Introduction

Our final project of NLP was disaster tweet classification. The primary objective is to develop and evaluate an NLP model capable of distinguishing tweets related to disaster and those that are not. Our report will describe our dataset, models, and purpose of using those models, the setup used for the models, the results and conclusions that we've derived from our models, and our future direction in this project.

Dataset

The "Natural Language Processing with Disaster Tweets" dataset sourced from Kaggle consists of 10873 tweets with three features (not including "id") and the target variable [0 for unrelated disaster tweet, 1 for disaster tweet]. We chose to do this text classification problem due to the increase in misinformation on today's social media platforms. The tweets from the dataset could either be related to the disaster, consisting of natural disasters (like earthquakes, hurricanes, or wildfires), extreme weather events (like heat waves and storms), or man-made disasters (like oil spills and transportation accidents). Or, the tweets could be completely unrelated to disasters, which would be classified as fake disasters. The three features of our dataset include 'text' (the body of the tweet), 'location' (where the tweet was sent from), and 'keyword' (the most notable word of the tweet relating to a disaster).

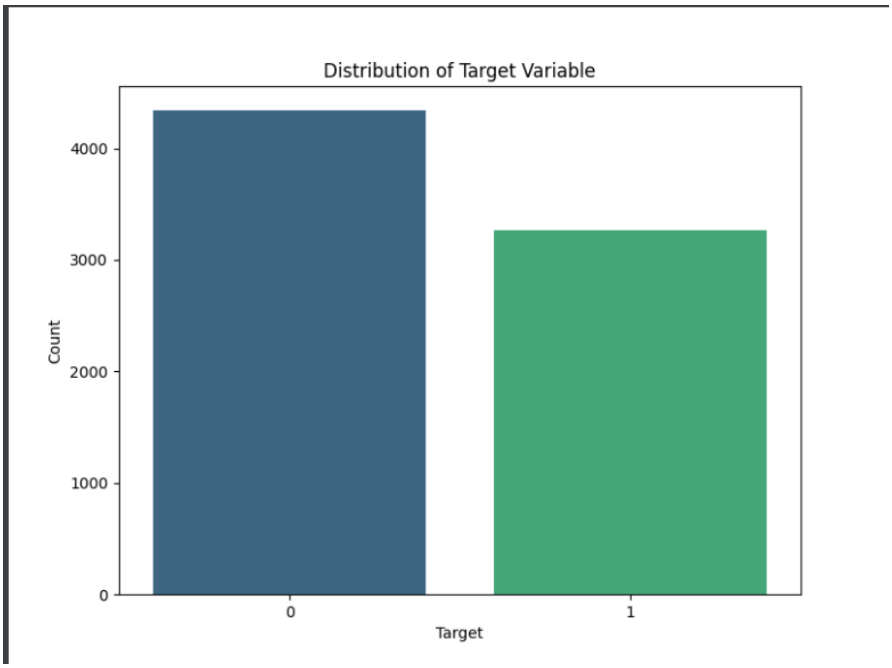


Figure 1: Distribution of Target Classes in the Dataset

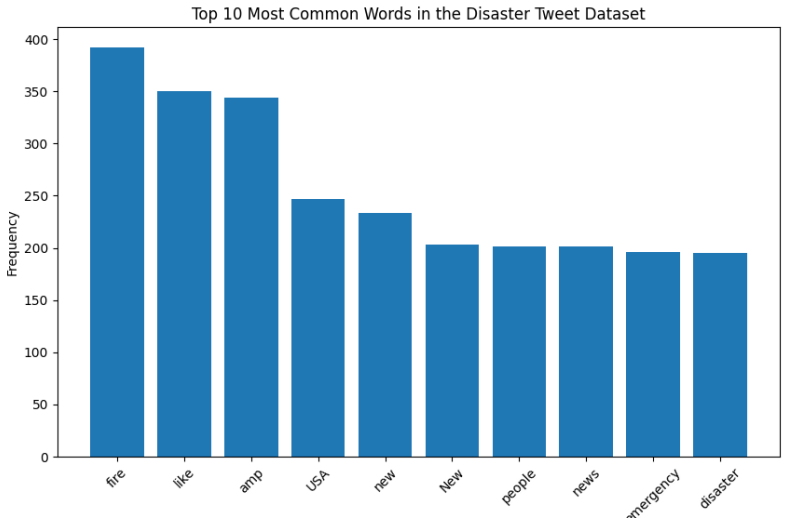


Figure 2: Most common words in the Dataset

As depicted in Figure 1, our dataset exhibits a balanced distribution, with 4342 instances for target 0 and 3271 instances for target 1. From Figure 2, we can see that the most common words in the dataset often relate to emergencies or disasters that would be of interest to the target classes in the tweets. “Fire”, “disaster”, “emergency”, and “news” were often in the dataset, and the dataset was also centered around regions in the USA.

