

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DL ASSINGMENT 2: Real or Not? NLP with Disaster Tweets

Submitted by:

Aditya Vashista

101703039

avashista_be17@thapar.edu

+91-9212291117

COE-02



THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

PATIALA-147001, PUNJAB

TABLE OF CONTENTS

| | |
|--------------------------|----|
| Introduction | 2 |
| Methodology | 3 |
| Python Script/Code | 5 |
| Output | 13 |

INTRODUCTION

Real or Not? NLP with Disaster Tweets; Predict which Tweets are about real disasters and which ones are not

Twitter has become an important communication channel in times of emergency. The iniquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

Hence in this Kaggle challenge the aim is to identify with given tweet text whether the given tweet is about a disaster/an accident or not.

We are allowed to use the deep learning approach only of RNN (LSTM/GRU)/Auto Encoders/GANS.

File descriptions

- train.csv - the training set, with 5 columns where 4 containing data regarding inputs for the model (id, keyword, location, text) and 1 output for validation.
- test.csv - the test set, with 4 columns containing data regarding inputs for the model (id, keyword, location, text).
- sample_submission.csv - a sample submission file in the correct format

METHODOLOGY

Since the input to the model is the text of the tweet, hence concepts of Natural Language Processing (NLP) have to be applied for data preprocessing and a suitable technique is to be used to convert the textual data into a suitable format (that is matrix of integers) for the deep learning model to understand.

The following preprocessing steps have been used:

- Removal of URLs
- Removal of tags
- Removal of Emojis
- Replacement of common abbreviations with full words of same exact meaning
- Removal of punctuations, numbers and extra spaces
- Removal of single character words (like a, I, etc.)
- Converting all the words to lower alphabets

After the preprocessing of sentences and getting reduced sentences the input data is to be converted in a bag of words and to be tokenized to matrix unique numbers where each number represents a unique word in the bag of words. This is done using `Tokenizer()` function included in tensorflow, keras library under preprocessing of texts library.

Since the structure of sentences that is arrangements of words in the sentence will matter significantly in determining the correct output, hence I used the approach of deep learning involving RNN (LSTM) and ANN using tensorflow backend.

Also, since we are dealing words, I have used an embedding layer; used for word embeddings and easy mapping of words to vectors and get similar words to the words used in training; as the first layer of each model used. The embedding layers gives its output as input to the first RNN layer in all models.

In order to make embedding matrix for the embedding layer, I have used pretrained GloVe pretrained word vectors, glove.6B.300d.txt

After training and testing several models I chose to take approach of ensemble modelling consisting of best 3 models.

The specification of the 3 models used in ensemble modelling are:

- Model 1: Embedding layer, followed by a single LSTM layer and the output layer

- Model 2: Embedding layer, followed by a single LSTM and ANN layer and the output layer
- Model 3: Embedding layer, followed by 2 LSTM and a single ANN layer and the output layer

Each model is trained with same input training data for 15 epochs in batch size of 128 on TPU of Google Collab.

Optimizer used: Adam

Activation function: for output layer: sigmoid; for hidden ANN layer: relu

Metrics: accuracy

Loss function: binary cross entropy

The validation of the models is done on the testing file for Kaggle challenge itself hence, validation score= Kaggle public score.

| Model | Training Loss | Training accuracy | Kaggle score |
|-----------------------|---------------|-------------------|--------------|
| <i>Model 1</i> | 0.2474 | 0.9166 | 0.80937 |
| <i>Model 2</i> | 0.3567 | 0.8551 | 0.81581 |
| <i>Model 3</i> | 0.3491 | 0.8575 | 0.81182 |
| <i>Ensemble Model</i> | - | - | 0.83941 |

Final Kaggle Score: 0.83941

| | | | |
|--|--|--------------|--------------|
| 9 submissions for Aditya Vashista #2 | | Sort by | Public Score |
| <div>AllSuccessfulSelected</div> | | | |
| Submission and Description | | Public Score | |
| submissionEnsemble.csv | | 0.83941 | |
| 2 hours ago by Aditya Vashista | | | |
| google colab ensemble approach result | | | |

Google colab link:

https://colab.research.google.com/drive/10ypRriMYfcBVDipW3Xy_un2USLYXUQBq?usp=sharing

PYTHON CODE/SCRIPT

```
# -*- coding: utf-8 -*-
```

```
"""DL ASSINGMENT 2: Real or Not? NLP with Disaster Tweets
```

