NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

IS6751
TEXT AND WEB MINING

TWEET-ECTIVE: DETECTING DISASTER TWEETS

Submitted by:

Christine (G2304472B)
Tan Si Heng (G2304495L)
Wang Shiyong (G23040223A)
Yang Shuang (G2204119B)

[word count:2999]

Table of Contents

1. Introduction

Social media platforms have emerged as pivotal sources for real-time information due to their widespread reach and rapid dissemination capabilities (Castillo, Mendoza, & Poblete, 2013), especially during emergencies. Hence, accurately classifying disaster-related content is crucial for public safety and emergency response.

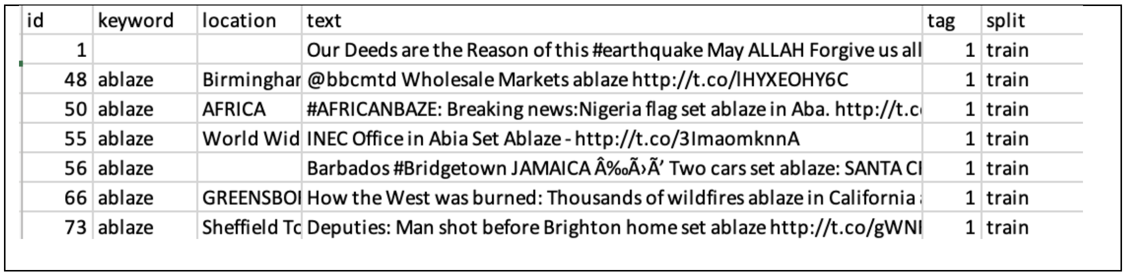
This project endeavors to identify and classify disaster-related content disseminated through social medias using natural language processing (NLP) and machine learning techniques. Such classification holds substantial value in disaster management applications. For instance, when integrated into a web API, the disaster tweet classification system can continuously monitor incoming tweets and automatically trigger alerts to emergency responders and the public.

The prototype developed in this project focuses on classifying English-language content sourced from Twitter, a widely used social media platform. The tweet data undergoes pre-processing and employed to train text classification models. These models are designed to discern textual similarities across disaster and non-disaster categories, thereby facilitating the detection of disaster-related content.

Disaster Tweet	Non-Disaster Tweet
How the West was burned: Thousands of wildfires ablaze in California alone	My room looks like a tornado passed through it and my OCD is not having it.
Environment Canada confirms 2nd tornado touched down last weekend Ã¥Ãˆ http://t.co/x8zqbwNfO1	The bomb was so appropriate ?? seen as my family and most Jamaicans love shout bullets !

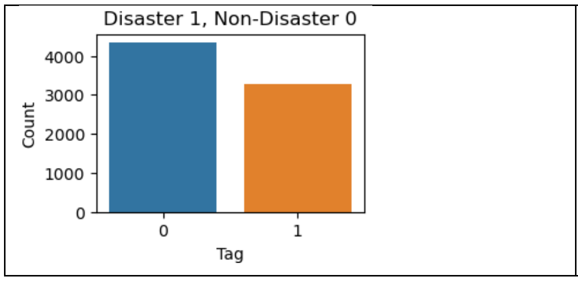
2. Dataset

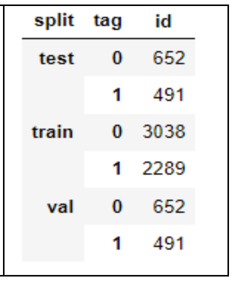
The dataset is obtained from Kaggle (Kaggle, 2023). The dataset contains the text of tweet, and tag that indicates whether the given tweet is **a disaster (1) or non-disaster (0)**. Other columns include id, location, keyword.



id	keyword	location	text	tag	split
1			Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	1	train
48	ablaze	Birminghar	@bbcmtd Wholesale Markets ablaze http://t.co/lHYXEOHY6C	1	train
50	ablaze	AFRICA	#AFRICANBAZE: Breaking news:Nigeria flag set ablaze in Aba. http://t.c	1	train
55	ablaze	World Wid	INEC Office in Abia Set Ablaze - http://t.co/3ImaomknnA	1	train
56	ablaze		Barbados #Bridgetown JAMAICA Â‰Ã›Ã' Two cars set ablaze: SANTA CI	1	train
66	ablaze	GREENSBOI	How the West was burned: Thousands of wildfires ablaze in California	1	train
73	ablaze	Sheffield Tc	Deputies: Man shot before Brighton home set ablaze http://t.co/gWNI	1	train

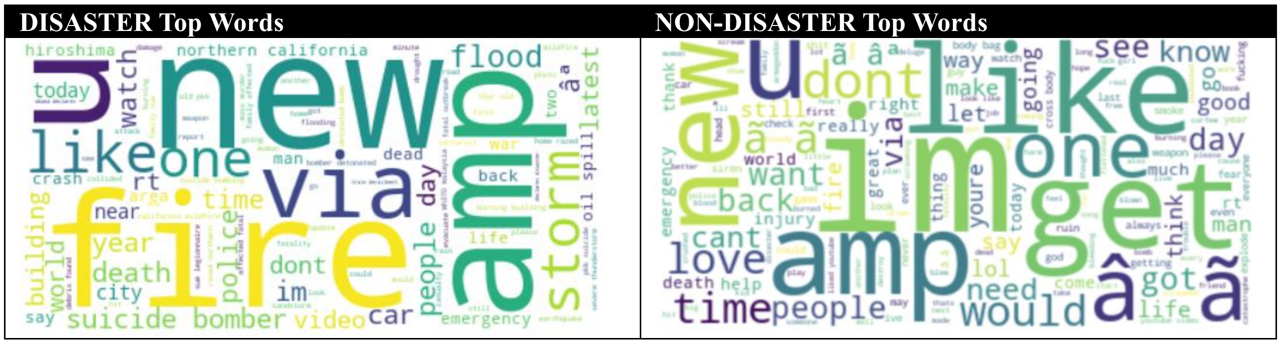
The dataset is evaluated to be an imbalanced dataset, class weights are applied during training in all models to address this issue, and then split into train (70%), validation (15%) and test (15%).





