.

Embedding	τ	ρ	MAE
Word2Vec	0.045	0.387	2.315
GloVe	0.473	0.460	2.046
FastText	0.085	0.364	2.340
BERT	0.486	0.492	2.057
SBERT	0.516	0.511	1.672
SimCSE	0.507	0.503	1.947
MyModel	**0.528**	**0.518**	**1.343**

Bold values indicate the highest results obtained.

The TF-IDF-weighted averaging method requires calculating the frequency and inverse document frequency for each word and then performing weighted averaging based on these values. This method's calculation process is relatively complex, especially when processing large-scale text data, resulting in a heavy computational burden. Although the DPCS method also requires the computation of word frequencies, the DPCS algorithm simplifies the weight calculation by introducing a smoothing coefficient and further optimizes the process by subtracting shared information, resulting in a more efficient overall computation. Table 6 presents the results of cosine similarity calculated using the DPCS algorithm. The model proposed in this paper again excels in terms of τ, ρ, and MAE, with values of 0.530, 0.518, and 1.320, respectively. Compared to other models, this model demonstrates stronger robustness and accuracy in the smooth inverse frequency method. Therefore, the DPCS algorithm provides a more comprehensive theoretical basis, and its concept of smooth inverse document frequency is theoretically supported, making the model more rigorous and interpretable.

Table 6. Dynamic Principal Component Smoothing for SemEval-2016 dataset.

Embedding	τ	ρ	MAE
Word2Vec	0.444	0.432	1.380
GloVe	0.436	0.421	1.423
FastText	0.474	0.467	1.376
BERT	0.451	0.457	2.081
SBERT	0.511	0.501	1.390
SimCSE	0.525	0.514	1.434
MyModel	**0.530**	**0.518**	**1.320**

Bold values indicate the highest results obtained.

Figure 5 shows the overall changes in the model's loss, Pearson correlation (τ), Spearman correlation (ρ), and mean absolute error (MAE) during the training process. The model rapidly optimized in the first 200 training rounds, after which the various metrics stabilized, indicating good convergence. In the training of deep learning models, loss functions are primarily employed to optimize model parameters. However, as the loss value approaches zero, gradients often diminish significantly, resulting in a slower and less stable optimization process. Conversely, evaluation metrics such as Pearson correlation, Spearman correlation, and MAE demonstrate higher sensitivity to subtle variations in prediction outcomes. This indicates that even when the loss function stabilizes, evaluation metrics may exhibit fluctuations due to the local characteristics of specific dataset subsets. Furthermore, loss functions are typically computed as the mean or aggregate across all training samples, whereas evaluation metrics prioritize the localized features of the test set. Differences in model performance on these local features can substantially impact metric fluctuations. This distinction underscores the decoupling between loss functions and evaluation metrics and highlights the critical role of local characteristics in comprehensive model evaluation. Moreover, as the training progressed, the correlation metrics remained at high levels, and the mean absolute error gradually decreased. Although there were slight fluctuations in the model during the later stages of training, its overall performance reached an ideal state, reflecting the model's strong generalization capability.

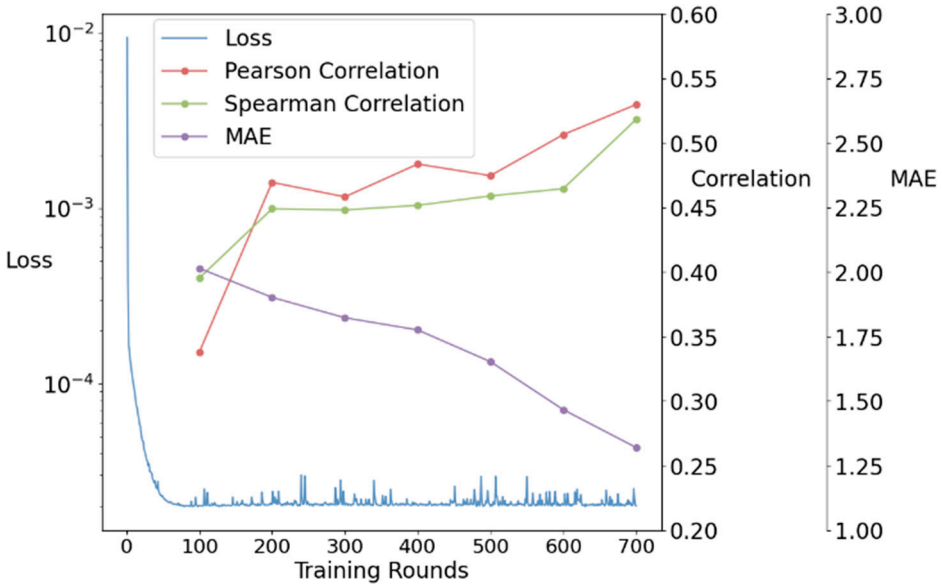


Figure 5. Comprehensive analysis of loss and metrics over training rounds.

