



Analyzing and Predicting Tweets to Identify Disasters

Venkatesh Murugesh

Robin Sharma

Rohan Kamath

Rohan Bhowmick

Parikshit Urs

Kshitija Landge

Rutvij Patil

INTRODUCTION

- In times of **emergency**, social media platforms have emerged to be vital communication tools. One such important platform is **Twitter**.
- Since smartphones allow users to **tweet** about any emergency in **real time**, organizations like news agencies and disaster relief agencies are constantly **keeping an eye** on these tweets.
- There is a need to **segregate real and fake** disaster tweets to take prompt action in response to the disaster.
- The goal is to develop a system that can **predict** whether a tweet is related to a real disaster or not.

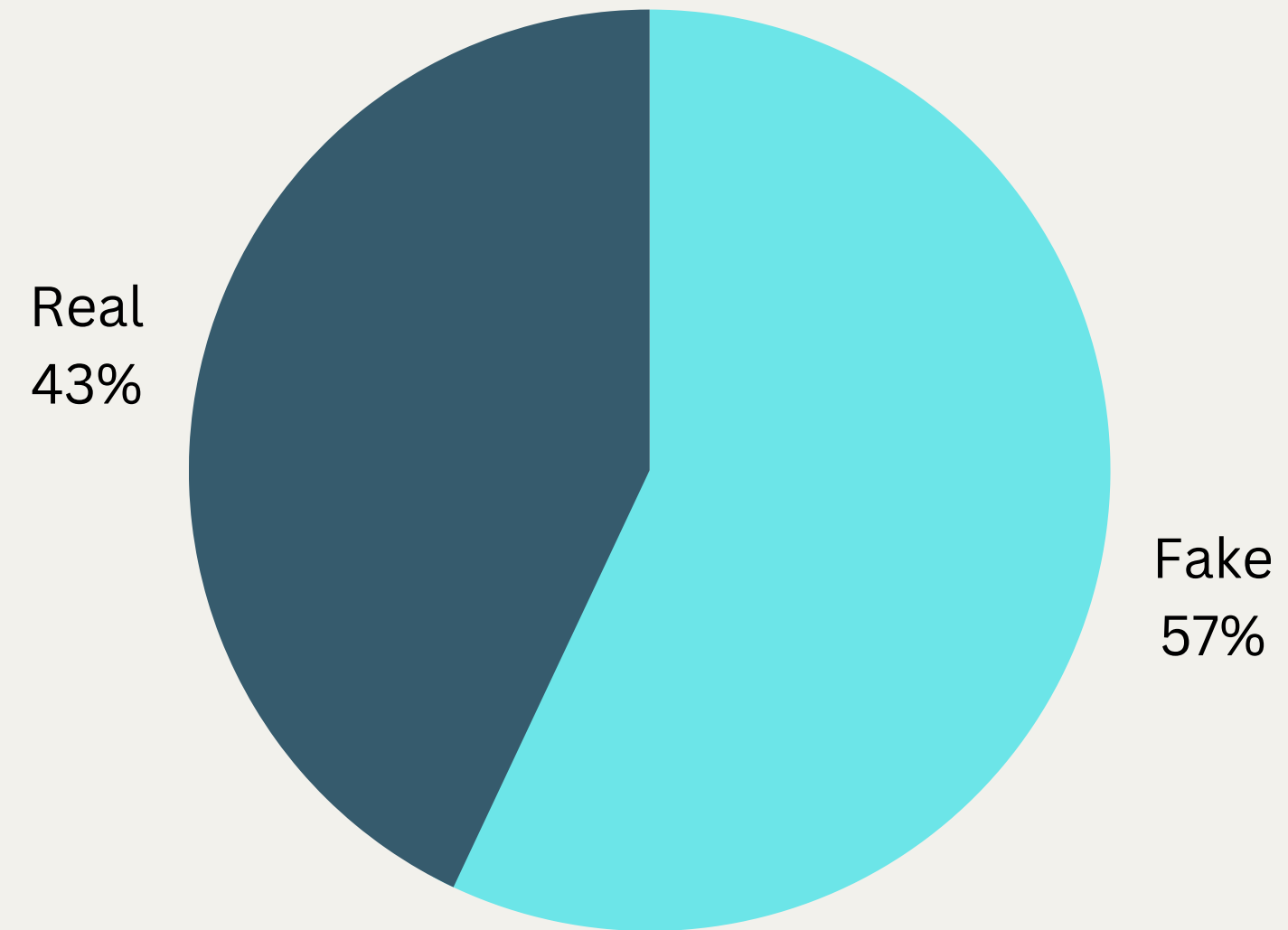


ABOUT THE DATASET

train.csv - contains 7000+ tweets

test.csv - contains 3000+ tweets

Fig 1: Training Tweet Class Distribution



Attributes of the dataset

Text

The text of the tweet

Location

The location where the tweet was sent from

Keyword

Contains a disaster specific word

Target

denotes whether a tweet about disaster is real or not (1/0)