Description of the deep learning network and training algorithm

Naive Bayes:

The naive Bayes classifier is a popular supervised machine-learning algorithm used for classification tasks such as text classification. It is categorized as a generative learning algorithm, implying that it models the distribution of inputs associated with a specific class or category. This methodology relies on the assumption that the features of the input data are conditionally independent given the class, enabling the algorithm to make prompt and accurate predictions. One of the strengths of naive Bayes is its simplicity. It's easy to implement and computationally efficient, making it particularly suitable for large datasets. The algorithm performs well even with limited computational resources. Below is the equation which is used to calculate the posterior probability using naive Bayes.

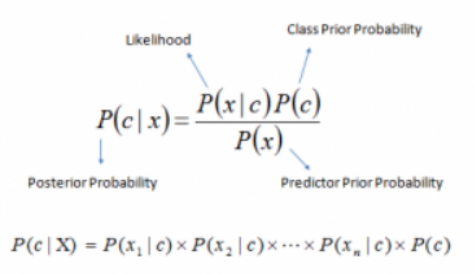
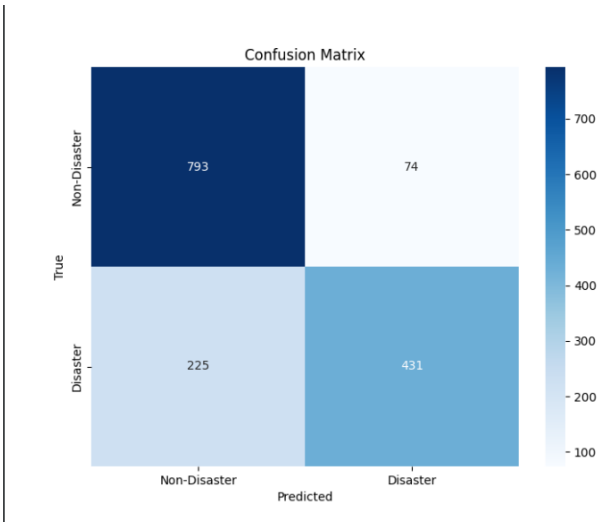


Figure 3: Equation to calculate the posterior probability using naive Bayes

In Figure 3, P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes). P(c) is the prior probability of class. P(x|c) is the likelihood which is the probability of the predictor given class. P(x) is the prior probability of the predictor.

After training my naive Bayes mode I created the confusion matrix, you can see that for the Non-disaster tweet, 756 tweets are being correctly classified as Non-diaster., whereas 109 are being misclassified. For the disaster tweet, 469 are being correctly classified as disaster and 170 are being misclassified.



Bidirectional LSTM:

Bidirectional Lstm is a recurrent neural network primarily used for natural language processing. Unlike the standard LSTM, where the input flows in one direction, in bidirectional LSTM, the input flows in both directions. It is capable of utilizing information from both sides. So the

BiLSTM adds one more LSTM layer which reverses the direction of information flow. Which implied that the input sequence flows backward in the additional LSTM layer. Then we combine the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or concatenation. The advantage of this mode is that every component of the input sequence has information that is both past and present. For this reason, BiLstm can produce a more meaningful output as it has both forward and backward layers.

