Proceedings of CONF-MLA 2024 Workshop: Securing the Future: Empowering Cyber Defense with Machine Learning and Deep Learning
DOI: 10.54254/2755-2721/94/2024MELB0084

An Empirical Exploration of the Effectiveness of Word Embedding Techniques for Emotion Recognition

Puan Wang

School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, 210023, China

b21030722@njupt.edu.cn

Abstract. Emotion recognition is an important research direction in the field of natural language processing. It is widely used in social media monitoring, product feedback analysis, and medical diagnosis, and it has significant practical value. The task of emotion recognition aims to recognize the emotional orientation of text through analysis. Word embedding technology is a key component in emotion recognition tasks; it helps the model capture sentiment relationships within the context by converting words into dense vectors, which is crucial for the model's performance. This paper is based on the Internet Movie Database (IMDB) and investigates the performances of different word embedding methods, including Rand, Static, Non-static, and Multi-channel, in sentiment recognition tasks. It also evaluates their effects on various neural network architectures. The experimental results show that random initialization performs best, especially in complex networks, demonstrating strong classification capability and training efficiency. This indicates that deep learning models have strong adaptive capabilities in sentiment recognition tasks. Even without the help of pre-trained word vectors, the model can still effectively capture the sentiment information in text.

Keywords: Emotion recognition, deep learning, word embedding.

1. Introduction

Emotion recognition is an important research direction in the field of natural language processing. As the internet becomes more widespread, users produce massive amounts of text data on platforms such as social media, movie reviews, and product feedback [1,2]. These data contain abundant emotional information. Product developers could automatically analyze and extract sentiment tendencies through emotion recognition techniques, which could provide significant value in many practical applications. For example, in the business field, emotion recognition could help companies better understand customers' emotions to improve their products or services. In social media monitoring, it could be used to track real-time fluctuations in public sentiment to provide decision support. Additionally, in the medical field, emotion recognition could help analyze patients' moods and assist in diagnostic and treatment processes.

Neural networks and deep learning methods have powerful capabilities in processing complex data structures, especially showing significant advantages in natural language processing tasks [3,4]. The word embedding method is one of the core techniques in deep learning for representing textual semantics

DOI: 10.54254/2755-2721/94/2024MELB0084

in text sentiment analysis tasks. Different embedding methods can capture different levels of semantic information and show varying effects in the task.

A great embedding method can effectively capture contextual relationships and accurately represent the semantics of low-frequency words, which could improve the model's generalization capability [5,6]. However, choosing inappropriate word embedding methods will cause the model to not fully understand the semantic relations between the texts. This can limit the model's generalization ability, especially in processing low-frequency words and complex contexts. Such limitations will manifest in subsequent emotion recognition tasks as a decrease in classification abilities or an inability to adapt to new data.

Therefore, it is crucial to choose an appropriate word embedding method for emotion recognition tasks. It can not only improve the model's classification capability but also effectively avoid overfitting or underfitting problems during training. The focus of this research is to explore the impact of different word embedding methods on the model's emotional recognition ability. Specifically, this paper will evaluate the performance of random initialization, static, non-static, and multi-channel embedding methods across various neural network structures, including text Convolutional Neural Network (TextCNN), Recurrent Neural Network (RNN), Bi-directional RNN (BiRNN) [7]. By using experiments to compare the effectiveness of these embedding methods on emotion recognition tasks, this paper could provide effective guidance for designing emotional classification structures.

2. Methodologies

This research explores the impact of different word embedding methods on several base neural network models [7,8]. This section provides a detailed introduction to the types of embedding methods and neural network structures used in this study.

2.1. Word embedding approaches

2.1.1. Rand. This type of initialization uses random assignment to populate the embedding layer's matrix, with each word being assigned a word vector randomly. Typically, it employs either a uniform distribution or a normal distribution to generate the vector representations. This method relies entirely on the model to learn effective word representations, meaning it is not restricted by a pre-trained corpus and can adaptively learn features specific to tasks.

2.1.2. Static. This initialization technique uses pre-trained word vector models, such as FastText, Word2Vec, and Global Vectors for Word Representation (GloVe) [9]. These models are typically trained on large-scale corpora, enabling them to learn rich and generalizable semantic information. The term "static" indicates that the word vector representations will not be fine-tuned during training, which can make the word vector representations more stable throughout the training process.

