btctweets_sentiment_prediction-cnn-lstm-05312020

June 21, 2020

1 Bitcoin Tweets Sentiment Analysis Prediction – CNN-LSTM

[1]:

```
# Data analysis and wrangling
import pandas as pd
import numpy as np
import os
import string
import csv

# Visualization
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn
from wordcloud import WordCloud

# Sentiment prediction
import re
import nltk
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence

from tensorflow.keras import backend, models, layers
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Dense, Embedding, Conv1D, MaxPooling1D,␣
 ↪Dropout, LSTM
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
```

```
Using TensorFlow backend.
```

[2]:

```
# Upload the processed clean tweets

df = pd.read_csv('cleaned_tweet_data_05312020.csv',
                    header = 0,
                    error_bad_lines=False,
                    engine='python',
                    usecols=[2,5])

df.head()
```

```
[2]:                                           clean_text sentiment
     0  goldman sachs hosting client call bitcoin gold…   neutral
     1                 ok president trump endorsing tommy  positive
     2  bitcoin day nwhen btc first used commercial tr…  positive
     3  great icle nif way could avoided know like min…  positive
     4                       bitcoin btc current price gbp   neutral
```

```
[3]: df.tail()
```

```
[3]:                                              clean_text sentiment
     2015  xrp xrpcommunity btc bitcoin eth ltc vet oh ye…  positive
     2016  yes world global pandemic world going recessio…   neutral
     2017  last days grayscale bitcoin trust bought bitco…   neutral
     2018  pretty incredible imho nwe need creativity lik…  positive
     2019                               drop sachs nget sats   neutral
```

1.1 Exploratory Data Analysis

```
[4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2020 entries, 0 to 2019
Data columns (total 2 columns):
clean_text    2020 non-null object
sentiment     2020 non-null object
dtypes: object(2)
memory usage: 31.7+ KB
```

```
[5]: df.describe()
```

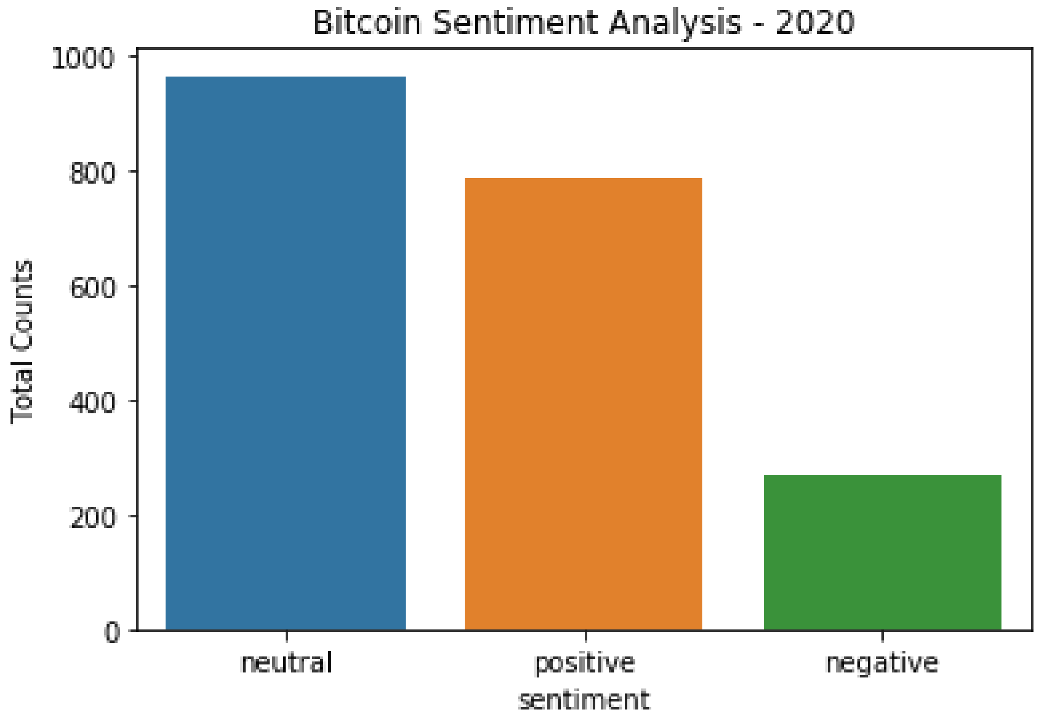
```
[5]:                    clean_text sentiment
     count                    2020      2020
     unique                   1391         3
     top     tick tock less blocks   neutral
     freq                       36       964
```

```
[6]: tweets_count = df['sentiment'].value_counts()
     tweets_count
```

```
[6]: neutral     964
     positive    788
     negative    268
     Name: sentiment, dtype: int64
```

```
[7]: # Plot sentiment mood count
     seaborn.countplot(x='sentiment', data=df)
     plt.title('Bitcoin Sentiment Analysis - 2020')
     plt.ylabel('Total Counts')
```

```
[7]: Text(0, 0.5, 'Total Counts')
```



```
[8]: # add text length column

     df['text_length'] = df['clean_text'].apply(len)
```

```
[9]: df.head()
```

```
[9]:                                           clean_text sentiment  text_length
     0  goldman sachs hosting client call bitcoin gold…   neutral           60
     1                  ok president trump endorsing tommy  positive           34
     2  bitcoin day nwhen btc first used commercial tr…  positive           87
     3  great icle nif way could avoided know like min…  positive           68
     4                       bitcoin btc current price gbp   neutral           29
```

```
[10]: df.tail()
```

