from importlib import reload

import sys

from imp import reload

import warnings

warnings.filterwarnings('ignore')

if sys.version[0] == '2':

reload(sys)

sys.setdefaultencoding("utf-8")

In [2]:

import pandas as pd

df1 = pd.read\_csv('../input/word2vec-nlp-tutorial/labeledTrainData.tsv', delimiter="**\t**")

df1 = df1.drop(['id'], axis=1)

df1.head()

Out[2]:

|  | sentiment | review |
| --- | --- | --- |
| 0 | 1 | With all this stuff going down at the moment w... |
| 1 | 1 | \The Classic War of the Worlds\" by Timothy Hi... |
| 2 | 0 | The film starts with a manager (Nicholas Bell)... |
| 3 | 0 | It must be assumed that those who praised this... |
| 4 | 1 | Superbly trashy and wondrously unpretentious 8... |

In [3]:

df2 = pd.read\_csv('../input/imdb-review-dataset/imdb\_master.csv',encoding="latin-1")

df2.head()

Out[3]:

|  | Unnamed: 0 | type | review | label | file |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | test | Once again Mr. Costner has dragged out a movie... | neg | 0\_2.txt |
| 1 | 1 | test | This is an example of why the majority of acti... | neg | 10000\_4.txt |
| 2 | 2 | test | First of all I hate those moronic rappers, who... | neg | 10001\_1.txt |
| 3 | 3 | test | Not even the Beatles could write songs everyon... | neg | 10002\_3.txt |
| 4 | 4 | test | Brass pictures (movies is not a fitting word f... | neg | 10003\_3.txt |

In [4]:

df2 = df2.drop(['Unnamed: 0','type','file'],axis=1)

df2.columns = ["review","sentiment"]

df2.head()

Out[4]:

|  | review | sentiment |
| --- | --- | --- |
| 0 | Once again Mr. Costner has dragged out a movie... | neg |
| 1 | This is an example of why the majority of acti... | neg |
| 2 | First of all I hate those moronic rappers, who... | neg |
| 3 | Not even the Beatles could write songs everyon... | neg |
| 4 | Brass pictures (movies is not a fitting word f... | neg |

In [5]:

df2 = df2[df2.sentiment != 'unsup']

df2['sentiment'] = df2['sentiment'].map({'pos': 1, 'neg': 0})

df2.head()

Out[5]:

|  | review | sentiment |
| --- | --- | --- |
| 0 | Once again Mr. Costner has dragged out a movie... | 0 |
| 1 | This is an example of why the majority of acti... | 0 |
| 2 | First of all I hate those moronic rappers, who... | 0 |
| 3 | Not even the Beatles could write songs everyon... | 0 |
| 4 | Brass pictures (movies is not a fitting word f... | 0 |

In [6]:

df = pd.concat([df1, df2]).reset\_index(drop=True)

df.head()

Out[6]:

|  | review | sentiment |
| --- | --- | --- |
| 0 | With all this stuff going down at the moment w... | 1 |
| 1 | \The Classic War of the Worlds\" by Timothy Hi... | 1 |
| 2 | The film starts with a manager (Nicholas Bell)... | 0 |
| 3 | It must be assumed that those who praised this... | 0 |
| 4 | Superbly trashy and wondrously unpretentious 8... | 1 |

In [7]:

import re

from nltk.stem import WordNetLemmatizer

from nltk.corpus import stopwords

stop\_words = set(stopwords.words("english"))

lemmatizer = WordNetLemmatizer()

def clean\_text(text):

text = re.sub(r'[^\w\s]','',text, re.UNICODE)

text = text.lower()

text = [lemmatizer.lemmatize(token) for token **in** text.split(" ")]

text = [lemmatizer.lemmatize(token, "v") for token **in** text]

text = [word for word **in** text if **not** word **in** stop\_words]

text = " ".join(text)

return text

df['Processed\_Reviews'] = df.review.apply(lambda x: clean\_text(x))

