DEEP LEARNING

(CS6005)

MINI PROJECT

On: Natural Language Processing (NLP)

EMOTION DETECTION FROM TEXT

-Aparna S S
2018103008
CSE – ‘P’ batch

Date- 21/05/2021

Problem Description:

We all as humans have emotions and predicting these emotions is not an easy task for fellow beings itself. In such a case predicting one's emotion by the machine is a very difficult task. But difficulty is not a phrase for neural networks that too when deep learning is concerned. We have a special set of processing for classifying text as text classification is more difficult than from images (since images have pixel size but text have only word length). This set is called as Natural Language Processing. There are different techniques for text classification using NLP, one of which is word2vector. In this method, the words are mapped with their corresponding vectors (formed by converting the words to corresponding integrals based on the word_index) to create the embedding matrix which is then fed to the convolution layer which can be CNN model, LSTM or Bert model etc. In this project, we have not given the input as text rather speech which is converted to speech. This makes the user's work much easier as we all find speaking more comfortable than typing. This speech is converted to text using Google API. This converted text is further subjected to NLP technique and the technique used here is wod2vector. Once the embedding matrix is fed to the CNN layer, the emotion is predicted with this trained model.

Dataset details:

The dataset used for word embedding is **wiki-news-300d-1M.vec** which is a pre-trained word vector dataset with 1million word vectors trained on Wikipedia 2017. The first line of the file contains the number of words in the vocabulary and the size of the vectors. Each line contains a word followed by its vectors, like in the default fastText text format. Each value is space separated. Words are ordered by descending frequency.

The dataset used for pre-processing training is a manually created dataset from various sources having 7936 records as train data and 3394 records as the test data with 5 emotions being anger, joy, neutral, sad and fear. The two data are matched to get the embedding matrix which is fed to the convolution for prediction.

url- https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip

Modules:

1. Converting speech to text

- ✓ The voice for which the emotion has to be predicted is given to the microphone of the device which can be a laptop, mobile phone, desktop, tablet etc. The voice is recognized by the device using the **speech_recognition** module of Pyaudio.
- ✓ With this recognized speech as the source, the system listens to the voice by adjusting ambient noises which can be any sort of noise other than human voice.
- ✓ This detected voice is then converted to text using **Google API** with the conversion language as **American English**. The converted text is then displayed.
- ✓ In case, the voice is not provided when asked to or the system is not able to understand the voice then **UnknownValueError** is thrown.
- ✓ In case, there are connectivity issues or problems with the server, then **RequestError** is thrown.

2. Collecting data

- ✓ The dataset is collected from various sources having 7936 records as train data and 3394 records as the test data with 5 emotions being anger, joy, neutral, sad and fear.
- ✓ These datasets are loaded and the feature(text) and target(emotion) are stored in the train and test part of the data.
- ✓ The emotion count of the various emotions in the created dataset are-

```
joy        2326
sadness    2317
anger      2259
neutral    2254
fear       2171
Name: Emotion, dtype: int64
```

3. Pre-processing

This is the vital and the most crucial part of text processing. This includes the basic NLP techniques of tokenization and padding.

- ✓ Remove the hashtags, username and other inconsistent data so that the accuracy is better. This data is tokenized using **word_tokenize** which returns the tokenized copy of text using NLTK's recommended word tokenizer. Tokenization refers to the splitting of a large text to smaller one i.e. a paragraph to sentence or a sentence to words.

- ✓ **fit_on_texts** updates internal vocabulary based on a list of texts. This method creates the vocabulary index based on word frequency.

  texts_to_sequences transform each text in texts to a sequence of integers. So, it basically takes each word in the text and replaces it with its corresponding integer value from the word_index dictionary.

  The vocab_size is the length of the word_index +1 since 0 is reserved for padding.

- ✓ Since these converted integrals must be of the same length for passing to the convolution, the data is **padded** and the maximum length is **500**.

4. Word embedding

- ✓ Word Embedding is a representation of text where words that have the similar meaning have a similar representation. We will use 300dimensional word vectors pre-trained on Wikipedia articles. We can also train the w2v model with our data, however our dataset is quite small and trained word vectors might not be as good as using pretrained w2v.
- ✓ As all the words in our dataset may be not present in the pretrained w2v we create vectors for the words which are not present in the pretrained dataset and map them to the corresponding words. This creates the **embedding**

matrix. The new words which are not present in the pretrained dataset are displayed.