ABOUT THE DATASET

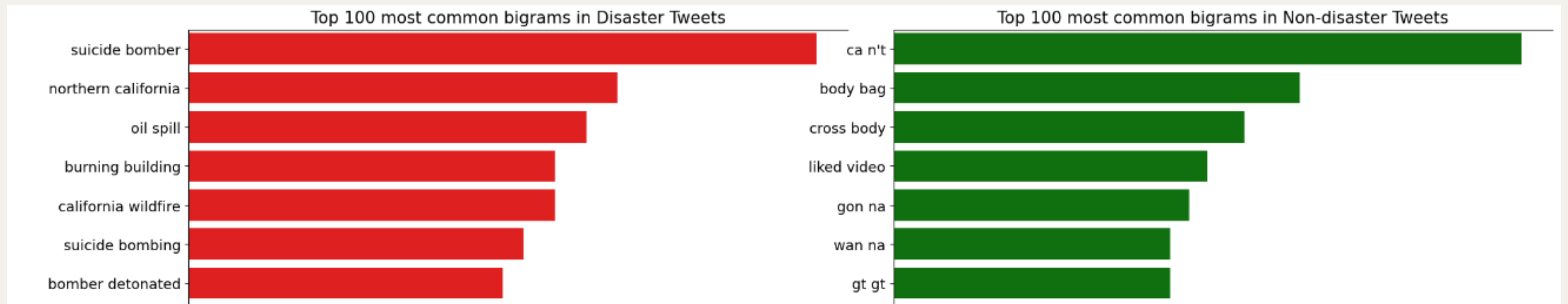


Fig. 2: Top 100 Bigrams in Disaster and Non-Disaster Tweets

ABOUT THE DATASET