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/10ypRriMYfcBVDipW3Xy_un2USLYXUQBq

```
# IMPORTING LIBRARIES
```

```
"""
```

```
import pandas as pd
```

```
import numpy as np
```

```
import re
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Embedding,LSTM,Dense,SpatialDropout1D
```

```
from tensorflow.keras.optimizers import Adam
```

```
"""# uploading datasets
```

```
"""
```

```
train=pd.read_csv("/content/drive/MyDrive/dpAssingment/assingment 2/train.csv")
```

```
test=pd.read_csv("/content/drive/MyDrive/dpAssingment/assingment 2/test.csv")
```

```
print(train.size,train.shape)
```

```
print(test.size,test.shape)
```

```
print(train.head(10))
```

```
print(len(train[train['target']==1]))
```

```
print(len(train[train['target']==0]))
```

```
""""# PRE PROCESSING OF DATA""""
```

```
def removeURL(text):
```

```
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)
```

```
def removeTags(text):
```

```
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)
```

```
def onlyWords(text):
```

```
    text = re.sub('[^a-zA-Z]', ' ', text)
    # Single character removal
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)
    # Removing multiple spaces
    text = re.sub(r'\s+', ' ', text)
    return text
```

```
def removeEmoji(text):
```

```
    emoji_pattern = re.compile("[
        u'\U0001F600-\U0001F64F' # emoticons
        u'\U0001F300-\U0001F5FF' # symbols & pictographs
        u'\U0001F680-\U0001F6FF' # transport & map symbols
        u'\U0001F1E0-\U0001F1FF' # flags (iOS)
        u'\U00002702-\U000027B0"
        u'\U000024C2-\U0001F251"
        "]" + , flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

