# SENTIMENT ANALYSIS BY LSTM ON HONG KONG TWEETS

Maineri Michela
4804458
Empirical research

## Our Goal:
## to understand whether sentiment tweets about Hong Kong are positive or negative

### What is sentiment analysis?

It's an automated process that uses AI to identify positive, negative opinions from text. In a world where we generate 2.5 quintillion bytes of data every day, sentiment analysis is useful for social media monitoring as it allows to gain an overview of the wider public opinion behind certain topics.

### How to perform it?

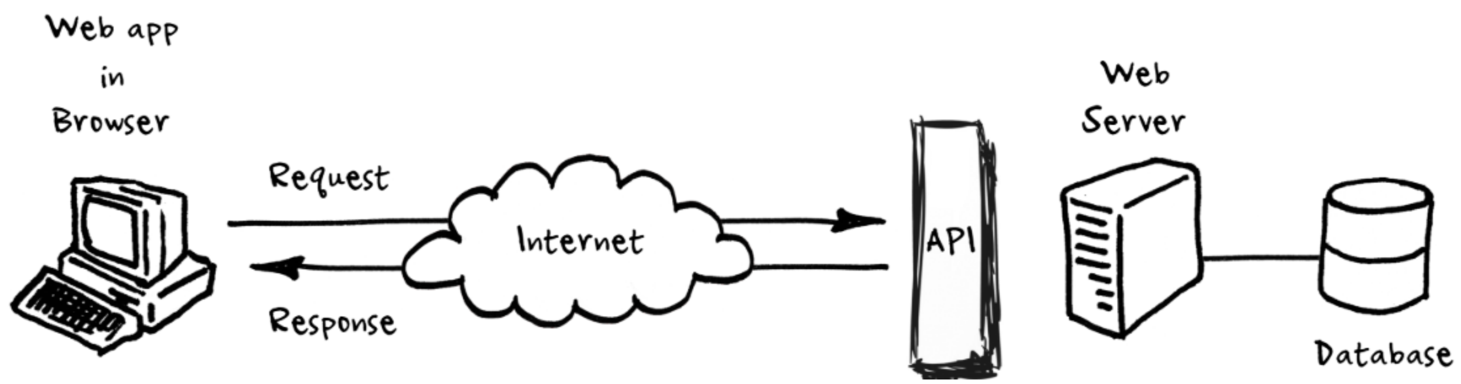
Neural network → since the inputs are texts, the most suitable NN to be used is a Recurrent Neural Network.

words/sentences = sequential inputs → we are going to build a **LSTM** network on a very large training dataset in order then to predict and capture the sentiment on a dataset exclusively composed of Hong Kong tweets.

## Creation of Hong Kong tweets dataset
on which we will accomplish our final prediction

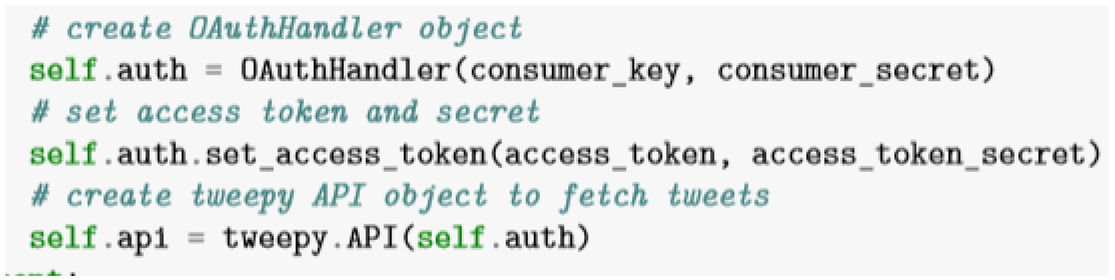
Tweepy.API

Request Twitter Developers Account to obtain consumer key and access token



Application Programming Interface is an interface or communication protocol between a client and a server intended to simplify the building of client-side software.

Tweepy library is open-sourced and enables Python to communicate with Twitter platform and to use its API. Tweepy supports accessing Twitter via Basic Authentication and OAuth method



```
# 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)
```

create a code that allow us to collect tweets text on whatever topic we are interested in → we proceed in defining `Python Class`: a blueprint for an object, that can contain variables and functions (called method) in order to get our dataset ready-to-use

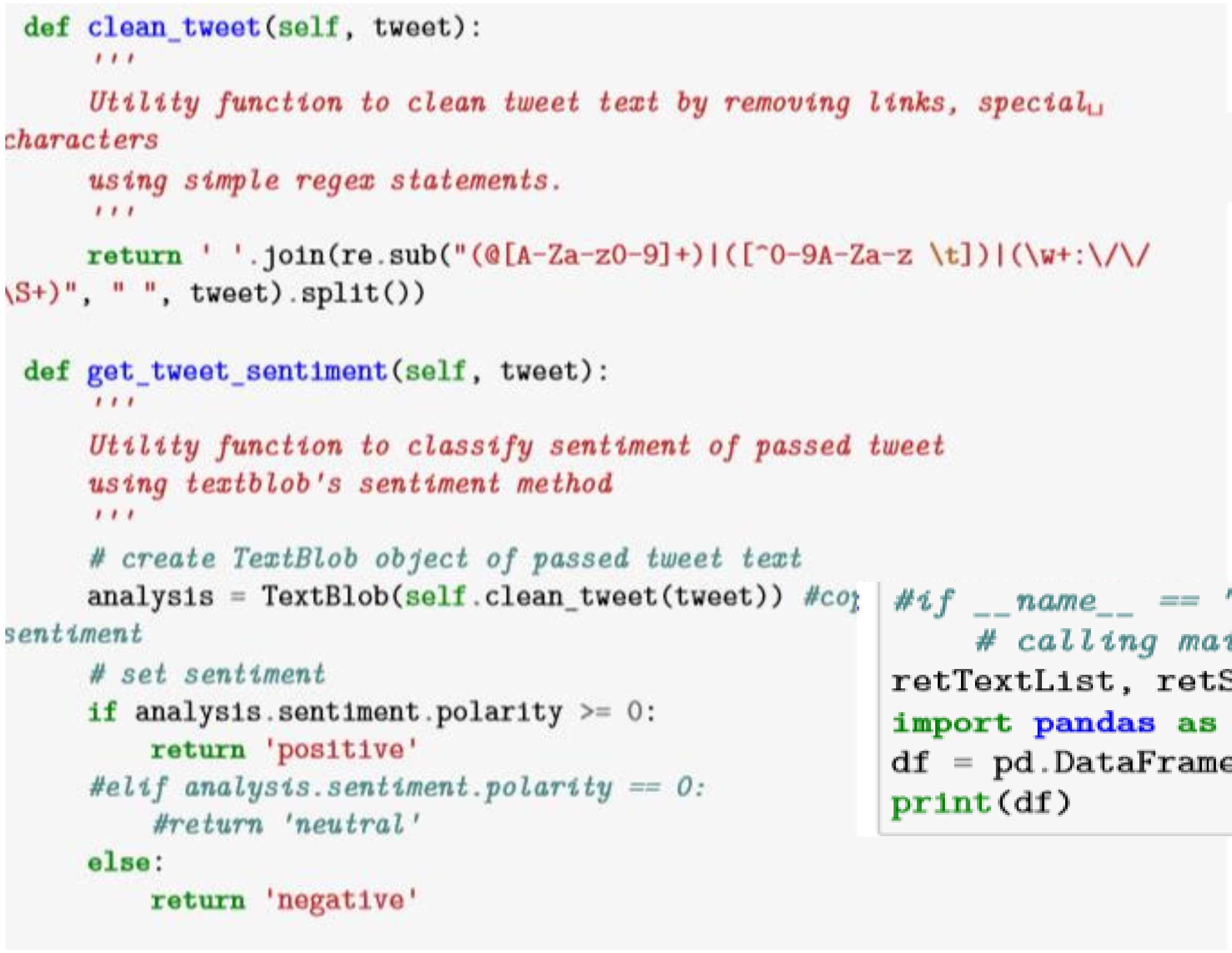
Textual inputs → require also a preprocessing textual transformation :

