Text Classification:

Data

```
1. we have total of 20 types of documents(Text files) and total 18828 documents(text files).
2. You can download data from this link (https://drive.google.com/open?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM), in that you wil
l get documents.rar folder.
If you unzip that, you will get total of 18828 documnets. document name is defined as'ClassLabel_DocumentNumberInThatLabel'.
so from document name, you can extract the label for that document.
4. Now our problem is to classify all the documents into any one of the class.
5. Below we provided count plot of all the labels in our data.
```

Assignment:

sample document

```
Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>
>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now.  And there is no "alternative", but the point
>is, "rationality" isn't an alternative either.  The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them.

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--
Have you washed your brain today?
```

In [2]:

```python
import pandas as pd
import numpy as np
import re

from sklearn.preprocessing import label_binarize


import pdb
```

In [3]:

```python
# # extracting zip file (in current director)

# from zipfile import ZipFile
# with ZipFile('documents2.zip', 'r') as zipObj:
#    # Extract all the contents of zip file in current directory
#    zipObj.extractall()
```

loading files

In [4]:
```
# reading file_name from folder

import rarfile
rf = rarfile.RarFile('documents.rar' )

rf.namelist()[:2]
```

Out[4]:
```
['documents/alt.atheism_49960.txt', 'documents/alt.atheism_51060.txt']
```

In [5]:
```
# reading file from folder "documents"

file = []
for f in rf.namelist():
    # f is filename we directly opening file fron 'documents' folder
    try:
        e = open(f, 'r')
#         print(f)
        file.append(e.read())
    except:
        print("error on loading all file")
        print(len(file))

```

```
error on loading all file
18828
```

Preprocessing:

useful links: http://www.pyregex.com/ (http://www.pyregex.com/)

1. Find all emails in the document and then get the text after the "@". and then split those texts by '.'
after that remove the words whose length is less than or equal to 2 and also remove'com' word and then combine those words by space.
In one doc, if we have 2 or more mails, get all.
Eg:[test@dm1.d.com, test2@dm2.dm3.com]-->[dm1.d.com, dm3.dm4.com]-->[dm1,d,com,dm2,dm3,com]-->[dm1,dm2,dm3]-->"dm1 dm2 dm3"
append all those into one list/array. (This will give length of 18828 sentences i.e one list for each of the document).
Some sample output was shown below.

> In the above sample document there are emails [jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mangoe@cs.umd.edu]

preprocessing:
[jcopelan@nyx.cs.du.edu, 65882@mimsy.umd.edu, mangoe@cs.umd.edu] ==> [nyx cs du edu mimsy umd edu cs umd edu] ==>
[nyx edu mimsy umd edu umd edu]

2. Replace all the emails by space in the original text.

In [6]:

```
text1 = """Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>
>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now.  And there is no "alternative", but the point
>is, "rationality" isn't an alternative either.  The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them. new york

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--
Have you washed your brain today?"""
print(len(text1))
```

```
621
```

instruction

3. Get subject of the text i.e. get the total lines where "Subject:" occur and remove
the word which are before the ":" remove the newlines, tabs, punctuations, any special chars.
Eg: if we have sentance like "Subject: Re: Gospel Dating @ \r\r\n" --> You have to get "Gospel Dating"
Save all this data into another list/array.

4. After you store it in the list, Replace those sentances in original text by space.

5. Delete all the sentances where sentence starts with **"Write to:"** or **"From:"**.
> In the above sample document check the 2nd line, we should remove that

6. Delete all the tags like "< anyword >"
> In the above sample document check the 4nd line, we should remove that "< 65882@mimsy.umd.edu >"

7. Delete all the data which are present in the brackets.
In many text data, we observed that, they maintained the explanation of sentence
or translation of sentence to another language in brackets so remove all those.
Eg: "AAIC-The course that gets you HIRED(AAIC - Der Kurs, der Sie anstellt)" --> "AAIC-The course that gets you HIRED"

> In the above sample document check the 4nd line, we should remove that "(Charley Wingate)"

8. Remove all the newlines('\n'), tabs('\t'), "-", "\".

9. Remove all the words which ends with **":"**.
Eg: "Anyword:"
> In the above sample document check the 4nd line, we should remove that "writes:"

