Assignment:

In [1]:

```
import os
import regex as re
import pandas as pd
from tqdm import tqdm
import numpy as np
```

In [2]:

```
original_txt_lst = []
file_label_lst = []
for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
  with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file), 'r+',e
    doc_data = f.read()
    original_txt_lst.append(doc_data)
    label = re.split("_",file)
    file_label_lst.append(label[0])
```

In [3]:

```
# Preprocessing email
def collect_email_replace():
    total_email_dict = {}
    for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
      try:
          with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file)
            doc_data = f.read()
            email_lst = re.findall(r"[a-zA-Z0-9\.\-+_]+@[a-zA-Z0-9\.\-+_]+", doc_data)
            total_email_dict[file] = email_lst
          if email_lst:
            for email in email_lst:
                doc_data = doc_data.replace(email, " ")
                with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents",
                  f.write(doc_data)
      except Exception as err:
        print(err)
    print("The below documents doesn't have email ids")
    for key in range(len(total_email_dict.keys())):
      if not list(total_email_dict.items())[key][1]:
        print(list(total_email_dict.items())[key][0])
    return total_email_dict

def process_email_words(total_email_dict):
    email_words = []
    for key in range(len(total_email_dict.keys())):
      try:
        each_doc_email_lst = list(total_email_dict.items())[key][1]
        if each_doc_email_lst:
          emailaddr_words = []
          for email in each_doc_email_lst:
            emailaddr = re.split("@", email)[1:]
            word_lst = re.split("\\.", emailaddr[0])
            count = 0
            while count < len(word_lst):
              if  len(word_lst[count]) <= 2 or word_lst[count] == 'com':
                if count == 0:
                    if count == len(word_lst[count]):
                        word_lst = []
                        break
                    else:
                        word_lst = word_lst[count+1:]
                else:
                  word_lst1 = word_lst[:count]
                  if count+1 < len(word_lst):
                      word_lst2 = word_lst[count+1:]
                      word_lst = word_lst1+word_lst2
                  else:
                      word_lst = word_lst1
                  count-=1
              else:
                count+=1
            if word_lst:
                emailaddr_words.extend(word_lst)
          email_words.append(' '.join(item for item in emailaddr_words))
        else:
          email_words.append(' '.join(item for item in each_doc_email_lst))
      except Exception as err:
        print(email)
        print(err)
```

```
    return email_words

def preprocessed_emails():
    email_dict = collect_email_replace()
    email_wordslst = process_email_words(email_dict)
    return email_wordslst
```

In [4]:

```
# Preprocessing subjects
def collect_subject_fro_write_tag_brkt_data():
    all_sub_dict = {}
    fro_write_dict = {}
    tags_dict = {}
    brkt_data_dict = {}
    for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
      try:
          with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file)
            doc_data1 = f.readlines()
            fro_write_lst = []
            tags_lst = []
            brkt_data_lst = []
            for line in doc_data1:
                if re.search("Subject:", line):
                    all_sub_dict[file] = line
                fro = re.findall(r"\bFrom:", line)
                write = re.findall(r"\bWrite to:", line)
                if len(fro) >= 1 or len(write) >= 1:
                    fro_write_lst.append(line)
                tags = re.findall(r'\<.*?\>', line)
                tags_lst.extend(tags)
                brkt_data = re.findall(r'\(.*?\)', line)
                brkt_data_lst.extend(brkt_data)
            fro_write_dict[file] = fro_write_lst
            tags_dict[file] = tags_lst
            brkt_data_dict[file] = brkt_data_lst
      except Exception as err:
        print(err)
    return all_sub_dict,fro_write_dict,tags_dict,brkt_data_dict

def replace_subject_fro_write_tag_brkt_data(all_sub_dict,fro_write_dict,tags_dict,brkt_data
    for i, file in enumerate(os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"
      try:
          with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file)
            doc_data = f.read()
            subject_lst = list(all_sub_dict.values())
            if subject_lst[i]:
                doc_data = doc_data.replace(subject_lst[i], " ")
            fro_wri_lst = list(fro_write_dict.values())[i]
            if fro_wri_lst:
                for item in fro_wri_lst:
                    doc_data = doc_data.replace(item, " ")
            tag_list = list(tags_dict.values())[i]
            if tag_list:
                for item in tag_list:
                    doc_data = doc_data.replace(item, " ")
            brk_dat_lst = list(brkt_data_dict.values())[i]
            if brk_dat_lst:
                for item in brk_dat_lst:
                    doc_data = doc_data.replace(item, " ")
            with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", fil
                f.write(doc_data)
      except Exception as err:
        print(err)

def process_subject_words(all_sub_dict):
    all_sub_words = []
    for key in range(len(all_sub_dict.keys())):
```

```
        sub = list(all_sub_dict.items())[key][1]
        if sub:
            edit_sub = re.split("Subject:", sub)
            edit_sub = re.split(r"\s", edit_sub[1])
            sub_word_lst = []
            for i, word in enumerate(edit_sub):
                if word:
                    edit_wrd_lst = re.split(r"\.", word)
                    if len(edit_wrd_lst)>1:
                        for item in edit_wrd_lst:
                            new_word = ''.join(filter(str.isalnum, item))
                            sub_word_lst.append(new_word)
                    else:
                        new_word = ''.join(filter(str.isalnum, edit_wrd_lst[0]))
                        sub_word_lst.append(new_word)
            all_sub_words.append(' '.join(item for item in sub_word_lst))
        else:
            all_sub_words.append(' '.join(item for item in sub))
    return all_sub_words

def preprocessed_subject_fro_write_tag_brkt_data():
    sub_dict,fro_write_dict,tag_dict,brkt_dict = collect_subject_fro_write_tag_brkt_data()
    subject_wordlst = process_subject_words(sub_dict)
    replace_subject_fro_write_tag_brkt_data(sub_dict,fro_write_dict,tag_dict,brkt_dict)
    return subject_wordlst
```

