In [32]:
```
def final(test_data):
  from keras.preprocessing.text import Tokenizer
  from keras.preprocessing.sequence import pad_sequences
  import pandas as pd
  import seaborn as sns
  import numpy as np
  import pickle
  from sklearn import preprocessing
  from sklearn.model_selection import RandomizedSearchCV
  from sklearn.metrics import log_loss
  from sklearn.metrics import confusion_matrix
  from sklearn.model_selection import train_test_split
  from gensim.scripts.glove2word2vec import glove2word2vec
  from keras.callbacks import EarlyStopping
  from keras.layers import Bidirectional,BatchNormalization,Flatten,Dropout,M
  from keras.preprocessing.text import Tokenizer
  from keras.preprocessing.sequence import pad_sequences
  from keras.utils import np_utils
  from keras.models import Model
  from keras import backend as K
  from datetime import datetime
  from tensorflow.keras.models import model_from_json
  import re
  def preprocess(sentence):
    sentence = sentence.lower()
    sentence = re.sub(r"\x92", "'",sentence)
    sentence = re.sub(r'[0-9\.]+', '',sentence)
    sentence = re.sub(r'\-',' ',sentence)
    sentence = re.sub(r' +', ' ',sentence)
    sentence = re.sub(r'[?!,.]','',sentence)
    sentence = re.sub(r"y'know",'you know',sentence)
    sentence = re.sub(r"\'t", " not", sentence)
    sentence = re.sub(r"\'re", " are", sentence)
    sentence = re.sub(r"\'s", " is", sentence)
    sentence = re.sub(r"\'d", " would", sentence)
    sentence = re.sub(r"\'ll", " will", sentence)
    sentence = re.sub(r"\'t", " not", sentence)
    sentence = re.sub(r"\'ve", " have", sentence)
    sentence = re.sub(r"\'m", " am", sentence)
    return sentence
  new_test=[]
  for i in (test_data.Utterance.values):
    new_test.append(preprocess(i))
  test_data['P_Utterance']=new_test
  X_test = test_data.P_Utterance
  with open('tokenizer.pickle', 'rb') as handle:
      b = pickle.load(handle)
  vocab_size = len(b.word_index) + 1 # for index zero we have to add +1
  encoded_docs_test = b.texts_to_sequences(X_test)
  max_length = 50
  padded_docs_test = pad_sequences(encoded_docs_test, maxlen=max_length, padd
  with open('meld_inputmatrix.pickle', 'rb') as handle:
    input_matrix = pickle.load(handle)
  with open('label_encoder.pickle', 'rb') as handle:
    prep = pickle.load(handle)
  label_dict = dict(zip(prep.transform(prep.classes_), prep.classes_ ))
  sequence_input = Input(shape=(50,))
  embedding_layer = Embedding(vocab_size, 300,weights=[input_matrix],trainabl
  embedded_sequences = embedding_layer(sequence_input)
  x = Conv1D(256, 3, activation='relu')(embedded_sequences)
  x = Dropout(0.5)(x)
  x = BatchNormalization()(x)
```

In [33]:
```
test_data = pd.read_csv('MELD-master/data/MELD/test_sent_emo.csv')
final(test_data)
```

```
Test point  0  predicted_label  neutral
Test point  1  predicted_label  neutral
Test point  2  predicted_label  neutral
Test point  3  predicted_label  neutral
Test point  4  predicted_label  neutral
Test point  5  predicted_label  neutral
Test point  6  predicted_label  neutral
Test point  7  predicted_label  joy
Test point  8  predicted_label  neutral
Test point  9  predicted_label  neutral
Test point  10  predicted_label  neutral
Test point  11  predicted_label  neutral
Test point  12  predicted_label  surprise
Test point  13  predicted_label  neutral
Test point  14  predicted_label  neutral
Test point  15  predicted_label  neutral
Test point  16  predicted_label  neutral
Test point  17  predicted_label  neutral
Test point  18  predicted_label  neutral
Test point  19  predicted_label  neutral
```

In [ ]:
```
```