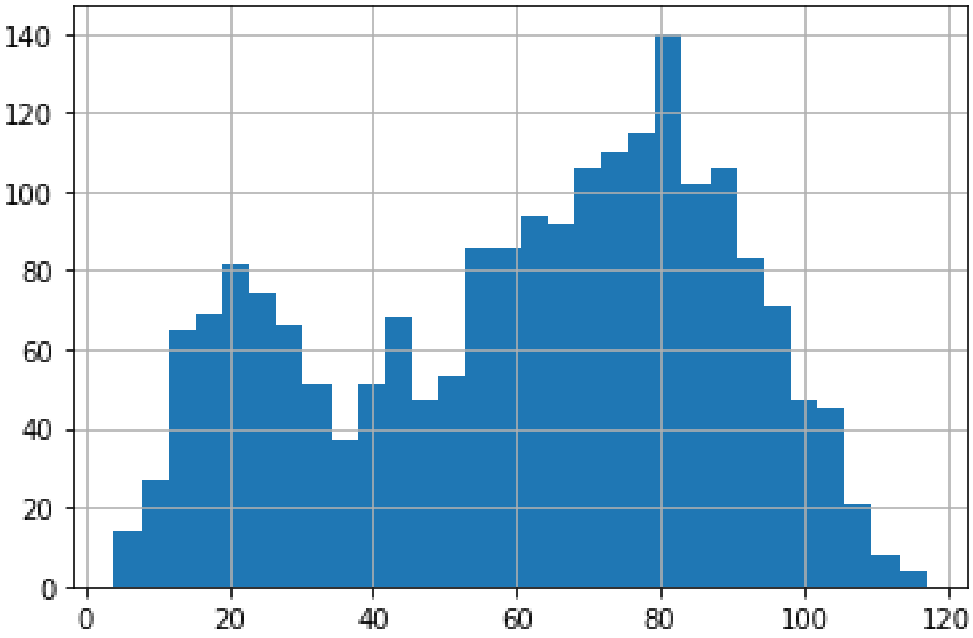
```
[10]:                                            clean_text sentiment  text_length
      2015  xrp xrpcommunity btc bitcoin eth ltc vet oh ye…  positive           96
      2016  yes world global pandemic world going recessio…   neutral           56
      2017  last days grayscale bitcoin trust bought bitco…   neutral           70
      2018  pretty incredible imho nwe need creativity lik…  positive           68
      2019                             drop sachs nget sats   neutral           20
```

```
[11]: df['text_length'].describe()
```

```
[11]: count    2020.000000
      mean       61.235149
      std        27.054467
      min         4.000000
      25%        39.000000
      50%        66.500000
      75%        83.000000
      max       117.000000
      Name: text_length, dtype: float64
```