-Utility function to clean tweet text by removing links/ special character/ punctuation

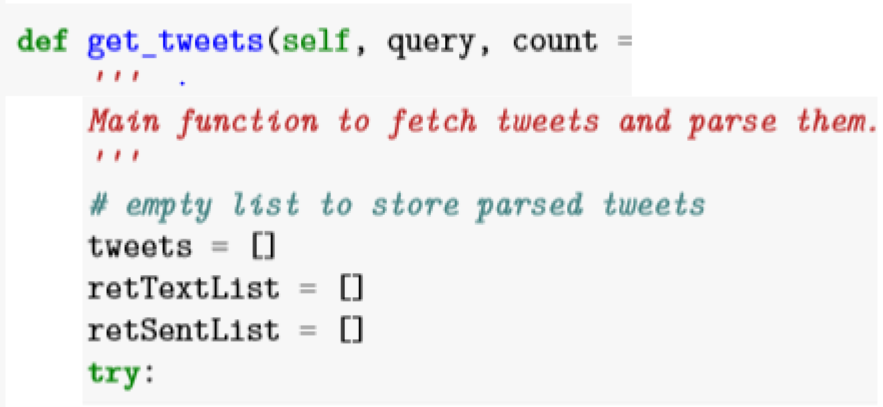
-Utility function to classify sentiment of passed tweet: create TextBlob object of passed tweet text analysis (i.e. analysis.sentiment.polarity >= 0: positive). It returns a quick sentiment analysis.

-Main function to fetch tweets and parse them. fetched_tweets = self.api.search(query ='Hong Kong', count =200)

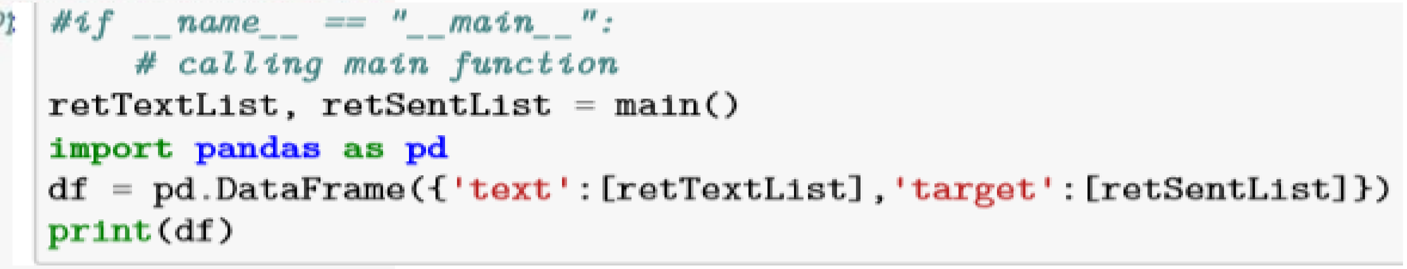
# parsing tweets one by one



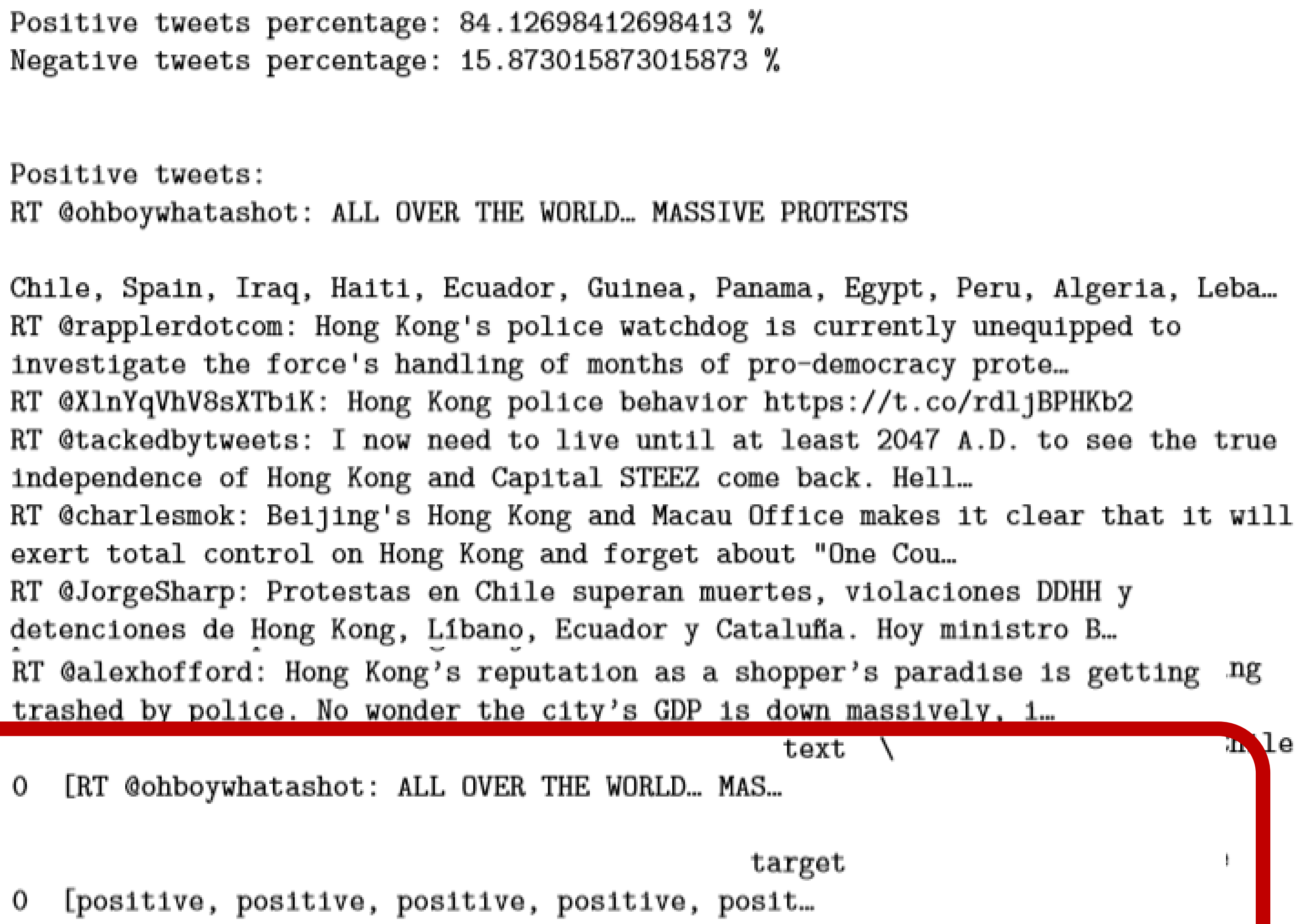
```
 def clean_tweet(self, tweet):
     '''
     Utility function to clean tweet text by removing links, special␣
characters
     using simple regex statements.
     '''
     return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/
\S+)", " ", tweet).split())

 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)) #co
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 =
    '''
    Main function to fetch tweets and parse them.
    '''
    # empty list to store parsed tweets
    tweets = []
    retTextList = []
    retSentList = []
    try:
```

