In [3]:

```
#https://drive.google.com/uc?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM&export=download
!wget --header="Host: doc-00-7s-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135
Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7" --header
="Referer: https://drive.google.com/uc?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM&export=download" --hea
der="Cookie: AUTH_smvf2o367eja801lv3hmgenqovh1tbcl_nonce=6jb4na2ggl3gk;
_ga=GA1.2.1476194893.1587472682" --header="Connection: keep-alive" "https://doc-00-7s-
docs.googleusercontent.com/docs/securesc/9csvdbmvo9gt489glsl99tqs7subbnk6/vbp4eb98t40uel485thqmv438
jpp/1598356500000/00484516897554883881/01088116874641946513/1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM?e=do
wnload&authuser=0&nonce=6jb4na2ggl3gk&user=01088116874641946513&hash=j0l5m7rcg7oublfdi6lad7tfnpj1p
-c -O 'documents.rar'
```

```
--2020-08-25 11:56:30--  https://doc-00-7s-
docs.googleusercontent.com/docs/securesc/9csvdbmvo9gt489glsl99tqs7subbnk6/vbp4eb98t40uel485thqmv438
jpp/1598356500000/00484516897554883881/01088116874641946513/1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM?
e=download&authuser=0&nonce=6jb4na2ggl3gk&user=01088116874641946513&hash=j0l5m7rcg7oublfdi6lad7tfnp
64
Resolving doc-00-7s-docs.googleusercontent.com (doc-00-7s-docs.googleusercontent.com)...
172.217.203.132, 2607:f8b0:400c:c07::84
Connecting to doc-00-7s-docs.googleusercontent.com (doc-00-7s-
docs.googleusercontent.com)|172.217.203.132|:443... connected.
HTTP request sent, awaiting response... 416 Requested range not satisfiable

    The file is already fully retrieved; nothing to do.
```

In [4]:

```
!pip3 install patool
```

```
Requirement already satisfied: patool in /usr/local/lib/python3.6/dist-packages (1.12)
```

In [5]:

```
!pip3 install pyunpack
```

```
Requirement already satisfied: pyunpack in /usr/local/lib/python3.6/dist-packages (0.2.1)
Requirement already satisfied: easyprocess in /usr/local/lib/python3.6/dist-packages (from
pyunpack) (0.3)
Requirement already satisfied: entrypoint2 in /usr/local/lib/python3.6/dist-packages (from
pyunpack) (0.2.1)
Requirement already satisfied: argparse in /usr/local/lib/python3.6/dist-packages (from
entrypoint2->pyunpack) (1.4.0)
```

In [6]:

```
from pyunpack import Archive
```

In [7]:

```
import os
```

In [8]:

```
Archive("/content/documents.rar").extractall('/content')
```

In [9]:

```
len(os.listdir("/content/documents"))
```

Out[9]:

```
18828
```

In [10]:

```
text_files=[]
```

In [11]:

```
d=r"/content/documents"
for i in os.listdir(r"/content/documents"):
    fullpath=os.path.join(d,i)
    a=open(fullpath, encoding="latin1")
    text_files.append(a.read())
```

In [12]:

```
import re
```

In [13]:

```
def findmailid(document):
    result=re.findall("[\w\.]+@[\w\.]+" , document) #finding email addresses
    string1=[]
    for k in result:
        result1=re.findall("@[\w\.-]+", k)
        wq=re.sub("\@", "", result1[0])
        split1=re.split("\." , wq)
        for j in split1:
            if j!='com' and len(j)>2:
                string1.append(j)
    x=" ".join(string1)

    return x
```

In [14]:

```
def subject(document):
    a1=re.findall("Subject:.+", document) # finding the subject (whole line of subject)
    a2=re.sub("Subject:.+:","",a1[0]) # removing anything before :
    a3=re.sub("[^A-Za-z0-9]+"," ",a2) #removing special charcater , punctuation , newline
    return a3
```

In [15]:

```
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
```

In [16]:

```
#https://medium.com/@bhor733/data-cleaning-and-preprocessing-in-data-science-and-machine-learning-8931c36c04ed
def chunking(document):
    actual_GPE_lst=[]
    GPE_lst=[]
    person_lst=[]
    for sent in nltk.sent_tokenize(document):
        for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
```

```
            if hasattr(chunk, 'label'):
                a=chunk.label()
                if a=="GPE":
                    d=' '.join(c[0] for c in chunk)
                    actual_GPE_lst.append(d)
                    b='_'.join(c[0] for c in chunk)
                    GPE_lst.append(b)
                if a=="PERSON":
                    b=' '.join(c[0] for c in chunk)
                    person_lst.append(b)
    if len(GPE_lst)!=0:
        for i in range(len(GPE_lst)):
            if GPE_lst[i]!=actual_GPE_lst[i]:
                #document=re.sub(actual_GPE_lst[i] , GPE_lst[i] , document)
                document=document.replace(actual_GPE_lst[i], GPE_lst[i])

    if len(person_lst)!=0:
        for j in range(len(person_lst)):
            #document=re.sub(person_lst[j], " ", document)
            document=document.replace(person_lst[j],"")

    return document
```

In [17]:

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/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]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[17]:

```
True
```

In [18]:

```
s0="i am living in the New York and My name is Srikanth Varma"
chunking(s0)
```

Out[18]:

```
'i am living in the New_York and My name is '
```

In [19]:

```
def removing_(document):
    rt1=[]
    a=re.findall("[_][a-zA-Z]+[_]",document)     # finding    _word_
    for i in a:
        a1=re.sub("_"," ",i)
        rt1.append(a1)
    for i in range(len(rt1)):
        document=re.sub(a[i],rt1[i],document)

    rt2=[]
    b=re.findall("[_][a-zA-Z]+",document)        # finding   _word
    for i in b:
        b1=re.sub("_"," ",i)
        rt2.append(b1)
    for i in range(len(rt2)):
```

```
        document=re.sub(b[i],rt2[i],document)

    rt3=[]
    c=re.findall("[a-zA-Z]+[_]",document)        # finding    word_
    for i in c:
        c1=re.sub("_"," ",i)
        rt3.append(c1)
    for i in range(len(rt3)):
        document=re.sub(c[i],rt3[i],document)

    return document
```

In [20]:

```
#example of step14
s1="_lebron_ james is the best player in _world . leaker_ can actually win the championship this _
year"
removing_(s1)
```

Out[20]:

```
' lebron  james is the best player in  world . leaker  can actually win the championship this  yea
r'
```

In [21]:

```
def splitandremove_(document):
    kt1=[]
    d=re.findall("[a-zA-z][a-zA-Z]_[a-zA-Z]+",document)  #removing dr_berlin --> berlin
    for i in d:
        d1=re.split("_",i)
        kt1.append(d1[1])
    for j in range(len(kt1)):
        document=re.sub(d[j],kt1[j],document)

    kt2=[]
    e=re.findall("[a-zA-z]_[a-zA-Z]+",document) #removing d_berlin --> berlin
    for i in e:
        e1=re.split("_", i)
        kt2.append(e1[1])
    for j in range(len(kt2)):
        document=re.sub(e[j],kt2[j],document)

    return document
```

In [22]:

```
#example of step15
s2="d_berlin dr_berlin , dr_dre is the name still running the game"
splitandremove_(s2)
```

Out[22]:

```
'berlin berlin , dre is the name still running the game'
```

In [23]:

```
def step16(document):
    document=document.lower() # making every word lower case
    document=' '.join([w for w in document.split() if len(w)>2 and len(w)<15 ]) # removing words wi
ch are leass than 2 and greater than 15 in length
    return document
```

In [24]:

```
#example of step16
s3="derrick rose was best player untill his injurie hopehecometohisoldform , toogoodtoofast1111 ,
derickroseicanflywhynot   i am okay ye  what"
step16(s3)
```

Out[24]:

```
'derrick rose was best player untill his injurie okay what'
```

In [25]:

```
def step17(document):
    document=re.sub("[^a-zA-Z_ ]+"," ",document)
    return document
```

In [26]:

```
def preprocess(document):
    preproessed_email=findmailid(document)  #step1
    preproessed_subject=subject(document)    #step3
    preproessed_text1=re.sub("[\w\.]+@[\w\.]+", " ", document)    #step2
    preproessed_text2=re.sub("Subject:.+"," ",preproessed_text1)  #step4
    preproessed_text3=re.sub("From:.+"," ",preproessed_text2)
    preproessed_text4=re.sub("Write to:.+"," ", preproessed_text3)
    preproessed_text5=re.sub("<.*?>"," ",preproessed_text4)  #step6
#https://medium.com/@jorlugaqui/how-to-strip-html-tags-from-a-string-in-python-7cb81a2bbf44
    preproessed_text6=re.sub("[(].+[)]"," ",preproessed_text5) # step7
    preproessed_text7=preproessed_text6.replace("\n"," ") #step8
    preproessed_text7=preproessed_text7.replace("\t"," ") #step8
    preproessed_text7=preproessed_text7.replace("\ ", " ") #step8
    preproessed_text7=preproessed_text7.replace("-"," ") #step8
#https://medium.com/@bhor733/data-cleaning-and-preprocessing-in-data-science-and-machine-learning-8931c36c04ed
    preproessed_text8=re.sub("\w+:"," ",preproessed_text7) #step9
    preproessed_text9=decontracted(preproessed_text8) #step10
    preproessed_text10=chunking(preproessed_text9) #Step11,12 chunking
    preproessed_text11=re.sub('[0-9]',' ',preproessed_text10) #step13
    preproessed_text12=removing_(preproessed_text11) #step14
    preproessed_text13=splitandremove_(preproessed_text12) #step15
    preproessed_text14=step17(preproessed_text13) #step17
    preproessed_text15=step16(preproessed_text14)  #step16



    return preproessed_text15,preproessed_email,preproessed_subject
```

creating lables

In [27]:

```
labels=[]
for i in os.listdir(r"/content/documents"):
    labels.append(i.split('_')[0])
```

