Untitled2

November 13, 2019

[438]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪the files in the input directory

import os
cwd = os.getcwd()
```

[ ]:
```

```

[ ]:
```

```

[439]:
```
data= pd.read_csv("/Users/michelamaineri/Downloads/training.10000.csv",␣
↪encoding = "latin-1")
header= ['target','id','date','flag','user','text']
data.set_axis(header,axis=1,inplace=True)
data_ready=data.drop(['id','date','flag','user'],axis=1)
data.head()
data_ready.head()

#/Users/michelamaineri/Downloads/training.600.csv
```

[439]:
```
   target                                               text
0       0  is upset that he can't update his Facebook by …
1       0  @Kenichan I dived many times for the ball. Man…
2       0    my whole body feels itchy and like its on fire
3       0  @nationwideclass no, it's not behaving at all…
4       0                      @Kwesidei not the whole crew
```

[ ]:
```

```

[440]:
```
#data= pd.read_csv("/Users/michelamaineri/Downloads/training.1600000.processed.
↪noemoticon.csv",encoding='latin-1')
#header= ['target','id','date','flag','user','text']
#data.set_axis(header,axis=1,inplace=True)
```

```
#data_ready=data.drop(['id','date','flag','user'],axis=1)
#data.head()
#data_ready.head()
```

[441]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (3.4.5)
Requirement already satisfied: six in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from nltk)
(1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

[442]:
```
#import os
#os.environ["KMP_DUPLICATE_LIB_OK"]="TRUE"
```

[443]:
```
import string
```

[444]:
```
string.punctuation
```

[444]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

[445]:
```
from nltk.tokenize import TweetTokenizer
```

[446]:
```
from nltk.corpus import stopwords
```

[447]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (2.0.0)
Requirement already satisfied: wrapt>=1.11.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.11.2)
Requirement already satisfied: termcolor>=1.1.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.1.0)
Requirement already satisfied: astor>=0.6.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (0.8.0)
Requirement already satisfied: wheel>=0.26 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (0.33.6)
Requirement already satisfied: tensorflow-estimator<2.1.0,>=2.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (2.0.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (3.1.0)
Requirement already satisfied: six>=1.10.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.12.0)
Requirement already satisfied: absl-py>=0.7.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (0.8.1)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.1.0)
Requirement already satisfied: grpcio>=1.8.6 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.24.3)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.17.2)
Requirement already satisfied: keras-applications>=1.0.8 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (1.0.8)
Requirement already satisfied: tensorboard<2.1.0,>=2.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (2.0.1)
Requirement already satisfied: google-pasta>=0.1.6 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (0.1.7)
Requirement already satisfied: gast==0.2.2 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (0.2.2)
Requirement already satisfied: protobuf>=3.6.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorflow) (3.10.0)
Requirement already satisfied: h5py in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras-
applications>=1.0.8->tensorflow) (2.9.0)
Requirement already satisfied: werkzeug>=0.11.15 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow) (0.16.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow) (1.6.3)
Requirement already satisfied: markdown>=2.6.8 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow) (3.1.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow) (0.4.1)
Requirement already satisfied: setuptools>=41.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
tensorboard<2.1.0,>=2.0.0->tensorflow) (41.4.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from google-
auth<2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow) (0.2.7)
Requirement already satisfied: rsa>=3.1.4 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from google-
auth<2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow) (4.0)
Requirement already satisfied: cachetools>=2.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from google-
auth<2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow) (3.1.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (1.2.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
pyasn1-modules>=0.2.1->google-
auth<2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow) (0.4.7)
Requirement already satisfied: requests>=2.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from requests-
oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (2.22.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from requests-
oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (3.1.0)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
requests>=2.0.0->requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (2019.9.11)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
requests>=2.0.0->requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
requests>=2.0.0->requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (2.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from
requests>=2.0.0->requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow) (1.24.2)
Note: you may need to restart the kernel to use updated packages.
```

[448]:
```
pip install keras
```

```
Requirement already satisfied: keras in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (2.3.1)
Requirement already satisfied: pyyaml in
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(5.1.2)
Requirement already satisfied: numpy>=1.9.1 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(1.17.2)
Requirement already satisfied: keras-applications>=1.0.6 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(1.0.8)
Requirement already satisfied: h5py in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(2.9.0)
Requirement already satisfied: six>=1.9.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(1.12.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(1.1.0)
Requirement already satisfied: scipy>=0.14 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from keras)
(1.3.1)
Note: you may need to restart the kernel to use updated packages.
```

[449]:
```python
from keras.preprocessing.sequence import pad_sequences
```

[450]:
```python
from keras.preprocessing import sequence
```

[451]:
```python
from keras.preprocessing.text import Tokenizer
```

[452]:
```python
from keras.models import Sequential
```

[453]:
```python
from keras.layers import Dense, LSTM, SpatialDropout1D, Embedding
```

[454]:
```python
from keras.optimizers import Adam
```

[455]:
```python
from keras.utils import to_categorical
```

[456]:
```python
from nltk.stem import PorterStemmer
```

[457]:
```python
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

[458]:
```python
import string
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from keras.layers import Dense, LSTM, SpatialDropout1D, Embedding
from keras.optimizers import Adam
from keras.utils import to_categorical
from nltk.stem import PorterStemmer
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

[459]:
```
punct = list(string.punctuation)
import nltk
nltk.download('stopwords')
stopword_list = stopwords.words('english') + punct + ['rt','via', '...']
stemmer= PorterStemmer()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/michelamaineri/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[460]:
```
# define a function for data cleaning / preprocessing
def sentense_to_words(raw_review):
    text = raw_review.lower()
    tokens = TweetTokenizer().tokenize(text=text)
    clean_tokens= [stemmer.stem(tok) for tok in tokens if tok not in␣
 ↪stopword_list and not tok.isdigit() and not tok.startswith('@')and not tok.
 ↪startswith('#')and not tok.startswith('http')]
    return( " ".join(clean_tokens))
```

[466]:
```
# test the function for one tweet
tweet=sentense_to_words( data_ready['text'][9998])
print(data_ready['text'][9998], len(tweet) )
print(tweet)
```

```
happy #charitytuesday @theNSPCC @SparksCharity @SpeakingUpH4H  5
happi
```

[467]:
```
corpus=[]
sent_len_list=[]
for i in range(0,len(data_ready)):
    corp= sentense_to_words(data_ready['text'][i])
    sent_len_list.append(len(corp))
    corpus.append(corp)
```

[468]:
```
#max_len=5
max_features=2000
```

[469]:
```
# creating vectorized corpus and padding
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(corpus)
X = tokenizer.texts_to_sequences(corpus)
```

```
X = pad_sequences(X, maxlen=max_len)
```

[ ]:
```

```

[470]:
```
print(data_ready['target'].values)
```

```
[0 0 0 … 4 4 4]
```

[ ]:
```

```

[471]:
```
# relabel the sentiments 4 as 1
label= data_ready['target'].values
new_label=list(map(lambda x:x if x!= 4 else 1,label))
Y=to_categorical(new_label)
```

[472]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (0.0)
Requirement already satisfied: scikit-learn in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from sklearn)
(0.21.3)
Requirement already satisfied: joblib>=0.11 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from scikit-
learn->sklearn) (0.13.2)
Requirement already satisfied: scipy>=0.17.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from scikit-
learn->sklearn) (1.3.1)
Requirement already satisfied: numpy>=1.11.0 in
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-packages (from scikit-
learn->sklearn) (1.17.2)
Note: you may need to restart the kernel to use updated packages.
```

[473]:
```
print(X_train)
```

```
[[   0    0    6   14  747]
 [   0    0   42   11  204]
 [   0    0    0    6   14]
 …
 [ 170  379  176  256 1186]
 [  38   13   69  275  142]
 [ 423 1015   83  282   84]]
```

[474]:
```
display(sum(Y)/len(Y))
```

```
array([0.49995, 0.50005], dtype=float32)
```

```
[475]: print(Y)
```

```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 …
 [0. 1.]
 [0. 1.]
 [0. 1.]]
```

```
[510]: # train test split
from keras.regularizers import l2
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.33,␣
 ↪random_state=42)
#print(X_train, X_test, y_train, y_test)
#print('shape of training set: {}' .format(X_train.shape))
#print('shape of valid set: {}' .format(X_train.shape))
#print('shape of test set: {}' .format(X_test.shape))
#model.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
 ↪dropout=0.2))

classifier = Sequential()
classifier.add(Embedding(max_features,128,input_length = X.shape[1], mask_zero␣
 ↪=True)
#classifier.add(keras.layers.SpatialDropout1D(0.4))
classifier.add(LSTM(196,dropout=0.2,recurrent_dropout=0.
 ↪2,return_sequences=False))
classifier.add(Dense(2,activation='sigmoid'))
classifier.compile(loss = 'binary_crossentropy', optimizer='adam',metrics =␣
 ↪['accuracy'])
classifier.summary()
callback = [EarlyStopping(monitor='val_loss',␣
 ↪patience=5),ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',␣
 ↪save_best_only=True)]
history = classifier.fit(X_train, y_train,batch_size=32, callbacks= callback,␣
 ↪epochs=7,validation_data=(X_test, y_test))
```

```
          File "<ipython-input-510-a3cd431e7fd4>", line 14
        classifier.add(LSTM(196,dropout=0.2,recurrent_dropout=0.
↪2,return_sequences=False))
                  ^
    SyntaxError: invalid syntax
```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

[482]:

```
pd.DataFrame(history.history)
```

[482]:

```
   val_loss  val_accuracy      loss  accuracy
0  0.589871      0.700167  0.635795  0.647807
1  0.609916      0.701667  0.464295  0.801900
2  0.663314      0.690333  0.355935  0.860194
```

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

[ ]:

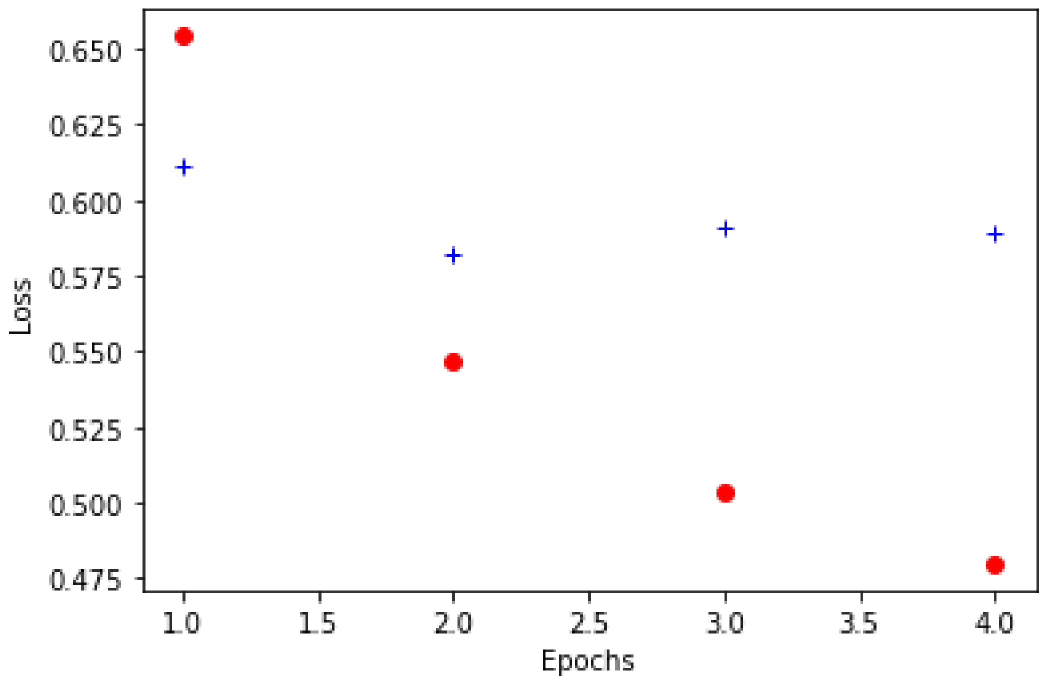
```
#history = model.fit(X, Y, epochs=num_steps,␣
 ↪batch_size=batch_size,validation_data=(X_test,Y_test))#validation_split=0.05
```

[398]:

```
from matplotlib import pyplot as plt
history_dict=history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'ro')
plt.plot(epochs, val_loss_values, 'b+')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

[398]:

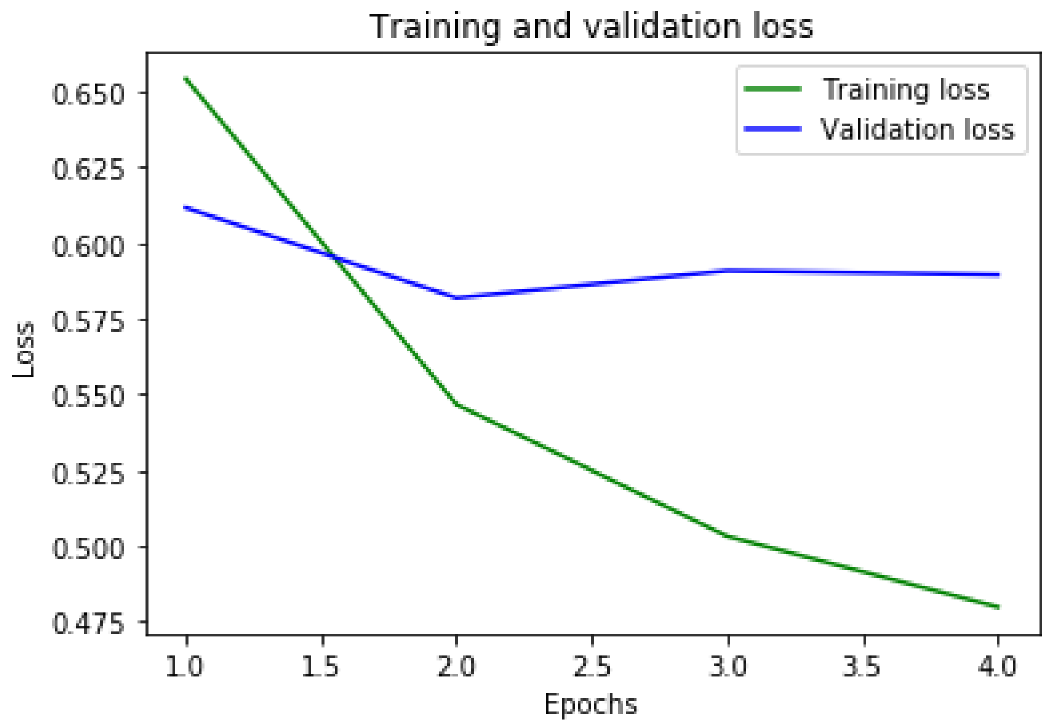
```
Text(0, 0.5, 'Loss')
```



```
[ ]:
```

```
[ ]:
```

```
[399]: from matplotlib import pyplot as plt
history_dict=history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'ro')
plt.plot(epochs, val_loss_values, 'b+')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.clf()
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'g', label='Training loss')
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()
```

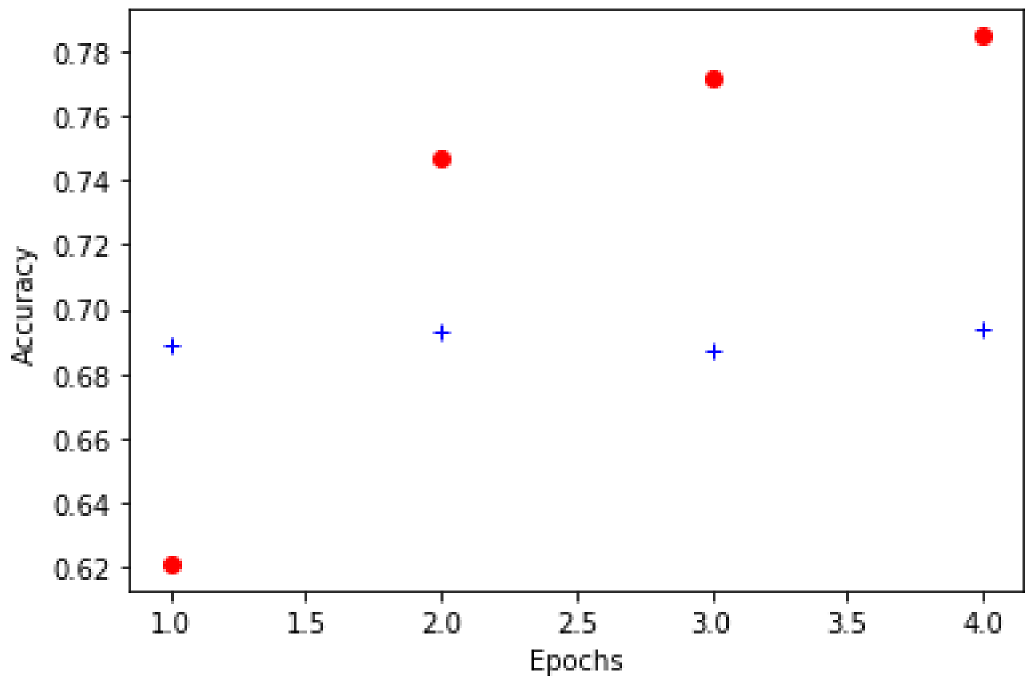


```
[ ]:
```

```
[ ]:
```

```
[400]: loss_values = history_dict['accuracy']
       val_loss_values = history_dict['val_accuracy']
       epochs = range(1, len(loss_values) + 1)
       plt.plot(epochs, loss_values, 'ro')
       plt.plot(epochs, val_loss_values, 'b+')
       plt.xlabel('Epochs')
       plt.ylabel('Accuracy')
```

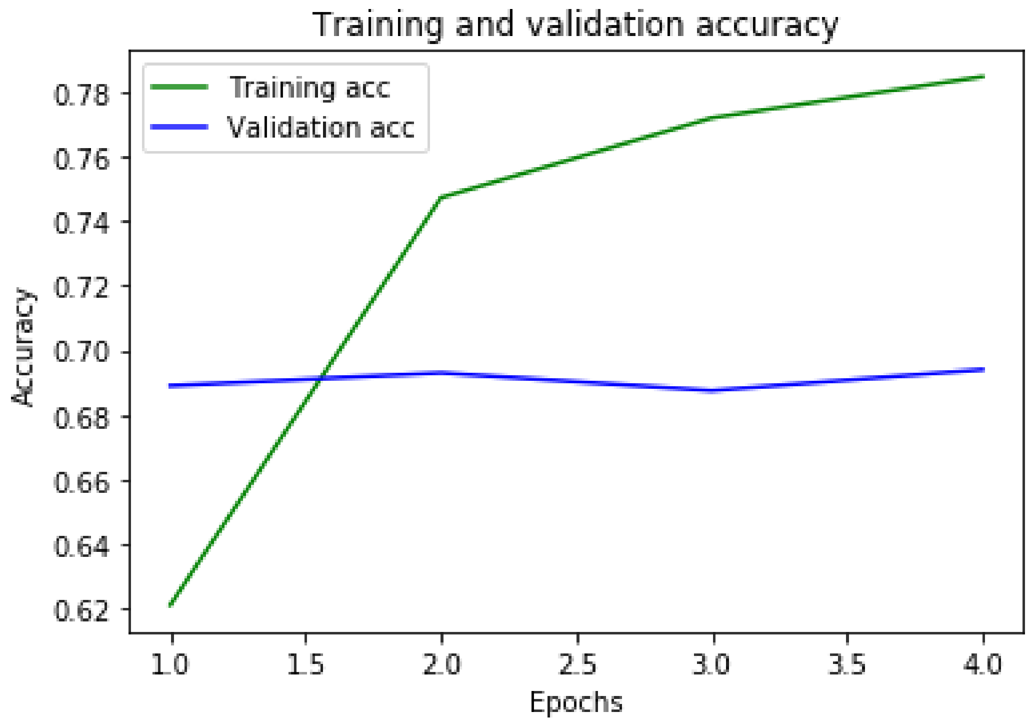
```
[400]: Text(0, 0.5, 'Accuracy')
```