split	tag	id
test	0	652
	1	491
train	0	3038
	1	2289
val	0	652
	1	491

The dataset comprises 7600 instances and 16,609 vocabularies, reflecting the varied terminologies in tweets. The informal language in tweets, however, further complicates the task, with frequent typos, abbreviations, and short forms common to social media. Words are often truncated or altered for character limits, and slang or acronyms may replace standard vocabulary. These characteristics pose challenges for the disaster tweet classification task, which requires careful pre-processing and robust model selection to handle data nuances.



3. Model Selection & Experiment Methods

To develop an effective disaster tweet classification system, we explored various machine learning architectures to assess their suitability and performance on the data, including Multi-Layer Perceptron (MLP), Recurrent Neural Networks (RNN) - specifically Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT), given their distinct attributes and capabilities in handling natural language data.

MLP serves as a fundamental model, providing a comparative grounding with its simplicity and efficiency. Its feedforward architecture and ability to learn non-linear patterns make MLP suitable for various classification tasks, offering insights into the basic complexity of the problem.

RNNs, specifically its variants GRU and LSTM, with the design to mitigate vanishing gradients, excel at handling sequential data and may capture temporal dependencies in tweets, aiding in accurate classification.

BERT, a cutting-edge NLP model, leverages deep learning and is pre-trained on extensive text data. Utilizing bidirectional context and attention mechanisms, it comprehends language nuances and excels in complex classification tasks. We aim to assess BERT's performance on our dataset, which primarily consists of short tweets.

For each technique listed above, multiple trials have been conducted to identify the best combinations of hyperparameters, from the most critical ones down to the less significant ones, i.e. learning rate, batch size, number of hidden units, etc.

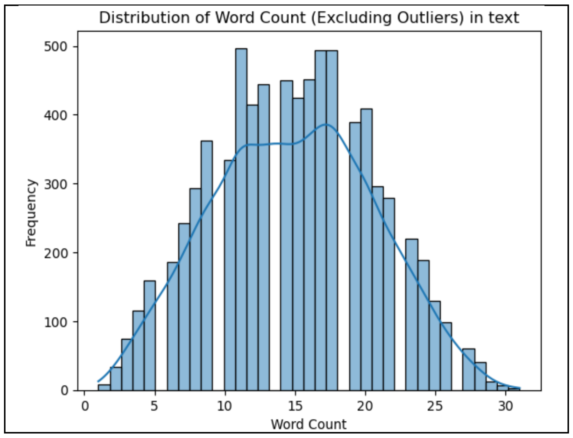
The technical approach is as follows:

- A baseline model is developed based on each technique, preserved as a reference point.
- Hyperparameters values are experimented, results including loss, accuracy and other metrics are recorded and written to csv file for tracking and analysis.
- For MLP, RNN, GRU and LSTM, the top 5 models and their applied hyperparameters are evaluated, and the best performing model is recommended. For BERT, the optimal set of hyperparameters are retrieved from the tuning process and applied to get the best model.

4. Data Preprocessing

MLP, RNN, GRU and LSTM models

Punctuation, stop words are removed. Case fold applied to remove some noise and variations. Lemmatization is applied for MLP to help the model with the generalization to new or unseen data, but not applied for GRU and LSTM, since these models are sensitive to the sequence of words. Tweet data is generally short with average of 15 words as shown below. Lemmatization that reduces words to their basic forms may lead to the loss of context for GRU and LSTM models.



Unigram and bigram are applied for MLP, but not to RNNs as it has the capability to capture sequential relationship.

BERT model

BERT's data pre-processing is distinct, using WordPiece tokenization to split words into sub-word units, aiding in handling out-of-vocabulary words and grasping sub-word meanings across languages. BERT also masks some input text during training to predict masked words, enhancing its bidirectional context learning and performance in various language tasks.

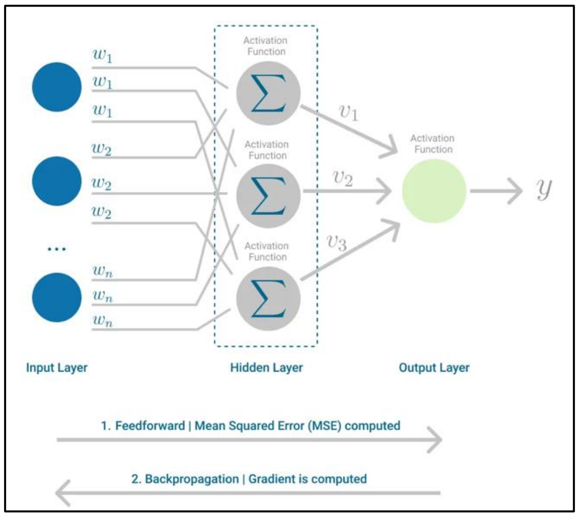
For BERT, the following data cleaning is included:

- Merging the 'keyword' column into the 'text' column to address missing values and add context.
- Removing URLs to reduce noise.
- Adding spaces around special characters to maintain word context.
- Expanding contractions for improved tokenization.