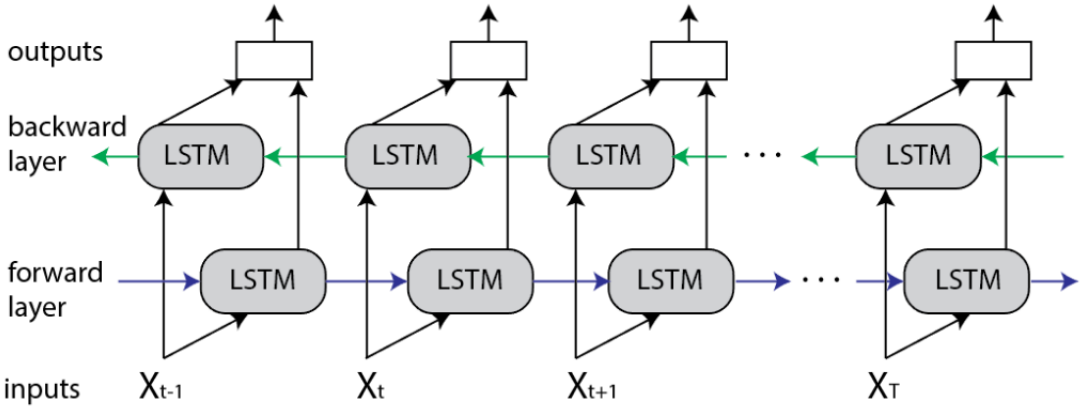


Figure 4: Bidirectional LSTM architecture

So training my bilstm for 20 epochs, For each epoch I am printing F1_score, validation loss, and training loss. whichever epoch is giving me the f1_ best score I am saving the model and I am printing the confusion matrix for it as seen in the below figure. So In the confusion matrix, you can see that for the Non-disaster tweet, 756 tweets are being correctly classified as Non-diaster., whereas 109 are being misclassified. For the disaster tweet 469 are being correctly classified as disaster and 170 are being misclassified.

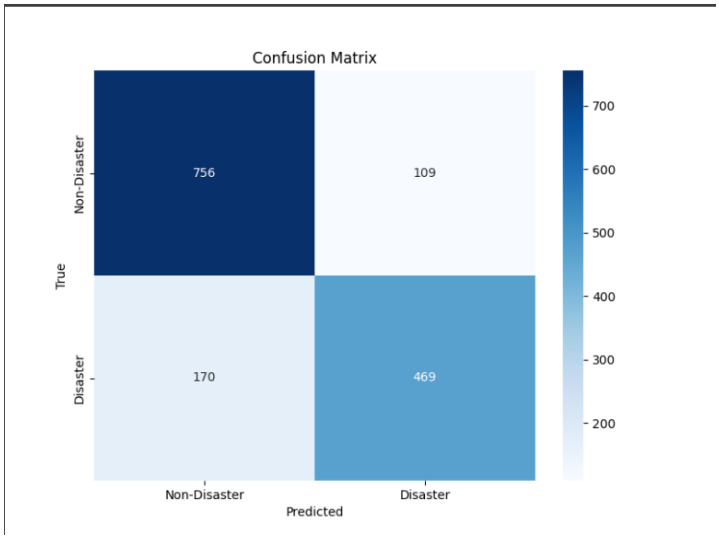
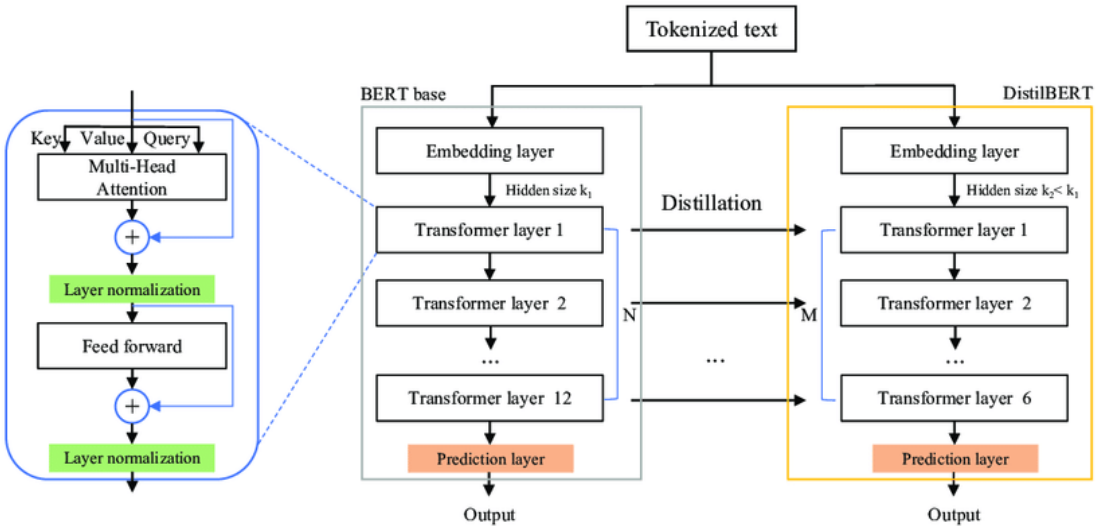


