Disaster Classification Using Tweets

Introduction

- On Twitter, people share real-time updates about disasters, emergencies, and how these events affect them and their communities.
- This instant flow of information makes Twitter a vital platform for understanding and responding to disasters quickly and effectively
- Early Warning: Identify disaster events as they unfold, enabling swift response from authorities and relief organizations.
- Information Verification: Filter out rumors and misinformation, ensuring that accurate information reaches the public.

Problem Statement

This project aims to develop a robust machine learning model that can classify Twitter tweets as either genuine disaster-related or non-disaster.

PREDICTION: DISASTER

PREDICTION: NON-DISASTER

Different approaches

Preprocessing

1) Text cleaning
2) Stemming/Lemmatization
3) Spelling correction
4) Handling abbreviations
5) Normalization
6) Named entity recognition

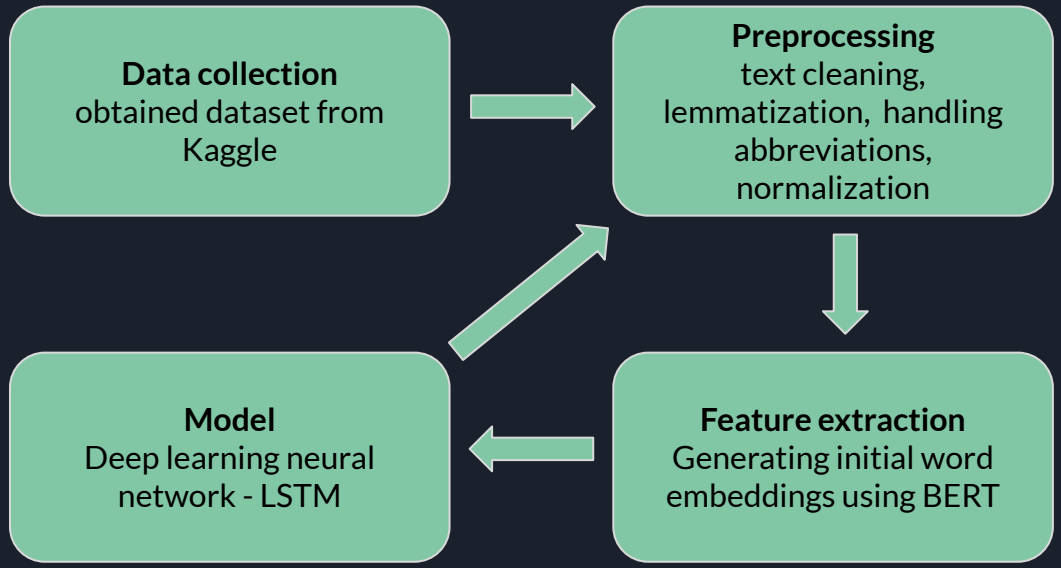
Feature Extraction

1) Bag of Words
2) TF-IDF
3) Word embeddings
4) N-grams

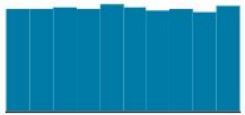
Machine learning models

1) Traditional ML models like Naive Bayes, Logistic Regression, SVM
2) Deep learning models like RNN,CNN

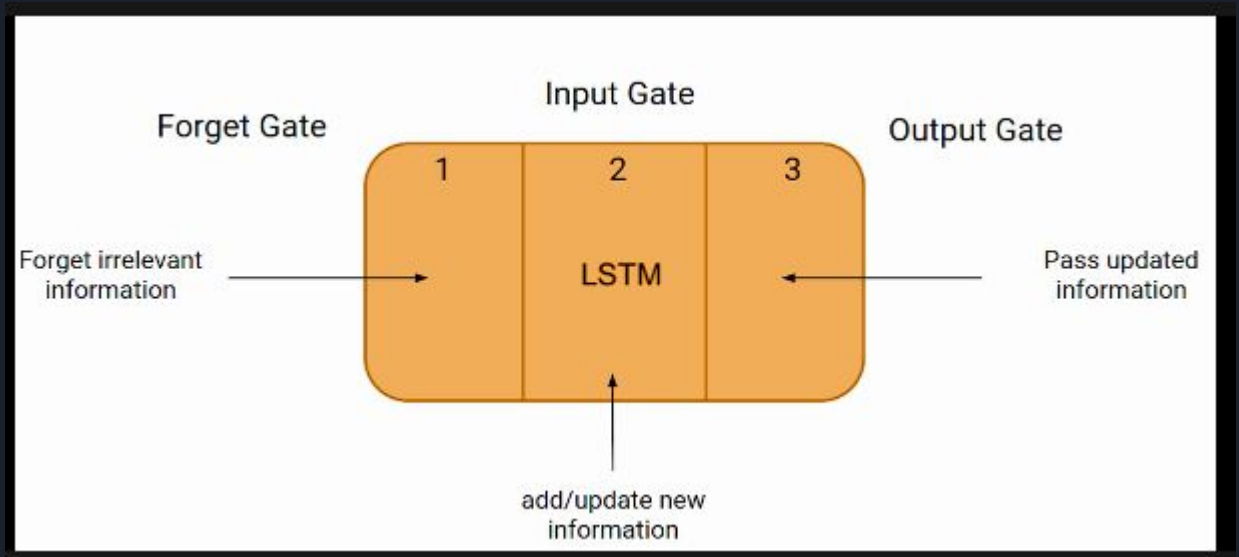
Our implementation



Dataset

id	keyword	location	text	target
 1 10.9k	222 unique values	[null] 33% USA 1% Other (4976) 65%	7503 unique values	 0 1
1			Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all	1
4			Forest fire near La Ronge Sask. Canada	1
5			All residents asked to 'shelter in place' are being notified by officers. No other evacuation or she...	1

Long - Short term memory neural network



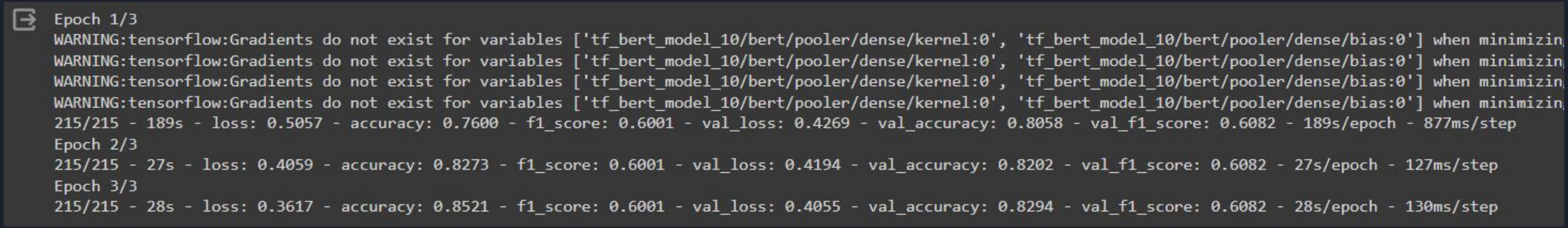
Models summary

```python
def bert_lstm_model():
    bert_encoder = TFBertModel.from_pretrained(model_name)
    input_word_ids = tf.keras.Input(shape=(max_length,), dtype=tf.int32, name="input_ids")
    last_hidden_states = bert_encoder(input_word_ids)[0]
    x = tf.keras.layers.LSTM(100, dropout=0.3, recurrent_dropout=0.3)(last_hidden_states)
    output = tf.keras.layers.Dense(1, activation='sigmoid')(x)
    model = tf.keras.Model(inputs=input_word_ids, outputs=output)

    return model
```

```python
[ ] import tensorflow_addons as tfa
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_ids (InputLayer)      [(None, 140)]             0

 tf_bert_model_10 (TFBertMod  TFBaseModelOutputWithPoo  167356416
 el)                         lingAndCrossAttentions(l
                             ast_hidden_state=(None,
                             140, 768),
                              pooler_output=(None, 76
                             8),
                              past_key_values=None, h
                             idden_states=None, atten
                             tions=None, cross_attent
                             ions=None)

 lstm_8 (LSTM)               (None, 100)               347600

 dense_8 (Dense)             (None, 1)                 101

=================================================================
Total params: 167,704,117
Trainable params: 167,704,117
Non-trainable params: 0
_________________________________________________________________
```

Accuracy of our model



```
Epoch 1/3
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model_10/bert/pooler/dense/kernel:0', 'tf_bert_model_10/bert/pooler/dense/bias:0'] when minimizin
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model_10/bert/pooler/dense/kernel:0', 'tf_bert_model_10/bert/pooler/dense/bias:0'] when minimizin
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model_10/bert/pooler/dense/kernel:0', 'tf_bert_model_10/bert/pooler/dense/bias:0'] when minimizin
WARNING:tensorflow:Gradients do not exist for variables ['tf_bert_model_10/bert/pooler/dense/kernel:0', 'tf_bert_model_10/bert/pooler/dense/bias:0'] when minimizin
215/215 - 189s - loss: 0.5057 - accuracy: 0.7600 - f1_score: 0.6001 - val_loss: 0.4269 - val_accuracy: 0.8058 - val_f1_score: 0.6082 - 189s/epoch - 877ms/step
Epoch 2/3
215/215 - 27s - loss: 0.4059 - accuracy: 0.8273 - f1_score: 0.6001 - val_loss: 0.4194 - val_accuracy: 0.8202 - val_f1_score: 0.6082 - 27s/epoch - 127ms/step
Epoch 3/3
215/215 - 28s - loss: 0.3617 - accuracy: 0.8521 - f1_score: 0.6001 - val_loss: 0.4055 - val_accuracy: 0.8294 - val_f1_score: 0.6082 - 28s/epoch - 130ms/step
```

Link to our colab notebook

https://github.com/reubenjrouse/NLP_DL.git