In [8]:

df.head()

Out[8]:

|  | review | sentiment | Processed\_Reviews |
| --- | --- | --- | --- |
| 0 | With all this stuff going down at the moment w... | 1 | stuff go moment mj ive start listen music watc... |
| 1 | \The Classic War of the Worlds\" by Timothy Hi... | 1 | classic war world timothy hines entertain film... |
| 2 | The film starts with a manager (Nicholas Bell)... | 0 | film start manager nicholas bell give welcome ... |
| 3 | It must be assumed that those who praised this... | 0 | must assume praise film greatest film opera ev... |
| 4 | Superbly trashy and wondrously unpretentious 8... | 1 | superbly trashy wondrously unpretentious 80 ex... |

In [9]:

df.Processed\_Reviews.apply(lambda x: len(x.split(" "))).mean()

Out[9]:

128.51009333333334

In [10]:

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.layers import Dense , Input , LSTM , Embedding, Dropout , Activation, GRU, Flatten

from keras.layers import Bidirectional, GlobalMaxPool1D

from keras.models import Model, Sequential

from keras.layers import Convolution1D

from keras import initializers, regularizers, constraints, optimizers, layers

max\_features = 6000

tokenizer = Tokenizer(num\_words=max\_features)

tokenizer.fit\_on\_texts(df['Processed\_Reviews'])

list\_tokenized\_train = tokenizer.texts\_to\_sequences(df['Processed\_Reviews'])

maxlen = 130

X\_t = pad\_sequences(list\_tokenized\_train, maxlen=maxlen)

y = df['sentiment']

embed\_size = 128

model = Sequential()

model.add(Embedding(max\_features, embed\_size))

model.add(Bidirectional(LSTM(32, return\_sequences = True)))

model.add(GlobalMaxPool1D())

model.add(Dense(20, activation="relu"))

model.add(Dropout(0.05))

model.add(Dense(1, activation="sigmoid"))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

batch\_size = 100

epochs = 3

model.fit(X\_t,y, batch\_size=batch\_size, epochs=epochs, validation\_split=0.2)

Using TensorFlow backend.

Train on 60000 samples, validate on 15000 samples

Epoch 1/3

60000/60000 [==============================] - 326s 5ms/step - loss: 0.3485 - acc: 0.8409 - val\_loss: 0.2761 - val\_acc: 0.8940

Epoch 2/3

60000/60000 [==============================] - 361s 6ms/step - loss: 0.2172 - acc: 0.9156 - val\_loss: 0.1667 - val\_acc: 0.9425

Epoch 3/3

60000/60000 [==============================] - 285s 5ms/step - loss: 0.1691 - acc: 0.9373 - val\_loss: 0.1547 - val\_acc: 0.9489

Out[10]:

<keras.callbacks.History at 0x7f78cff4df60>

In [11]:

df\_test=pd.read\_csv("../input/word2vec-nlp-tutorial/testData.tsv",header=0, delimiter="**\t**", quoting=3)

df\_test.head()

df\_test["review"]=df\_test.review.apply(lambda x: clean\_text(x))

df\_test["sentiment"] = df\_test["id"].map(lambda x: 1 if int(x.strip('"').split("\_")[1]) >= 5 else 0)

y\_test = df\_test["sentiment"]

list\_sentences\_test = df\_test["review"]

list\_tokenized\_test = tokenizer.texts\_to\_sequences(list\_sentences\_test)

X\_te = pad\_sequences(list\_tokenized\_test, maxlen=maxlen)

prediction = model.predict(X\_te)

y\_pred = (prediction > 0.5)

from sklearn.metrics import f1\_score, confusion\_matrix

print('F1-score: **{0}**'.format(f1\_score(y\_pred, y\_test)))

print('Confusion matrix:')

confusion\_matrix(y\_pred, y\_test)

F1-score: 0.9583734359961502

Confusion matrix:

Out[11]:

array([[12013, 551],

[ 487, 11949]])