CLASSIFICATION OF PROBABLE DISASTER TWEETS

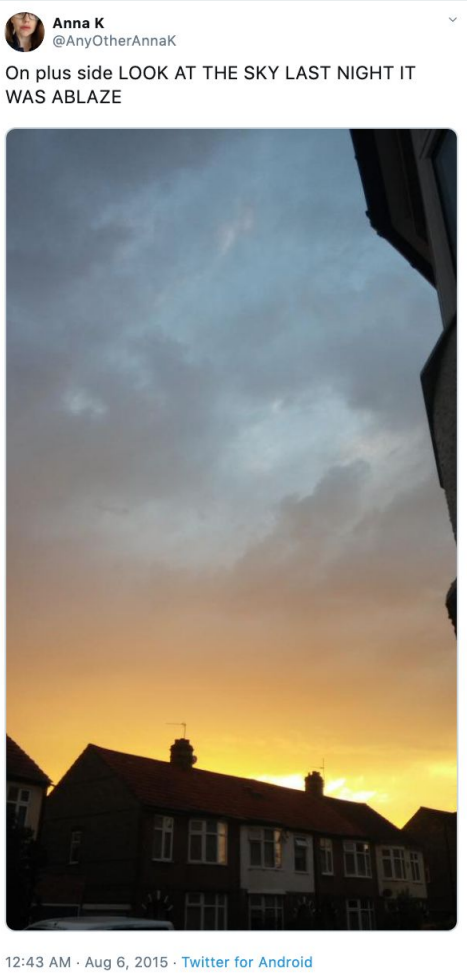
Neeraj Adhikari
Archana Chittoor

Introduction

- Twitter has become a key communication channel.
- Many agencies monitor Twitter to detect disasters early.

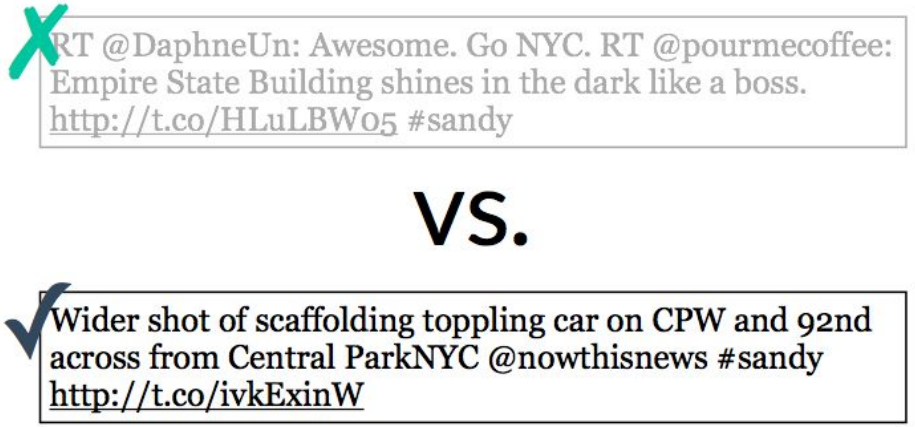
Introduction

- Tweet describes the sky as ‘ABLAZE’ .
- Could be mistaken for a real disaster.
- Some mechanism needed to verify if tweet refers to an actual disaster.
- Machine learning can discriminate real disaster tweets with an acceptable level of reliability.



Objective

To classify tweets into disaster and non-disaster tweets.



Data Description

Dataset consists of 10000 tweets, divided into training and test data.
Features in **training** data:

- **Id** - unique identifier for each tweet
- **Text** - the text of the tweet
- **Location** - Location the tweet was sent from
- **Keyword** - a particular keyword from the tweet
- **Target** - 1 for a real disaster tweet, 0 if not

Features in **test** data: All the above except Target

Preprocessing

- **Data Cleaning:**
  Remove URLs, username mentions, mis-encoded characters and split condensed hashtags using **wordninja[3].**

- **Tokenization and Lemmatization:**
  Use **SpaCy** library[2] to tokenize, lemmatize and eliminate stop-words.

Preprocessing

- **Vectorization**
  Perform feature extraction by converting to 300-dimensional word vectors using **GloVe [4]**, trained on Common Crawl English corpus.