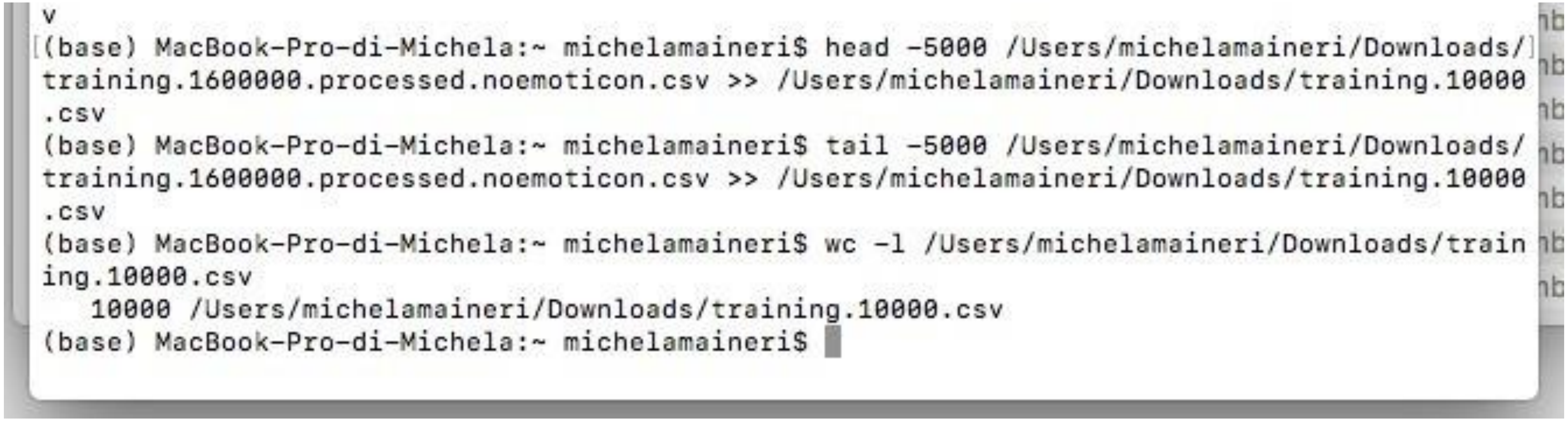


```
#if __name__ == "__main__":
    # calling main function
retTextList, retSentList = main()
import pandas as pd
df = pd.DataFrame({'text':[retTextList],'target':[retSentList]})
print(df)
```

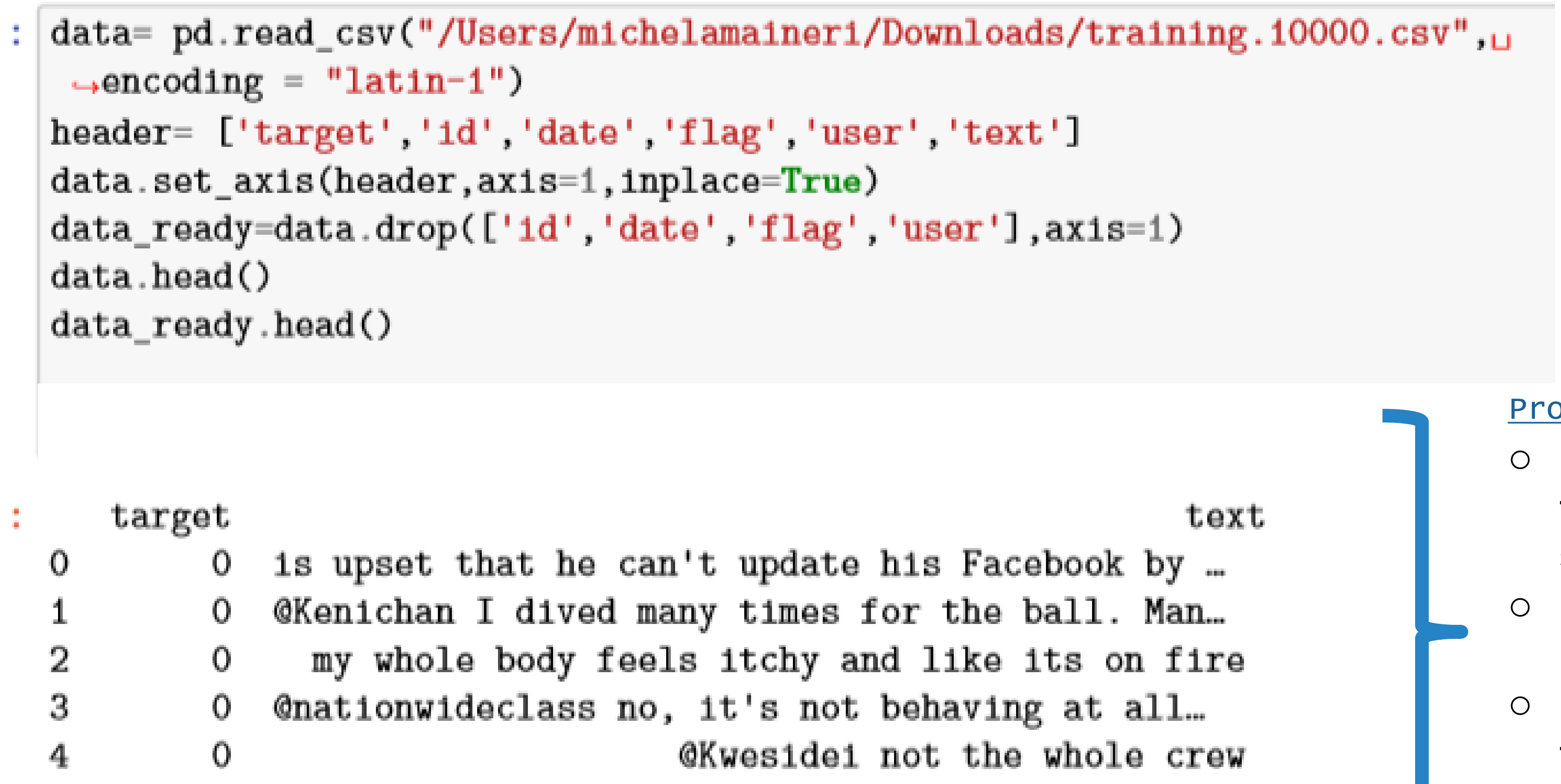


```
Positive tweets percentage: 84.12698412698413 %
Negative tweets percentage: 15.873015873015873 %


Positive tweets:
RT @ohboywhatashot: ALL OVER THE WORLD… MASSIVE PROTESTS

Chile, Spain, Iraq, Haiti, Ecuador, Guinea, Panama, Egypt, Peru, Algeria, Leba…
RT @rapplerdotcom: Hong Kong's police watchdog is currently unequipped to
investigate the force's handling of months of pro-democracy prote…
RT @XlnYqVhV8sXTbiK: Hong Kong police behavior https://t.co/rdljBPHKb2
RT @tackedbytweets: I now need to live until at least 2047 A.D. to see the true
independence of Hong Kong and Capital STEEZ come back. Hell…
RT @charlesmok: Beijing's Hong Kong and Macau Office makes it clear that it will
exert total control on Hong Kong and forget about "One Cou…
RT @JorgeSharp: Protestas en Chile superan muertes, violaciones DDHH y
detenciones de Hong Kong, Líbano, Ecuador y Cataluña. Hoy ministro B…
RT @alexhofford: Hong Kong’s reputation as a shopper’s paradise is getting   ng
trashed by police. No wonder the city’s GDP is down massively, i…
                                                     text  \                 le
0  [RT @ohboywhatashot: ALL OVER THE WORLD… MAS…

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

## Training dataset.csv


```
v
(base) MacBook-Pro-di-Michela:~ michelamaineri$ head -5000 /Users/michelamaineri/Downloads/
training.1600000.processed.noemoticon.csv >> /Users/michelamaineri/Downloads/training.10000
.csv
(base) MacBook-Pro-di-Michela:~ michelamaineri$ tail -5000 /Users/michelamaineri/Downloads/
training.1600000.processed.noemoticon.csv >> /Users/michelamaineri/Downloads/training.10000
.csv
(base) MacBook-Pro-di-Michela:~ michelamaineri$ wc -l /Users/michelamaineri/Downloads/train
ing.10000.csv
   10000 /Users/michelamaineri/Downloads/training.10000.csv
(base) MacBook-Pro-di-Michela:~ michelamaineri$ 
```

## Data used to create the model



