Natural Language Processing with Disaster Tweets

Cheryl Stanley
c3stanley@ucsd.edu

1 Introduction

In the age of social media and as smartphones have become an increasingly ubiquitous form of communication, Twitter has become an increasingly widespread and commonly used tool for alerting the global population to current events and news in real time. The goal of this project is to use sentiment analysis and natural language processing to analyze whether or not a tweet was referring to a disaster or not.

The latent ability, using Natural Language Processing techniques, to instantly classify a tweet as referring to a disaster would greatly impact and benefit disaster relief initiatives, allowing awareness to spread quickly. Awareness of disasters as they happen would also allow subsequent relief and aid to be administered in a timely manner. The first step towards reaching this goal is understanding the best way to preprocess the textual data given to us in the tweets, classify sentiment of the tweets available to us which is done through analysis of different models and techniques, as we will subsequently explore in this paper. The following are the steps completed for this task, listed out

- Collected / preprocessed Disaster Tweets Kaggle dataset: DONE
- Build and train Neural Network (baseline model) on collected dataset and examine its performance: DONE
- Make pretrained DistilBERT model with learnable parameters perform better than the baseline model: DONE

2 Related Work

There are a number of existing academic journals and research papers that have tackled the topic of sentiment analysis techniques using natural language processing and machine learning classifications of the social media space.

One such research journal which tackled a similar task to my current one of predicting sentiment from tweets is “Social Media Sentiment Analysis On Twitter Datasets” [2]. In this paper, the authors investigate sentiment analysis with a focus on hate speech detection in tweets. Random Forest and SVC achieved the highest accuracy (95%), highlighting their effectiveness for

this task. The paper outlines preprocessing steps like tokenization and noise removal, which align with my task of preparing tweets for sentiment analysis.

In another paper, "Social Media Contents Based on Sentiment Analysis and Prediction System" [3] , the stated research goal was to develop a system to detect events in real-time on social media (e.g., disasters, accidents) and analyze users' sentiments. In the paper, the authors utilize keywords and user-generated social media content as event sensors. A CNN and an LSTM were utilized, and performed well on the event detection tasks.

Another paper which was relevant to my task was "Twitter Sentiment Analysis using Natural Language Processing" [4] which. In this paper, an Artificial Recurrent Neural Network (ARNN) with bidirectional LSTM is proposed to automate the classification of Twitter sentiments (positive, negative, and neutral). A Bag of Words (BOW) model was used for feature extraction and text was tokenized and converted into vectors, as well as padded for uniformity. Based on the results of this paper, I decided to also utilize a similar approach with how I implemented my model.Another paper with similar premise to my goal of classifying disaster tweets using sentiment analysis was the paper, "Social media sentiment analysis: lexicon versus machine learning",[5] This paper looked into classifying Facebook comments into positive, negative or neutral sentiments. The combined approach of LIWC2015 lexicon-based approach along with classifiers such as Maximum Entropy and Bagging improved positive sentiment classification greatly without compromising negative sentiment classification performance.

I also discovered "DistiliBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper, and Lighter" [6] while I was looking for a pretrained model to enhance performance. In the paper it discusses how it is possible to reduce the BERT model by 40% while still leveraging most of BERT's faculties for understanding and being 60% faster, and was appealing due to its ability to replicate BERT level performance in less time.

3 Dataset

The dataset consists of labelled tweets with the following columns: id - a unique identifier for each tweet, **text** - the text of the tweet, **location** - the location the tweet was sent from (may be blank) and **keyword** - a particular keyword from the tweet (may be blank). In the training file only, a **target** column exists which denotes whether the tweet is about a real disaster (1) or not (0).

Some relevant statistics include there being an average word count of 14.91, median word count of 15.00, max word count of 31, and min word count of 1.

Dataset Statistics

Training Set Size	6090
Validation Set Size	1523
Test Set Size	3523

Here are some sample input / output pairs for reference:

Sample Input/Output Pairs:

Example 1:
Input: Courageous and honest analysis of need to use Atomic Bomb in 1945. #Hiroshima70 Japanese military refused surrender. https://t.co/VhmtyTptGR
Output: 1

Example 2:
Input: @ZachZaidman @670TheScore wld b a shame if that golf cart became engulfed in flames. #boycottBears
Output: 0

As you can see, from the sample input, output pairs, tweets can often have incomplete or misspelled words. Words that are associated with disasters (such as "Bomb" in Example 1) can often be a strong indicator that the Tweet is referring to an actual disaster, resulting in an output of 1 indicating it is in fact a disaster tweet. However in some cases words associated with disasters (such as "flames" in Example 2) can be used associated with theoretical situations or sarcasm, resulting in an output of 0.