In [28]:

```
len(labels)
```

Out[28]:

```
18828
```

In [29]:

```
only_lables=set(labels)
len(only_lables)
```

Out[29]:

```
20
```

In [30]:

```
label_dict={}
```

In [31]:

```
for i , j in enumerate(only_lables):
    label_dict[j]=i
```

In [32]:

```
print(label_dict)
```

```
{'talk.politics.misc': 0, 'sci.med': 1, 'talk.politics.guns': 2, 'comp.sys.ibm.pc.hardware': 3, 'r
ec.motorcycles': 4, 'soc.religion.christian': 5, 'rec.sport.baseball': 6, 'alt.atheism': 7,
'rec.sport.hockey': 8, 'misc.forsale': 9, 'comp.os.ms-windows.misc': 10, 'sci.electronics': 11,
'comp.sys.mac.hardware': 12, 'talk.politics.mideast': 13, 'comp.graphics': 14, 'sci.space': 15, 'c
omp.windows.x': 16, 'talk.religion.misc': 17, 'sci.crypt': 18, 'rec.autos': 19}
```

In [33]:

```
true_labels=[]
for i in labels:
    true_labels.append(label_dict.get(i))
```

In [34]:

```
len(true_labels)
```

Out[34]:

```
18828
```

preprocessing

In [35]:

```
from tqdm import tqdm
```

In [36]:

```
text=[]
preprocessed_emails=[]
preprocessed_subjects=[]
preprocessed_texts=[]
for i in tqdm(range(len(text_files))):
    text.append(text_files[i])
    preprocessed_text,preprocessed_email,preprocessed_subject=preprocess(text_files[i])
    preprocessed_emails.append(preprocessed_email)
    preprocessed_subjects.append(preprocessed_subject)
    preprocessed_texts.append(preprocessed_text)
```

```
100%|██████████| 18828/18828 [32:03<00:00,  9.79it/s]
```

In [37]:

```
len(text)
```

Out[37]:

```
18828
```

In [38]:

```
import pandas as pd
```

In [39]:

```
data=pd.DataFrame(list(zip(text,labels,preprocessed_texts,preprocessed_subjects,preprocessed_emails
)),columns=["text","class","preprocessed_texts","preprocessed_subjects","preprocessed_emails"])
```

In [40]:

```
data
```

Out[40]:

	text	class	preprocessed_texts	preprocessed_subjects	preprocessed_emails
0	From: meg_arnold@qm.sri.com (Meg Arnold)\nSubj...	sci.med	looking for statistics the prevalence disorder...	Subject Botulinum Toxin type A	sri sri
1	From: robert@isgtec.com (Robert Osborne)\nSubj...	talk.politics.guns	posted talk politics guns from can salyzyn bee...	Subject Waco	isgtec ve6mgs ampr org world std isgtec
2	From: strom@Watson.Ibm.Com (Rob Strom)\nSubjec...	alt.atheism	article ida org however hate economic terroris...	soc motss et al Princeton axes matching funds...	Watson Ibm Com harder ccr harder ccr watson ibm
3	From: jejones@microware.com (James Jones)\nSub...	comp.graphics	article article muchado and cpu does not help ...	Rumours about 3DO	microware mercury unt edu unt edu rchland ibm ...
4	From: dma7@po.CWRU.Edu (Daniel M. Alt)\nSubjec...	comp.graphics	have very large file macintosh canvas somethin...	Subject Interesting conversion Problem	CWRU Edu
...	...	...	...	...	...
18823	From: sesrock@andy.bgsu.edu (Stuart Esrock)\nS...	rec.sport.hockey	joe pitt edu come boston where the hell are yo...	Bruins	andy bgsu edu
18824	From: The Always Fanatical: Patrick Ellis <IO1...	rec.sport.hockey	well just read the boston globe that while not...	Subject Keenan signs Plus WALSH	MAINE MAINE EDU
18825	From: zmed16@trc.amoco.com (Michael)\nSubject:...	misc.forsale	have fostex track recorder for sale excellent ...	4 TRACK RECORDER	trc amoco trc amoco
18826	From: baseball@catch-the-fever.scd.ucar.edu (G...	rec.sport.baseball	rec sport baseball wrote the original poster w...	HBP BB BIG CAT	catch QUCDN QueensU ncar ucar edu
18827	From: steerr@h01.UUCP (R. William Steer)\nSubj...	comp.windows.x	does anybody have server for that they are wil...	Subject X server for NT	h01 UUCP

18828 rows × 5 columns

In [41]:

```
data.shape
```

Out[41]:

```
(18828, 5)
```

In [42]:

```
data.columns
```

Out[42]:

```
Index(['text', 'class', 'preprocessed_texts', 'preprocessed_subjects',
       'preprocessed_emails'],
      dtype='object')
```

In [43]:

```
data.iloc[400]
```

Out[43]:

```
text                  From: bobbe@vice.ICO.TEK.COM (Robert Beauchain...
class                                                       alt.atheism
preprocessed_texts    article what you base your belief atheism your...
preprocessed_subjects  After 2000 years can we say that Christian Mo...
preprocessed_emails   vice ICO TEK COM news cso uiuc edu alexia lis ...
```

Name: 400, dtype: object

In [44]:

```
data.to_csv('preprocessed_data.csv') # saving dataframe to csv file
```

checking output on alt.atheism_49960 as per instrictions

In [45]:

```
t=open("/content/documents/alt.atheism_49960.txt",encoding="latin1")
```

In [46]:

```
t1=t.read()
```

In [47]:

```
pre_textt1,pre_emailt1,pre_subjectt1=preprocess(t1)
```

In [48]:

```
pre_textt1
```

Out[48]:

'archive atheism resources alt atheism archive resources last december atheist resources addresses atheist organizations usa freedom from religion foundation fish bumper stickers and assorted other atheist paraphernalia are available from the freedom from religion foundation the evolution design s evolution designs sell the fish fish symbol like the ones stick their cars but with feet and the word written inside the deluxe moulded plastic fish postpaid the people the san francisco bay area can get fish from try mailing for net people who directly the price per fish american atheist pres s aap publish various atheist books critiques the bible lists biblical contradictions and one such book the bible handbook ball and foote american isbn edition bible contradictions absurdities atro cities immoralities contains ball the bible contradicts itself aap based the king version the bibl e austin prometheus books sell books including holy horrors alternate address prometheus african americans for humanism organization promoting black secular humanism and uncovering the history bl ack freethought they publish quarterly newsletter aah examiner united kingdom rationalist press as sociation national secular society street holloway road london london british humanist association south place ethical society lamb conduit passage conway hall london red lion square london fax the national secular society publish the freethinker monthly magazine founded germany ibka bund der un d berlin germany ibka publish miz materialien und zur zeit politisches journal der und hrsg ibka m iz postfach berlin germany for atheist books write ibdk ucherdienst der hannover germany fiction t homas disch the claus compromise short story the ultimate proof that exists all characters and eve nts are fictitious any similarity living dead gods well walter miller canticle for leibowitz one g em this post atomic doomsday novel the monks who spent their lives copying blueprints from saint l eibowitz filling the sheets paper with ink and leaving white lines and letters edgar pangborn davy post atomic doomsday novel set clerical states the church for example forbids that anyone produce describe use any substance containing atoms philip dick wrote many philosophical and thought provoking short stories and novels his stories are bizarre times but very approachable wrote mainl y but wrote about people truth and religion rather than technology although often believed that ha d met some sort remained sceptical amongst his novels the following are some fallible alien deity summons group craftsmen and women remote planet raise giant cathedral from beneath the oceans when the deity begins demand faith from the earthers pot healer unable comply polished ironic and amusi ng novel maze death noteworthy for its description technology based religion valis the schizophrenic hero searches for the hidden mysteries gnostic ity after reality fired into his brai n pink laser beam unknown but possibly divine origin accompanied his dogmatic and dismissively ath eist friend and assorted other odd characters the divine invasion invades making young woman pregn ant she returns from another star system unfortunately she terminally ill and must assisted dead m an whose brain wired hour easy listening music margaret atwood the handmaid story based the premise that the congress mysteriously assassinated and quickly take charge the nation set right a gain the book the diary woman life she tries live under the new theocracy women right own property revoked and their bank accounts are closed sinful luxuries are outlawed and the radio only used fo r readings from the bible crimes are punished doctors who performed legal abortions the old world are hunted down and hanged atwood writing style difficult get used first but the tale grows more a nd more chilling goes various authors the bible this somewhat dull and rambling work has often bee n criticized however probably worth reading only that you will know what all the fuss about exists many different versions make sure you get the one true version non fiction peter rosa vicars chris t although seems even catholic this very enlighting history papal immoralities adulteries fallacies etc german gottes erste die dunkle seite des michael martin philosophical justification philadelphia usa detailed and scholarly justification atheism contains outstanding appendix defini

ng terminology and usage this tendentious area argues both for negative atheism the non belief the existence god and also for positive atheism the belief the non existence god includes great refutations the most challenging arguments for god particular attention paid refuting contempory theists such and swinburne pages isbn the case against ity comprehensive critique ity which considers the best contemporary defences ity and demonstrates that they are unsupportable and incoherent pages isbn turner the johns hopkins university press baltimore usa subtitled the origins unbelief america examines the way which unbelief became mainstream alternative world view focusses the period and while considering france and britain the emphasis american and particularly new england development s neither religious history secularization atheism rather the intellectual history the fate single idea the belief that exists pages isbn george seldes the great thoughts ballantine new york usa dictionary quotations different kind concentrating statements and writings which explicitly implicitly present the person philosophy and world view includes obscure opinions from many people for some popular observations traces the way which various people expressed and twisted the idea over the centuries quite number the quotations are derived from cardiff what religion and views religion pages isbn richard swinburne the existence oxford this book the second volume trilogy that began with the coherence theism this work swinburne attempts construct series inductive arguments for the existence his arguments which are somewhat tendentious and rely upon the imputation late century western values and aesthetics which supposedly simple can conceived were decisively rejected the miracle theism the revised edition the existence swinburne includes appendix which makes somewhat incoherent attempt rebut mackie the miracle theism oxford this volume contains comprehensive review the principal arguments for and against the existence ranges from the classical philosophical positions descartes anselm through the moral arguments newman kant and the recent re statements the classical theses and swinburne also addresses those positions which push the concept beyond the realm the rational such those kierkegaard and well replacements for such axiarchism the book delight read less formalistic and better written than works and refreshingly direct when compared with the hand waving swinburne haught holy illustrated history religious murder and madness prometheus looks religious persecution from ancient times the present day and not only library congress catalog card number norm allen african american anthology see the listing for african americans for humanism above gordon stein anthology atheism and rationalism prometheus anthology covering wide range subjects including the devil and morality and the history freethought comprehensive bibliography edmund cohen the mind the bible believer prometheus study why people become and what effect has them net resources there small mail based archive server mantis which carries archives old alt atheism moderated articles and assorted other files for more information send mail archive saying help send atheism index and will mail back reply mathew'

