Exploring LoRA Techniques for Sentiment Classification of IMDb Reviews Using DistilBERT

Author - Prabhleen Kaur
prabhleenkaur@ucsd.edu

1 Introduction

This project employs parameter-efficient fine-tuning, LoRA (low-rank adaptation), on a pre-trained DistilBERT to compare with a fully-trained DistilBERT and Logistic Regression model for the purpose of sentiment classification of IMDb movie reviews.

- The task involves sentiment classification on the IMDB movie reviews dataset, which contains binary labels for reviews as positive or negative.
- Logistic Regression with TF-IDF features and DistilBERT model are selected for baselines for sentiment analysis
- LoRA (Low-Rank Adaptation) is applied to a pre-trained DistilBERT model, which reduces the number of trainable parameters (to nearly 1%) during fine-tuning, making the process more computationally efficient, and time efficient.
- AdaLoRA, an extension of LoRA, is also explored, allowing for adaptive learning of low-rank updates during training to further improve performance.
- A detailed analysis of common misclassification is performed, focusing on errors related to negations and contrast words that tend to confuse all models in this project.
- The performance of LoRA-enhanced DistilBERT and AdaLoRA is compared against the baseline logistic regression model, DistilBERT, analyzing the impact of low-rank adaptation on accuracy and efficiency.

2 Related Work

The task of sentiment analysis, particularly the binary classification of movie reviews, has been a focus of extensive research. This project explores the performance of DistilBERT, its Low-Rank Adaptation (LoRA) variants, and traditional machine learning techniques. The review highlights the evolution of NLP methodologies and their impact on sentiment classification tasks.

The introduction of the Transformer architecture by Vaswani et al. (Vaswani et al., 2017) revolutionized NLP by leveraging attention mechanisms to effectively model dependencies within language data. This innovation paved the way for BERT, a pre-trained bidirectional transformer developed by Devlin et al. (Devlin et al., 2018), which achieved state-of-the-art results on a variety of NLP tasks. However, BERT's computational demands led to the development of more efficient models like DistilBERT, proposed by Sanh et al. (Sanh et al., 2019). DistilBERT uses knowledge distillation to reduce BERT's size by 60% while retaining 97% of its performance, making it an ideal choice for scenarios requiring faster inference and lower resource consumption.

Parameter-efficient fine-tuning methods have further enhanced the adaptability of models like DistilBERT. Low-Rank Adaptation (LoRA), introduced by Hu et al. (Hu et al., 2021), optimizes fine-tuning by freezing pre-trained weights and learning task-specific low-rank updates, significantly reducing computational overhead. AdaLoRA, an extension proposed by Zhang et al. (Zhang et al., 2023), dynamically adjusts update ranks during training, improving efficiency and performance. These innovations have made transformer models even more accessible for tasks such as sentiment analysis.

Traditional machine learning approaches, de-

spite being overshadowed by modern transformer models, remain valuable for benchmarking. Logistic regression, as demonstrated by Pang et al. (Pang et al., 2002), effectively classifies sentiment using features like TF-IDF and bag-of-words. While these methods lack the sophistication of transformer-based models, they offer simplicity and efficiency, serving as reliable baselines for comparison.

3 Dataset

The IMDb dataset used in this study is sourced from the Hugging Face `datasets` library (Datasets, 2020). The IMDb dataset consists of 50,000 movie reviews labeled with their sentiment, 0 or 1, signifying negative and positive sentiment respectively. Two reasons for selecting this data were its large size and balanced sentiment labels. It is widely used for sentiment classification tasks due to the good quality of data. Additionally, being available through the Hugging Face datasets library ensures easy access and integration with various different models.

For the faster computation time, for this project, only half the dataset was used, which is 25,000 movie reviews. The train-validation-test split used was 70%-15%-15% throughout this project for all the models. A sample from this dataset looks like this, where label is the sentiment label and text is the movie review.

```
{
    "label": 0,
    "text": "Whoever wrote the
    screenplay for this movie..\n"
}
```

4 Baselines

What are your baselines, Additionally, explain how each one works, and list the hyperparameters you used and how you tuned them. Describe your train/validation/test split. If you have tuned any hyperparameters on your test set, expect a major point deduction.