5. Multi-Neural Network Model (MLP)

Understanding MLP

A MLP is a class of artificial neural network consists of at least three layers of nodes: an input layer, one or more hidden layers, and an output layer. An MLP takes a set of inputs and transforms them into an output. Each node (also known as a neuron) in one layer connects with a certain weight to every node in the following layer. These weights are the parameters that the model learns during training. The transformation from one layer to another involves a weighted sum of the inputs, followed by a non-linear activation function. The activation function introduces non-linearities that enable the model to learn complex patterns.



Activation Function
w_1
w_2
w_n
v_1
v_2
v_3
y
Input Layer
Hidden Layer
Output Layer
1. Feedforward | Mean Squared Error (MSE) computed
2. Backpropagation | Gradient is computed

(Bento, 2021)

Developing MLP

For MLP experiments, base code is developed based on *3_5_Classifying_Yelp_Review_Sentiment.ipynb* (Na, 2023) which was modified to suit Disaster Tweet dataset with below initial parameters to develop a baseline model as reference point for improvement. MLP baseline's **Test Loss** is: **0.54**, and **Test Accuracy** is **77.67.**

MLP Baseline			
Model	**LRate**	**Batch Size**	**Hidden Layer**
Baseline (t0)	0.001	64	0

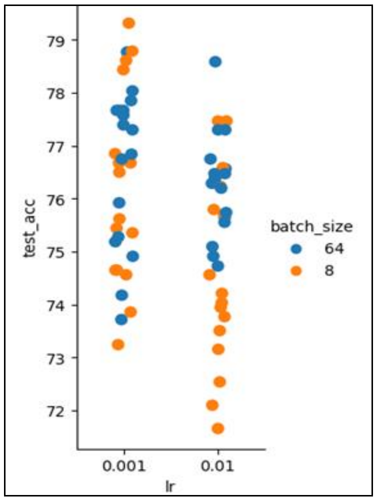
Finding the best hyperparameters. The methodology applied for hyperparameter tuning is by conducting multiple trials using the following combination.

- N-gram: 1, 2
- Learning rate: 1e-2, 1e-3
- Batch size: 8, 64
- Hidden dim: 20, 100
- Hidden layer: 0, 2

Tuning the Learning Rate and Batch Size

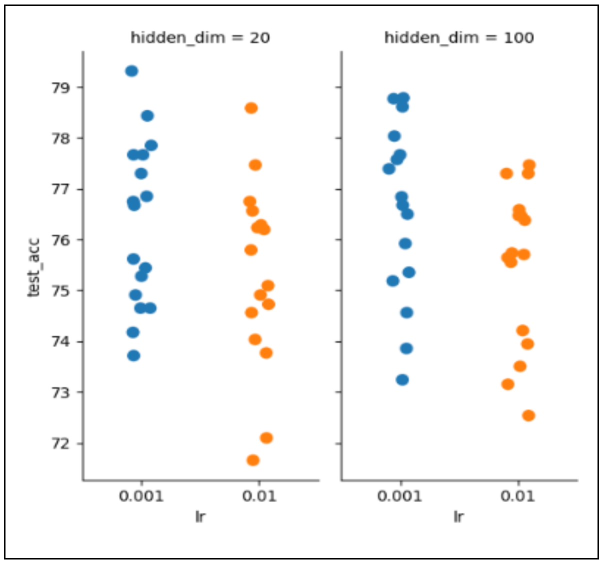
Learning rate is the first hyperparameter to tune which if considered one of the most crucial hyperparameter since it determines the weight update during training. Its combination with batch size is tested at the same time.

Trial result shows that lower learning rate with smaller batch size produces higher test accuracy and lower test losses. Since the 'Disaster' data set is small, a smaller batch size is indeed more suitable. A smaller learning rate is needed to maintain stability.



Finding the Best Number of Hidden Units

The number of hidden units is the 2nd most important; 20 and 100 units are experimented. The trial result shows that lower number of hidden neurons results in higher accuracy with insignificant improvement. Higher hidden size indeed proven to make the model more effective by adding variation and features to small sequence of tweets leading to improved accuracy.

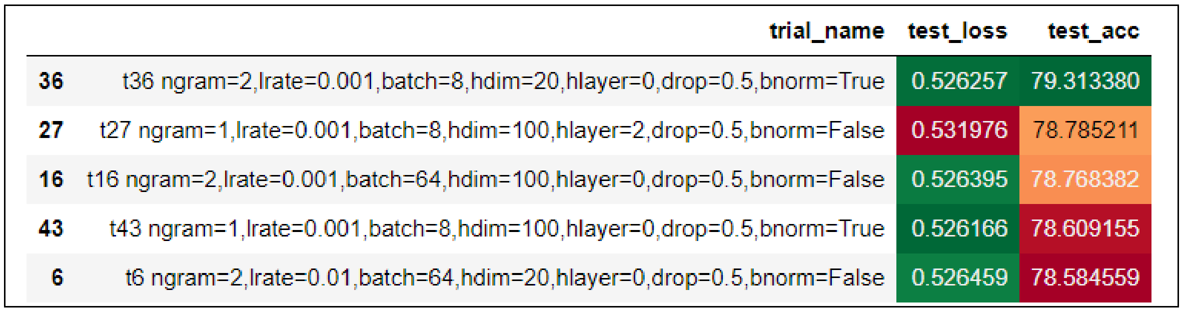


Regularization with Drop-Out, and Batch Norm

The MLP baseline model shows higher training accuracy than validation's data which indicates overfitting. Regularization techniques i.e. Drop-Out, and Batch Norm are in this case required to help address the issue.

