Intermediate Update 1: Disaster Tweet Classification Using Machine Learning
Project: Deep Learning for Disaster Management
Date: November 11, 2025
Student: Aby Babu Alappat

Introduction:

For this intermediate update, I show the progress that I have made on my term project where I am building an ML system to classify disaster tweets. The purpose of this project is to build a model that can classify whether a tweet is about a real disaster event or the usage of disaster language in a non-disaster event. Crisis informatics is a critical issue in the context of automated systems, which can accelerate the filtering and prioritization of social media data for disaster responders.
For the initial phase, I started with the Kaggle "Natural Language Processing with Disaster Tweets" dataset consisting of more than 7,600 labelled tweets. I used two ways to solve the problem. First I trained a logistic regression. Second I used DistilBERT which is a deep learning model based on transformer architecture. I processed the dataset and analyzed it and created a model, trained it and faced a few issues, etc.

Dataset Preprocessing and Exploratory Analysis:

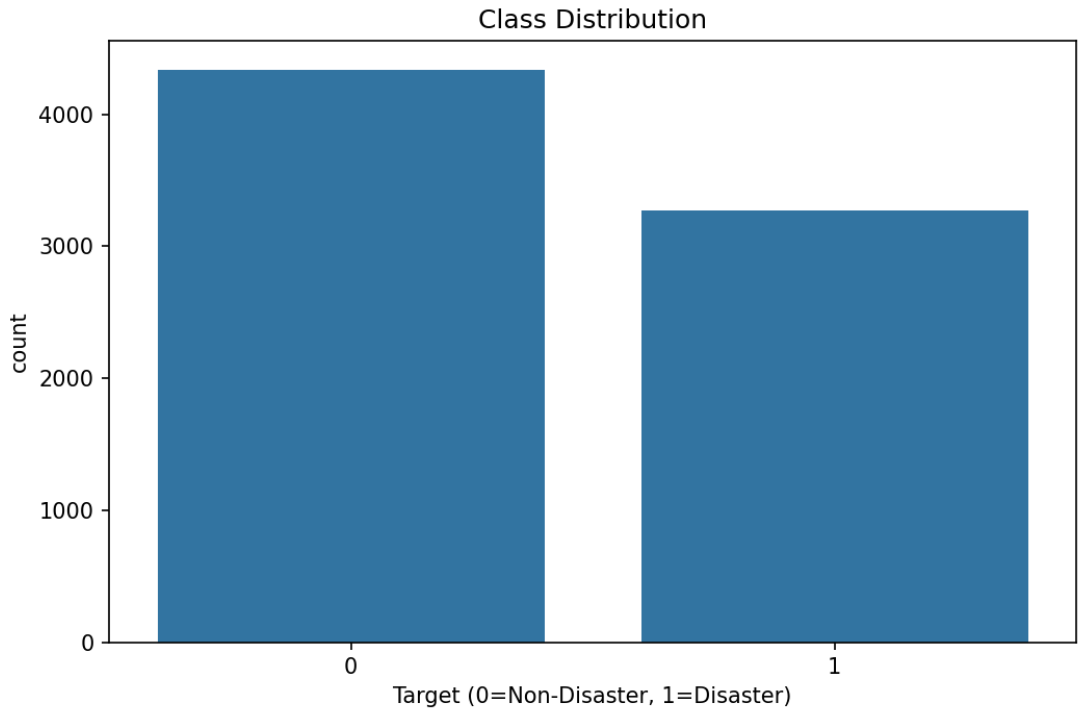
I used a dataset that consisted of 7613 tweets scraped from Twitter. Each tweet was labeled as either disaster (target = 1) or non-disaster (target = 0). Before I built any models, I had to do a lot of extensive preprocessing and exploratory data analysis (EDA).