4.1 Logistic Regression with TF-IDF

The logistic regression baseline model uses TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to transform text data into numerical features and applies logistic regression for classification. A scikit-learn `Pipeline` integrates feature extraction and model training:

TF-IDF Vectorization converts text data into a numerical matrix based on term frequency and inverse document frequency, removing common English stop words and retaining the top 5000 most frequent terms (`max_features=5000`). The vectorizer considers unigrams and bigrams (`ngram_range=(1, 2)`) for richer contextual understanding. A linear classifier predicts the probability of positive or negative sentiment using learned weights for TF-IDF features. It uses the `liblinear` solver, which is efficient for small to medium-sized datasets, and regularization to control overfitting.

Hyperparameter tuning was performed using Grid Search Cross-Validation (`GridSearchCV`) with 3-fold cross-validation on the training set. The dataset was split into 70-15-15% for train-val-test respectively. No tuning was done on the test set to prevent data leakage and ensure unbiased evaluation.

Hyperparameter	Range / Best Value
tfidf max_features	3000, 5000, **7000**
tfidf ngram_range	(1, 1), **(1, 2)**, (1, 3)
classifier C	0.1, **1**, 10
classifier penalty	l1, **l2**

Table 1: Hyperparameters Tuned for Logistic Regression Baseline with Best Values Highlighted

The optimized model was evaluated on the test set, achieving the following performance:

- **Accuracy:** 0.89
- **Precision:** 0.88
- **Recall:** 0.87
- **F1-Score:** 0.88

4.2 DistilBERT Model

This baseline model processes text by tokenizing the input using the DistilBertTokenizer, which converts the text into tokens that the model can understand. It then performs sequence classification, outputting the probability of each class (positive or negative sentiment). The model is fine-tuned on the IMDB dataset for binary sentiment classification.

Hyperparameter Tuning was performed using Optuna (Akiba et al., 2019). Optuna explores the search space by sampling hyperparameters using the Tree-structured Parzen Estimator (TPE) algorithm, evaluating each configuration based on the objective function, and iteratively refining the search to find the optimal set. 15 configurations were tested according to the table below and the best hyperparameter values were found. The same train-val-test split was performed, of 70-15-15. The validation loss was calculated for each configuration to find the best model. No test data was used here to prevent data leakage.

Hyperparam	Best Value	Search Space
learning rate	0.0003186	[5e-6, 5e-4]
batch size	64	{8, 16, 32, 64}
warmup steps	339	[0, 500]
weight decay	0.2463	[0.0, 0.3]
train epochs	7	[2, 10]

Table 2: Best Hyperparameters and Search Space for DistilBERT Sentiment Classification

The configuration specified above resulted in the best validation cross-entropy loss of 0.3955. As a comparison, for binary classification, a loss of 0.693 would indicate random guessing, as that corresponds to a model that predicts a probability of 0.5 for both classes. A loss of 0.3955 indicates the model is outperforming random guessing significantly, and is able to generalize well. Training time for DistilBERT using the best hyperparameters was about 80 minutes using the Google Colab T4 GPU. Overall accuracy for the best model was 91%.

Table 3: DistilBERT Model Performance

	Precision	Recall	F1
0	0.89	0.92	0.91
1	0.92	0.89	0.91

5 PEFT techniques: LoRA and AdaLoRA

PEFT (Parameter-Efficient Fine-Tuning) techniques, such as LoRA and AdaLoRA, provide efficient methods for adapting large pre-trained models to specific tasks without fine-tuning all model parameters. Only 1.2% of the parameters that were trained for DistilBERT were finetuned using LoRA and AdaLoRA.