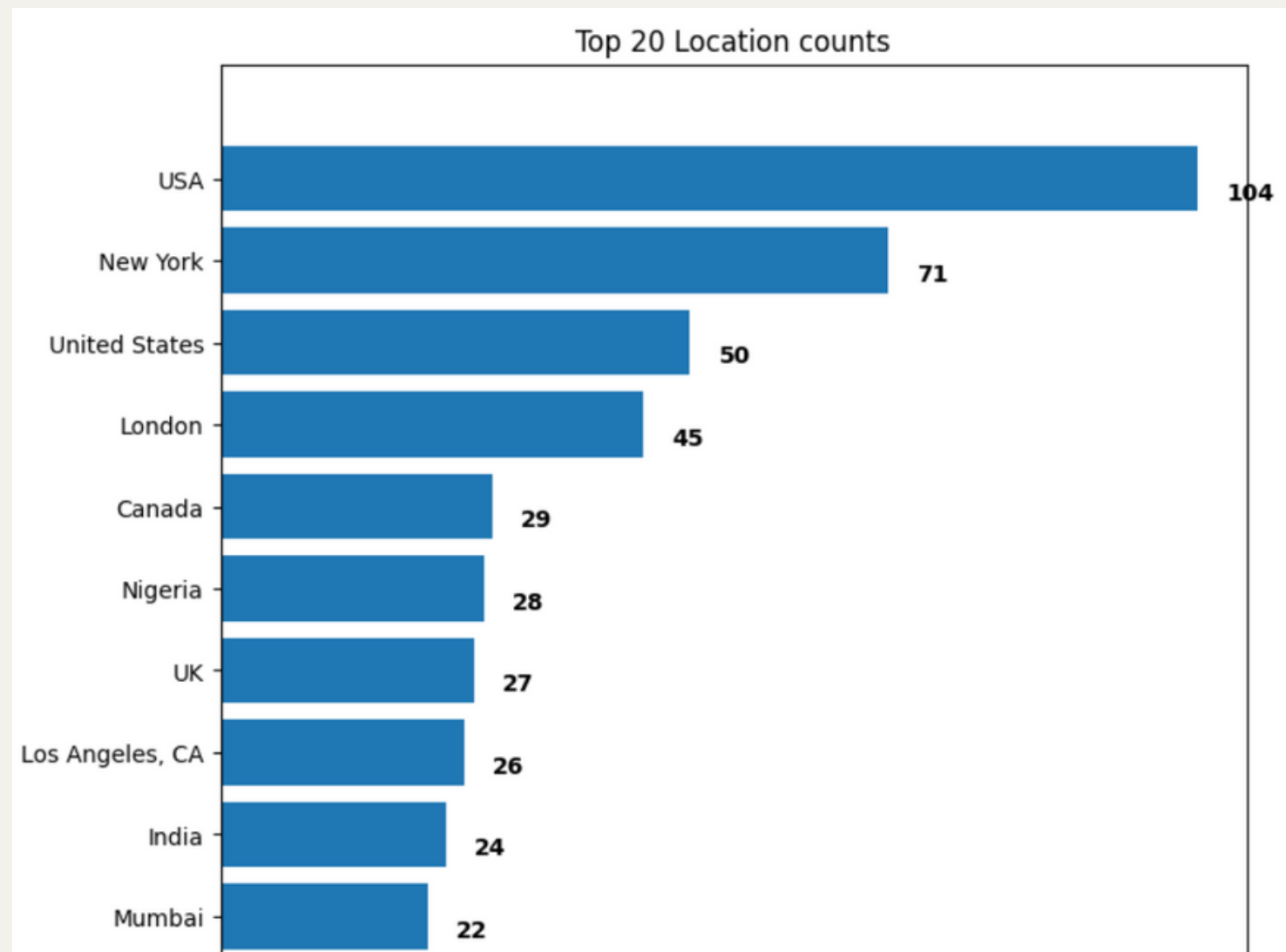


Fig. 3: Top 20 Location Counts

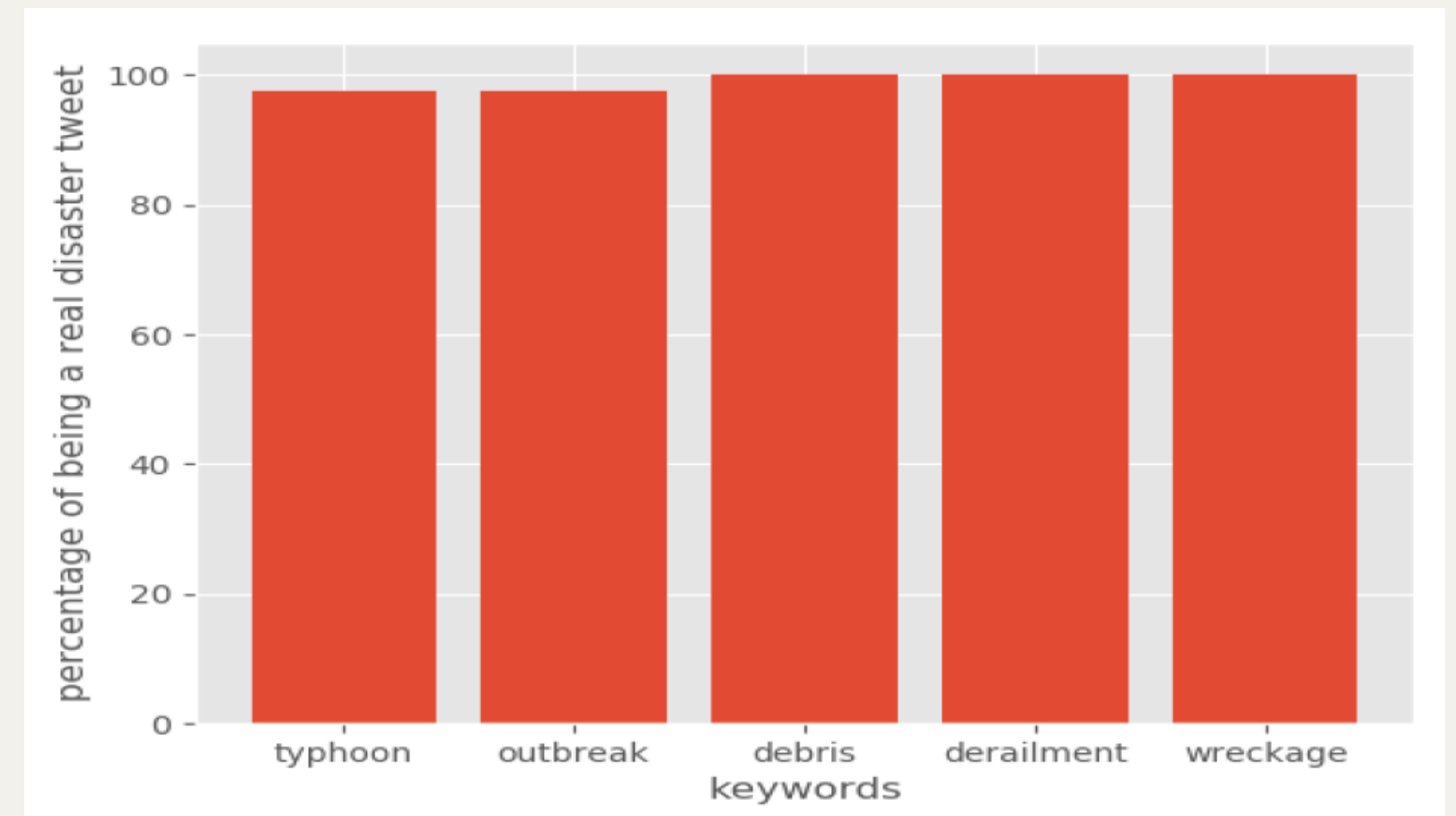


