

Predict which Tweets are about real disasters and which ones are not

Team 4: Jiameng Sun, Yanwen Duan

Introduction

We live in a world of information explosion. We rely on the information to live a safe and normal life. However, not all information is real. Fake information, produced intentionally or unintentionally, has caused so much confusion, anxiety, and even economic loss. In contrast, real information can save lives. According to a study by Pew Research Center, Twitter served as a lifeline of information during Hurricane Sandy. In six days, people sent more than 20 million tweets about Sandy in six days. Many of the tweets helped people get help and avoid danger.

In this project, we would like to focus on Tweets and information about disasters. We seek to provide a solution to discern which tweet is about real disasters. Though there are models which can perform the same task with around 83% accuracy, we attempt to use a new disaster-related tweets database and BERTweet, a new BERT model pre-trained with tweets to improve the performance of one of the previous models. We also create a python program utilizing the BERTweet model to tell users if the tweet they provide is about real disasters. The report consists of the following sections: 1. Data: we show the data cleaning and data visualization of two databases. One database is available on Kaggle and is commonly used by previous researchers. The other is a new database we find on Kaggle for this project; 2. Model: we introduce BERT and BERTweet and report our training process 3. Analysis and Conclusion: we compare the performance of the two models, the effect of data, and summarize the findings of our project. 4. Web Application: we show the python program we build with Streamlit and BERTweet.

Data

General overview

In this project, we use two datasets from Kaggle.com. The two datasets have the same format with five columns (id, keyword, location, text(Tweets), target(label)). The first dataset comes from Kaggle's Natural Language Processing with Disaster Tweets. It is commonly used by other disaster tweets classification models and is split into a train set and a test set. The train set contains 7613 samples and all samples have labels that indicate if the tweet is about real disasters(1: yes, it is about real disasters; 2: no, it is not). The test set consists of 3263 samples but the samples come without the labels. The second dataset is a new dataset we find on Kaggle, uploaded by Viktor S with 11371 samples which all have labels. We combine the train set of the first dataset with the second dataset, which creates a new dataset of 18984 samples. Then we cleaned the new dataset for visualization and modeling. After the data cleaning, the new dataset was split into a new train set and a new test set with the `train_test_split` function from `sklearn.model_selection`. The train set contains 70% (13289) samples of the new dataset while the test set contains 30% (5695) samples.

Data cleaning

This section displays how we clean the train set for data exploratory analysis. To clean the data, we first remove URLs, emojis, HTML tags, and punctuations. Then we tokenize the tweet text, convert all text to lowercase, remove stopwords, and add ‘part of speech’ tags to each word. We convert the text to wordnet format to apply word lemmatizer. Finally, we convert tokenized text to string for data visualization (figure 1). **prepare_training_data.py** is used for data cleaning, while **data_visualization_first_modeling.ipynb** contains the same code for data cleaning but with figures which show the process such as the one below.

id	keyword	location	text	target	text_clean	tokenized	lower	stopwords_removed	pos_tags	wordnet_pos	lemmatized	lemma_str
0	1	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...	[deeds, reason, earthquake, may, allah, forgiv...	[(deeds, NNS), (reason, NN), (earthquake, NN), (allah, J...]	[(deeds, n), (reason, n), (earthquake, n), (ma...	[deed, reason, earthquake, may, allah, forgive...	deed reason earthquake may allah forgive u
1	4	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]	[forest, fire, near, la, ronge, sask, canada]	[(forest, JJ), (fire, NN), (near, IN), (la, J...]	[(forest, a), (fire, n), (near, n), (la, a), (...]	[forest, fire, near, la, ronge, sask, canada]	forest fire near la ronge sask canada
2	5	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...	[residents, asked, shelter, place, notified, o...	[(residents, NNS), (asked, VBD), (shelter, JJ), (place, NN), (notified, V...]	[(residents, n), (asked, v), (shelter, a), (place, n), (notified, v), (...]	[resident, ask, shelter, place, notify, office...]	resident ask shelter place notify officer evac...
3	6	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[(13000, CD), (people, NNS), (receive, JJ), (wildfires, NN), (evacuation, NN), (...]	[(13000, n), (people, n), (receive, a), (wildfires, n), (evacuation, n), (...]	[13000, people, receive, wildfire, evacuation, ...]	13000 people receive wildfire evacuation order...
4	7	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...	[got, sent, photo, ruby, alaska, smoke, wildfi...	[(got, VBD), (sent, JJ), (photo, NN), (ruby, NN), (alaska, NN), (smoke, NN), (wildfire, NN), (...]	[(got, v), (sent, a), (photo, n), (ruby, n), (alaska, n), (smoke, n), (wildfire, n), (...]	[get, sent, photo, ruby, alaska, smoke, wildfi...	get sent photo ruby alaska smoke wildfires pou...

Figure1. Data Cleaning

Data Visualization

We use **data_visualization_first_modeling.ipynb** for data visulization. Figure 2 displays the distribution of labels. 1 indicates that the tweet is about real disasters while 0 indicates that it is not. As we can see from the figure, the distribution is skewed. We try to fix the skewness with nlpaug, an NLP augmter library but don't get noticeable effects during training and testing. Therefore, we decide not to use the library during our final version of training and testing. Figure 3 to Figure 5 displays the most common unigrams, bigrams, trigrams, and named entities in non-disaster tweets and disaster tweets. The figures show big differences between the non-disaster tweets and disaster tweets.

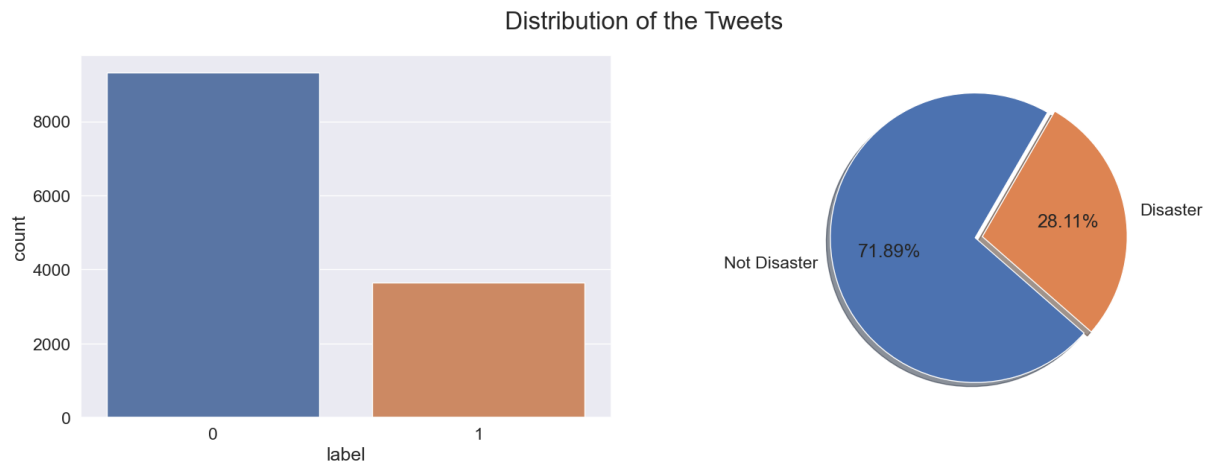


Figure 2. Label distribution

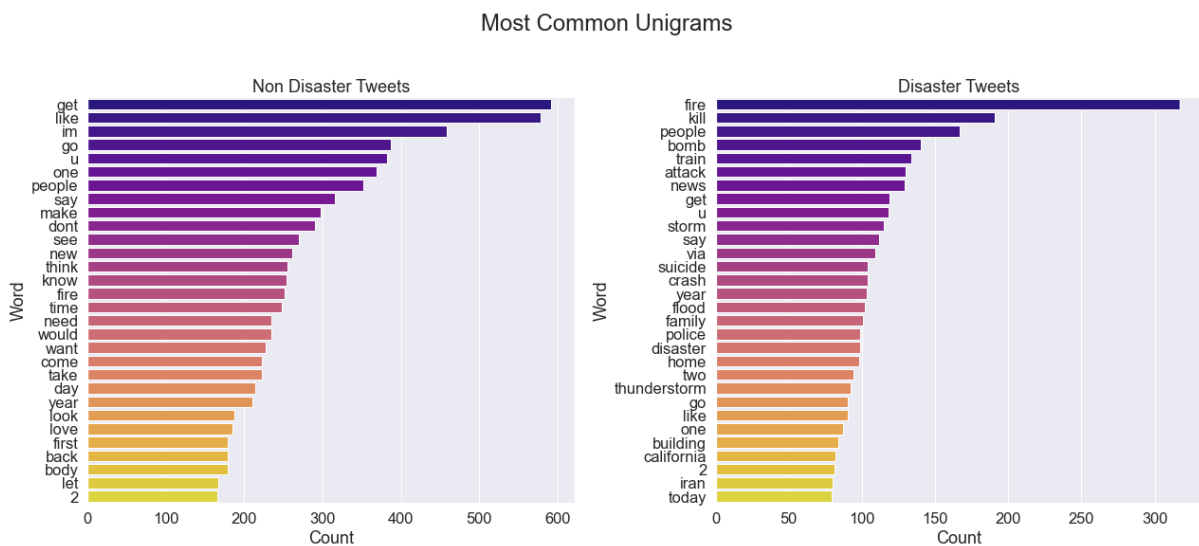


Figure 3. Most common unigrams

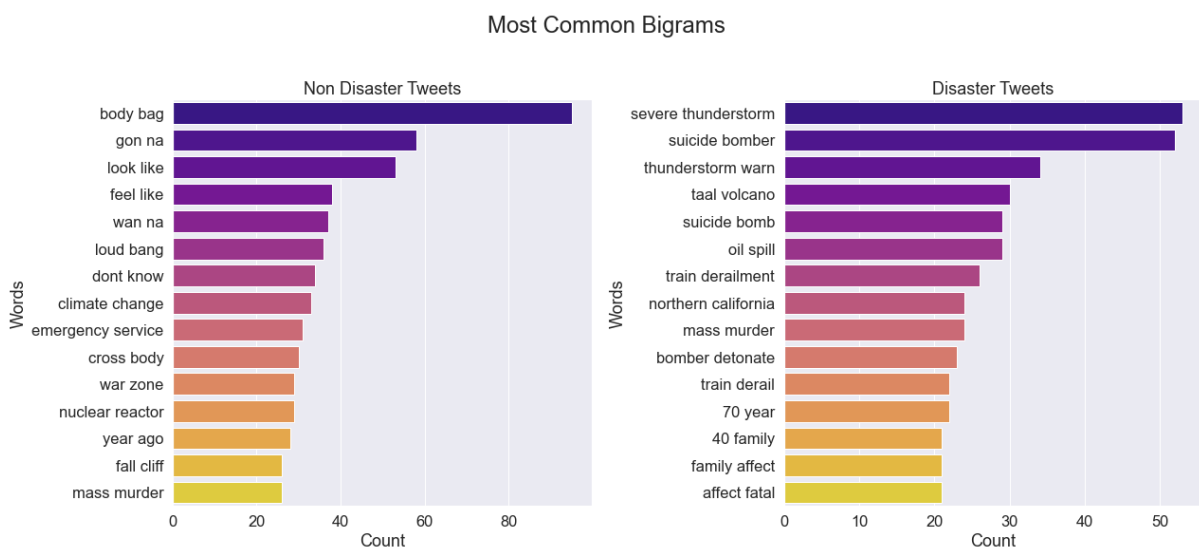


Figure 4. Most common bigrams

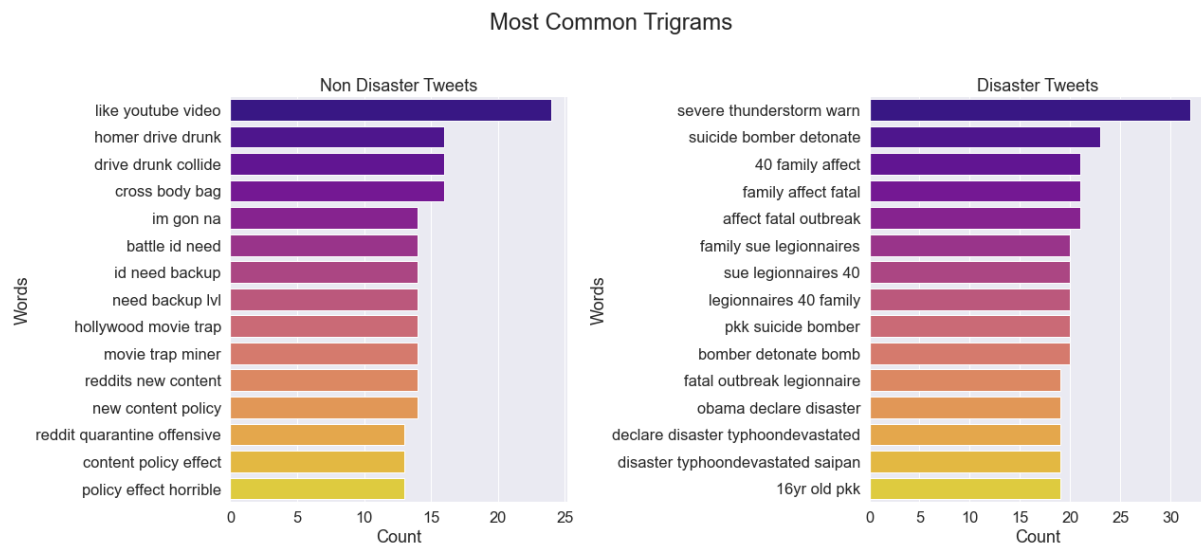


Figure 5. Most common trigrams

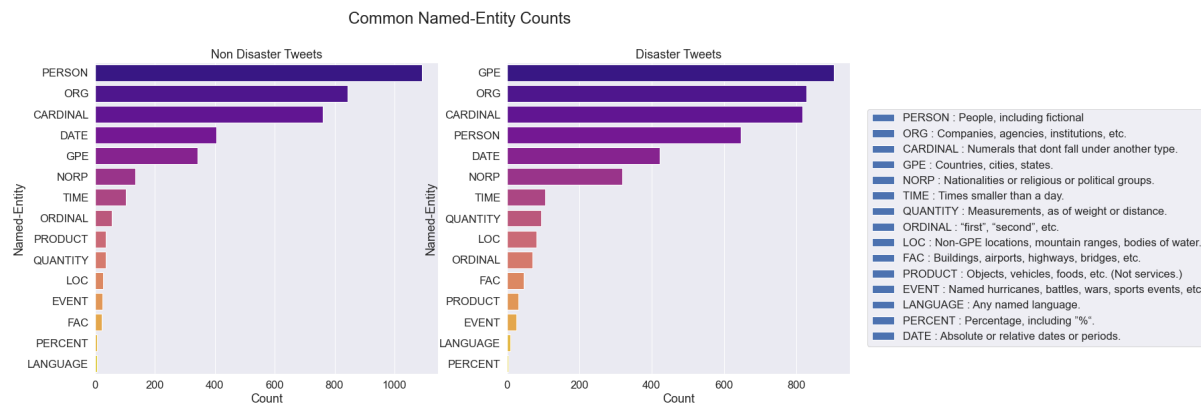


Figure 6. Common Name-Entity Counts

Model

What is BERT and why do we choose it?

BERT is the state-of-the-art language model with impressive accuracy for multiple Natural Language Processing tasks. Previous models have a hard time learning long sentences. They suffer from short-term memory and the vanishing gradient due to the nature of back-propagation. BERT overcomes this weakness by utilizing transformers and it performs well for Tweet classification tasks. After comparing previous solutions for the problem, we find that BERT performs the best among other models such as RNN and LSTM.

What is BERTweet?

According to Liu et al. (2019), Lee et al. (2020), and Canete et al. (2020), BERT can be improved by pre-training with context-specific data. For our problem, we find BERTweet, a BERT model pre-trained with tweets instead of general text data such as Wikipedia.

Model architecture

We import the BERT-base-uncased model from the Hugging Face library. It has 12 layers and 110M parameters. The following picture shows a simplified architecture of a fine-tuned

BERT-base classifier. The pre-trained BERTweet model is also available from the Hugging Face library. It has 12 layers and 135M parameters.

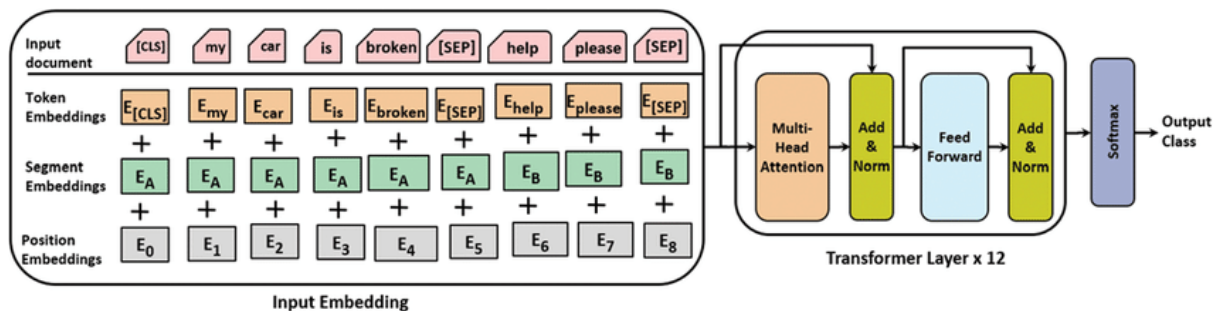


Figure 7. Fine-tuned BERT-base classifier

Fine-tune the pre-trained model

data_visualization_first_modeling.ipynb is used for fine-tuning of BERT with the old train set and the new test set. **run_twitter_classification.py** is used for fine-tuning of BERT and BERTweet with the new train set and the new test set. Model is fine-tuned based on the task of text classification while the loss is based on binary cross entropy. We run the model on GPU from Google Colab to train the model. Since each text in our dataset is relatively short between 10 to 20 tokens, we used the smart batch padding (to the maximum length of the tokens in one batch) on the fly instead of pre-padding to max length 512. This can help speed up the training process.

Thanks to the Hugging Face library and WandB, a central dashboard to keep track of hyperparameters, system metrics, and predictions, we are able to fine-tune and test different models effectively with command-line arguments. That's why we decide to use

run_twitter_classification.py to fine-tune BERT and BERTweet with the new train set.

```
!python3 run_twitter_classification.py \
  --model_name_or_path vinai/bertweet-base \
  --train_file ./joined_train.csv \
  --validation_file ./joined_test.csv \
  --do_train \
  --do_eval \
  --max_seq_length 128 \
  --per_device_train_batch_size 64 \
  --per_device_eval_batch_size 64 \
  --learning_rate 2e-5 \
  --num_train_epochs 3 \
  --output_dir ./model/ \
  --evaluation_strategy epoch \
  --dataloader_drop_last \
  --overwrite_output_dir \
  --logging_steps 5 \
  --pad_to_max_length False
```

Figure 8.

Model performance

We collect the following metrics from WandB. The train set for the first model is the train set of the first dataset commonly used by other researchers. It contains only 7613 samples. The train sets for the other two models are the new train set which we explain in the data overview section. The test set is the same for all three models.

BERT-Base trained with the train set of the first dataset (7613 samples)

BERT-Base (Total time: 820.000s)

Epoch/Metric	Training Loss	Test Loss	Test Accuracy	Test F1
1	0.339	0.449	0.818	0.708
2	0.337	0.449	0.818	0.708
3	0.334	0.449	0.818	0.708

The following models were trained with the new train set (13289 samples)

BERT-Base (Total time: 744.282s)

Epoch/Metric	Training Loss	Test Loss	Test Accuracy	Test F1
1	0.371	0.350	0.857	0.713
2	0.276	0.343	0.863	0.744
3	0.196	0.383	0.856	0.742

BERTweet-Base (Total time: 770.702s)

Epoch/Metric	Training Loss	Test Loss	Test Accuracy	Test F1
1	0.386	0.358	0.861	0.710
2	0.331	0.341	0.864	0.740
3	0.191	0.363	0.869	0.745

Analysis and Conclusion

BERT-Base(7613 samples) vs BERT-Base(13289 samples)

If we compare the best results, the accuracy increases 3.8 percentage points after we train with more data while the F1 score increases by 3.4 percentage points. BERT's test accuracy and F1 scores don't improve in any epoch. We suspect that this is due to the relatively large size of the test dataset(5695 samples) compared with the train dataset.

BERT-Base(13289 samples) vs BERTweet(13289 samples)

After comparing the best results, we find that the accuracy increases 0.6 percentage points and the F1score increases 0.1 percentage points.

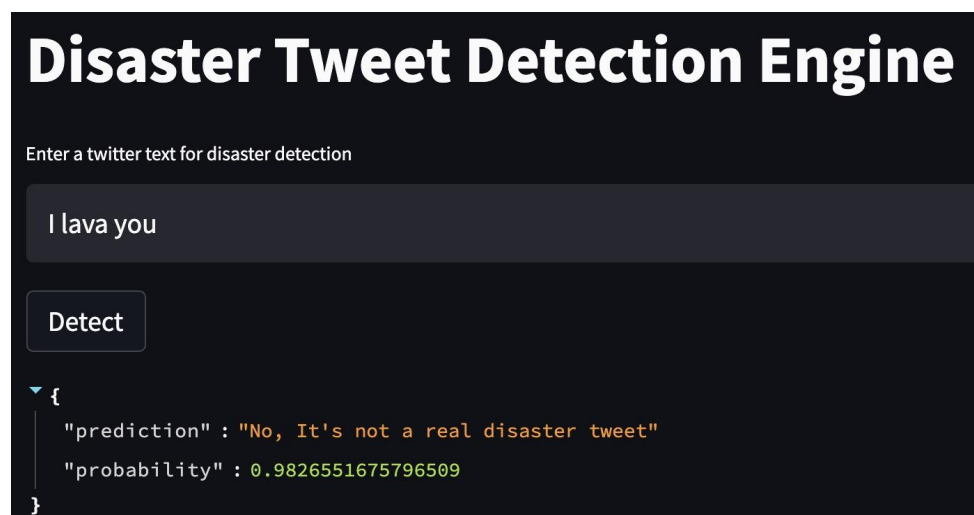
Our projects show that more data brings better results. The benefit of data even outweighs the improved model in our case. We believe that with more data, the BERT and BERTweet models can keep improving. After all, the number of our samples(13289) is 1/10000 of the number of BERT's parameters (110M).

Web Application

We have built a web application with Streamlit to detect the real disaster tweet based on our fine-tuned model. Users can type in a tweet context and hit the Detect button to detect whether it is about a real disaster. Moreover, the application would also output the probability. Here are two examples:



```
{
  "prediction": "Yes, It's a real disaster tweet"
  "probability": 0.9833039045333862
}
```



```
{
  "prediction": "No, It's not a real disaster tweet"
  "probability": 0.9826551675796509
}
```

Reference

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Nguyen, D. Q., Vu, T., & Nguyen, A. T. (2020). BERTweet: A pre-trained language model for English Tweets. *arXiv preprint arXiv:2005.10200*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

<https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>

<https://www.kaggle.com/gunesevitan/nlp-with-disaster-tweets-eda-cleaning-and-bert#5.-Mislabeled-Samples>

<https://www.kaggle.com/c/nlp-getting-started>

<https://www.kaggle.com/vstepanenko/disaster-tweets>

<https://www.pewresearch.org/fact-tank/2013/10/28/twitter-served-as-a-lifeline-of-information-during-hurricane-sandy/>

<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

<https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9>