Type of Parameter	Count
Trainable Parameters	813,458
Total Parameters	67,768,480

Table 4: PEFT Statistics for DistilBERT

- LoRA (Low-Rank Adaptation): A technique that reduces the number of parameters required to adapt a pre-trained model by introducing low-rank updates.
- AdaLoRA (Adaptive LoRA): An extension of LoRA that allows for adaptive learning during training, improving the model's ability to adapt to the data.

5.1 Implementation

The `train_lora_model` function utilizes the Hugging Face Transformers library to load a pre-trained DistilBERT model. It applies PEFT (Parameter Efficient Fine-Tuning) using the `prepare_model_for_kbit_training` and `get_peft_model` functions, which introduce low-rank adaptation to efficiently fine-tune the model by reducing the number of trainable parameters. The LoraConfig is used to specify the rank, dropout and alpha values. For the implementation of AdaLoRA, `AdaLoraConfig` was used, which introduces adaptive ranks for low-rank updates during training. An initial rank and a maximum target rank is provided, allowing the model to dynamically adjust the rank of low-rank matrices as training progresses. The following hyperparameters were learnt to be the best during hyperparameter tuning using Optuna. 15 configurations were tested for each of the approaches.

Hyperparam	LoRA	AdaLoRA
Rank	16	8(Initial), 16(Target)
lora_alpha	64	32
Dropout	0.0124	0.163377
Learning Rate	0.000479	9e-05
Epochs	3	5

Table 5: Hyperparameter Comparison between LoRA and AdaLoRA Models

5.2 Compute

The experiments were run on Google Colab with GPU acceleration. The training process was conducted on the following virtual machine specifications:

- **CPU**: Virtual machine CPU on Google Colab
- **GPU**: Tesla T4 (NVIDIA)
- **GPU RAM**: 15GB
- **System RAM**: 12.7GB

The models were trained using PyTorch and Hugging Face's Transformers library. The GPU resources available on Google Colab were sufficient for training the LoRA and AdaLoRA models, with no major issues encountered during the training process.

5.3 Runtime

The hyperparameter finetuning for both models took around 3 hours each, and the model for each of the approaches took around 25 minutes.

Table 6: LoRA Model Performance

Class	Precision	Recall	F1
0	0.89	0.95	0.92
1	0.94	0.89	0.92

Table 7: AdaLoRA Model Performance

Class	Precision	Recall	F1
0	0.90	0.90	0.90
1	0.90	0.90	0.90

We can see that LoRA finetuning outperforms DistilBERT full-tuning even though it learns only 1.2% of the weights and trains for less than 1/3rd of the time. LoRA performs better than Logistic Regression with TF-IDF because it leverages the power of deep learning models, such as DistilBERT, which are able to capture more complex patterns and semantic information in text. While Logistic Regression with TF-IDF relies on manually engineered features, LoRA finetunes pretrained models, enabling them to adapt more effectively to the specific task. LoRA's approach allows it to benefit from the vast amount of knowledge encoded in the pre-trained language models, improving performance on sentiment analysis tasks.

AdaLoRA on the other hand, does slightly worse than LoRA. It underperforms compared to DistilBERT, but still better than Logistic Regression. Overall, all 4 models are very close in accuracy.

One of the reasons for AdaLoRA unexpected low performance could be that more hyperparameter tuning was required. AdaLoRA introduces additional complexity with adaptive rank. As a part of this project, only 15 configurations of hyperparameters were evaluated due to computation and time constraints, which might not have been enough. If more hyperparameter configurations were tested or if the rank and learning rate were adjusted more carefully, AdaLoRA could potentially outperform LoRA in future experiments.

6 Error analysis

Two kinds of errors were of special interest in this project: ones with negations and ones with contrast words. These tended to confuse all models.