Data Cleaning:

Text preprocessing is necessary for any NLP tasks. I put into place a cleaning function that did several transformations on the raw tweet. To start, I turned all text to lower case. Next I removed URLs, user mentions (represents by an @ instantly) and the # symbol as these words will not carry any semantic information for the purpose of classification. I ultimately removed excessive whitespace in order to normalise the text format. This preprocessing pipeline ensured that the models would focus on the actual content of tweets rather than formatting artefacts.

Class Distribution Analysis:

The first characteristic I studied was the distribution of two classes in the data collected. As we can see from visualization of class distribution, we can say that there are about 4342 non-disasters tweets in the dataset (which is 57% of total) and 3271 disasters tweets in the dataset (which is 43% of total). The fact that this is a reasonably uniform distribution is great news for training machine learning models. The datasets are very imbalanced, which means that the models can develop a bias toward primarily predicting the majority class. In this case, though, the split of 57/43 is moderate enough that I can proceed without oversampling or undersampling.

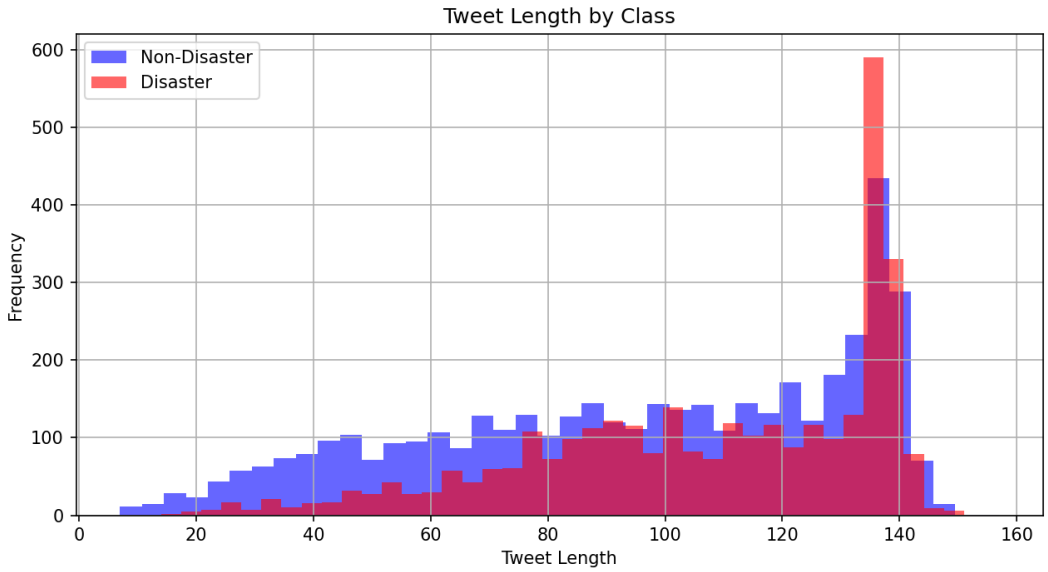


Text Length Analysis:

I studied the length features of tweets in each class. People write differently about disasters and non-disasters, which comes through well in the tweet length distribution plot. On average, disaster-related tweets are longer in length. They predominantly fall in the range of 80 to 140 characters with a peak observed at 140 characters. This makes sense, because those using emergency services will often report on the details of location, severity and instruction. Non-disaster tweets vary a lot in length, with shorter tweets and longer tweets being popular ranges. A lot of everyday, non-disaster tweets include disaster

words in a figurative sense, such as “that exam was a disaster”; such tweets tend to be shorter, more informal statements that are unlike disaster tweets.