- ✓ With this embedding matrix as the weight create the embed_layer which is fed as the input to the convolution layer.

5. Convolution and Performance metrics

- ✓ Create the **1D convolution sequential model** with the embed_layer and the final activation of the dense layer as the **softmax** activation.
- ✓ Trian the model with a **batch size** of **256** and for **6 epochs**.
- ✓ Calculate the performance metrics namely accuracy, F1 score and confusion matrix.

6. Prediction

- ✓ With the trained model, predict the emotion of the converted text from the Google API.

Code and results:

Converting speech to text:

1. The device's microphone is enabled to hear the voice for which the emotion has to be detected.

```
import speech_recognition as sr

r = sr.Recognizer()

speech = sr.Microphone(device_index=0)
```

2. The system listens to the voice by adjusting ambient noises and stores this as the audio.

```
with speech as source:
    print("say something!...")
    audio = r.adjust_for_ambient_noise(source)
    audio = r.listen(source)
```

3. The detected voice is converted into text (American English) using the **google API** and is stored as the message.

```
try:
    recog = r.recognize_google(audio, language = 'en-US')
    print("You said: " + recog)
    message = [recog]
```

4. In case, the voice is not provided when asked to or the system is not able to understand the voice then **UnknownValueError** is thrown.

```
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
```

5. In case, there are connectivity issues or problems with the server, then **RequestError** is thrown.

```
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

Output:

```
say something!...
You said: weather was good
```

Collecting data:

1. The dataset is collected from various sources having 7936 records as train data and 3394 records as the test data with 5 emotions being anger, joy, neutral, sad and fear.

   The glimpse of the dataset is:

Train data-

Emotion	Text
neutral	There are tons of other paintings that I think are better .
sadness	Yet the dog had grown old and less capable , and one day the gillie had come and explained with great sorrow that the dog had suffered a stroke , and must be put down .
fear	When I get into the tube or the train without paying for the ticket.
fear	This last may be a source of considerable disquiet and one might not at first see how such obviously ` immoral " content could be defended as part of a system of morality .
anger	She disliked the intimacy he showed towards some of them , was resentful of the memories they shared of which she was not a part , and felt excluded .
sadness	When my family heard that my Mother's cousin who lives in England wrote us to tell that he had cancer of the lymph glands.
joy	Finding out I am chosen to collect norms for Chinese aphasia (I will contribute to China's catching up with the West in neuropsychology).
anger	A spokesperson said : ` Glen is furious that the new ` Anarchy " promo features footage of Sid Vicious as well as himself .
neutral	Yes .
sadness	When I see people with burns I feel sad, actually I can not even express my feelings as I think that they must suffer a lot.
fear	I had gone to the hospital for my research and got late in reaching home. I feared that when I reached home there would be a quarrel because of my being late.
sadness	One day I heard from a friend that the boy I loved had gone out with her and not with me.
joy	Connor 's voice was gleeful .
anger	Very funny . What's wrong with you today ? You are my secretary and you are not supposed to talk to me in that tone of voice . Do you know that ?
neutral	Perhaps we need to have more babies ! Tina gave birth to a baby boy yesterday .
neutral	Why ?
joy	` I am very happy with the way I am playing but I accept that he believes he should be England 's fly-half .
fear	When I have to leave the baby in the carriage and go shopping, I fear that something will happen to the baby.
anger	No , you used to be . But not now . You are in love with someone else . You are in love with my friend , Janet . You appreciate her very much . You think she is beautiful and fun to be with and you think I â€™ m dumb and uninteresting .
anger	Wow ! Why so much ? I thought they were getting you an assistant .
fear	Infiltration in our lives. The illusion of so we accept their political Solutions. #Tyranny
joy	Spontaneous picnic with sister and friend and children - great.
sadness	Upon moving away from home for the first time to a different city, I felt no longer \under the wing\" of my parents even though they would always be there to help me. I felt a sense of loss, I was now all alone in the world, responsible for my ov
neutral	Over there .
sadness	When I learnt that I had failed an exam, This not only influenced my emotions but also other important tasks, it also made my record imperfect.
neutral	Me , too , I can â€™ t stop scratching . They are everywhere ! Sneaky little jerks .

Test data-

Emotion	Text
sadness	I experienced this emotion when my grandfather passed away.
neutral	when I first moved in , I walked everywhere . But within a week , I had my purse stolen â€" just a block away from the police station ! Now , I always take public transportation .
anger	` Oh ! " she bleated , her voice high and rather indignant .
fear	However , does the right hon. Gentleman recognise that profound disquiet has been expressed about some of the proposals in the Bill , particularly the fast-track ones ?
sadness	My boyfriend didn't turn up after promising that he was coming.
neutral	It's freezing .
sadness	That â€™ s not all ! I also had to finish writing a sales report for my boss . In the end , I finished everything . I wonder what will be waiting for me tomorrow morning .
anger	I don't have a warrant .
neutral	I guess so .
sadness	I was just robbed !
neutral	This is it ?
neutral	Well , do you think I have to ?
fear	They seemed anxious and hesitant about leaving , as if uncertain of which direction to take .
anger	I experienced anger most recently when I had committed a sin which I had gone a week and a half without doing. I had made a vow to God and had blown it. Now I had to start all over.
neutral	cars might be convenient , but they're so bad for the environment .
sadness	When a close friend of mine who was with me in first year could not make it to the second year.
sadness	No . Only a few pounds . But my passport was in the bag . That â€™ what I â€™ m really worry about .
neutral	For what ?
joy	Her headache seemed to be getting worse and worse , and Raffaella 's command of English slipped as she grew increasingly excited .
fear	A leader is a coach, not a judge. #lean #systemsthinking #development #HR
joy	I was very happy when my scholarship to continue studying at university was approved after it had been cancelled.
fear	Being alone in Europe and having to catch a train and bus to the airport, then board the plane alone.
joy	Manufacturers could play on the excitement of a commentary to sell their sets .
anger	` You 're angry I 'm here , aren't you ? "
joy	See you then .
fear	Yes , he was . I'm not imagining it.Finally , just when I got home , I turned around and looked at him.He was just standing there . He didn't smile . He just stood there . It was so obvious.What should I do ? I'm so scare

2. These datasets are loaded and the feature(text) and target(emotion) are stored in the train and test part of the data.