```
[ ]:
```

```
[ ]:
```

```
[401]: plt.clf()
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
plt.plot(epochs, acc, 'g', 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()
plt.show()
```



```
[ ]:
```

```
[ ]:
```

```
[381]: #scores = model.evaluate(X_test, y_test)
       #print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
[ ]:
```

```
[531]: import re
       import tweepy
       from tweepy import OAuthHandler
       from textblob import TextBlob

       class TwitterClient(object):
           '''
           Generic Twitter Class for sentiment analysis.
           '''
           def __init__(self):
               '''
               Class constructor or initialization method.
               '''
               # keys and tokens from the Twitter Dev Console
```

```
        consumer_key = 'l3A0i3tJA0LicAS93HjQlejWV'
        consumer_secret = 'GNzqcwyQgKaes3AqBJDEPsq16af5nqgaMnBeVlmRrfSYcX4XQt'
        access_token = '1189207516791615488-D8jjGLV7LUzKa07MpLNvPO9YMir1Nh'
        access_token_secret = 'Jcvp02RFzzTPGQ8jrHDUfGLHzHfjv96vNJxg7ZEbwyeXm'

        # attempt authentication
        try:
            # create OAuthHandler object
            self.auth = OAuthHandler(consumer_key, consumer_secret)
            # set access token and secret
            self.auth.set_access_token(access_token, access_token_secret)
            # create tweepy API object to fetch tweets
            self.api = tweepy.API(self.auth)
        except:
            print("Error: Authentication Failed")

    def clean_tweet(self, tweet):
        '''
        Utility function to clean tweet text by removing links, special␣
 ↪characters
        using simple regex statements.
        '''
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/
 ↪\S+)", " ", tweet).split())

    def get_tweet_sentiment(self, tweet):
        '''
        Utility function to classify sentiment of passed tweet
        using textblob's sentiment method
        '''
        # create TextBlob object of passed tweet text
        analysis = TextBlob(self.clean_tweet(tweet)) #copia fai variabile e fai␣
 ↪sentiment
        # set sentiment
        if analysis.sentiment.polarity >= 0:
            return 'positive'
        #elif analysis.sentiment.polarity == 0:
            #return 'neutral'
        else:
            return 'negative'

    def get_tweets(self, query, count = 10):
        '''
        Main function to fetch tweets and parse them.
        '''
        # empty list to store parsed tweets
        tweets = []
        retTextList = []
        retSentList = []
        try:
            # call twitter api to fetch tweets
            fetched_tweets = self.api.search(q = query, count = count)

            # parsing tweets one by one
            retTextList = []
            retSentList = []
            for tweet in fetched_tweets:
                # empty dictionary to store required params of a tweet
                parsed_tweet = {}

                # saving text of tweet
                parsed_tweet['text'] = tweet.text
                retTextList.append(tweet.text)
                # saving sentiment of tweet
                parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)
                retSentList.append(self.get_tweet_sentiment(tweet.text))
                # appending parsed tweet to tweets list
                if tweet.retweet_count > 0:
                    # if tweet has retweets, ensure that it is appended only␣
 ↪once

                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)
                else:
                    tweets.append(parsed_tweet)

            # return parsed tweets
            return tweets, retTextList, retSentList

        except tweepy.TweepError as e:
            # print error (if any)
            print("Error : " + str(e))

def main():
    # creating object of TwitterClient Class
    api = TwitterClient()
    # calling function to get tweets
    tweets, retTextList, retSentList = api.get_tweets(query = 'Hong Kong',␣
 ↪count = 200)

    # picking positive tweets from tweets
    ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']
    # percentage of positive tweets
    print("Positive tweets percentage: {} %".format(100*len(ptweets)/
↪len(tweets)))
    # picking negative tweets from tweets
    ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']
    # percentage of negative tweets
    print("Negative tweets percentage: {} %".format(100*len(ntweets)/
↪len(tweets)))
    # percentage of neutral tweets
   # print("Neutral tweets percentage: {} % \ ".format(str(100*len(tweets -␣
↪ntweets - ptweets)/len(tweets))))
    #print(100*len(tweets - ntweets - ptweets)/len(tweets))
    # printing first 5 positive tweets
    print("\n\nPositive tweets:")
    for tweet in ptweets[:10]:
        print(tweet['text'])

    # printing first 5 negative tweets
    print("\n\nNegative tweets:")
    for tweet in ntweets[:10]:
        print(tweet['text'])
    return retTextList, retSentList
```

[ ]:

```
```

[532]:

```
#if __name__ == "__main__":
    # calling main function
retTextList, retSentList = main()
import pandas as pd
df = pd.DataFrame({'text':[retTextList],'target':[retSentList]})
print(df)
```

```
Positive tweets percentage: 85.71428571428571 %
Negative tweets percentage: 14.285714285714286 %


Positive tweets:
RT @LeaderHoyer: I am deeply concerned by the aggressive actions taken by the
Hong Kong Police Force today during a standoff between protes…
RT @smithmarion: We had a great meeting with @POTUS at the @WhiteHouse &amp; he
truly cares about the victims of communism. He has previously c…
RT @SenMarkey: Congress must pass the #HongKong Human Rights &amp; Democracy Act
to support these brave protesters &amp; send a message to Beijing…
RT @benedictrogers: I call on the world to act to save Hong Kong, to put
pressure on Carrie Lam to pull back from the brink, demand an end…
This is the beginning of the end for Hong Kong as a commercial and financial
hub, Shenzhen will be the next port of… https://t.co/kvLLjpT233
RT @niccijsmith: Cambridge University under pressure to revoke #HongKong chief
```

```
Carrie Lam’s honorary fellowship' | via @telegraph https://t…
RT @SenSchumer: We need to act on the tragedy unfolding in Hong Kong. Why is
President Trump giving the Chinese Communist Party a free hand…
Countries with the most residents with a net worth of $30 million or more in the
world, 2018.

 US: 81,340
 China… https://t.co/vegFonK2Tj
RT @FinancialTimes: Jamil Anderlini: If societal breakdown can happen in Hong
Kong, it can happen anywhere. And it will take decades to bui…
RT @SolomonYue: DC, "Liberate Hong Kong, the revolution of our times" flag is
still there at early dawn for the world to see! God bless ALL…



Negative tweets:
RT @HKNordicHearts: SOS! HKPF is massacring in the Chinese University of Hong
Kong! They are killing students! Students are trapped with no…
@FinancialTimes #HongKongPolice shoot at least 2356 tear gas to the Chinese
University of Hong Kong. What a excessi… https://t.co/oPDfkOksBj
20191113
#HKPoliceState murder #HKCitizens
Dangerous hong kong, pls don't come here.
#HKHumanRightsandDemocracyAct… https://t.co/WPYT1BW111
RT @onlyyoontv: "The world needs to see that the United States will stand up and
say this is wrong, we stand with the people of #HongKong,"…
RT @Moira_Ooops: A civilian's wrist was broken by riot police in Central, Hong
Kong https://t.co/SvtOpIvg5T
RT @hoccgoomusic: Hong Kong media today.

Only @appledaily_hk and @EpochTimes hv news about yesterday protests, all other
six with the same…
RT @nomad99hk: A scene from Holy Cross Church in Hong Kong:

1. Riot Police rush in church to arrest protesters, while they need to have a…
RT @SenatorMenendez: It's been half a year since #HongKong citizens took to the
streets in the millions to protest the erosion of democracy…
RT @HKMarkSimon: Joe Biden is awful on Hong Kong &amp; China. He’d bring in
Martin Lee &amp; Anson Chan, give them a picture, and then we’d be shaf…
                                               text  \
0  [RT @LeaderHoyer: I am deeply concerned by the…

                                             target
0  [positive, positive, positive, positive, posit…
```

```
[ ]:
```

```
[551]: display(df.set_index('text').iloc[:,0])
       print(type(df.set_index('text').iloc[:,0]))
       def sentense_to_words(raw_review):
           text = raw_review.lower()
           tokens = TweetTokenizer().tokenize(text=text)
           clean_tokens= [stemmer.stem(tok) for tok in tokens if tok not in␣
        ↪stopword_list and not tok.isdigit() and not tok.startswith('@')and not tok.
        ↪startswith('#')and not tok.startswith('http')]
           return( " ".join(clean_tokens))
       maxlen=5
       tokens = pad_sequences(tokens, maxlen=maxlen)
       sentiment = model.predict(np.array(tokens), batch_size=32, verbose = 2)[0][0]
       print()
       print('Sentiment =', sentiment)
       if (round(sentiment) == 0):
           print('Negative')
       else:
           print('Positive')
```

```
text
[RT @LeaderHoyer: I am deeply concerned by the aggressive actions taken by the Hong Kong Polic
Name: target, dtype: object


<class 'pandas.core.series.Series'>

Sentiment = 0.49860168
Negative
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[384]: print(type(df.set_index('text').iloc[:,0]))
```

```
<class 'pandas.core.series.Series'>
```

```
[513]: from keras.regularizers import l2
       from keras.optimizers import Adam
       embed_dim = 128
       lstm_out = 150
       adam =Adam(lr =1e-6)
       model1 = Sequential()
       model1.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
        ↪mask_zero=True))
```

```
#model1.add(SpatialDropout1D(0.4))
model1.add(LSTM(lstm_out,kernel_regularizer=l2(0.001), dropout_U=0.2,␣
 ↪dropout_W=0.2))

model1.add(Dense(2,activation='sigmoid'))
model1.compile(loss = 'binary_crossentropy', optimizer=adam,metrics =␣
 ↪['accuracy'])
print(model1.summary())

y = pd.get_dummies(data['target']).values
print(y)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.1,␣
 ↪random_state = 42)
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
history1 = model1.fit(X_train,␣
 ↪y_train,batch_size=32,epochs=5,validation_data=(X_test, y_test))
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:9: UserWarning: Update your `LSTM` call to the
Keras 2 API: `LSTM(150, kernel_regularizer=<keras.reg…, dropout=0.2,
recurrent_dropout=0.2)`
  if __name__ == '__main__':
```