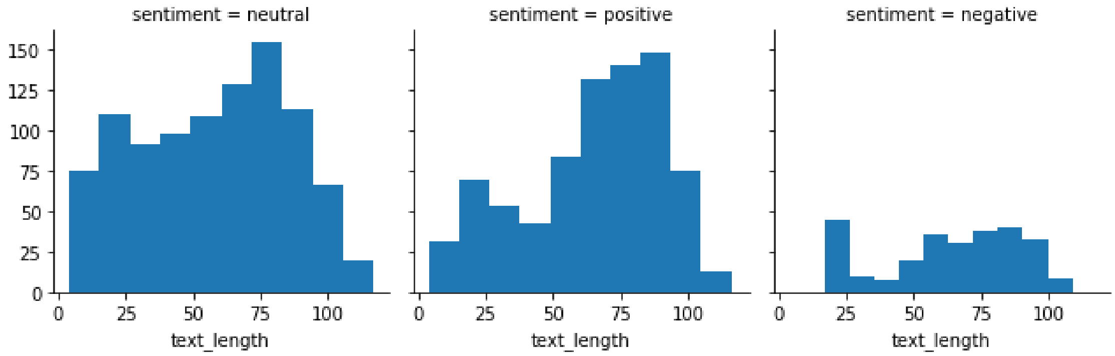
```
[12]: df['text_length'].hist(bins=30)
```

```
[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1d95e61ac08>
```

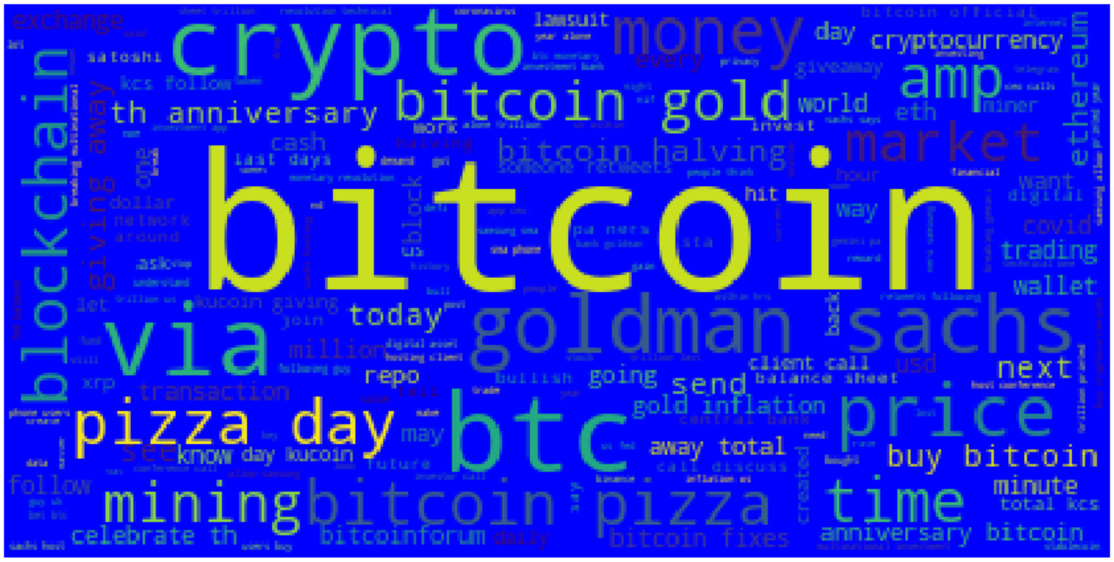


```
[13]: plot = seaborn.FacetGrid(df,col='sentiment')
      plot.map(plt.hist,'text_length')
```

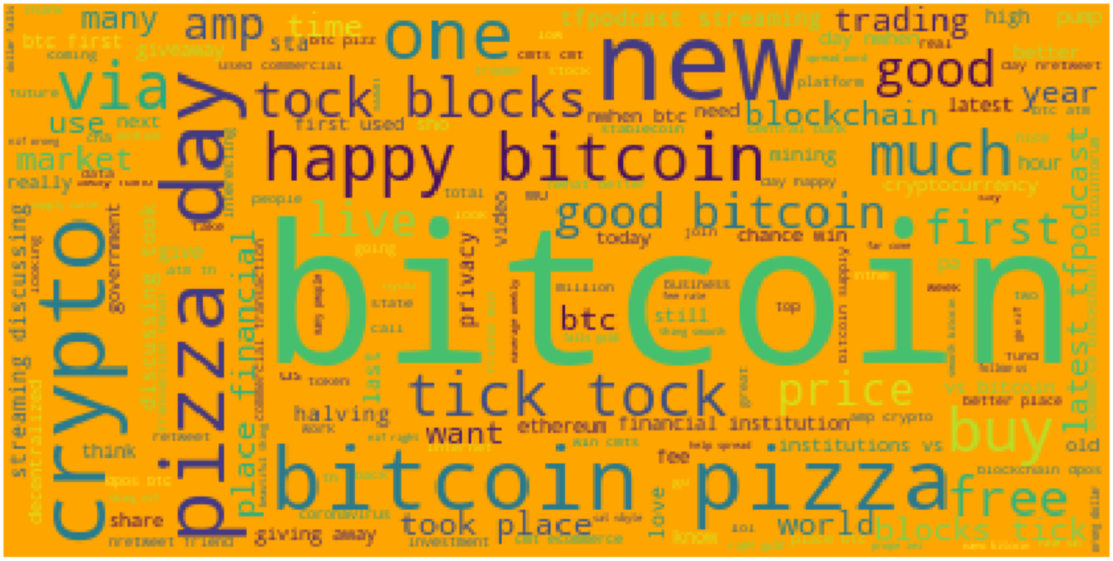
```
[13]: <seaborn.axisgrid.FacetGrid at 0x1d95f9a6748>
```



