We downloaded the data from Kaggle to Google Colab using the Kaggle API. Our data came from the Kaggle competition called “Natural Language Processing with Disaster Tweets,” and the goal of the competition was to predict whether Tweets are about real disasters. We made two changes to our coding environment: mounting our Google Drive files and changing the runtime to GPUs. The downloaded data came as a zipped folder, which we unzipped to find testing and training datasets. We converted the files to data frames for our EDA.

We found that the training data consists of five columns: id, keyword, location, text, and target. There are 7,613 tweets in the training dataset, and each tweet has a unique ID in the id column. The target column is a binary column only available in the training dataset where [1] corresponds to a disaster tweet and a [0] corresponds to a non-disaster tweet. The training set included 4,342 non-disaster tweets and 3,271 disaster tweets. Each tweet may have any number of 222 unique keywords (derailment, wreckage, etc.), and many tweets also include a unique location. Figures with the top locations and keywords appear in the index. The text column contains the contents of each tweet. The length of each tweet ranged from 7 characters to 157 characters, with the average tweet being 101 characters and a standard deviation being 34 characters. (It’s worth noting that there is a spike of tweets with around 140 characters, as demonstrated by the histogram in the index. In comparison, the number of words per tweet is much more normally distributed with an average of 15 and a standard deviation of six words.

We processed the text columns with a variety of data cleaning steps. We removed punctuation marks and converted common abbreviations and symbols to their text equivalents ($ to dollar; b4 to before; etc.). We also made changes such as removing non-printable characters, HTML beacons, and emojis as well as replacing URLs, mentions (@), numbers and some figures (such as <3) with text equivalents. Finally, we tokenize the cleaned text data,

which consists of converting the text to numerical values, before splitting into a training set and a testing set using an 80%-20% split.

We created six long short-term memory (LSTM) Models through a 3x2 completely crossed design. We adjusted the LSTM units for each model by changing the number of units (128, 256, 512) and batch sizes (32 or 64). Our first model, which had 128 units and a batch size of 32, performed best on the testing subset of our training data; it received an accuracy score of 0.777. However, the worst of the LSTM Models (Model 2) still received an accuracy score of 0.747, so the parameters we changed didn't seem to make a major impact. Model 1 received a Kaggle score of 0.765. Our full LSTM model results appear in the index.

In addition to the LSTM Models, implemented Google's BERT for our disaster natural language processing. BERT stands for (Bidirectional Encoder Representations from Transformers), and the mode differs from RNN based models as it's a bidirectional model, meaning it looks at both sides of a token. The BERT model has lots of uses, but can be tuned by adding a layer to the model. We're trying to build a classification model, so that is the layer that we add to our model. The code used in the BERT Model section is based on a relatively popular Kernel solution, though numerous aspects needed updating to get the code running. While we could not manipulate the model to create a train-test split and get the goodness of fit metrics, our BERT model's predictions received a Kaggle score of 0.831.

We were happy with our results, as both of our Kaggle scores were relatively high and we understand where we would go next with this project. Our next steps for the LSTM Models would be testing additional parameter changes (or packages) to see if we could further improve our results. We would also like to explore additional modifications and applications for Google's BERT model, as our current BERT code doesn't interact well with the rest of our LSTM models.

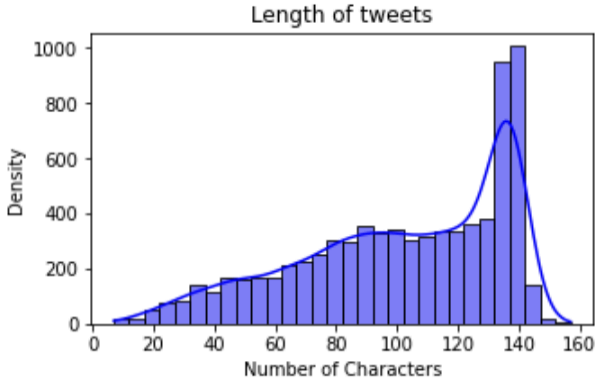
Index:

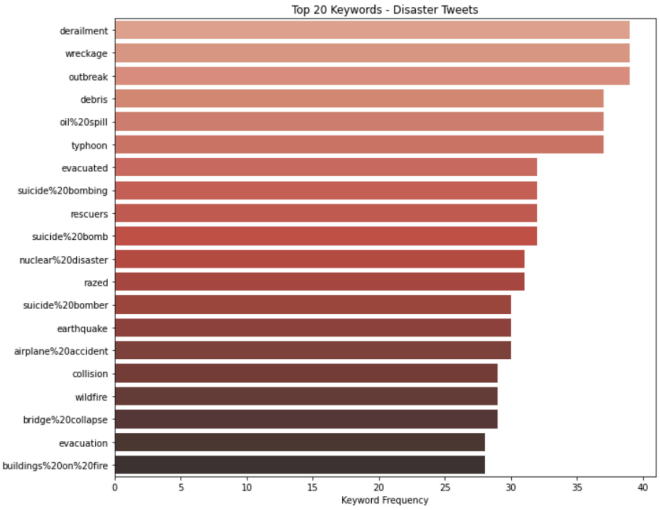
LSTM Model:

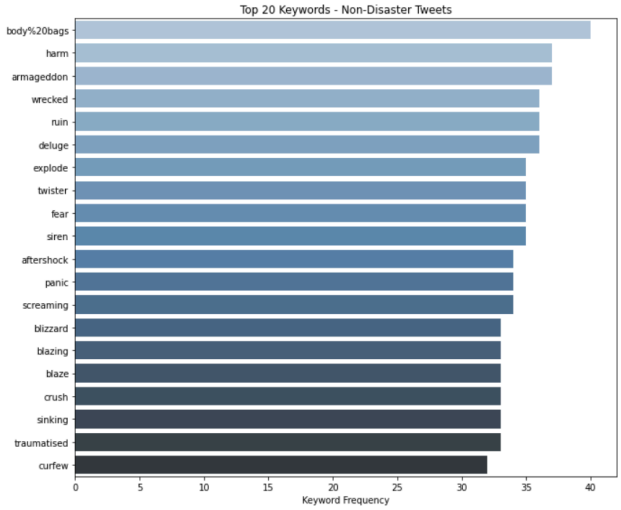
BERT Model:

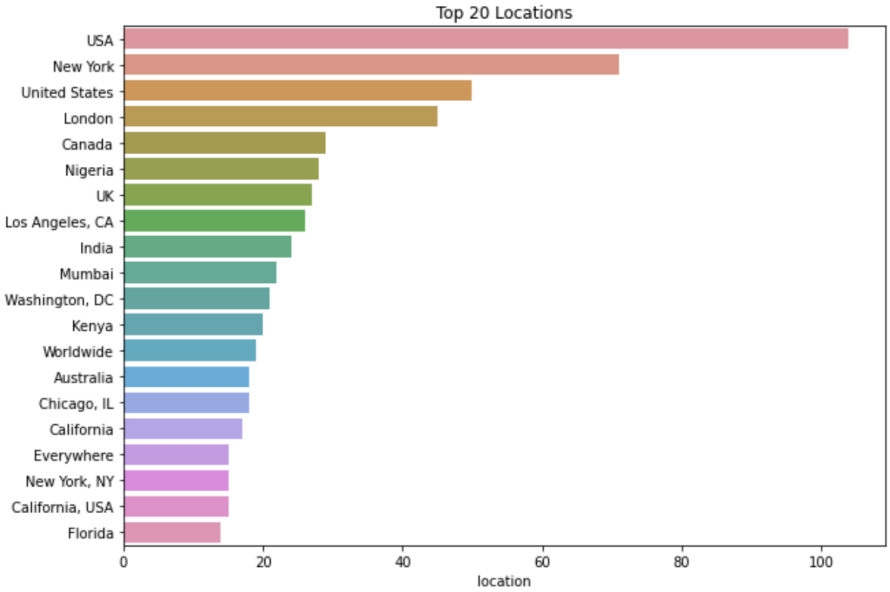
LSTM Model Goodness of Fit Tests:

	Model	Units	Batch Size	Epochs	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy
0	LSTM Model 1	128	32	10	0.130170	0.953859	0.637036	0.776756
1	LSTM Model 2	256	32	10	0.136804	0.953038	0.670723	0.746553
2	LSTM Model 3	512	32	10	0.116587	0.961248	0.679498	0.766251
3	LSTM Model 4	128	64	10	0.150015	0.950575	0.600004	0.774787
4	LSTM Model 5	256	64	10	0.130680	0.954680	0.615145	0.764281
5	LSTM Model 6	512	64	10	0.127265	0.960263	0.634791	0.762311


Length of tweets
Density
Number of Characters
0
200
400
600
800
1000
0
20
40
60
80
100
120
140
160


Top 20 Keywords - Disaster Tweets
derailment
wreckage
outbreak
debris
oil%20spill
typhoon
evacuated
suicide%20bombing
rescuers
suicide%20bomb
nuclear%20disaster
razed
suicide%20bomber
earthquake
airplane%20accident
collision
wildfire
bridge%20collapse
evacuation
buildings%20on%20fire
Keyword Frequency


Top 20 Keywords - Non-Disaster Tweets
body%20bags
harm
armageddon
wrecked
ruin
deluge
explode
twister
fear
siren
aftershock
panic
screaming
blizzard
blazing
blaze
crush
sinking
traumatised
curfew
Keyword Frequency


Top 20 Locations
USA
New York
United States
London
Canada
Nigeria
UK
Los Angeles, CA
India
Mumbai
Washington, DC
Kenya
Worldwide
Australia
Chicago, IL
California
Everywhere
New York, NY
California, USA
Florida
location