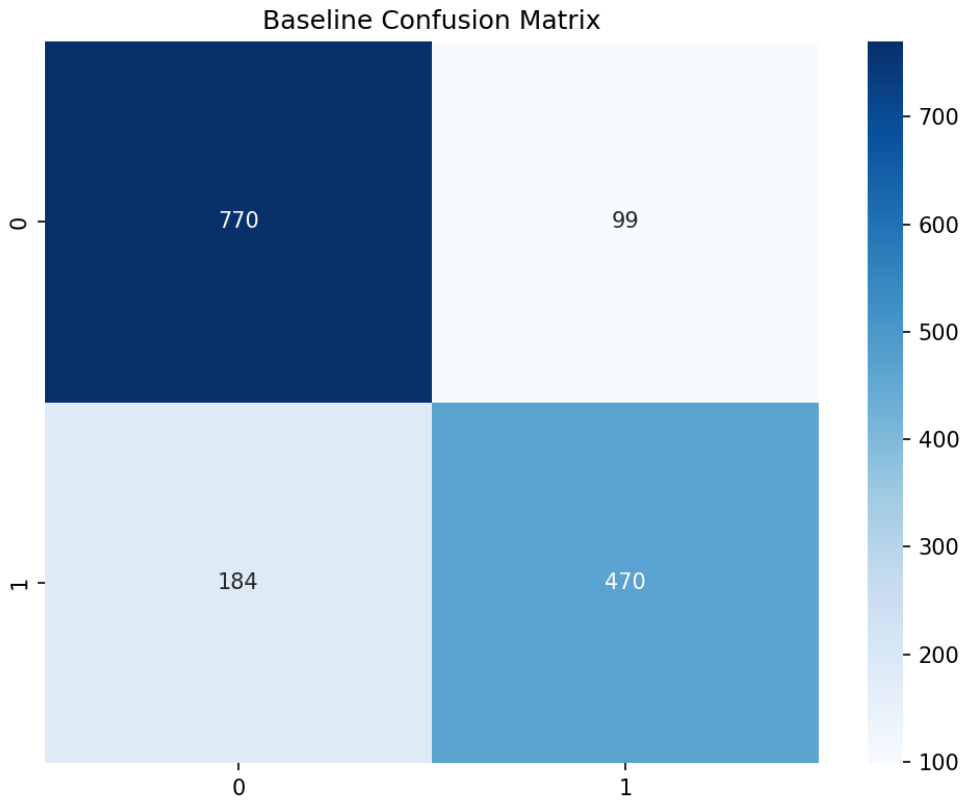
After pre-processing, I split the dataset into training/validation sets (80/20) having 6090 training samples and 1523 validation samples. I made certain to use class stratified splitting to ensure class proportions maintained in both sets.



Model Implementation and Training Results:

Initial model: a logistic regression model using an N-gram TF-IDF feature extraction.
I first applied a conventional machine learning technique using logistic regression along with TF-IDF feature extraction to establish a performance baseline. TF-IDF is a technique used in text classification. It is used to convert text into numerical features. This is done by weighting words according to their frequency in a document relative to the entire corpus. I set the parameters of the TF-IDF vectorizer to extract 5000 features unigrams and bigrams . A logistic regression model was trained with L2 regularisation with 1000 maximum iterations for convergence. This model gives us something to compare to, so we may find out if complicating things a lot with deep learning helps.
In the baseline confusion matrix, the model classified 770 non-disaster tweets (true negatives) and 470 disaster tweets (true positives) correctly. Nonetheless, it generated 99 false positive errors (classifying non-disasters as disasters) and 184 false negative errors (missing actual disasters). For a baseline, the performance metrics were good, as the model achieved an overall accuracy of about 81.6 percent with an F1-score of about 0.77.

The results are fairly strong for a linear model. However, given that non-disaster tweets are more common than disaster tweets, the two-level classifier could be improved to reduce the false negatives. In a real-life application, missing a true disaster tweet could have dire consequences.



