MSDS 458 AI and ML

Final Paper Deep Learning with Twitter

Danish S. Moti

Northwestern University

Abstract

According to the Pew Research Center, almost seven in ten Twitter users get their news from the site, the author of this research paper is one of those seven. In ML, a tremendous amount of research is done on Twitter sentiment analysis. The sentiment analyses can code a tweet based on a positive or negative tone. While researching the many datasets on Twitter sentiment analysis, I discovered that I could tell what timeframe these Tweets were created based on the topics the users discussed. Mainly, I could read that the sample tweets overlapped with some national tragedies. Often the reports from Twitter outpace the statements from professional journalists and first responders because the citizen journalists emersed in the event experience the circumstances as they unfold.
The purpose of this assignment is to create a training model that can most accurately discover the presence of a disaster and code what type of disaster is taking place.

Introduction

Responses to national and local tragedies often require coordination between governments and private companies. The business case for this research assignment is that a better algorithm for detecting disasters can save precious time for the government and companies that must address these disasters. At crises, seconds sometimes separate low casualty counts from higher casual counts.
Suppose citizen journalists on Twitter can accurately share information about a breaking news event. In that case, those tasked with responding to these events must read Twitter accounts of what is taking place and the official outlets of reporting tragedies. Therefore, this research assignment aims to train and model data on disasters to discover what deep machine learning algorithm we should rely on.

Literature review

Governments have spent Trillions of dollars trying to make their country as safe as possible. While expensive tools can detect the rumbling of the earth or rising waves in the ocean, a much cheaper option has been to use Twitter as a sensor for natural and non-natural disasters. A previous study has gone even further than to note the presence of a natural disaster to pinpoint precisely where the citizens needed help (Hernandez-Suarez et al., 2019). Hernandez-Suarez et al. (2019) used Named Entity Recognition (NER) to extract named-entity pairs to discover geotags. The authors of that study also show the difficulty in deciphering what a disaster is and what is not. Companies like Mayday.ai now actively mine Tweets to detect the presence of natural disasters. Mayday.ai can position satellites and traffic cameras to observe what users have tweeted about remotely.

A second study on the use of Twitter during natural disasters shows that researchers can sift through Tweets to get particular pieces of information that can help first responders allocate their limited resources (Alam et al., 2018). Alam et al. (2018) provide a stratified report on how three hurricanes impacted the Gulf Coast in 2017. They took images shared by Twitter users and classified them based on the damage seen in the photos (none-sever). They could also classify damage as damage to personal property or infrastructure and utilities. This approach of categorizing Tweets in a disaster also helped the researchers track the public sentiment after the disaster had passed. This light literature review shows that advanced methods are applied to public Tweets to learn what is happening in the world.

Methods

Good NLP research starts with good data that is cleaned and prepared for experimentation. The raw data holds over 11,000 Tweets associated with disaster-related events like *fire*, *outbreak*, *bomb*, etc. The tweets are worldwide and were collected on January 14th, 2020 (Kaggle, 2021). After collection, the Tweets were manually classified as actual disaster events or not. Unfortunately, there is not too much more information about this dataset. Many researchers on Kaggle and GitHub have used this data set, but they cannot tell the reader much more than the sentences that I have previously mentioned about the data. To learn more about the dataset, I run some basic EDA to explore the data before experimentation. Table 1 shows the distribution of disasters and non-disasters Tweets in the training data set. There are roughly 4500 non-disaster Tweets and 3300 disaster-related tweets. Table 2 shows the percentage of disaster Tweets by country. The distribution of Tweets by country shows two things I could have predicted. First, I expected that the distribution would favor those places in the world where internet connectivity is far-reaching.

Second, I expected the Tweets would come from areas with a significant disaster before January 14th, 2020. Table 2 shows exactly what I expected. The countries on the list are generally places known to have greater internet connectivity and experienced a disaster close to when the Tweets were collected. Alam et al. (2018) also showed that the top term frequencies change significantly based on the date the Tweets were collected. Turning to the dataset in this paper, January 14th, 2020, is around when COVID was spreading; there were wildfires in Australia, there was increased discussion about a war with Iran, and volcano eruption in the Philippines.