In [49]:

```
pre_emailt1
```

Out[49]:

```
'mantis netcom mantis'
```

In [50]:

```
pre_subjectt1
```

Out[50]:

```
' Atheist Resources'
```

In [51]:

```
#https://www.geeksforgeeks.org/join-two-text-columns-into-a-single-column-in-pandas/
data["preprocessed_full_text"]=data["preprocessed_texts"].str.cat(data["preprocessed_subjects"],sep=" ")
data["preprocessed_full_text"]=data["preprocessed_full_text"].str.cat(data["preprocessed_emails"],sep=" ")
```

In [52]:

```
data.head()
```

Out[52]:

	text	class	preprocessed_texts	preprocessed_subjects	preprocessed_emails	preprocessed_full_text
0	From: meg_arnold@qm.sri.com (Meg Arnold)\nSubj...	sci.med	looking for statistics the prevalence disorder...	Subject Botulinum Toxin type A	sri sri	looking for statistics the prevalence disorder...
1	From: robert@isgtec.com	talk.politics.guns	posted talk politics guns from can	Subject Waco	isgtec ve6mgs ampr	posted talk politics guns

	text	class	preprocessed_texts	preprocessed_subjects	preprocessed_emails	preprocessed_full_text
	(Robert Osborne)\nSubj...	talk.politics.guns	guns from can salyzyn bee...	Subject Waco	org world std isgtec	from can salyzyn bee...
2	From: strom@Watson.Ibm.Com (Rob Strom)\nSubjec...	alt.atheism	article ida org however hate economic terroris...	soc motss et al Princeton axes matching funds...	Watson Ibm Com harder ccr harder ccr watson ibm	article ida org however hate economic terroris...
3	From: jejones@microware.com (James Jones)\nSub...	comp.graphics	article article muchado and cpu does not help ...	Rumours about 3DO	microware mercury unt edu unt edu rchland ibm ...	article article muchado and cpu does not help ...
4	From: dma7@po.CWRU.Edu (Daniel M. Alt)\nSubjec...	comp.graphics	have very large file macintosh canvas somethin...	Subject Interesting conversion Problem	CWRU Edu	have very large file macintosh canvas somethin...

In [53]:

```
data1=data.drop(["preprocessed_texts","preprocessed_subjects","preprocessed_emails","class","text"
], axis=1 )
data1.head(4)
```

Out[53]:

	preprocessed_full_text
0	looking for statistics the prevalence disorder...
1	posted talk politics guns from can salyzyn bee...
2	article ida org however hate economic terroris...
3	article article muchado and cpu does not help ...

In [54]:

```
data1.shape
```

Out[54]:

```
(18828, 1)
```

In [55]:

```
from sklearn.model_selection import train_test_split
x_train_data,x_test_data,y_train,y_test=train_test_split(data1,true_labels,test_size=0.25,stratify
=true_labels)
```

In [56]:

```
x_train_data.shape
```

Out[56]:

```
(14121, 1)
```

In [57]:

```
x_test_data.shape
```

Out[57]:

```
(4707, 1)
```

In [58]:

```
len(y_train)
```

Out[58]:

```
14121
```

In [59]:

```
len(y_test)
```

Out[59]:

```
4707
```

In [60]:

```
x_train_data.head(2)
```

Out[60]:

	preprocessed_full_text
18073	has anybody gotten cview work color mode tride...
9240	loftus gues the others not ring the bell but t...

In [61]:

```
print(type(x_train_data))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [62]:

```
x_train_data1=[]
for i in x_train_data.preprocessed_full_text:
    x_train_data1.append(i)
```

In [63]:

```
len(x_train_data1)
```

Out[63]:

```
14121
```

In [64]:

```
x_test_data1=[]
for j in x_test_data.preprocessed_full_text:
    x_test_data1.append(j)
```

In [65]:

```
len(x_test_data1)
```

Out[65]:

```
4707
```

In [66]:

```
len(x_test_data1[0])
```

Out[66]:

```
437
```

In [67]:

```
import tensorflow as tf
```

```
one_hot_y_train=tf.keras.utils.to_categorical(y_train,20)
```

In [68]:

```
one_hot_y_train.shape
```

Out[68]:

```
(14121, 20)
```

In [69]:

```
one_hot_y_test=tf.keras.utils.to_categorical(y_test,20)
```

In [70]:

```
one_hot_y_test.shape
```

Out[70]:

```
(4707, 20)
```

tokenization

In [120]:

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

In [121]:

```
tokenizer=Tokenizer(num_words=40000,filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n',oov_token="<oov>
")  # i removed "_" from the filters so it will consider it as word
tokenizer.fit_on_texts(x_train_data1)
```

In [122]:

```
word_index=tokenizer.word_index
len(word_index)  #total number of words in all documnets
```

Out[122]:

```
75740
```

In [123]:

```
x_train_sequence=tokenizer.texts_to_sequences(x_train_data1)
```

In [124]:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
x_train_padded=pad_sequences(x_train_sequence,maxlen=1500,padding='post')
```

In [125]:

```
x_train_padded.shape
```

Out[125]:

```
(14121, 1500)
```

In [126]:

```
x_test_sequence=tokenizer.texts_to_sequences(x_test_data1)
x_test_padded=pad_sequences(x_test_sequence,maxlen=1500,padding='post')
```

In [127]:

```
x_test_padded.shape
```

Out[127]:

```
(4707, 1500)
```

In [128]:

```
print(type(x_test_padded))
```

```
<class 'numpy.ndarray'>
```

callbacks

for f1 score callback

In [129]:

```
#from callback assignment
from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
class f1_and_auc_score(tf.keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        ## on begin of training, we are creating a instance varible called metrics11
        ## it is a dict with keys [loss, acc, val_loss, val_acc]
        self.metrics11={'f1_score': []}

    def on_epoch_end(self, epoch, logs={}):
        y_pred_final=[]
        y_predict_proba=self.model.predict(x_test_padded)  #it predicts probability
        for i in y_predict_proba:
            y_pred_final.append(np.argmax(i)) #whichever is maximum value thats index is class

        #for j in range(len(y_test1)):
        #    if y_predict_proba[j] <= 0.5:
        #        y_pred_final.append(0)
        #    else:
        #        y_pred_final.append(1)

        self.metrics11["f1_score"].append(f1_score(y_test,y_pred_final,average='micro'))
        #self.metrics11["auc_values"].append( roc_auc_score(y_test1,y_predict_proba))

        print("f1 score is ",self.metrics11["f1_score"][epoch])
```

In [130]:

```
metrics12=f1_and_auc_score()
```

Tensorboard callback

In [131]:

```
%load_ext tensorboard
```

In [132]:

```
!rm -rf ./logs/
import datetime
```

In [133]:

```
import os
logdir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1,write_graph=True,wri
te_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [134]:

```
logdir
```

Out[134]:

```
'logs/20200825-124823'
```

callback for when val_accuracy reaches 75%

In [135]:

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_accuracy') > 0.70):
            print("\nReached %2.2f%% accuracy, so stopping training!!" %(0.70*100))
            self.model.stop_training = True
```

In [136]:

```
val_acc_callback=myCallback()
```

modelcheckpoint callback

In [137]:

```
filepath="/content/best_model_L1h5"
```

In [138]:

```
#https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint
best_model_weight=tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_accuracy', verbose=0, s
ave_best_only=True,save_weights_only=True, mode='auto', save_freq='epoch')
```

creating a model

pretrained glove embedding

In [139]:

```
!wget --header="Host: downloads.cs.stanford.edu" --header="User-Agent: Mozilla/5.0 (Windows NT 10.
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36" --header
="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7" --header
="Cookie: _ga=GA1.2.571392400.1592314058" --header="Connection: keep-alive"
"http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip" -c -O 'glove.6B.zip'
```

```
--2020-08-25 12:48:34--  http://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 416 Requested Range Not Satisfiable
```

```
    The file is already fully retrieved; nothing to do.
```

In [140]:

```
import zipfile
zip_ref = zipfile.ZipFile('/content/glove.6B.zip', 'r')
zip_ref.extractall()
zip_ref.close()
```

In [141]:

```
embeddings_index = dict()
```

In [142]:

```
import numpy as np
f = open('/content/glove.6B.200d.txt') #for every word we will 200 d (features)
for line in f:
 values = line.split()
 word = values[0]
 coefs = np.asarray(values[1:], dtype='float32')
 embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

In [143]:

```
len(embeddings_index)
```

Out[143]:

```
400000
```

In [144]:

```
vocab_size=len(tokenizer.word_index)+1

vocab_size
```

Out[144]:

```
75741
```

In [145]:

```
embedding_matrix = np.zeros((vocab_size,200))
```

In [146]:

```
embedding_matrix.shape
```

Out[146]:

```
(75741, 200)
```

In [147]:

```
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 [148]:

```
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten,Conv1D
,MaxPool1D,concatenate,Embedding
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense,Input
from tensorflow.keras.models import Model
```

