Natural Language Processing with Disaster Tweets

XIU ZHEN HUANG
16.10.2025

Overview

- Brief Description of Dataset
- Data Cleaning
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Baseline Model
- RNN Model
- RNN Tuning
- Results and Analysis
- Conclusion
- Future Work

The full project locates at the git repository.

Brief Description of Dataset

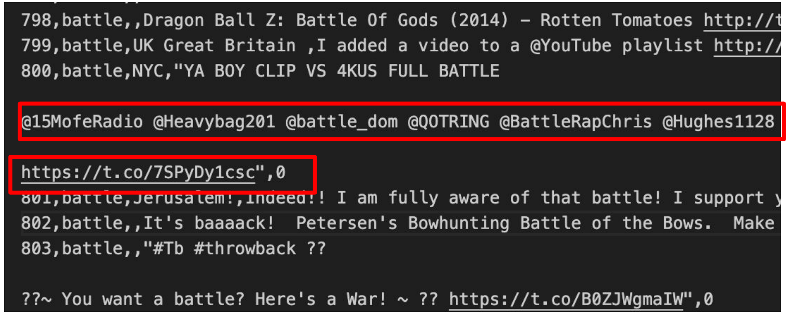
- This dataset is from Kaggle, it presents a text classification task in the field of **Natural Language Processing (NLP)**.
- It is a binary classification problem, where the goal is to determine **whether a given tweet describes a real disaster or not**.
- The dataset contains **7613 samples** and includes several missing values that require data cleaning prior to feature engineering.
- Both the train and test files contain **three input features — keyword, location, and text — and one target label**, where 1 indicates a disaster-related tweet and 0 indicates a non-disaster tweet.

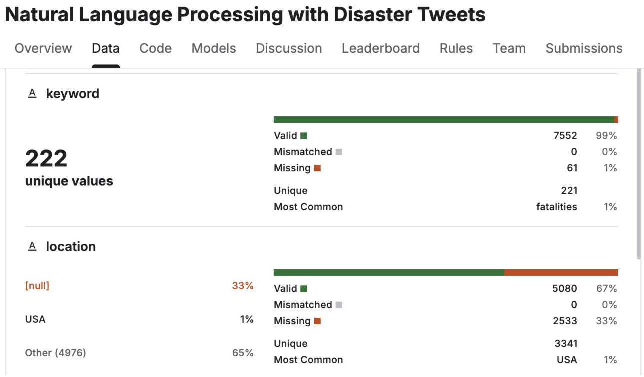
```
(7613, 5) (3263, 4)
```

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1

Data Cleaning

- There are a lot of **symbols, user taggings, and URL strings** in the tweets, which I considered unimportant for model training. Therefore, I removed these patterns during the initial cleaning process.
- The dataset also contains some **missing values** in the location and keyword fields. These missing entries were filled with empty strings to ensure consistent input for subsequent preprocessing.

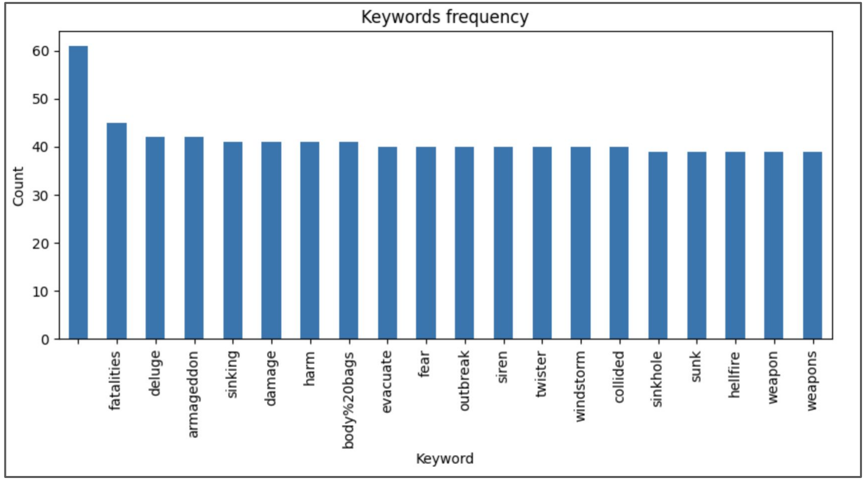




Exploratory Data Analysis (EDA)

```
nan keyword count: 61 (0.80%)
nan location count: 2533 (33.27%)
```