```
: 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()
```

```
:    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
```

Processing textual input:

- use Tokenizer to vectorize the text and convert it into sequence of integers;
- restrict the tokenizer to use only top most common words;
- use pad_sequences to convert the sequences into 2-D numpy array.

As we already know, RECURRENT NEURAL NETWORK main feauters are:

- shares the same weights across several time steps
- Each unit of the output is a function of the previous members of the output
- Each unit of the output is produced using the same update rule applied to the previous outputs

• Sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence

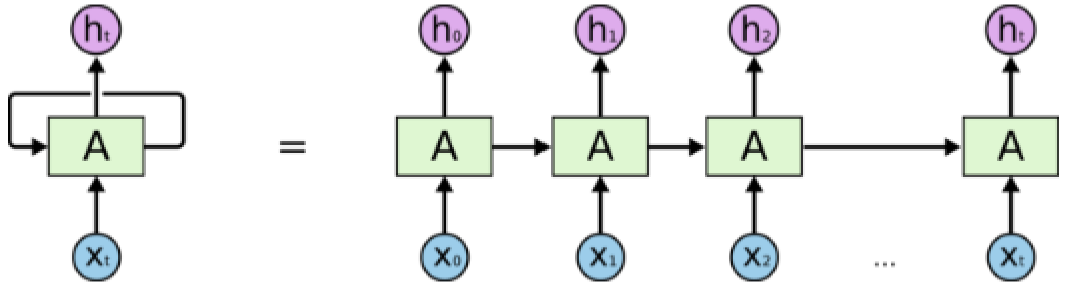
• We introduce cycles in the network to represent the influence of the present value of a variable on its own value at a future time step

- **RNN** can model sequence of data so that each sample can be assumed to be dependent on previous ones.

The formula for the current state is $h_t = f(h_{t-1}, x_t)$

Disadvantages: The exploding gradients problem refers to the large increase in the norm of the gradient during training. Such events are caused by the explosion of the long term components, which can grow exponentially more then short term ones The vanishing gradients problem refers to the opposite behaviour, when long term components go exponentially fast to norm 0, making it impossible for the model to learn correlation between temporally distant event.

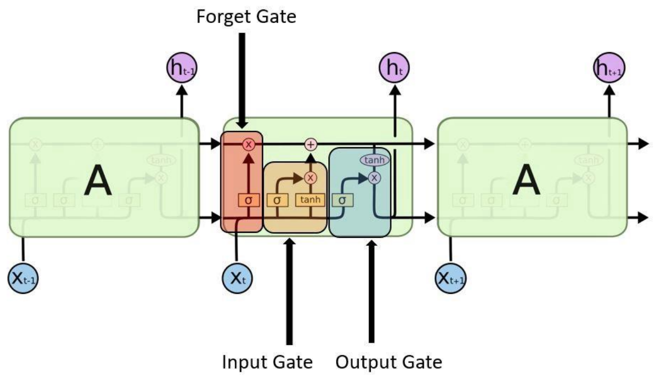
Capturing long range connections in a sequence: • Long Short Term Memory (Hochreiter and Schmidhuber 1997)



An unrolled recurrent neural network.

## What is Long Short Term Memory?

In LSTM, our model learns what information to store in long term memory and what to get rid of.



### LSTM Architecture

**1. Input gate** — discover which value from input should be used to modify the memory. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance rangingfrom**-1** to **1.**

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

**2. Forget gate** — discover what details to be discarded from the block. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input (Xt) and outputs a number between 0(*omit this*)and 1(*keep this*)for each number in the cell state Ct−1

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

**3. Output gate** — the input and the memory of the block is used to decide the output. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from**-1** to **1** and multiplied with output of **Sigmoid.**

$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

## Building our LSTM network

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

The sequential model is the standard to create a network with a sequence of layers, one after the other. Keras is a high-level neural networks API, focus on enabling fast experimentation.

```
keras.layers.Embedding(input_dim, output_dim,
embeddings_initializer='uniform',
embeddings_regularizer=None,
activity_regularizer=None,
embeddings_constraint=None,
mask_zero=False, input_length=None)
```

eg. [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]

few hyper parameters:
embed_dim : encodes the input sequence into a sequence of dense vectors of dimension embed_dim→ Turns positive integers (indexes) into dense vectors of fixed size. This layer can only be used as the first layer in a model.

```
model2.add(LSTM(lstm_out,kernel_regularizer=l2(0
.001), dropout_U=0.2,
```

lstm_out: transforms the vector sequence into a single vector of size lstm_out, containing information about the entire sequence.