In [149]:

```
def model1():

    embedding_layer=Embedding(vocab_size,200,weights=[embedding_matrix], input_length=1500,
trainable=False)

    input_layer=Input(shape=(1500,), dtype='int32')

    embedding_layer1=embedding_layer(input_layer)

    layer3a=Conv1D(filters=12,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regulariz
ers.l2(l2=0.01))(embedding_layer1)

    layer3b=Conv1D(filters=8,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regularize
rs.l2(l2=0.01))(embedding_layer1)

    layer3c=Conv1D(filters=6,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regularize
rs.l2(l2=0.01))(embedding_layer1)

    layer4=tf.keras.layers.Concatenate()([layer3a, layer3b, layer3c])

    layer5=MaxPool1D(pool_size=2)(layer4)

    layer6a=Conv1D(filters=12,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regulariz
ers.l2(l2=0.01))(layer5)

    layer6b=Conv1D(filters=8,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regularize
rs.l2(l2=0.01))(layer5)

    layer6c=Conv1D(filters=6,kernel_size=3,activation='relu',kernel_regularizer=tf.keras.regularize
rs.l2(l2=0.01))(layer5)

    layer7=tf.keras.layers.Concatenate()([layer6a,layer6b,layer6c])

    layer8=MaxPool1D(pool_size=2)(layer7)

    layer9=Conv1D(filters=12,kernel_size=4,activation='relu')(layer8)

    layer10=Flatten()(layer9)

    layer11=Dense(128,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=28)
,kernel_regularizer=tf.keras.regularizers.l2(l2=0.01))(layer10)

    layer12=Dropout(0.3)(layer11)


layer13=Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=29),ker
nel_regularizer=tf.keras.regularizers.l2(l2=0.01))(layer12)


output_layer=Dense(20,activation="softmax",kernel_initializer=tf.keras.initializers.glorot_normal(
seed=3))(layer13)

    model1=Model(input_layer,output_layer ,name="model_1")

    return model1
```

In [150]:

```
model1=model1()
model1.summary()
```

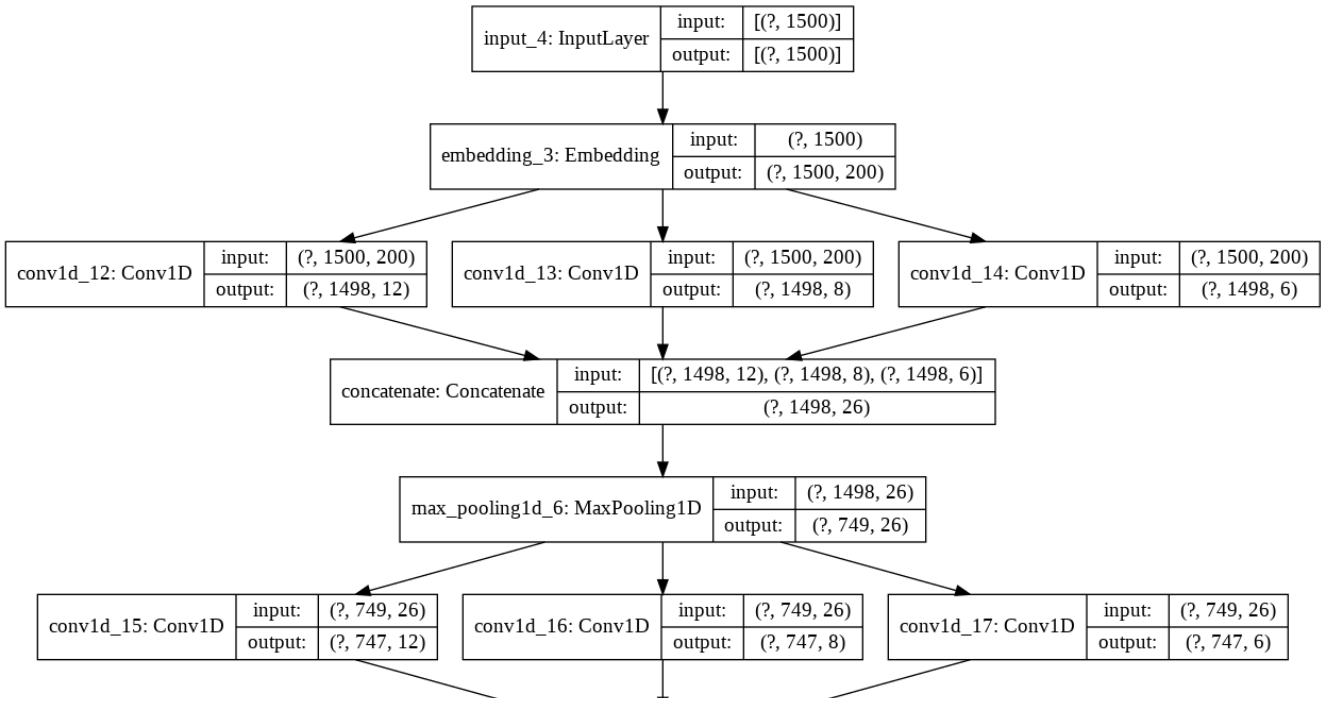
```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_4 (InputLayer)         [(None, 1500)]            0
```

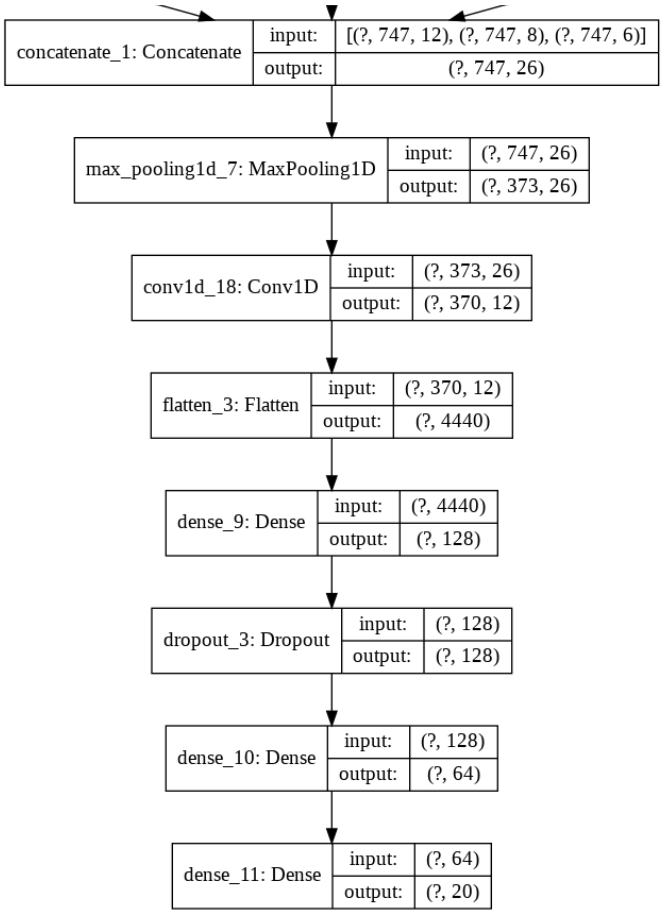
```
embedding_3 (Embedding)         (None, 1500, 200)    15148200    input_4[0][0]
conv1d_12 (Conv1D)              (None, 1498, 12)     7212        embedding_3[0][0]
conv1d_13 (Conv1D)              (None, 1498, 8)      4808        embedding_3[0][0]
conv1d_14 (Conv1D)              (None, 1498, 6)      3606        embedding_3[0][0]
concatenate (Concatenate)       (None, 1498, 26)     0           conv1d_12[0][0]
                                                                 conv1d_13[0][0]
                                                                 conv1d_14[0][0]
max_pooling1d_6 (MaxPooling1D)  (None, 749, 26)      0           concatenate[0][0]
conv1d_15 (Conv1D)              (None, 747, 12)      948         max_pooling1d_6[0][0]
conv1d_16 (Conv1D)              (None, 747, 8)       632         max_pooling1d_6[0][0]
conv1d_17 (Conv1D)              (None, 747, 6)       474         max_pooling1d_6[0][0]
concatenate_1 (Concatenate)     (None, 747, 26)      0           conv1d_15[0][0]
                                                                 conv1d_16[0][0]
                                                                 conv1d_17[0][0]
max_pooling1d_7 (MaxPooling1D)  (None, 373, 26)      0           concatenate_1[0][0]
conv1d_18 (Conv1D)              (None, 370, 12)      1260        max_pooling1d_7[0][0]
flatten_3 (Flatten)             (None, 4440)         0           conv1d_18[0][0]
dense_9 (Dense)                 (None, 128)          568448      flatten_3[0][0]
dropout_3 (Dropout)             (None, 128)          0           dense_9[0][0]
dense_10 (Dense)                (None, 64)           8256        dropout_3[0][0]
dense_11 (Dense)                (None, 20)           1300        dense_10[0][0]
==================================================================================================
Total params: 15,745,144
Trainable params: 596,944
Non-trainable params: 15,148,200
__________________________________________________________________________________________________
```

In [151]:

```
from tensorflow.keras.utils import plot_model
plot_model(model1, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Out[151]:





In [152]:

```
model1.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='categorical_crossentropy',metrics
=['accuracy'])
```

In [153]:

```
tf.keras.backend.clear_session()
```

In [154]:

```
model1.fit(x_train_padded,one_hot_y_train,epochs=100,batch_size=20,validation_data=(x_test_padded,o
ne_hot_y_test),callbacks=[val_acc_callback,metrics12,best_model_weight,tensorboard_callback])
```

```
Epoch 1/100
  1/707 [..............................] - ETA: 0s - loss: 7.7340 - accuracy:
0.0000e+00WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/summary_ops_v2.py:1277: stop (from
tensorflow.python.eager.profiler) is deprecated and will be removed after 2020-07-01.
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
  2/707 [..............................] - ETA: 32s - loss: 7.6375 - accuracy: 0.0500
WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch
time: 0.0249s vs `on_train_batch_end` time: 0.0659s). Check your callbacks.
707/707 [==============================] - ETA: 0s - loss: 2.8582 - accuracy: 0.2296f1 score is  0
.3322710856171659
707/707 [==============================] - 12s 17ms/step - loss: 2.8582 - accuracy: 0.2296 - val_l
oss: 2.1961 - val_accuracy: 0.3323
Epoch 2/100
705/707 [============================>.] - ETA: 0s - loss: 1.9944 - accuracy: 0.4110f1 score is  0
.4518801784576163
707/707 [==============================] - 11s 15ms/step - loss: 1.9942 - accuracy: 0.4110 - val_l
oss: 1.9095 - val_accuracy: 0.4519
Epoch 3/100
707/707 [==============================] - ETA: 0s - loss: 1.8373 - accuracy: 0.4746f1 score is  0
.47185043552156364
707/707 [==============================] - 11s 16ms/step - loss: 1.8373 - accuracy: 0.4746 - val_l
oss: 1.9042 - val_accuracy: 0.4719
Epoch 4/100
```

