Installation

```
!pip install --upgrade tensorflow
!pip install --upgrade scikit-learn
!pip install -q tfds-nightly tensorflow matplotlib
!pip install nltk
```

```
    Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.
    Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: gast<0.5.0,>=0.2.1 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in /usr/local/lib/p
    Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in /usr/local/lib/py
    Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: wheel<1.0,>=0.32.0 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: flatbuffers<3.0,>=1.12 in /usr/local/lib/python3.7/dist-
    Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: tensorboard~=2.6 in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/d
    Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: keras<2.8,>=2.7.0rc0 in /usr/local/lib/python3.7/dist-pa
    Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-p
    Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.
    Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pytho
    Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/
    Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-p
    Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-
    Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist
    Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-pa
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/li
    Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pa
    Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (
```

```
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import tensorflow as tf
import nltk
import sklearn
import string

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras import optimizers
from tensorflow.keras import losses
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mou
```

```
# Import other common libraries
#from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import itertools
import re
```

```
trainfile = "/content/drive/MyDrive/ColabNotebooks/Covid19_test.csv"
testfile = "/content/drive/MyDrive/ColabNotebooks/Covid19_test.csv"
```

```
# Import Train data set
df_train = pd.read_csv(trainfile,
                encoding='latin-1')
df_train.head()
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
				02-03-	#Panic buying hits #NewYork City	

```
# Import Test data set
df_test = pd.read_csv(testfile,
                encoding='latin-1',
                low_memory=False)
df_test.head()
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
				02-03-	#Panic buying hits #NewYork City	

```
#train - choose only 3 columns
df_train = df_train[['TweetAt','OriginalTweet','Sentiment']]
df_train = df_train.dropna(subset=['OriginalTweet'])
df_train.head()
```

	TweetAt	OriginalTweet	Sentiment
0	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	Extremely Negative
1	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	Positive
2	02-03-2020	Find out how you can protect yourself and love...	Extremely Positive
3	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	Negative
4	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	Neutral

```
#test - choose only 2 columns and 1 cols for sentiment
df_test = df_test[['TweetAt','OriginalTweet']]
df_test = df_test.dropna(subset=['OriginalTweet'])
df_test.loc[:,'Sentiment'] = np.nan
df_test.head()
```

	TweetAt	OriginalTweet	Sentiment
0	02-03-2020	TRENDING: New Yorkers encounter empty supermar...	NaN
1	02-03-2020	When I couldn't find hand sanitizer at Fred Me...	NaN
2	02-03-2020	Find out how you can protect yourself and love...	NaN
3	02-03-2020	#Panic buying hits #NewYork City as anxious sh...	NaN
4	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav...	NaN

Split Dataset

```
train_df, val_df = np.split(df_train.sample(frac=1), [int(0.9*len(df_train))])
test_df = df_test
print(len(train_df), 'training examples')
print(len(val_df), 'validation examples')
print(len(test_df), 'test examples')
```

```
3418 training examples
380 validation examples
3798 test examples
```

```
#train_df_sample = train_df.sample(400000)
train_df_sample = train_df
X_train = train_df_sample.drop("Sentiment", axis=1) # drop labels for training set
y_train = train_df_sample["Sentiment"].copy()


#val_df_sample = val_df.sample(32000)
val_df_sample = val_df
X_val = val_df_sample.drop("Sentiment", axis=1) # drop labels for validation set
y_val = val_df_sample["Sentiment"].copy()


#test_df_sample = test_df.sample(20000)
test_df_sample = test_df
X_test = test_df_sample.drop("Sentiment", axis=1) # drop labels for test set
y_test = test_df_sample["Sentiment"].copy()


print("X_train: " + str(X_train.shape) + str(type(X_train)))
print("y_train: "+ str(y_train.shape)+ str(type(y_train)))
print("X_val: " + str(X_val.shape) + str(type(X_val)))
```

```
print("y_val: "+ str(y_val.shape)+ str(type(y_val)))
print("X_test: " + str(X_test.shape) + str(type(X_test)))
print("y_test: "+ str(y_test.shape)+ str(type(y_test)))
```

```
X_train: (3418, 2)<class 'pandas.core.frame.DataFrame'>
y_train: (3418,)<class 'pandas.core.series.Series'>
X_val: (380, 2)<class 'pandas.core.frame.DataFrame'>
y_val: (380,)<class 'pandas.core.series.Series'>
X_test: (3798, 2)<class 'pandas.core.frame.DataFrame'>
y_test: (3798,)<class 'pandas.core.series.Series'>
```