Fig. 4: Keywords having maximum real disaster tweets

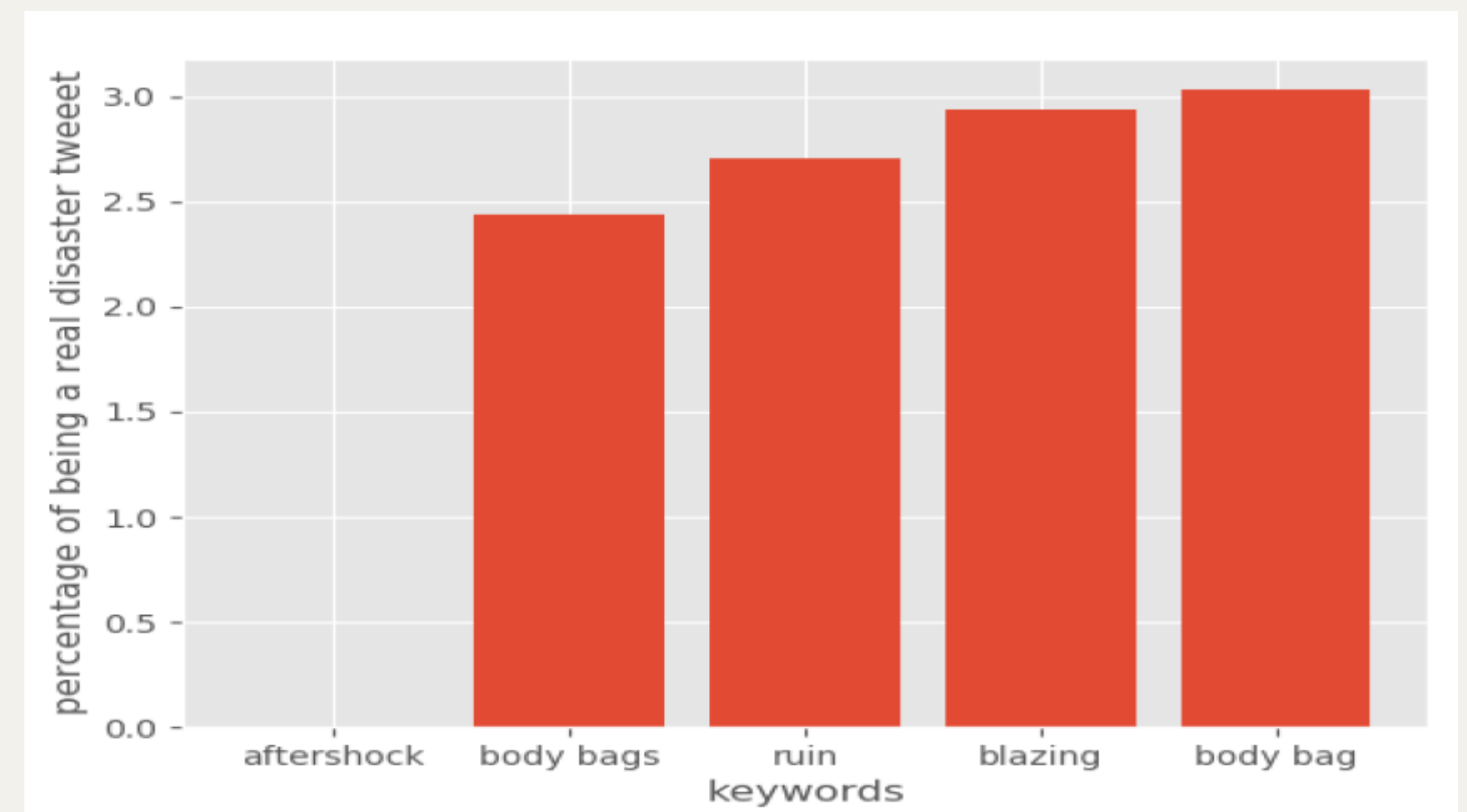
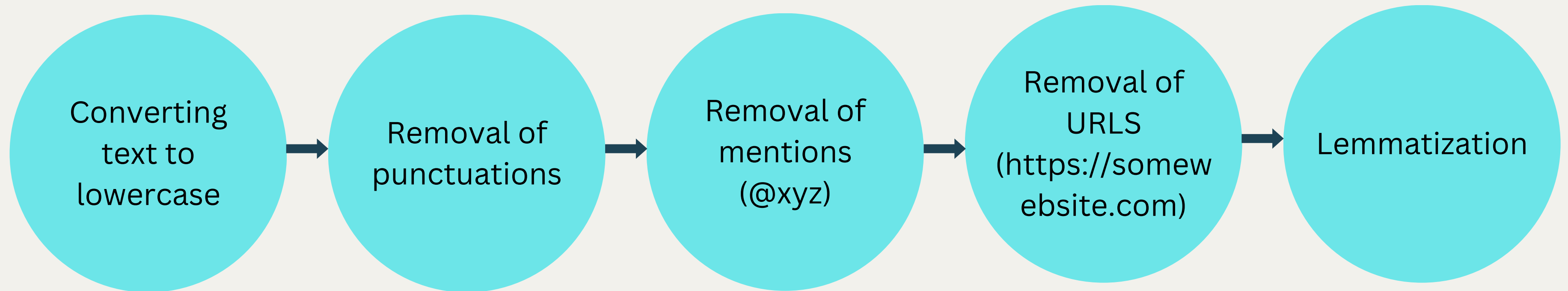