One potential reason that this classification task could be challenging are lack of context: The model might struggle with tweets that require broader context or world knowledge. The model would need to be trained extensively to get a handle on worldwide data, and might not generalize well to specific situations or events.

Another reason is that the model might not understand sarcasm to the same degree that a human being can, and short or incomplete tweets and misspelled language might exacerbate this.

3.1 Data Preprocessing

Since the dataset is heavily used and was obtained from Kaggle, the data was relatively clean and preprocessing was very minimal, however a few steps were taken to ensure the expected inputs were fed to the model.

Since the users were anonymized, this dataset encodes the usernames via a specific unique integer (the *id* column of the train and test files). Tweets often allow users to share the user's current location, however this is optional and can be toggled off. The same can be said about keywords or tags. These attributes are encoded in the *keyword, location,* columns of the train and test files.

Due to the fact that these values are optional when creating and publishing tweets, many of the entries may have missing values for this, so when reading the train and test files I handled missing values by filling them with empty strings, ensuring there would be no

"*NaN*" values in columns, which would cause issues.

The most significant preprocessing step was the text_vectorization step, where text was tokenized, vocabulary was created out of the most frequent 10,000 tokens *(max_tokens = 10,000),* and padding / truncation was utilized such that all sequences have a length of 50, either by truncating longer sequences or padding shorter ones *(output_sequence_length = 50)*. Lastly, the vectorizer was then applied to transform the text data, converting the raw text into sequences of integer indices ready for input into the subsequent neural network and DistilBERT model.

---

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.

The baseline model is a simple sequential neural network, utilizing the TensorFlow and Keras libraries, which has an embedding layer, a global average pooling layer, and 2 dense layers with dropout in between. The input text was converted into vectors and passed through an embedding layer to create dense vector representations of the words. Then the pooling layer reduces the dimensionality by taking average across the sequence dimension. What follows are 2 dense layers with dropout in between for feature extraction and classification, whose output is a single neuron with sigmoid activation for binary classification (disaster or not disaster).

The hyperparameters can be seen in the below table, although some notable ones are the training and validation data split where training data and test data are initially split into a 60 /30 split, and the train_test_split function if further used to divide the train data into an 80 / 20 training / validation data split from the initial training data. This leaves a final training / validation / test ratio of 48% train. 12% validation and 40% test. This split for validation is important especially since the labels for test and validation data are unknown when participating in a kaggle competition, and the model performance is measured off of performance on validation data. The test set is separate and only used for final predictions on the Kaggle website, and provided such that expected columns in the test data for the purpose of making predictions could be known. Hyperparameters include BATCH_SIZE of 32, EPOCHS = 5, VAL_SPLIT = 0.2, max_tokens = 10k, output_sequence_length = 50, embedding_output_dim = 64, dropout_rate = 0.3, and hidden_layer_units = 16

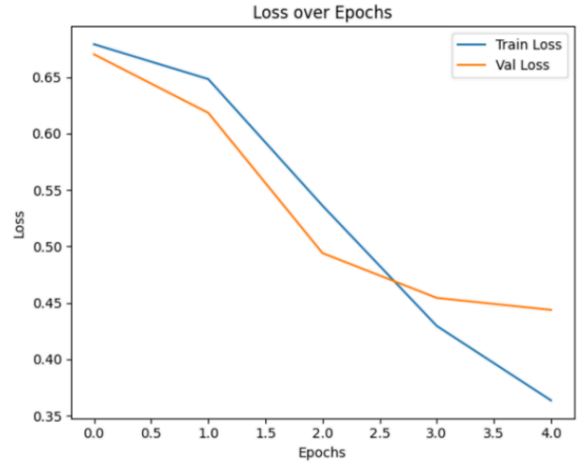
Hyperparameters

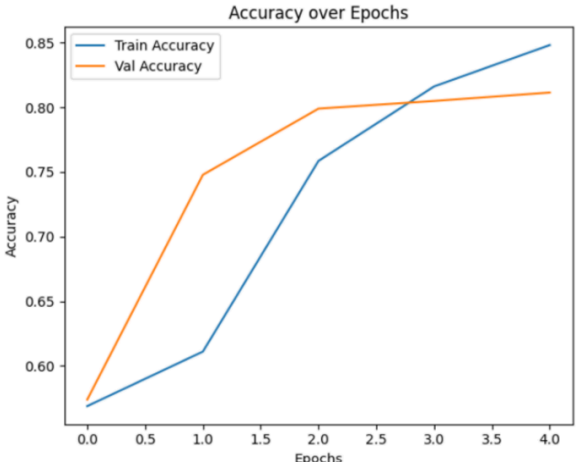
BATCH_SIZE	32
EPOCHS	5
VAL_SPLIT	0.2
max_tokens	10,000