Methodology

1. Multilayer Perceptron (MLP)
2. K Nearest Neighbors
3. Convolutional Neural Network
4. Long Short-Term Memory Network (LSTM)
5. Model Ensemble

Multilayer Perceptron (MLP)

- Two hidden layers of size 500 and 50 each
- Default settings: ReLU activations, Adam optimizer, initial learning rate 0.001
- 5-fold cross-validation

Average cross-validation F1 score on training set: **68.37%**

K Nearest Neighbors (KNN)

- Implemented as a baseline for performance
- Used Scikit-learn library
- K-value 7 provided reasonable results

Average cross-validation F1 score on training set: **71.1%**

Convolutional Neural Network (CNN)

- Implemented in PyTorch
- 1-dimensional convolutions (length of sentence)
- 300-dimensional word vector as input activations in 300 different channels
- Adam optimizer
- Batch gradient descent

Convolutional Neural Network (CNN)

- Size-5 convolutional layer with 300 input channels, 4 output channels and ReLU activation
- Size-3 convolutional layer with 4 input channels, 8 output channels and ReLU activation
- FC layer with 40 outputs and ReLU activation
- FC layer with 40 inputs, 2 outputs (for 2 classes) and Softmax activation

Average cross-validation F1 score on training set: **75.6%**

Long Short-Term Memory Network

- PyTorch implementation, Two Layers, both bidirectional
- Retains word history
- Composition:
  - Stacked LSTM cells with hidden and cell state size 300
  - FC with input size 300, output size 30, ReLU activation
  - FC with input size 30, output size 2, Softmax activation

Average cross-validation F1 score on training set: **75.2%**

Model Ensemble

- Uses previous models KNN, CNN and LSTM
- Each model equally weighted
- Overall performance improvement

Average cross-validation
F1 score on training set:
76.4%

Experiments

1. Selection of Filter sizes for CNN:

-	1	3	5	7
1	0.434	0.498	0.734	0.539
3	0.726	0.627	0.722	0.721
5	0.729	**0.749**	0.725	0.738
7	0.708	0.742	0.749	0.723

Experiments

2. Selection of Channel sizes for CNN:

-	2	4	6	8	10	12
2	0.722	0.521	0.700	0.619	0.470	0.600
4	0.704	0.739	0.736	**0.752**	0.746	0.747
6	0.526	0.742	0.728	0.729	0.742	0.748
8	0.723	0.743	0.705	0.729	0.749	0.728
10	0.720	0.725	0.730	0.731	0.741	0.741
12	0.733	0.723	0.740	0.747	0.719	0.749

Table 2. F1 scores for different values of layer 1 channel size (rows) and layer 2 channel size (columns).

Experiments

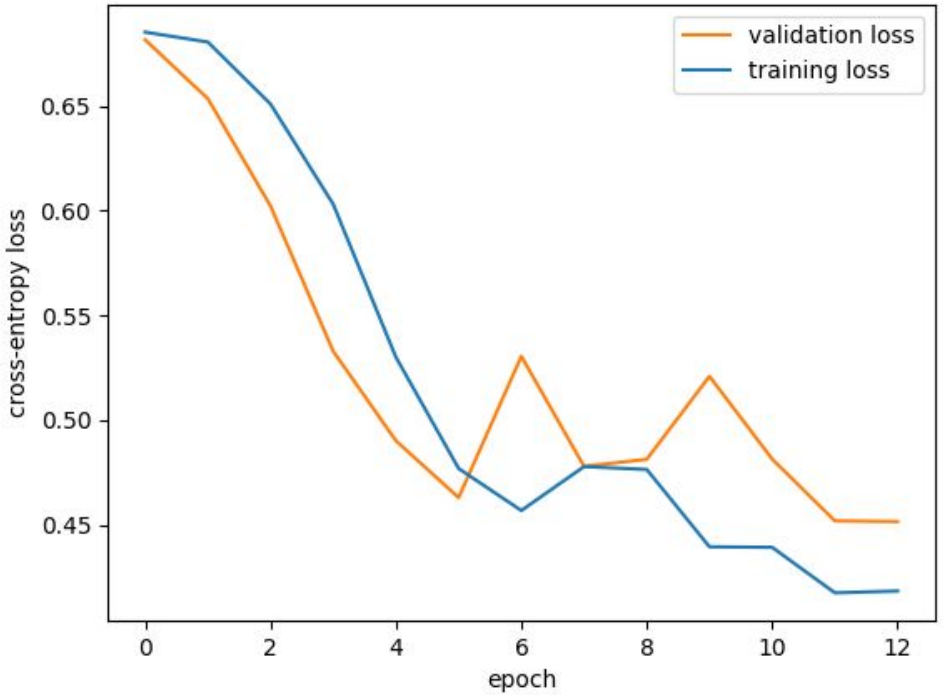
3. Selection of layer count and hidden state sizes on LSTM:

-	20	40	60	90	150	300
1	0.699	0.648	0.704	0.691	0.704	0.717
2	0.669	0.683	0.666	0.696	0.724	**0.735**

Table 3. F1 scores for different values of layer count (rows) and and hidden state dimension (columns).