2.1.3. Non-static. This initialization method also employs a pre-trained word vector model to generate the embedding matrix. However, unlike the static method, the embedding matrix in this approach can be updated during the training process. It utilizes a pre-trained model to obtain the initial values, which are then fine-tuned through backpropagation. This allows the word vectors to adapt to the specific task and dataset.

2.1.4. Multi-channel. In this method, multi-channel refers to the combination of static and non-static embedding methods to create multiple channels, similar to multi-channel image inputs in computer vision. Both channels use a pre-trained word vector model, but one channel is trainable while the other is not. They are then combined and fed into the neural network model, which can make the embedding process more stable during training while still allowing for sufficient flexibility to fine-tune itself.

DOI: 10.54254/2755-2721/94/2024MELB0084

2.2. Neural network modules

This research adopts several deep learning models as the basic modules. These models each have their own advantages in natural language processing tasks [8,10].

2.2.1. TextCNN. TextCNN is an implementation of CNN in text classification. It can effectively capture local n-gram features. The various types of convolutional layers extract text features at different granularities through sliding windows. Therefore, it is well-suited for handling short texts and identifying important local features in a task.

2.2.2. RNN. RNN models sequence data through a recursive structure with shared parameters, making it suitable for processing sequential data and capturing contextual dependencies in text sequences. However, because it uses shared parameters to process the data, it is prone to the vanishing gradient problem. Additionally, it may struggle to capture long-range dependencies.

2.2.3. Long Short-Term Memory (LSTM). LSTM is an improved version of the RNN, specifically designed to alleviate the vanishing gradient problem. By introducing memory cells and gating mechanisms, the LSTM can effectively capture long-range dependencies and performs particularly well in handling long texts.

2.2.4. Gated Recurrent Unit (GRU). GRU is a simplified version of the LSTM. It eliminates the memory cell and directly stores information in the hidden layers. This modification reduces its parameter count and can make it more computationally efficient than the LSTM. Its structure is particularly efficient for handling medium-length text sequences but may perform worse than LSTM in tasks that require the ability to process long-range dependencies.

2.2.5. BiRNN, BiGRU, BiLSTM. "Bi-" means bidirectional, indicating that these network layers use a bidirectional recurrent neural network structure. In traditional unidirectional RNNs, LSTMs, and GRUs, input data is passed from the start to the end of the sequence, and the network only considers past information. However, in bidirectional networks, the neural network passes information simultaneously from both the start and the end, and the outputs from both directions are integrated together. This allows the model to consider not only past information but also future information for prediction. Specifically, there are two sub-networks in a bidirectional network: one processes the forward sequence, and the other processes the backward sequence. Their outputs are combined for subsequent processing. The bidirectional structure is often used in natural language processing tasks, such as translation and sentiment classification, where information at the current moment depends not only on past words but also on subsequent words.

2.3. Architectures of neural network

In this research, each neural network consists of an input layer, an embedding layer, a core layer, such as TextCNN, RNN, LSTM, etc., and an output layer. The structures of all neural networks are shown in Figure 1.

The text sequence is input to each model as a tensor, and the embedding layer converts it from a vector to a word vector matrix. This matrix is then passed to the core neural network. In TextCNN, the convolutional layer uses convolutional kernels of different sizes (2, 3, 4, 5) to extract n-gram features of various granularities. Following this, a global max pooling layer is used to extract the most important features and concatenate them. In other models (RNN, BiRNN, LSTM, BiLSTM, GRU, BiGRU), the word vector matrix is fed into the respective RNN/BiRNN/LSTM/BiLSTM/GRU/BiGRU layer, with the size of the hidden layer set to 128. After passing through the core neural network, data enters a dense layer with 250 neurons, which use Rectified Linear Unit (ReLU) as the activation function. Subsequently, there is a dropout layer with a dropout rate of 0.3. Finally, data is classified by the output layer, which has a single neuron with a sigmoid activation function.