```
Model: "sequential_113"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_90 (Embedding)     (None, 5, 128)            256000
_________________________________________________________________
lstm_102 (LSTM)              (None, 150)               167400
_________________________________________________________________
dense_121 (Dense)            (None, 2)                 302
=================================================================
Total params: 423,702
Trainable params: 423,702
Non-trainable params: 0
_________________________________________________________________
None
[[1 0]
 [1 0]
 [1 0]
 …
 [0 1]
 [0 1]
 [0 1]]
(8999, 5) (8999, 2)
(1000, 5) (1000, 2)
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

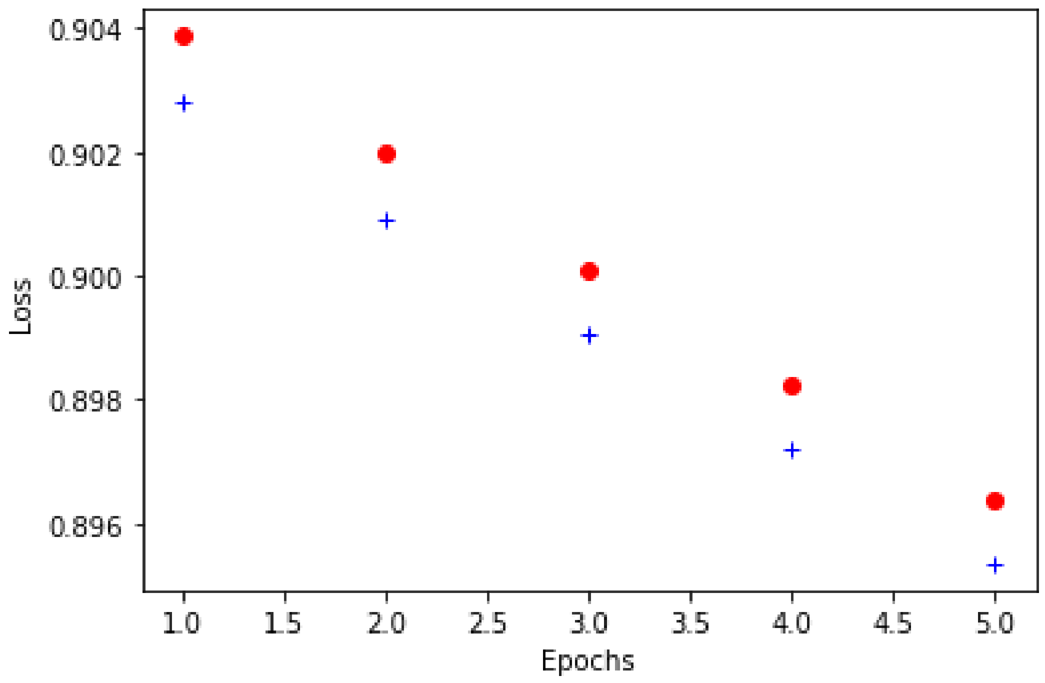
```
Train on 8999 samples, validate on 1000 samples
Epoch 1/5
8999/8999 [==============================] - 97s 11ms/step - loss: 0.9039 -
accuracy: 0.4921 - val_loss: 0.9028 - val_accuracy: 0.4995
Epoch 2/5
8999/8999 [==============================] - 37s 4ms/step - loss: 0.9020 -
accuracy: 0.4955 - val_loss: 0.9009 - val_accuracy: 0.5045
Epoch 3/5
8999/8999 [==============================] - 34s 4ms/step - loss: 0.9001 -
accuracy: 0.4952 - val_loss: 0.8990 - val_accuracy: 0.5085
Epoch 4/5
8999/8999 [==============================] - 34s 4ms/step - loss: 0.8982 -
accuracy: 0.4965 - val_loss: 0.8972 - val_accuracy: 0.5110
Epoch 5/5
8999/8999 [==============================] - 34s 4ms/step - loss: 0.8964 -
accuracy: 0.5023 - val_loss: 0.8953 - val_accuracy: 0.5125
```

[514]:
```
pd.DataFrame(history1.history)
```

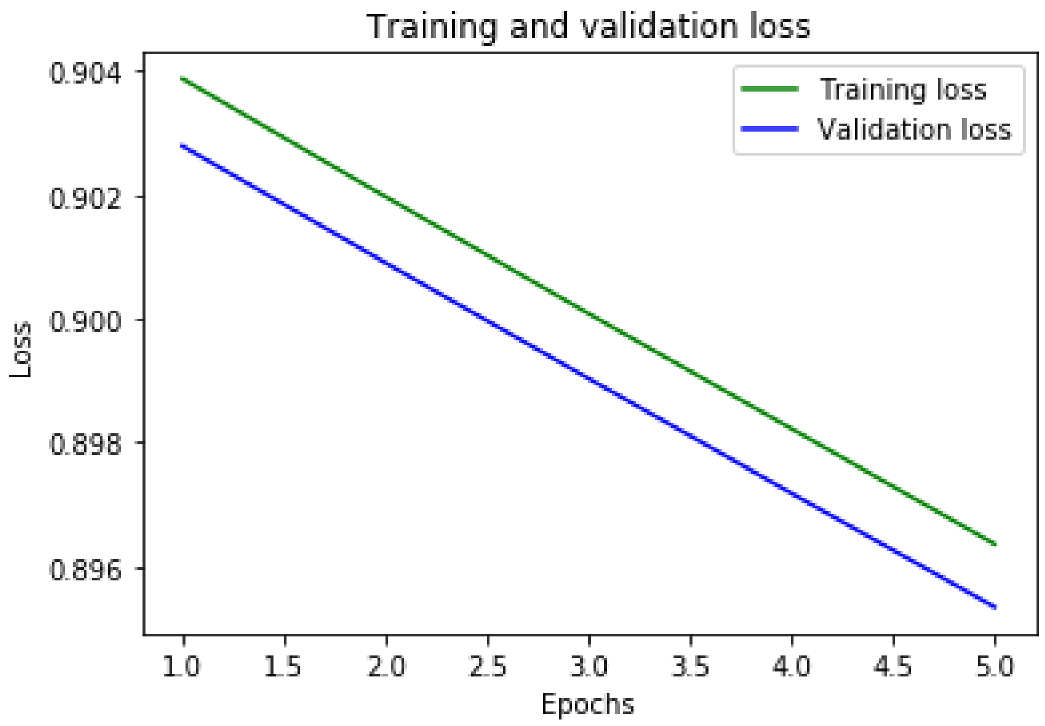
[514]:
```
   val_loss  val_accuracy      loss  accuracy
0  0.902788        0.4995  0.903874  0.492055
1  0.900905        0.5045  0.901970  0.495499
2  0.899036        0.5085  0.900086  0.495222
3  0.897184        0.5110  0.898224  0.496500
4  0.895348        0.5125  0.896365  0.502334
```

[516]:
```
from matplotlib import pyplot as plt
history_dict=history1.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'ro')
plt.plot(epochs, val_loss_values, 'b+')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

[516]:
```
Text(0, 0.5, 'Loss')
```

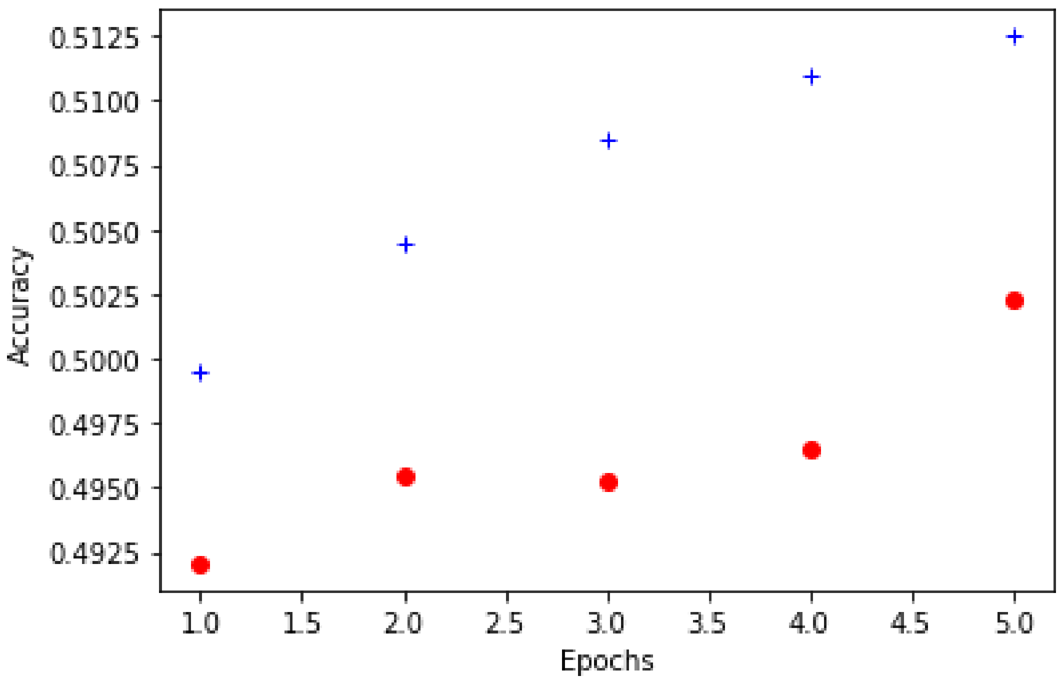


[517]:
```
plt.clf()
loss = history1.history['loss']
val_loss = history1.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'g', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

