AI Course Project Report

Natural Language Processing with Disaster Tweets

Lexy Andershock, Gian Fernandez-Aleman, Ethan Miller, Kevin Lam, David Long, and Rylee Petrole

Introduction

Our group chose Option A – Natural Language Processing with Disaster Tweets to explore NLP applications and machine learning implementations. The goal of this assignment was to fine-tune a BERT classifier model to detect whether a Twitter Tweet is announcing a disaster or not. The "Original Model" section details our first attempt at fine-tuning, with minimal adjustments to the provided code. The "Advanced Model" section details the team's decision-making process for adjusting our model's parameters to achieve a higher accuracy score and overcome the original model's shortcomings. The hardware used was a 2024 MacBook, with 24 GB of unified memory and 12 CPU cores.

Original Model

For our first model, we chose to prioritize speed performance over all else. We reasoned that once we developed a working prototype, we could further optimize it for accuracy and generalization, so we aimed to create this model as quickly as possible. The parameters that optimized performance were a large batch size of 100 and a small epoch size of 1, as seen in Table 1. We did not adjust the learning rate from its original value of 1e-5.

	Batch Size	**Learning Rate**	**Epochs**	**Accuracy**	**F1 Training Score**	**F1 Validation Score**	**Training Time (*avg. sec x epoch)***
Original Model	100	1e-5	1	0.80692	0.79	0.79	697s x 1
Advanced Model	32	2e-5	3	0.82899	0.84	0.80	675s x 3

Table 1 – Parameter & Accuracy Comparison across the Models

The original model performed moderately well, with an accuracy of 0.80692, and had a relatively short training time of approximately 11.6 minutes, as seen in Table 1. The model also generalized well, as evidenced by the similar F1 scores between the training and validation sets in Figure 1. The total core usage fluctuated between 6 and 9, indicating that the system was suffering from CPU throttling due to the large batch size. This model also had a moderate rate of false negatives (N=560) and false positives (N=533), and its accuracy score left room for improvement. With these shortcomings in mind, we created our second model.

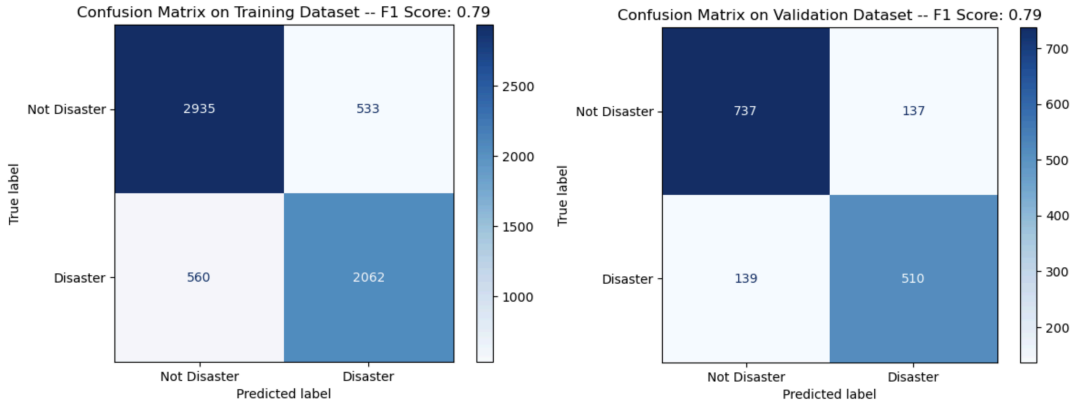

Figure 1 – Confusion Matrices for Original Model

Advanced Model

We used the paper by Sun et al. [1] to guide our implementation decisions for the advanced model. In this paper, they discovered that a learning rate of 2e-5 or lower is necessary to prevent catastrophic forgetting, a phenomenon in which pre-trained knowledge is erased during the learning of new information. Thus, we choose a slightly larger learning rate of 2e-5 to speed up convergence while overcoming this issue. We also lowered the batch size to 32, a power of 2, which helps with GPU integration. This batch size was enough to optimize GPU usage without CPU throttling. Since our largest floating-point vector's size was 151, this batch size also fit within our machine's memory limitations. During training, 9 of 12 cores were used, resulting in 75% GPU utilization. We chose not to balance the dataset because it was not severely imbalanced: 57% of the training set were Not Disaster Tweets, and 43% were Disaster Tweets. For the number of epochs, Devlin et al.'s paper [2] found that three epochs were usually sufficient for GLUE (general language understanding evaluation tasks). We implemented an early-stopping function based on observed plateauing in value loss, with a maximum epoch limit of four. Our model stopped after three epochs, which is consistent with the literature.

The advanced model saw a slight improvement, with an accuracy score of 0.82899. The F1 score on the training data rose to 0.84, and there were fewer false positives (N=197) and false negatives (N=576), as shown in Figure 2. However, the lack of improvement in the validation set indicates that this model suffers from overfitting. This model took 33.7 minutes to train.

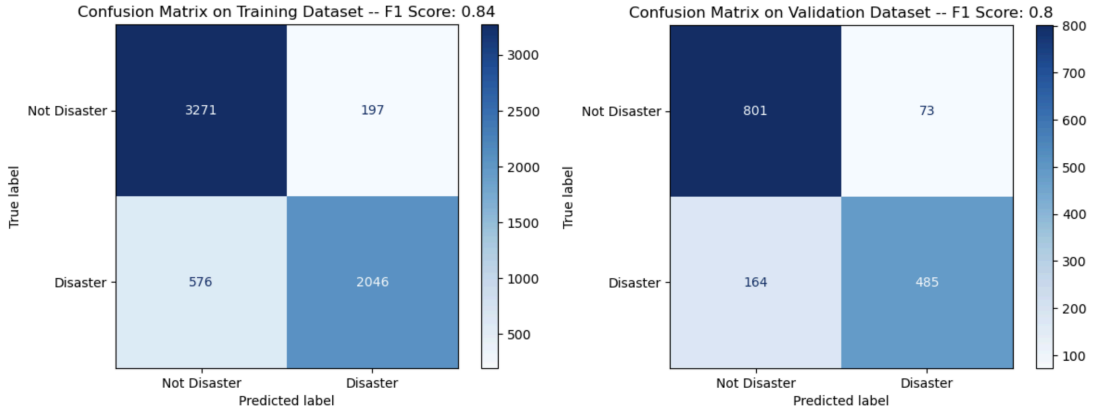

Figure 2 – Confusion Matrices for Advanced Model

Conclusion

While slight improvements were observed in the advanced model, significant trade-offs in training time were noted, and its F1 score suggests overfitting. We speculate that the BERT classifier model will see minimal improvement in accuracy, even with the implementation of other techniques. Future research should focus on using other pre-trained models to compare their performance with BERT's.

References

[1] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to Fine-Tune BERT for Text Classification?," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019. doi: 10.1007/978-3-030-32381-3_16.

[2] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019.