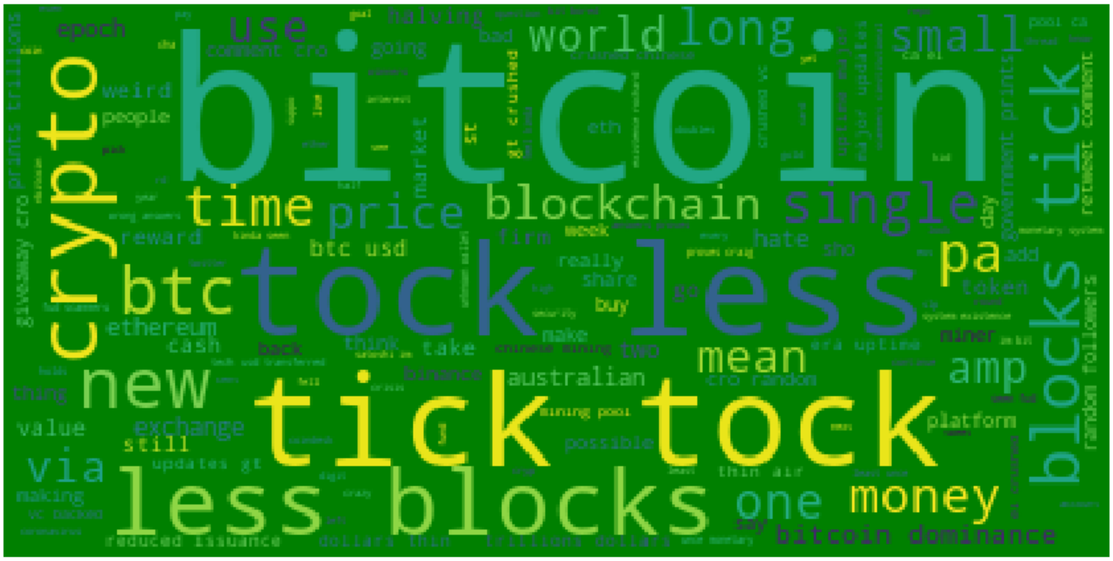
```
[14]: # visualization using wordcloud for the Neutral tweets

      tweets_neutral = df[df['sentiment']=='neutral']
      words = ' '.join(tweets_neutral['clean_text'])

      wordcloud = WordCloud(background_color='blue').generate(words)

      plt.figure(1,figsize=(15, 12))
      plt.imshow(wordcloud)
      plt.axis('off')
      plt.show()
```



```
[15]: # visualization using wordcloud for the Positive tweets

      tweets_positive = df[df['sentiment']=='positive']
      words = ' '.join(tweets_positive['clean_text'])

      wordcloud = WordCloud(background_color='orange').generate(words)

      plt.figure(1,figsize=(15, 12))
      plt.imshow(wordcloud)
      plt.axis('off')
      plt.show()
```



```
[16]: # visualization using wordcloud for the Negative tweets

      tweets_negative = df[df['sentiment']=='negative']
      words = ' '.join(tweets_negative['clean_text'])

      wordcloud = WordCloud(background_color='green').generate(words)

      plt.figure(1,figsize=(15, 12))
      plt.imshow(wordcloud)
      plt.axis('off')
      plt.show()
```