- The **keyword feature** shows a strong semantic correlation with the disaster-related content. For example, keywords such as “wreckage” are associated with disaster contexts. In contrast, keywords like “panicking,” often appear in non-disaster contexts.
- We can use the keyword column as an additional feature in deep learning model.



keyword	
wreckage	1.000000
debris	1.000000
derailment	1.000000
outbreak	0.975000
oil%20spill	0.973684
typhoon	0.973684
suicide%20bombing	0.969697
suicide%20bomber	0.967742
bombing	0.931034
suicide%20bomb	0.914286

keyword	
panicking	0.060606
blew%20up	0.060606
traumatised	0.057143
screaming	0.055556
electrocute	0.031250
body%20bag	0.030303
blazing	0.029412
ruin	0.027027
body%20bags	0.024390
aftershock	0.000000

Feature Engineering

- For the traditional machine learning baseline, I used **TF-IDF vectorization** to represent the textual data.
- For RNN, I tokenized the text using Keras Tokenizer and padded the sequences to the same length. I used **pre-trained GloVe embeddings** to build an embedding matrix. It provides the RNN with **semantic information from GloVe** instead of learning word meanings from start.
- I also added extra numeric features, like keyword, sentiment score, to check if they could actually help the model perform better.

Baseline Model

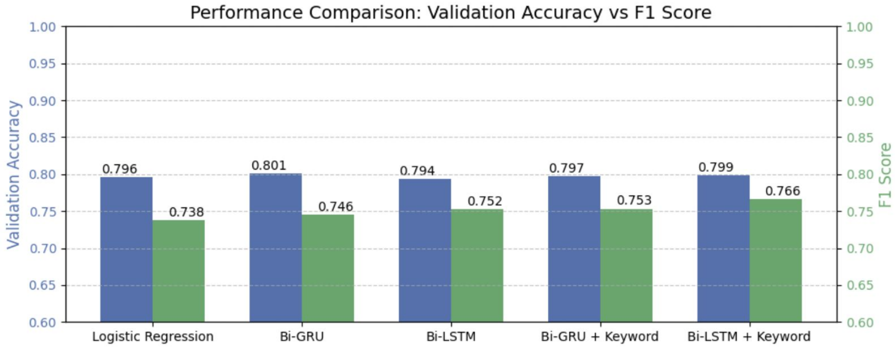
- For the traditional machine learning baseline, I trained a **Logistic Regression classifier** on the dataset
- The model reaches approximately **79.6% validation accuracy and an F1-score of around 0.738**, showing that a simple linear model can already capture a significant portion of the disaster-related semantics through word frequency patterns.

RNN Model

I compared the following RNN family models and especially I want to know whether other feature inputs, like keyword, sentiment score, have impact on RNN model :

- Bidirectional GRU
- Bidirectional LSTM
- Bidirectional GRU with keyword, sentiment score
- Bidirectional LSTM with keyword, sentiment score

Results and Analysis



The chart compares the performance of five models — Logistic Regression, Bi-GRU, Bi-LSTM, Bi-GRU + Keyword, and Bi-LSTM + Keyword — using validation accuracy and F1-score as evaluation metrics.

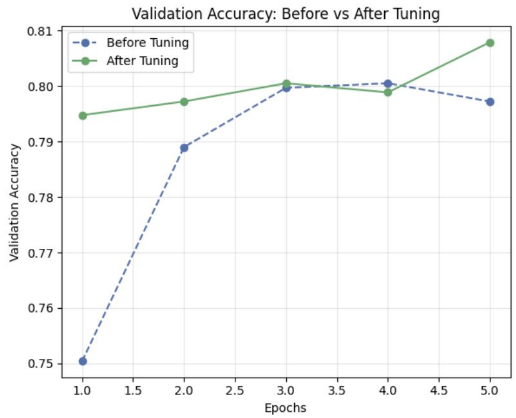
The baseline achieved around 0.796 accuracy and 0.738 F1-score, while both Bi-GRU and Bi-LSTM models reached around 79.4%-80.1% accuracy and 0.746–0.766 F1-score.