```
model2.add(Dense(2,activation='sigmoid'))
```

## How much to update: Adam algorithm

- for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments;
- well suited for large data and/or parameters or for non-stationary objectives and problems with very noisy / sparse gradients;
- Adam’s update rule is its careful choice of stepsizes. the effective step taken in parameter space at timestep t $\Delta_t = \alpha \cdot \widehat{m}_t / \sqrt{\widehat{v}_t}$.

$$v_t = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2 \quad \Longrightarrow$$

$$\begin{aligned}\mathbb{E}[v_t] &= \mathbb{E}\left[(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \cdot g_i^2\right] \\ &= \mathbb{E}[g_t^2] \cdot (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} + \zeta \\ &= \mathbb{E}[g_t^2] \cdot (1-\beta_2^t) + \zeta\end{aligned}$$

- updates exponential moving averages of the gradient (mt) and the squared gradient (vt) where the hyper-parameters β1, β2 ∈ [0, 1) control their exponential decay rates; estimates of the mean and the uncentered variance of the gradient → these moving averages are initialized as (vectors of) 0’s, leading to moment estimates that are biased towards zero, especially during the initial timesteps, and especially when the decay rates are small (i.e. the βs are close to 1). The good news is that this initialization bias can be easily counteracted, resulting in bias-corrected estimates mt and vt.

Reference: *Diederik P. Kingma*, *Jimmy Ba*, “Adam: A Method for Stochastic Optimization”
Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015 https://arxiv.org/pdf/1412.6980v9.pdf

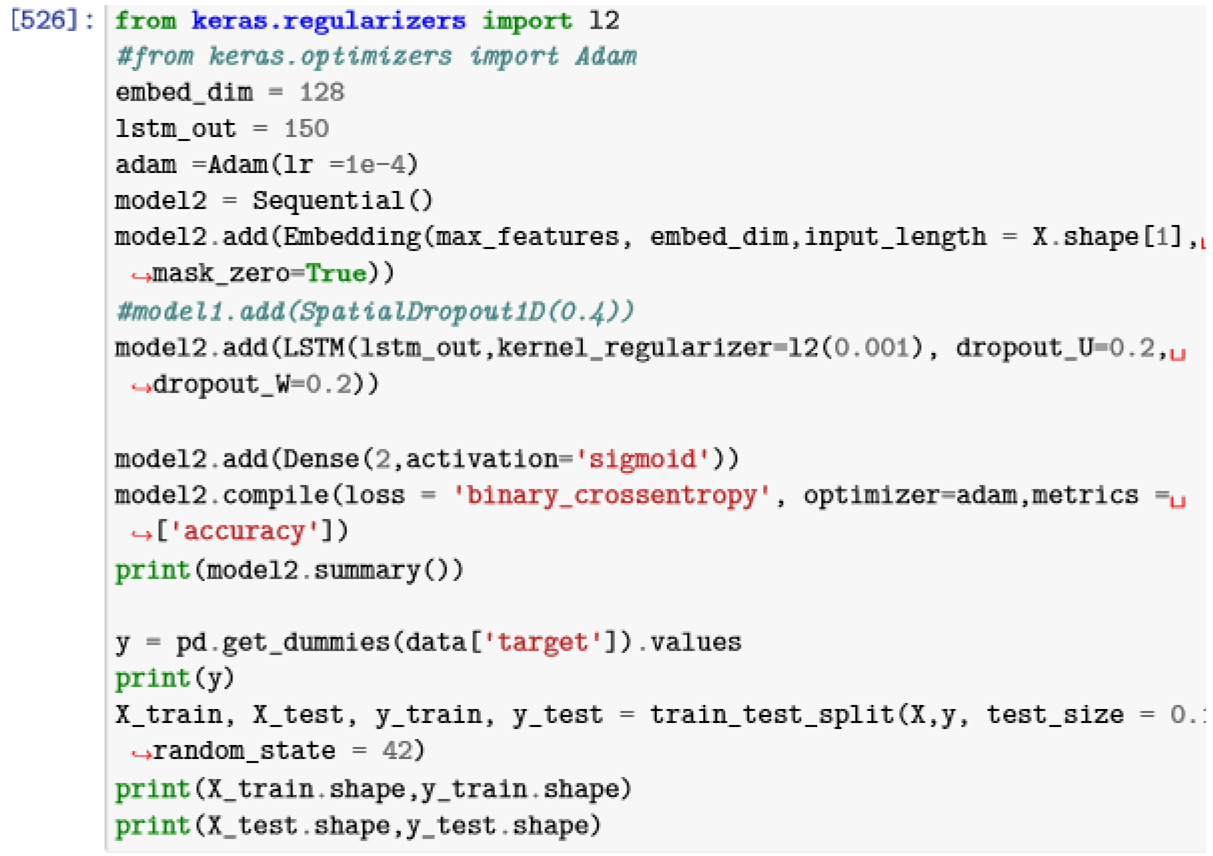
**Overfitting**: learning training data really well but fails to generalize the knowledge to the test data. Val loss increases while the train loss decreases. NN have too many weights and will overfit the data at the global minimum of L(W).
Remedy: early stopping rule or method for regularization

How often to update • batch learning: more efficient to process smaller batches at a time. **batch size** is the number of samples processed before the model is updated

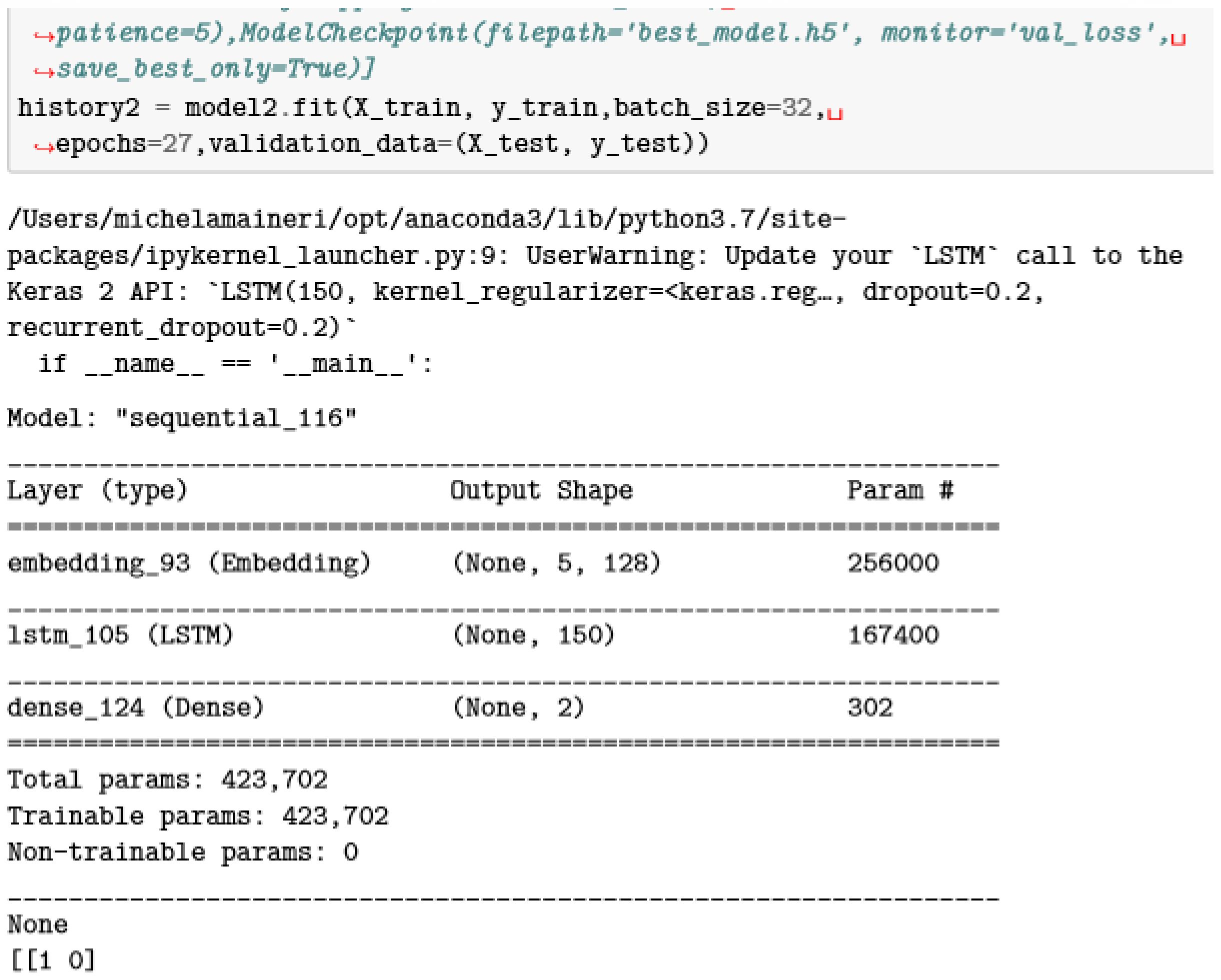
OUR MODEL

**Dropout**→ form of regularization that is performed when learning a network, typically at different rates at the different layers. The term dropout refers to dropping out units temporarily removing it from the network, nodes are randomly set to zero with probability ϕ, and inflate the remaining ones by a factor (1 − ϕ). ϕ is a tuning parameter.

**training epoch**: refers to one sweep through the entire training set • A for-loop over the number of epochs where each loop proceeds over the training dataset. Within this for-loop, is another nested for-loop that iterates over each batch of samples.

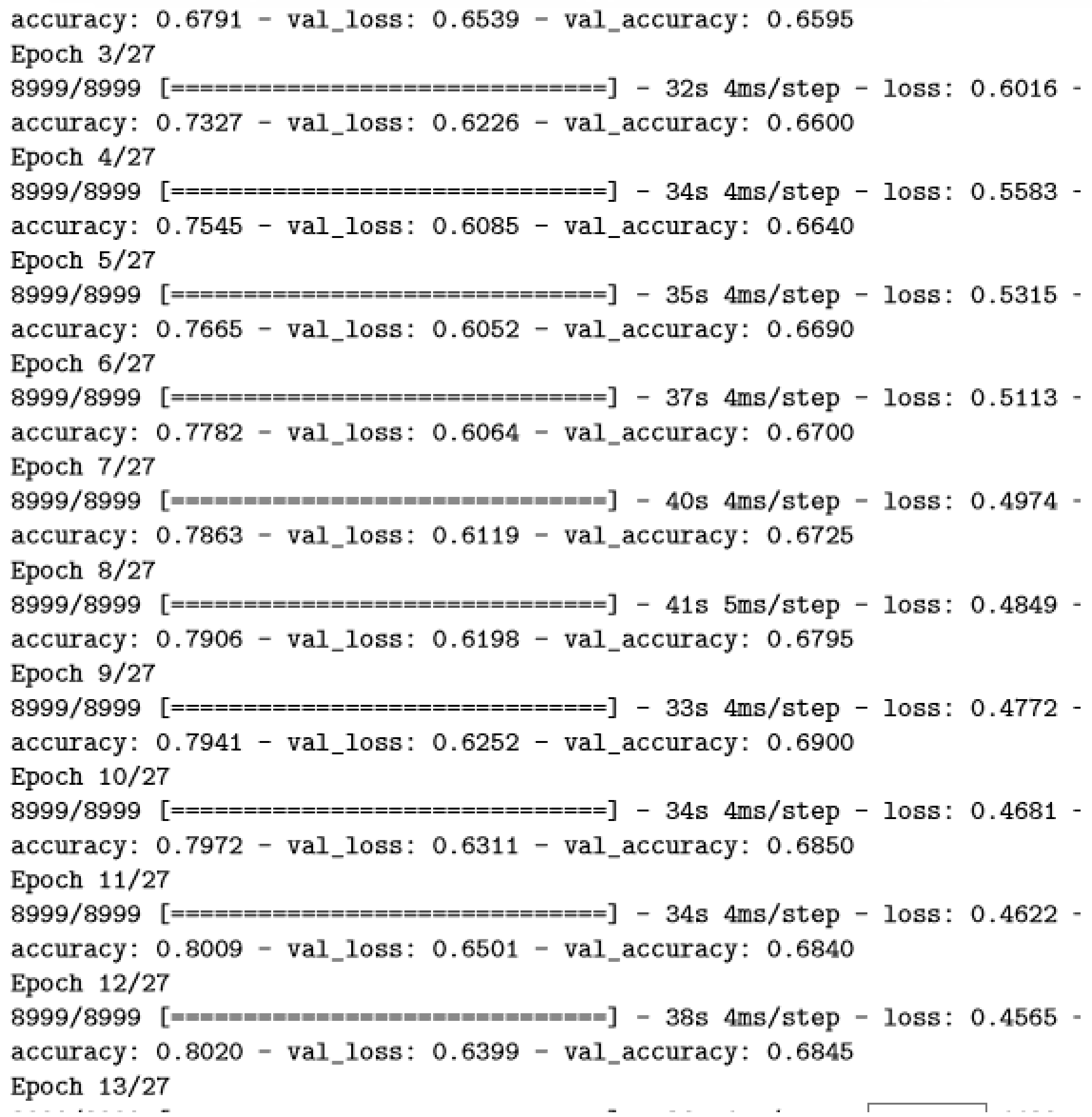