Evidently, the model that applies **drop-out rate (0.5)** results in higher test accuracy and lower test lost, and regularization such as batch normalization further improves the model.

Best 5 MLP models. The top 5 models and their applied hyperparameters are as per below table. The best model has **Test Accuracy: 79.31** and **Test Loss**: **0.53.**



	trial_name	test_loss	test_acc
36	t36 ngram=2,lrate=0.001,batch=8,hdim=20,hlayer=0,drop=0.5,bnorm=True	0.526257	79.313380
27	t27 ngram=1,lrate=0.001,batch=8,hdim=100,hlayer=2,drop=0.5,bnorm=False	0.531976	78.785211
16	t16 ngram=2,lrate=0.001,batch=64,hdim=100,hlayer=0,drop=0.5,bnorm=False	0.526395	78.768382
43	t43 ngram=1,lrate=0.001,batch=8,hdim=100,hlayer=0,drop=0.5,bnorm=True	0.526166	78.609155
6	t6 ngram=2,lrate=0.01,batch=64,hdim=20,hlayer=0,drop=0.5,bnorm=False	0.526459	78.584559

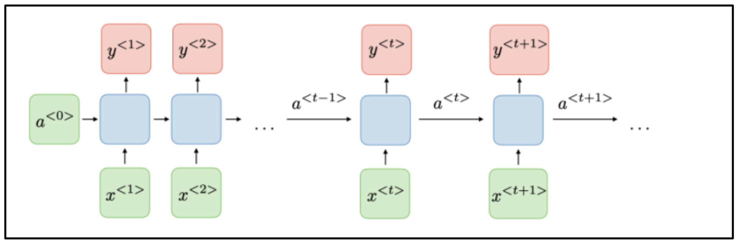
Conclusion. Despite the overfitting issue, applying smaller batch size, lower learning rate and regularization improves the MLP model from **Test Loss**: **0.54** to **0.53**, and **Test Accuracy**: **77.67** to **79.31.** A smaller batch size allows the model to generalize better, and a lower learning rate helps to maintain the stability.

In addition, the speed of processing of MLP makes such models practical for disaster warning use-case. During our MLP experiments, a total of 64 MLP models were evaluated, and this process was completed in just over an hour.

6. Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) Networks

Understanding RNNs and its variants GRU & LSTM

RNNs are a class of artificial neural networks designed for sequence prediction problems and tasks that require memory of past information (Elman, 1990). Unlike traditional feedforward networks, RNNs possess a form of internal memory that helps them to maintain context and process sequences of data. While RNN is well-suited for tasks involving sequences, such as time series prediction and natural language processing, vanilla RNNs often struggle to capture long-term dependencies due to issues like vanishing or exploding gradients (Razvan, Tomas, & Yoshua, 2013).



(Afshine & Shervine, 2023)

To address these limitations, variants of RNNs were developed, namely GRUs and LSTMs.

GRUs introduce gating units that modulate the flow of information to be remembered or forgotten at each time step which makes it easier to capture dependencies for sequences of varied lengths (Kyunghyun, et al., 2014).

LSTMs also use gating units to regulate the flow of information. However, LSTMs have an additional memory cell that aids in learning long-term dependencies, making LSTMs highly effective for tasks that require learning from context over long sequences (Sepp & Jürgen, 1997).

Developing RNNs (GRU & LSTM)

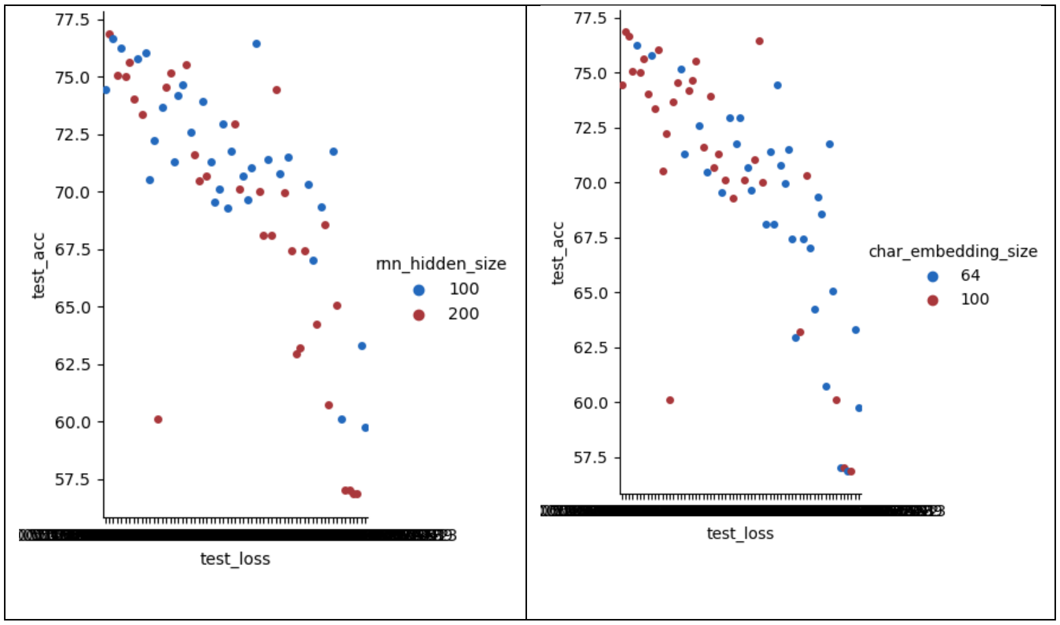
For RNN experiments, the base code is developed based on Chapter-6-Surname-Classification-with-RNNs-v2-use-library-RNN.ipynb (Na, 2023) which was updated to suit Disaster Tweet dataset. The initial RNN baseline model's Test Loss is: **0.82**, with Test Accuracy: **65.44**, which is significantly lower than MLP baseline since RNNs typically require larger dataset to learn effectively.