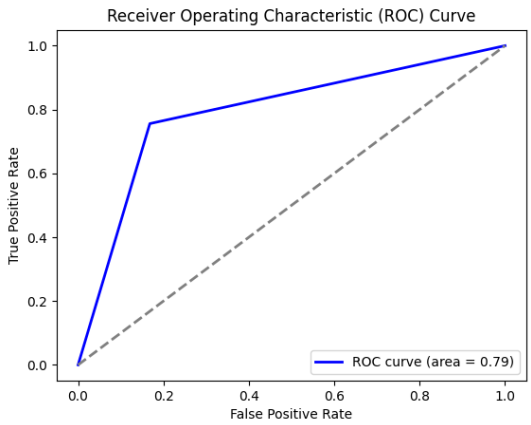
Figure :

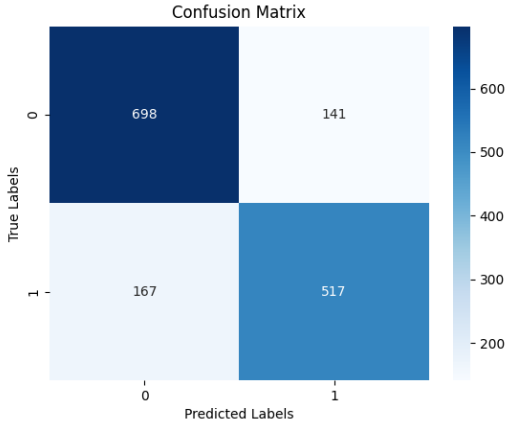
DistilBERT:

DistilBERT is a transformer-based model created from knowledge distillation, trained to learn the logits of the original BERT model. For example, while BERT's distribution of a multiclass classification problem could have a softmax distribution of .6, .2, and .2 for three different classes, the training goal of DistilBERT is to score a .6, .2, and .2 for the same data observation. In doing so, the hope is that the knowledge distillation will allow DistilBERT to track the same context and patterns that the original and larger BERT model used to create the scores in the first place. DistilBERT has 40% less parameters than the original BERT model, trains 60% quicker, and scores 95% or above of what the original BERT model's scores on language capability benchmarks. The transformer layers of DistilBERT are half of the transformer layers of the original BERT base (as you can see from the figure below), and in the paper "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter" by Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf, it was noted that it's language capabilities would often outperform an ELMo encoder model with two bidirectional LSTMs.



From the ROC curve and confusion matrix below, we can see that, after training for 5 epochs, the AUC was .79, and the confusion matrix is fairly well distributed across true positives and true negatives, along with false positives and false negatives. The f1 score reached 0.797, and the accuracy was .798. Due to the distribution of the confusion matrix, we can see that the model is not heavily leaning towards classifying true negatives nor true positives, and although the false positives were more than the false negatives, this was also due to the original distribution of the dataset having more class 0s than class 1s.





Implementation and Experimental Setup

Our procedure to create, compare, and test our models was to train through a few epochs before comparing our model results. We kept the preprocessing (other than specific tokenizers or embedding layer setup) the same for each of our models.

Naive Bayes:

a) Ngram_range:
I have used (1,2) as the Ngram_range when calculating tfidf vectorizer of the dataset. I have experimented with different Ngram_range but using this was giving me the best accuracy in the model.

b) Max_features :
I have used the 10000 as the max_features when calculating tfidf vectorizer of the dataset. I have experimented with different Ngram_range but using this was giving me the best accuracy in the model.

Bidirectional LSTM:

a) Epochs: 20
We used 20 epochs to finetune our Bidirectional model on our dataset. First I kept epochs to 10 only but I found out that my F1 score was increasing as I increased the epoch and I am saving the model that gives the best F1 score.

b) Batch Size: 32
We use a batch size of 32 for the Bilstm model to balance computational efficiency.

c) Learning Rate: 3e-4

I am using 3e-4 as the learning rate because it gives me the best result in terms of validation accuracy and f1_score. I also used different 1e-3 and 1e-4 but the model did not give me that high accuracy as the 3e-4 did.

d) Dropout :
I have added a dropout layer in the model architecture to avoid overfitting the data. I have used two dropout layers of 0.5 one after the Lstm and one after the first fully connected layer.

e) Glove embedding matrix :
I also used the GloVe embedding matrix, 'glove.6B.300d.txt,' for the embedding layer of my model. Through experimentation, I discovered that my validation accuracy increased compared to using a self-created embedding layer.