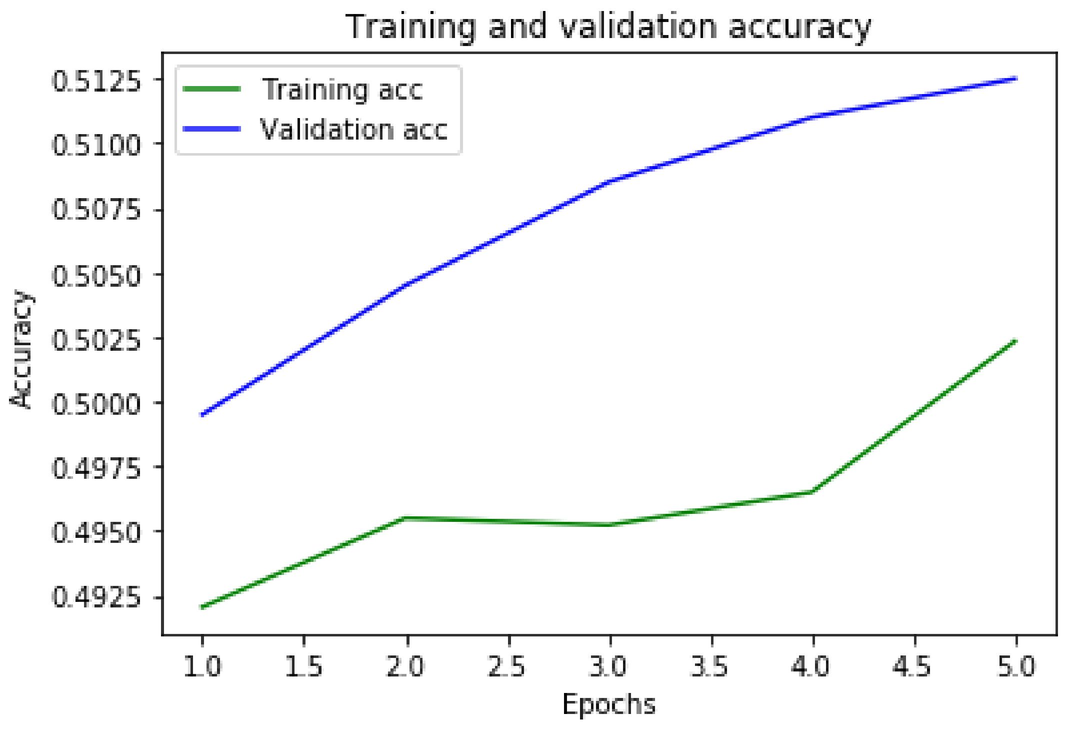


```
[518]: loss_values = history_dict['accuracy']
       val_loss_values = history_dict['val_accuracy']
       epochs = range(1, len(loss_values) + 1)
       plt.plot(epochs, loss_values, 'ro')
       plt.plot(epochs, val_loss_values, 'b+')
       plt.xlabel('Epochs')
       plt.ylabel('Accuracy')
```

```
[518]: Text(0, 0.5, 'Accuracy')
```



[520]:
```
plt.clf()
acc = history1.history['accuracy']
val_acc = history1.history['val_accuracy']
plt.plot(epochs, acc, 'g', 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()
plt.show()
```



[526]:
```
from keras.regularizers import l2
#from keras.optimizers import Adam
embed_dim = 128
lstm_out = 150
adam =Adam(lr =1e-4)
model2 = Sequential()
model2.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
 ↪mask_zero=True))
#model1.add(SpatialDropout1D(0.4))
model2.add(LSTM(lstm_out,kernel_regularizer=l2(0.001), dropout_U=0.2,␣
 ↪dropout_W=0.2))

model2.add(Dense(2,activation='sigmoid'))
model2.compile(loss = 'binary_crossentropy', optimizer=adam,metrics =␣
 ↪['accuracy'])
print(model2.summary())

y = pd.get_dummies(data['target']).values
print(y)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.1,␣
 ↪random_state = 42)
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
#callback = [EarlyStopping(monitor='val_loss',␣
↪patience=5),ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',␣
↪save_best_only=True)]
history2 = model2.fit(X_train, y_train,batch_size=32,␣
↪epochs=27,validation_data=(X_test, y_test))
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:9: UserWarning: Update your `LSTM` call to the
Keras 2 API: `LSTM(150, kernel_regularizer=<keras.reg…, dropout=0.2,
recurrent_dropout=0.2)`
  if __name__ == '__main__':
```

```
Model: "sequential_116"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_93 (Embedding)     (None, 5, 128)            256000
_________________________________________________________________
lstm_105 (LSTM)              (None, 150)               167400
_________________________________________________________________
dense_124 (Dense)            (None, 2)                 302
=================================================================
Total params: 423,702
Trainable params: 423,702
Non-trainable params: 0
_________________________________________________________________
None
[[1 0]
 [1 0]
 [1 0]
 …
 [0 1]
 [0 1]
 [0 1]]
(8999, 5) (8999, 2)
(1000, 5) (1000, 2)
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

```
Train on 8999 samples, validate on 1000 samples
Epoch 1/27
8999/8999 [==============================] - 56s 6ms/step - loss: 0.8302 -
accuracy: 0.5805 - val_loss: 0.7712 - val_accuracy: 0.6040
Epoch 2/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.6988 -
```

```
accuracy: 0.6791 - val_loss: 0.6539 - val_accuracy: 0.6595
Epoch 3/27
8999/8999 [==============================] - 32s 4ms/step - loss: 0.6016 -
accuracy: 0.7327 - val_loss: 0.6226 - val_accuracy: 0.6600
Epoch 4/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.5583 -
accuracy: 0.7545 - val_loss: 0.6085 - val_accuracy: 0.6640
Epoch 5/27
8999/8999 [==============================] - 35s 4ms/step - loss: 0.5315 -
accuracy: 0.7665 - val_loss: 0.6052 - val_accuracy: 0.6690
Epoch 6/27
8999/8999 [==============================] - 37s 4ms/step - loss: 0.5113 -
accuracy: 0.7782 - val_loss: 0.6064 - val_accuracy: 0.6700
Epoch 7/27
8999/8999 [==============================] - 40s 4ms/step - loss: 0.4974 -
accuracy: 0.7863 - val_loss: 0.6119 - val_accuracy: 0.6725
Epoch 8/27
8999/8999 [==============================] - 41s 5ms/step - loss: 0.4849 -
accuracy: 0.7906 - val_loss: 0.6198 - val_accuracy: 0.6795
Epoch 9/27
8999/8999 [==============================] - 33s 4ms/step - loss: 0.4772 -
accuracy: 0.7941 - val_loss: 0.6252 - val_accuracy: 0.6900
Epoch 10/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.4681 -
accuracy: 0.7972 - val_loss: 0.6311 - val_accuracy: 0.6850
Epoch 11/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.4622 -
accuracy: 0.8009 - val_loss: 0.6501 - val_accuracy: 0.6840
Epoch 12/27
8999/8999 [==============================] - 38s 4ms/step - loss: 0.4565 -
accuracy: 0.8020 - val_loss: 0.6399 - val_accuracy: 0.6845
Epoch 13/27
8999/8999 [==============================] - 36s 4ms/step - loss: 0.4482 -
accuracy: 0.8053 - val_loss: 0.6621 - val_accuracy: 0.6780
Epoch 14/27
8999/8999 [==============================] - 29s 3ms/step - loss: 0.4449 -
accuracy: 0.8090 - val_loss: 0.6526 - val_accuracy: 0.6820
Epoch 15/27
8999/8999 [==============================] - 29s 3ms/step - loss: 0.4383 -
accuracy: 0.8085 - val_loss: 0.6689 - val_accuracy: 0.6780
Epoch 16/27
8999/8999 [==============================] - 29s 3ms/step - loss: 0.4331 -
accuracy: 0.8161 - val_loss: 0.6821 - val_accuracy: 0.6790
Epoch 17/27
8999/8999 [==============================] - 37s 4ms/step - loss: 0.4259 -
accuracy: 0.8140 - val_loss: 0.7072 - val_accuracy: 0.6825
Epoch 18/27
8999/8999 [==============================] - 33s 4ms/step - loss: 0.4201 -
```