The next preprocessing step in NLP is to clean and prepare the data. Reading might be easy for the human eye, but the Tweets must be turned into numerical vectors to make the Tweets easy for the computer to read. Tokenization counts everything that occurs between spaces as a unique entity. The tokens then fill out a sparse matrix of 1s and 0s that light up as a 1, when the unique token appears. After tokenization, it is essential to remove stopwords or the smaller words in a sentence that do not hold any interpretable meaning. Table 3 shows a word cloud of the most frequently occurring words. The reader can see that stopwords have been removed. Had the stopwords not been removed, the most frequent words in the cloud would have words like *the*, *a*, and *it*.

The research design attempts to create an algorithm that accurately predicts disasters. I want to test the difference between Tweets that are disasters and Tweets that are not. Table 4 shows a comparison between Tweets that are classified as disasters and Tweets that are not. Table 4 shows that generally speaking, Tweets that are not disaster-related tend to be longer in word count than disaster-related Tweets.

The modeling also relies on tfidf-matrices. A Tfidf-vectorizer learns the unique vocabulary from one documents and tests if that vocabulary is a good predictor in another document. One might think that the most common words would indicate the most about the corpus of Tweets is about. However, these familiar words might just be common statements like "hello" or "bye," which really does not shed light on what the Tweet is about. I apply sklearn and a tfidf-vectorizer to create a supervised learning method. This method will train on one portion of the data and then be tested on another, sequestered set of data.

Results

The results are based on how accurately a model correctly predicts the presence of a disaster. The first test runs a support vector machine (SVM). The SVM classifier is trained on the training set of Tweets and returns an F1 score of 80%. Additional experimental models are run to compare. The first four tests are shown below.

Simple SVM					
		Precision	Recall	f1-score	support
	0	0.81	0.85	0.83	886
	1	0.78	0.72	0.75	637
accuracy				0.8	1523
macro avg		0.79	0.79	0.79	1523
weighted avg		0.8	0.8	0.8	1523

MultinomialNB				
	Precision	Recall	f1-score	support
0	0.72	0.9	0.8	3682
1	0.81	0.53	0.64	2790
accuracy			0.74	6472
macro avg	0.76	0.72	0.72	6472
weighted avg	0.76	0.74	0.73	6472

Logistic Regression				
	Precision	Recall	f1-score	support
0	0.69	0.95	0.8	3682
1	0.97	0.42	0.57	2790
accuracy			0.72	6472
macro avg	0.76	0.69	0.68	6472
weighted avg	0.76	0.72	0.7	6472

Random Forrest				
	Precision	Recall	f1-score	support
0	0.69	0.93	0.8	3682
1	0.84	0.45	0.59	2790
accuracy			0.72	6472
macro avg	0.77	0.69	0.69	6472
weighted avg	0.76	0.73	0.71	6472

This is still a surface-level finding. I turn to deep neural networks to delve deeper into what model accurately predicts disaster Tweets. I have created my "base" NN model with three layers and no regularization in the sequential model shown in Table 5. I also added a 1D Global average pooling to take a 2-dimensional tensor and resize it a single, maximum of all input sizes. In Table 6, I run a DNN with the same three dense layers. Table 7 shows another DNN with three dense layers and three dropout layers. I run the ReLU activation function in all of these models, which takes smaller numbers down to zero and larger numbers up to one.

One of the main functions of this assignment is to justify why layers are added or not. The addition of the first hidden layer determines the activation for the second layer. When there is an addition of a second or third layer, the model is considered a deeper model. In experiments 5 to 7, I adjust parameters to improve the accuracy score of the model. Generally, adding more layers and making the model deeper improved accuracy, but it also increased computational time. The best performing model here has been the three-layer DNN with three dropout layers. The dropout improved the model and prevented the problem of overfitting. Adding layers could have enhanced the model incrementally, but not by significant margins. Furthermore, the model will not always achieve greater accuracy with more layers depending on the dropout rate. This is similar to the finding I had in the earlier research assignment, one cannot blindly add more nodes, and layers expect the model to improve. I generally found that tweaking hyperparameters with three dense layers would allow me to get an accuracy score of around 88%. With the DNN with three layers and three dropouts, I was able to get the highest scores, around 88% accuracy.