RNN baseline						
Model	**LRate**	**Batch Size**	**Hidden Size**	**Input Size**	**Num Layer**	**RNN Type**
Baseline (t0)	0.001	64	100	64	1	RNN

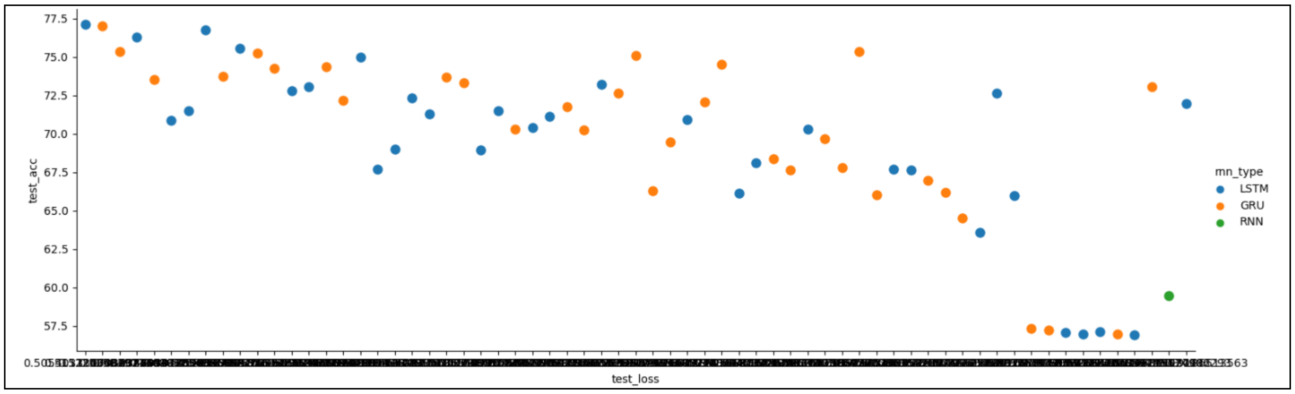
Finding the best hyperparameters. The methodology applied for RNN hyperparameter tuning is by conducting trials using the following combination.

- RNN types: GRU, LSTM
- Learning rate: 1e-2, 1e-3
- Batch size: 8, 64
- RNN Hidden size: 100, 200
- Embedding size: 64, 100
- Num layers: 1, 2
- Bidirectional: True

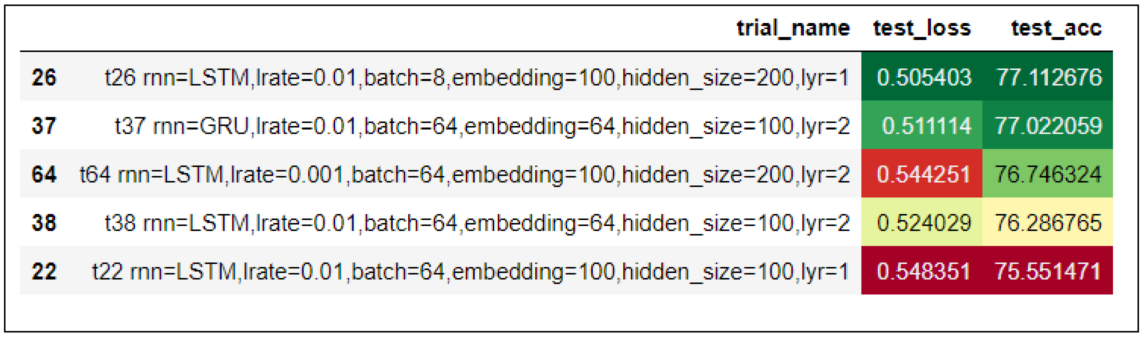
Due to overfitting issue as indicated by training accuracy that is higher than validation's, regularization using drop-out rate 0.5 is also applied to RNN models. The result of experiments indicates that models with larger hidden sizes and larger input / embedding size produce higher test accuracy and lower test loss since larger embedding and hidden sizes increase the model capacity enabling it to capture more complex patterns and relationships in the data. In short, the model can learn more effectively as a result.



LSTM models also generally perform better than GRU as LSTM handles vanishing gradient problems and capturing long-range dependencies better.