Fig. 5: Keywords having least real disaster tweets

METHODOLOGY

A) DATA PREPROCESSING



B) TEXT TO VECTOR CONVERSION

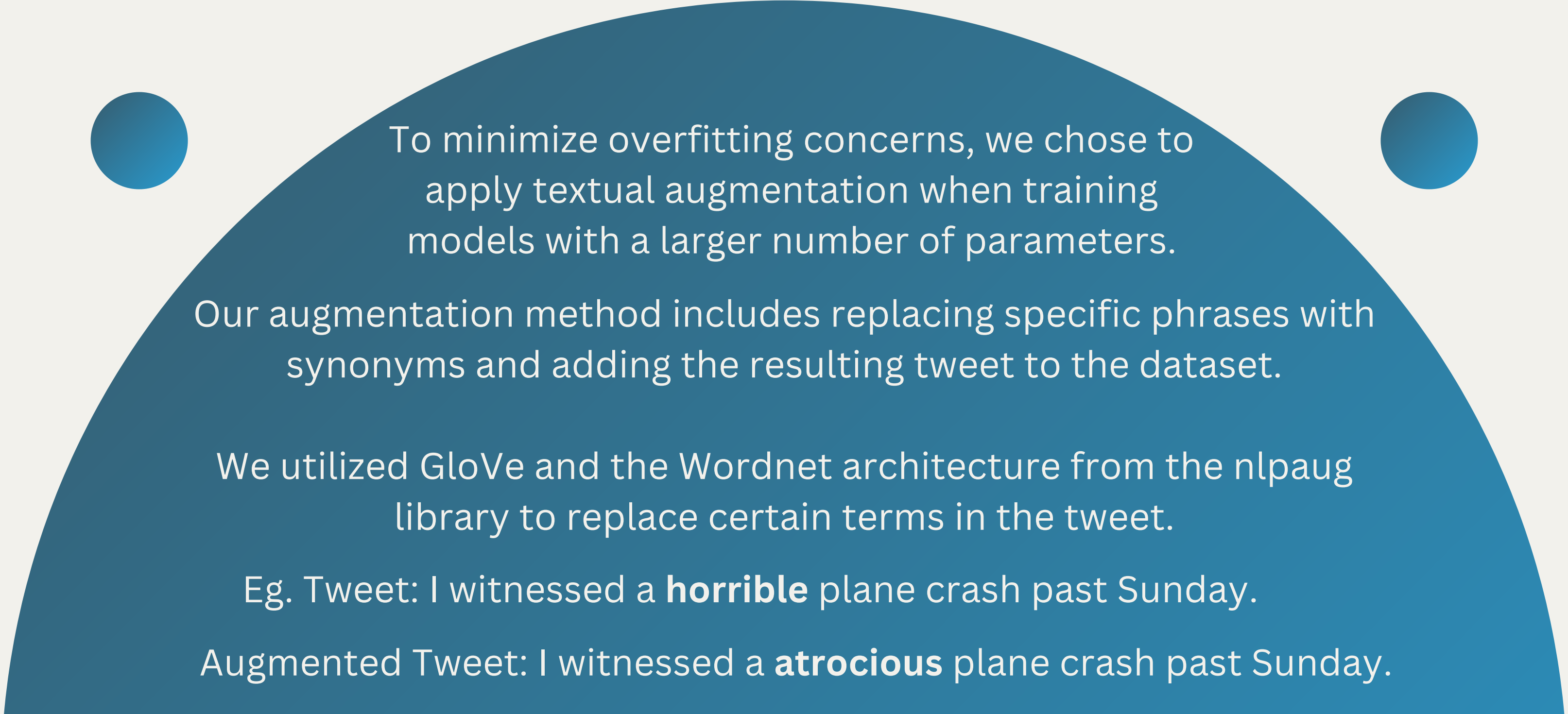
- Count Vectorization
- TF-IDF vectorization
- N-gram modelling
- GloVe embeddings

C) CLASSIFICATION MODELS

- Traditional ML Algorithms
- LSTM Network



DATA AUGMENTATION



To minimize overfitting concerns, we chose to apply textual augmentation when training models with a larger number of parameters.

Our augmentation method includes replacing specific phrases with synonyms and adding the resulting tweet to the dataset.

We utilized GloVe and the Wordnet architecture from the nlpaug library to replace certain terms in the tweet.

Eg. Tweet: I witnessed a **horrible** plane crash past Sunday.

Augmented Tweet: I witnessed a **atrocious** plane crash past Sunday.




APPROACH 1: ML ALGORITHMS

Sr. No.	Experiment
01	CountVec + Traditional ML Algorithms
02	TF-IDF + Traditional ML Algorithms
03	TF-IDF with N-gram modelling + Traditional ML Algorithms

- Logistic Regression
- Naive Bayes
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)