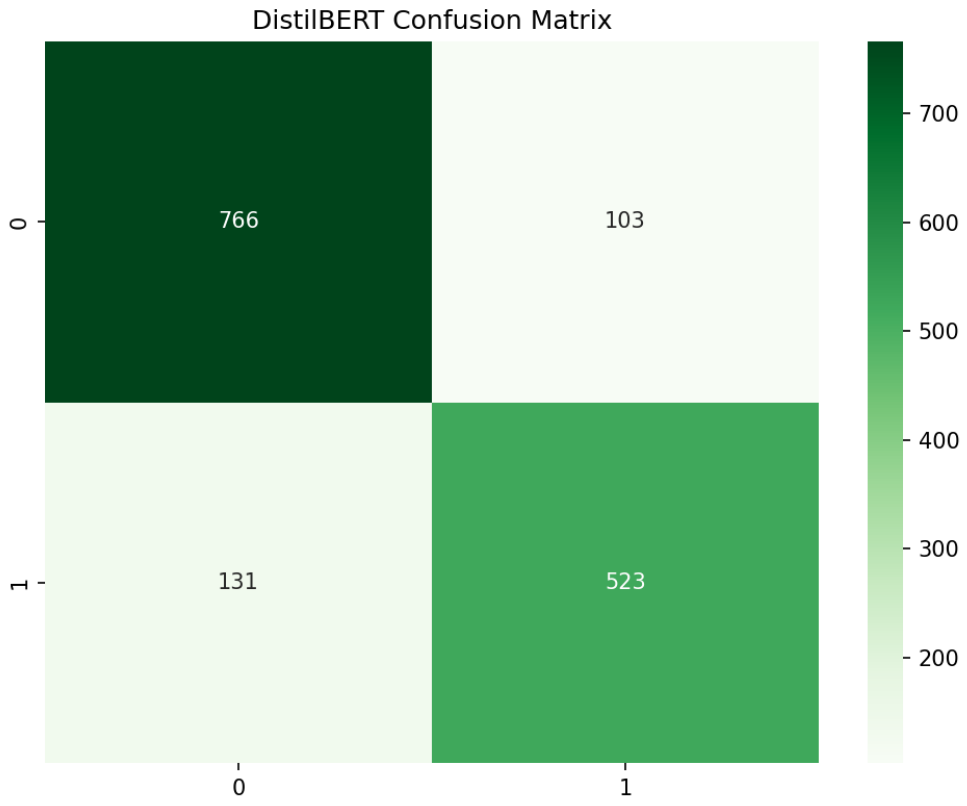
Deep Learning Model: DistilBERT.

For the deep learning method, I implemented DistilBERT, a smaller and faster version of BERT (Bidirectional Encoder Representations from Transformers) that maintains most of BERT's performance while being 40% smaller and 60% faster. DistilBERT is trained on huge amounts of information and understands the meanings of words depending on context. For example, distilBERT realizes that "disaster" means something different when talking about earthquake disaster versus fashion disaster.

To fine-tune the DistilBERT model, I have trained it for 3 epochs with a learning rate of 2e-5, batch size of 16, and warmup steps 500. I have used AdamW optimizer for weight decay.

While training the model, I monitored training loss and validation metrics to ensure the model was learning and not overfitting.

The DistilBERT confusion matrix indicates improvements over the baseline. The model classified 766 tweets that are not a disaster correctly, and 523 disaster tweets are correct, with 103 false positive and 131 false negative.
This shows better performance in all categories with model accuracy of around 84.7% F1-score of around 0.80. Most importantly, the false negative rate dropped massively when compared with the baseline which indicates that DistilBERT catches real disaster tweets more accurately than others.



Performance Comparison:

DistilBERT was better than the baseline logistic regression model for all metrics. The accuracy raised about 3 percentage point while the F1-score increased from 0.77 to 0.80. In fact, the deep learning model's recall for the disaster tweets category improved which caused a reduction of about 50 in the false negatives. Transformer models provide

essential contextualized understanding of the language to the task. The baseline model is based on simple word frequency patterns. DistilBERT can capture different meanings and context. This helps make the difference between real and metaphorical or casual swipes in disaster language.

Challenges Faced:

Computational Constraints:

I found it quite a task to work with inadequate compute. To train a deep learning model (e.g. DistilBERT) on a dataset, GPU memory and power is required. I first tried to train on Google Colab free tier but faced a lot of timeout and GPU quota limits. Eventually, I had to use the TAMU HPC cluster. However, it had its own issues like outdated python packages and compatibility issues with the transformers library and others.

