Enhancing Textual Entailment Classification with Pooling Variants on BERT

Berkay Altıntaş
Department of Electrical and Electronics Engineering
Bilkent University, Ankara, Turkey
berkay.altintas@ug.bilkent.edu.tr

***Abstract*—This study investigates Birectional Encoder Representations from Transformers (BERT) based models for sentence pair classification on the Recognizing Textual Entailment (RTE) task from the GLUE benchmark. We begin by fine-tuning BERT using the conventional [CLS] token representation, a common approach for classification tasks. Following this baseline model, two alternative strategies for sentence-level embedding, max pooling and attention-based pooling mechanism, are implemented. Different hyperparameters such as learning rate, input length, and dropout probability are tuned to enhance model performance. These findings show that the [CLS] based pooling achieves the best results, while max pooling and attention-based pooling also perform competitively even though they do not overperformed the baseline model.**

***Index Terms*—BERT, RTE, Textual Entailment, Text Classification, GLUE, Attention Pooling, Max Pooling, Transformer Models, Natural Language Processing**

I. Introduction

In this work, Bidirectional Encoder Representations from Transformers (BERT) [1] and two alternative representation strategies are explored for a text classification task. The objective is to fine-tune a pre-trained BERT model on the Recognizing Textual Entailment (RTE) task, a sentence pair classification problem that is part of the GLUE benchmark. Specifically, the model must determine whether a given hypothesis logically follows from its associated premise, making it a binary classification problem.

In the first part of this study, the standard approach is followed by using the [CLS] token as the representation of the entire input sequence. This token embedding is passed to a classifier for prediction. We utilize the `BertForSequenceClassification` model from the Hugging Face library for implementation [2]. Several hyperparameters—such as learning rate, number of training epochs, maximum input length, and dropout rate—are tuned to optimize model performance. The best-performing model is uploaded to the Hugging Face Model Hub for reproducibility.

In the second part, we introduce two alternatives to the [CLS] token for sentence representation: Max Pooling and an Attention Pooling [3] applied over BERT's token-level outputs. These methods aim to capture more informative features by aggregating contextual embeddings. The effects of the same hyperparameters are also investigated for these alternative approaches.

II. Architectures

A. Bidirectional Encoder Representations from Transformers (BERT)

BERT is built upon the encoder component of the Transformer architecture [4]. As illustrated in Figure 1, it begins by converting input tokens into dense vector embeddings and adding them positional encodings to provide order information. Then, these representations are processed through a stack of identical transformer encoder layers, each comprising multi-head self-attention mechanisms, feedforward networks, residual connections, and layer normalization.

Multi-Head Self-Attention Mechanism: This mechanism allows each token to attend to every other token in the sequence, including itself. It does so by projecting the embeddings into three vectors: queries (Q), keys (K), and values (V). These are used to compute a weighted sum of all values based on the similarity between queries and keys. Using multiple attention heads allows the model to capture different types of relationships in parallel.

Feedforward Neural Network: After self-attention, each token representation is passed through a position-wise feedforward network. This network typically includes two linear layers with a non-linear activation (like GELU or ReLU) in between, allowing for additional transformation of the token-level information.

Residual Connections and Layer Normalization: To stabilize training and help gradient flow through the deep network, residual connections are used around both the self-attention and the feedforward sublayers. After each residual connection, layer normalization is applied. This setup helps the model train efficiently and converge faster.

Output Representations: After passing through all layers, each token has a contextualized embedding that reflects its meaning based on surrounding words. For classification tasks, the special [CLS] token which is added to the beginning of the input is often used as the summary representation of the entire sequence.

One of BERT's key strengths lies in its ability to model bidirectional context. Unlike the original Transformer which includes both encoder and decoder components, BERT relies solely on the encoder and captures relationships in both forward and backward directions within the input sequence. This deep contextual understanding enables BERT to perform

effectively on a variety of NLP tasks, including sentence pair classification.