```
705/707 [============================>.] - ETA: 0s - loss: 1.7475 - accuracy: 0.5214f1 score is  0
.5706394731251327
707/707 [==============================] - 11s 16ms/step - loss: 1.7470 - accuracy: 0.5214 - val_l
oss: 1.6412 - val_accuracy: 0.5706
Epoch 5/100
704/707 [============================>.] - ETA: 0s - loss: 1.6681 - accuracy: 0.5642f1 score is  0
.5419587847886127
707/707 [==============================] - 11s 15ms/step - loss: 1.6677 - accuracy: 0.5642 - val_l
oss: 1.7221 - val_accuracy: 0.5420
Epoch 6/100
703/707 [============================>.] - ETA: 0s - loss: 1.6115 - accuracy: 0.5798f1 score is  0
.591884427448481
707/707 [==============================] - 11s 15ms/step - loss: 1.6117 - accuracy: 0.5796 - val_l
oss: 1.5740 - val_accuracy: 0.5919
Epoch 7/100
707/707 [==============================] - ETA: 0s - loss: 1.5673 - accuracy: 0.5997f1 score is  0
.5963458678563841
707/707 [==============================] - 11s 16ms/step - loss: 1.5673 - accuracy: 0.5997 - val_l
oss: 1.5774 - val_accuracy: 0.5963
Epoch 8/100
705/707 [============================>.] - ETA: 0s - loss: 1.5544 - accuracy: 0.6174f1 score is  0
.6314000424899087
707/707 [==============================] - 11s 16ms/step - loss: 1.5551 - accuracy: 0.6174 - val_l
oss: 1.5273 - val_accuracy: 0.6314
Epoch 9/100
704/707 [============================>.] - ETA: 0s - loss: 1.5072 - accuracy: 0.6259f1 score is  0
.6197153176120671
707/707 [==============================] - 11s 15ms/step - loss: 1.5081 - accuracy: 0.6254 - val_l
oss: 1.6302 - val_accuracy: 0.6197
Epoch 10/100
705/707 [============================>.] - ETA: 0s - loss: 1.4856 - accuracy: 0.6417f1 score is  0
.6305502443169747
707/707 [==============================] - 11s 15ms/step - loss: 1.4854 - accuracy: 0.6415 - val_l
oss: 1.5413 - val_accuracy: 0.6306
Epoch 11/100
702/707 [============================>.] - ETA: 0s - loss: 1.4595 - accuracy: 0.6504f1 score is  0
.6114297854259614
707/707 [==============================] - 11s 15ms/step - loss: 1.4614 - accuracy: 0.6503 - val_l
oss: 1.5930 - val_accuracy: 0.6114
Epoch 12/100
705/707 [============================>.] - ETA: 0s - loss: 1.4725 - accuracy: 0.6548f1 score is  0
.645209262800085
707/707 [==============================] - 11s 16ms/step - loss: 1.4726 - accuracy: 0.6546 - val_l
oss: 1.4992 - val_accuracy: 0.6452
Epoch 13/100
705/707 [============================>.] - ETA: 0s - loss: 1.4119 - accuracy: 0.6646f1 score is  0
.6662417675801997
707/707 [==============================] - 11s 16ms/step - loss: 1.4118 - accuracy: 0.6645 - val_l
oss: 1.4166 - val_accuracy: 0.6662
Epoch 14/100
702/707 [============================>.] - ETA: 0s - loss: 1.4196 - accuracy: 0.6684f1 score is  0
.6596558317399618
707/707 [==============================] - 11s 15ms/step - loss: 1.4199 - accuracy: 0.6681 - val_l
oss: 1.4639 - val_accuracy: 0.6597
Epoch 15/100
707/707 [==============================] - ETA: 0s - loss: 1.3872 - accuracy: 0.6771f1 score is  0
.6165285744635649
707/707 [==============================] - 11s 15ms/step - loss: 1.3872 - accuracy: 0.6771 - val_l
oss: 1.5574 - val_accuracy: 0.6165
Epoch 16/100
706/707 [============================>.] - ETA: 0s - loss: 1.3799 - accuracy: 0.6810f1 score is  0
.6755895474824729
707/707 [==============================] - 11s 16ms/step - loss: 1.3799 - accuracy: 0.6810 - val_l
oss: 1.4155 - val_accuracy: 0.6756
Epoch 17/100
705/707 [============================>.] - ETA: 0s - loss: 1.3823 - accuracy: 0.6820f1 score is  0
.6698534098151689
707/707 [==============================] - 11s 15ms/step - loss: 1.3820 - accuracy: 0.6820 - val_l
oss: 1.4376 - val_accuracy: 0.6699
Epoch 18/100
704/707 [============================>.] - ETA: 0s - loss: 1.3621 - accuracy: 0.6910f1 score is  0
.6579562353940939
707/707 [==============================] - 11s 15ms/step - loss: 1.3616 - accuracy: 0.6912 - val_l
oss: 1.4857 - val_accuracy: 0.6580
Epoch 19/100
706/707 [============================>.] - ETA: 0s - loss: 1.3823 - accuracy: 0.6909f1 score is  0
.6779264924580413
```

```
707/707 [==============================] - 11s 16ms/step - loss: 1.3823 - accuracy: 0.6908 - val_l
oss: 1.4014 - val_accuracy: 0.6779
Epoch 20/100
706/707 [==============================>.] - ETA: 0s - loss: 1.3359 - accuracy: 0.6970f1 score is  0
.6547694922455917
707/707 [==============================] - 11s 16ms/step - loss: 1.3358 - accuracy: 0.6970 - val_l
oss: 1.5033 - val_accuracy: 0.6548
Epoch 21/100
703/707 [==============================>.] - ETA: 0s - loss: 1.3298 - accuracy: 0.6986f1 score is  0
.6462715105162524
707/707 [==============================] - 11s 15ms/step - loss: 1.3299 - accuracy: 0.6985 - val_l
oss: 1.5283 - val_accuracy: 0.6463
Epoch 22/100
706/707 [==============================>.] - ETA: 0s - loss: 1.3739 - accuracy: 0.6914f1 score is  0
.686212024644147
707/707 [==============================] - 11s 16ms/step - loss: 1.3739 - accuracy: 0.6914 - val_l
oss: 1.3824 - val_accuracy: 0.6862
Epoch 23/100
705/707 [==============================>.] - ETA: 0s - loss: 1.3126 - accuracy: 0.7048f1 score is  0
.6634799235181644
707/707 [==============================] - 11s 15ms/step - loss: 1.3126 - accuracy: 0.7048 - val_l
oss: 1.4851 - val_accuracy: 0.6635
Epoch 24/100
703/707 [==============================>.] - ETA: 0s - loss: 1.3128 - accuracy: 0.7042f1 score is  0
.6694285107287019
707/707 [==============================] - 11s 15ms/step - loss: 1.3133 - accuracy: 0.7040 - val_l
oss: 1.4472 - val_accuracy: 0.6694
Epoch 25/100
702/707 [==============================>.] - ETA: 0s - loss: 1.3335 - accuracy: 0.7103f1 score is  0
.6524325472700234
707/707 [==============================] - 11s 15ms/step - loss: 1.3326 - accuracy: 0.7104 - val_l
oss: 1.5218 - val_accuracy: 0.6524
Epoch 26/100
704/707 [==============================>.] - ETA: 0s - loss: 1.2961 - accuracy: 0.7122f1 score is  0
.687061822817081
707/707 [==============================] - 11s 16ms/step - loss: 1.2958 - accuracy: 0.7122 - val_l
oss: 1.4397 - val_accuracy: 0.6871
Epoch 27/100
704/707 [==============================>.] - ETA: 0s - loss: 1.3107 - accuracy: 0.7156f1 score is  0
.668153813469301
707/707 [==============================] - 11s 15ms/step - loss: 1.3106 - accuracy: 0.7155 - val_l
oss: 1.4766 - val_accuracy: 0.6682
Epoch 28/100
705/707 [==============================>.] - ETA: 0s - loss: 1.2952 - accuracy: 0.7124f1 score is  0
.6250265561929041
707/707 [==============================] - 12s 16ms/step - loss: 1.2951 - accuracy: 0.7124 - val_l
oss: 1.6550 - val_accuracy: 0.6250
Epoch 29/100
705/707 [==============================>.] - ETA: 0s - loss: 1.3059 - accuracy: 0.7160f1 score is  0
.6834501805821117
707/707 [==============================] - 11s 16ms/step - loss: 1.3065 - accuracy: 0.7162 - val_l
oss: 1.4657 - val_accuracy: 0.6835
Epoch 30/100
704/707 [==============================>.] - ETA: 0s - loss: 1.2793 - accuracy: 0.7237f1 score is  0
.6524325472700234
707/707 [==============================] - 11s 15ms/step - loss: 1.2794 - accuracy: 0.7235 - val_l
oss: 1.5374 - val_accuracy: 0.6524
Epoch 31/100
705/707 [==============================>.] - ETA: 0s - loss: 1.2964 - accuracy: 0.7144f1 score is  0
.6874867219035479
707/707 [==============================] - 11s 16ms/step - loss: 1.2972 - accuracy: 0.7144 - val_l
oss: 1.4229 - val_accuracy: 0.6875
Epoch 32/100
704/707 [==============================>.] - ETA: 0s - loss: 1.2729 - accuracy: 0.7225f1 score is  0
.6577437858508605
707/707 [==============================] - 11s 15ms/step - loss: 1.2727 - accuracy: 0.7224 - val_l
oss: 1.4664 - val_accuracy: 0.6577
Epoch 33/100
703/707 [==============================>.] - ETA: 0s - loss: 1.2787 - accuracy: 0.7252f1 score is  0
.6817505842362439
707/707 [==============================] - 11s 15ms/step - loss: 1.2783 - accuracy: 0.7252 - val_l
oss: 1.3850 - val_accuracy: 0.6818
Epoch 34/100
703/707 [==============================>.] - ETA: 0s - loss: 1.2851 - accuracy: 0.7199f1 score is  0
.6987465476949225
707/707 [==============================] - 11s 16ms/step - loss: 1.2844 - accuracy: 0.7203 - val_l
oss: 1.3885 - val accuracy: 0.6987
```