1.2 Training and Testing sets

[17]:

```
# Encode Categorical Variable for Deep Learning
# Apply get_dummies

x = df['clean_text']
y = pd.get_dummies(df['sentiment']).values
num_classes = df['sentiment'].nunique()
```

[18]:

```
# fix random seed for reproducibility
np.random.seed(42)

# Split into 80% train and 20% test sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.20,
                                                    stratify=y,
                                                    random_state=42)

print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(1616,) (404,) (1616, 3) (404, 3)
```

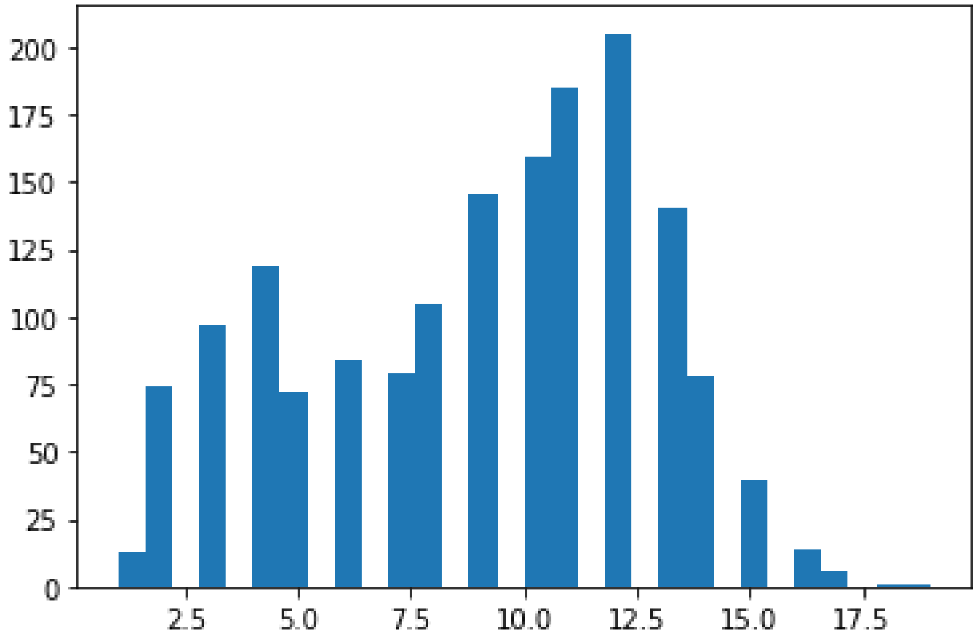
1.2.1 Tokenize Text

[19]:

```
# Tokenize Text

# Define max number of words in dictionary
max_features = 20000
```

```
tokenizer = Tokenizer(num_words=max_features, lower=True)
tokenizer.fit_on_texts(list(x_train))

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

[20]:
```
totalNumWords = [len(feature) for feature in x_train]
plt.hist(totalNumWords,bins = 30)
plt.show()
```



[21]:
```
# Based on the plot histogram, setting feature maxWords
maxWords = 20

x_train = sequence.pad_sequences(x_train, maxlen=maxWords)
x_test = sequence.pad_sequences(x_test, maxlen=maxWords)

print(x_train.shape,x_test.shape)
```

```
(1616, 20) (404, 20)
```

1.3 Model – CNN Long Short-Term Memory Network (CNN LSTM)

[22]:
```
# Build a CNN-LSTM Model

backend.clear_session()
```

```
embedding_size = 300
epoch = 25
batch_size = 64

model = models.Sequential()
model.add(Embedding(max_features, embedding_size, input_length=x_train.
 ↪shape[1]))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(96, dropout=0.15, recurrent_dropout=0.15))

model.add(Dense(num_classes * 4, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes * 2, activation='relu'))
model.add(Dropout(0.2))

# Output layer
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer = 'adam',
               loss = 'categorical_crossentropy',
               metrics = ['accuracy'])
print(model.summary())