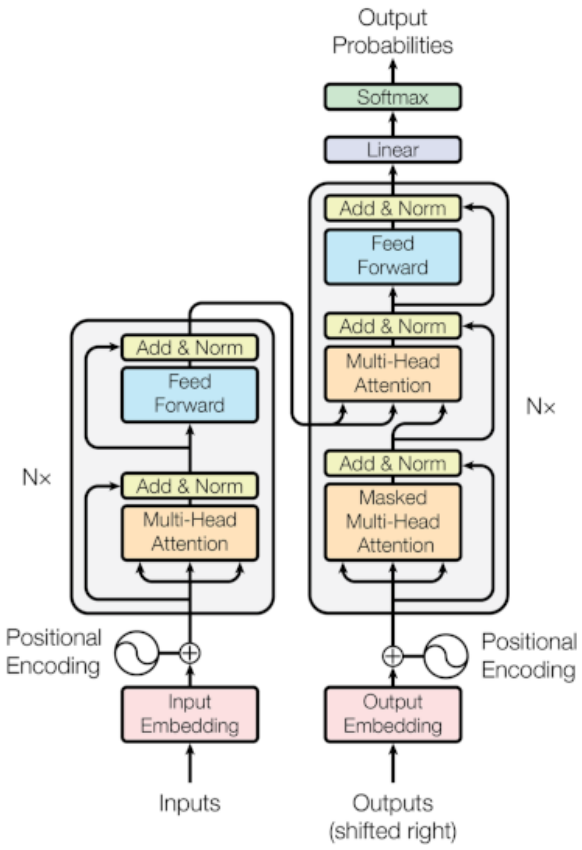

Fig. 1: BERT model architecture

B. Different Pooling Strategies for BERT Embeddings

1) Max Pooling: Max pooling is a strategy used to produce a fixed-size sentence representation from BERT's token embeddings by selecting the maximum value across all tokens in each embedding dimension. This method is useful to select the most important features from the input sequence.

Let $H \in \mathbb{R}^{L\times d}$ be the matrix of hidden states (token embeddings) produced by BERT, where L is the sequence length and d is the hidden size. Let $A \in \{0,1\}^L$ be the attention mask, where $A_i = 1$ indicates a valid token and $A_i = 0$ represents a padding token.

To prevent padded tokens from affecting the max operation, we first modify the hidden state matrix by adding a large negative value (e.g., -10^9) to the padded positions. This is done as follows:

$$H' = H + (1 - A) \cdot (-10^9)$$

Here, the mask $(1 - A)$ is broadcasted along the hidden dimension d, so that each padded token's embedding is effectively suppressed.

Next, we apply the max operation across the sequence dimension taking the maximum value in each column:

$$z = \max(H', \text{dim} = 1)$$

This yields a single vector $z \in \mathbb{R}^d$, where the j-th component of z is the maximum value among all valid tokens' j-th embedding components:

$$z_j = \max_{i:A_i=1} (H'_{i,j})$$

After obtaining this pooled representation, a dropout layer is applied to improve generalization:

$$z' = \text{Dropout}(z)$$

Then, the transformed vector is passed through a fully connected linear layer for classification:

$$o = Wz' + b$$

where $W \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^k$ are the weights and biases of the classification layer, and k is the number of output classes (typically 2 for binary classification).

Finally, the cross-entropy loss is used to train the model:

$$\mathcal{L} = -\log\left(\frac{e^{o_y}}{\sum_{j=1}^{k} e^{o_j}}\right)$$

where o_y is the logit corresponding to the correct class label y.

This approach allows the model to extract the most important information from the entire sequence while offering an alternative to using the `[CLS]` token for sentence-level representation.

2) Attention-Based Pooling: Attention-based pooling is a method used to compute a sentence-level representation by learning which tokens in the input are more important for the task. Instead of selecting the maximum value like in max pooling attention pooling computes a *weighted sum* of token embeddings, where the weights are learned through training.