In [5]:

```
#Preprocessing text
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

# Refered from https://stackoverflow.com/a/47091490/4084039
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def word_processing():
    for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
      try:
        with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file),
            doc_data1 = f.readlines()
            line_lst = []
            for line in doc_data1:
                if line != "\n":
                    line = decontracted(line)
                    wordslst = re.split("\s",line)
                    if wordslst:
                        edited_word_lst = []
                        for word in wordslst:
                            if (word != ""):
                                if re.search(":", word):
                                    word = ""
                                else:
                                    word = re.sub('[^A-Za-z]+', ' ', word)
                                if len(word) > 3 and len(word)<16:
                                    edited_word_lst.append(word)
                                else:
                                    edited_word_lst.append("")
                        line = " ".join(edited_word_lst)
                    line_lst.append(line)
        with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", "temp")
            for line in line_lst:
                f1.write(line)
                f1.write("\n")
        os.remove(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file))
        os.rename(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", "temp")
      except Exception as err:
        print(err)

def chunking():
    # The below code refered from https://stackoverflow.com/questions/31836058/nltk-named-e
    for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
```

```
        try:
            with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file)
              doc_data = f.read()
              parse_tree = nltk.ne_chunk(nltk.tag.pos_tag(doc_data.split()), binary=True)  #
              for t in parse_tree.subtrees():
                  if t.label() == 'PERSON':
                      if len(t)>=2:
                          for name in t:
                              doc_data = doc_data.replace(name[0], " ")
                      else:
                          doc_data = doc_data.replace(t[0][0], " ")
                  if t.label() == 'GPE':
                      if len(t)>=2:
                          for item in t:
                              old_word = ' '.join(item)
                              new_word = '_'.join(item)
                          doc_data = doc_data.replace(old_word, new_word)
              with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", fil
                  f.write(doc_data.lower())
        except Exception as err:
          print(err)

def preprocessed_text():
    for i in range(3):
        word_processing()
    chunking()
    preprocessed_txt_lst = []
    for file in os.listdir("D:\Applied_AI\Assignments\Assignment_21\documents"):
      with open(os.path.join("D:\Applied_AI\Assignments\Assignment_21\documents", file), 'r
        doc_data = f.read()
        preprocessed_txt_lst.append(doc_data)
    return preprocessed_txt_lst
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\PRASAD\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\PRASAD\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\PRASAD\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [6]:

```
def preprocess():
    preprocessed_email = preprocessed_emails()
    preprocessed_subject = preprocessed_subject_fro_write_tag_brkt_data()
    preprocess_text = preprocessed_text()
    return preprocessed_email,preprocessed_subject,preprocess_text
pre_email,pre_subj,pre_text = preprocess()
```