Best 5 RNN models. The top 5 models and their applied hyperparameters are as follows. The best model has **Test Accuracy: 77.11** and **Test Loss**: **0.51.**



	trial_name	test_loss	test_acc
26	t26 rnn=LSTM,lrate=0.01,batch=8,embedding=100,hidden_size=200,lyr=1	0.505403	77.112676
37	t37 rnn=GRU,lrate=0.01,batch=64,embedding=64,hidden_size=100,lyr=2	0.511114	77.022059
64	t64 rnn=LSTM,lrate=0.001,batch=64,embedding=100,hidden_size=200,lyr=2	0.544251	76.746324
38	t38 rnn=LSTM,lrate=0.01,batch=64,embedding=64,hidden_size=100,lyr=2	0.524029	76.286765
22	t22 rnn=LSTM,lrate=0.01,batch=64,embedding=100,hidden_size=100,lyr=1	0.548351	75.551471

Conclusion. LSTM model managed to improve the baseline RNN model from **Test Loss**: **0.82** to **0.51**, and **Test Accuracy**: **65.44** to **77.11.** However, although GRU and LSTM are suitable for text classification tasks, the small data set causes RNN models to generally perform poorer than MLP model in our experiments. RNNs do, in fact, require a larger dataset to perform optimally. The size and quality of the dataset is indeed crucial for text classification tasks.

7. Bidirectional Encoder Representations from Transformers (BERT)

Understanding BERT

BERT is built upon the Transformer architecture and leverages a mechanism called attention, allowing it to focus on different words for a given input. Unlike traditional models that process words in a sentence sequentially, BERT is bidirectional, meaning it considers the context from both before and after a word simultaneously. This bidirectional approach enables a more robust understanding of context and word relationships within a sentence.

One of the standout features of BERT is its pre-training on large corpora of text data. During this phase, BERT learns to predict masked words in a sentence (a task known as masked language modeling) and to distinguish between two sentences (next sentence prediction). These pre-training tasks allow BERT to develop a rich understanding of language semantics and context.

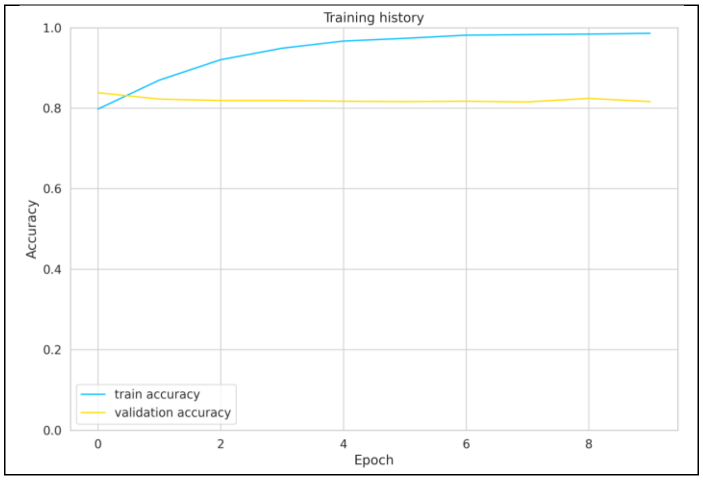
Developing BERT

For BERT experiments, the base code is developed based on *sentiment_analysis_with_bert_for_colab_v3t.ipynb* (Na, 2023) which was modified to suit Disaster Tweet dataset with below initial parameters to develop a baseline model as reference point for improvement. BERT baseline's **Test Loss** is: **1.07**, and **Test Accuracy** is **82.05.**

BERT Baseline									
Model	Bert Variant	Batch Size	Num of Epochs	Max Seq Length	Dropout Rate	Learning Rate	Weight Decay	Num of Warmup Steps	Early Stopping Criteria
Baseline	bert-base-uncased	16	10	128	0.3	2e-5	NA	0	1

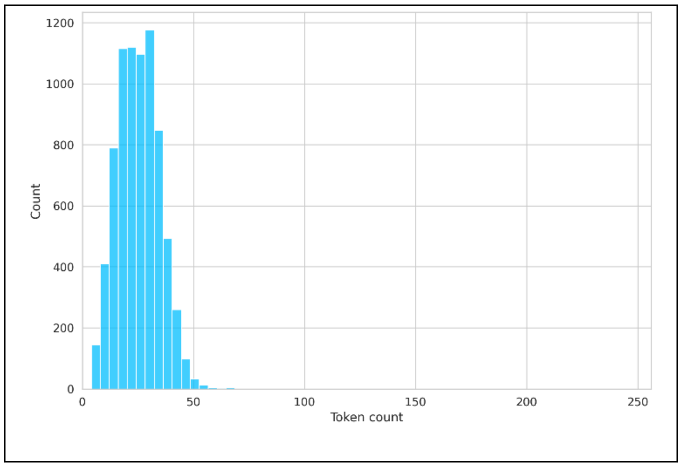
Overfitting

The training accuracy of the BERT baseline model is consistently higher than the validation accuracy. This indicates that the model is overfitted. To tackle this overfitting issue, and simultaneously improve model performance, hyperparameter tuning is required.



Tuning the Maximum Sequence Length

Before tuning the other hyperparameters, the maximum sequence length used for the tokenizer can be tuned first by checking the distribution of the tweets. As seen in the histogram below, most of the tweets have less than 100 token counts after tokenizing. Hence, a maximum sequence length of 128 is chosen to provide some buffer.