10. Decontractions, replace words like below to full words.
please check the donors choose preprocessing for this
Eg: can't -> can not, 's -> is, i've -> i have, i'm -> i am, you're -> you are, i'll --> i will

There is no order to do point 6 to 10. but you have to get final output correctly

11. Do chunking on the text you have after above preprocessing.

Text chunking, also referred to as shallow parsing, is a task that
follows Part-Of-Speech Tagging and that adds more structure to the sentence.
So it combines the some phrases, named entities into single word.
So after that combine all those phrases/named entities by separating "_".
And remove the phrases/named entities if that is a "Person".
You can use **nltk.ne_chunk** to get these.
Below we have given one example. please go through it.

useful links:
https://www.nltk.org/book/ch07.html (https://www.nltk.org/book/ch07.html)
https://stackoverflow.com/a/31837224/4084039 (https://stackoverflow.com/a/31837224/4084039)
http://www.nltk.org/howto/tree.html (http://www.nltk.org/howto/tree.html)
https://stackoverflow.com/a/44294377/4084039 (https://stackoverflow.com/a/44294377/4084039)

preprocessing

In [7]:
```
# 3., .4  - extracting email info
def email_extractor(text):
    l1,l2,l3 =[],[],[];

    email = re.findall(r"[A-Za-z0-9_%+-.]+"
                     r"@[A-Za-z0-9.-]+"
                     r"\.[A-Za-z]{2,5}",text)
    for x in email:
        lst = x.split(sep ="@")
        l1.append(lst[1])

    for x in l1:

        lst = re.sub('\.',',', x)
        l2.append(lst)

    for x in l2:
        lst = x.split(sep=',')
        lst = [x  for x in lst if (len(x)>2 and x not in ["edu","com"] )]
        str = " ".join(lst)
        l3.append(str)

    email_info = " ".join(l3)

    return email_info


r = email_extractor(file[1])
r
```

Out[7]: 'mantis mantis mantis'

In [8]:
```
file[1]
```

Out[8]:
```
'From: mathew <mathew@mantis.co.uk>\nSubject: Alt.Atheism FAQ: Introduction to Atheism\n\nArchive-name: atheism/introduction\nAlt-
atheism-archive-name: introduction\nLast-modified: 5 April 1993\nVersion: 1.2\n\n-----BEGIN PGP SIGNED MESSAGE-----\n\n
An Introduction to Atheism\n                        by mathew <mathew@mantis.co.uk>\n\nThis article attempts to provide a general i
ntroduction to atheism.  Whilst I\nhave tried to be as neutral as possible regarding contentious issues, you\nshould always rememb
er that this document represents only one viewpoint.  I\nwould encourage you to read widely and draw your own conclusions; some\nr
elevant books are listed in a companion article.\n\nTo provide a sense of cohesion and progression, I have presented this article
\nas an imaginary conversation between an atheist and a theist.  All the\nquestions asked by the imaginary theist are questions wh
ich have been cropped\nup repeatedly on alt.atheism since the newsgroup was created.  Some other\nfrequently asked questions are a
nswered in a companion article.\n\nPlease note that this article is arguably slanted towards answering questions\nposed from a Chr
istian viewpoint.  This is because the FAQ files reflect\nquestions which have actually been asked, and it is predominantly Christ
ians\nwho proselytize on alt.atheism.\n\nSo when I talk of religion, I am talking primarily about religions such as\nChristianity,
Judaism and Islam, which involve some sort of superhuman divine\nbeing.  Much of the discussion will apply to other religions, but
some of it\nmay not.\n\n"What is atheism?"\n\nAtheism is characterized by an absence of belief in the existence of God.\nSome athe
ists go further, and believe that God does not exist.  The former is\noften referred to as the "weak atheist" position, and the la
tter as "strong\natheism".\n\nIt is important to note the difference between these two positions.  "Weak\natheism" is simple scept
icism; disbelief in the existence of God.  "Strong\natheism" is a positive belief that God does not exist.  Please do not\nfall in
to the trap of assuming that all atheists are "strong atheists".\n\nSome atheists believe in the non-existence of all Gods; others
limit their\natheism to specific Gods, such as the Christian God, rather than making\nflat-out denials.\n\n"But isn\'t disbelievin
g in God the same thing as believing he doesn\'t exist?"\n\nDefinitely not.  Disbelief in a proposition means that one does not be
```