Preprocessing

```
# Punctuation Removal
punctuation_removal = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    # remove https links
    clean_tweet = re.sub(r'http\S+', '', text)
    # remove username
    clean_tweet = re.sub('@[^\s]+','', clean_tweet)
    # convert text to lowercase
    clean_tweet = clean_tweet.lower()
    # remove numbers
    clean_tweet = re.sub('\d', ' ', clean_tweet)
    # remove whitespaces
    clean_tweet = ' '.join(clean_tweet.split())

    return clean_tweet.translate(str.maketrans('', '', punctuation_removal))


X_train["OriginalTweet"] = X_train["OriginalTweet"].apply(lambda text: remove_punctuation(te

X_val["OriginalTweet"] = X_val["OriginalTweet"].apply(lambda text: remove_punctuation(text))

X_test["OriginalTweet"] = X_test["OriginalTweet"].apply(lambda text: remove_punctuation(text


#remove stopwords
stop_words = set(stopwords.words('english'))
X_train["OriginalTweet"] = X_train["OriginalTweet"].apply(lambda x: ' '.join([word for word i
X_val["OriginalTweet"] = X_val["OriginalTweet"].apply(lambda x: ' '.join([word for word in x.
X_test["OriginalTweet"] = X_test["OriginalTweet"].apply(lambda x: ' '.join([word for word in


vocab_size = 50000
embedding_dim = 16
max_length = 100
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
```

Label and Tweet Encoding

```
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_val = label_encoder.fit_transform(y_val)
y_test = label_encoder.fit_transform(y_test)
y_train[:5]
```

```
array([0, 4, 2, 0, 3])
```

```
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(X_train['OriginalTweet'])
word_index = tokenizer.word_index

train_sequences = tokenizer.texts_to_sequences(X_train['OriginalTweet'])
train_padded = pad_sequences(train_sequences, maxlen=max_length, padding=padding_type, trunca

val_sequences = tokenizer.texts_to_sequences(X_val['OriginalTweet'])
val_padded = pad_sequences(val_sequences, maxlen=max_length, padding=padding_type, truncating

test_sequences = tokenizer.texts_to_sequences(X_test['OriginalTweet'])
test_padded = pad_sequences(test_sequences, maxlen=max_length, padding=padding_type, truncati


# Need this block to get it to work with TensorFlow 2.x
training_padded = np.array(train_padded)
training_labels = np.array(y_train)

val_padded = np.array(val_padded)
val_labels = np.array(y_val)

testing_padded = np.array(test_padded)
testing_labels = np.array(y_test)
```

Model

```
model = tf.keras.Sequential([
  tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

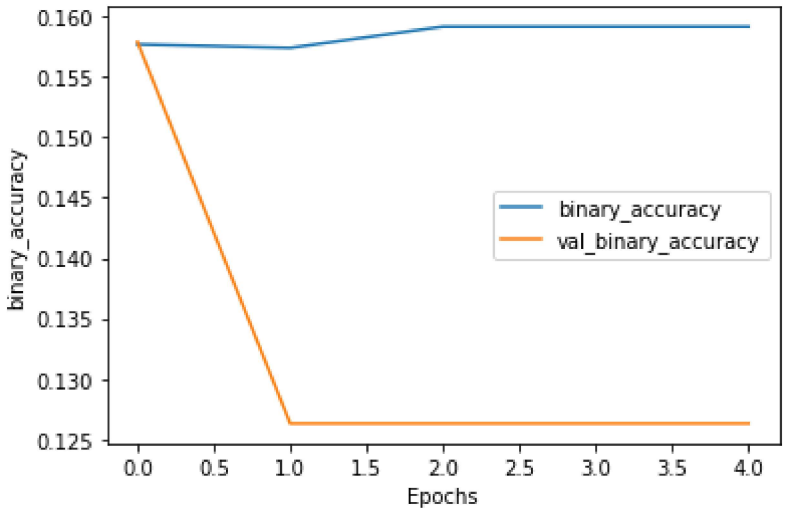
```
=================================================================
 embedding_2 (Embedding)     (None, 100, 16)           800000

 global_average_pooling1d_2   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_4 (Dense)             (None, 6)                 102

 dense_5 (Dense)             (None, 1)                 7

=================================================================
Total params: 800,109
Trainable params: 800,109
Non-trainable params: 0
_________________________________________________________________
```

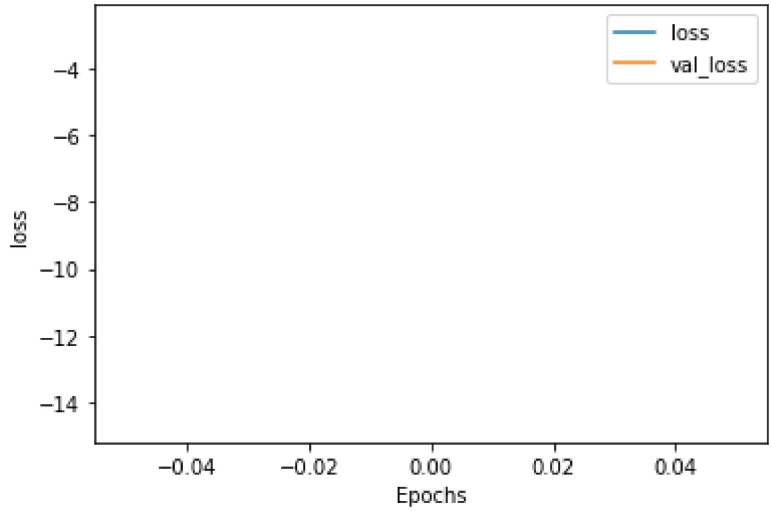
```
model.compile(loss=losses.BinaryCrossentropy(),
              optimizer='sgd',
              metrics=tf.metrics.BinaryAccuracy())
```