Finding the best hyperparameters. The methodology applied for BERT hyperparameter tuning is different from the previous models, due to the high computational requirements and long runtime of the BERT model. Hence, Optuna, a hyperparameter optimization library is utilized, where multiple trials were conducted for the following hyperparameters. The results are then recorded in a csv file.

- Bert Variant: bert-base-uncased, bert-large-uncased, bert-base-cased, bert-large-cased

- Learning Rate: 1e-6 to 1e-3
- Dropout Rate: 0.0 to 0.5
- Batch Size: 16, 32, 64
- Weight Decay: 1e-6 to 1e-3
- Num of Warmup Steps: 0, 500, 1000
- Num of Epochs: 5 to 10

Optuna employs a trial-based approach to efficiently search for the optimal set of hyperparameters. It operates using Bayesian optimization, where it intelligently selects new hyperparameter combinations that are likely to improve the model's performance, effectively narrowing down the search space with each trial. This process allows for the automation of hyperparameter tuning, as well as the cutting down of computational resources due to its efficiency.

For this BERT model, a total of 8 trials were ran in search of the optimal hyperparameters. To monitor for overfitting, the validation loss of the model is set as the target for the tuning process, which is shown in the table below.

BERT Hyperparameter Tuning

Model	Bert Variant	Batch Size	Num of Epochs	Max Seq Length	Dropout Rate	Learning Rate	Weight Decay	Num of Warmup Steps	Early Stopping Criteria	Val Loss
Trial 0	bert-large-uncased	32	10	128	0.464908922	1.01E-06	2.62E-06	1000	1	0.440151
Trial 1	bert-base-cased	64	7	128	0.375526321	3.24E-06	8.03E-05	500	1	0.443279
Trial 2	bert-base-cased	16	5	128	0.31359567	7.51E-05	1.09E-06	1000	1	0.61569
Trial 3	bert-base-cased	64	7	128	0.083025126	1.17E-06	2.04E-05	500	1	0.449117
Trial 4	bert-large-cased	16	5	128	0.362011787	8.95E-06	2.75E-06	1000	1	0.734694
Trial 5	bert-base-uncased	16	7	128	0.457236712	4.89E-05	2.15E-05	500	1	0.992589
Trial 6	bert-large-uncased	32	8	128	0.014929712	1.16E-04	8.35E-06	0	1	0.67581
Trial 7	bert-base-uncased	16	6	128	0.401242087	1.37E-06	1.96E-06	1000	1	0.432982

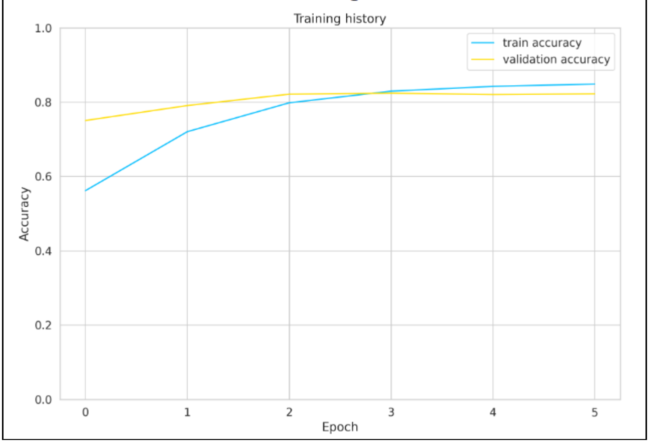
From the hyperparameter tuning process, Trial 7 yields the lowest validation loss, providing the optimal set of hyperparameters for the BERT model.

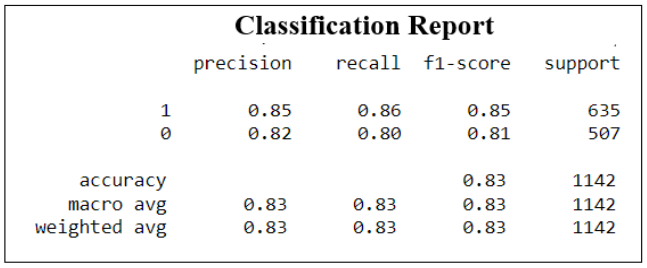
Best BERT model. By applying the best hyperparameters retrieved from the tuning process, the best BERT model is attained. The best model has **Test Accuracy: 83.45** and **Test Loss**: **0.41.**

BERT Best Model

Model	Bert Variant	Batch Size	Num of Epochs	Max Seq Length	Dropout Rate	Learning Rate	Weight Decay	Num of Warmup Steps	Early Stopping Criteria	Test Acc	Test Loss
Best	bert-base-uncased	16	6	128	0.401242087	1.37E-06	1.96E-06	1000	1	0.834501	0.408416

Overfitting Check





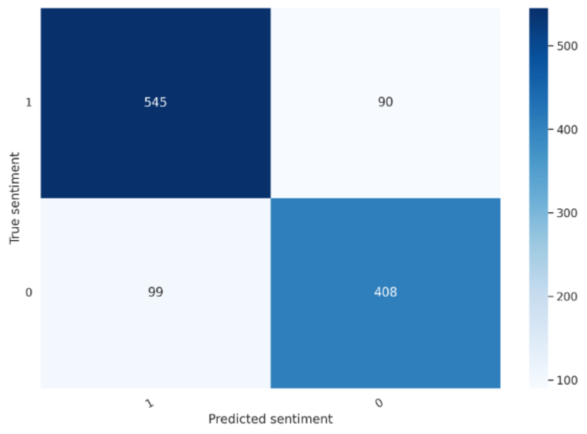
Classification Report

```
              precision    recall  f1-score   support

           1       0.85      0.86      0.85       635
           0       0.82      0.80      0.81       507

    accuracy                           0.83      1142
   macro avg       0.83      0.83      0.83      1142
weighted avg       0.83      0.83      0.83      1142
```