In [9]:
```
txt = "From to:  FFRF, P.O. Box 750, Madison, WI 53701.\n"
txt = txt.lower()
n_txt = re.sub(r"from to:"       # start with 'From:'
                    r"[-A-Za-z0-9.:, @<>_)(]*"," ",txt, re.I)
n_txt
```

Out[9]:
```
' \n'
```

In [10]:

```
def preprocessing(txt):
#     sub = re.findall("[^'subject:']*\w+r'\n'$", text, re.IGNORECASE)
#     sub = re.findall("[/n]{2}$", text, re.IGNORECASE)

    # extracting subject
    sub = re.findall(r"subject:+"
           r"[-A-Za-z0-9.:, @]+", txt, re.I)
    sub2 = " ".join(sub)
    sub2 = re.sub(r"Subject:[ ]*","",sub2 , re.I)
    subject = re.sub(r"[A-Za-z0-9_-]+:","",sub2 , re.I)

    # removing Email
    txt = re.sub(r"[A-Za-z0-9_%+-.]+"
                  r"@[A-Za-z0-9.-]+"
                  r"\.[A-Za-z]{2,5}"," ",txt)

    # 16 convert all to lower
    txt = txt.lower()

    # 5. removing line  start with 'write to:' and 'From :'  "Write to:" or "From:"
    n_txt = re.sub(r"write\ to:"  # start with 'write to:'
                   r"[-A-Za-z0-9.:, @<>_]*"," ", txt, re.I)

    n_txt = re.sub(r"from:"       # start with 'From:'
                   r"[-A-Za-z0-9.:, @<>_)(]*"," ",n_txt, re.I)

    #6. delete tags like - "< anyword >"
    n_txt = re.sub(r"<+"                      # start with
                   r"[-A-Za-z0-9.:, @<>_]+"  # selecting every charector
                   ">"," ",n_txt)

    # 7.Delete all the data which are present in the brackets
    n_txt =re.sub(r"\("
                  r"[-A-Za-z0-9.:, @<>_\"\n\t]+"
                  "\)"," ",n_txt)
    # 8. Remove all the newlines('\n'), tabs('\t'), "-", "\".
    n_txt = re.sub("[\t\n]", ' ',n_txt)

    # 9. Remove all the words which ends with ":".
    n_txt = re.sub(r"[A-Za-z0-9_-]+:",'',n_txt )

    # 10. replace can't -> can not, 's -> is, i've -> i have, i'm -> i am, you're -> you are, i'll --> i will
    n_txt = re.sub("can\'t ","can not ",n_txt, re.I)
    n_txt = re.sub("n\'t "," not ",n_txt, re.I)
    n_txt = re.sub("\'s ","is",n_txt, re.I)
```

```
    n_txt = re.sub("i\'ve ", "i have " ,n_txt, re.I)
    n_txt = re.sub("i\'m ", " i am " ,n_txt, re.I)
    n_txt = re.sub("you\'re ", "you are " ,n_txt, re.I)
    n_txt = re.sub("i\'ll ", "i will " ,n_txt, re.I)
    n_txt = re.sub("it\'s ", "it is " ,n_txt, re.I)

    # self
    n_txt = re.sub(r"[ -]{2,20}",' ',n_txt )
    n_txt = re.sub(r"[\"><?]+",'',n_txt )


#     print(txt)
#     print("*"*100)
#     print(n_txt)

    return n_txt, subject


txt, subject = preprocessing(text1)
txt
```

Out[10]:

```
' a word of advice in article  i have said 100 times that there is no alternative that should think you might have caught on by now.
and there is no alternative, but the point is, rationality is not an alternative either. the problems of metaphysical and religious
knowledge are unsolvable or i should say, humans cannot solve them. new york how does that saying those who say it can not be done s
hould not interrupt those who are doing it. jim have you washed your brain today'
```

Chunking

In [11]:

```
from nltk import pos_tag
from nltk import RegexpParser
from nltk import ne_chunk, pos_tag, word_tokenize
from nltk.tree import Tree
import time
```