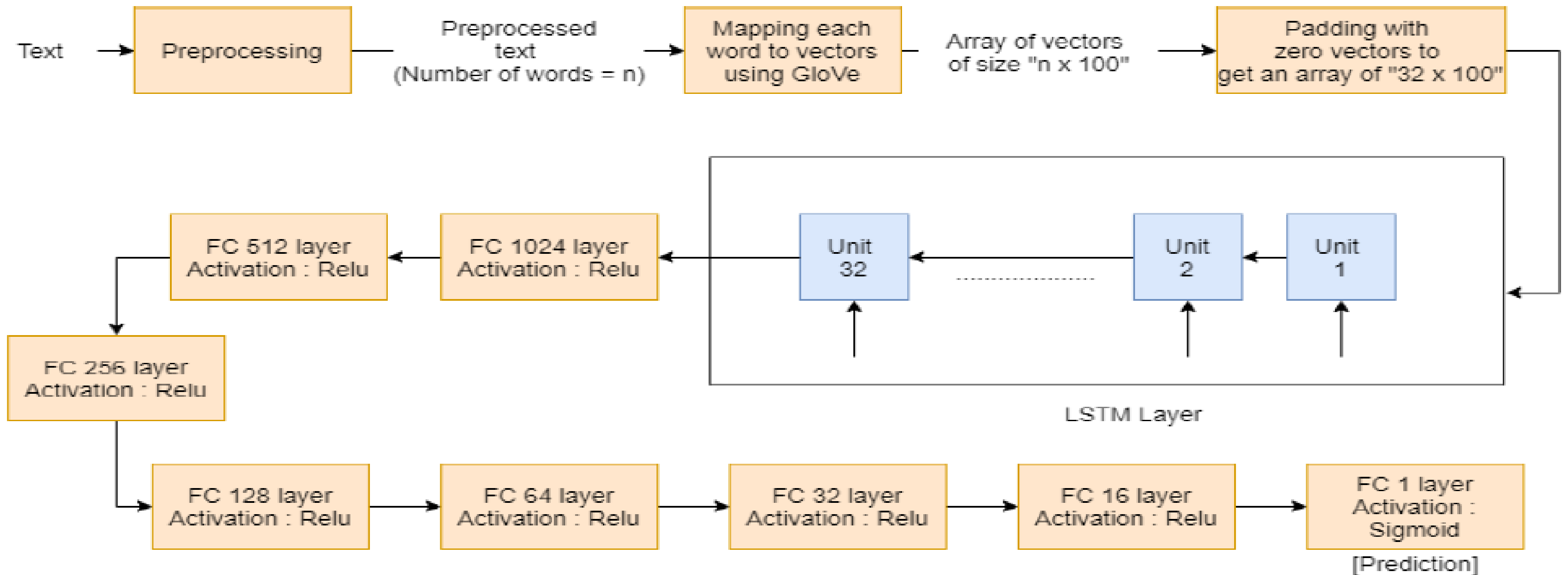
APPROACH 2: GLOVE EMBEDDING + LSTM MODEL + WEIGHTED PROBABILITY SCORE

- We used the LSTM network with pre-trained GloVe embeddings to determine the probability of a true disaster tweet for the test data.
 - Then, for each tweet with an associated keyword, we took the average of the probability estimated by the LSTM network.
 - And the probability acquired by dividing the proportion of tweets that were actually for the keyword by the percentage of tweets that were real for the keyword.
- 
- 
- 



APPROACH 3: LSTM MODEL + GLOVE EMBEDDING

- We used a pre-trained GloVe embedding layer from Twitter to represent each phrase as a 100-dimensional vector.
 - The maximum length of a tweet was set at 32 characters. Tweets with more than 32 characters were truncated at the conclusion, while those with fewer characters were padded with zero vectors were fed into LSTM layer.
 - The output from these cells was routed through dense levels, with the last layer producing the chance of a tweet being genuine.
- 
- 
- 