```
def fullAbb(text):
```

```
    abbreviations = { "$" : " dollar ", "€" : " euro ", "4ao" : "for adults only", "a.m" : "before midday", "a3" :
    "anytime anywhere anyplace", "aamof" : "as a matter of fact", "acct" : "account", "adih" : "another day in hell",
    "afaic" : "as far as i am concerned", "afaict" : "as far as i can tell", "afaik" : "as far as i know", "afair" : "as far as i
    remember", "afk" : "away from keyboard", "app" : "application", "approx" : "approximately", "apps" :
    "applications", "asap" : "as soon as possible", "asl" : "age, sex, location", "atk" : "at the keyboard", "ave." :
    "avenue", "aymm" : "are you my mother", "ayor" : "at your own risk", "b&b" : "bed and breakfast", "b+b" : "bed
    and breakfast", "b.c" : "before christ", "b2b" : "business to business", "b2c" : "business to customer", "b4" :
    "before", "b4n" : "bye for now", "b@u" : "back at you", "bae" : "before anyone else", "bak" : "back at
```

keyboard", "bbbg" : "bye bye be good", "bbc" : "british broadcasting corporation", "bbias" : "be back in a second", "bbl" : "be back later", "bbs" : "be back soon", "be4" : "before", "bfn" : "bye for now", "blvd" : "boulevard", "bout" : "about", "brb" : "be right back", "bros" : "brothers", "brt" : "be right there", "bsaaw" : "big smile and a wink", "btw" : "by the way", "bwl" : "bursting with laughter", "c/o" : "care of", "cet" : "central european time", "cf" : "compare", "cia" : "central intelligence agency", "csl" : "can not stop laughing", "cu" : "see you", "cul8r" : "see you later", "cv" : "curriculum vitae", "cwot" : "complete waste of time", "cya" : "see you", "cyt" : "see you tomorrow", "dae" : "does anyone else", "dbmib" : "do not bother me i am busy", "diy" : "do it yourself", "dm" : "direct message", "dwh" : "during work hours", "e123" : "easy as one two three", "eet" : "eastern european time", "eg" : "example", "embm" : "early morning business meeting", "encl" : "enclosed", "encl." : "enclosed", "etc" : "and so on", "faq" : "frequently asked questions", "fawc" : "for anyone who cares", "fb" : "facebook", "fc" : "fingers crossed", "fig" : "figure", "fimh" : "forever in my heart", "ft." : "feet", "ft" : "featuring", "ftl" : "for the loss", "ftw" : "for the win", "fwiw" : "for what it is worth", "fyi" : "for your information", "g9" : "genius", "gahoy" : "get a hold of yourself", "gal" : "get a life", "gcse" : "general certificate of secondary education", "gfn" : "gone for now", "gg" : "good game", "gl" : "good luck", "glhf" : "good luck have fun", "gmt" : "greenwich mean time", "gmta" : "great minds think alike", "gn" : "good night", "g.o.a.t" : "greatest of all time", "goat" : "greatest of all time", "goi" : "get over it", "gps" : "global positioning system", "gr8" : "great", "gratz" : "congratulations", "gyal" : "girl", "h&c" : "hot and cold", "hp" : "horsepower", "hr" : "hour", "hrh" : "his royal highness", "ht" : "height", "ibrb" : "i will be right back", "ic" : "i see", "icq" : "i seek you", "icymi" : "in case you missed it", "idc" : "i do not care", "idgadf" : "i do not give a damn fuck", "idgaf" : "i do not give a fuck", "idk" : "i do not know", "ie" : "that is", "i.e" : "that is", "ifyp" : "i feel your pain", "IG" : "instagram", "iirc" : "if i remember correctly", "ilu" : "i love you", "ily" : "i love you", "imho" : "in my humble opinion", "imo" : "in my opinion", "imu" : "i miss you", "iow" : "in other words", "irl" : "in real life", "j4f" : "just for fun", "jic" : "just in case", "jk" : "just kidding", "jsyk" : "just so you know", "l8r" : "later", "lb" : "pound", "lbs" : "pounds", "ldr" : "long distance relationship", "lmao" : "laugh my ass off", "lmfao" : "laugh my fucking ass off", "lol" : "laughing out loud", "ltd" : "limited", "ltns" : "long time no see", "m8" : "mate", "mf" : "motherfucker", "mfs" : "motherfuckers", "mfw" : "my face when", "mofo" : "motherfucker", "mph" : "miles per hour", "mr" : "mister", "mrw" : "my reaction when", "ms" : "miss", "mte" : "my thoughts exactly", "nagi" : "not a good idea", "nbc" : "national broadcasting company", "nbd" : "not big deal", "nfs" : "not for sale", "ngl" : "not going to lie", "nhs" : "national health service", "nnn" : "no reply necessary", "nsfl" : "not safe for life", "nsfw" : "not safe for work", "nth" : "nice to have", "nvr" : "never", "nyc" : "new york city", "oc" : "original content", "og" : "original", "ohp" : "overhead projector", "oic" : "oh i see", "omdb" : "over my dead body", "omg" : "oh my god", "omw" : "on my way", "p.a" : "per annum", "p.m" : "after midday", "pm" : "prime minister", "poc" : "people of color", "pov" : "point of view", "pp" : "pages", "ppl" : "people", "prw" : "parents are watching", "ps" : "postscript", "pt" : "point", "ptb" : "please text back", "pto" : "please turn over", "qpsa" : "what happens", "#que pasa", "ratchet" : "rude", "rbl" : "read between the lines", "rlrt" : "real life retweet", "rofl" : "rolling on the floor laughing", "roflol" : "rolling on the floor laughing out loud", "roflmao" : "rolling on the floor laughing my ass off", "rt" : "retweet", "ruok" : "are you ok", "sfw" : "safe for work", "sk8" : "skate", "smh" : "shake my head", "sq" : "square", "srsly" : "seriously", "ssdd" : "same stuff different day", "tbh" : "to be honest", "tbs" : "tablespoonful", "tbsp" : "tablespoonful", "tfw" : "that feeling when", "thks" : "thank you", "tho" : "though", "thx" : "thank you", "tia" : "thanks in advance", "til" : "today i learned", "tl;dr" : "too long i did not read", "tldr" : "too long i did not read", "tmb" : "tweet me back", "tntl" : "trying not to laugh", "t tyl" : "talk to you later", "u" : "you", "u2" : "you too", "u4e" : "yours for ever", "utc" : "coordinated universal time", "w/" : "with", "w/o" : "without", "w8" : "wait", "wassup" : "what is up", "wb" : "welcome back", "wtf" : "what the fuck", "wtg" : "way to go", "wtpa" : "where the party at", "wuf" : "where are you from", "wuzup" : "what is up", "wywh" : "wish you were here", "yd" : "yard", "wyd" : "what are you doing",

"doin" : "doing", "ygrt" : "you got that right", "ynk" : "you never know", "zzz" : "sleeping bored and tired" }

l=text.split()

for i in range(len(l)):

if l[i].lower() in abbreviations.keys():

l[i]=abbreviations[l[i].lower()]

text=' '.join(l)

return text