In [12]:

```
# geting chucked data
# need to remove person name
# modified of ref: https://stackoverflow.com/questions/31836058/nltk-named-entity-recognition-to-a-python-list/31837224#31837224

def get_continuous_chunks(text):
    chunked = ne_chunk(pos_tag(word_tokenize(text)))
    continuous_chunk = []
    continuous_chunk_person = []
    current_chunk = []
    for i in chunked:

        if type(i) == Tree and len(i)>1 and i.label() not in["PERSON", "ORGANIZATION"] : # label for person use  - "PERSON"
                name = np.array(i.leaves(),)[:,0]
                current_chunk.append(list(name))
#                 pdb.set_trace()


        if type(i) == Tree and len(i)>1 and i.label() =="PERSON" : # label for person use  - "PERSON"
                name = np.array(i.leaves())[:,0]
                j_name = " ".join(list(name))
                if j_name not in continuous_chunk:
                    continuous_chunk_person.append(j_name)




#                 pdb.set_trace()
        if current_chunk:
                named_entity = " ".join(current_chunk[0])
                if named_entity not in continuous_chunk:
                        continuous_chunk.append(named_entity)
                        current_chunk = []
        else:
                continue
    return continuous_chunk, continuous_chunk_person

get_continuous_chunks(file[0])
```

Out[12]:
```
(['North Hollywood', 'New York', 'New England'],
 ['Laurel Canyon',
  'Darwin Fish',
  'Lynn Gold',
  'Cameron Road',
  'Holy Horrors',
  'Glenn Drive',
  'Norm R. Allen',
  'United Kingdom Rationalist',
  'Islington High',
  'Germany IBKA',
  'Atheisten Postfach',
  'Konfessionslosen Postfach',
  'Fiction THOMAS',
  'Philip K. Dick Dick',
  'Joe Fernwright',
  'Bantam Press',
  'Temple University Press',
  'Temple University',
  'Without Creed',
  'Without Creed',
  'Ballantine Books',
  'Great Men Think',
  'RICHARD SWINBURNE',
  'Clarendon Paperbacks',
  'Holy Horrors'])
```

In [13]:
```
# replace with original text
def replace_remove_chunk_word(text):

    entity_name, person_name= get_continuous_chunks(text)

    # replacing entity_name with _
    if len(entity_name)>0:
        for i in range(len(entity_name)):
            dual = re.sub("\ ","_", entity_name[i])
            text = re.sub(r"{}".format(entity_name[i]), dual,  text,  re.I)

    # removing person_name
    if len(person_name)>0:
        for p in range(len(person_name)):
            text = re.sub(r"{}".format(person_name[p]), " ",  text,  re.I)

    return text

```

testing w.r.t single file

In [14]:
```
dual_name = get_continuous_chunks(file[2])
new_text = replace_remove_chunk_word(txt[0])
dual_name, new_text[:100]
```

Out[14]:
```
(([], ['Gospel Dating', 'Charley Wingate', 'James Felder']), ' ')
```

