COSC420: Assignment 2 Report

Peyton Mou (2212984)

May 18, 2025

1 Introduction

This report presents the implementation of a sentiment classification model based on BERT (Bidirectional Encoder Representations from Transformers), conducted on IMDB movie reviews and the Stanford Sentiment Treebank datasets. There are two tasks: first task is pretraining and finetuning the model to match benchmark development accuracies; the second is exploring multiple approaches aiming to improve model performance.

The model successfully achieves baseline development accuracies on both pretrain and finetune options. Regarding the enhancement experiment, NLP tools, MSE loss, additional CLS layers with a learning rate scheduler, and pretraining on external datasets followed by finetuning are conducted, with more details of methodology and results explained in this report.

2 Bert-based Model for Sentiment Classification

2.1 Methodology and Implementation

The BERT-based sentiment classifier model was implemented, and can successfully pass unit tests after adding several missing components (multi-head self-attention, layer normalization, embedding function, BertLayer feed-forward, and AdamW optimizer), as explained below:

- **BertSelfAttention Class:** Implements the scaled dot-product attention. For each token, it computes attention scores via the dot product of query and key, scaled by the square root of the hidden dimension and passed through a softmax. Multiple heads allow capturing diverse semantic relations.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

- **BertLayer Class:** Encapsulates a transformer block consisting of multi-head self-attention and a position-wise feed-forward network. Each sublayer uses residual connections and layer normalization, enabling the model to refine token embeddings hierarchically.

- **BertModel Class:** Stacks several `BertLayer` blocks after an embedding layer, which sums token, positional, and type embeddings. Type embeddings are all zeros due to single-sentence inputs. The embedding layer captures both semantics and positional meaning.
- **AdamW Optimizer:** Enhances Adam by decoupling weight decay from gradient updates. Unlike traditional Adam, it applies weight decay directly to parameter values, improving training stability and regularization.

2.2 Experiments and Results

The model was evaluated on two datasets: IMDB movie reviews and Stanford Sentiment Treebank (SST). The encoder-only architecture was configured with 12 transformer layers, a hidden size of 768, and 12 attention heads. Experiments were conducted using seed 1234, exploring a range of hyperparameters (learning rate, dropout, batch size) and training epochs under both finetune and pretrain options.

As in Table 1 for IMDB, the finetune model achieved dev accuracy 0.971 and test accuracy 0.516. In pretrain option, an initial configuration with a learning rate of 1e-5 and batch size of 8 yielded only 0.751 dev accuracy after 50 epochs, falling short of the baseline. To improve generalization, the learning rate increased to 1e-4 and batch size to 16, reaching a higher dev accuracy peaking at 0.882 by epoch 50 and plateauing through epoch 70, indicating a larger learning rate and batch size are beneficial for pretraining mode. Repeating the same hyperparameters but reducing to 10 epochs reached a comparable dev accuracy of 0.796 close to baseline.

Table 1: BERT Model Performance on IMDB Dataset

Option	**LR**	**BS**	**Dropout**	**Epochs**	**Train Acc**	**Dev Acc**	**Test Acc**
Finetune	1e-5	8	0.3	10	1.000	0.971	0.516
Pretrain	1e-5	8	0.3	50	0.736	0.751	0.414
Pretrain	1e-4	16	0.1	10	0.775	0.780	/
Pretrain	1e-4	16	0.1	50	0.882	0.882	/
Pretrain	1e-4	16	0.1	70	0.898	0.882	0.475
Pretrain	1e-4	16	0.1	10	0.778	0.796	0.504

For SST, as in Table 2, the model reached the baseline dev accuracy of 0.401 in pretrain setup with just 3 epochs, using a lower dropout rate 0.1, a higher learning rate 8e-5, and a batch size 16. Under finetune setting, the model achieved dev accuracy 0.524 after 10 epochs using default hyperparameters, also meeting the baseline. These results indicate that the implemented model is well-structured and can capture linguistic patterns through training.

Table 2: BERT Model Performance on Stanford Sentiment Treebank

Option	**LR**	**BS**	**Dropout**	**Epochs**	**Train Acc**	**Dev Acc**	**Test Acc**
Finetune	1e-5	8	0.3	10	0.985	0.524	0.519
Pretrain	8e-5	16	0.1	3	0.414	0.401	0.405

3 Improving Model's Performance

Several approached were explored to enhance model performance on sentiment classification, however most struggled to surpass the baseline. While some methods realized marginal increase only for test accuracy (typically 1-2%), overfitting is still noticeable.

3.1 NLP Tools

To improve semantic and syntactic understanding, pre-existing NLP tools like Part-of-Speech (POS) tagging, Dependency Parsing (DEP), and WordNet (WN) were applied. However, as shown in Table 3, combining POS, DEP and WN features resulted in weaker performance: the dev accuracy for SST dropped to 0.304, and for IMDB reached only 0.661, significantly below the finetune baseline. Furthermore, an ablation study on SST revealed that POS alone performed the worst dev accuracy 0.312, while DEP and WN individually achieved 0.515, matching the baseline but no real gain. For IMDB, the ablation results showed that removing POS slightly improved the performance, the best configuration with only DEP or WN peaked at 0.963 dev acc and 0.529 test acc, gaining a limited increase.

The lower performance is attributed to two reasons: the first is adding external NLP features disrupt the learned representations of pretrained embeddings, leading to noise rather than enhancement; the second is simply incorporating the raw tags and extra information without checking how well they match to the model could weaken the original useful patterns.