```

def preProcessing(x):

    for i in range(len(x)):
        x[i]=removeURL(x[i])
        x[i]=removeTags(x[i])
        x[i]=removeEmoji(x[i])
        x[i]=fullAbb(x[i])
        x[i]=onlyWords(x[i])
        x[i]=x[i].lower()
    return x

x,y=train.iloc[:,3].values,train.iloc[:,4]
x_test=test.iloc[:,3].values
x=preProcessing(x)
x_test=preProcessing(x_test)

x[:5]

x_test[:5]

y

words,maxwords,maxlen=0,0,0
u=[]
for i in x:
    maxwords=maxwords if maxwords>len(i) else len(i)
    l=i.split()
    maxlen=maxlen if maxlen>len(l) else len(l)
    for j in l:
        if j not in u:
            u.append(j)
            words+=1
print(words)
print(maxwords)
print(maxlen)

```

```
""""# Tokenization""""
```

```
tokenizer=Tokenizer()
```

```
tokenizer.fit_on_texts(x)
```

```
x= tokenizer.texts_to_sequences(x)
```

```
x_test= tokenizer.texts_to_sequences(x_test)
```

```
x[:5]
```

```
x_test[:3]
```

```
vocab_size = len(tokenizer.word_index) + 1
```

```
print(vocab_size)
```

```
maxlen = 65
```

```
x = pad_sequences(x, padding='post', maxlen=maxlen)
```

```
x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)
```

```
""""# EMBEDDING""""
```

```
embeddings_dictionary = dict()
```

```
glove_file1 = open('/content/drive/MyDrive/dpAssingment/assingment 2/glove.6B.300d.txt', encoding="utf8")
```

```
for line in glove_file1:
```

```
    records = line.split()
```

```
    word = records[0]
```

```
    vector_dimensions = np.asarray(records[1:], dtype='float32')
```

```
    embeddings_dictionary [word] = vector_dimensions
```

```
glove_file1.close()
```

```
embedding_matrix = np.zeros((vocab_size, 300))
```

```
for word, index in tokenizer.word_index.items():
```

```
    embedding_vector = embeddings_dictionary.get(word)
```

```
    if embedding_vector is not None:
```

```
        embedding_matrix[index] = embedding_vector
```

```
embedding_layer = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=maxlen ,  
trainable=False)
```

```
"""# MODEL 1"""
```

```
model1 = Sequential()
model1.add(embedding_layer)
model1.add(LSTM(128))
model1.add(Dense(1, activation='sigmoid'))
model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model1.fit(x,y,epochs=15,batch_size=128)

model1.save('/content/drive/MyDrive/dpAssingment/assingment 2/model1.h5')

model1.summary()

sub=pd.read_csv("/content/drive/MyDrive/dpAssingment/assingment 2/sample_submission.csv")

ypred1=model1.predict(x_test)
ypred1.resize(len(ypred1))
ypred1=(ypred1>0.5)
for i in range(len(ypred1)):
    sub['target'][i]=int(ypred1[i])
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submission1.csv',index=False)

sub.head(10)
```

```
"""# MODEL 2"""
```

```
model2=Sequential()
model2.add(embedding_layer)
model2.add(SpatialDropout1D(0.2))
model2.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model2.add(Dense(units=10,activation="relu"))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model2.fit(x,y,epochs=15,batch_size=128)

model2.save("/content/drive/MyDrive/dpAssingment/assingment 2/model2.h5")
```

```

model2.summary()

ypred2=model2.predict(x_test)
ypred2.resize(len(ypred2))
ypred2=(ypred2>0.5)
for i in range(len(ypred2)):
    sub['target'][i]=int(ypred2[i])
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submission2.csv',index=False)
sub.head(10)

"""
# MODEL 3"""

model3=Sequential()
model3.add(embedding_layer)
model3.add(SpatialDropout1D(0.2))
model3.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2,return_sequences=True))
model3.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model3.add(Dense(units=10,activation="relu"))
model3.add(Dense(1, activation='sigmoid'))
model3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model3.fit(x,y,epochs=15,batch_size=128)

model3.save("/content/drive/MyDrive/dpAssingment/assingment 2/model3.h5")

model3.summary()

ypred3=model3.predict(x_test)
ypred3.resize(len(ypred3))
ypred3=(ypred3>0.5)
for i in range(len(ypred3)):
    sub['target'][i]=int(ypred3[i])
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submission3.csv',index=False)
sub.head(10)

"""# ENSEMBLE APPROACH"""

```

```
from statistics import mode
for i in range(len(ypred1)):
    sub['target'][i]=int(mode([ypred1[i],ypred2[i],ypred3[i]]))
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submissionEnsemble.csv',index=False)
sub.head(10)
```

OUTPUTS

Model summaries:

Model 1:

```
model1.summary()
```

```
Model: "sequential"
```

| Layer (type) | Output Shape | Param # |
|-----------------------|-----------------|---------|
| embedding (Embedding) | (None, 65, 300) | 4861200 |
| lstm (LSTM) | (None, 128) | 219648 |
| dense (Dense) | (None, 1) | 129 |