Confusion Matrix



Conclusion. BERT model, after hyperparameter tuning using Optuna, is able to improve **Test Loss**: **1.07** to **0.41**, and **Test Accuracy**: **82.05** to **83.45.** The training accuracy is also aligned with the validation accuracy, indicating that there no longer an overfitting issue with the model. The classification results for the model demonstrate a well-balanced performance, with an overall accuracy of 83%. It achieves high precision for both disaster-related and non-disaster tweets (85% and 82%, respectively), indicating a low rate of false positives. The model's recall rate, particularly for disaster-related tweets, is strong at 86%, confirming its effectiveness in identifying such tweets, making it a reliable choice for text classification tasks.

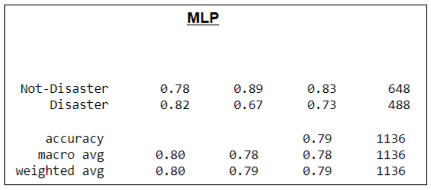
8. Discussion & Conclusion

The project explored the efficacy of three distinct machine learning models—MLP, RNN (GRU&LSTM) and BERT—for classifying tweets into disaster-related and non-disaster categories. The comparative analysis revealed nuanced insights into the performance and adaptability of each model in addressing the challenges posed by the dataset.

The MLP model showed resilience against overfitting and improved generalization with careful tuning parameters. The exploration of RNNs, specifically LSTM and GRU were found to be slightly outperformed by the MLP model in this context. The BERT model emerged as the most proficient, achieving a commendable test accuracy of 83.45% and demonstrating well-rounded performance with high precision and recall rates for both categories, the low loss shows that it can effectively discern nuances in disaster-related tweets, however, imposes higher computational cost.

In conclusion, while each model presented its own strengths and areas of improvement, the results collectively emphasize the importance of data size and quality in model performance. BERT, with its robust contextual understanding and strong performance metrics, stood out as a reliable choice for disaster tweet classification. The findings from this project underscore the potential of fine-tuning and optimizing machine learning models to effectively classify and disseminate critical information in real-time during emergencies, thereby contributing to efficient and timely disaster management.

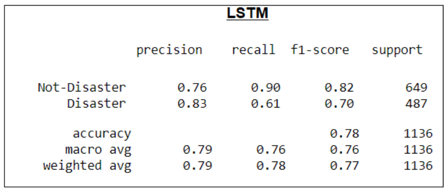
Classification Report



MLP

```
Not-Disaster       0.78      0.89      0.83       648
    Disaster       0.82      0.67      0.73       488

    accuracy                           0.79      1136
   macro avg       0.80      0.78      0.78      1136
weighted avg       0.80      0.79      0.79      1136
```

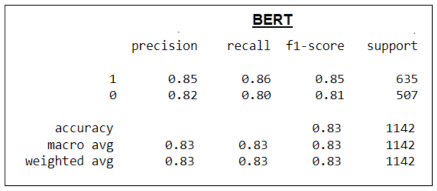


LSTM

```
              precision    recall  f1-score   support

Not-Disaster       0.76      0.90      0.82       649
    Disaster       0.83      0.61      0.70       487

    accuracy                           0.78      1136
   macro avg       0.79      0.76      0.76      1136
weighted avg       0.79      0.78      0.77      1136
```



BERT

```
              precision    recall  f1-score   support

           1       0.85      0.86      0.85       635
           0       0.82      0.80      0.81       507

    accuracy                           0.83      1142
   macro avg       0.83      0.83      0.83      1142
weighted avg       0.83      0.83      0.83      1142
```

9. Bibliography

Danah, B., Scott, G., & Gilad, L. (2010). Tweet, Tweet, Retweet: Conversational Aspects of Retweeting on Twitter. *Annual Hawaii International Conference on System Sciences (HICSS)* (pp. 1-10). USA: IEEE.

Castillo, C., Mendoza, M., & Poblete, B. (2013). Predicting information credibility in time-sensitive social media. *Internet Research: Electronic Networking Applications and Policy*.

Bento, C. (2021, September 21). *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis*. Retrieved from Medium: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141

Afshine, A., & Shervine, A. (2023, 11 02). *Recurrent Neural Networks cheatsheet*. Retrieved from Stanford Edu: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 179-211.

Razvan, P., Tomas, M., & Yoshua, B. (2013). On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning* (pp. 1310-1318). Atlanta, Georgia, USA: PMLR 28.

Kyunghyun, C., Bart, v. M., Caglar, G., Dzmitry, B., Fethi, B., Holger, S., & Yoshua, B. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* USA: Cornell University.

Sepp, H., & Jürgen, S. (1997). *Long Short-Term Memory.* USA: Massachusetts Institute of Technology.

Takeshi, S., Makoto, O., & Yutaka, M. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. *19th International conference on World wide web* (pp. 851-860). New York, United States: Association for Computing Machinery.