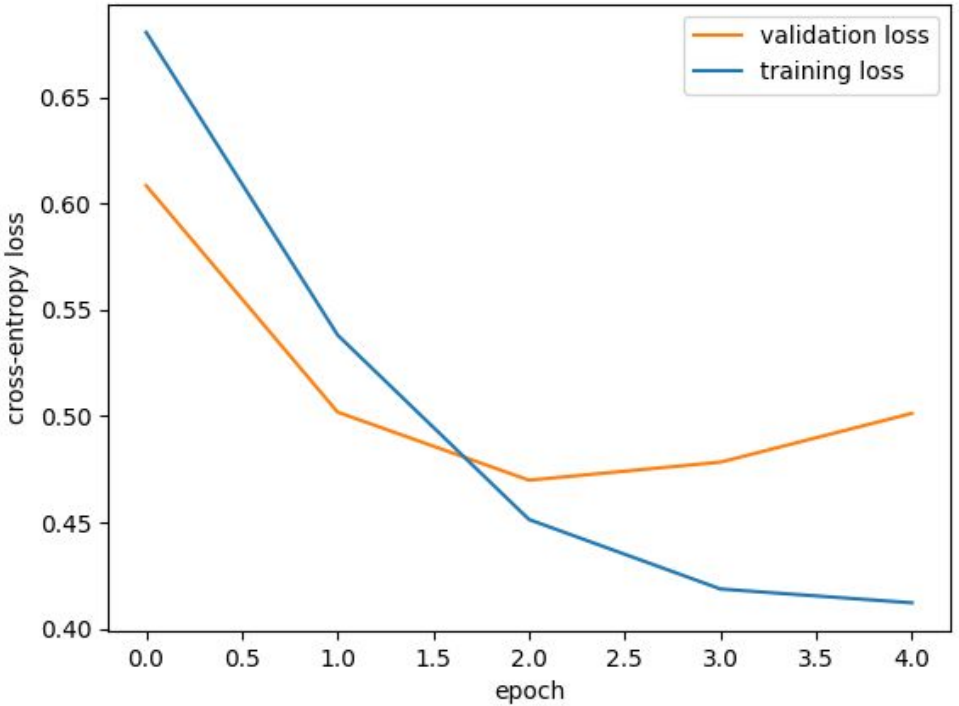
Analysis: Losses

CNN



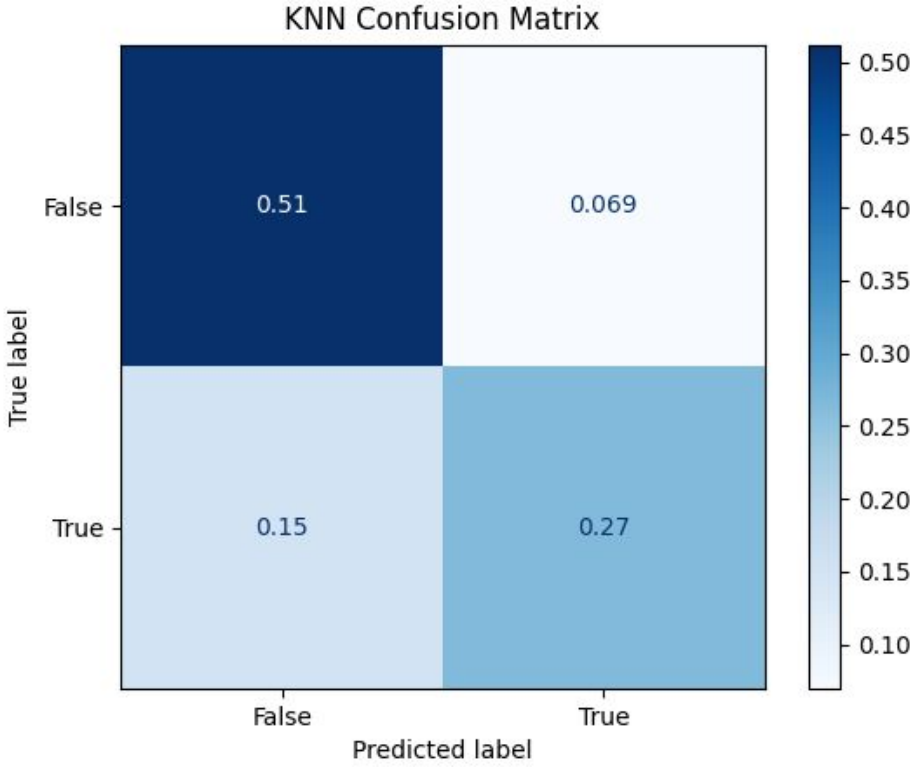
Analysis: Losses

LSTM



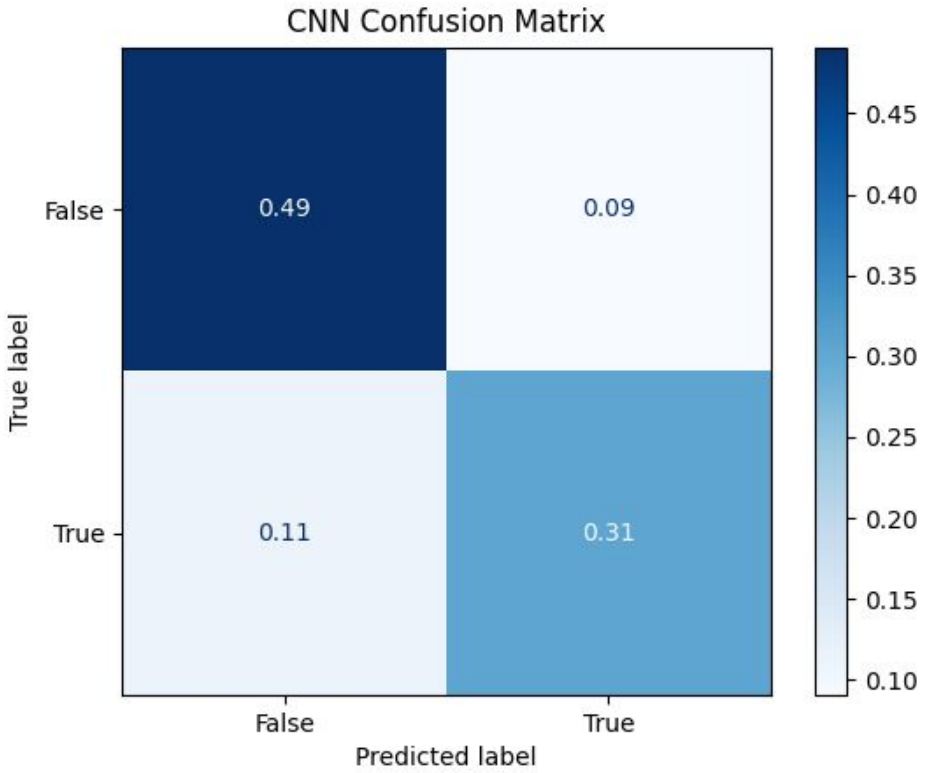