Let $H \in \mathbb{R}^{L\times d}$ be the matrix of hidden token embeddings from BERT, where L is the sequence length and d is the hidden dimension. The goal is to produce a single embedding vector that captures the most informative aspects of the sequence.

First, an attention score e_i is computed for each token h_i by projecting it with a learnable parameter vector:

$$e_i = W_a h_i$$

where $W_a \in \mathbb{R}^{1\times d}$ is a trainable attention weight.

To prevent padded tokens from affecting the result, their attention scores are masked by assigning them a large negative value (e.g., $-\infty$), effectively removing them during softmax normalization.

The attention weights are computed by applying the softmax function to the scores:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{L} \exp(e_j)}$$

These weights are used to calculate a weighted sum of the token embeddings:

$$z = \sum_{i=1}^{L} \alpha_i h_i$$

The resulting vector $z \in \mathbb{R}^d$ is the final pooled representation of the sentence. As in max pooling, this vector is passed through a dropout layer and a linear classifier:

$$z' = \text{Dropout}(z) \quad \text{and} \quad o = Wz' + b$$

where $W \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$ are the weight and bias parameters of the classification layer, and k is the number of output classes.

Finally, cross-entropy loss is used to optimize the model during training:

$$\mathcal{L} = -\log\left(\frac{e^{o_y}}{\sum_{j=1}^{k} e^{o_j}}\right)$$

This method allows the model to focus more on informative tokens and less on irrelevant ones, potentially improving classification performance by producing more task-specific sentence representations.

III. Results

A. Part 1: Using [CLS] Token

The model was trained 24 times using different hyperparameters as defined in Table I.

TABLE I: Hyperparameter Search Space Used in Training

Hyperparameter	Range / Choices
`learning_rate`	1×10^{-6} to 5×10^{-5}
`num_train_epochs`	{2, 3, 4}
`per_device_train_batch_size`	{16, 32}
`max_length`	{128, 256, 512}
`classifier_dropout`	0.1 to 0.5

Then, the best accuracy (0.6751) is obtained with the following setup:

- **Learning rate:** $3.7693294228753754 \times 10^{-5}$
- **Epochs:** 4
- **Batch size:** 16
- **Max length:** 512
- **Dropout rate:** 0.10166181013063974

Figure 2 presents the validation loss curve of the best-performing run. As shown in the figure, the loss begins to increase after a certain point, which may indicate the onset of overfitting.

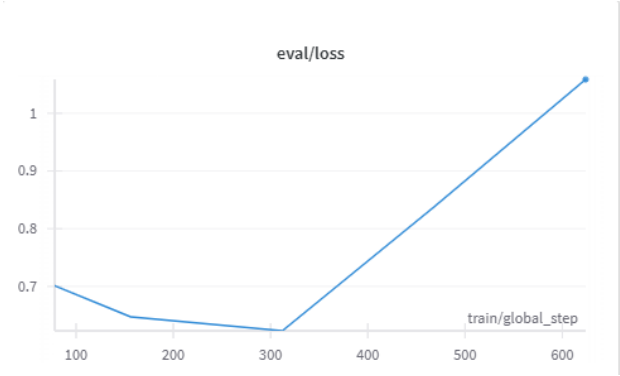

Fig. 2: Validation loss curve of the best-performing run

Figure 3 illustrates the validation accuracy curve of the best-performing run. As shown in the figure, accuracy increases rapidly during the initial training steps due to fine-tuning, and then gradually stabilizes in the later stages.

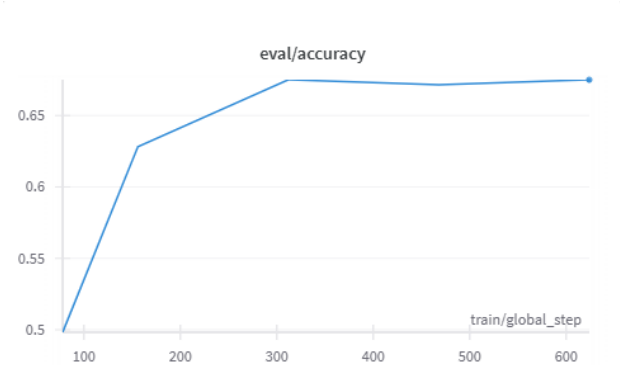

Fig. 3: Validation accuracy curve of the best-performing run

As a result, the following loss and accuracy values were obtained, as shown in Table II.