About code modification, `tokenizer.py` (line 2550) added function to extract linguistic features for each token, `cls_nlp.py` added functions to build linguistic vocabularies and encode features into integer IDs, `BertDataset` class modified to align these features with tokens, `bert.py` extended with embedding layers for POS, DEP and WN, `config.py` added additional inputs arguments.

Table 3: Model Performance with NLP Tools

Data	Approach	LR	BS	Dropout	Epochs	Train Acc	Dev Acc	Test Acc
IMDB	pos/dep/wn	1e-5	16	0.3	15	0.699	0.661	0.619
	pos	1e-5	16	0.3	10	0.665	0.661	0.619
	dep	1e-5	16	0.3	10	1	0.963	0.529
	wn	1e-5	16	0.3	10	1	0.963	0.529
SST	pos/dep/wn	1e-5	16	0.3	15	0.377	0.304	0.292
	pos	1e-5	64	0.3	10	0.339	0.312	0.296
	dep	1e-5	16	0.3	10	0.982	0.515	0.534
	wn	1e-5	16	0.3	10	0.982	0.515	0.534

3.2 MSE Loss and Enhanced CLS Layers

Next, two alternative methods were explored: one is replacing standard classification loss with MSE (mean squared error) loss, aiming to facilitate smoother prediction confidence, while as in Table 4 the updated model resulted dev acc 0.959 and 0.507 respectively for IMDB and SST, which is still within the baseline range. This code is updated in `cls_mse.py`.

In addition, the second method added 3 linear layers and 2 layer normalization for classification head, as code updated in `cls_layer.py`. Furthermore, a cosine annealing warm restarts scheduler was introduced to dynamically adjust the learning rate throughout training. In Table 4, *Warm* refers to the warm-up steps before decay begins, *LR* is the initial base learning rate, and *A Drop* is the attention dropout probability. This modified model achieved test accuracy 0.532 on SST, slightly higher than baseline.

Table 4: Model Performance with MSE Loss or Adding CLS Layer with LR Scheduler

Data	**Method**	**LR**	**Warm**	**BS**	**Dropout**	**A Drop**	**Epochs**	**Dev Acc**	**Test Acc**
IMDB	mse	1e-5	-	16	0.3	-	30	0.959	0.516
	cls/lr	1e-5	300	8	0.45	0.2	15	0.963	0.512
	cls/lr	1e-5	300	8	0.4	0.2	15	0.967	0.506
SST	mse	1e-5	-	64	0.3	-	10	0.507	0.520
	cls/lr	3e-5	600	8	0.4	0.2	15	0.506	0.518
	cls/lr	1e-5	300	8	0.45	0.1	10	0.511	0.532

3.3 External Dataset Pretraining Followed by Finetuning

Finally, I explored a transfer learning approach that involves pretraining on larger external datasets, followed by finetuning on target dataset. Theoretically, the model can learn generalizable sentiment features, and also be adaptive to task-specific nuances through transfer learning.

For IMDB, the model was pretrained on the 2-label SST2 dataset (Stanford Sentiment Treebank 2), after which the learned weights were saved. These weights were then loaded to finetune the model on IMDB. As shown in Table 5, the final dev accuracy reached 0.963, comparable to baseline but no significant improvement.

For SST, the model was pretrained on the 5-label Yelp dataset. Due to computing resource and time constraints, only 30% of the Yelp dataset (195,000 examples) used for 1 epoch, and 10% (65,000 examples) for 3 epochs. Although loss is decreasing, no increase of dev accuracy was observed, limiting further pretraining. The pretrain saved model was loaded in and continue finetuning on SST for 10 epochs, and finally reached dev accuracy 0.505 and test accuracy 0.515. The lack of a significant performance improvement is likely due to the pretraining phase being conducted on a small subset of Yelp, resulted in underfitting and preventing the model from effectively learning meaningful sentiment representations.

About code modification, `create_data` in `cls_yelp.py` was added loading external datasets (Yelp or SST-2 from the HuggingFace datasets library), and convert them to a format consistent with our SST/IMDB data pipeline; `train` was modified to handle separate datasets for train/dev and test. In `cls_transfer.py`, one line was added to load a pretrained model checkpoint, enabling continuous finetuning from previously saved weights.

Table 5: Model Performance of External Dataset Pretraining Followed by Finetuning

Data	Train Data Size	Option	LR	BS	Dropout	Epochs	Train Acc	Dev Acc	Test Acc
SST-2	67,349	Pretrain	1e-5	8	0.25	1	0.728	0.567	0.465
IMDB	1,707	Finetune	1e-5	8	0.25	10	1.00	0.963	0.506
Yelp-10%	195,000	Pretrain	1e-5	8	0.25	1	0.466	0.219	0.247
Yelp-30%	65,000	Pretrain	1e-5	8	0.25	3	0.518	0.206	0.348
SST	8,544	Finetune	1e-5	8	0.25	10	0.985	0.505	0.515

4 Conclusion

This project implemented and evaluated a BERT-based sentiment classification model on the IMDB and SST datasets. The model achieved strong baseline development accuracies of 0.971 for IMDB and 0.524 for SST using finetuning option.

Several enhancement strategies were explored, including NLP tools, MSE loss, integrating additional classification layers with a learning rate scheduler, and transfer learning via pretraining on external larger datasets. While these efforts resulted in a slight improvement, most notably an SST test accuracy of 0.534, indicating limited benefit from small-scale modifications.

For future work, deeper improvements could come from systematic hyperparameter tuning, scaling up pretraining with larger and more diverse sentiment corpora, leveraging more computing resources, and experimenting with more advanced and robust model architectures.