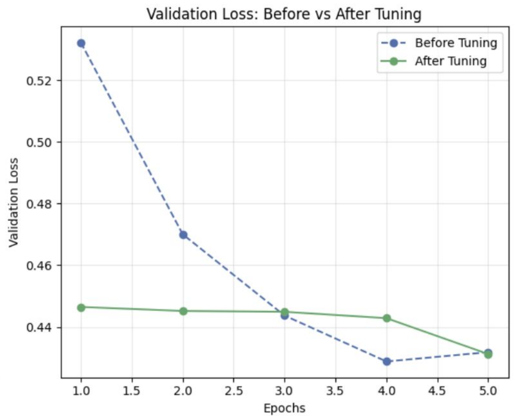
I think it's not good enough but it still indicates that sequential neural networks has potential to better capture contextual dependencies in tweets.

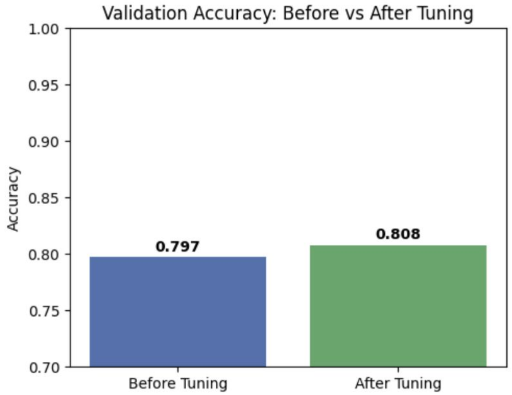
RNN Tuning

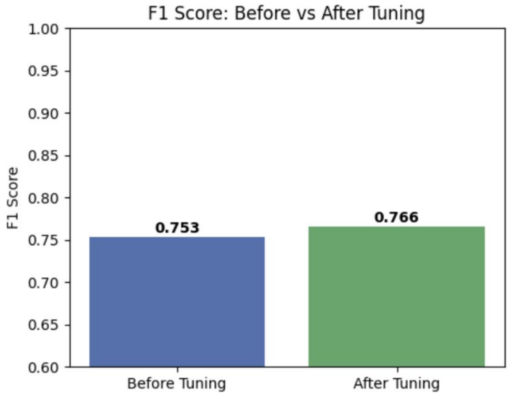
The figures compare the validation performance before and after hyperparameter tuning using RandomSearch.

After tuning, we can see the Bi-GRU + Keyword model has higher validation accuracy, it gets a better generalization and optimization stability.









Conclusion

The final comparison summarizes the model evolution from the traditional Logistic Regression baseline to the deep learning models.

The tuned Bi-GRU + Keyword model further enhanced validation accuracy and F1-score, showing better generalization and balance between precision and recall.

It shows that both **architectural design** and **hyperparameter optimization are** importance in achieving optimal NLP performance.

model	accuracy	f1-score
Logistic Regression	0.796	0.738
Bi-GRU	0.801	0.746
Bi-LSTM	0.794	0.752
Bi-GRU + Keyword	0.797	0.753
Bi-LSTM + Keyword	0.799	0.766
Bi-GRU + Keyword Tuning	0.808	0.766

Future Work

Although Bi-GRU + Keyword after tuning performed slightly better, it also required more training time and computational resources :

- Expanding data diversity with NLP data augmentation (synonym replacement, back translation, etc.)
- Fine-tuning pretrained embeddings or using contextual models (BERT, RoBERTa, etc.)

Reference

- GloVe: Global Vectors for Word Representation
- Tokenization vs Embeddings
- Keras Tokenizer
- Keras Embedding layer
- Keras RandomSearch
- Introduction to NLP with Disaster Tweets
- Classification of Disaster Tweets Using Natural Language Processing Pipeline