TABLE II: Final Evaluation Metrics of the Best Run

Metric	Value
Loss	1.0588
Accuracy	0.6751

The best model can be accessed from the following Hugging Face entry: hyperref

The best model can be accessed from the following Hugging Face link: `https://huggingface.co/berkayaltntas/bert-base-uncased-finetuned-rte-run_3`.

B. Part 2: Using Max Pooling

In this part, the model was trained 24 times using different hyperparameters, as defined in Table III.

TABLE III: Hyperparameter Search Space Used in Training

Hyperparameter	Range / Choices
learning_rate	{1e-6, 3e-5}
n_epochs	{2, 3, 4}
max_length	{16, 32, 64}
batch_size	{16, 32, 64}

Then, the best accuracy (0.6138) is obtained with the following setup:

- **Learning rate:** 2.4×10^{-5}
- **Epochs:** 4
- **Batch size:** 32
- **Max length:** 64

Moreover, the final validation accuracy and the final validation loss can be seen in Table IV.

TABLE IV: Final Training and Validation Performance

Metric	Value
Validation Loss	0.52
Validation Accuracy (%)	61.38

C. Part 3: Using Attention Pooling

In this part, the model was trained 24 times using different hyperparameters, as defined in Table V.

TABLE V: Hyperparameter Set Used in Training

Hyperparameter	Range / Choices
lr	{1e-6, 3e-5}
n_epochs	{2, 3, 4}
max_length	{16, 32, 64}
batch_size	{16, 32, 64}

Then, the best accuracy (0.5646) is obtained with the following setup:

- **Learning rate:** 1×10^{-6}
- **Epochs:** 4
- **Batch size:** 64
- **Max length:** 16

Moreover, the final validation accuracy and the final validation loss can be seen in Table VI.

TABLE VI: Final Training and Validation Performance

Metric	Value
Validation Loss	0.69
Validation Accuracy (%)	56.46

IV. Discussion and Conclusion

The goal of this study was to assess the impact of different pooling strategies on BERT's performance for the Recognizing Textual Entailment (RTE) task. While BERT is typically fine-tuned using the [CLS] token as a summary representation, this work also explored two alternative methods: max pooling and attention-based pooling, utilizing information differently from the sequence of token embeddings. Among these, the [CLS]-based approach achieved the highest accuracy, 0.6751, demonstrating its reliability on RTE. Despite their theoretical strengths, neither max pooling nor attention pooling led to increase in performance. This may indicate that, for relatively small datasets like RTE, simpler methods may generalize better than custom arrangements that increase model complexity.

These findings suggest that BERT's default architecture and training objectives already encourage it to encode relevant sentence-level information in the [CLS] token. Additional pooling mechanisms, although potentially more expressive, may not have enough data to learn effectively without risking overfitting. Therefore, while alternative pooling remains an interesting area of exploration, it may require larger datasets or further optimization techniques to show meaningful improvements.

In conclusion, the [CLS]-based strategy continues to serve as a strong and robust benchmark for sentence pair classification tasks, particularly in small datasize scenarios. However, future research could improve the performance of alternative pooling strategies by introducing stronger regularization, leveraging data augmentation, or scaling up the dataset size. These steps may help to reach full potential of more sophisticated sentence representation techniques in textual entailment and other NLP tasks.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.

[2] Hugging Face, "BERT — transformers 3.0.2 documentation," Hugging Face, 2020. [Online]. Available: https://huggingface.co/transformers/v3.0.2/model_doc/bert.html#bertforsequenceclassification

[3] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *Proc. ICLR*, 2017.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.