Transformer from Scratch: Implementation and Evaluation on IMDb Movie Reviews

Jaehyun Lee
Korea University
jaehyun1030@korea.ac.kr

Abstract

This paper presents the implementation and evaluation of a Transformer-based text classification model on the IMDb movie review dataset. We construct the Transformer architecture from scratch, including positional encoding and multi-head attention mechanisms, to classify movie reviews into positive or negative sentiments. Additionally, we fine-tune a pre-trained BERT model on the same dataset to compare performance. Our results demonstrate that while the self-implemented Transformer performs reasonably well, the BERT-based model achieves significantly higher accuracy due to its deep contextual understanding. This work highlights the importance of pre-training in modern NLP models and provides insights into the internal workings of Transformer architectures.

1. Introduction

Natural Language Processing (NLP) has undergone a major paradigm shift with the introduction of Transformer-based architectures. Unlike Recurrent Neural Networks (RNNs), Transformers utilize self-attention mechanisms to capture long-range dependencies in textual data more efficiently and in parallel. This architecture forms the foundation of many state-of-the-art language models such as BERT, GPT, and T5.

In this work, we aim to implement a Transformer-based classifier from scratch to perform binary sentiment classification on the IMDb movie review dataset. The primary objective is to understand the inner mechanisms of the Transformer model, including positional encoding, multi-head attention, and encoder layers. To evaluate the effectiveness of our implementation, we also fine-tune a pre-trained BERT model on the same dataset and compare its performance with our self-built model.

Through this assignment, we highlight the performance gap between pre-trained language models and models trained from scratch. Our results emphasize the benefits of large-scale pre-training and demonstrate how Transformer components interact to process and classify natural language.

2. Implementation

We implemented a Transformer-based text classifier from scratch using PyTorch. The model includes core components such as **Positional Encoding**, **Multi-Head Attention**, and **Encoder Layers**, following the original architecture from Vaswani et al. [**?**]. The IMDb dataset was split into 15,000 training, 5,000 validation, and 5,000 test samples, as illustrated in Figure 1.

2.1. Positional Encoding

Since Transformers lack an inherent notion of token order, we implemented sinusoidal positional encoding. The positional encoding was added to the token embeddings and calculated as follows:

$$\mathrm{PE}_{\mathrm{pos},2i} = \sin\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right) \tag{1}$$

$$\mathrm{PE}_{\mathrm{pos},2i+1} = \cos\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right) \tag{2}$$

This was implemented in PyTorch using sine for even indices and cosine for odd indices.

2.2. Multi-Head Attention

To enable the model to attend to information from different representation subspaces, we used multi-head attention. For each head, we projected the input into query, key, and value using separate linear layers:

```
self.w_q = nn.Linear(d_model, d_model)
self.w_k = nn.Linear(d_model, d_model)
self.w_v = nn.Linear(d_model, d_model)
```

The scaled dot-product attention was computed as:

$$\text{scores} = \frac{QK^\top}{\sqrt{d_k}} \tag{3}$$

$$\text{attn\_weights} = \text{softmax}(\text{scores}) \tag{4}$$

$$\text{output} = \text{dropout}(\text{attn\_weights}) \cdot V \tag{5}$$

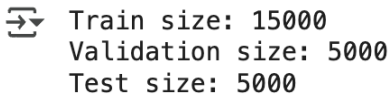

Figure 1. Dataset split: 15,000 for training, 5,000 for validation, and 5,000 for testing.

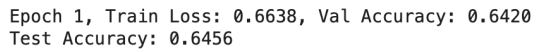

Figure 2. Training loss and validation/test accuracy of our Transformer model after one epoch.

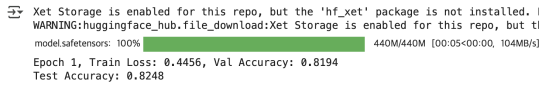

Figure 3. Training loss and validation/test accuracy of fine-tuned BERT model after one epoch.

2.3. Transformer Encoder Layer

The structure follows the standard Transformer encoder design, where the input first passes through a multi-head self-attention layer followed by a residual connection, dropout, and layer normalization. This is then followed by a feed-forward network (FFN), again combined with a residual connection, dropout, and normalization.

3. Discussion

We compare the performance of our self-implemented Transformer with a pre-trained BERT model on the IMDb dataset. The training and evaluation were conducted for a single epoch in both settings. The results reveal a significant performance gap between the two approaches.

As shown in Figure 2, our Transformer model achieved a validation accuracy of 64.20% and a test accuracy of 64.56% after one epoch. The corresponding training loss was 0.6638. While this demonstrates that the model learns meaningful representations, the performance is limited due to the lack of pretraining and relatively small model capacity.

In contrast, the fine-tuned BERT model achieved a validation accuracy of 81.94% and a test accuracy of 82.48%, with a lower training loss of 0.4456 (Figure 3). This illustrates the power of transfer learning using pre-trained language models, which leverage large-scale unsupervised learning to capture rich contextual information before task-specific fine-tuning.

Overall, this comparison highlights the importance of pretraining in modern NLP tasks. While building a Transformer from scratch is a valuable learning experience, pre-trained models like BERT offer a substantial advantage in terms of accuracy and convergence speed, even with minimal fine-tuning.

References