```
Epoch 35/100
703/707 [============================>.] - ETA: 0s - loss: 1.2686 - accuracy: 0.7250
Reached 70.00% accuracy, so stopping training!!
f1 score is  0.7000212449543234
707/707 [==============================] - 11s 16ms/step - loss: 1.2683 - accuracy: 0.7251 - val_l
oss: 1.3817 - val_accuracy: 0.7000
```

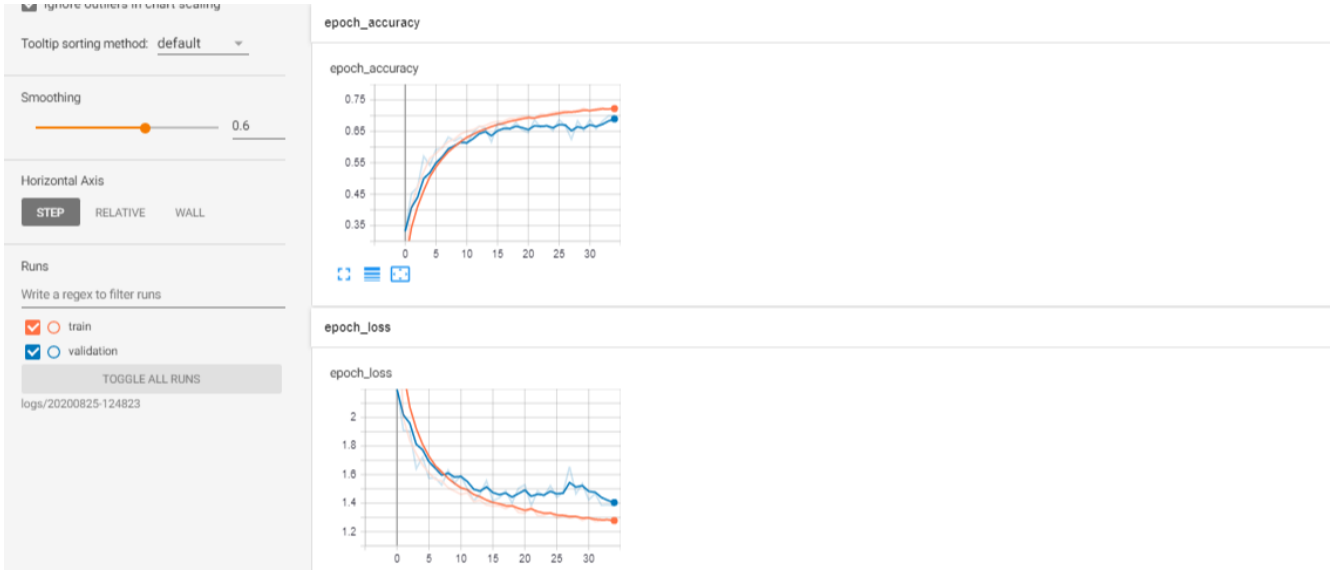
Out[154]:

```
<tensorflow.python.keras.callbacks.History at 0x7fe4e809fa58>
```

In [185]:

```
from IPython.display import Image
Image(filename=r'/content/model22_1.png')
```

Out[185]:



In [ ]:

Model-2 : Using 1D convolutions with character embedding

callback for f1 score for model2

In [155]:

```
#from callback assignment
from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
class f1_and_auc_score_model2(tf.keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        ## on begin of training, we are creating a instance varible called metrics11
        ## it is a dict with keys [loss, acc, val_loss, val_acc]
        self.metrics11={'f1_score': []}

    def on_epoch_end(self, epoch, logs={}):
        y_pred_final=[]
        y_predict_proba=self.model.predict(x_test_char_padded)  #it predicts probability
        for i in y_predict_proba:
            y_pred_final.append(np.argmax(i)) #whichever is maximum value thats index is class

        #for j in range(len(y_test1)):
        #    if y_predict_proba[j] <= 0.5:
        #        y_pred_final.append(0)
        #    else:
        #        y_pred_final.append(1)
```

```
        self.metrics11["f1_score"].append(f1_score(y_test,y_pred_final,average='micro'))
        #self.metrics11["auc_values"].append( roc_auc_score(y_test1,y_predict_proba))

        print("f1 score is ",self.metrics11["f1_score"][epoch])
```

In [156]:

```
metrics123=f1_and_auc_score_model2()
```

stops epochs when vall accuracy reaches more than 12%

In [157]:

```
class myCallback_model2(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_accuracy') > 0.12):
            print("\nReached %2.2f%% accuracy, so stopping training!!" %(0.12*100))
            self.model.stop_training = True
```

In [158]:

```
val_acc_callback_model2=myCallback_model2()
```

tensorboard callback

In [159]:

```
import os
logdir2= os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback2= tf.keras.callbacks.TensorBoard(logdir2, histogram_freq=1,write_graph=True,wr
ite_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
```

In [160]:

```
logdir2
```

Out[160]:

```
'logs/20200825-125651'
```

modelcheckpoint callback

In [161]:

```
filepath2="/content/best_model_L2h5"
```

In [162]:

```
#https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint
best_model_weight2=tf.keras.callbacks.ModelCheckpoint(filepath2, monitor='val_accuracy', verbose=0
, save_best_only=True,save_weights_only=True, mode='auto', save_freq='epoch')
```

Tokenization

In [163]:

```
t=Tokenizer(num_words=None,char_level=True,filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n',oov_token
="<oov1>") #removed underscore( ) from filters because we want it as charcter
```

```
t.fit_on_texts(x_train_data1)
```

In [164]:

```
char_index=t.word_index
len(char_index)
```

Out[164]:

```
39
```

In [165]:

```
print(char_index)
```

```
{'<oov1>': 1, ' ': 2, 'e': 3, 't': 4, 'a': 5, 'o': 6, 'n': 7, 'i': 8, 's': 9, 'r': 10, 'h': 11, 'l
': 12, 'd': 13, 'c': 14, 'u': 15, 'm': 16, 'p': 17, 'g': 18, 'w': 19, 'y': 20, 'f': 21, 'b': 22, '
v': 23, 'k': 24, 'x': 25, 'j': 26, 'q': 27, 'z': 28, '_': 29, '1': 30, '0': 31, '2': 32, '3': 33,
'4': 34, '6': 35, '5': 36, '8': 37, '9': 38, '7': 39}
```

In [166]:

```
adde=[]
for i in x_train_data1:
    adde.append(len(i))
```

In [167]:

```
aq1=sum(adde)/len(adde)
aq1
```

Out[167]:

```
1169.6768642447419
```

In [168]:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
x_train_char_sequence=t.texts_to_sequences(x_train_data1)
x_train_char_padded=pad_sequences(x_train_char_sequence,maxlen=1200,padding='post')
```

In [169]:

```
x_train_char_padded.shape
```

Out[169]:

```
(14121, 1200)
```

In [170]:

```
x_test_char_sequence=t.texts_to_sequences(x_test_data1)
x_test_char_padded=pad_sequences(x_test_char_sequence,maxlen=1200,padding='post')
```

In [171]:

```
x_test_char_padded.shape
```

Out[171]:

```
(4707, 1200)
```

character level glove embeddings

In [172]:

```
embeddings_index1 = dict()
```

In [173]:

```
import numpy as np
f = open("/content/glove.840B.300d-char.txt") # for every word we will 300 d (features)
for line in f:
 values = line.split()
 word = values[0]
 coefs = np.asarray(values[1:], dtype='float32')
 embeddings_index1[word] = coefs
f.close()
print('Loaded %s charcter vectors.' % len(embeddings_index1))
```

```
Loaded 94 charcter vectors.
```

In [174]:

```
vocab_size1=len(t.word_index)+1
vocab_size1
```

Out[174]:

```
40
```

In [175]:

```
embedding_matrix1 = np.zeros((vocab_size1,300))
```

In [176]:

```
for char, i in t.word_index.items():
 embedding_vector1 = embeddings_index1.get(char)
 if embedding_vector1 is not None:
  embedding_matrix1[i] = embedding_vector1
```

In [177]:

```
embedding_matrix1.shape
```

Out[177]:

```
(40, 300)
```

In [178]:

```
def model2():

    embedding_layer23=Embedding(vocab_size1,300, input_length=1200,weights=[embedding_matrix1],
trainable=False)

    input_layer11=Input(shape=(1200,), dtype='int32')

    embedded_sequences=embedding_layer23(input_layer11)

    layer11=Conv1D(filters=32,kernel_size=3,activation='relu')(embedded_sequences)

    layer12=Conv1D(filters=16,kernel_size=3,activation='relu')(layer11)

    layer13=MaxPool1D(pool_size=3)(layer12)

    layer14=Conv1D(filters=16,kernel_size=3,activation='relu')(layer13)

    layer15=Conv1D(filters=8,kernel_size=3,activation='relu')(layer14)

    layer16=MaxPool1D(pool_size=2)(layer15)

    layer17=Flatten()(layer16)
```

```
    layer18=Dropout(0.5)(layer17)

    layer19=Dense(512,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=28)
,kernel_regularizer=tf.keras.regularizers.l2(l2=0.01))(layer18)

    layer20=Dense(256,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=28)
,kernel_regularizer=tf.keras.regularizers.l2(l2=0.01))(layer19)


output_layer1=Dense(20,activation="softmax",kernel_initializer=tf.keras.initializers.glorot_normal
(seed=3))(layer20)

    model2=Model(input_layer11,output_layer1,name="model_2")

    return model2
```

In [179]:

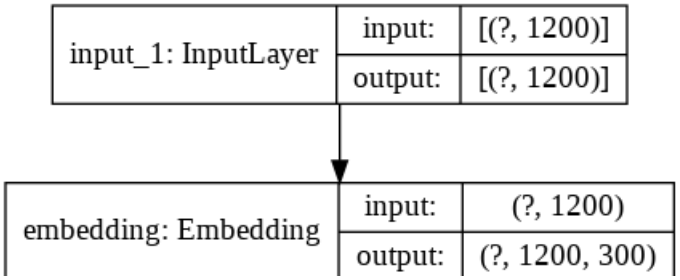
```
model2=model2()
model2.summary()
```