5. Conclusions

This study demonstrates the effectiveness of an enhanced sentence vector generation method for semantic text similarity (STS) calculations and sentiment analysis tasks. The sentiment analysis experiment, conducted on the ACL IMDB dataset, employed a Stochastic Gradient Descent (SGD) classifier and achieved high performance across precision, recall, and F1 score metrics. These results validate the capability of the proposed methods in accurately classifying sentiment and enhancing the downstream integration of sentiment-aware features.

In STS computations, the optimization of KLD significantly improves the discriminative power of sentence vectors in the semantic space, enabling the model to better capture subtle semantic differences between sentences. Additionally, the sentence embedding generation framework based on RoBERTa knowledge distillation integrates the deep semantic information of the pre-trained language model with the aforementioned method, resulting in higher semantic accuracy and computational efficiency.

To further enhance the model's performance, this study introduces a feature selection technique that automatically selects the most relevant features to the task from the sentence embeddings generated by RoBERTa, thereby reducing redundant information and further optimizing the effectiveness of the sentence representations. Test results on the SemEval-2016 dataset indicate that this method surpasses other approaches across various similarity calculation evaluation metrics, particularly excelling in the DPCS-weighted cosine similarity evaluation, where it achieves the best performance. This framework exhibits strong robustness and generalization capabilities in handling complex natural language tasks and marginally outperforms the BERT model in experiments, underscoring the potential of knowledge distillation in enhancing model performance.

Overall, the improved method proposed in this paper demonstrates superior performance in both semantic text similarity calculations and sentiment analysis tasks, offering a more effective and flexible solution, especially in terms of semantic representation, model efficiency, and robustness.

Author Contributions: The initial concept for the proposed method was developed by J.H., who also designed the experimental setup and performed the data analysis. L.Y. played a key role in implementing the Dynamic Principal Component Smoothing (DPCS) algorithm, contributed to the creation of the evaluation metrics, and offered valuable feedback on the methodology. Both authors collaborated on drafting and revising the manuscript, providing significant contributions to the study. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to privacy concerns.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, Y.; Afzal, N.; Fu, S.; Wang, L.; Shen, F.; Rastegar-Mojarad, M.; Liu, H. MedSTS: A Resource for Clinical Semantic Textual Similarity. *Lang Resour. Eval.* **2020**, *54*, 57–72. [CrossRef]
2. Kang, Y.; Cai, Z.; Tan, C.-W.; Huang, Q.; Liu, H. Natural Language Processing (NLP) in Management Research: A Literature Review. *J. Manag. Anal.* **2020**, *7*, 139–172. [CrossRef]
3. Wang, L.; Yoon, K.-J. Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3048–3068. [CrossRef] [PubMed]
4. Yao, T.; Zhai, Z.; Gao, B. Text Classification Model Based on fastText. In Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, 20–22 March 2020; pp. 154–157.
5. Bekamiri, H.; Hain, D.S.; Jurowetzki, R. PatentSBERTa: A Deep NLP Based Hybrid Model for Patent Distance and Classification Using Augmented SBERT. *Technol. Forecast. Soc. Chang.* **2024**, *206*, 123536. [CrossRef]
6. Mokoatle, M.; Marivate, V.; Mapiye, D.; Bornman, R.; Hayes, V.M. A Review and Comparative Study of Cancer Detection Using Machine Learning: SBERT and SimCSE Application. *BMC Bioinform.* **2023**, *24*, 112. [CrossRef]
7. Alammary, A.S. Arabic Questions Classification Using Modified TF-IDF. *IEEE Access* **2021**, *9*, 95109–95122. [CrossRef]
8. Kirişci, M. New Cosine Similarity and Distance Measures for Fermatean Fuzzy Sets and TOPSIS Approach. *Knowl. Inf. Syst.* **2023**, *65*, 855–868. [CrossRef]
9. Li, Z.; Yang, Y.; Li, L.; Wang, D. A Weighted Pearson Correlation Coefficient Based Multi-Fault Comprehensive Diagnosis for Battery Circuits. *J. Energy Storage* **2023**, *60*, 106584. [CrossRef]
10. Chatterjee, S. A New Coefficient of Correlation. *J. Am. Stat. Assoc.* **2021**, *116*, 2009–2022. [CrossRef]
11. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011*; Lin, D., Matsumoto, Y., Mihalcea, R., Eds.; Association for Computational Linguistics: Portland, OR, USA, 2011; pp. 142–150.
12. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
13. Maiese, A.; Santoro, P.; La Russa, R.; De Matteis, A.; Turillazzi, E.; Frati, P.; Fineschi, V. Crossbow Injuries: A Case Report with Experimental Reconstruction Study and a Systematic Review of Literature. *J. Forensic Leg. Med.* **2021**, *79*, 102147. [CrossRef] [PubMed]
14. Du, X.; Yan, J.; Zhang, R.; Zha, H. Cross-Network Skip-Gram Embedding for Joint Network Alignment and Link Prediction. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 1080–1095. [CrossRef]

15. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014*; Moschitti, A., Pang, B., Daelemans, W., Eds.; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543.
16. Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; Mikolov, T. FastText.Zip: Compressing Text Classification Models. *arXiv* **2016**, arXiv:1612.03651.
17. Sarzynska-Wawer, J.; Wawer, A.; Pawlak, A.; Szymanowska, J.; Stefaniak, I.; Jarkiewicz, M.; Okruszek, L. Detecting Formal Thought Disorder by Deep Contextualized Word Representations. *Psychiatry Res.* **2021**, *304*, 114135. [CrossRef]
18. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019*; Volume 1 (Long and Short Papers); Burstein, J., Doran, C., Solorio, T., Eds.; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186.
19. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
20. Suwanda, R.; Syahputra, Z.; Zamzami, E.M. Analysis of Euclidean Distance and Manhattan Distance in the K-Means Algorithm for Variations Number of Centroid K. *J. Phys. Conf. Ser.* **2020**, *1566*, 012058. [CrossRef]
21. Fernando, B.; Herath, S. Anticipating Human Actions by Correlating Past with the Future with Jaccard Similarity Measures. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; IEEE: Nashville, TN, USA, 2021; pp. 13219–13228.
22. Chakraborty, S.; Ding, Y.; Allamanis, M.; Ray, B. CODIT: Code Editing with Tree-Based Neural Models. *IEEE Trans. Softw. Eng.* **2022**, *48*, 1385–1399. [CrossRef]
23. Chen, W.; Li, Y.; Tsangaratos, P.; Shahabi, H.; Ilia, I.; Xue, W.; Bian, H. Groundwater Spring Potential Mapping Using Artificial Intelligence Approach Based on Kernel Logistic Regression, Random Forest, and Alternating Decision Tree Models. *Appl. Sci.* **2020**, *10*, 425. [CrossRef]
24. Huang, G.; Quo, C.; Kusner, M.J.; Sun, Y.; Weinberger, K.Q.; Sha, F. Supervised Word Mover's Distance. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona Spain, 5–10 December 2016; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 4869–4877.
25. Arora, S.; Liang, Y.; Ma, T. A Simple But Tough-To-Beat Baseline for Sentence Embeddings. 6 February 2017. Available online: https://dblp.org/rec/conf/iclr/AroraLM17.html (accessed on 11 October 2024).
26. Bu, Y.; Zou, S.; Liang, Y.; Veeravalli, V.V. Estimation of KL Divergence: Optimal Minimax Rate. *arXiv* **2018**, arXiv:1607.02653. [CrossRef]
27. Namoun, A.; Alshanqiti, A. Predicting Student Performance Using Data Mining and Learning Analytics Techniques: A Systematic Literature Review. *Appl. Sci.* **2021**, *11*, 237. [CrossRef]
28. Deng, J.; Li, W.; Chen, Y.; Duan, L. Unbiased Mean Teacher for Cross-Domain Object Detection. *arXiv* **2021**, arXiv:2003.00707.
29. Agirre, E.; Banea, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Mihalcea, R.; Rigau, G.; Wiebe, J. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), San Diego, CA, USA, 16–17 June 2016*; Bethard, S., Carpuat, M., Cer, D., Jurgens, D., Nakov, P., Zesch, T., Eds.; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 497–511.