```
Total params: 5,080,977
```

```
Trainable params: 219,777
```

```
Non-trainable params: 4,861,200
```

Model 2:

```
model2.summary()
```

```
Model: "sequential_1"
```

| Layer (type) | Output Shape | Param # |
|------------------------------|-----------------|---------|
| embedding (Embedding) | (None, 65, 300) | 4861200 |
| spatial_dropout1d (SpatialDr | (None, 65, 300) | 0 |
| lstm_1 (LSTM) | (None, 128) | 219648 |
| dense_1 (Dense) | (None, 10) | 1290 |
| dense_2 (Dense) | (None, 1) | 11 |

```
Total params: 5,082,149
```

```
Trainable params: 220,949
```

```
Non-trainable params: 4,861,200
```

Model 3:

```
model3.summary()
```

Model: "sequential_2"

| Layer (type) | Output Shape | Param # |
|---------------------------------|-----------------|---------|
| embedding (Embedding) | (None, 65, 300) | 4861200 |
| spatial_dropout1d_1 (Spatial | (None, 65, 300) | 0 |
| lstm_2 (LSTM) | (None, 65, 128) | 219648 |
| lstm_3 (LSTM) | (None, 128) | 131584 |
| dense_3 (Dense) | (None, 10) | 1290 |
| dense_4 (Dense) | (None, 1) | 11 |
| Total params: 5,213,733 | | |
| Trainable params: 352,533 | | |
| Non-trainable params: 4,861,200 | | |

Uploading and data preprocessing:

```
print(train.size,train.shape)
print(test.size,test.shape)

print(train.head(10))

print(len(train[train['target']==1]))
print(len(train[train['target']==0]))
```

```
38065 (7613, 5)
13052 (3263, 4)
   id keyword  ...      text target
0    1    NaN  ...  Our Deeds are the Reason of this #earthquake M...      1
1    4    NaN  ...      Forest fire near La Ronge Sask. Canada      1
2    5    NaN  ...  All residents asked to 'shelter in place' are ...      1
3    6    NaN  ...  13,000 people receive #wildfires evacuation or...      1
4    7    NaN  ...  Just got sent this photo from Ruby #Alaska as ...      1
5    8    NaN  ...  #RockyFire Update => California Hwy. 20 closed...      1
6   10    NaN  ...  #flood #disaster Heavy rain causes flash flood...      1
7   13    NaN  ...  I'm on top of the hill and I can see a fire in...      1
8   14    NaN  ...  There's an emergency evacuation happening now ...      1
9   15    NaN  ...  I'm afraid that the tornado is coming to our a...      1

[10 rows x 5 columns]
3271
4342
```

```
x,y=train.iloc[:,3].values,train.iloc[:,4]
x_test=test.iloc[:,3].values
x=preProcessing(x)
x_test=preProcessing(x_test)
```

```
x[:5]
```

```
array(['our deeds are the reason of this earthquake may allah forgive us all',
      'forest fire near la ronge sask canada',
      'all residents asked to shelter in place are being notified by officers no other evacuation or shelter in place orders are expected',
      ' people receive wildfires evacuation orders in california',
      'just got sent this photo from ruby alaska as smoke from wildfires pours into school'],
      dtype=object)
```

```
x_test[:5]
```

```
array(['just happened terrible car crash',
      'heard about earthquake is different cities stay safe everyone ',
      'there is forest fire at spot pond geese are fleeing across the street cannot save them all',
      'apocalypse lighting spokane wildfires',
      'typhoon soudelor kills in china and taiwan'], dtype=object)
```

```
y
```

```
0      1
1      1
2      1
3      1
4      1
..
7608   1
7609   1
```

Tokenization:

```
words,maxwords,maxlen=0,0,0
u=[]
for i in x:
    maxwords=maxwords if maxwords>len(i) else len(i)
    l=i.split()
    maxlen=maxlen if maxlen>len(l) else len(l)
    for j in l:
        if j not in u:
            u.append(j)
            words+=1
print(words)
print(maxwords)
print(maxlen)
```

```
16203
156
31
```



```
tokenizer=Tokenizer()
tokenizer.fit_on_texts(x)
x= tokenizer.texts_to_sequences(x)
x_test= tokenizer.texts_to_sequences(x_test)
```