Analysis: Class Distribution of Predictions

KNN Classifier



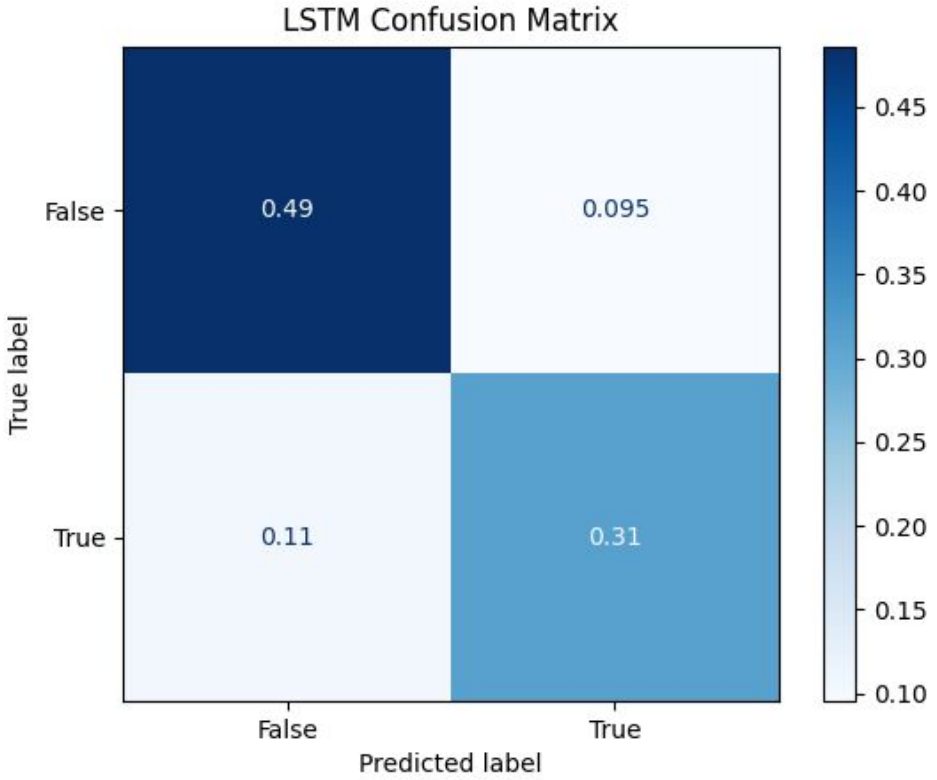
Analysis: Class Distribution of Predictions