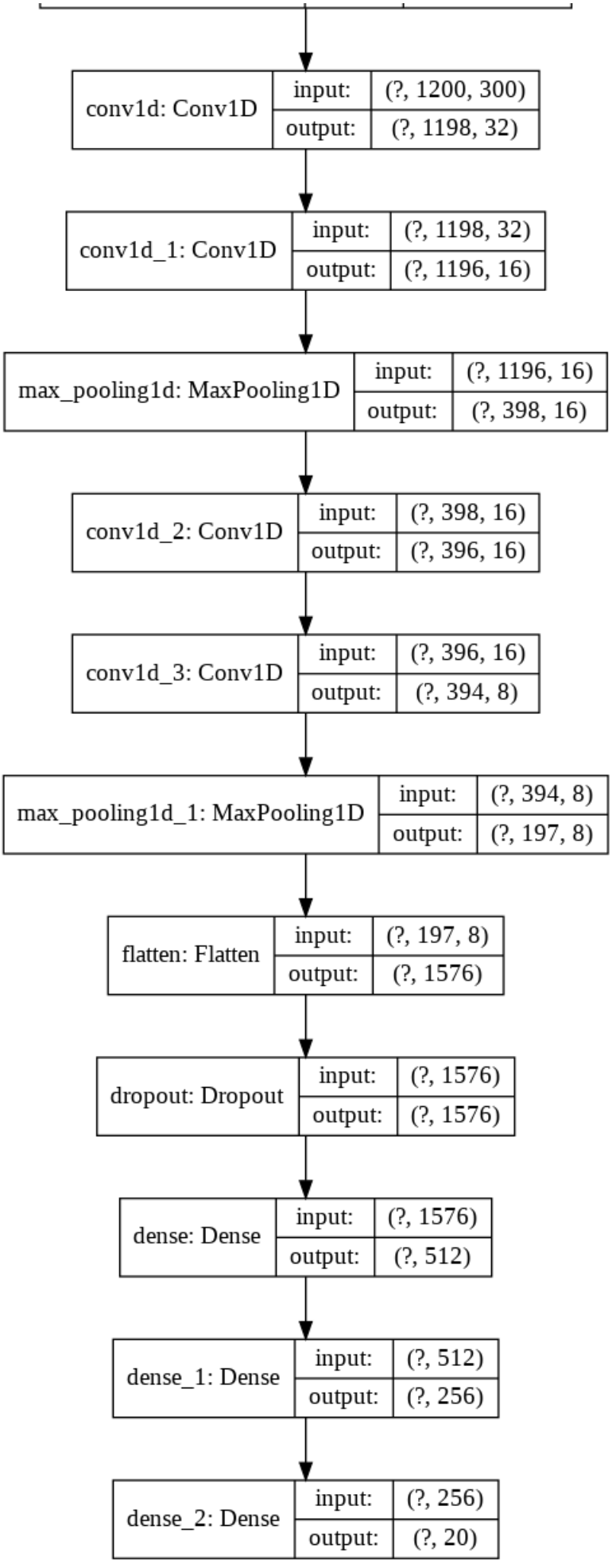
```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1200)]            0
_________________________________________________________________
embedding (Embedding)        (None, 1200, 300)         12000
_________________________________________________________________
conv1d (Conv1D)              (None, 1198, 32)          28832
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 1196, 16)          1552
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 398, 16)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 396, 16)           784
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 394, 8)            392
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 197, 8)            0
_________________________________________________________________
flatten (Flatten)            (None, 1576)              0
_________________________________________________________________
dropout (Dropout)            (None, 1576)              0
_________________________________________________________________
dense (Dense)                (None, 512)               807424
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 20)                5140
=================================================================
Total params: 987,452
Trainable params: 975,452
Non-trainable params: 12,000
_________________________________________________________________
```

In [180]:

```
from tensorflow.keras.utils import plot_model
plot_model(model2, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Out[180]:





In [277]:

```
tf.keras.backend.clear_session()
```

In [181]:

```
model2.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),loss='categorical_crossentropy',metrics
=['accuracy'])
```

In [182]:

```
model2.fit(x_train_char_padded,one_hot_y_train,epochs=100,batch_size=100,validation_data=(x_test_ch
ar_padded,one_hot_y_test),callbacks=[metrics123,val_acc_callback_model2,tensorboard_callback2,best
_model_weight2])
```

```
Epoch 1/100
  2/142 [..............................] - ETA: 9s - loss: 18.1126 - accuracy:
0.0750WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0355s vs `on_train_batch_end` time: 0.1044s). Check your callbacks.
140/142 [============================>.] - ETA: 0s - loss: 5.7559 - accuracy: 0.0725f1 score is  0
.07478223921818568
142/142 [==============================] - 8s 59ms/step - loss: 5.7329 - accuracy: 0.0726 - val_lo
ss: 3.0447 - val_accuracy: 0.0748
Epoch 2/100
141/142 [============================>.] - ETA: 0s - loss: 2.9742 - accuracy: 0.0823f1 score is  0
.07329509241555131
142/142 [==============================] - 8s 55ms/step - loss: 2.9741 - accuracy: 0.0826 - val_lo
ss: 2.9562 - val_accuracy: 0.0733
Epoch 3/100
141/142 [============================>.] - ETA: 0s - loss: 2.9472 - accuracy: 0.0869f1 score is  0
.07648183556405354
142/142 [==============================] - 8s 55ms/step - loss: 2.9472 - accuracy: 0.0869 - val_lo
ss: 2.9496 - val_accuracy: 0.0765
Epoch 4/100
141/142 [============================>.] - ETA: 0s - loss: 2.9426 - accuracy: 0.0838f1 score is  0
.07669428510728701
142/142 [==============================] - 8s 55ms/step - loss: 2.9426 - accuracy: 0.0837 - val_lo
ss: 2.9497 - val_accuracy: 0.0767
Epoch 5/100
141/142 [============================>.] - ETA: 0s - loss: 2.9417 - accuracy: 0.0823f1 score is  0
.07945612916932229
142/142 [==============================] - 8s 55ms/step - loss: 2.9418 - accuracy: 0.0824 - val_lo
ss: 2.9431 - val_accuracy: 0.0795
Epoch 6/100
141/142 [============================>.] - ETA: 0s - loss: 2.9398 - accuracy: 0.0862f1 score is  0
.07541958784788613
142/142 [==============================] - 8s 55ms/step - loss: 2.9397 - accuracy: 0.0863 - val_lo
ss: 2.9529 - val_accuracy: 0.0754
Epoch 7/100
141/142 [============================>.] - ETA: 0s - loss: 2.9453 - accuracy: 0.0879f1 score is  0
.07839388145315487
142/142 [==============================] - 8s 55ms/step - loss: 2.9454 - accuracy: 0.0878 - val_lo
ss: 2.9583 - val_accuracy: 0.0784
Epoch 8/100
141/142 [============================>.] - ETA: 0s - loss: 2.9618 - accuracy: 0.0913f1 score is  0
.08179307414489059
142/142 [==============================] - 8s 55ms/step - loss: 2.9618 - accuracy: 0.0914 - val_lo
ss: 2.9712 - val_accuracy: 0.0818
Epoch 9/100
141/142 [============================>.] - ETA: 0s - loss: 2.9530 - accuracy: 0.0927f1 score is  0
.09007860633099637
142/142 [==============================] - 8s 55ms/step - loss: 2.9530 - accuracy: 0.0927 - val_lo
ss: 2.9534 - val_accuracy: 0.0901
Epoch 10/100
141/142 [============================>.] - ETA: 0s - loss: 2.9464 - accuracy: 0.0928f1 score is  0
.09071595496069683
142/142 [==============================] - 8s 55ms/step - loss: 2.9462 - accuracy: 0.0928 - val_lo
ss: 2.9456 - val_accuracy: 0.0907
Epoch 11/100
141/142 [============================>.] - ETA: 0s - loss: 2.9402 - accuracy: 0.0974f1 score is  0
.09156575313363076
142/142 [==============================] - 8s 55ms/step - loss: 2.9403 - accuracy: 0.0974 - val_lo
ss: 2.9492 - val_accuracy: 0.0916
Epoch 12/100
141/142 [============================>.] - ETA: 0s - loss: 2.9390 - accuracy: 0.0983f1 score is  0
.09135330359039727
142/142 [==============================] - 8s 55ms/step - loss: 2.9390 - accuracy: 0.0982 - val_lo
```