DOI: 10.54254/2755-2721/94/2024MELB0084

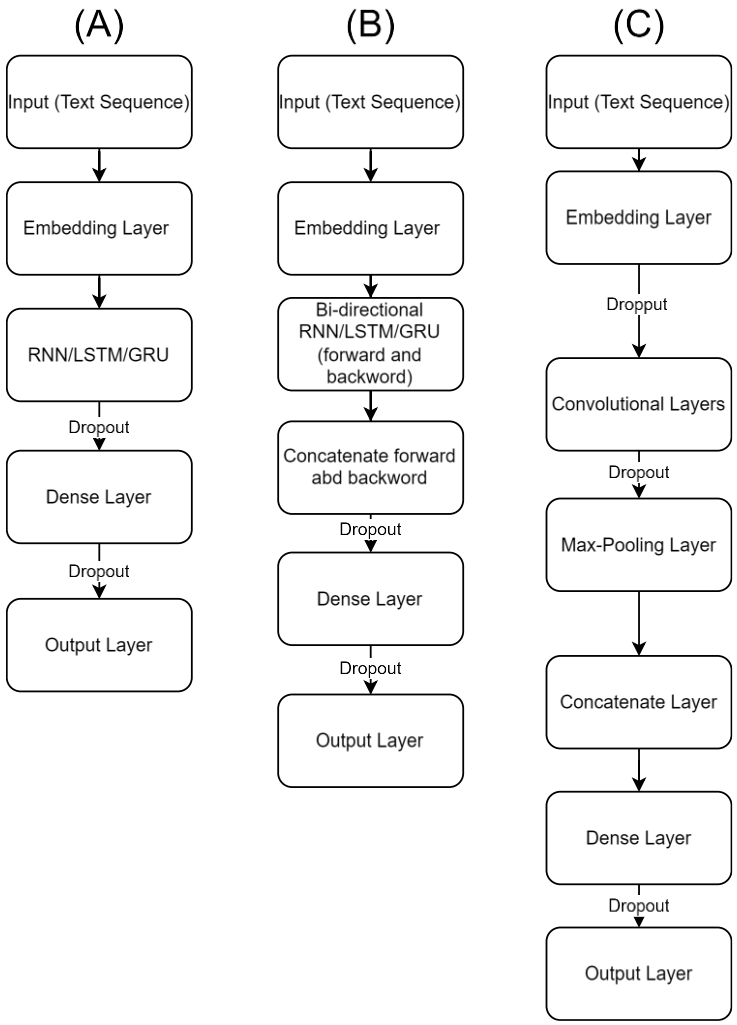

Figure 1. Model architectures of (a) RNN/LSTM/GRU, (b) Bi-RNN/LSTM/GRU, and (c) CNN (Figure Credits: Original).

2.4. Evaluation metrics

To comprehensively evaluate the model's performance, this research employs five commonly used classification evaluation metrics: accuracy, precision, recall, F1 score, and the time required for the models to achieve the optimal valid loss. Through these evaluation metrics, researchers can measure the model's performance from multiple dimensions, including classification accuracy, as well as the model's training efficiency and time cost.

3. Results and discussion

3.1. Dataset and preprocessing

In this research, the program utilizes the Internet Movie Database (IMDB) which was published by Stanford University [11]. This dataset includes a large number of movie reviews from the Internet Movie Database, and it has been preprocessed to fit the binary classification task for sentiment analysis. The dataset is commonly used for sentiment analysis tasks in testing classification, with each review labeled as positive or negative.

The dataset consists of 50,000 movie reviews, with 25,000 allocated for training and 25,000 for testing. It contains an equal number of positive and negative reviews, which helps avoid the class

DOI: 10.54254/2755-2721/94/2024MELB0084

imbalance problem. Each review is presented in English text, and all reviews have corresponding labels, where 0 indicates a negative comment and 1 indicates a positive comment. To prevent neutral critiques from causing confusion, each comment selected for the dataset has a clear emotional inclination. For a further understanding of sentence length, the distribution of sentence lengths is shown in Figure 2.

To simplify calculations, the program retains only the most common 20,000 words, discarding those that rarely emerge. All film reviews have been padded or truncated to the same length. In this research, statistical analysis was conducted to determine the optimal sentence length, which helps avoid excessive padding or truncation that could lead to significant loss of sentiment information.

Figure 2. Distribution of sentence length (Figure Credits: Original).

3.2. Training details

This research used the FastText pre-trained model "crawl-300d-2M-subword" to perform word vector representation. This model is developed by Meta's AI research group and is trained on a large-scale web corpus from Common Crawl. It contains 2 million words and uses subword information to build the embedding model. Given that user comments often contain non-standard spellings and rare phrases, using the crawl-300d-2M-subword model can aid in analyzing these words.