In [15]:
```
#chuking and replace with original text

txt = preprocessing(file[2])

# dual_name = get_continuous_chunks(txt[0])
new_text = replace_remove_chunk_word(txt[0])

# print(dual_name)

print(new_text)
```

```
 gospel dating in article well, john has a quite different, not necessarily more elaborated theology. there is some evidence that he
must have known luke, and that the content of q was known to him, but not in a 'canonized' form.  this is a new argument to me. coul
d you elaborate a little  the argument goes as q-oid quotes appear in john, but not in the almost codified way they were in matthew
or luke. however, they are considered to be similar enough to point to knowledge of q as such, and not an entirely different source.
assuming that he knew luke would obviously put him after luke, and would give evidence for the latter assumption.  i do not think th
is follows. if you take the most traditional attributions, then luke might have known john, but john is an elder figure in either ca
se. we're talking spans of time here which are well within the range of lifetimes. we are talking date of texts here, not the age of
the authors. the usual explanation for the time order of mark, matthew and luke does not consider their respective ages. it says mat
thew has read the text of mark, and luke that of matthew . as it is assumed that john knew the content of lukeistext. the evidence f
or that is not overwhelming, admittedly.  earlier manuscripts of john have been discovered.  interesting, where and which how are th
ey dated how old are they  unfortunately, i have not got the info at hand. it was in the late '70s or early '80s, and it was possibl
y as old as ce 200.  when they are from about 200, why do they shed doubt on the order on putting john after the rest of the three i
don't see your point, it is exactly what james felder said. they had no first hand knowledge of the events, and it obvious that at l
east two of them used older texts as the base of their account. and even the association of luke to paul or mark to peter are not ge
nerally accepted.  well, a genuine letter of peter would be close enough, wouldn't it  sure, an original together with id card of se
nder and receiver would be fine. so whatisthat supposed to say am i missing something and i don't think a one step removed source is
that bad. if luke and mark and matthew learned their stories directly from diciples, then i really cannot believe in the sort of big
transformation from jesus to gospel that some people posit. in news reports, one generally gets no better information than this.  an
d if john is a diciple, then there's nothing more to be said.  that john was a disciple is not generally accepted. the style and lan
guage together with the theology are usually used as counterargument. the argument that john was a disciple relies on the claim in t
he gospel of john itself. is there any other evidence for it one step and one generation removed is bad even in our times. compare t
hat to reports of similar events in our century in almost illiterate societies. not even to speak off that believers are not necessa
rily the best sources. it is also obvious that mark has been edited. how old are the oldest manuscripts to my knowledge the oldest i
s quite after any of these estimates, and it is not even complete.  the only clear editing is problem of the ending, and it is basic
ally a hopeless mess. the oldest versions give a strong sense of incompleteness, to the point where the shortest versions seem to br
eak off in midsentence. the most obvious solution is that at some point part of the text was lost. the material from verse 9 on is p
retty clearly later and seems to represent a synopsys of the end of luke.  in other words, one does not know what the original of ma
rk did look like and arguments based on mark are pretty weak. but how is that connected to a redating of john benedikt
```

In [18]:
```
# print(file[2])
```

example for chunking

In [16]:
```
#note - get_continuous_chunks() return 2 list - entity names and person names

#i am living in the New York
print("i am living in the New York -->", get_continuous_chunks("i am living in the New York -->"))
print(" ")
print("-"*50)
print(" ")
#My name is Srikanth Varma
print("My name is Srikanth Varma -->", get_continuous_chunks("My name is Srikanth Varma -->"))

print("\nafter removing:-", replace_remove_chunk_word("My name is Srikanth Varma -->"))
```

```
i am living in the New York --> (['New York'], [])

--------------------------------------------------

My name is Srikanth Varma --> ([], ['Srikanth Varma'])

after removing:- My name is   -->
```

We did chunking for above two lines and then We got one list where each word is mapped to a POS(parts of speech) and also if you see "New York" and "Srikanth Varma", they got combined and represented as a tree and "New York" was referred as "GPE" and "Srikanth Varma" was referred as "PERSON".
so now you have to Combine the "New York" with "_" i.e "New_York"
and remove the "Srikanth Varma" from the above sentence because it is a person.

In [17]:
```
preprocessing(text1)
```

Out[17]:
```
(' a word of advice in article  i have said 100 times that there is no alternative that should think you might have caught on by no
w. and there is no alternative, but the point is, rationality is not an alternative either. the problems of metaphysical and religio
us knowledge are unsolvable or i should say, humans cannot solve them. new york how does that saying those who say it can not be don
e should not interrupt those who are doing it. jim have you washed your brain today',
 'A word of advice')
```

In [18]:
```
text1 = text1 + " _word_ "
print(text1)
```

```
Subject: A word of advice
From: jcopelan@nyx.cs.du.edu (The One and Only)

In article < 65882@mimsy.umd.edu > mangoe@cs.umd.edu (Charley Wingate) writes:
>
>I've said 100 times that there is no "alternative" that should think you
>might have caught on by now.  And there is no "alternative", but the point
>is, "rationality" isn't an alternative either.  The problems of metaphysical
>and religious knowledge are unsolvable-- or I should say, humans cannot
>solve them. new york

How does that saying go: Those who say it can't be done shouldn't interrupt
those who are doing it.

Jim
--
Have you washed your brain today? _word_
```

part-2 preprossesing

13. Replace all the digits with space i.e delete all the digits.
> In the above sample document, the 6th line have digit 100, so we have to remove that.