DistilBERT:

a) Epochs: 5
We used five epochs to finetune our DistilBERT model on our dataset. Originally, the number of epochs was lowered to 3 and the weight decay was also lowered, but the results of the model were nearly the same (slightly lower than the .79 f1 score on the validation set) no matter the change.

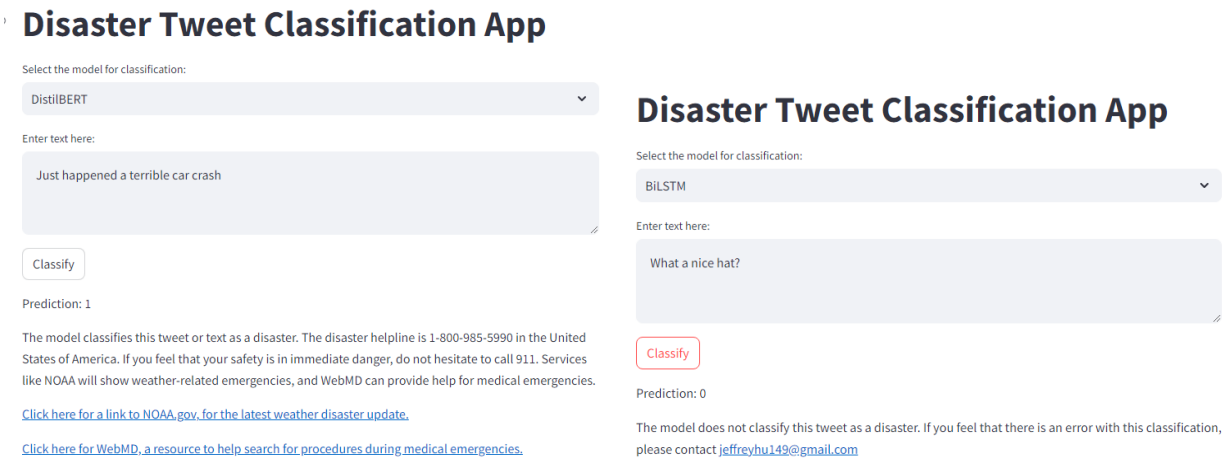
b) Batch Size: 16
We used a batch size of 16 for our DistilBERT model to balance computation and time costs.

c) Learning Rate: 1e-6
As this is a pre-trained model without transfer learning, we used a low learning rate to keep its original language processes intact from its knowledge distillation process.

d) Weight Decay: 0.05
We also kept the weight decay high to counter the possibility of overfitting our models, even on such low epochs. Given that our dataset was mostly comprised of short tweets, we didn't want the DistilBERT model to lose the language capabilities that it originally learned by overtraining phrases that were not always grammatically correct or ordered.

Results

From the performance of our models, we saw that, while the neural networks from both constructed BiLSTM and pre-trained DistilBERT could outperform the Naive Bayes model, we also found that the models all performed relatively closely. This could be in part, due to the dataset structure of how some words were clear indications of one target or the other.
Below are some sample outputs from our app.



Summary and Conclusions

From our project, we can show that an affordable and light gpu model can be constructed to tackle this fake and real disaster tweet problem. We did run into a few problems with preprocessing, for example, due to filtering out stopwords using Spacy, past and present tense were ambiguous to the model in some situations, so past disasters and ongoing disasters were treated the same (I was in a car accident vs. I am in a car accident).
In the future, we plan to pay closer attention to the specific words/show embeddings and associations of the disaster and do additional data analysis on the focus/robustness of our models.
For future exploration, we could automate disaster classification on hashtagged tweets to filter out false or mistakenly tagged tweets from interfering with the information provided by the true disaster tweets and updates. We could also further classify positively predicted tweets between which type of disaster it is (fire, hurricane, man-made disasters, etc.), and give referral links/phone numbers as to which resource is best suited to the disaster.

References

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter. arXiv preprint arXiv:1910.01108.
Dataset:
https://www.kaggle.com/code/harrycheng5/nlp-fake-tweets-classification
https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm
https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
https://github.com/amir-jafari/NLP/blob/master/Lecture_08/Lecture%20Code/3-LSTM_Sentiment_Analysis_Custom.py