```
The below documents doesn't have email ids
alt.atheism_53121.txt
alt.atheism_53806.txt
comp.os.ms-windows.misc_9671.txt
comp.sys.ibm.pc.hardware_61034.txt
comp.sys.mac.hardware_51507.txt
comp.sys.mac.hardware_51904.txt
comp.sys.mac.hardware_52152.txt
rec.sport.baseball_104352.txt
rec.sport.baseball_104418.txt
rec.sport.baseball_104471.txt
rec.sport.hockey_53642.txt
rec.sport.hockey_53671.txt
```

In [7]:

```
print(pre_text[0])
```

```
assisted dead whose brain wired hour easy listening music
margaret atwood
handmaid tale
story based premise that congress mysteriously
assassinated fundamentalists quickly take charge nation
right again book diary woman life tries live
under christian theocracy women right property revoked
their bank accounts closed sinful luxuries outlawed

radio only used readings from bible crimes punished
doctors performed legal abortions world
hunted down hanged atwood writing style difficult used
first tale grows more more chilling goes
various authors
bible
this somewhat dull rambling work often been criticized however
probably worth reading only that will know what fuss
about exists many different versions make sure
true version
books fiction
peter rosa
```

In [8]:

```
data = {
  "text": original_txt_lst,
  "class": file_label_lst,
  "preprocessed_text": pre_text,
  "preprocessed_subject": pre_subj,
  "preprocessed_emails": pre_email,
}

df = pd.DataFrame(data)
```

In [9]:

```
df.columns
```

Out[9]:

```
Index(['text', 'class', 'preprocessed_text', 'preprocessed_subject',
       'preprocessed_emails'],
      dtype='object')
```

In [10]:

```
df.iloc[400]
```

Out[10]:

```
text                    From: perry@dsinc.com (Jim Perry)\nSubject: Re...
class                                                          alt.atheism
preprocessed_text       this response originally fell into bucket repo...
preprocessed_subject         Re Is Morality Constant was Re Biblical Rape
preprocessed_emails     dsinc darkside osrhe uoknor edu okcforum osrhe...
Name: 400, dtype: object
```

In [10]:

```
df["preprocess_file"] = df["preprocessed_text"] + df["preprocessed_subject"] + df["preproce
```

In [11]:

```
df.head(1)
```

Out[11]:

	text	class	preprocessed_text	preprocessed_subject
0	From: mathew <mathew@mantis.co.uk>\nSubject: A...	alt.atheism	resources\ndecember\natheist resources\naddres...	Alt Atheism FAQ Atheist Resources

In [12]:

```
X = np.array(df["preprocess_file"])
y = np.array(df["class"])
```

In [13]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[13]:

```
((14121,), (4707,), (14121,), (4707,))
```

In [14]:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [15]:

```
tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
tokenizer.fit_on_texts(X_train)
sequences_X_train = tokenizer.texts_to_sequences(X_train)
prepad_sequence_X_train = pad_sequences(sequences_X_train, maxlen=1000)
sequences_X_test = tokenizer.texts_to_sequences(X_test)
prepad_sequence_X_test = pad_sequences(sequences_X_test, maxlen=1000)
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
prepad_sequence_X_train.shape, prepad_sequence_X_test.shape
```

```
89529
```

Out[15]:

```
((14121, 1000), (4707, 1000))
```

In [16]:

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(handle_unknown='ignore')
enc.fit(y_train.reshape(-1, 1))
y_train_onhot = enc.transform(y_train.reshape(-1, 1)).toarray()
y_test_onhot = enc.transform(y_test.reshape(-1, 1)).toarray()

y_train_onhot.shape, y_test_onhot.shape
```

Out[16]:

```
((14121, 20), (4707, 20))
```

In [17]:

```
# Below code is referenced from https://machinelearningmastery.com/use-word-embedding-layer
# load the whole embedding into memory
from numpy import asarray

embeddings_index = {}
f = open('D:\Applied_AI\Assignments\Assignment_21/glove.6B.300d.txt', encoding="utf8", erro
doc_data = f.readlines()
for line in doc_data:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
```

In [18]:

```
# create a weight matrix for words in training docs
from numpy import zeros

embedding_matrix = zeros((vocab_size, 300))
for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [19]:

```
# there are other ways of doing this: https://www.dlology.com/blog/quick-guide-to-run-tensc
%load_ext tensorboard
# Clear any logs from previous runs
#del -rf ./logs/
```

In [20]:

