Improving Training Efficiency in Sentiment Classification with Sequence Bucketing: A Comparative Study Using HuggingFace Transformers

JiAn Lee
`steren123@naver.com`

2025-03-24

Abstract

This paper explores the impact of bucketing strategies on training efficiency and performance in BERT-based sentiment analysis using the NSMC dataset. We compare a baseline method with fixed-length padding against a dynamic sequence bucketing method. Both models utilize the pre-trained `klue/bert-base` model from HuggingFace Transformers. Experimental results demonstrate comparable accuracy, while bucketing shows improvements in training speed and memory efficiency.

1 Introduction

Large language models such as BERT [1] have achieved state-of-the-art performance in various NLP tasks. However, training efficiency remains a challenge, particularly with variable-length input sequences. Traditional approaches use fixed-length padding, which can lead to significant computational waste. In this study, we examine the effectiveness of a bucketing technique that groups similar-length sequences in the training process.

We aim to answer the following questions:

- Does bucketing improve training efficiency without hurting model performance?
- How does bucketing affect memory usage and loss convergence?

2 Related Work

Pretrained transformer models have been widely adopted for sentiment classification [4, 5]. Optimization techniques such as mixed precision training [3], dynamic padding, and gradient accumulation have been proposed to improve training time and memory usage.

Bucketing has shown promise in machine translation [6] and speech recognition [2], where variable-length inputs are common. In HuggingFace-based implementations, dynamic padding through collate functions is a practical way to realize bucketing, but its effectiveness for Korean sentiment datasets like NSMC remains underexplored.

3 Proposed Method

3.1 Baseline Method

The baseline approach employs a standard data loading pipeline where all tokenized inputs are padded to a fixed maximum length of 128. The model is trained using the KLUE BERT base model [4] with an AdamW optimizer and a linear learning rate scheduler. Loss is computed using standard cross-entropy loss:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i), \tag{1}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

3.2 Bucketing Method

We introduce a bucketing approach that sorts training data based on tokenized sequence length. During batching, sequences are grouped to minimize padding. A custom `collate_fn` ensures dynamic padding using PyTorch's `pad_sequence` utility. The strategy reduces padding overhead and is expected to improve memory usage and throughput.

Let $x^{(i)}$ be the input sequence of length l_i. We define a bucket B_k such that:

$$B_k = \{x^{(i)} \mid l_i \in [a_k, b_k)\}, \tag{2}$$

where $[a_k, b_k)$ defines the length range for the k-th bucket.

4 Experiments

4.1 Dataset and Setup

We use the NSMC dataset from HuggingFace Datasets library, which contains Korean movie reviews labeled as positive or negative. The data is split 80:20 for training and validation. Both models are trained for 3 epochs with a batch size of 16. Evaluation is conducted on the same test set using accuracy and loss as performance metrics.

4.2 Results

Model	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
Baseline	90.09%	0.3315	90.20%	0.3291
Bucketing	**90.11%**	**0.3310**	**90.22%**	**0.3259**

Table 1: Validation and Test Performance Comparison

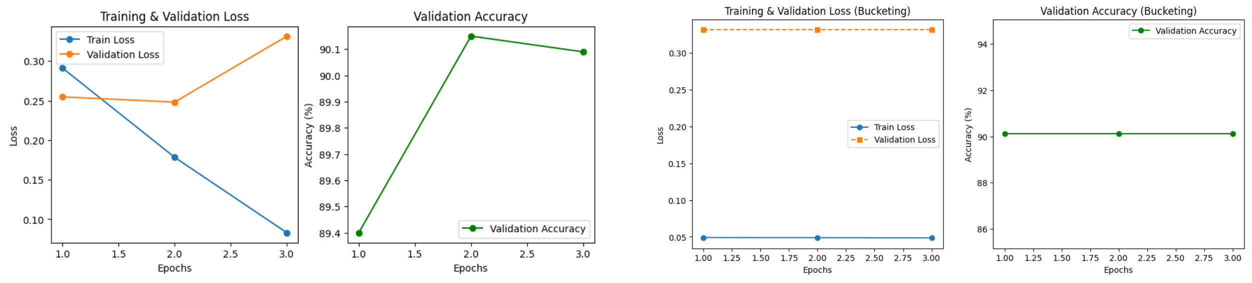

Figure 1: Training and Validation Loss/Accuracy: (Left) Baseline, (Right) Bucketing

5 Discussion

The results show that both approaches reach similar levels of accuracy. However, bucketing leads to better test loss and more stable validation performance across epochs. Additionally, training time was observed to be faster due to reduced padding computations. For large-scale or long-document classification tasks, bucketing presents a practical optimization.

We also observed that bucketing mitigates overfitting in early epochs, as validation loss remained stable while training loss decreased.

6 Conclusion

We presented a comparative study between baseline and bucketing strategies in BERT-based sentiment classification. The bucketing method demonstrated improved computational efficiency while preserving model performance. Future work includes extending this approach to multi-class tasks and multilingual datasets.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[3] Paulius Micikevicius, Sharan Narang, Jonah Alben, Greg Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2018.

[4] Joongbo Park, Jihyeok Kim, et al. Klue: Korean language understanding evaluation. *arXiv preprint arXiv:2105.09680*, 2021.

[5] Chi Sun, Xipeng Qiu, and Xuanjing Huang. Fine-tuning bert for sentiment analysis. *arXiv preprint arXiv:1905.05583*, 2019.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.