```
accuracy: 0.8213 - val_loss: 0.7068 - val_accuracy: 0.6825
Epoch 19/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.4158 -
accuracy: 0.8211 - val_loss: 0.7394 - val_accuracy: 0.6785
Epoch 20/27
8999/8999 [==============================] - 35s 4ms/step - loss: 0.4076 -
accuracy: 0.8295 - val_loss: 0.7352 - val_accuracy: 0.6790
Epoch 21/27
8999/8999 [==============================] - 35s 4ms/step - loss: 0.4044 -
accuracy: 0.8271 - val_loss: 0.7492 - val_accuracy: 0.6830
Epoch 22/27
8999/8999 [==============================] - 33s 4ms/step - loss: 0.3969 -
accuracy: 0.8316 - val_loss: 0.7681 - val_accuracy: 0.6850
Epoch 23/27
8999/8999 [==============================] - 33s 4ms/step - loss: 0.3913 -
accuracy: 0.8378 - val_loss: 0.7672 - val_accuracy: 0.6760
Epoch 24/27
8999/8999 [==============================] - 33s 4ms/step - loss: 0.3858 -
accuracy: 0.8403 - val_loss: 0.7706 - val_accuracy: 0.6860
Epoch 25/27
8999/8999 [==============================] - 32s 4ms/step - loss: 0.3794 -
accuracy: 0.8421 - val_loss: 0.7829 - val_accuracy: 0.6795
Epoch 26/27
8999/8999 [==============================] - 31s 3ms/step - loss: 0.3772 -
accuracy: 0.8434 - val_loss: 0.7824 - val_accuracy: 0.6810
Epoch 27/27
8999/8999 [==============================] - 32s 4ms/step - loss: 0.3736 -
accuracy: 0.8473 - val_loss: 0.8179 - val_accuracy: 0.6760
```

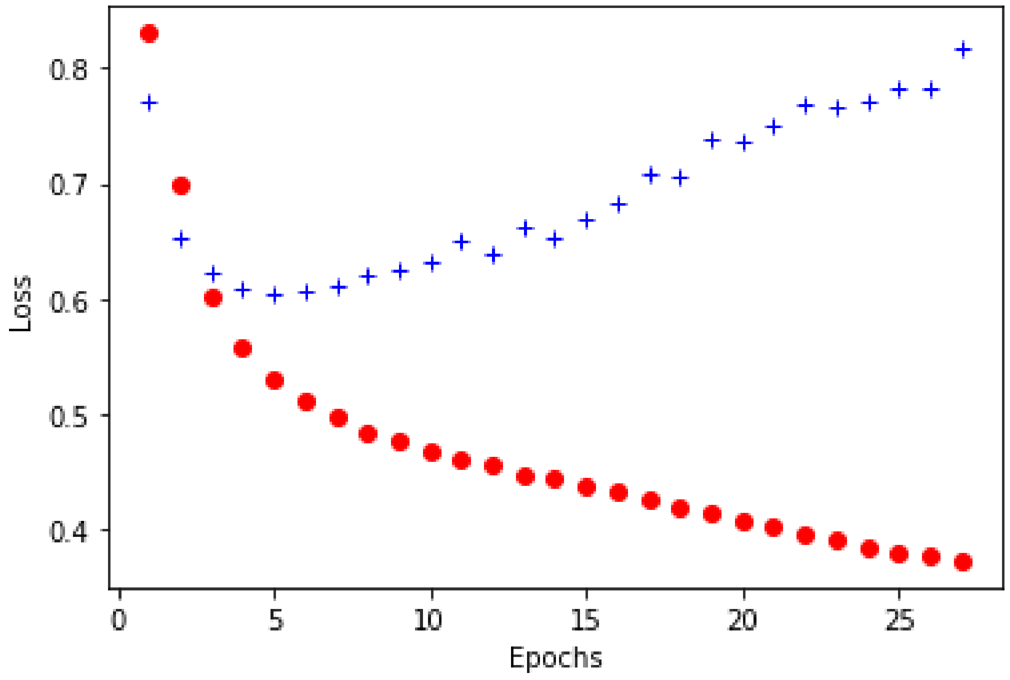
[527]:
```
pd.DataFrame(history2.history)
```

[527]:
```
    val_loss  val_accuracy      loss  accuracy
0   0.771186        0.6040  0.830200  0.580453
1   0.653876        0.6595  0.698771  0.679075
2   0.622557        0.6600  0.601621  0.732693
3   0.608541        0.6640  0.558349  0.754528
4   0.605196        0.6690  0.531537  0.766530
5   0.606366        0.6700  0.511270  0.778198
6   0.611906        0.6725  0.497436  0.786310
7   0.619846        0.6795  0.484859  0.790643
8   0.625208        0.6900  0.477162  0.794144
9   0.631108        0.6850  0.468113  0.797200
10  0.650140        0.6840  0.462233  0.800922
11  0.639852        0.6845  0.456495  0.801978
12  0.662095        0.6780  0.448194  0.805312
13  0.652587        0.6820  0.444916  0.809034
14  0.668928        0.6780  0.438337  0.808479
```

```
15  0.682064        0.6790  0.433148  0.816146
16  0.707233        0.6825  0.425947  0.813979
17  0.706842        0.6825  0.420078  0.821314
18  0.739445        0.6785  0.415825  0.821147
19  0.735158        0.6790  0.407620  0.829537
20  0.749153        0.6830  0.404413  0.827147
21  0.768057        0.6850  0.396867  0.831592
22  0.767173        0.6760  0.391251  0.837815
23  0.770641        0.6860  0.385789  0.840316
24  0.782936        0.6795  0.379435  0.842094
25  0.782365        0.6810  0.377152  0.843427
26  0.817878        0.6760  0.373553  0.847316
```

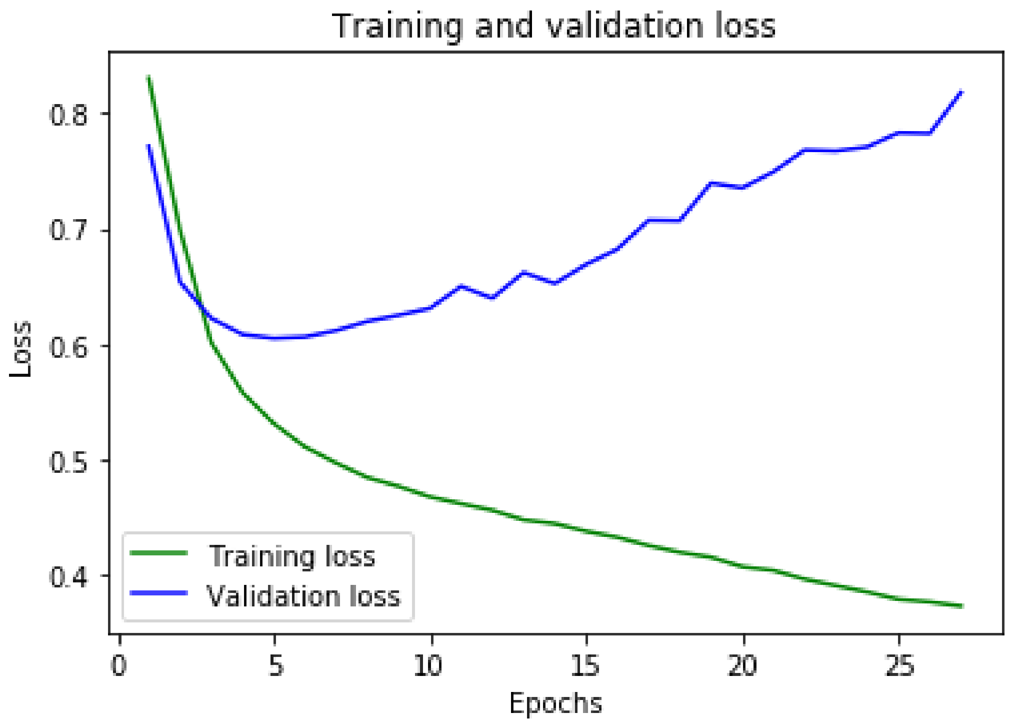
[538]:
```
from matplotlib import pyplot as plt
history_dict=history2.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'ro')
plt.plot(epochs, val_loss_values, 'b+')
plt.xlabel('Epochs')
plt.ylabel('Loss')
```

[538]:
```
Text(0, 0.5, 'Loss')
```



[ ]:

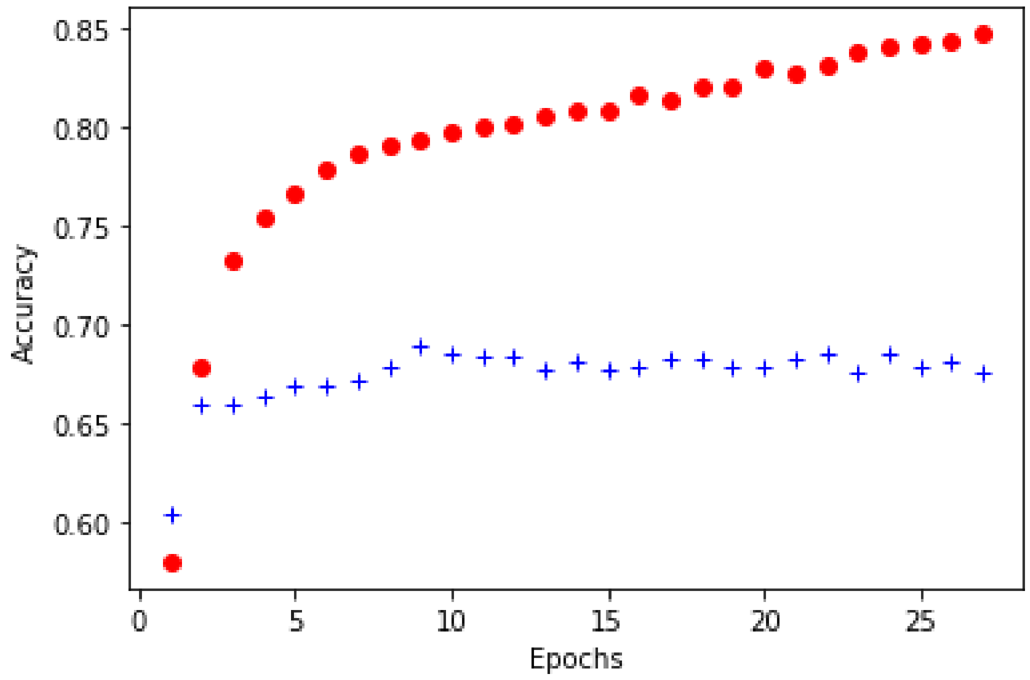
[539]:
```
plt.clf()
loss = history2.history['loss']
val_loss = history2.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'g', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



[ ]:

[540]:
```
loss_values = history_dict['accuracy']
val_loss_values = history_dict['val_accuracy']
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, 'ro')
plt.plot(epochs, val_loss_values, 'b+')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
[540]: Text(0, 0.5, 'Accuracy')
```

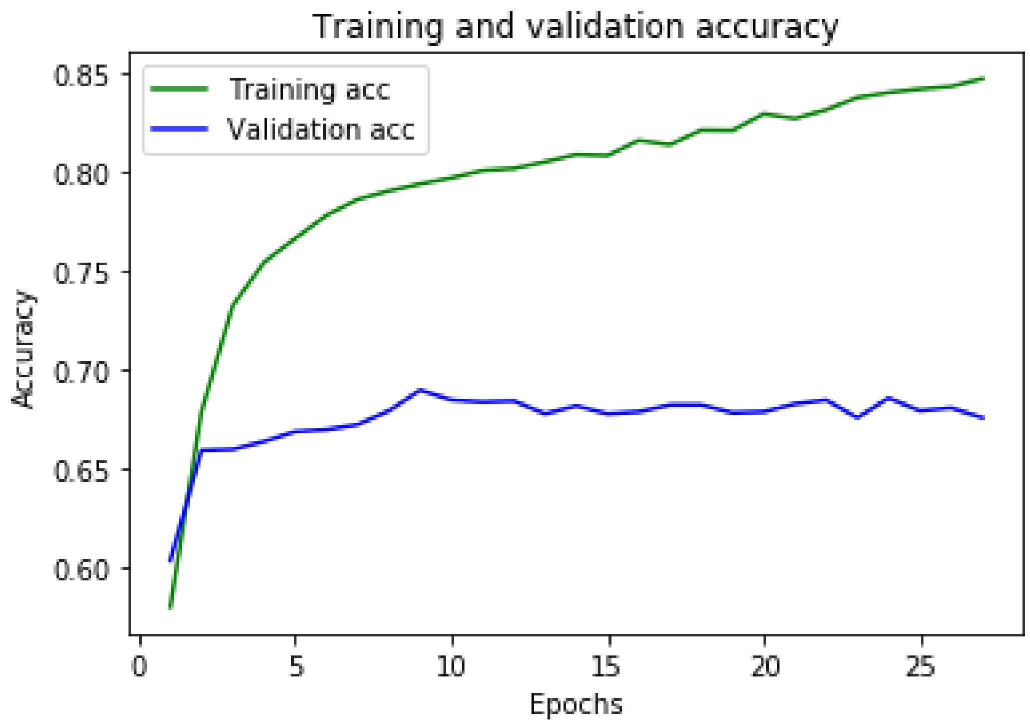


```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[541]: plt.clf()
acc = history2.history['accuracy']
val_acc = history2.history['val_accuracy']
plt.plot(epochs, acc, 'g', 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()
plt.show()
```