```
[526]: from keras.regularizers import l2
       #from keras.optimizers import Adam
       embed_dim = 128
       lstm_out = 150
       adam =Adam(lr =1e-4)
       model2 = Sequential()
       model2.add(Embedding(max_features, embed_dim,input_length = X.shape[1],␣
        ↪mask_zero=True))
       #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.
        ↪random_state = 42)
       print(X_train.shape,y_train.shape)
       print(X_test.shape,y_test.shape)
```

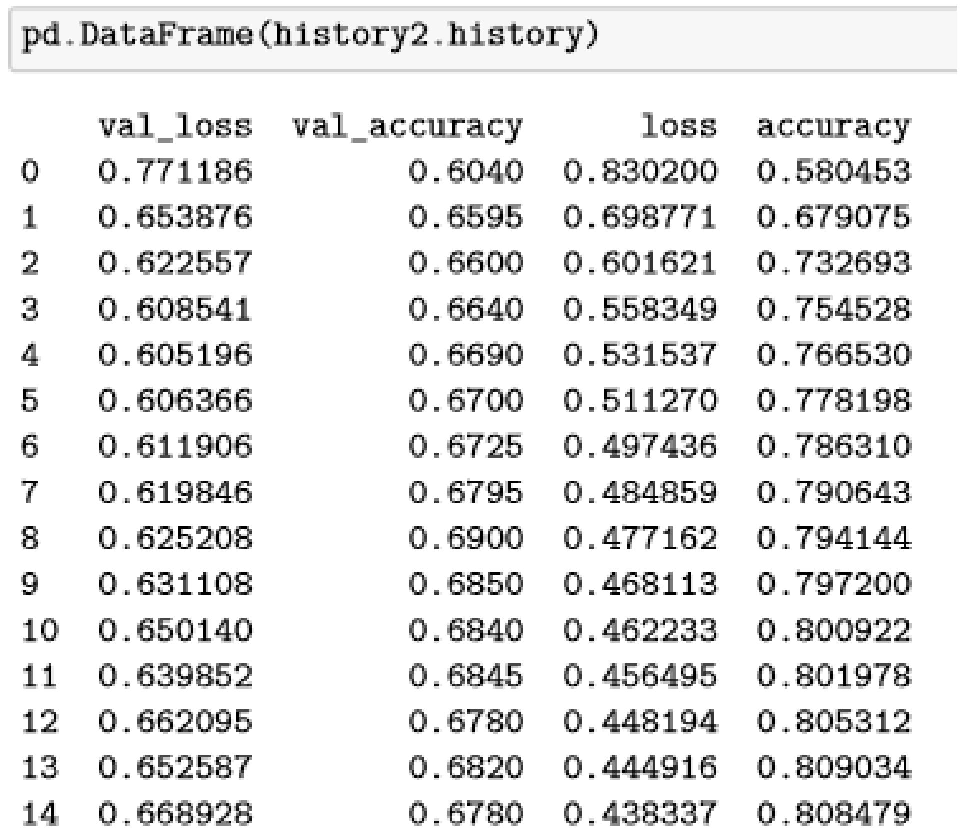


```
 ↪patience=5),ModelCheckpoint(filepath='best_model.h5', monitor='val_loss',␣
 ↪save_best_only=True)]
history2 = model2.fit(X_train, y_train,batch_size=32,␣
 ↪epochs=27,validation_data=(X_test, y_test))
```

```
/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]
```



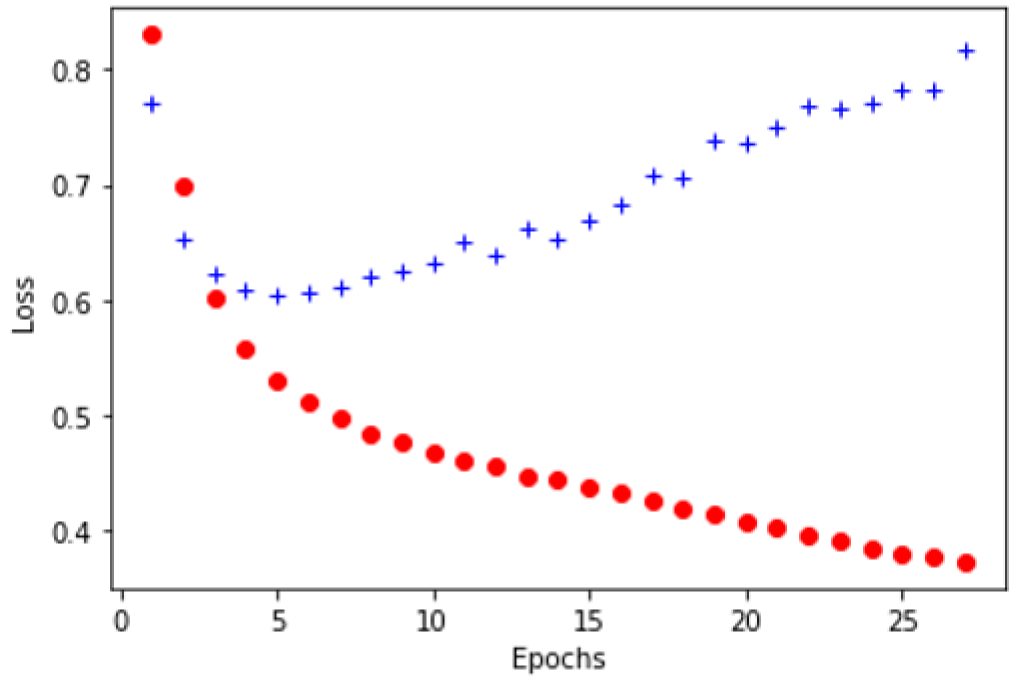
```
accuracy: 0.6791 - val_loss: 0.6539 - val_accuracy: 0.6595
Epoch 3/27
8999/8999 [==============================] - 32s 4ms/step - loss: 0.6016 -
accuracy: 0.7327 - val_loss: 0.6226 - val_accuracy: 0.6600
Epoch 4/27
8999/8999 [==============================] - 34s 4ms/step - loss: 0.5583 -
accuracy: 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
```



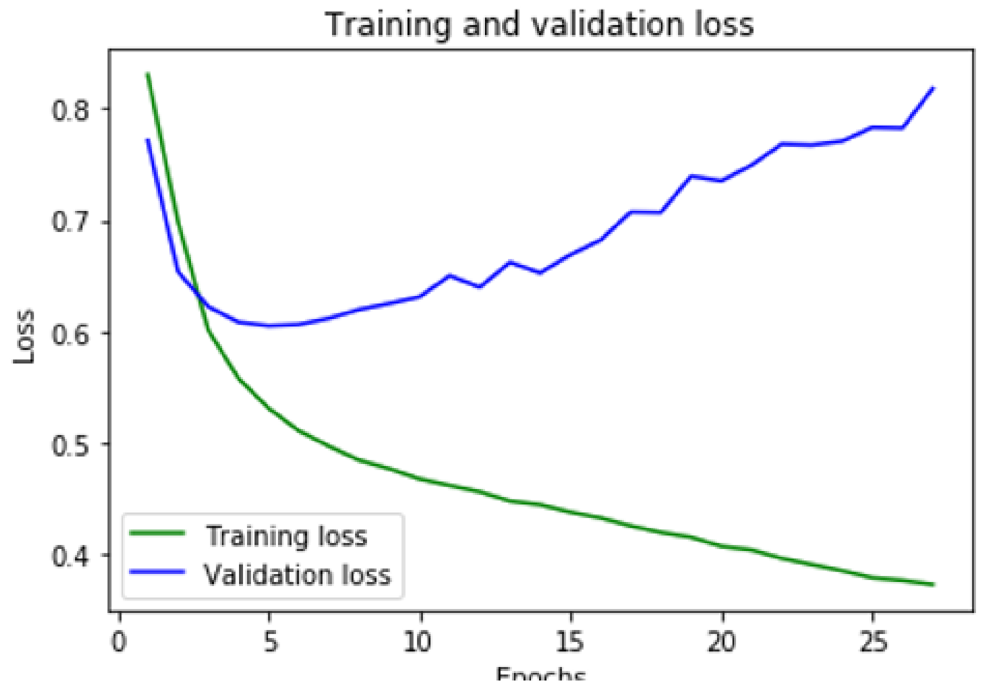
```
pd.DataFrame(history2.history)
```