Conclusions

The various experiments run in the research assignment show that Twitter is a reliable resource to observe the presence of disasters. One of the biggest challenges is to make sure that one pulls the correct timeframe of data from Twitter. As I showed early in my EDA, the disasters in my dataset depended on when the Tweets were pulled. Therefore, if one wanted to measure how to detect precisely hurricane information in Tweets, pulling tweets from hurricane season would serve the researcher the best. Alam et al. (2018) conducted their research on three hurricanes in one season. My EDA on that data set would have given me more information on flooding but would have excluded other disasters in the available dataset that I have used here.

A deep CNN, or a DNN, is the best model to measure how accurately a Tweet detects a disaster. For better data, I would also recommend adding images to see diasters. While this has been done in other research assignments, my computations and experience limitations did not allow me to process natural language about disasters and 2d images on catastrophe. The seven experiments and EDA that came before it show the wealth of information that one can extract from Twitter. The researcher needs to have a specific objective to look for in the Tweets or be inundated by the information.

Tables

Table 1: Disaster vs. Non-disasters (disaster = 1, non-disaster = 0)

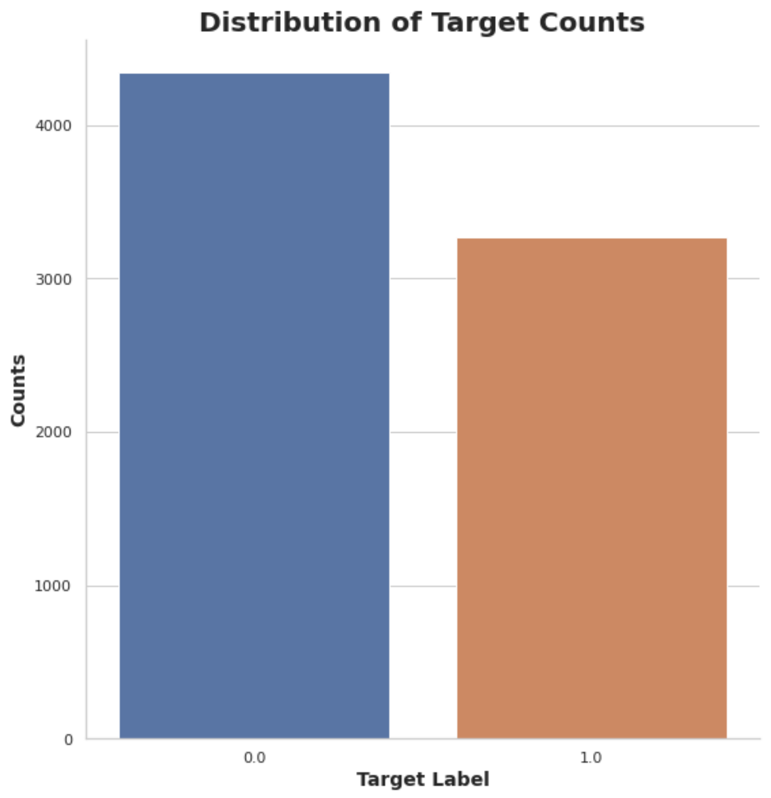


Table 2: Disaster Tweets per Country

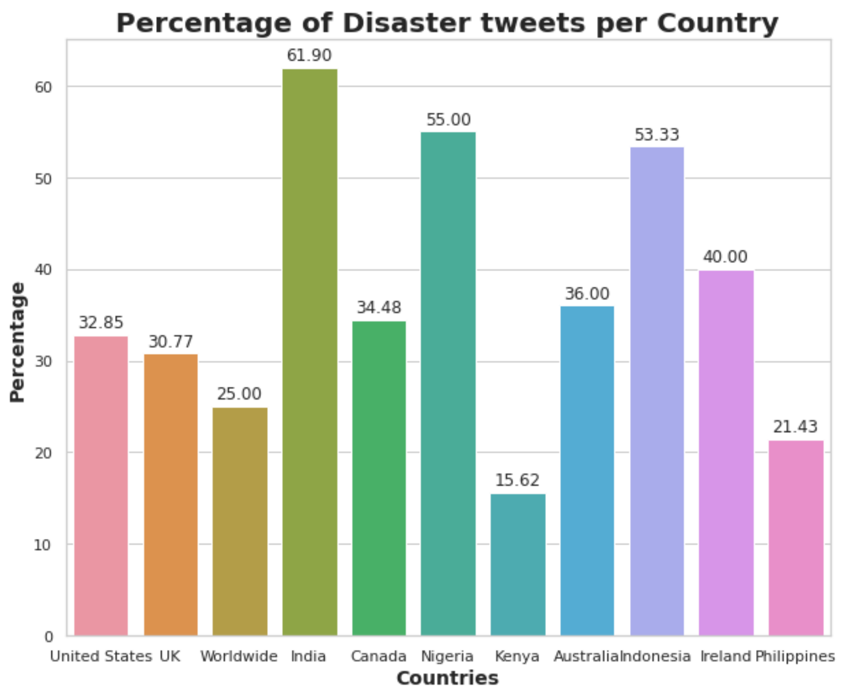


Table 3: Word cloud

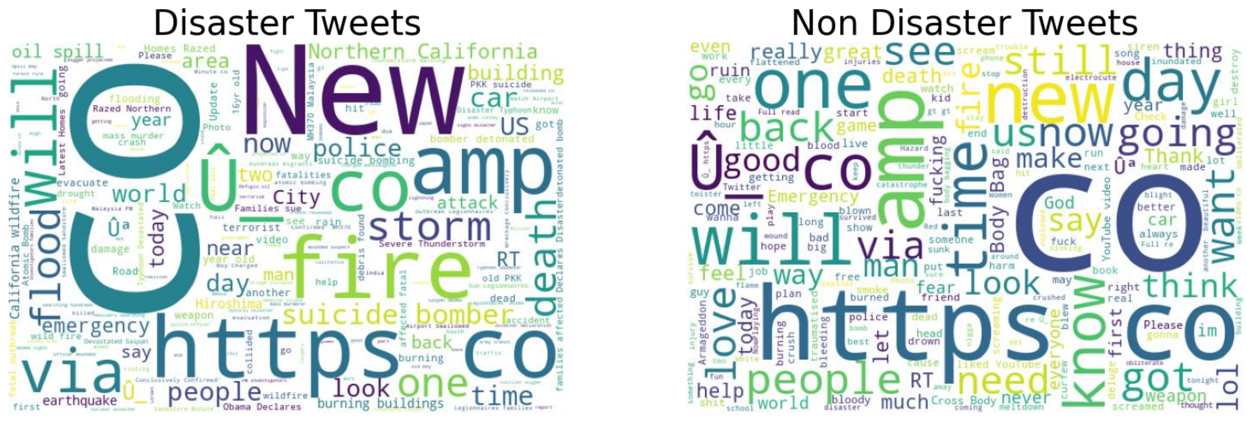


Table 4: Word count by disaster or not

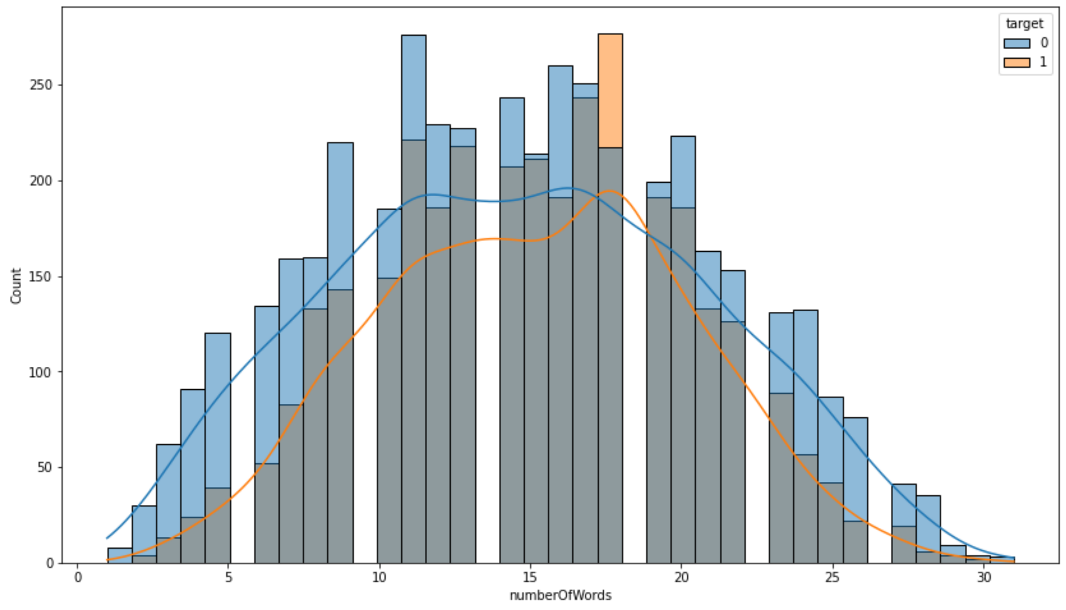
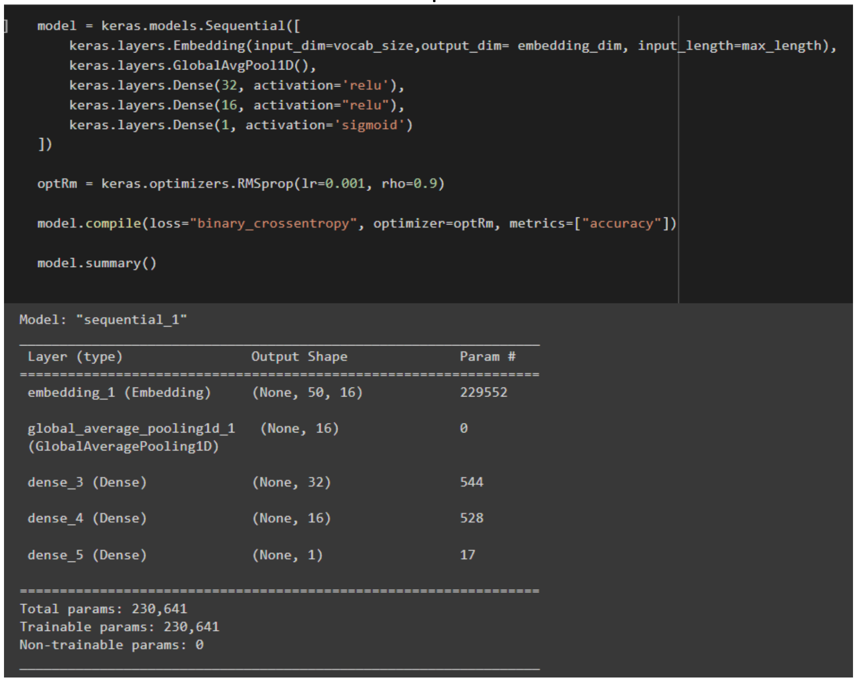