Final Model Architecture: GloVe Embedding with LSTM



RESULTS

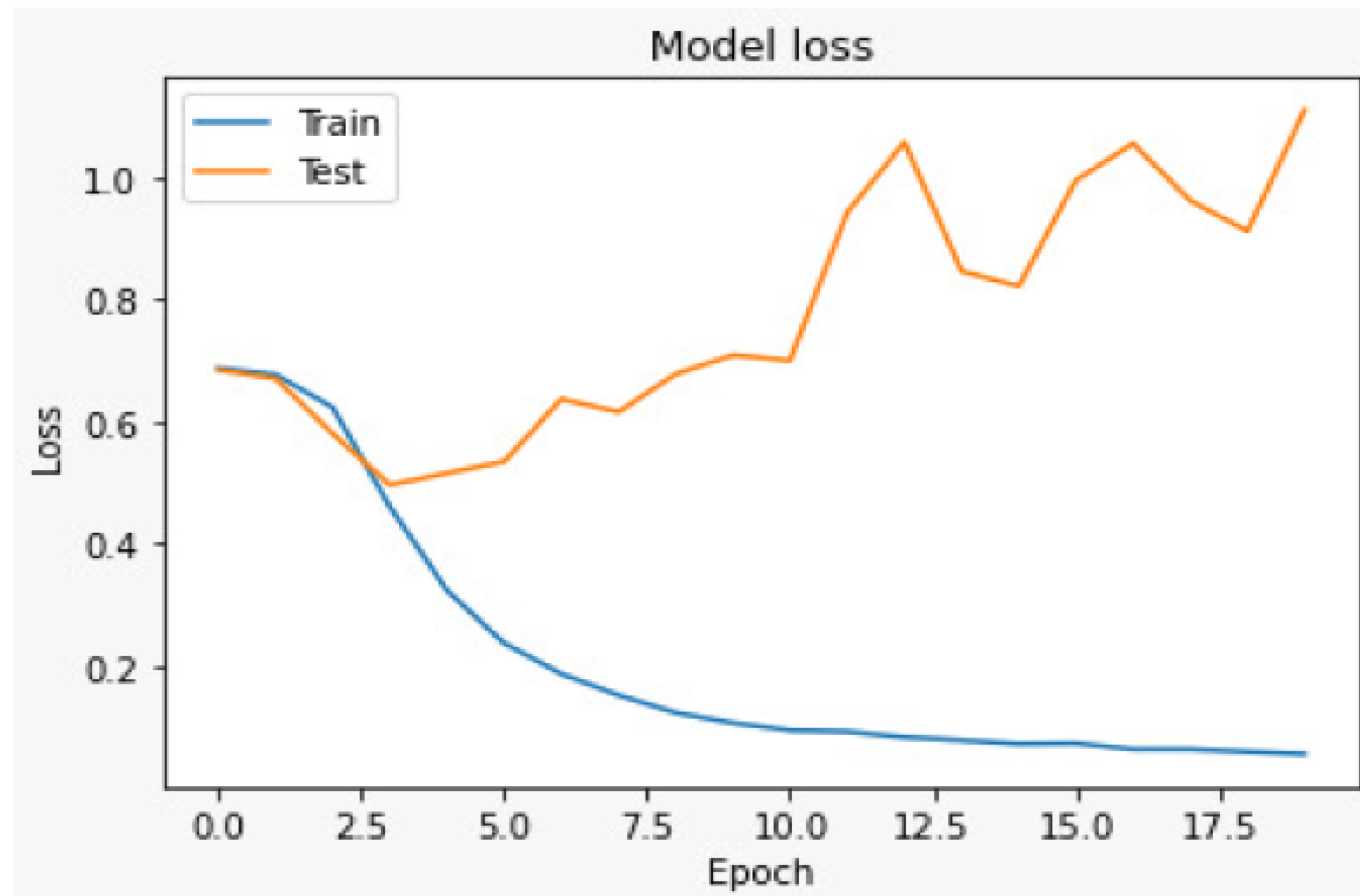
ML Algorithms

Technique	Best Performance Model	Accuracy
CountVec	Multinomial Naive Bayes	0.759
TF-IDF	Multinomial Naive Bayes	0.744
TF-IDF with N-Gram	Logistic Regression	0.737

RESULTS

Deep Learning Models:

Trainable Embedding Layer + LSTM: Observing the train/test loss curve, we discovered that overfitting was caused by a lack of data for training both the embedding layer and the LSTM model.



RESULTS



Deep Learning Models

Pre-trained Glove Embedding + LSTM

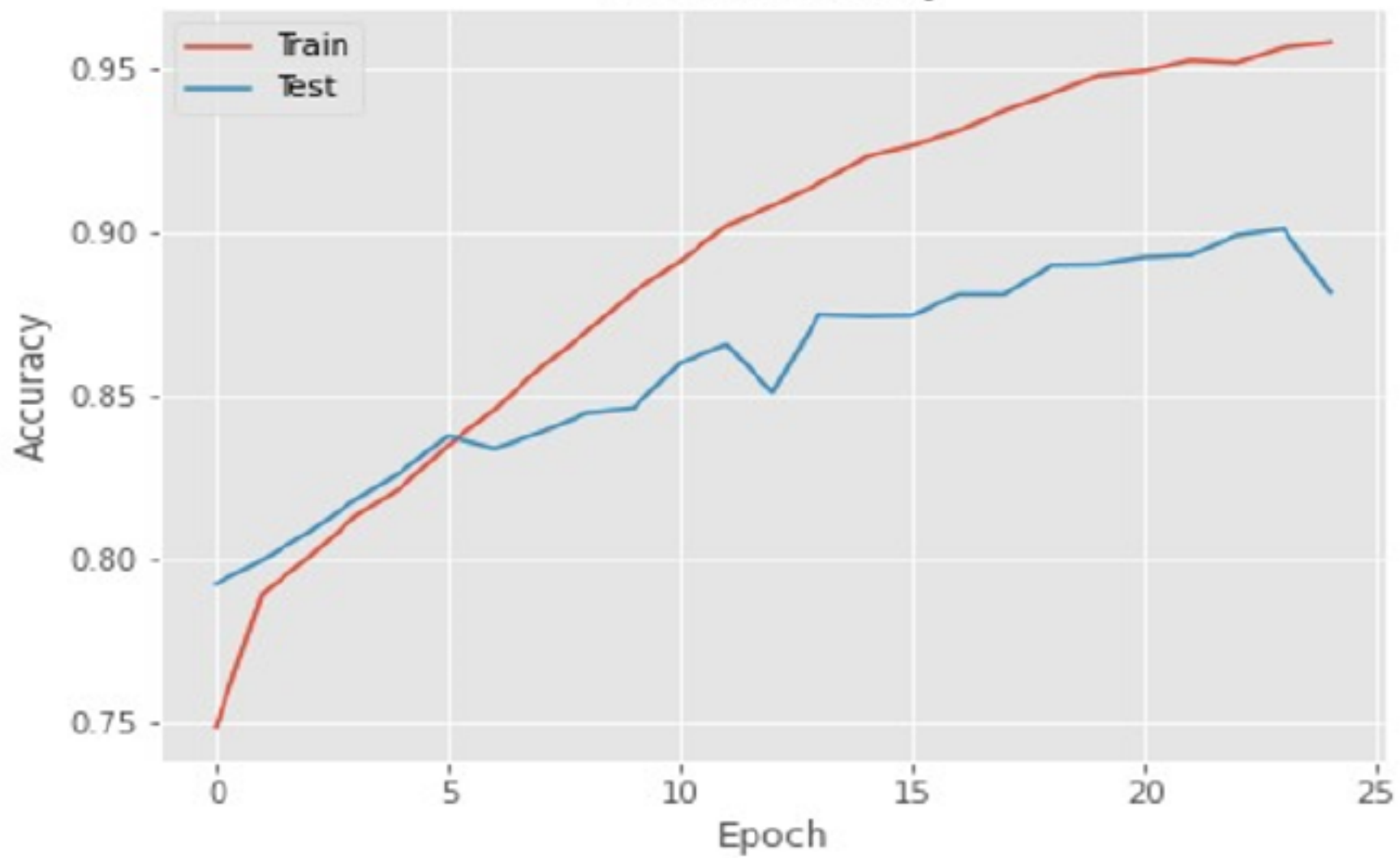
The mean F1 score for this embedding model with Twitter data was 0.832.