```
x[:5]
```

```
[[108, 4396, 19, 1, 835, 4, 16, 245, 129, 1556, 4397, 82, 35],
 [183, 40, 215, 716, 6501, 6502, 1168],
 [35,
  1686,
  1441,
  3,
  1864,
  2,
  664,
  19,
  127,
  6503,
  15,
  1687,
  36,
  373,
  246,
  57,
  1864,
  2,
  664,
  1346,
  19,
  1064],
 [54, 4398, 1442, 246, 1346, 2, 85],
 [29, 95, 1169, 16, 315, 17, 6504, 1688, 25, 263, 17, 1442, 6505, 66, 180]]
```

```
x_test[:3]
```

```
[[29, 881, 1884, 118, 88],  
 [468, 51, 245, 7, 1153, 2546, 590, 1984, 210],  
 [64, 7, 183, 40, 14, 793, 3410, 19, 4888, 837, 1, 717, 1355, 341, 93, 35]]
```

```
vocab_size = len(tokenizer.word_index) + 1  
print(vocab_size)  
maxlen = 65
```

```
16204
```

```
x = pad_sequences(x, padding='post', maxlen=maxlen)  
x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)
```

```
x[:3]
```

```
array([[ 108, 4396,   19,    1,  835,    4,   16,  245,  129, 1556, 4397,  
        82,   35,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0],  
       [ 183,   40,  215,  716, 6501, 6502, 1168,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0],  
       [  35, 1686, 1441,    3, 1864,    2,  664,   19,  127, 6503,   15,  
       1687,   36,  373,  246,   57, 1864,    2,  664, 1346,   19, 1064,
```

```
x_test[0]
```

```
array([ 29,  881, 1884,  118,   88,    0,    0,    0,    0,    0,    0,  
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
        0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0],  
      dtype=int32)
```

Model Training:

Model 1:

```
model1 = Sequential()
model1.add(embedding_layer)
model1.add(LSTM(128))
model1.add(Dense(1, activation='sigmoid'))
model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model1.fit(x,y,epochs=15,batch_size=128)
```

```
Epoch 1/15
60/60 [=====] - 15s 251ms/step - loss: 0.5613 - accuracy: 0.7109
Epoch 2/15
60/60 [=====] - 15s 250ms/step - loss: 0.4553 - accuracy: 0.8124
Epoch 3/15
60/60 [=====] - 15s 249ms/step - loss: 0.4112 - accuracy: 0.8304
Epoch 4/15
60/60 [=====] - 15s 250ms/step - loss: 0.3961 - accuracy: 0.8388
Epoch 5/15
60/60 [=====] - 15s 248ms/step - loss: 0.3837 - accuracy: 0.8449
Epoch 6/15
60/60 [=====] - 15s 248ms/step - loss: 0.3738 - accuracy: 0.8504
Epoch 7/15
60/60 [=====] - 15s 249ms/step - loss: 0.3900 - accuracy: 0.8413
Epoch 8/15
60/60 [=====] - 15s 249ms/step - loss: 0.3548 - accuracy: 0.8627
Epoch 9/15
60/60 [=====] - 15s 247ms/step - loss: 0.3323 - accuracy: 0.8739
Epoch 10/15
60/60 [=====] - 15s 252ms/step - loss: 0.3232 - accuracy: 0.8736
Epoch 11/15
60/60 [=====] - 15s 249ms/step - loss: 0.2979 - accuracy: 0.8881
Epoch 12/15
60/60 [=====] - 15s 249ms/step - loss: 0.2981 - accuracy: 0.8903
Epoch 13/15
60/60 [=====] - 15s 250ms/step - loss: 0.2698 - accuracy: 0.9057
Epoch 14/15
60/60 [=====] - 15s 253ms/step - loss: 0.2533 - accuracy: 0.9095
Epoch 15/15
60/60 [=====] - 15s 250ms/step - loss: 0.2474 - accuracy: 0.9166
<tensorflow.python.keras.callbacks.History at 0x7fbda56ae588>
```

Output:

| sub.head(10) | | |
|--------------|----|--------|
| | id | target |
| 0 | 0 | 1 |
| 1 | 2 | 1 |
| 2 | 3 | 1 |
| 3 | 9 | 1 |
| 4 | 11 | 1 |
| 5 | 12 | 0 |

Model 2:

```
model2=Sequential()  
model2.add(embedding_layer)  
model2.add(SpatialDropout1D(0.2))  
model2.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))  
model2.add(Dense(units=10,activation="relu"))  
model2.add(Dense(1, activation='sigmoid'))  
model2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])  
model2.fit(x,y,epochs=15,batch_size=128)
```