[547]:
```
from keras.regularizers import l2
from keras.optimizers import Adam
embed_dim = 128
lstm_out = 196
#adam =Adam(lr =1e-8)
model3 = Sequential()
model3.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
 ↪mask_zero=True))
#model1.add(SpatialDropout1D(0.4))
model3.add(LSTM(lstm_out,kernel_regularizer=l2(0.001), dropout_U=0.2,␣
 ↪dropout_W=0.2))

model3.add(Dense(2,activation='sigmoid'))
model3.compile(loss = 'binary_crossentropy', optimizer='adam',metrics =␣
 ↪['accuracy'])
print(model3.summary())

y = pd.get_dummies(data['target']).values
print(y)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.1,␣
 ↪random_state = 42)
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
callback = [EarlyStopping(monitor='val_loss',␣
↪patience=3),ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',␣
↪save_best_only=True)]
history3= model3.fit(X_train, y_train,batch_size=32,callbacks =callback,␣
↪epochs=15,validation_data=(X_test, y_test))
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:9: UserWarning: Update your `LSTM` call to the
Keras 2 API: `LSTM(196, kernel_regularizer=<keras.reg…, dropout=0.2,
recurrent_dropout=0.2)`
  if __name__ == '__main__':
```

```
Model: "sequential_122"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_99 (Embedding)     (None, 5, 128)            256000
_________________________________________________________________
lstm_111 (LSTM)              (None, 196)               254800
_________________________________________________________________
dense_130 (Dense)            (None, 2)                 394
=================================================================
Total params: 511,194
Trainable params: 511,194
Non-trainable params: 0
_________________________________________________________________
None
[[1 0]
 [1 0]
 [1 0]
 …
 [0 1]
 [0 1]
 [0 1]]
(8999, 5) (8999, 2)
(1000, 5) (1000, 2)
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

```
Train on 8999 samples, validate on 1000 samples
Epoch 1/15
  64/8999 […] - ETA: 5:26:17 - loss: 0.9096 -
accuracy: 0.5625
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
```

```
packages/keras/callbacks/callbacks.py:95: RuntimeWarning: Method
(on_train_batch_end) is slow compared to the batch update (0.860301). Check your
callbacks.
  % (hook_name, delta_t_median), RuntimeWarning)
```

```
8999/8999 [==============================] - 191s 21ms/step - loss: 0.6430 -
accuracy: 0.6601 - val_loss: 0.5748 - val_accuracy: 0.7045
Epoch 2/15
8999/8999 [==============================] - 48s 5ms/step - loss: 0.5179 -
accuracy: 0.7590 - val_loss: 0.5678 - val_accuracy: 0.7100
Epoch 3/15
8999/8999 [==============================] - 44s 5ms/step - loss: 0.4819 -
accuracy: 0.7808 - val_loss: 0.5890 - val_accuracy: 0.7045
Epoch 4/15
8999/8999 [==============================] - 42s 5ms/step - loss: 0.4543 -
accuracy: 0.8023 - val_loss: 0.6045 - val_accuracy: 0.6980
Epoch 5/15
8999/8999 [==============================] - 47s 5ms/step - loss: 0.4237 -
accuracy: 0.8159 - val_loss: 0.6470 - val_accuracy: 0.6905
```

[552]:
```
embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
 ↪dropout=0.2))
model.add(LSTM(lstm_out, dropout_U=0.2, dropout_W=0.2))
model.add(Dense(2,activation='sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer='adagrad',metrics =␣
 ↪['accuracy'])
print(model.summary())

y = pd.get_dummies(data['target']).values
print(y)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.33,␣
 ↪random_state = 42)
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
#history1 = model.fit(X_train,␣
 ↪y_train,batch_size=32,epochs=10,validation_data=(X_test, y_test))
batch_size = 32
model.fit(X_train, y_train, nb_epoch = 7, batch_size=batch_size, verbose = 2)

#classifier.summary()
#callback = [EarlyStopping(monitor='val_loss',␣
 ↪patience=2),ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',␣
 ↪save_best_only=True)]
```

```
#history1 = model.fit(X_train,␣
↪y_train,batch_size=32,epochs=10,validation_data=(X_test, y_test))
validation_size = 200

X_validate = X_test[-validation_size:]
Y_validate = y_test[-validation_size:]
X_test = X_test[:-validation_size]
Y_test = y_test[:-validation_size]
score,acc = model.evaluate(X_test, Y_test, verbose = 2, batch_size = batch_size)
print("score: %.2f" % (score))
print("acc: %.2f" % (acc))

pos_cnt, neg_cnt, pos_correct, neg_correct = 0, 0, 0, 0
for x in range(len(X_validate)):

    result = model.predict(X_validate[x].reshape(1,X_test.
↪shape[1]),batch_size=32,verbose = 2)[0]

    if np.argmax(result) == np.argmax(Y_validate[x]):
        if np.argmax(Y_validate[x]) == 0:
            neg_correct += 1
        else:
            pos_correct += 1

    if np.argmax(Y_validate[x]) == 0:
        neg_cnt += 1
    else:
        pos_cnt += 1



print("pos_acc", pos_correct/pos_cnt*100, "%")
print("neg_acc", neg_correct/neg_cnt*100, "%")
```

```
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:5: UserWarning: The `dropout` argument is no
longer support in `Embedding`. You can apply a `keras.layers.SpatialDropout1D`
layer right after the `Embedding` layer to get the same behavior.
  """
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:6: UserWarning: Update your `LSTM` call to the
Keras 2 API: `LSTM(196, dropout=0.2, recurrent_dropout=0.2)`


Model: "sequential_123"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_100 (Embedding)    (None, 5, 128)          256000
_________________________________________________________________
lstm_112 (LSTM)              (None, 196)             254800
_________________________________________________________________
dense_131 (Dense)            (None, 2)               394
=================================================================
Total params: 511,194
Trainable params: 511,194
Non-trainable params: 0
_________________________________________________________________
None
[[1 0]
 [1 0]
 [1 0]
 …
 [0 1]
 [0 1]
 [0 1]]
(6699, 5) (6699, 2)
(3300, 5) (3300, 2)

/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:18: UserWarning: The `nb_epoch` argument in `fit`
has been renamed `epochs`.
/Users/michelamaineri/opt/anaconda3/lib/python3.7/site-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "

Epoch 1/7
 - 32s - loss: 0.6077 - accuracy: 0.6617
Epoch 2/7
 - 20s - loss: 0.4844 - accuracy: 0.7674
Epoch 3/7
 - 22s - loss: 0.4318 - accuracy: 0.8025
Epoch 4/7
 - 21s - loss: 0.3893 - accuracy: 0.8315
Epoch 5/7
 - 20s - loss: 0.3509 - accuracy: 0.8430
Epoch 6/7
 - 20s - loss: 0.3235 - accuracy: 0.8609
Epoch 7/7
 - 20s - loss: 0.3024 - accuracy: 0.8709
score: 0.80
acc: 0.70
pos_acc 73.45132743362832 %
neg_acc 63.2183908045977 %
```

[ ]:

[ ]:

[553]:
```
print(result)
```

```
[0.07031877 0.9291189 ]
```

[ ]:

[ ]:

[ ]:

[337]:
```
import os
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import advanced_activations
from keras.optimizers import Adam

from sklearn.model_selection import StratifiedKFold
   # Parameters
learning_rate = 0.001
num_steps = 15
batch_size = 128
n_fold = 5

# Network Parameters
n_hidden_1 = 25 # 1st layer number of neurons
n_hidden_2 = 10 # 2nd layer number of neurons
#num_input = 6 # MNIST data input (img shape: 28*28)
num_classes = 1 # MNIST total classes (0-9 digits)