```
ss: 2.9435 - val_accuracy: 0.0914
Epoch 13/100
141/142 [============================>.] - ETA: 0s - loss: 2.9329 - accuracy: 0.0977f1 score is  0
.08965370724452942
142/142 [==============================] - 8s 55ms/step - loss: 2.9326 - accuracy: 0.0977 - val_lo
ss: 2.9383 - val_accuracy: 0.0897
Epoch 14/100
141/142 [============================>.] - ETA: 0s - loss: 2.9278 - accuracy: 0.1012f1 score is  0
.08986615678776291
142/142 [==============================] - 8s 55ms/step - loss: 2.9279 - accuracy: 0.1012 - val_lo
ss: 2.9430 - val_accuracy: 0.0899
Epoch 15/100
141/142 [============================>.] - ETA: 0s - loss: 2.9190 - accuracy: 0.0989f1 score is  0
.09071595496069683
142/142 [==============================] - 8s 55ms/step - loss: 2.9190 - accuracy: 0.0989 - val_lo
ss: 2.9329 - val_accuracy: 0.0907
Epoch 16/100
141/142 [============================>.] - ETA: 0s - loss: 2.9176 - accuracy: 0.0995f1 score is  0
.08986615678776291
142/142 [==============================] - 8s 55ms/step - loss: 2.9179 - accuracy: 0.0995 - val_lo
ss: 2.9360 - val_accuracy: 0.0899
Epoch 17/100
141/142 [============================>.] - ETA: 0s - loss: 2.9242 - accuracy: 0.1002f1 score is  0
.09262800084979816
142/142 [==============================] - 8s 55ms/step - loss: 2.9242 - accuracy: 0.1002 - val_lo
ss: 2.9311 - val_accuracy: 0.0926
Epoch 18/100
141/142 [============================>.] - ETA: 0s - loss: 2.9168 - accuracy: 0.1015f1 score is  0
.09029105587422986
142/142 [==============================] - 8s 55ms/step - loss: 2.9168 - accuracy: 0.1015 - val_lo
ss: 2.9404 - val_accuracy: 0.0903
Epoch 19/100
141/142 [============================>.] - ETA: 0s - loss: 2.9182 - accuracy: 0.1009f1 score is  0
.09092840450393032
142/142 [==============================] - 8s 55ms/step - loss: 2.9183 - accuracy: 0.1009 - val_lo
ss: 2.9268 - val_accuracy: 0.0909
Epoch 20/100
141/142 [============================>.] - ETA: 0s - loss: 2.9125 - accuracy: 0.0989f1 score is  0
.09347779902273211
142/142 [==============================] - 8s 55ms/step - loss: 2.9127 - accuracy: 0.0988 - val_lo
ss: 2.9309 - val_accuracy: 0.0935
Epoch 21/100
141/142 [============================>.] - ETA: 0s - loss: 2.9119 - accuracy: 0.1005f1 score is  0
.09687699171446781
142/142 [==============================] - 8s 55ms/step - loss: 2.9118 - accuracy: 0.1006 - val_lo
ss: 2.9324 - val_accuracy: 0.0969
Epoch 22/100
141/142 [============================>.] - ETA: 0s - loss: 2.9108 - accuracy: 0.1025f1 score is  0
.09454004673889951
142/142 [==============================] - 8s 56ms/step - loss: 2.9108 - accuracy: 0.1023 - val_lo
ss: 2.9229 - val_accuracy: 0.0945
Epoch 23/100
141/142 [============================>.] - ETA: 0s - loss: 2.9115 - accuracy: 0.0984f1 score is  0
.0922031017633312
142/142 [==============================] - 8s 55ms/step - loss: 2.9115 - accuracy: 0.0984 - val_lo
ss: 2.9323 - val_accuracy: 0.0922
Epoch 24/100
141/142 [============================>.] - ETA: 0s - loss: 2.9097 - accuracy: 0.1013f1 score is  0
.09284045039303165
142/142 [==============================] - 8s 54ms/step - loss: 2.9097 - accuracy: 0.1013 - val_lo
ss: 2.9393 - val_accuracy: 0.0928
Epoch 25/100
141/142 [============================>.] - ETA: 0s - loss: 2.9171 - accuracy: 0.0985f1 score is  0
.09305289993626514
142/142 [==============================] - 8s 55ms/step - loss: 2.9170 - accuracy: 0.0986 - val_lo
ss: 2.9523 - val_accuracy: 0.0931
Epoch 26/100
141/142 [============================>.] - ETA: 0s - loss: 2.9210 - accuracy: 0.0999f1 score is  0
.09241555130656469
142/142 [==============================] - 8s 55ms/step - loss: 2.9211 - accuracy: 0.1000 - val_lo
ss: 2.9384 - val_accuracy: 0.0924
Epoch 27/100
141/142 [============================>.] - ETA: 0s - loss: 2.9128 - accuracy: 0.1019f1 score is  0
.09517739536859995
142/142 [==============================] - 8s 54ms/step - loss: 2.9130 - accuracy: 0.1018 - val_lo
ss: 2.9270 - val_accuracy: 0.0952
Epoch 28/100
```

```
141/142 [============================>.] - ETA: 0s - loss: 2.9061 - accuracy: 0.0992f1 score is  0
.09071595496069683
142/142 [==============================] - 8s 55ms/step - loss: 2.9060 - accuracy: 0.0992 - val_lo
ss: 2.9407 - val_accuracy: 0.0907
Epoch 29/100
141/142 [============================>.] - ETA: 0s - loss: 2.9177 - accuracy: 0.1014f1 score is  0
.09454004673889951
142/142 [==============================] - 8s 55ms/step - loss: 2.9179 - accuracy: 0.1014 - val_lo
ss: 2.9319 - val_accuracy: 0.0945
Epoch 30/100
141/142 [============================>.] - ETA: 0s - loss: 2.9128 - accuracy: 0.1013f1 score is  0
.0932653494794986
142/142 [==============================] - 8s 55ms/step - loss: 2.9128 - accuracy: 0.1013 - val_lo
ss: 2.9327 - val_accuracy: 0.0933
Epoch 31/100
141/142 [============================>.] - ETA: 0s - loss: 2.9112 - accuracy: 0.0996f1 score is  0
.09454004673889951
142/142 [==============================] - 8s 55ms/step - loss: 2.9112 - accuracy: 0.0997 - val_lo
ss: 2.9289 - val_accuracy: 0.0945
Epoch 32/100
141/142 [============================>.] - ETA: 0s - loss: 2.9129 - accuracy: 0.1010f1 score is  0
.0932653494794986
142/142 [==============================] - 8s 55ms/step - loss: 2.9132 - accuracy: 0.1008 - val_lo
ss: 2.9446 - val_accuracy: 0.0933
Epoch 33/100
141/142 [============================>.] - ETA: 0s - loss: 2.9178 - accuracy: 0.1050f1 score is  0
.09347779902273211
142/142 [==============================] - 8s 54ms/step - loss: 2.9179 - accuracy: 0.1052 - val_lo
ss: 2.9353 - val_accuracy: 0.0935
Epoch 34/100
141/142 [============================>.] - ETA: 0s - loss: 2.9132 - accuracy: 0.1020f1 score is  0
.09857658806033567
142/142 [==============================] - 8s 55ms/step - loss: 2.9131 - accuracy: 0.1020 - val_lo
ss: 2.9331 - val_accuracy: 0.0986
Epoch 35/100
141/142 [============================>.] - ETA: 0s - loss: 2.9090 - accuracy: 0.1059f1 score is  0
.09900148714680264
142/142 [==============================] - 8s 55ms/step - loss: 2.9087 - accuracy: 0.1059 - val_lo
ss: 2.9287 - val_accuracy: 0.0990
Epoch 36/100
141/142 [============================>.] - ETA: 0s - loss: 2.9082 - accuracy: 0.1040f1 score is  0
.09836413851710218
142/142 [==============================] - 8s 54ms/step - loss: 2.9081 - accuracy: 0.1041 - val_lo
ss: 2.9320 - val_accuracy: 0.0984
Epoch 37/100
141/142 [============================>.] - ETA: 0s - loss: 2.9178 - accuracy: 0.1056f1 score is  0
.09369024856596558
142/142 [==============================] - 8s 55ms/step - loss: 2.9180 - accuracy: 0.1054 - val_lo
ss: 2.9465 - val_accuracy: 0.0937
Epoch 38/100
141/142 [============================>.] - ETA: 0s - loss: 2.9224 - accuracy: 0.1061f1 score is  0
.09623964308476737
142/142 [==============================] - 8s 55ms/step - loss: 2.9222 - accuracy: 0.1063 - val_lo
ss: 2.9253 - val_accuracy: 0.0962
Epoch 39/100
141/142 [============================>.] - ETA: 0s - loss: 2.9017 - accuracy: 0.1067f1 score is  0
.10919906522200977
142/142 [==============================] - 8s 55ms/step - loss: 2.9018 - accuracy: 0.1066 - val_lo
ss: 2.9049 - val_accuracy: 0.1092
Epoch 40/100
141/142 [============================>.] - ETA: 0s - loss: 2.8799 - accuracy: 0.1139f1 score is  0
.10686212024644147
142/142 [==============================] - 8s 55ms/step - loss: 2.8800 - accuracy: 0.1139 - val_lo
ss: 2.9043 - val_accuracy: 0.1069
Epoch 41/100
141/142 [============================>.] - ETA: 0s - loss: 2.8659 - accuracy: 0.1182f1 score is  0
.11387295517314638
142/142 [==============================] - 8s 55ms/step - loss: 2.8657 - accuracy: 0.1183 - val_lo
ss: 2.8762 - val_accuracy: 0.1139
Epoch 42/100
141/142 [============================>.] - ETA: 0s - loss: 2.8529 - accuracy: 0.1206f1 score is  0
.11557255151901423
142/142 [==============================] - 8s 55ms/step - loss: 2.8530 - accuracy: 0.1205 - val_lo
ss: 2.8674 - val_accuracy: 0.1156
Epoch 43/100
141/142 [============================>.] - ETA: 0s - loss: 2.8413 - accuracy: 0.1238f1 score is  0
.11302315700021245
```

```
142/142 [==============================] - 8s 55ms/step - loss: 2.8410 - accuracy: 0.1239 - val_lo
ss: 2.8669 - val_accuracy: 0.1130
Epoch 44/100
141/142 [============================>.] - ETA: 0s - loss: 2.8493 - accuracy: 0.1265f1 score is  0
.11727214786488209
142/142 [==============================] - 8s 55ms/step - loss: 2.8496 - accuracy: 0.1265 - val_lo
ss: 2.8693 - val_accuracy: 0.1173
Epoch 45/100
141/142 [============================>.] - ETA: 0s - loss: 2.8389 - accuracy: 0.1299f1 score is  0
.11939664329721691
142/142 [==============================] - 8s 55ms/step - loss: 2.8390 - accuracy: 0.1299 - val_lo
ss: 2.8632 - val_accuracy: 0.1194
Epoch 46/100
141/142 [============================>.] - ETA: 0s - loss: 2.8355 - accuracy: 0.1236f1 score is  0
.12152113872955173

Reached 12.00% accuracy, so stopping training!!
142/142 [==============================] - 8s 55ms/step - loss: 2.8355 - accuracy: 0.1236 - val_lo
ss: 2.8535 - val_accuracy: 0.1215
```

Out[182]:

```
<tensorflow.python.keras.callbacks.History at 0x7fe82823d5c0>
```

In [186]:

```
from IPython.display import Image
Image(filename=r'/content/model2.png')
```

Out[186]:

conclusion

- character level word embedding based model have less accuracy because firstly you will have much longer sentences as compared to word level embedding model
- so charcater language models are not good as compared to word language model at capturing long range dependancies between how the earlier parts of the sentence also affect the later part of the sentence
- charcter level models are also just more computationally expensive to train
- here i choose maxlen=1200 becuase i take average of len of each document(characters) i found that 1200 is good value

In [ ]: