Effect of Padding Length Setting during Fine-tuning

KimYoungSook@aiffel-online13-rs

June 4, 2025

Abstract

When training a pretrained model on a new dataset for Korean natural language processing, the default tokenizer's max_length may be excessively large, making it inefficient to use as is. Analyzing the data length distribution to set an appropriate max_length is effective in terms of both performance and computation time. Using dynamic padding alone is less optimal; combining it with a properly set max_length can reduce computation time efficiently while minimizing the impact on performance.

1 Introduction

This study aims to investigate, through comparative experiments, how much Bucketing and Dynamic Padding can improve computational efficiency when fine-tuning a pretrained BERT (Bidirectional Encoder Representations from Transformers)-based language model for Korean Natural Language Processing (NLP), and how they affect model performance. Additionally, it seeks to determine an efficient max token setting by analyzing the data distribution.

2 Test Environment

2.1 Pretrained Model and Dataset Used for Evaluation

For Korean Natural Language Processing (NLP), the pretrained model used in this study is 'klue/bert-base'. The 'klue/bert-base' model has approximately 110 million parameters and consists of a 12-layer BERT architecture. It can be applied to various Korean NLP tasks such as sentence classification, named entity recognition, question answering, and semantic similarity evaluation.

The model utilizes the WordPiece tokenization method and has a vocabulary of approximately 32,000 tokens built from a Korean language corpus.

The NSMC (Naver Sentiment Movie Corpus) is a publicly available dataset for sentiment analysis in Korean. It consists of movie review sentences collected from Naver, each labeled with a binary sentiment tag (positive/negative). It is available on Hugging Face under the name 'e9t/nsmc', and contains 150,000 training samples and 50,000 test samples.

2.2 Evaluation Metrics

The evaluation metrics used are classification metrics: Accuracy and F1-Score. Accuracy measures how often the predicted data matches the actual data. Recall represents the proportion of true positive predictions among all actual positive cases. Precision indicates the proportion of true positive cases among all instances predicted as positive. Recall and precision have an inverse relationship—when recall increases, precision tends to decrease, and vice versa. Therefore, the F1-Score provides a balance between precision and recall, yielding a higher value when both metrics are well-balanced without favoring either side.

2.3 Token Length Distribution Analysis

When the 'klue/bert-base' tokenizer is used to tokenize the NSMC dataset, it outputs

$$'attention\_mask', 'input\_ids'$$

with a length of up to the tokenizer's model_max_length, which is 512 tokens.

However, the NSMC data has an average token length of 17.8 and a maximum of 117, meaning that using the full 512-token length leads to unnecessary memory usage and computational overhead.

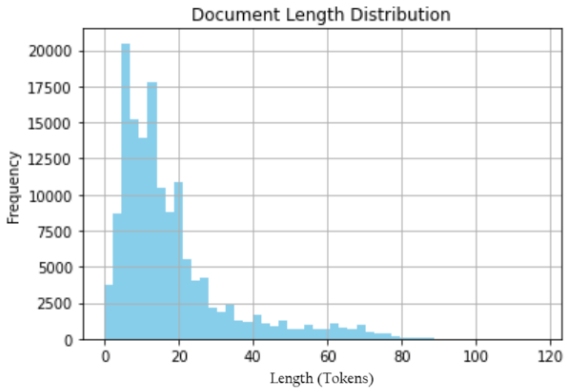

Figure 1: Token Length Distribution of NMSC Datasets

Therefore, using an appropriately shorter length instead of the tokenizer's model_max_length in 'klue/bert-base' is computationally more efficient.

In this study, we experimentally investigate whether setting an appropriate fixed length or using dynamic padding with a less appropriate length yields better results in terms of performance and computation time.

For the experiment, we set the appropriate length to 64 tokens, which covers 97.5% of all samples. We compare the performance and computation time between static padding and dynamic padding at 64 tokens. As a comparison group, we also evaluate static padding at 40 tokens, which covers 91% of all samples, to observe its impact on both performance and computation time.

3 Trade-offs of Using Dynamic Padding

Through the token length distribution, it is evident that the default max_length of 512 in the 'klue/bert-base' tokenizer is excessively large for the NSMC dataset. In this study, we conduct a comparative experiment between static padding and dynamic padding, using a maximum token length of 64, which covers 97.5% of all samples.

For reference, the padding direction is set to 'right', and dynamic padding is implemented using a data_collator, with bucketing applied via group_by_length. We used 120,000 samples for training, 30,000 for validation, and 50,000 for evaluation.

3.1 Performance Comparison

As a result of training and evaluating on the NSMC dataset for 3 epochs, applying static padding to data truncated to 64 tokens yielded better accuracy and F1-Score.

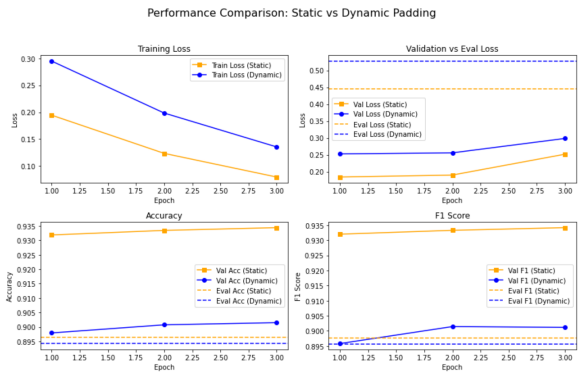

Figure 2: Performance Comparison : Static vs Dynamic Padding

Validation accuracy and F1score were approximately 3Based on these experimental results, it can be inferred that using static padding with an appropriate length is advantageous in terms of performance compared to using dynamic padding.