CNN Classifier



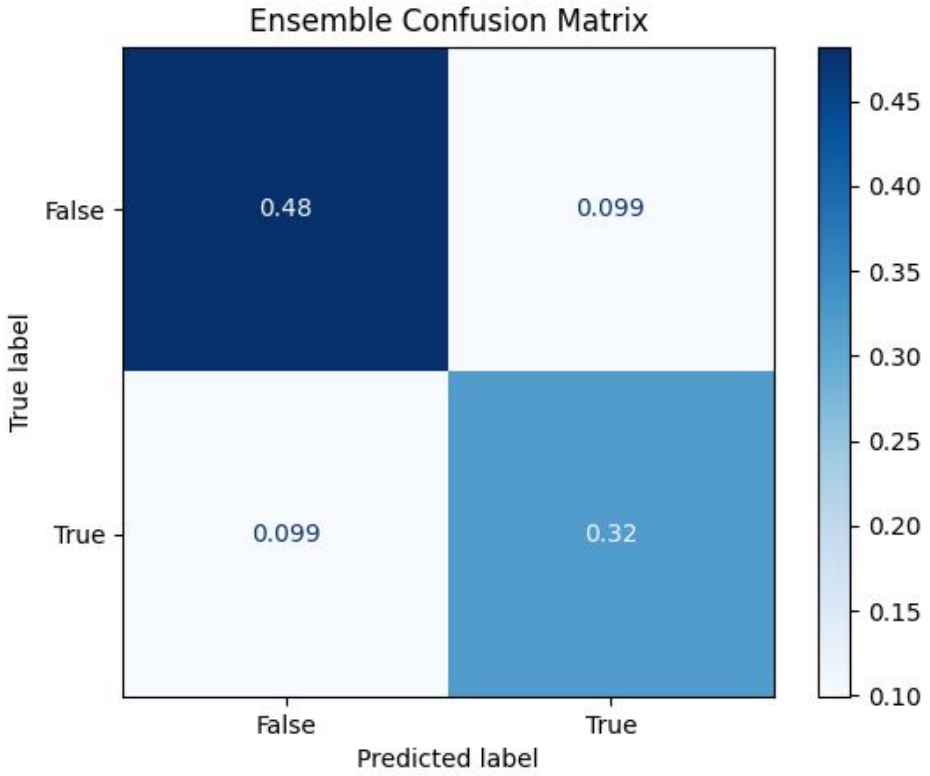
Analysis: Class Distribution of Predictions

LSTM Classifier



Analysis: Class Distribution of Predictions

Ensemble



Kaggle Score

Score of 0.79447

Conclusion

- LSTM gave a slightly lower score as compared to CNN.
- Model Ensemble with KNN, CNN and LSTM gave the highest accuracy, closely followed by the CNN model.

References

1. [n. d.]. Real or Not? NLP with Disaster Tweets. https://www.kaggle.com/c/nlp-getting-started/overview. ([n. d.]). accessed: 2020-04-06.
2. [n. d.]. SpaCy: Industrial Strength Natural Language Processing. https://spacy.io/. ([n. d.]). accessed: 2020-04-06.
3. [n. d.]. Word Ninja. https://github.com/keredson/wordninja. ([n. d.]). accessed: 2020-04-06.
4. Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global Vectors for Word Representation. EMNLP 14, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Questions

THANK YOU

Neeraj Adhikari
Archana Chittoor