```
data_train = pd.read_csv('data/data_train.csv', encoding='utf-8')
data_test = pd.read_csv('data/data_test.csv', encoding='utf-8')

X_train = data_train.Text
X_test = data_test.Text

y_train = data_train.Emotion
y_test = data_test.Emotion

data = data_train.append(data_test, ignore_index=True)
```

Emotion value count in the dataset-

```
joy         2326
sadness     2317
anger       2259
neutral     2254
fear        2171
Name: Emotion, dtype: int64
```

Glimpse of data(head)-

	Emotion	Text
0	neutral	There are tons of other paintings that I thin...
1	sadness	Yet the dog had grown old and less capable , a...
2	fear	When I get into the tube or the train without ...
3	fear	This last may be a source of considerable disq...
4	anger	She disliked the intimacy he showed towards so...
5	sadness	When my family heard that my Mother's cousin w...

Pre-processing:

1. Remove the hashtags, username so that the data is consistent and the accuracy is better.

```
def clean_text(data):

    # remove hashtags and @usernames
    data = re.sub(r"(#[\d\w\.]+)", '', data)
    data = re.sub(r"(@[\d\w\.]+)", '', data)

    # tekenization using nltk
    data = word_tokenize(data)

    return data
```

```
texts = [' '.join(clean_text(text)) for text in data.Text]

texts_train = [' '.join(clean_text(text)) for text in X_train]
texts_test = [' '.join(clean_text(text)) for text in X_test]
```

```
print(texts_train[92])
```

Output:

```
a bit ? I 'm extremely annoyed that he did n't phone me when he promised me that he would ! He 's such a liar .
```

2. Convert the text into sequence of integers and count the unique tokens.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(texts)

sequence_train = tokenizer.texts_to_sequences(texts_train)
sequence_test = tokenizer.texts_to_sequences(texts_test)

index_of_words = tokenizer.word_index

# vacab size is number of unique words + reserved 0 index for padding
vocab_size = len(index_of_words) + 1

print('Number of unique words: {}'.format(len(index_of_words)))
```

Output-

```
Number of unique words: 12087
```

3. As all the inputs has to be of the same length **pad** the input data.

```
X_train_pad = pad_sequences(sequence_train, maxlen = max_seq_len )
X_test_pad = pad_sequences(sequence_test, maxlen = max_seq_len )
```

Where the max_seq_len is the maximum length of the text which is 500.