All of these models are based on the TensorFlow and Keras frameworks and are run in NVIDIA GeForce RTX 4090 GPU environments. For training, all models used the Adam optimizer with its default settings for the learning rate to ensure consistency across experiments. The batch sizes were adjusted for different model architectures to balance computational efficiency and VRAM usage, with TextCNN using a batch size of 2048, LSTM and BiLSTM using 512, and RNN, BiRNN, GRU, and BiGRU using a batch size of 1024. Training was capped at 100 epochs, with an early stopping mechanism in place to halt training after 10 epochs without improvement in validation loss. The output layers of all models used a sigmoid activation function suitable for binary classification, while the hidden dense layers employed ReLU to facilitate calculation and avoid vanishing gradient issues. To further combat overfitting, L2 regularization was applied to TextCNN models, and a dropout rate of 0.3 was uniformly incorporated across all models. The loss function used for training was Binary Crossentropy, which is appropriate for the binary classification task.

DOI: 10.54254/2755-2721/94/2024MELB0084

3.3. Performance comparison

This research compares the performances of four different types of word embedding methods—Random Initialization, Static Pre-trained Embeddings, Non-static Pre-trained Embeddings, and Multi-channel Embeddings—across various model structures (CNN, RNN, LSTM, GRU, BiRNN, BiLSTM, BiGRU). To mitigate the impact of random factors during model training on the conclusions, the program was run individually five times for each setup. This work recorded the maximum value of validation accuracy from each experiment and used its average value as the final performance indicator, as listed in Table 1. Additionally, the program recorded the average time taken for each network structure to achieve the best validation loss, as shown in Table 2.

Table 1. Comparison of accuracy across various word embedding methods and models.

	Rand	Static	NonStatic	MultiChannel
BiGRU	0.8682	0.7803	0.8579	0.8641
BiLSTM	0.8794	0.7766	0.8780	0.8782
BiRNN	0.8461	0.6470	0.8340	0.8203
CNN	0.8914	0.8600	0.8858	0.8653
GRU	0.8528	0.8028	0.8592	0.8564
LSTM	0.8755	0.7456	0.8693	0.8668
RNN	0.8474	0.5762	0.8337	0.8040

Table 2. Comparison of time cost across various word embedding methods and models.

	Rand	Static	NonStatic	MultiChannel
BiGRU	16.73	142.12	18.38	19.87
BiLSTM	17.47	176.14	19.41	20.96
BiRNN	58.71	154.83	54.89	56.93
CNN	48.70	325.70	52.33	421.83
GRU	10.33	108.35	9.98	13.26
LSTM	9.372	88.64	11.19	12.42
RNN	25.30	34.49	33.22	36.10

4. Discussion

In this research, the random initialization word embedding method performed well in the vast majority of models. As Table 1 shows, random initialization became the best embedding method in almost all types of neural network architectures except for the GRU. This result breaks the regular expectation that using pre-trained word embedding models should generally be better than randomly initializing the word embedding matrix and training it on the dataset. In fact, the random embedding method even achieved a very high validation accuracy in the TextCNN structure (0.8914). It demonstrates that even without using a pre-trained embedding model, the random initialization method still has enough capability to help the deep learning model learn effective feature representations from the training dataset, especially in complex networks such as TextCNN and BiLSTM. On the other hand, this embedding method also became the fastest converging method among all the embedding methods except in the GRU structure.

Even though pre-trained word vector models contain high-level semantic information, the experimental conclusion shows that pre-trained embeddings (Static and Non-static) did not achieve the best performance in most neural network structures. It also shows that using a non-static embedding method is significantly better than using a static embedding method. Readers can see a significant decrease in accuracy in Table 1 when changing the embedding method from non-static to static. The most severe decline is in the RNN structure (non-static's accuracy is 0.8337, but static's accuracy is only 0.576183999), which is close to randomly guessing the classification.

DOI: 10.54254/2755-2721/94/2024MELB0084

From the perspective of time cost, as shown in Table 2, using a static embedding method takes several times longer than other embedding methods, except in the RNN. However, considering its average accuracy in the RNN structure is only 57.62%, it is fair to say that this model is unable to fit properly.