#
activationFun = 'relu'

class CreateNN:
    def __init__(self,**kargs):
        self.X_train = kargs['xt']
        self.y_train = kargs['yt']
        self.kFold = kargs['kf']
        self.i = 1
        self.num_input = self.X_train.shape[1]
    def modelDefinition(self):
        #logger.info('DEFINITION OF THE MODEL')
        self.model = Sequential()
        self.model.add(Dense(self.num_input, input_dim = self.
↪num_input,activation=activationFun))
        self.model.add(Dense(n_hidden_1,activation = activationFun))
        self.model.add(Dense(n_hidden_2,activation = activationFun))
        self.model.add(Dense(num_classes,activation = 'sigmoid'))
        print(self.model.summary())

    def modelCompile(self):
        #logger.info('COMPILATION OF THE MODEL')
        adam = Adam(lr = learning_rate)
        self.model.compile(loss = 'binary_crossentropy', optimizer =␣
↪adam,metrics = ['accuracy'])

    def modelEval(self):
        #logger.info('EVALUATION OF THE MODEL')
        totalScores = list()
        #logger.info('START OF THE CROSS VALIDATION')
        for X_train,y_test in self.kFold.split(self.X_train, self.Y_train):
            data.iloc[text]('WORKING ON FOLD %i',self.i)
            print('train set',train)
            history = self.model.fit(self.X_train.iloc[train], self.y_train.
↪iloc[train],
                                     epochs=num_steps,
                                     batch_size = batch_size)␣
↪#validation_data=(self.X_train.iloc[test], self.Y_train.iloc[test])
            scores = self.model.evaluate(self.X_train.iloc[test], self.y_train.
↪iloc[test])
            totalScores.append(scores[1])
            self.i += 1
        return history, self.model, totalScores

def main():
    #Inizialization of the class LoadData
    #logger.info('INIZIALIZATION OF LOADDATA')
    #ld = LoadData(tr=trainFile)
    #df2Pred = ld.readFiles(predFile)
    #X_train, Y_train = ld.prepareTrain()
    kfold = StratifiedKFold(n_splits=n_fold)
    #logger.info('INIZIALIZATION OF CreateNN')
    cnn = CreateNN(xt=X_train,yt=y_train,kf=kfold)
    cnn.modelDefinition()
    cnn.modelCompile()
    history, model, totalScores = cnn.modelEval()
    #logger.info('EVALUATION COMPLETED')
    #logger.info("FOR THE ACTUAL MODEL THE RESULTS OF %s IS: %.2f%%+/-%.2f%%" %␣
↪(model.metrics_names[1], np.mean(totalScores),np.std(totalScores)))
    return X_train, y_train, history,totalScores

X_train,y_train, history,totalScores = main()
```

```
Model: "sequential_77"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_87 (Dense)             (None, 5)                 30
_________________________________________________________________
dense_88 (Dense)             (None, 25)                150
_________________________________________________________________
dense_89 (Dense)             (None, 10)                260
_________________________________________________________________
dense_90 (Dense)             (None, 1)                 11
=================================================================
Total params: 451
Trainable params: 451
Non-trainable params: 0
_________________________________________________________________
None
```

```
       ␣
↪---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call␣
↪last)

        <ipython-input-337-7aad066840bb> in <module>
         77     return X_train, y_train, history,totalScores
         78
    ---> 79 X_train,y_train, history,totalScores = main()


        <ipython-input-337-7aad066840bb> in main()
         72     cnn.modelDefinition()
         73     cnn.modelCompile()
    ---> 74     history, model, totalScores = cnn.modelEval()
         75     #logger.info('EVALUATION COMPLETED')
         76     #logger.info("FOR THE ACTUAL MODEL THE RESULTS OF %s IS: %.2f%%+/
↪-%.2f%%" % (model.metrics_names[1], np.mean(totalScores),np.std(totalScores)))


        <ipython-input-337-7aad066840bb> in modelEval(self)
         50         totalScores = list()
         51         #logger.info('START OF THE CROSS VALIDATION')
    ---> 52         for X_train,y_test in self.kFold.split(self.X_train, self.
↪Y_train):
         53             data.iloc[text]('WORKING ON FOLD %i',self.i)
         54             print('train set',train)


    AttributeError: 'CreateNN' object has no attribute 'Y_train'
```

[278]:
```
import pandas as pd
class CreateRNN:
    def __init__(classifier,**kargs):
        classifier.X_train = kargs['xt']
        classifier.y_train = kargs['yt']
        classifier.kFold = kargs['kf']
        classifier.i = 1
        classifier.num_input = classifier.X_train.shape[1]

    def modelDefinition(classifier):
        #logger.info('DEFINITION OF THE MODEL')
        #self.model = Sequential()
        #self.model.add(Dense(self.num_input, input_dim = self.
↪num_input,activation=activationFun))
        #self.model.add(Dense(n_hidden_1,activation = activationFun))
        #self.model.add(Dense(n_hidden_2,activation = activationFun))
        #self.model.add(Dense(num_classes,activation = 'sigmoid'))
        #print(self.model.summary())
        classifier.model = Sequential()
        classifier.model.add(LSTM(200,dropout=0.3,recurrent_dropout=0.3,␣
↪return_sequences=False))
        classifier.model.add(Dense(2, activation='sigmoid'))
        classifier.model.add(Dense(2,activation = activationFun))
        classifier.model.compile(loss = 'binary_crossentropy',␣
↪optimizer='adam',metrics = ['accuracy'])
        print(classifier.model.summary())

def modelCompile():
        #logger.info('COMPILATION OF THE MODEL')
        adam = Adam(lr = learning_rate)
        classifier.model.compile(loss = 'binary_crossentropy', optimizer =␣
↪adam,metrics = ['accuracy'])
def modelEval(classifier):
        #logger.info('EVALUATION OF THE MODEL')
        totalScores = list()
        #logger.info('START OF THE CROSS VALIDATION')
        #for train,test in self.kFold.split(self.X_train, self.Y_train):
```

```
        #   logger.info('WORKING ON FOLD %i',self.i)
        for X_train,y_test in self.kFold.split(self.X_train, self.Y_train):
            data.iloc[text]('WORKING ON FOLD %i',self.i)
            print('train set',train)
            history = classifier.model.fit(self.X_train.iloc[train], classifier.
↪y_train.iloc[train],
                                 epochs=num_steps,
                                 batch_size = batch_size)␣
↪#validation_data=(self.X_train.iloc[test], self.Y_train.iloc[test])
            scores = classifier.model.evaluate(classifier.X_train.iloc[test],␣
↪classifier.y_train.iloc[test])
            totalScores.append(scores[1])
            self.i += 1
        return history, self.model, totalScores

def main():
    #Inizialization of the class LoadData
    #logger.info('INIZIALIZATION OF LOADDATA')
    #ld = LoadData(tr=trainFile)
    #df2Pred = ld.readFiles(predFile)
    #X_train, Y_train = ld.prepareTrain()
    kfold = StratifiedKFold(n_splits=n_fold)
    #logger.info('INIZIALIZATION OF CreateNN')
    RNN = CreateRNN(xt=X_train,yt=y_train,kf=kfold)
    RNN.modelDefinition()
    RNN.modelCompile()
    history, model, totalScores = RNN.modelEval()
    #logger.info('EVALUATION COMPLETED')
    #logger.info("FOR THE ACTUAL MODEL THE RESULTS OF %s IS: %.2f%%+/-%.2f%%" %␣
↪(model.metrics_names[1], np.mean(totalScores),np.std(totalScores)))
    return X_train, y_train, history,totalScores

X_train,y_train, history,totalScores = main()
```

```
      ␣
↪---------------------------------------------------------------------------

      ValueError                                Traceback (most recent call␣
↪last)

      <ipython-input-278-22767df0278b> in <module>
       60     return X_train, y_train, history,totalScores
       61
  ---> 62 X_train,y_train, history,totalScores = main()
```

```
<ipython-input-278-22767df0278b> in main()
 53     #logger.info('INIZIALIZATION OF CreateNN')
 54     RNN = CreateRNN(xt=X_train,yt=y_train,kf=kfold)
---> 55     RNN.modelDefinition()
 56     RNN.modelCompile()
 57     history, model, totalScores = RNN.modelEval()


<ipython-input-278-22767df0278b> in modelDefinition(classifier)
 21         classifier.model.add(Dense(2,activation = activationFun))
 22         classifier.model.compile(loss = 'binary_crossentropy',␣
↪optimizer='adam',metrics = ['accuracy'])
---> 23         print(classifier.model.summary())
 24
 25 def modelCompile():


~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/network.py in␣
↪summary(self, line_length, positions, print_fn)
1318         if not self.built:
1319             raise ValueError(
-> 1320                 'This model has not yet been built. '
1321                 'Build the model first by calling build() '
1322                 'or calling fit() with some data. '


ValueError: This model has not yet been built. Build the model first by␣
↪calling build() or calling fit() with some data. Or specify input_shape or␣
↪batch_input_shape in the first layer for automatic build.
```

[ ]:

[ ]:

[ ]:

[265]:
```
def main():
    #Inizialization of the class LoadData
    #logger.info('INIZIALIZATION OF LOADDATA')
    #ld = LoadData(tr=trainFile)
    #df2Pred = ld.readFiles(predFile)
    #X_train, Y_train = ld.prepareTrain()
    kfold = StratifiedKFold(n_splits=n_fold)
    #logger.info('INIZIALIZATION OF CreateNN')
    RNN = CreateRNN(xt=X_train,yt=y_train,kf=kfold)
    RNN.modelDefinition()
    RNN.modelCompile()
    history, model, totalScores = RNN.modelEval()
    #logger.info('EVALUATION COMPLETED')
    #logger.info("FOR THE ACTUAL MODEL THE RESULTS OF %s IS: %.2f%%+/-%.2f%%" %␣
↪(model.metrics_names[1], np.mean(totalScores),np.std(totalScores)))
    return X_train, y_train, history,totalScores

X_train,y_train, history,totalScores = main()
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-265-f3cd41510ecf> in <module>
        15     return X_train, y_train, history,totalScores
        16 
   ---> 17 X_train,y_train, history,totalScores = main()


       <ipython-input-265-f3cd41510ecf> in main()
         8     #logger.info('INIZIALIZATION OF CreateNN')
         9     RNN = CreateRNN(xt=X_train,yt=y_train,kf=kfold)
   ---> 10     RNN.modelDefinition()
        11     RNN.modelCompile()
        12     history, model, totalScores = RNN.modelEval()


       <ipython-input-264-1d39ffa2188d> in modelDefinition(classifier)
        20         #self.model.add(Dense(n_hidden_2,activation = activationFun))
        21         classifier.model.compile(loss = 'binary_crossentropy',␣
↪optimizer='adam',metrics = ['accuracy'])
   ---> 22         print(classifier.model.summary())
        23 
        24 def modelCompile():


       ~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/network.py in␣
↪summary(self, line_length, positions, print_fn)
      1318         if not self.built:
      1319             raise ValueError(
   -> 1320                 'This model has not yet been built. '
      1321                 'Build the model first by calling build() '
      1322                 'or calling fit() with some data. '
```

```
        ValueError: This model has not yet been built. Build the model first by␣
↪calling build() or calling fit() with some data. Or specify input_shape or␣
↪batch_input_shape in the first layer for automatic build.
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```