3.2 Comparison of Computation Time

For training runtime, static padding took 3315 seconds, while dynamic padding took 2711 seconds, resulting in a reduction of 604 seconds. This corresponds to a savings of approximately 201 seconds per epoch, which is about 18Since the NSMC dataset contains 150,000 training samples, applying dynamic padding to larger datasets would be a very effective option for reducing training time.

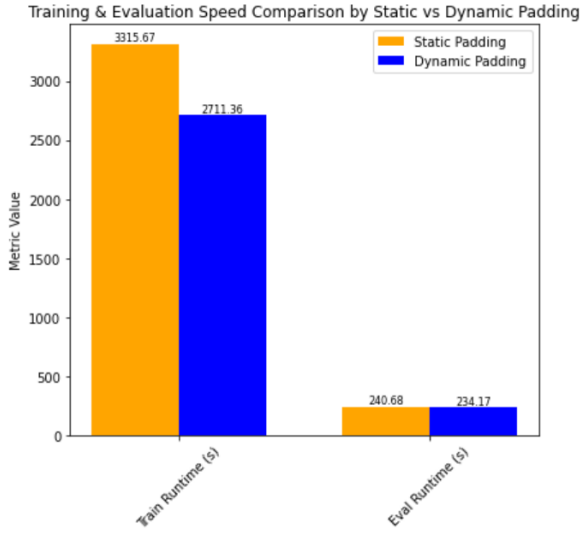

Figure 3: Computation Time Comparison : Static vs Dynamic Padding

4 Static Padding Token Length Setting

In the previous experiment, static padding was tested using a token length of 64, which covers 97.5% of all samples. But is 64 tokens the optimal length for static padding? To find out, we conduct a comparative experiment using a token length of 40, which covers 91% of all samples.

4.1 Performance Comparison According to Max_length Setting

When tokenizing with a max_length of 40, only 40 tokens are generated, and padding is added to the right for any tokens shorter than this length. Comparing the performance and training time of the

'klue/bert-base' model trained with a max_length of 40 versus 64, the model trained with a max_length of 64 showed better accuracy and F1-Score performance.

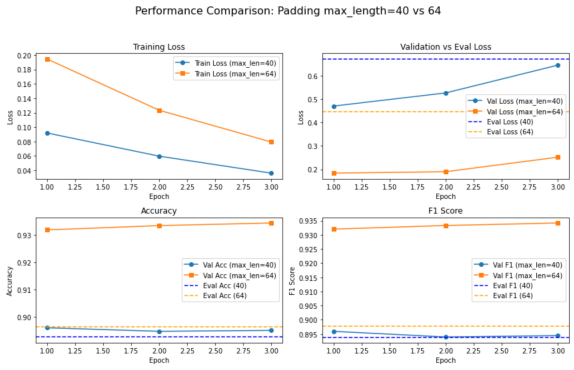

Figure 4: . Performance Comparison : Static Padding size 40 vs 64

These results indicate that including 97.5% of the data leads to better training outcomes than including only 91%, highlighting the importance of setting an appropriate padding max_length according to the data distribution.

4.2 Comparison of Computation Time According to Max_length Setting

The time required for training and evaluation was proportional to the token length. Shorter token lengths resulted in less computation time.

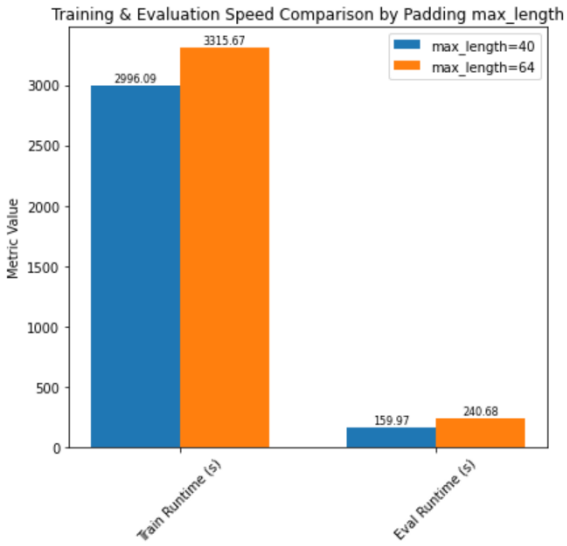

Figure 5: Computation Time Comparison : Static Padding size 40 vs 60

5 Conclusions

In this experiment, analyzing the data distribution showed that the best performance was achieved with a max token length of 64, which covers 97.5% of all samples. Using dynamic padding resulted in approximately an 18% decrease in performance. Based on these results, it is more effective to set an appropriate max_length by analyzing the dataset's length distribution rather than using the pretrained model's default max_length for training. While dynamic padding can be used to reduce training time, it should be noted that this may come with some loss in performance.

References

"Fine-tuning BERT-based NLP Models for Sentiment Analysis of Korean Reviews: Optimizing the sequence length" @articlehwang2024fine, title=Fine-tuning BERT-based NLP Models for Sentiment Analysis of Korean Reviews: Optimizing the sequence length, author=Hwang, Sunga and Park, Seyeon and Jang, Beakcheol, journal=Journal of Internet Computing and Services, volume=25, number=4, pages=47–56, year=2024, publisher=Korean Society for Internet Information