(TextVectorization)	
output_sequence_length	50
embedding_output_dim	64
dropout_rate	0.3
hidden_layer_units	16

Figure 1: Baseline Model Parameters

Below are the results for the baseline model (left, training / validation loss graphed and right, training / validation accuracy graphed). I achieved a test accuracy of **79.93%** for my baseline model.





Baseline Model (Neural Network with no pretrained embeddings) Training and Validation Loss (Left) Training and Validation Accuracy (Right)

5 Your Approach

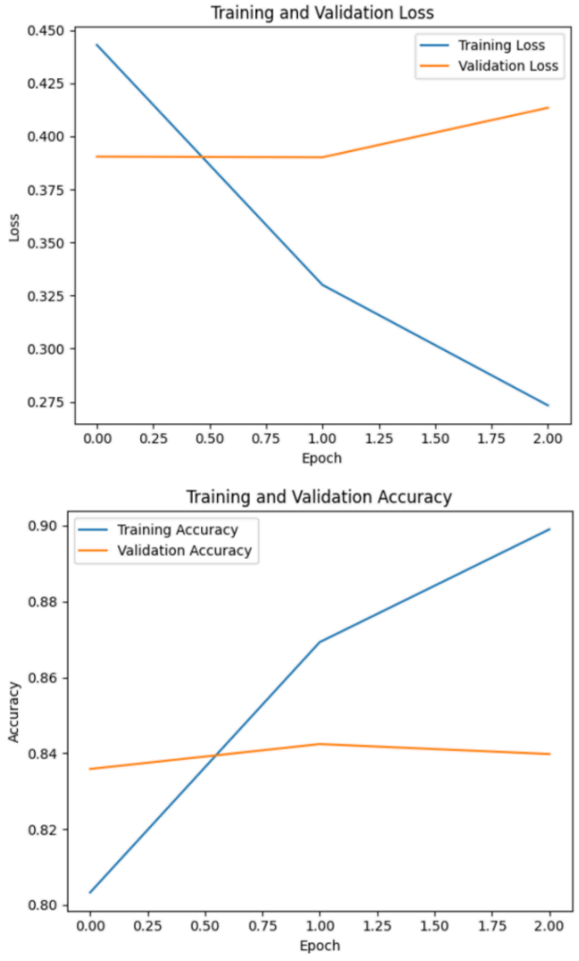
Here is a list of things to include under this section.

- (CONCEPTUAL APPROACH) I used a DistilBERT pretrained model with learnable parameters and a DISTILBERT tokenizer which converts the text into a format the model can understand. The entire model is fine-tuned on the Disaster Tweet dataset, allowing it to adapt its pretrained model parameters to this specific dataset. The model outputs logits which are then converted into disaster tweet predictions.
- (WORKING IMPLEMENTATION) I was able to complete a working implementation which successfully classifies disaster tweets with an accuracy of 80%, and implemented both models. All models are locared in different labelled cells in the CSE 256 Final Project.ipynb file.
- (COMPUTE) I ran my code on my personal laptop with a notebook in Google Colab,
and utilized the provided T4 GPU. My code is organized into separate labelled cells for the baseline and more advanced model, and both take no longer than a few minutes each at most to run. On Google Colab, some hacks I used besides the GPU utilization with cuda included the usage of drive.mount to read from

and write to files located in my drive. I did run into a few issues with using BERT from the HuggingFace library, but I was able to resolve it by updating all of my packages to their latest version

• (RESULTS) As can be seen in the table below, my baseline model which utilized a Neural Network was able to achieve an accuracy of **79.93%,** while my advanced model of BERT with Learnable Parameters outperforms this by almost 4%, achieving an accuracy of **83.64%,**

Below are the plots for training and validation loss for my advanced model (BERT with learnable parameters and pretrained embeddings). I achieved an improved performance, obtaining a test accuracy of
83.64%, 4% higher than my baseline model.



Advanced Model (BERT with learnable parameters and pretrained embeddings)

Training and Validation Loss (Top) / Training and Validation Accuracy (Bottom)

• (RUNTIME) How long did it take to train your model?
It took no longer than 3 minutes to run the complex model due to my utilization of the GPU.