```
from sklearn.metrics import f1_score

class microf1_score(tf.keras.callbacks.Callback):
    def __init__(self,validation_data):
      self.x_test = validation_data[0]
      self.y_test= validation_data[1]

    def on_train_begin(self, logs={}):
        self.history={'val_f1score': []}

    def on_epoch_end(self, epoch, logs={}):
        y_pred= self.model.predict(self.x_test)
        #y_label_pred=np.argmax(y_pred,axis=1)
        f1score = f1_score(self.y_test,y_pred,average='micro')
        self.history['val_f1score'].append(f1score)
```

In [31]:

```
from tensorflow.keras.models import Model
from keras.models import Sequential
from keras.layers import Embedding
from keras.layers.convolutional import Conv1D
from keras.layers import Activation, Dense, Input, Flatten
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping

input_layer = Input(shape=(1000,))
embed_layer = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=1000, tra
layer3_1 = Conv1D(32,5,activation='relu')(embed_layer)
layer3_2 = Conv1D(32,5,activation='relu')(embed_layer)
layer3_3 = Conv1D(32,5,activation='relu')(embed_layer)
layer3_4 = Conv1D(32,5,activation='relu')(embed_layer)
layer4 = tf.keras.layers.Concatenate()([layer3_1, layer3_2, layer3_3, layer3_4])
maxpool1 = tf.keras.layers.MaxPooling1D(5)(layer4)
layer6_1 = Conv1D(32,5,activation='relu')(maxpool1)
layer6_2 = Conv1D(32,5,activation='relu')(maxpool1)
layer6_3 = Conv1D(32,5,activation='relu')(maxpool1)
layer6_4 = Conv1D(32,5,activation='relu')(maxpool1)
layer7 = tf.keras.layers.Concatenate()([layer6_1, layer6_2, layer6_3, layer6_4])
maxpool2 = tf.keras.layers.MaxPooling1D(5)(layer7)
layer9_1 = Conv1D(32,5,activation='relu')(maxpool2)
layer9_2 = Conv1D(32,5,activation='relu')(maxpool2)
layer9_3 = Conv1D(32,5,activation='relu')(maxpool2)
layer9_4 = Conv1D(32,5,activation='relu')(maxpool2)
layer10 = tf.keras.layers.Concatenate()([layer9_1, layer9_2, layer9_3, layer9_4])
maxpool3 = tf.keras.layers.MaxPooling1D(5)(layer10)
layer12 = Conv1D(32,5,activation='relu')(maxpool3)
flatten = Flatten()(layer12)
dropout = tf.keras.layers.Dropout(0.1)(flatten)
layer16 = Dense(128,activation='relu')(dropout)
output = Dense(20,activation='softmax')(layer16)

model1 = Model(inputs=input_layer,outputs=output)

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model1.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])

filepath="./best_model-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_be

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.02, patience=1, verbose=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./logs")

model1.fit(prepad_sequence_X_train,y_train_onhot,epochs=100,validation_data=(prepad_sequenc
```

```
Epoch 1/100
221/221 [==============================] - 403s 2s/step - loss: 2.2611 - acc
uracy: 0.2370 - val_loss: 1.5440 - val_accuracy: 0.4417

Epoch 00001: val_accuracy improved from -inf to 0.44168, saving model to .\b
est_model-01-0.4417.h5
Epoch 2/100
221/221 [==============================] - 374s 2s/step - loss: 1.3115 - acc
uracy: 0.5350 - val_loss: 1.1446 - val_accuracy: 0.5949
```

```
Epoch 00002: val_accuracy improved from 0.44168 to 0.59486, saving model to
.\best_model-02-0.5949.h5
Epoch 3/100
221/221 [==============================] - 394s 2s/step - loss: 0.9557 - acc
uracy: 0.6632 - val_loss: 0.9689 - val_accuracy: 0.6745

Epoch 00003: val_accuracy improved from 0.59486 to 0.67453, saving model to
.\best_model-03-0.6745.h5
Epoch 4/100
221/221 [==============================] - 368s 2s/step - loss: 0.6941 - acc
uracy: 0.7539 - val_loss: 0.9312 - val_accuracy: 0.7121

Epoch 00004: val_accuracy improved from 0.67453 to 0.71213, saving model to
.\best_model-04-0.7121.h5
Epoch 5/100
221/221 [==============================] - 377s 2s/step - loss: 0.5084 - acc
uracy: 0.8206 - val_loss: 0.9236 - val_accuracy: 0.7317