Multi-channel embeddings combine static and non-static word vectors and could be more stable than just non-static and more flexible than static embeddings. In BiGRU and BiLSTM network structures, their accuracy is close to that of random initialization. However, Table 1 also shows that its validation accuracy is worse than that of non-static in five out of the seven models. In the two models where multi-channel is better, the multi-channel word embedding method only slightly outperforms the Non-static method in terms of accuracy (by less than 0.007). On the other hand, the multi-channel embedding method takes more time for models to fit in all kinds of structures, as shown in Table 2. Especially in TextCNN tasks, it takes nearly eight times longer to fit the model and ultimately results in poorer accuracy.

Overall, the random initialization method performs best in the binary classification of emotions. It shows that deep learning networks can effectively train themselves to represent the emotional inclination of words from random initialization. The significantly different performance between the static embedding method and the non-static and multi-channel embedding methods demonstrates the big difference between models that can optimize the embedding matrix and those that cannot. This also indirectly highlights the ability of deep learning models to train word embedding matrices to fit text sentiment classification tasks.

5. Conclusion

This text uses the IMDB dataset to explore different word embedding methods across multiple neural network structures in sentiment classification tasks. The dataset includes 50,000 movie reviews that have been pre-processed, with 25,000 reviews in the training dataset and 25,000 in the test dataset. Both contain a balanced number of positive and negative reviews, making it suitable for binary classification tasks.

This research compares the performance of random initialization, static pre-trained initialization, non-static pre-trained initialization, and multi-channel embedding methods on various neural networks (TextCNN, LSTM, BiLSTM, etc.). The results show that random initialization performs best on most models, especially achieving high accuracy with TextCNN and BiLSTM (TextCNN reaching 0.8914, BiLSTM reaching 0.8794). This indicates that random initialization can effectively help models study and capture textual emotional features even without information from pre-trained word vectors.

Although pre-trained word embedding models typically contain rich semantic information, experiments show that the non-static pre-trained embedding method is significantly better than the static method. Particularly in RNN models, using a static embedding method leads to a substantial drop in the model's accuracy, approaching the level of random guessing (only 0.5762). In contrast, using a non-static approach can dramatically improve the model's performance.

Although multi-channel word embeddings combine static and non-static methods, their performance is still slightly inferior to that of using non-static embedding methods in most models. Additionally, the research finds that random initialization not only outperforms other embedding methods in terms of validation accuracy but also has a clear advantage in training time. For instance, random initialization is significantly faster than static embedding, with an average training time of only 17.47 seconds compared to 176.14 seconds needed for static embedding methods.

In summary, this research indicates that the random initialization method performs exceptionally well in text sentiment classification tasks, especially in more complex neural network models such as TextCNN and BiLSTM. Even without relying on pre-trained embeddings, random initialization can enable the model to effectively learn word emotional features suitable for classification. This result demonstrates that deep learning models have strong self-learning abilities in sentiment classification tasks and can effectively capture word sentiment information from random initialization. The static word embedding method is markedly behind the non-static and multi-channel methods, further highlighting the importance of fine-tuning word vectors during the training process in sentiment analysis tasks.

DOI: 10.54254/2755-2721/94/2024MELB0084

References

[1] Zhang, J., Yin, Z., Chen, P., & Nichele, S. (2020). Emotion recognition using multi-modal data and machine learning techniques: A tutorial and review. Information Fusion, 59, 103-126.

[2] Bota, P. J., Wang, C., Fred, A. L., & Da Silva, H. P. (2019). A review, current challenges, and future possibilities on emotion recognition using machine learning and physiological signals. IEEE access, 7, 140990-141020.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[4] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. Neurocomputing, 187, 27-48.

[5] Li, Y., & Yang, T. (2018). Word embedding for understanding natural language: a survey. Guide to big data applications, 83-104.

[6] Wadud, M. A. H., Mridha, M. F., & Rahman, M. M. (2022). Word embedding methods for word representation in deep learning for natural language processing. Iraqi Journal of Science, 1349-1361.

[7] Yoon, K. (2014). Convolutional Neural Networks for Sentence Classification. . arXiv preprint arXiv:1408.5882

[8] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26.

[9] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing, 1532-1543.

[10] Smys, S., Chen, J. I. Z., & Shakya, S. (2020). Survey on neural network architectures with deep learning. Journal of Soft Computing Paradigm (JSCP), 2(03), 186-194.

[11] Lakshmipathi, N. (2018). IMDB Dataset of 50K Movie Reviews. URL: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews. Last Accessed: 2024/09/12