# Train the model
history = model.fit(x_train,
                    y_train,
                    epochs = epoch,
                    batch_size = batch_size,
                    validation_data=(x_test, y_test),
                    verbose = 1)


history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
epochs = range(1, len(history_dict['acc']) + 1)

hist = pd.DataFrame(history.history)
print(hist.head())
```

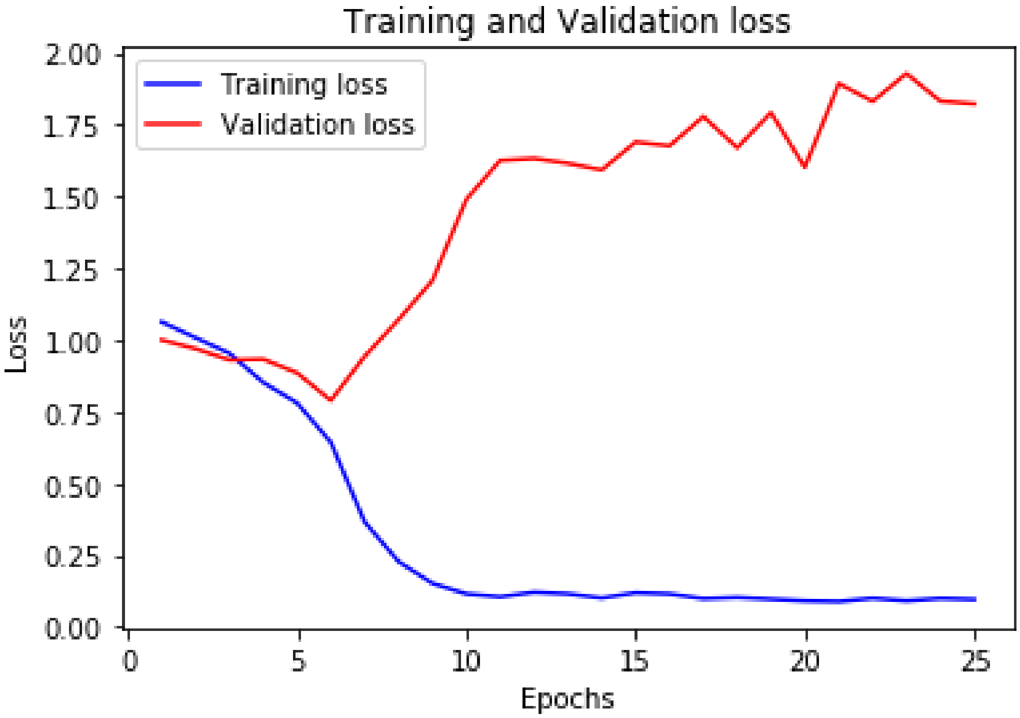
```
plt.plot(epochs, loss_values, 'b', label = 'Training loss')
plt.plot(epochs, val_loss_values, 'r', label = 'Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(epochs, acc_values, 'b', label = 'Training accuracy')
plt.plot(epochs, val_acc_values, 'r', label = 'Validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

results = model.evaluate(x_test, y_test)
print(model.metrics_names)
print(results)
```

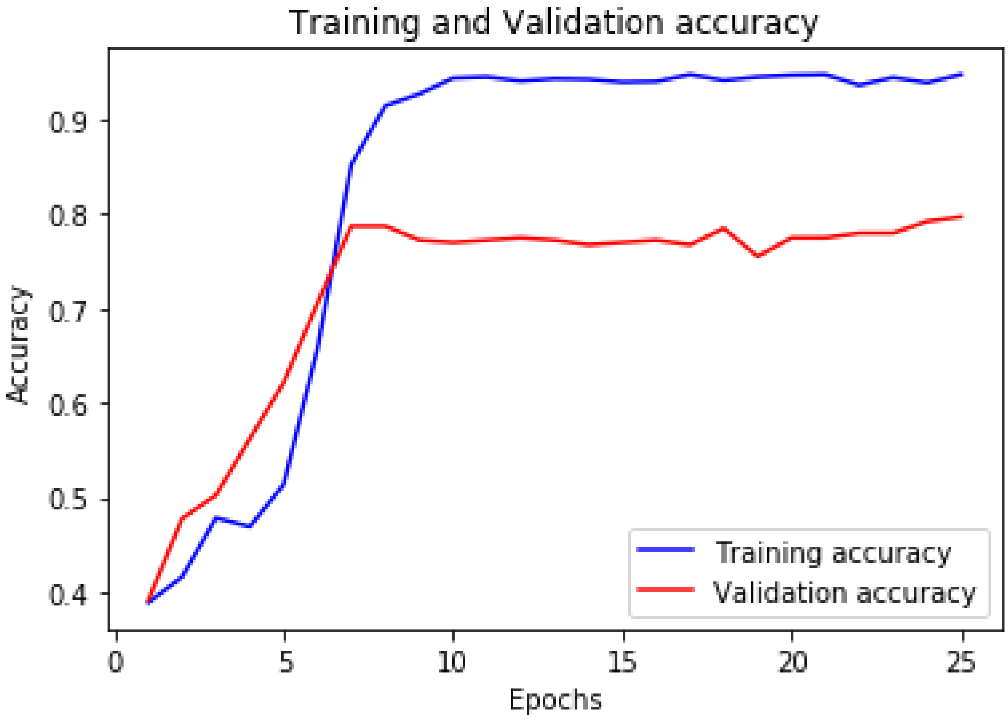
```
WARNING:tensorflow:From C:\cuong\lib\site-
packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\cuong\lib\site-
packages\tensorflow\python\keras\backend.py:4010: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 20, 300)           6000000
_________________________________________________________________
conv1d (Conv1D)              (None, 20, 32)            28832
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 10, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 10, 32)            3104
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 5, 32)             0
_________________________________________________________________
lstm (LSTM)                  (None, 96)                49536
```