```
num_epochs = 5
history = model.fit(training_padded,
                    training_labels,
                    epochs=num_epochs,
                    #batch_size=128,
                    validation_data=(val_padded, val_labels),
                    verbose=1)
```

```
Epoch 1/5
107/107 [==============================] - 1s 4ms/step - loss: -2.7083 - binary_accurac
Epoch 2/5
107/107 [==============================] - 0s 2ms/step - loss: nan - binary_accuracy: 0
Epoch 3/5
107/107 [==============================] - 0s 3ms/step - loss: nan - binary_accuracy: 0
Epoch 4/5
107/107 [==============================] - 0s 2ms/step - loss: nan - binary_accuracy: 0
Epoch 5/5
107/107 [==============================] - 0s 2ms/step - loss: nan - binary_accuracy: 0
```

```
def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()

plot_graphs(history, "binary_accuracy")
plot_graphs(history, "loss")
```






```
predictions = model.predict(test_padded)
print(predictions)
```

```
[[nan]
 [nan]
 [nan]
 ...
 [nan]
 [nan]
 [nan]]
```

```
df = pd.DataFrame(predictions, columns=['predictions'])
df.to_csv('prediction-remSW.csv', index=False)
```

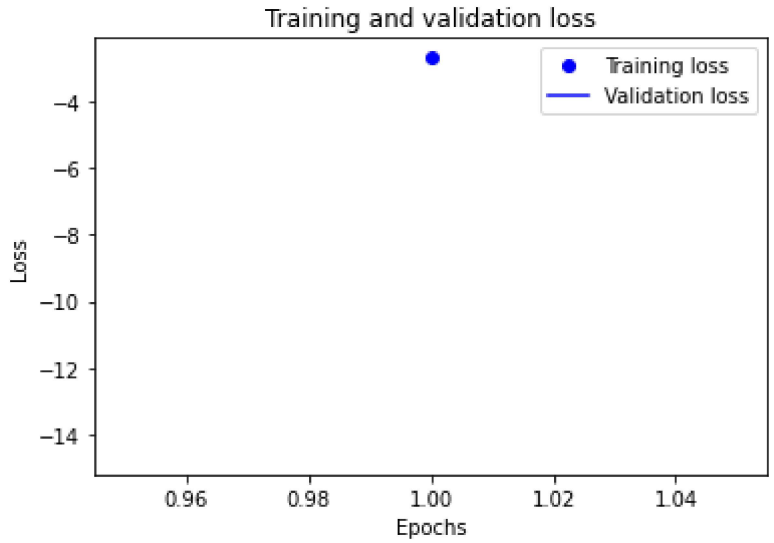
```
loss, accuracy = model.evaluate(testing_padded, testing_labels)

print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

```
119/119 [==============================] - 0s 1ms/step - loss: nan - binary_accuracy: 1
Loss:  nan
Accuracy:  1.0
```

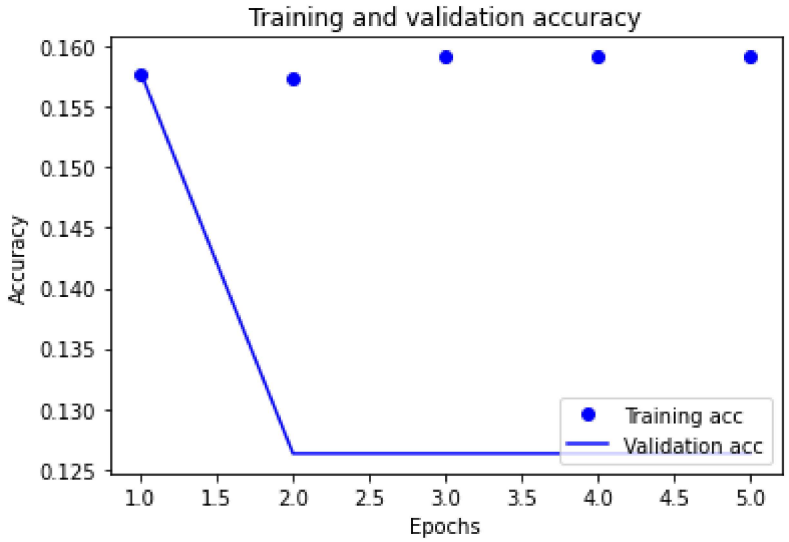
```
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'binary_accuracy', 'val_loss', 'val_binary_accuracy'])
```

```
acc = history_dict['binary_accuracy']
val_acc = history_dict['val_binary_accuracy']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')

plt.show()
```


Training and validation accuracy
Accuracy
Epochs
Training acc
Validation acc
0.160
0.155
0.150
0.145
0.140
0.135
0.130
0.125
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0