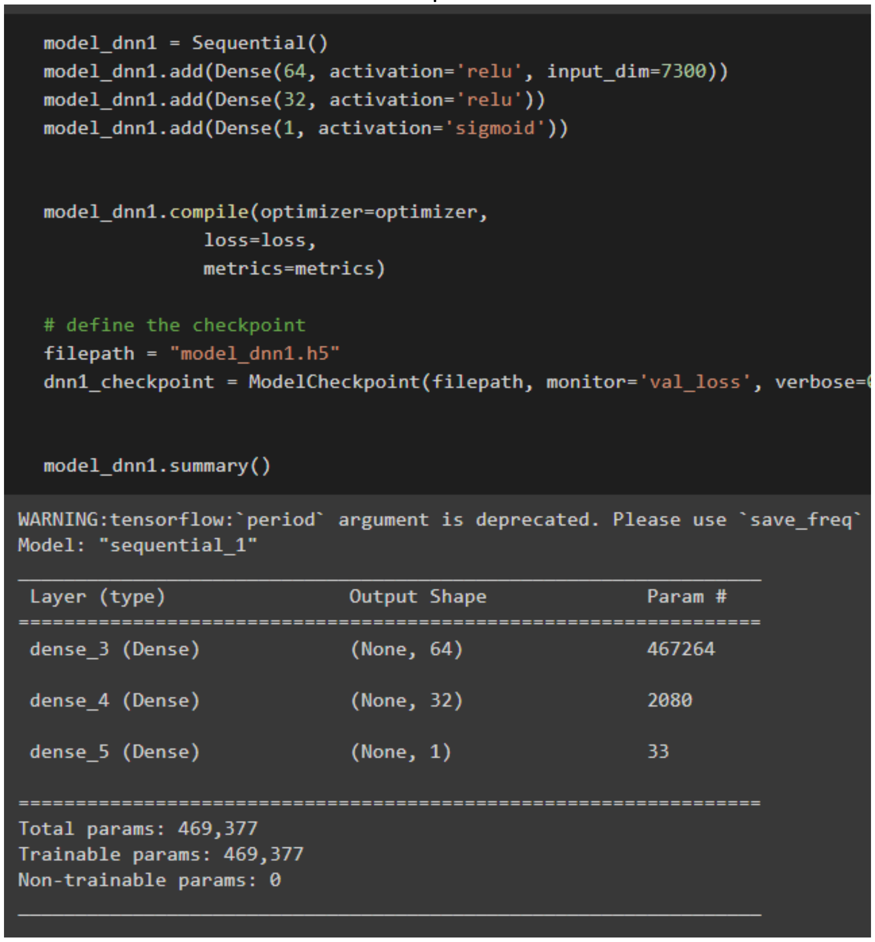
Table 5: Sequential NN



```
model = keras.models.Sequential([
    keras.layers.Embedding(input_dim=vocab_size,output_dim= embedding_dim, input_length=max_length),
    keras.layers.GlobalAvgPool1D(),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(16, activation="relu"),
    keras.layers.Dense(1, activation='sigmoid')
])

optRm = keras.optimizers.RMSprop(lr=0.001, rho=0.9)

model.compile(loss="binary_crossentropy", optimizer=optRm, metrics=["accuracy"])

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 50, 16)            229552

 global_average_pooling1d_1   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_3 (Dense)             (None, 32)                544

 dense_4 (Dense)             (None, 16)                528

 dense_5 (Dense)             (None, 1)                 17

=================================================================
Total params: 230,641
Trainable params: 230,641
Non-trainable params: 0
_________________________________________________________________
```