Epoch 00005: val_accuracy improved from 0.71213 to 0.73168, saving model to
.\best_model-05-0.7317.h5
Epoch 00005: early stopping
```

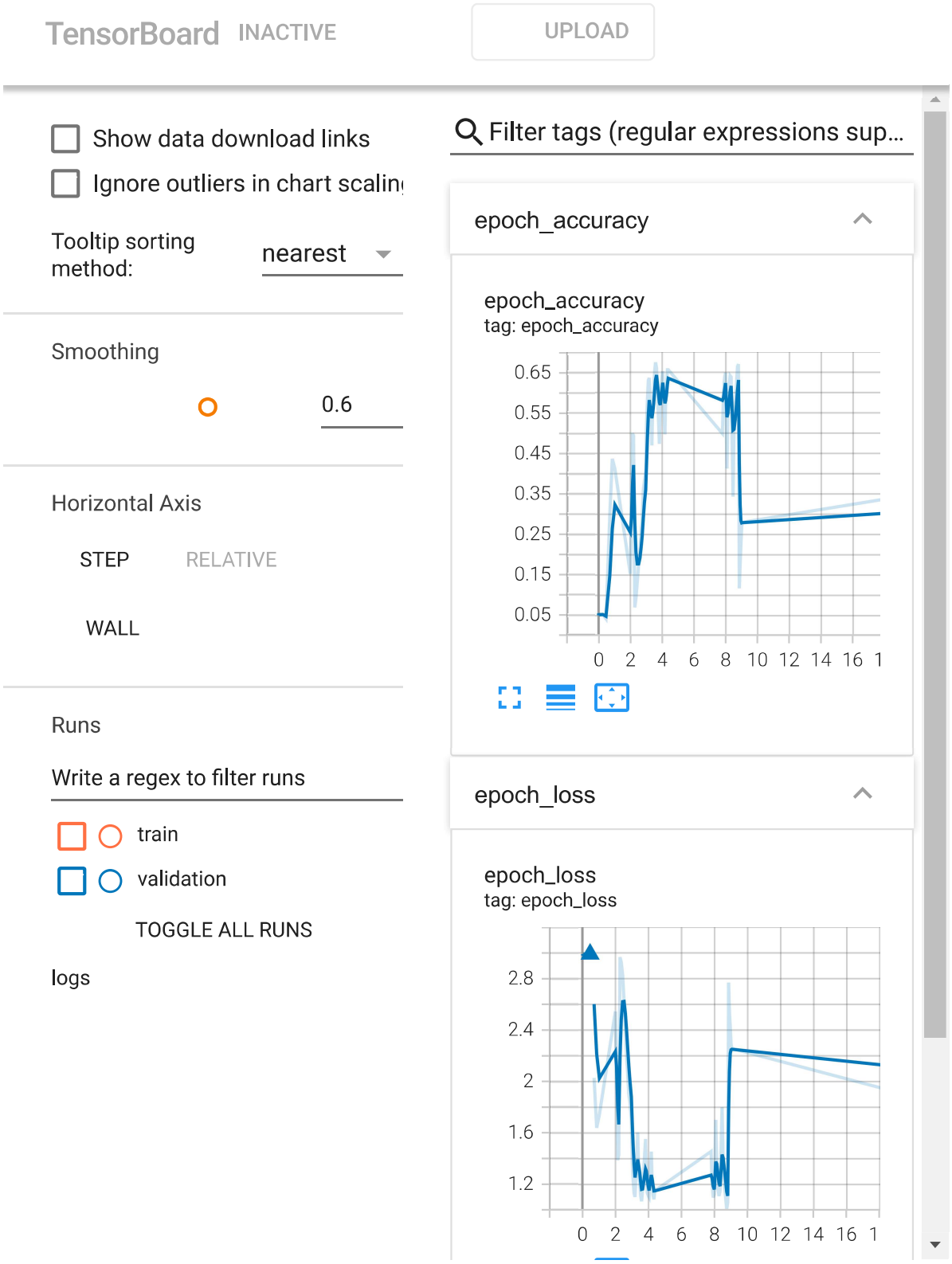
Out[31]:

```
<keras.callbacks.History at 0x26450314a20>
```

In [33]:

```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 20492), started 0:05:46 ago. (Use '!ki
ll 20492' to kill it.)
```



In [34]:

```
tokenizer1 = Tokenizer(num_words=None,char_level = True, oov_token='UNK')
tokenizer1.fit_on_texts(X_train)
```

In [35]:

```
tokenizer1.word_index
```

Out[35]:


```
{'UNK': 1,
 ' ': 2,
 'e': 3,
 't': 4,
 'a': 5,
 'i': 6,
 's': 7,
 'r': 8,
 'n': 9,
 'o': 10,
 'l': 11,
 'h': 12,
 'c': 13,
 'd': 14,
 '\n': 15,
 'u': 16,
 'm': 17,
 'p': 18,
 'g': 19,
 'w': 20,
 'y': 21,
 'b': 22,
 'f': 23,
 'v': 24,
 'k': 25,
 'x': 26,
 'j': 27,
 'z': 28,
 'q': 29,
 '1': 30,
 '0': 31,
 '2': 32,
 '-': 33,
 '3': 34,
 '4': 35,
 '6': 36,
 '5': 37,
 '8': 38,
 '9': 39,
 '7': 40,
 '_': 41,
 '+': 42}
```

In [36]:

```
X_train_seq = tokenizer1.texts_to_sequences(X_train)
X_train_pad_seq = pad_sequences(X_train_seq, maxlen=1000)
X_test_seq = tokenizer1.texts_to_sequences(X_test)
X_test_pad_seq = pad_sequences(X_test_seq, maxlen=1000)
vocab_size = len(tokenizer1.word_index) + 1
print(vocab_size)
X_train_pad_seq.shape, X_test_pad_seq.shape
```

```
43
```

Out[36]:

```
((14121, 1000), (4707, 1000))
```

In [37]:

```
# Below code is referenced from https://machinelearningmastery.com/use-word-embedding-layer
# load the whole embedding into memory

embeddings_index = {}
f = open('D:\Applied_AI\Assignments\Assignment_21\glove.840B.300d-char.txt', encoding="utf8
doc_data = f.readlines()
for line in doc_data:
    values = line.split()
    character = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[character] = coefs
f.close()
```

In [40]:

```
# create a weight matrix for words in training docs
from numpy import zeros

embedding_matrix = zeros((vocab_size, 300))
for word, i in tokenizer1.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [45]:

```
input_layer_c = Input(shape=(1000,))
embed_layer_c = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length=1000, t
layer3_c = Conv1D(32,5,activation='relu')(embed_layer_c)
layer4_c_1 = Conv1D(32,5,activation='relu')(layer3_c)
layer4_c_2 = Conv1D(32,5,activation='relu')(layer3_c)
concatlayer = tf.keras.layers.Concatenate()([layer4_c_1, layer4_c_2])
maxpool1_c = tf.keras.layers.MaxPooling1D(5)(concatlayer)
layer6_c = Conv1D(32,5,activation='relu')(maxpool1_c)
layer7_c = Conv1D(32,5,activation='relu')(layer6_c)
maxpool2_c = tf.keras.layers.MaxPooling1D(5)(layer7_c)
flatten_c = Flatten()(maxpool2_c)
dropout_c = tf.keras.layers.Dropout(0.1)(flatten_c)
layer11_c = Dense(128,activation='relu')(dropout_c)
output_c = Dense(20,activation='softmax')(layer11_c)

model2 = Model(inputs=input_layer_c,outputs=output_c)

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)

model2.compile(optimizer=optimizer, loss='categorical_crossentropy',metrics=['accuracy'])

filepath="./best_model-{epoch:02d}-{val_accuracy:.4f}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy', verbose=1, save_bes

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.02, patience=1, verbose=1)

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="./Char_logs")

model2.fit(X_train_pad_seq,y_train_onhot,epochs=100,validation_data=(X_test_pad_seq,y_test_
```

```
Epoch 1/100
221/221 [==============================] - 129s 580ms/step - loss: 2.9458 -
accuracy: 0.0788 - val_loss: 2.9126 - val_accuracy: 0.0833

Epoch 00001: val_accuracy improved from -inf to 0.08328, saving model to .\b
est_model-01-0.0833.h5
Epoch 2/100
221/221 [==============================] - 127s 577ms/step - loss: 2.9080 -
accuracy: 0.0910 - val_loss: 2.8751 - val_accuracy: 0.0952

Epoch 00002: val_accuracy improved from 0.08328 to 0.09518, saving model to
.\best_model-02-0.0952.h5
Epoch 00002: early stopping
```

Out[45]:

```
<keras.callbacks.History at 0x263c2e269b0>
```