Fake News Classifier Using LSTM

Dataset: https://www.kaggle.com/c/fake-news/data#

In [1]:
```
import pandas as pd
```

In [3]:
```
df=pd.read_csv('train/train.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

In [33]:
```
###Drop Nan Values
df=df.dropna()
```

In [34]:
```
## Get the Independent Features

X=df.drop('label',axis=1)
```

In [75]:
```
## Get the Dependent features
y=df['label']
```

In [77]:
```
X.shape
```

Out[77]:
```
(18285, 20)
```

In [76]:
```
y.shape
```

Out[76]:
```
(18285,)
```

In [9]:
```
import tensorflow as tf
```

In [10]:
```
tf.__version__
```

Out[10]:
```
'2.1.0'
```

In [52]:
```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
```

In [13]:
```
### Vocabulary size
voc_size=5000
```

Onehot Representation

In [38]:
```
messages=X.copy()
```

In [103…
```
messages['title'][1]
```

Out[103…
```
'FLYNN: Hillary Clinton, Big Woman on Campus - Breitbart'
```

In [40]:
```
messages.reset_index(inplace=True)
```

In [27]:
```
import nltk
import re
from nltk.corpus import stopwords
```

In [29]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Krish
[nltk_data]     Naik\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[29]:
```
True
```

In [42]:
```
### Dataset Preprocessing
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(messages)):
    print(i)
    review = re.sub('[^a-zA-Z]', ' ', messages['title'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('en
    review = ' '.join(review)
    corpus.append(review)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
 'laid american requir zip lip way grow bolder new york time',
 ...]
```

In [43]:

```python
onehot_repr=[one_hot(words,voc_size)for words in corpus]
onehot_repr
```

Out[43]:

```
[[533, 1014, 4256, 4618, 4250, 2098, 1624, 4170, 1313, 3707],
 [265, 3417, 2702, 4060, 172, 4509, 418],
 [4539, 982, 1183, 1184],
 [3395, 750, 3238, 1835, 1052, 794],
 [1386, 172, 3950, 3405, 1278, 1799, 172, 1200, 1924, 4427],
 [765,
  3359,
  1712,
  3168,
  2929,
  1342,
  794,
  4952,
  3227,
  4510,
  687,
  1605,
  3665,
  4829,
  418],
 [1635, 2322, 3125, 795, 4962, 1322, 646, 449, 3059, 3059, 4160],
 [336, 846, 3903, 907, 269, 3536, 1342, 3046, 3059, 3059, 4160],
 [157, 2391, 4775, 620, 4636, 280, 682, 3131, 1342, 197],
 [3771, 1351, 135, 4351, 4428, 4514, 4825, 1222],
 [4114, 776, 3737, 3293, 1464, 1881, 2681, 4313, 4763, 267, 3852],
 [1835, 249, 4250, 280, 1342, 269],
 [4736, 4622, 1354, 3296, 3260, 3827, 1119, 4060, 2845],
 [1422, 4093, 813, 2113, 1124, 644, 4402, 3059, 3059, 4160],
 [1616, 3179, 699, 3216, 966, 3059, 3059, 4160],
 [4658, 3846, 1787, 4519, 2836, 4144, 4039, 694, 3410, 3995],
 [3482, 4037, 3417],
 [4195, 67, 3891, 3701, 1342, 4452, 1336, 418],
 [1471, 1471, 2702, 1916, 3937, 2511, 3124, 4691, 1799],
 [4278, 2010, 1342, 4486, 4215, 418],
 [3154, 3333, 3612, 3494, 3251, 3124, 3446, 420, 4970, 3059, 3059, 4160],
 [2241, 510, 4541, 4718, 1454, 4328, 4449],
 [3665, 1304, 4997, 875, 3594, 1025, 2219, 1796, 344, 2292, 2085, 418],
 [1319, 3417, 1854, 1417, 4693, 4963, 464, 1011],
 [4164, 433, 2106, 673, 4806, 2492, 1916, 4041, 1807, 3059, 3059, 4160],
 [3417, 2702, 869, 3319, 3059, 3059, 4160],
 [4195, 67, 3891, 4264, 3463, 3271, 673, 418],
 [1481, 3512, 4776, 609, 4954, 3059, 3059, 4160],
 [654, 4672, 3445, 4945, 2613, 3538, 3206, 3087, 604, 3059, 3059, 4160],
 [4599, 985, 1878, 1796, 1768, 1143, 661, 989, 91, 3059, 3059, 4160],
 [269, 1627, 1454, 3550, 3455, 3337, 3059, 3059, 4160],
 [898, 2720, 1707, 3480, 3072],
 [4719, 2224, 3131, 2669],
 [2566, 1422, 1496, 498, 997, 3414, 848, 418],
 [2339, 1528, 2339, 3374, 2673, 1634, 3059, 3059, 4160],
 [1077, 722, 3023, 2914, 3774, 3806, 3326, 418],
 [3449, 1476, 1311, 2629],
 [3712, 3019, 625, 2958, 3955, 1741, 3059, 3059, 4160],
 [3135, 2638, 1170, 3766, 157, 2388, 1021, 4416, 418],
 [1292, 3825, 3298, 1955, 3239, 4459, 2106, 3059, 3059, 4160],
 [4081, 2343, 4495, 1146, 4333, 1175, 1480, 2608, 4829],
 [2939, 4112, 141, 1368, 2951, 3059, 329, 3059, 3059, 4160],
 [4086, 3594, 3618, 2612, 772, 419, 3019, 1699, 4326, 1462, 2147, 418],
 [2569, 2662, 1755],
 [2844, 1420, 3002, 24, 2513, 180, 860, 468, 3955, 418],
 [1342, 2669, 2569, 3519, 3565, 3634],
 [547, 3564, 979, 599, 1897, 3, 2852, 839, 3872, 4386, 4829],
 [2244, 2109, 1304, 3965, 691],
 [3701, 1342, 3125, 2669, 3524, 4135, 1835, 4452, 3014],
```

Embedding Representation

In [81]:
```
sent_length=20
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 4170 1313 3707]
 [   0    0    0 ...  172 4509  418]
 [   0    0    0 ...  982 1183 1184]
 ...
 [   0    0    0 ... 3059 3059 4160]
 [   0    0    0 ... 4076 2723  164]
 [   0    0    0 ... 3937 1837 2236]]
```

In [105…
```
embedded_docs[0]
```

Out[105…
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  533,
       1014, 4256, 4618, 4250, 2098, 1624, 4170, 1313, 3707])
```