---

6 Error Analysis

Types of Errors

1. Misclassified True Positive Examples (True Label = 1, Predicted Label = 0)

- **Example 1**:

  Text: "so you have a new weapon that can cause un - imaginable destruction."

  Analysis: The model struggles to recognize "unimaginable destruction" as indicative of a disaster. It may associate "weapon" more with general violence than disaster events.

- **Example 2**:

  Text: "my favorite lady came to our volunteer meeting hopefully joining

her youth collision and i am excite http : / / t. co / ij0wq490cs"

Analysis: The key context of disaster is lost due to the personal narrative, making it hard for the model to pick out disaster-related cues.

2. Misclassified True Negative Examples (True Label = 0, Predicted Label = 1)

- **Example 3**:

  Text: "woman electrocuted # red # redblood # videoclip..."

  Analysis: The word "electrocuted" might trigger a disaster-related prediction, though this refers to an isolated event rather than a disaster.

- **Example 4**:

  Text: "who is bringing the tornadoes and floods. who is bringing the climate change..."

  Analysis: While mentioning disasters, the tweet discusses them hypothetically or symbolically rather than reporting an actual event.

Based on these results, my model relies on surface-level features (i.e. keywords such as “tornadoes” and “floods” and fails to understand nuanced contexts (e.g. personal narratives in Example 2), It additionally struggles with implicit disaster-related cues (e.g. “unimaginable destruction, Example 1)

I believe that while DistilBERT may be better at capturing context, it struggles with disasters that are implied and require specific pre-existing knowledge (Example 1). It also struggles with distinguishing hypothetical from real disaster mentions (Example 5) and handling noisy, narrative-heavy tweets (Example 2)

---

7 Conclusion

I was not surprised by my results, as they successfully replicated results from the previous literature I surveyed, and demonstrated the effectiveness of leveraging advanced sentiment classification techniques using pre-trained models (DistilBERT, as well as Deep Learning models (Neural Networks). My DistilBERT model achieved 83.64% accuracy, a near 4% improvement from the baseline neural network’s 79.93%, which parallels the results from the “Social Media Contents Based on Sentiment Analysis and Prediction System” paper. The aforementioned research discussed the enhanced performance of deep learning algorithms as well as BERT and DistilBERT models for accuracy, recall, precision, and F1

score, which put my results into context and speak to the reproducibility of these results for future sentiment classification tasks for tweets / social media.

In the future, I would employ a similar approach to the "Social Media Sentiment Analysis: Lexicon versus Machine Learning paper, since its insights into positive or negative sentiment nuances can be adapted to understand linguistic challenges in distinguishing between disaster and non-disaster tweets (e.g., context, sarcasm, or emotive expressions), which is something my model struggled with.

9 Acknowledgements

ChatGPT was used as a starting point for the models, which I subsequently modified greatly to work specifically for the Twitter Disaster dataset.

References

[1] "New Research Analyzes Millions of Twitter Posts during Hurricanes to Understand How People Communicate in a Disaster." *Carnegie Mellon University's Heinz College*, www.heinz.cmu.edu/media/2021/September/new-research-analyzes-millions-of-twitter-posts-during-hurricanes-to-understand-how-people-communicate-in-a-disaster. Accessed 5 Dec. 2024.

[2] Tiwari, Shikha, et al. "Social Media Sentiment Analysis On Twitter Datasets." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2020, pp. 925–27. DOI.org (Crossref), https://doi.org/10.1109/ICACCS48705.2020.9074208.

[3] Chaurasia, Suhashini & Sherekar, Swati & Thakare, V. M.. (2021). Twitter Sentiment Analysis using Natural Language Processing. 1-5. 10.1109/ICCICA52458.2021.9697136.

[4] Sanh, Victor, et al. *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter*. arXiv:1910.01108, arXiv, 1 Mar. 2020. *arXiv.org*, https://doi.org/10.48550/arXiv.1910.01108.

[5] SoYeop Yoo, et al. "Social Media Contents Based Sentiment Analysis and Prediction System." *Expert Systems with Applications*, Pergamon, 28 Mar. 2018, www.sciencedirect.com/science/article/pii/S0957417418302124.

[6] Chedia Dhaoui, Cynthia M. Webster, Lay Peng Tan, (2017) "Social media sentiment analysis: lexicon versus machine learning", Journal of Consumer Marketing, Vol. 34 Issue: 6, pp.480-488, https://doi.org/10.1108/JCM-03-2017-2141