6.1 Negation

The average negation counts for correctly and incorrectly classified samples provide insights into how the models handle negations in the text. A negation was defined as presence of the words [" not ", " no ", "n't", "never", " nor ", " neither ", " none "] in this experiment. One such example

Table 8: Average Negations in Correctly and Incorrectly Classified Texts

Classification Type	Average Negations
Correctly Classified	3.31
Incorrectly Classified	3.43

review which was wrongly classified (true label:1, predicted: 0) by all the models had 6 such negations:

> "I **can't** believe people are looking for a plot in this film. This is Laural and Hardy. Lighten up already. These two were a riot. Their comic genius is as funny today as it was 70 years ago. **Not** a filthy word out of either mouth and they were able to keep audiences in stitches. Their comedy **wasn't** sophisticated by any stretch. If a whoopee cushion **can't** make you grin, there's **no** reason to watch any of the stuff these

guys did. It was a simpler time, and people laughed at stuff that was funny without a plot. I guess it takes a simple mind to enjoy this stuff, so I qualify. Two man comedy teams don't compute, We're just too sophisticated... **Aren't** we fortunate? "

This indicates that the models are making errors more often in samples that contain more negations.

6.2 Contrast Words

Contrast was defined as the presence of the following words: ["but", "although", "however", "though", "yet"].

DistilBERT and LoRA models exhibit the highest average contrast word counts in their errors, suggesting they struggle with more complex or contrasting language. While LoRA has fewer total errors than DistilBERT, their similar contrast word counts indicate comparable challenges with nuanced misclassifications. AdaLoRA, despite having the most errors, shows slightly lower contrast word counts, implying less complexity in its misclassifications.

Logistic Regression has the lowest contrast word count in errors, possibly reflecting simpler mistakes. Across all models, misclassified instances consistently have higher contrast word counts than correctly classified ones.

Table 9: Average Contrast Word Counts and Errors by Model

Model	Contrast Count	Errors
DistilBERT	3.09	349
LoRA	3.00	313
AdaLoRA	2.92	369
Logistic Regression	2.73	456

Table 10: Average Contrast Word Counts

Category	Contrast Count
Correctly Classified	2.48
Incorrectly Classified	2.90

7 Conclusion

In this project, low-rank adaptation techniques, LoRA and AdaLoRA, were applied to the DistilBERT model for sentiment analysis. LoRA demonstrated superior performance compared to DistilBERT, achieving better accuracy despite requiring fewer parameters and less training time. Surprisingly, AdaLoRA underperfomed LoRA, while still outperforming Logistic Regression in sentiment classification. A significant constraint during the project was the limited hyperparameter tuning, as only a small number of configurations were explored due to computational limitations. This may have prevented AdaLoRA from achieving its full potential. It is worthwhile to spend more effort of hyperparameter tuning as the next steps for this project.

The analysis showed that negations and contrast words were more frequently present in misclassified examples, suggesting that all models struggled with these linguistic features. LoRA's approach to fine-tuning pre-trained models proved effective by leveraging deep learning's ability to capture complex patterns. The results suggest that further exploration of hyperparameter configurations, especially for AdaLoRA, could enhance its performance.

8 Acknowledgements

I would like to thank Professor Ndapa Nakashole for teaching the course CSE256 and providing invaluable guidance throughout the semester. I also appreciate the efforts of the teaching assistant for their support and assistance in understanding the course material, as well as their timely replies of Piazza. Lastly, I am grateful to ChatGPT for helping me proofread and refine this report, and for helping me identify bugs in my code as well as help optimize it.

References

Akiba, T., Sano, S., Yanase, T., Ohta, H., and Koyama, J. (2019). Optuna: A next-generation hyperparameter optimization framework. *arXiv preprint arXiv:1907.10902*.

Datasets, H. F. (2020). Imdb dataset. Accessed: 2024-12-05.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Hu, E., Shen, Y., and Xu, Z. (2021). Lora: Low-rank adaptation of large language models. In *NeurIPS*.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*.

Zhang, H., Li, Z., and Liu, L. (2023). Adalora: Adaptive low-rank adaptation for large language models. *arXiv*.