4. Convert the labels into integral values as the text is integral.

```
encoding = {
    'joy': 0,
    'fear': 1,
    'anger': 2,
    'sadness': 3,
    'neutral': 4
}

# Integer labels
y_train = [encoding[x] for x in data_train.Emotion]
y_test = [encoding[x] for x in data_test.Emotion]
```

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Encoded values-

```
array([[0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0.],
       ...,
       [0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.]],
```

Word embedding:

1. The dataset used for the word embedding is the wiki news vector dataset and this is loaded.

```
import urllib.request
import zipfile
import os

fname = 'embeddings/wiki-news-300d-1M.vec'

if not os.path.isfile(fname):
    print('Downloading word vectors...')
    urllib.request.urlretrieve('https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip',
                               'wiki-news-300d-1M.vec.zip')
    print('Unzipping...')
    with zipfile.ZipFile('wiki-news-300d-1M.vec.zip', 'r') as zip_ref:
        zip_ref.extractall('embeddings')
    print('done.')

    os.remove('wiki-news-300d-1M.vec.zip')
```

2. Create the **embedded matrix** which maps each word in our corpus to the corresponding word vector.

```
def create_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # Adding again 1 because of reserved 0 index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))
    with open(filepath) as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(
                    vector, dtype=np.float32)[:embedding_dim]
    return embedding_matrix
```

The shape of embedded matrix -

```
(12088, 300)
```

3. The words which are not present in the pre-trained dataset are calculated.

```
# Inspect unseen words
new_words = 0

for word in index_of_words:
    entry = embedd_matrix[index_of_words[word]]
    if all(v == 0 for v in entry):
        new_words = new_words + 1

print('Words found in wiki vocab: ' + str(len(index_of_words) - new_words))
print('New words found: ' + str(new_words))
```

Output-

```
Words found in wiki vocab: 11442
New words found: 645
```

4. Create the **embedded layer** for the convolution.

```
embedd_layer = Embedding(vocab_size,
                         embed_num_dims,
                         input_length = max_seq_len,
                         weights = [embedd_matrix],
                         trainable=False)
```

Convolution:

1. Create the **1D convolution sequential model** with the embed_layer and the final activation of the dense layer as the **softmax** activation.

```
# Convolution
kernel_size = 3
filters = 256

model = Sequential()
model.add(embedd_layer)
model.add(Conv1D(filters, kernel_size, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(256, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.summary()
```

Model summary-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 300)          3626400
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 498, 256)          230656
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 1285
=================================================================
Total params: 3,924,133
Trainable params: 297,733
Non-trainable params: 3,626,400
_________________________________________________________________
```

2. Train the model with a **batch size** of **256** and for **6 epochs**.

```
batch_size = 256
epochs = 6

hist = model.fit(X_train_pad, y_train,
                 batch_size=batch_size,
                 epochs=epochs,
                 validation_data=(X_test_pad,y_test))
```

Model fit-

```
Train on 7934 samples, validate on 3393 samples
Epoch 1/6
7934/7934 [==============================] - 65s 8ms/step - loss: 1.3612 - acc: 0.4674 - val_loss: 1.0939 - val_acc: 0.6434
Epoch 2/6
7934/7934 [==============================] - 76s 10ms/step - loss: 0.8290 - acc: 0.7221 - val_loss: 0.7868 - val_acc: 0.7109
Epoch 3/6
7934/7934 [==============================] - 84s 11ms/step - loss: 0.6074 - acc: 0.7893 - val_loss: 0.7440 - val_acc: 0.7327
Epoch 4/6
7934/7934 [==============================] - 84s 11ms/step - loss: 0.4874 - acc: 0.8394 - val_loss: 0.7168 - val_acc: 0.7468
Epoch 5/6
7934/7934 [==============================] - 76s 10ms/step - loss: 0.3833 - acc: 0.8890 - val_loss: 0.7067 - val_acc: 0.7548
Epoch 6/6
7934/7934 [==============================] - 60s 8ms/step - loss: 0.2975 - acc: 0.9220 - val_loss: 0.7042 - val_acc: 0.7566
```

3. Calculate the performance metrics namely accuracy, F1 score and confusion matrix.