```
    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
```

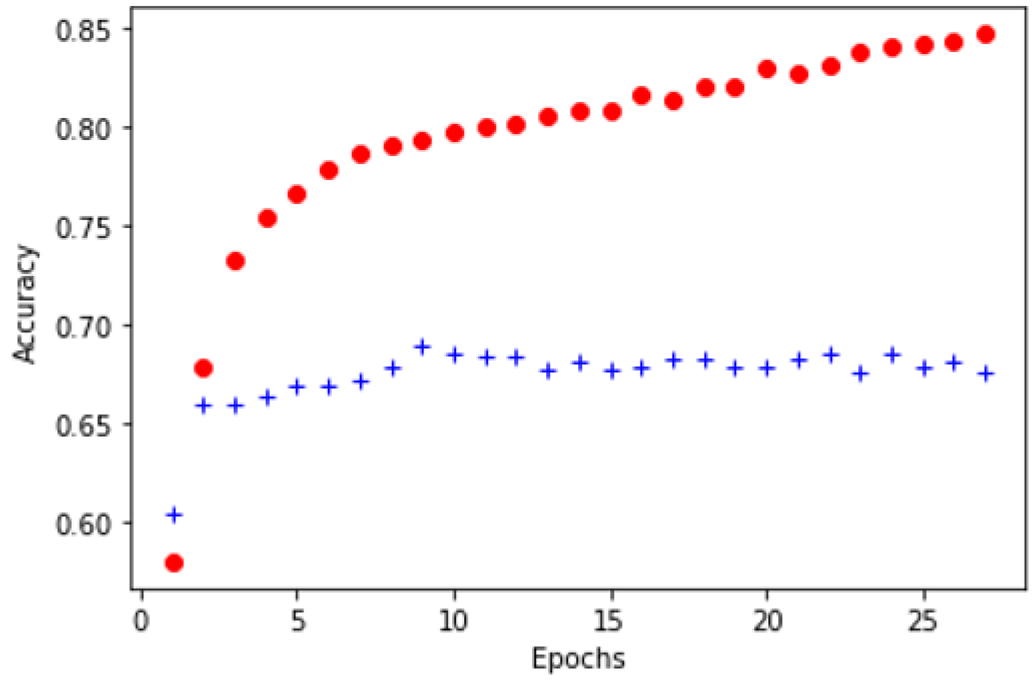
```
: from matplotlib import pyplot as plt
  history_dict=history2.history
  loss_values = history_dict['loss']
  val_loss_values = history_dict['val_loss']
  epochs = range(1, len(loss_values) + 1)
  plt.plot(epochs, loss_values, 'ro')
  plt.plot(epochs, val_loss_values, 'b+')
  plt.xlabel('Epochs')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