```
_________________________________________________________________
dense (Dense)                (None, 12)                1164
_________________________________________________________________
dropout (Dropout)            (None, 12)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 78
_________________________________________________________________
dropout_1 (Dropout)          (None, 6)                 0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 21
=================================================================
Total params: 6,082,735
Trainable params: 6,082,735
Non-trainable params: 0
_________________________________________________________________
None
Train on 1616 samples, validate on 404 samples
WARNING:tensorflow:From C:\cuong\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/25
1616/1616 [==============================] - 9s 6ms/sample - loss: 1.0634 - acc:
0.3886 - val_loss: 1.0004 - val_acc: 0.3911
Epoch 2/25
1616/1616 [==============================] - 4s 3ms/sample - loss: 1.0085 - acc:
0.4158 - val_loss: 0.9701 - val_acc: 0.4777
Epoch 3/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.9540 - acc:
0.4783 - val_loss: 0.9305 - val_acc: 0.5025
Epoch 4/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.8522 - acc:
0.4691 - val_loss: 0.9334 - val_acc: 0.5619
Epoch 5/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.7807 - acc:
0.5136 - val_loss: 0.8856 - val_acc: 0.6213
Epoch 6/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.6434 - acc:
0.6572 - val_loss: 0.7892 - val_acc: 0.7054
Epoch 7/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.3651 - acc:
0.8521 - val_loss: 0.9438 - val_acc: 0.7871
Epoch 8/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.2274 - acc:
0.9146 - val_loss: 1.0719 - val_acc: 0.7871
Epoch 9/25
```

```
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.1515 - acc:
0.9270 - val_loss: 1.2080 - val_acc: 0.7723
Epoch 10/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1150 - acc:
0.9437 - val_loss: 1.4904 - val_acc: 0.7698
Epoch 11/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.1047 - acc:
0.9449 - val_loss: 1.6264 - val_acc: 0.7723
Epoch 12/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1206 - acc:
0.9406 - val_loss: 1.6331 - val_acc: 0.7748
Epoch 13/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1150 - acc:
0.9431 - val_loss: 1.6162 - val_acc: 0.7723
Epoch 14/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1006 - acc:
0.9425 - val_loss: 1.5931 - val_acc: 0.7673
Epoch 15/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1187 - acc:
0.9394 - val_loss: 1.6905 - val_acc: 0.7698
Epoch 16/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.1147 - acc:
0.9400 - val_loss: 1.6773 - val_acc: 0.7723
Epoch 17/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.0984 - acc:
0.9474 - val_loss: 1.7794 - val_acc: 0.7673
Epoch 18/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.1026 - acc:
0.9412 - val_loss: 1.6707 - val_acc: 0.7847
Epoch 19/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.0966 - acc:
0.9449 - val_loss: 1.7929 - val_acc: 0.7550
Epoch 20/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.0906 - acc:
0.9468 - val_loss: 1.6023 - val_acc: 0.7748
Epoch 21/25
1616/1616 [==============================] - 4s 2ms/sample - loss: 0.0882 - acc:
0.9474 - val_loss: 1.8945 - val_acc: 0.7748
Epoch 22/25
1616/1616 [==============================] - 4s 3ms/sample - loss: 0.0999 - acc:
0.9363 - val_loss: 1.8327 - val_acc: 0.7797
Epoch 23/25
1616/1616 [==============================] - 4s 3ms/sample - loss: 0.0907 - acc:
0.9443 - val_loss: 1.9289 - val_acc: 0.7797
Epoch 24/25
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.0992 - acc:
0.9387 - val_loss: 1.8335 - val_acc: 0.7921
Epoch 25/25
```