In [53]:
```
## Creating model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, 20, 40)            200000    
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               56400     
_________________________________________________________________
dense (Dense)                (None, 1)                 101       
=================================================================
Total params: 256,501
Trainable params: 256,501
Non-trainable params: 0
_________________________________________________________________
None
```

In [83]:
```
len(embedded_docs),y.shape
```

Out[83]:
```
(18285, (18285,))
```

In [ ]:
```

```

In [84]:
```
import numpy as np
X_final=np.array(embedded_docs)
y_final=np.array(y)
```

In [86]:
```
X_final.shape,y_final.shape
```

Out[86]:
```
((18285, 20), (18285,))
```

In [87]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0
```

Model Training

```
In [106…
```

```python
### Finally Training
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=10,batch_size=64
```

```
Train on 12250 samples, validate on 6035 samples
Epoch 1/10
12250/12250 [==============================] - 4s 347us/sample - loss: 0.0041 - acc
uracy: 0.9991 - val_loss: 0.6781 - val_accuracy: 0.9130
Epoch 2/10
12250/12250 [==============================] - 3s 261us/sample - loss: 0.0030 - acc
uracy: 0.9989 - val_loss: 0.5203 - val_accuracy: 0.9102
Epoch 3/10
12250/12250 [==============================] - 4s 293us/sample - loss: 0.0038 - acc
uracy: 0.9990 - val_loss: 0.6349 - val_accuracy: 0.9062
Epoch 4/10
12250/12250 [==============================] - 3s 276us/sample - loss: 0.0037 - acc
uracy: 0.9989 - val_loss: 0.7011 - val_accuracy: 0.9052
Epoch 5/10
12250/12250 [==============================] - 3s 258us/sample - loss: 0.0016 - acc
uracy: 0.9998 - val_loss: 0.7310 - val_accuracy: 0.9089
Epoch 6/10
12250/12250 [==============================] - 4s 307us/sample - loss: 0.0013 - acc
```