14. After doing above points, we observed there might be few word's like
"_word_" (i.e starting and ending with the _), "_word" (i.e starting with the _),
"word_" (i.e ending with the _) remove the _ from these type of words.

15. We also observed some words like **"OneLetter_word"- eg: d_berlin,**
"TwoLetters_word" - eg: dr_berlin , in these words we remove the "OneLetter_" (d_berlin ==> berlin) and
"TwoLetters_" (de_berlin ==> berlin). i.e remove the words
which are length less than or equal to 2 after spliiting those words by "_".

16. Convert all the words into lower case and lowe case
and remove the words which are greater than or equal to 15 or less than or equal to 2.

17. replace all the words except "A-Za-z_" with space.

18. Now You got Preprocessed Text, email, subject. create a dataframe with those.
Below are the columns of the df.

In [19]:

```
# 13- 17, preprosses_2 with regular expression

def preprocessing2(text):

#      # 5. removing line  start with 'write to:' and 'From :'  "Write to:" or "From:"
#     n_txt = re.sub(r"write\ to:"  # start with 'write to:'
#                    r"[-A-Za-z0-9.:, @<>_]*"," ", txt)

#     n_txt = re.sub(r"From:"       # start with 'From:'
#                    r"[-A-Za-z0-9.:, @<>_)(]*"," ",n_txt, re.I)

#     #6. delete tags like - "< anyword >"
#     n_txt = re.sub(r"<+"                      # start with
#                    r"[-A-Za-z0-9.:, @<>_]+"  # selecting every charector
#                    ">"," ",n_txt)

    # 13 replacing digits with space
    n_txt = re.sub(r"[0-9]+"," ", text)

    # 14 removing _ in " _word_, _word, word_ " these type of words
    n_txt = re.sub(r"\ \_[-A-Za-z0-9.:,@<>_)]+", " ",n_txt)
    n_txt = re.sub(r"[-A-Za-z0-9.:,@<>_)]+\_\ ", " ",n_txt)

    # 15 remove start short word(length less than or equal to 2) "TwoLetters_word" - eg: (d_berlin ==> berlin)
    n_txt = re.sub(r"\ [A-za-z0-9]{1,2}\_[A-za-z0-9]+"," ", n_txt)

    # 16 convert all to lower
    # n_txt = n_txt.lower()

    # 17 replace all the words except "A-Za-z_" with space.
    n_txt = re.sub("[^A-Za-z_ ]"," ", n_txt)

    # removing extra space
    n_txt = re.sub("[ ]{2,5}"," ",n_txt)

    return n_txt

txt,d = preprocessing("""After de_bfl doing above dee_bfl poin_tkks  po_tkks  poin_tk, we observed d_berlin  there might be few w
  word_ (i.e ending with the _) remove the  from these type of words.""")
txt
```

Out[19]:

```
'after de_bfl doing above dee_bfl poin_tkks po_tkks poin_tk, we observed d_berlin there might be few word /islike _word_ , _word , w
ord_ remove the from these type of words.'
```

test preprocessing2 w.r.t. single file

In [20]:
```
# txt,subject = preprocessing(file[4])
# txt1 = replace_remove_chunk_word(txt)
# dual = get_continuous_chunks(txt1)

# txt  = preprocessing2(txt)

# dual, txt

```

In [21]:
```
# txt1
```

In [22]:
```
# file[4]
```

In [23]:
```
# dual = get_continuous_chunks(file[4])
# dual
```

To get above mentioned data frame --> Try to Write Total Preprocessing steps in One Function Named Preprocess as below.