```
Epoch 1/15  
60/60 [=====] - 30s 493ms/step - loss: 0.5966 - accuracy: 0.6958  
Epoch 2/15  
60/60 [=====] - 30s 495ms/step - loss: 0.4950 - accuracy: 0.7825  
Epoch 3/15  
60/60 [=====] - 30s 493ms/step - loss: 0.4564 - accuracy: 0.8003  
Epoch 4/15  
60/60 [=====] - 29s 492ms/step - loss: 0.4484 - accuracy: 0.8011  
Epoch 5/15  
60/60 [=====] - 30s 504ms/step - loss: 0.4377 - accuracy: 0.8089  
Epoch 6/15  
60/60 [=====] - 31s 517ms/step - loss: 0.4212 - accuracy: 0.8170  
Epoch 7/15  
60/60 [=====] - 29s 486ms/step - loss: 0.4208 - accuracy: 0.8168  
Epoch 8/15  
60/60 [=====] - 29s 488ms/step - loss: 0.4096 - accuracy: 0.8219  
Epoch 9/15  
60/60 [=====] - 29s 487ms/step - loss: 0.3975 - accuracy: 0.8287  
Epoch 10/15  
60/60 [=====] - 29s 491ms/step - loss: 0.3928 - accuracy: 0.8327  
Epoch 11/15  
60/60 [=====] - 30s 492ms/step - loss: 0.3939 - accuracy: 0.8319  
Epoch 12/15  
60/60 [=====] - 29s 490ms/step - loss: 0.3786 - accuracy: 0.8365  
Epoch 13/15  
60/60 [=====] - 29s 491ms/step - loss: 0.3821 - accuracy: 0.8365  
Epoch 14/15  
60/60 [=====] - 29s 484ms/step - loss: 0.3556 - accuracy: 0.8566  
Epoch 15/15  
60/60 [=====] - 29s 489ms/step - loss: 0.3567 - accuracy: 0.8551  
<tensorflow.python.keras.callbacks.History at 0x7fbd20c7fd0>
```

Output:

```
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submission2.csv',index=False)  
sub.head(10)
```

| | id | target |
|---|----|--------|
| 0 | 0 | 1 |
| 1 | 2 | 1 |
| 2 | 3 | 1 |
| 3 | 9 | 1 |
| 4 | 11 | 1 |
| 5 | 12 | 0 |

Model 3:

```
model3=Sequential()  
model3.add(embedding_layer)  
model3.add(SpatialDropout1D(0.2))  
model3.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2,return_sequences=True))  
model3.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))  
model3.add(Dense(units=10,activation="relu"))  
model3.add(Dense(1, activation='sigmoid'))  
model3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])  
model3.fit(x,y,epochs=15,batch_size=128)
```

```
Epoch 1/15  
60/60 [=====] - 55s 918ms/step - loss: 0.5930 - accuracy: 0.7008  
Epoch 2/15  
60/60 [=====] - 52s 871ms/step - loss: 0.4852 - accuracy: 0.7873  
Epoch 3/15  
60/60 [=====] - 52s 863ms/step - loss: 0.4500 - accuracy: 0.8044  
Epoch 4/15  
60/60 [=====] - 52s 862ms/step - loss: 0.4418 - accuracy: 0.8069  
Epoch 5/15  
60/60 [=====] - 52s 862ms/step - loss: 0.4347 - accuracy: 0.8108  
Epoch 6/15  
60/60 [=====] - 52s 862ms/step - loss: 0.4224 - accuracy: 0.8145  
Epoch 7/15  
60/60 [=====] - 52s 863ms/step - loss: 0.4399 - accuracy: 0.8097  
Epoch 8/15  
60/60 [=====] - 52s 861ms/step - loss: 0.4154 - accuracy: 0.8200  
Epoch 9/15  
60/60 [=====] - 52s 859ms/step - loss: 0.4065 - accuracy: 0.8250  
Epoch 10/15  
60/60 [=====] - 52s 862ms/step - loss: 0.3997 - accuracy: 0.8292  
Epoch 11/15  
60/60 [=====] - 53s 879ms/step - loss: 0.3949 - accuracy: 0.8336  
Epoch 12/15  
60/60 [=====] - 52s 865ms/step - loss: 0.3769 - accuracy: 0.8404  
Epoch 13/15  
60/60 [=====] - 55s 919ms/step - loss: 0.3735 - accuracy: 0.8416  
Epoch 14/15  
60/60 [=====] - 52s 867ms/step - loss: 0.3711 - accuracy: 0.8437  
Epoch 15/15  
60/60 [=====] - 51s 854ms/step - loss: 0.3491 - accuracy: 0.8575  
<tensorflow.python.keras.callbacks.History at 0x7fbd098eac8>
```

Output:

```
sub.head(10)
```

| | id | target |
|---|----|--------|
| 0 | 0 | 1 |
| 1 | 2 | 1 |
| 2 | 3 | 1 |
| 3 | 9 | 0 |
| 4 | 11 | 1 |
| 5 | 12 | 0 |

Ensemble Approach Output:

```
from statistics import mode
for i in range(len(ypred1)):
    sub['target'][i]=int(mode([ypred1[i],ypred2[i],ypred3[i]]))
sub.to_csv('/content/drive/MyDrive/dpAssingment/assingment 2/submissionEnsemble.csv',index=False)
sub.head(10)
```

| | id | target |
|---|----|--------|
| 0 | 0 | 1 |
| 1 | 2 | 1 |
| 2 | 3 | 1 |
| 3 | 9 | 1 |
| 4 | 11 | 1 |
| 5 | 12 | 0 |
| 6 | 21 | 0 |
| 7 | 22 | 0 |
| 8 | 27 | 0 |
| 9 | 29 | 0 |

Kaggle Submissions:

9 submissions for [Aditya Vashista #2](#)

Sort by **Most recent**

All Successful Selected

| Submission and Description | Public Score |
|---|--------------|
| submissionEnsemble.csv 2 hours ago by Aditya Vashista google colab ensemble approach result | 0.83941 |
| submission3.csv 2 hours ago by Aditya Vashista google colab model 3 submission | 0.81182 |
| submission2.csv 2 hours ago by Aditya Vashista google colab model 2 submission | 0.81581 |
| submission1.csv 2 hours ago by Aditya Vashista googlr colab model 1 output | 0.80937 |

googlr colab model 1 output ,

model 5

[submissionEm.csv](#)
 19 hours ago by [Aditya Vashista](#)
 embeded

[submissionM3.csv](#)
 20 hours ago by [Aditya Vashista](#)
 model 2,more lstm layers

[submissionM6.csv](#)
 19 hours ago by [Aditya Vashista](#)
 model 6,final

[submissionM1.csv](#)
 20 hours ago by [Aditya Vashista](#)
 model 1

[submissionM5.csv](#)
 20 hours ago by [Aditya Vashista](#)

Output in final submission CSV file:

| | A | B | C | | A | B | C |
|--------------------|----|--------|---|--------------------|-------|---|---|
| 1 | id | target | | 3238 | 10761 | 1 | |
| 2 | 0 | 1 | | 3239 | 10762 | 1 | |
| 3 | 2 | 1 | | 3240 | 10773 | 1 | |
| 4 | 3 | 1 | | 3241 | 10778 | 1 | |
| 5 | 9 | 1 | | 3242 | 10781 | 1 | |
| 6 | 11 | 1 | | 3243 | 10791 | 0 | |
| 7 | 12 | 0 | | 3244 | 10792 | 0 | |
| 8 | 21 | 0 | | 3245 | 10796 | 0 | |
| 9 | 22 | 0 | | 3246 | 10797 | 0 | |
| 10 | 27 | 0 | | 3247 | 10801 | 0 | |
| 11 | 29 | 0 | | 3248 | 10804 | 0 | |
| 12 | 30 | 0 | | 3249 | 10806 | 0 | |
| 13 | 35 | 0 | | 3250 | 10807 | 0 | |
| 14 | 42 | 0 | | 3251 | 10816 | 0 | |
| 15 | 43 | 0 | | 3252 | 10820 | 0 | |
| 16 | 45 | 0 | | 3253 | 10828 | 0 | |
| 17 | 46 | 1 | | 3254 | 10836 | 1 | |
| 18 | 47 | 0 | | 3255 | 10838 | 1 | |
| 19 | 51 | 0 | | 3256 | 10845 | 1 | |
| 20 | 58 | 0 | | 3257 | 10856 | 1 | |
| 21 | 60 | 0 | | 3258 | 10857 | 1 | |
| 22 | 69 | 0 | | 3259 | 10858 | 1 | |
| 23 | 70 | 1 | | 3260 | 10861 | 1 | |
| 24 | 72 | 0 | | 3261 | 10865 | 1 | |
| 25 | 75 | 1 | | 3262 | 10868 | 1 | |
| 26 | 84 | 0 | | 3263 | 10874 | 1 | |
| 27 | 87 | 0 | | 3264 | 10875 | 0 | |
| 28 | 88 | 1 | | 3265 | | | |
| submissionEnsemble | | | | submissionEnsemble | | | |