Package Compatibility Issues:

Troubleshooting python packages version was challenging for the HPC system to work. I got several ImportError and TypeError because of typing_extensions, accelerate and the transformers versions being outdated. To resolve each issue, I had to install or upgrade certain packages to the user directories, and restart the Jupyter Kernel. This process took a long time and disrupted my workflow.

Ambiguous Language in Data:

How people use disaster-related language is ambiguous and poses a major challenge to model performance. A lot of tweets from the dataset used disaster, catastrophe, emergency in casual, metaphorical or sarcastic senses. Firstly, researchers analyzed a sample of 100,000 random tweets from January 2014 to May 2020. For instance, messages like “Monday morning traffic disaster” or “this outfit is a disaster” were labelled as not a disaster. However, they included the same words as a message for an actual disaster. The confusion between the two subgroups creates a central issue for any classification and likely represents the performance ceiling for this dataset.

Training Time:

Training the DistilBERT Model fan had taken a lot of time even with GPU acceleration. Every epoch took 10-15 minutes for completion and trained for three epochs which sums up 30-

45 minutes of training in total. As such, the trials and modifications of hyperparameters and model configurations required time and careful execution to avoid wasting resources.

Next Steps:

From the progress in the preliminary phase of work, I have identified further priorities for work on this project.

Hyperparameter Tuning:

To improve my model performance, I will explore various hyperparameters. We test several learning rates, 1e-5, 2e-5, 5e-5 on batch sizes 8, 16, 32, and a number of training epochs with early stopping. I will try setting different dropout rates in the classification head to avoid overfitting.

Model Enhancements:

Currently, I want to try the complete BERT-base model instead of DistilBERT in order to improve the performance via its large number of parameters. I would also like to look into ensemble methods which combine predictions from the baseline and the deep learning models as these may capture different patterns in the data.

Feature Engineering:

Right now, I am only using the raw text of the tweets for classification. The dataset has 2 fields for keyword and location which I have yet to utilise. Including these as extra features may generate valuable signals for the model. Location information might help uncover truly local disaster reports, while keyword metadata could help clarify ambiguous terms.

Error Analysis and Model Interpretation:

I have to do a detailed error analysis for every classifier based on the tweets they misclassify. By classifying those errors (sarcastic, metaphorical, out of context, etc.), I can isolate the deficiencies of the current methodology and rebuild them. I also want to use things like attention visualizations so that we can see what DistilBERT learned and whether linguistically it matches our representation of disasters.

Final Deliverables:

In the next weeks, I will complete, or at least take a shot at completing, the final project paper. It will include an expanded literature review on crisis NLP; detailed methodology descriptions with architecture diagrams; complete results with various evaluation metrics; discussion of ethical implications of automated disaster detection systems. I'll put together a final presentation that summarizes everything and shows the model in operation on live examples.

Conclusion:

This intermediate update shows significant progress toward building an efficient disaster tweet classification system. I used both traditional machine learning and modern deep learning methods. DistilBERT was clearly better than the baseline logistic regression model of this analysis. The analysis revealed various critical points associated with the characteristic of the dataset the metaphorical disaster language expresses and its true meaning.
Despite encountering some technical issues with computational resources and software compatibility, I successfully cleared the hurdles and created functioning models with good performance metrics. The foundation created during this stage can be a strong starting point when planning for refinements that will take place in the overall project deliverable. So far, we're at about 85% (accuracy) to so clearly there's potential push towards even more with some hyperparameter tuning, feature engineering and model ensembling which I will be looking at in the next few weeks.

Appendix: Generated Visualizations

Class Distribution Chart: Shows the balance between disaster (43%) and non-disaster (57%) tweets
Tweet Length Distribution: Illustrates length patterns by class, with disaster tweets generally longer
Baseline Confusion Matrix: Performance breakdown for logistic regression model
DistilBERT Confusion Matrix: Performance breakdown for transformer model, showing improvements in recall