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

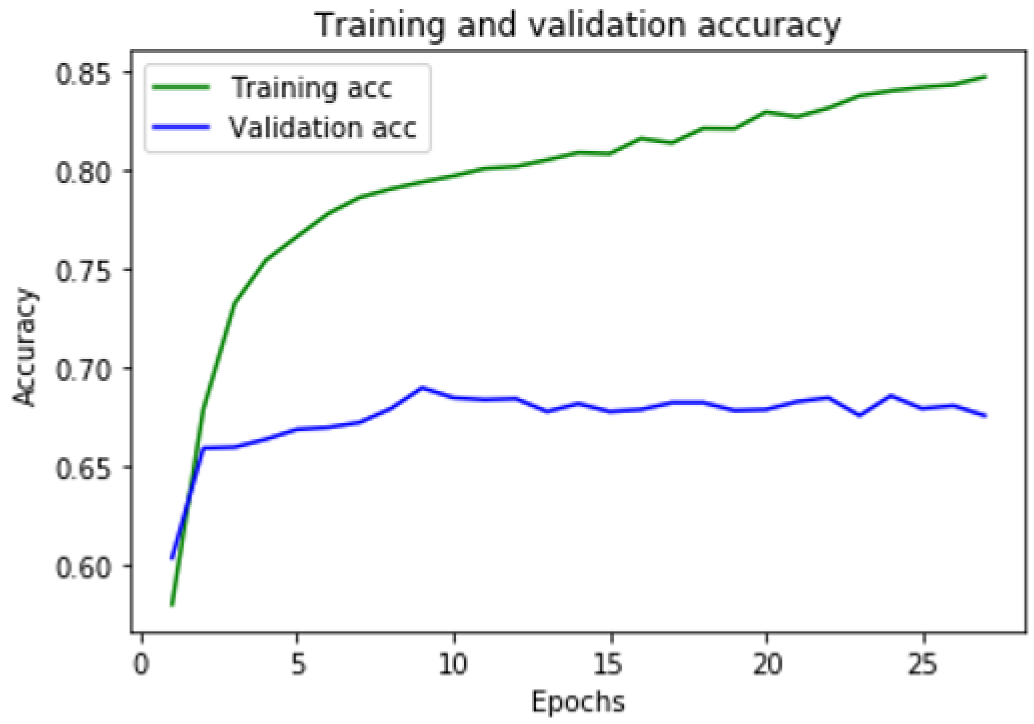


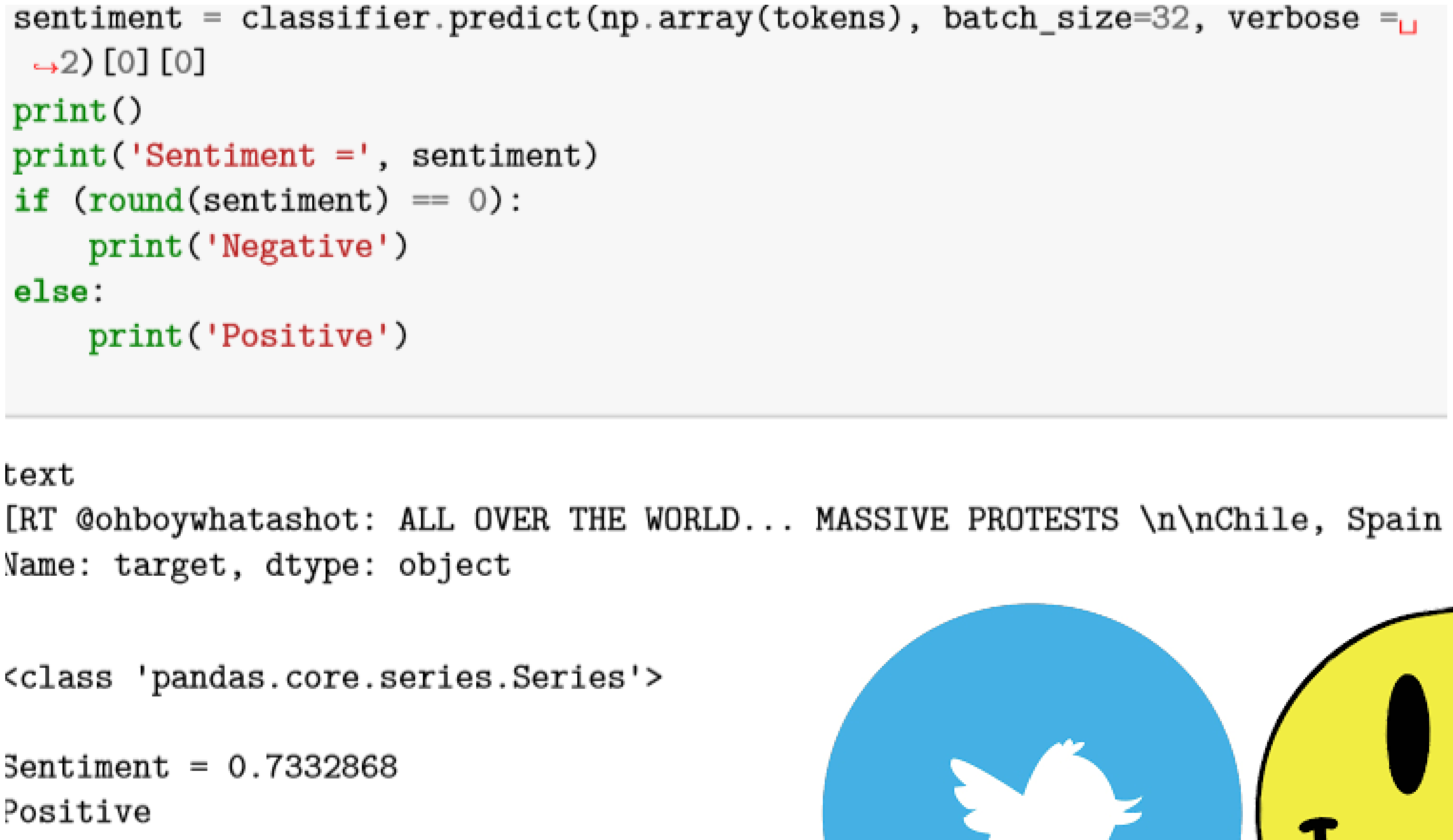
```
539]: plt.clf()
      loss = history2.history['loss']
      val_loss = history2.history['val_loss']
      epochs = range(1, len(loss) + 1)
      plt.plot(epochs, loss, 'g', label='Training loss')
      plt.plot(epochs, val_loss, 'b', label='Validation loss')
      plt.title('Training and validation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



```
40]: Text(0, 0.5, 'Accuracy')
```

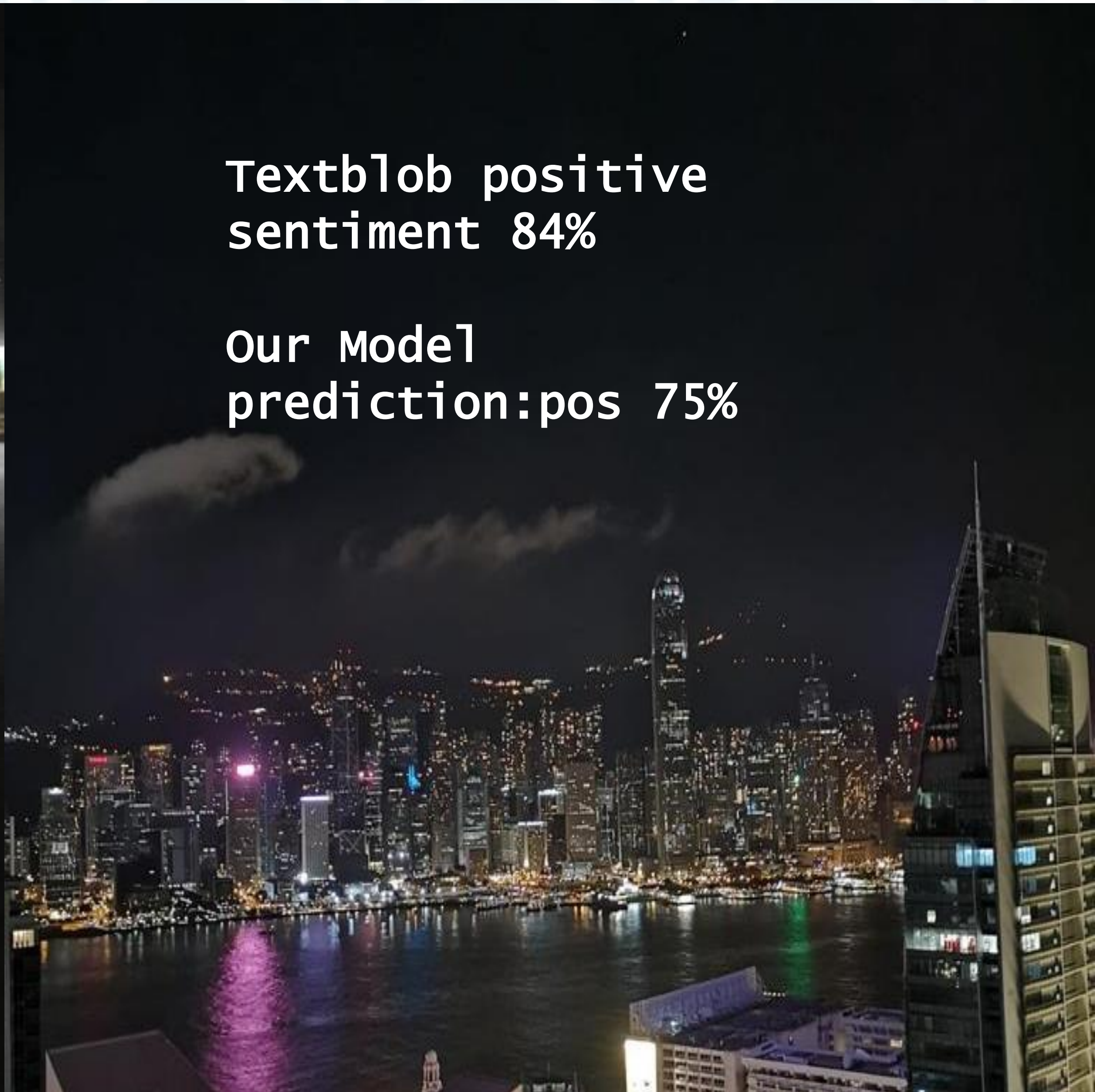






```
sentiment = classifier.predict(np.array(tokens), batch_size=32, verbose =␣
 ↪2)[0][0]
print()
print('Sentiment =', sentiment)
if (round(sentiment) == 0):
    print('Negative')
else:
    print('Positive')
```

```
text
[RT @ohboywhatashot: ALL OVER THE WORLD... MASSIVE PROTESTS \n\nChile, Spain
Name: target, dtype: object


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

Sentiment = 0.7332868
Positive
```



Textblob positive sentiment 84%

Our Model prediction:pos 75%