```
1616/1616 [==============================] - 3s 2ms/sample - loss: 0.0959 - acc:
0.9474 - val_loss: 1.8239 - val_acc: 0.7970
       loss       acc  val_loss   val_acc
0  1.063406  0.388614  1.000369  0.391089
1  1.008457  0.415842  0.970121  0.477723
2  0.953958  0.478342  0.930459  0.502475
3  0.852168  0.469059  0.933353  0.561881
4  0.780665  0.513614  0.885603  0.621287
```





```
404/404 [==============================] - 0s 128us/sample - loss: 1.8239 - acc:
0.7970
['loss', 'acc']
[1.8238952685110639, 0.7970297]
```

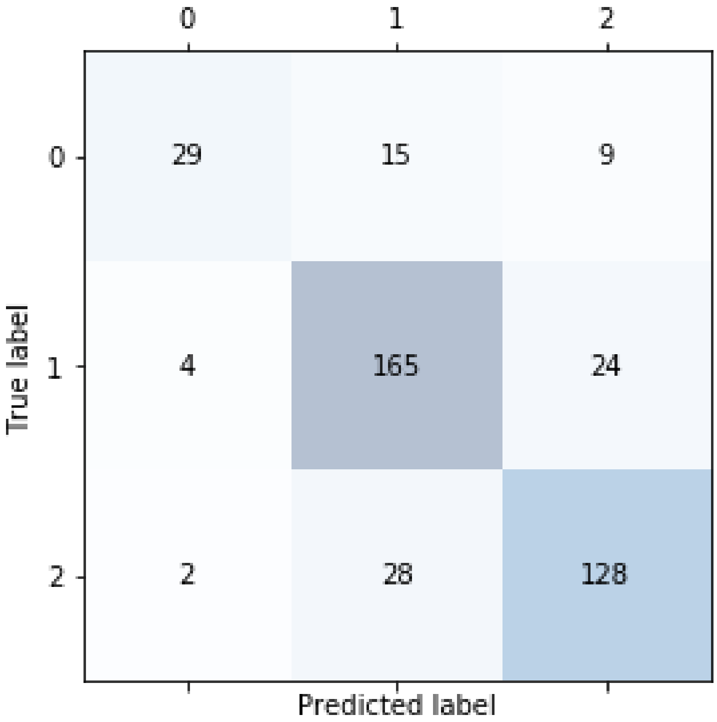
1.4 Model Prediction/Evaluation

[23]:

```
# Reference link: https://stackoverflow.com/questions/44189119/
↪how-to-plot-confusion-matrix-correctly

# Evaluate model with Test set

def model_evaluation():
    # predict class with test set
    y_pred_test =  model.predict_classes(x_test, batch_size=batch_size,␣
↪verbose=0)
    print('Accuracy:\t{:0.1f}%'.format(accuracy_score(np.
↪argmax(y_test,axis=1),y_pred_test)*100))

    #classification report
    print('\n')
    print(classification_report(np.argmax(y_test,axis=1), y_pred_test))
    #confusion matrix
    confmatrix = confusion_matrix(np.argmax(y_test,axis=1), y_pred_test)

    fig, ax = plt.subplots(figsize=(4, 4))
    ax.matshow(confmatrix, cmap=plt.cm.Blues, alpha=0.3)
    for i in range(confmatrix.shape[0]):
        for j in range(confmatrix.shape[1]):
            ax.text(x=j, y=i, s=confmatrix[i, j], va='center', ha='center')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.tight_layout()
```

```
[24]: print(model_evaluation())
```

```
Accuracy:       79.7%


              precision    recall  f1-score   support

           0       0.83      0.55      0.66        53
           1       0.79      0.85      0.82       193
           2       0.80      0.81      0.80       158

    accuracy                           0.80       404
   macro avg       0.81      0.74      0.76       404
weighted avg       0.80      0.80      0.79       404

None
```


0
1
2
0
1
2
True label
Predicted label
29
15
9
4
165
24
2
28
128