```
# Accuracy plot
plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()

# Loss plot
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

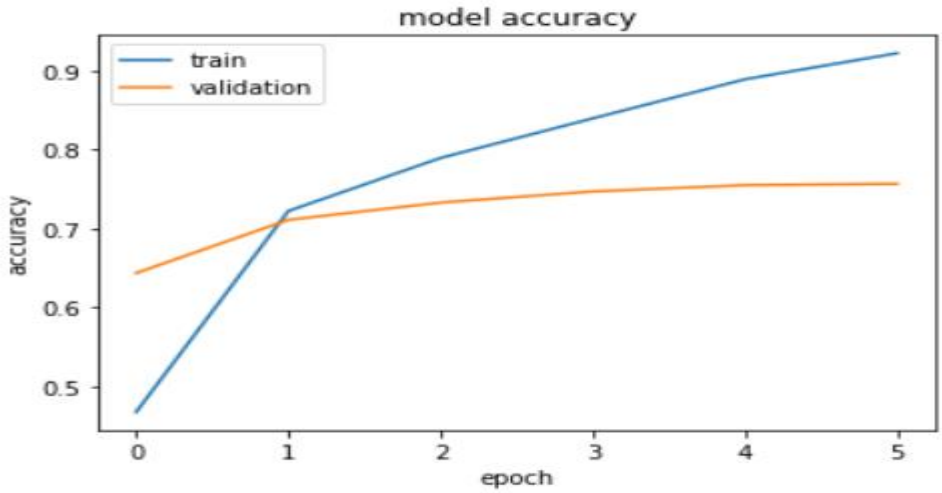
```
predictions = model.predict(X_test_pad)
predictions = np.argmax(predictions, axis=1)
predictions = [class_names[pred] for pred in predictions]
```

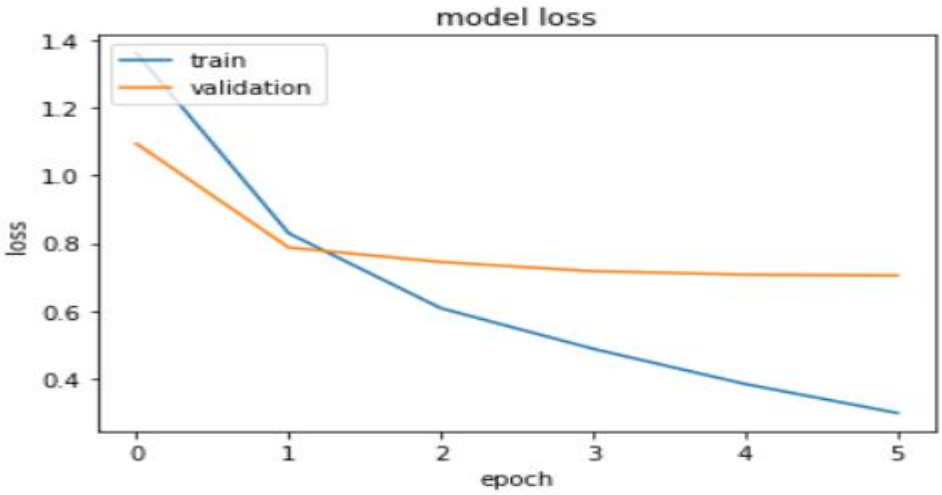
```
print("Accuracy: {:.2f}%".format(accuracy_score(data_test.Emotion, predictions) * 100))
print("\nF1 Score: {:.2f}".format(f1_score(data_test.Emotion, predictions, average='micro') * 100))
```

```
# Plot normalized confusion matrix
plot_confusion_matrix(data_test.Emotion, predictions, classes=class_names, normalize=True, title='Normalized confusion matrix')
plt.show()
```