- These are the parameters that performed the best:
- Batch size = 64
- Number of Epochs = 25
- Dropout Rate for Fully connected layers = 0.2
- Learning rate = 0.01
- Optimizer = Adam
- Loss Function = Binary Cross Entropy

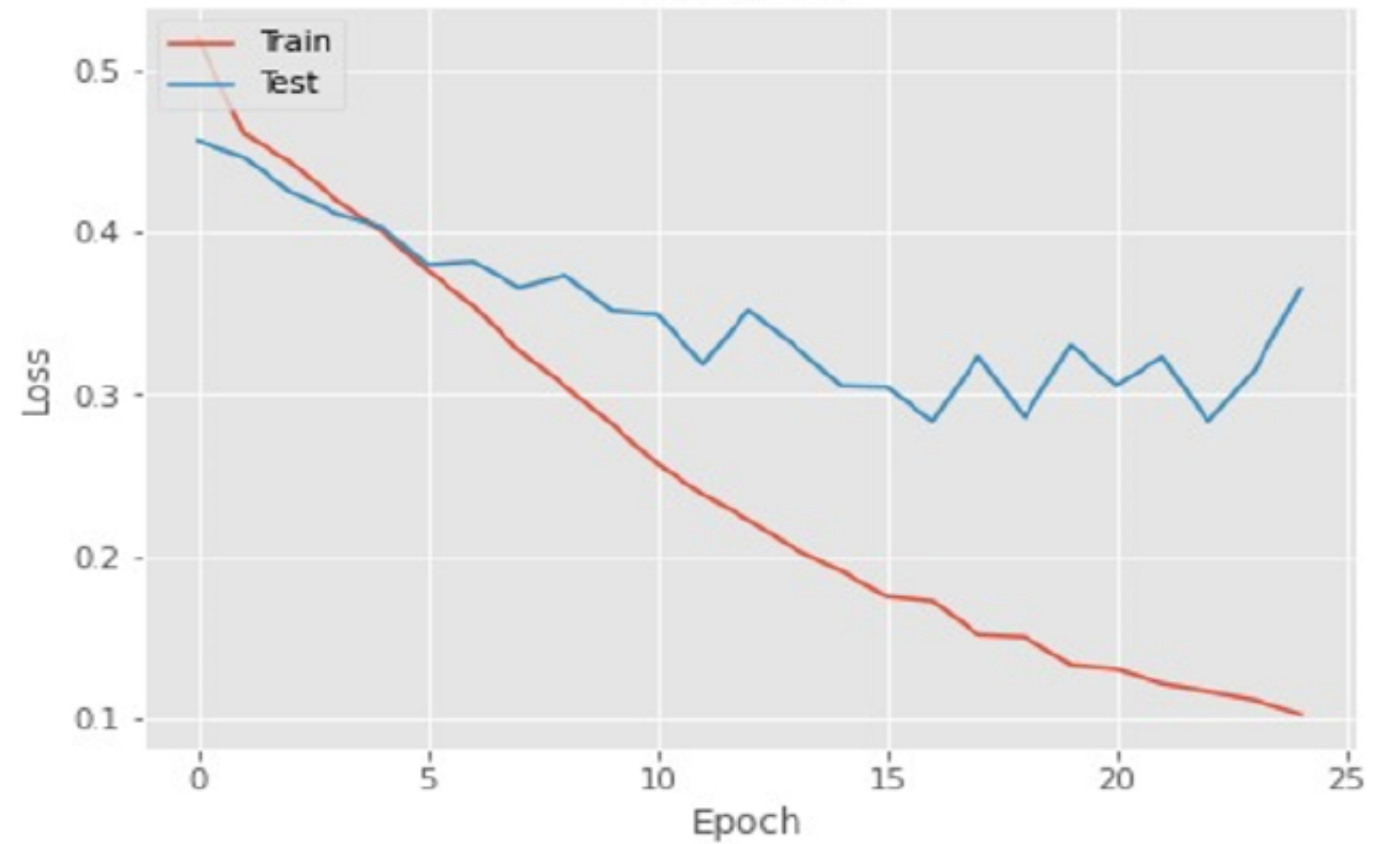
RESULTS

Final model accuracy and loss curves:

Model accuracy



Model loss





RESULTS

Weighted Probability Scores:

The following modifications were noticed for predictions on the test dataset when using weighted probabilities:

- 98 tweets had their class altered from real to fraudulent.
- 117 tweets had their class modified from fraudulent to authentic.

A mean F1 score of 0.830 was obtained.



RESULTS

Comparison of Deep Learning approaches

Architecture	Mean F1 scores on Test data
Glove + LSTM	0.832
Weighted Probability Score	0.830



THANK YOU!