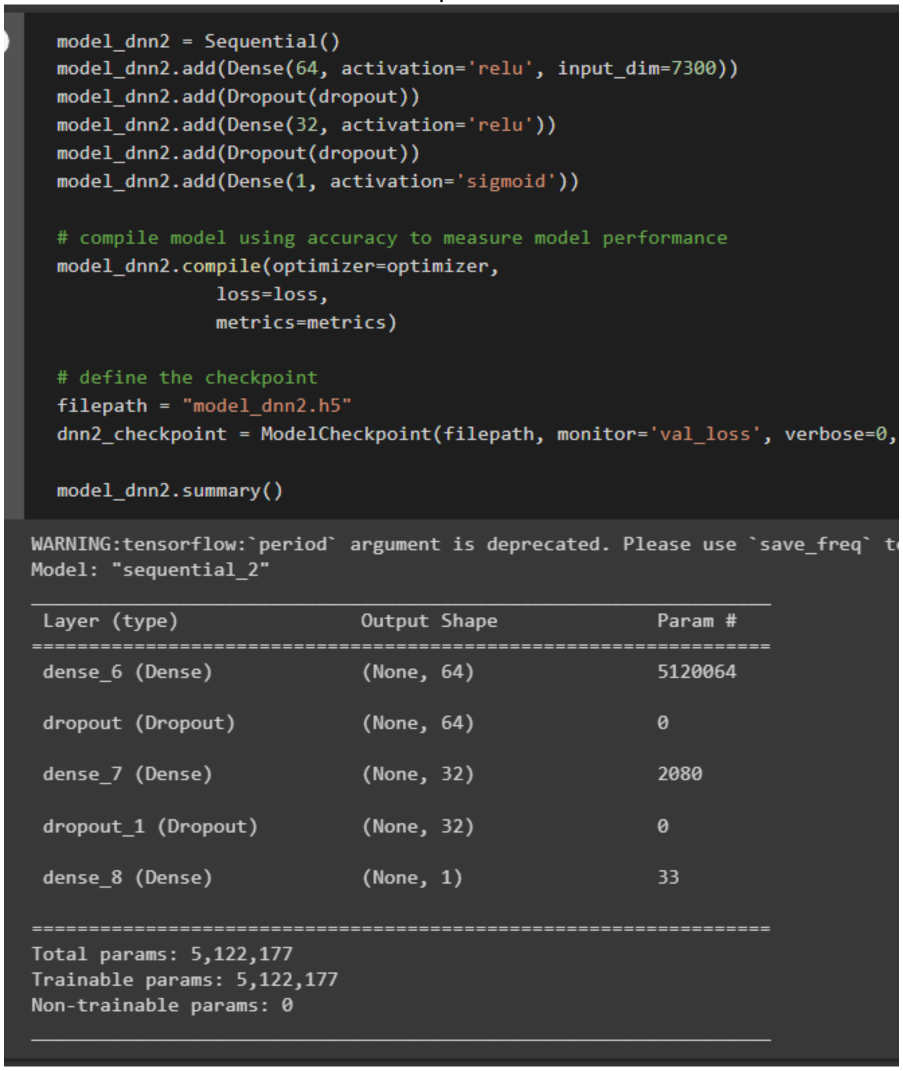
Table 6: Sequential DNN



```
model_dnn1 = Sequential()
model_dnn1.add(Dense(64, activation='relu', input_dim=7300))
model_dnn1.add(Dense(32, activation='relu'))
model_dnn1.add(Dense(1, activation='sigmoid'))


model_dnn1.compile(optimizer=optimizer,
              loss=loss,
              metrics=metrics)

# define the checkpoint
filepath = "model_dnn1.h5"
dnn1_checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=


model_dnn1.summary()
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq`
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 64)                467264

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 469,377
Trainable params: 469,377
Non-trainable params: 0
_________________________________________________________________
```

Table 7: Sequential DNN2



```
model_dnn2 = Sequential()
model_dnn2.add(Dense(64, activation='relu', input_dim=7300))
model_dnn2.add(Dropout(dropout))
model_dnn2.add(Dense(32, activation='relu'))
model_dnn2.add(Dropout(dropout))
model_dnn2.add(Dense(1, activation='sigmoid'))

# compile model using accuracy to measure model performance
model_dnn2.compile(optimizer=optimizer,
               loss=loss,
               metrics=metrics)

# define the checkpoint
filepath = "model_dnn2.h5"
dnn2_checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=0,

model_dnn2.summary()
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` t
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_6 (Dense)             (None, 64)                5120064   

 dropout (Dropout)           (None, 64)                0         

 dense_7 (Dense)             (None, 32)                2080      

 dropout_1 (Dropout)         (None, 32)                0         

 dense_8 (Dense)             (None, 1)                 33        

=================================================================
Total params: 5,122,177
Trainable params: 5,122,177
Non-trainable params: 0
_________________________________________________________________
```

References

Alam, F., Ofli, F., & Imran, M. (2018). *A Twitter Tale of Three Hurricanes: Harvey, Irma, and Maria*.

Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, K., Perez-Meana, H., Portillo-Portillo, J., Sanchez, V., & García Villalba, L. J. (2019). Using Twitter Data to Monitor Natural Disaster Social Dynamics: A Recurrent Neural Network Approach with Word Embeddings and Kernel Density Estimation. In *Sensors* (Vol. 19, Issue 7). https://doi.org/10.3390/s19071746

Kaggle. (2021). *Disaster Tweets: Real or Not? NLP with Disaster Tweets challenge add-on*. Kaggle.Com. https://www.kaggle.com/vstepanenko/disaster-tweets