Output-

Plot of accuracy and loss of the model-



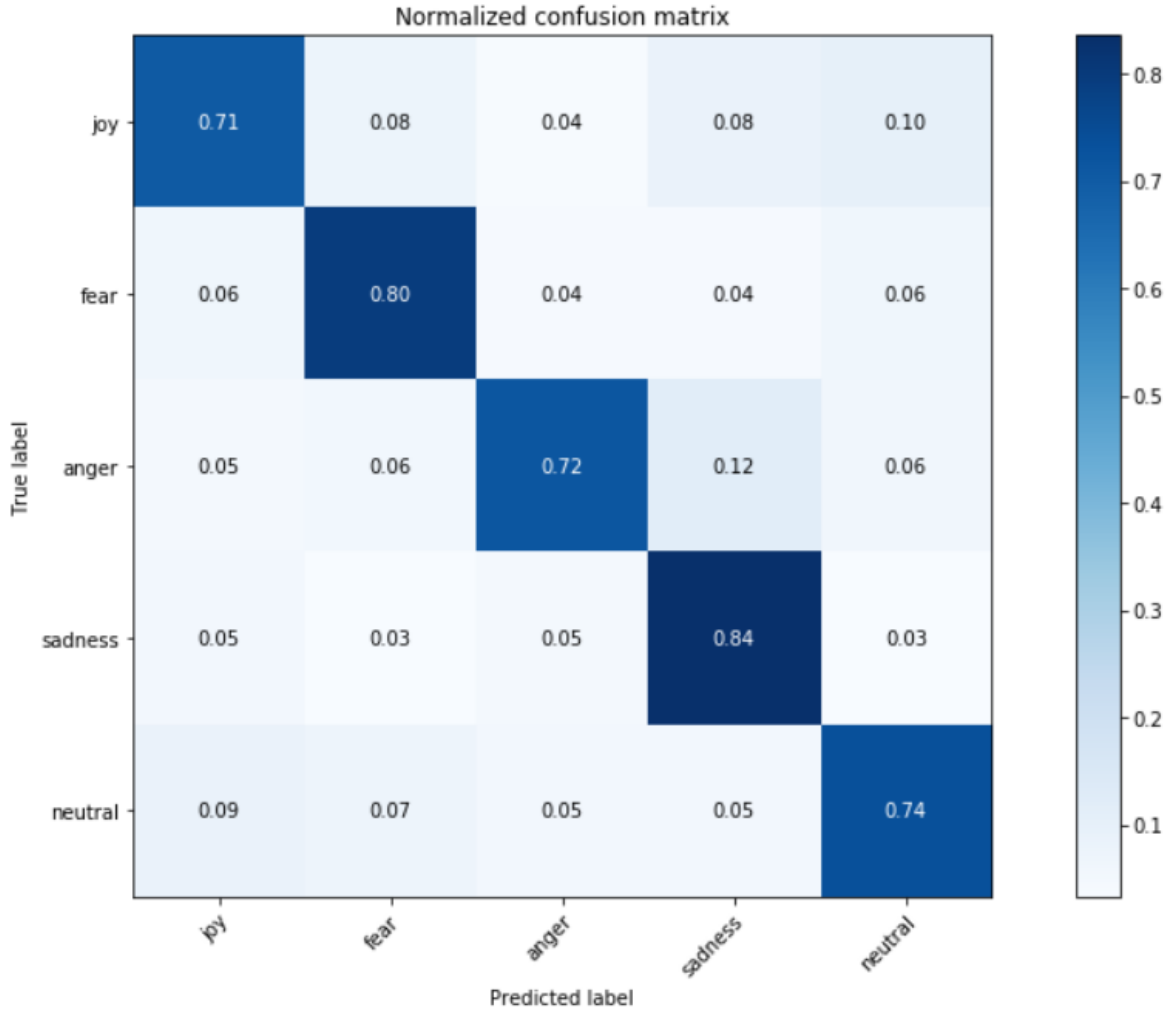


Accuracy and F1 score-

```
Accuracy: 75.66%

F1 Score: 75.66
```

Confusion matrix-



Emotion detection:

1. Predict the emotion of the message with the trained model.

```
from keras.models import load_model
model = load_model('models/cnn_w2v (copy).h5')
```

```
start_time = time.time()
pred = model.predict(padded)

print('Message: ' + str(message))
print('predicted: {} ({:.2f} seconds)'.format(class_names[np.argmax(pred)], (time.time() - start_time)))
predicted = class_names[np.argmax(pred)]
```

Output:

```
Message: ['weather was good']
predicted: neutral (0.28 seconds)
```

Inference:

The predicted emotion is neutral.

Conclusion:

Hence, the emotion of the given input text is predicted using NLP. We can use other NLP techniques also to predict the emotions but word2vector is a better and more efficient method for emotion detection.