In [24]:
```
def preprocess_all(Input_Text):
    """Do all the Preprocessing as shown above and
    return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that Text_data"""
    list_text =[]
    subjects = []
    list_of_preproessed_emails = []
    for i in range(len(Input_Text)):

        # prepossing + chucking
        email = email_extractor(Input_Text[i])
        txt, sub1 = preprocessing(Input_Text[i])
        subject = preprocessing2(sub1)
        new_text = replace_remove_chunk_word(txt)
        new_text = preprocessing2(new_text)


        # appending
        list_text.append(new_text)
        subjects.append(subject)
        list_of_preproessed_emails.append(email)
#         pdb.set_trace()

    return (list_of_preproessed_emails,subjects,list_text)
```

Training The models to Classify:

Creating Dataframe

In [ ]:
```

```

In [299]:
```
%%time
# took more than 1 h 40min

# processing all file data
list_of_preproessed_emails,subjects,list_texts = preprocess_all(file)
```

```
Wall time: 52min 21s
```

In [148]:
```
# extracting class/label from file name


file_name = rf.namelist()
classes = []
for i in file_name[:-1]:

    l = re.findall(r"\/[A-Za-z.]*", i)
    l = re.sub(r"\/","","," ".join(l))
    classes.append(l)

print(len(classes))
```

```
18828
```

In [288]:
```
classes
```

...

In [149]:
```
# reshaping data
# list_of_preproessed_emails = np.reshape(list_of_preproessed_emails,(-1,1))
# subjects = np.reshape(subjects, (-1,1))
# list_texts = np.reshape(list_texts, (-1,1))
# label1 = np.reshape(classes, (-1,1))

list_of_preproessed_emails.shape, subjects.shape, list_texts.shape, label1.shape
```

Out[149]:
```
((18828, 1), (18828, 1), (18828, 1), (18828, 1))
```

In [150]:
```
(np.unique(label1))
```

Out[150]:
```
array(['alt.atheism', 'comp.graphics', 'comp.os.ms',
       'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware',
       'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles',
       'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt',
       'sci.electronics', 'sci.med', 'sci.space',
       'soc.religion.christian', 'talk.politics.guns',
       'talk.politics.mideast', 'talk.politics.misc',
       'talk.religion.misc'], dtype='<U24')
```

In [283]:
```
# label[label=='alt.atheism'].shape
# label[label=='soc.religion.christian'].shape
# sns.distplot(label, kde=True, label= np.unique(label))
# label.shape

```

In [303]:
```
# list_texts
```

Out[303]:
```
[' alt atheism atheist resources atheism resources resources december  atheist resources addresses of atheist organizations usa fr
eedom from religion foundation darwin fish bumper stickers and assorted other atheist paraphernalia are available from the freedom
from religion foundation in the us  evolution designs evolution designs sell the darwin fish itisa fish symbol like the ones chris
tians stick on their cars but with feet and the word darwin written inside the deluxe moulded d plastic fish is  postpaid in the u
s  north hollywood ca people in the san francisco bay area can get darwin fish from lynn gold try mailing for net people who go to
lynn directly the price is  per fish american atheist press aap publish various atheist books critiques of the bible lists of bibl
ical contradictions and so on one such book the bible handbook by w p ball and g w foote american atheist press pp isbn   nd editi
on bible contradictions absurdities atrocities immoralities contains ball the bible contradicts itself aap based on the king james
version of the bible write american atheist press p o box austin tx  cameron road austin tx   prometheus books sell books includin
g haughtisholy horrors write east amherst street buffalo new york  an alternate address prometheus books glenn drive buffalo ny  a
frican americans for humanism an organization promoting black secular humanism and uncovering the history of black freethought the
y publish a quarterly newsletter aah examiner write norm r allen jr african americans for humanism p o box buffalo ny united kingd
om rationalist press association national secular society islington high street holloway road london n ew london n nl   british hu
manist association south place ethical society lamb s conduit passage conway hall london wc r rh red lion square  london wc r rl f
ax   the national secular society publish the freethinker a monthly magazine founded in germany ibka e v internationaler bund der
konfessionslosen und atheisten postfach d berlin germany ibka publish a miz miz vertrieb postfach d berlin germany for atheist boo
ks write ibdk internationaler bucherdienst der konfessionslosen postfach d hannover germany  books fiction thomas m disch the sant
a claus compromise short story the ultimate proof that santa exists all characters and events are fictitious any similarity to liv
ing or dead gods uh well walter m miller jr a canticle for leibowitz one gem in this post atomic doomsday novel is the monks who s
```

In [311]:
```
# classes
```

In [312]:
```
e1 = ["Geeks", "For", "Geeks"]
e2= ["Geeks", "For", "Geeks"]
e3 = ["Geeks", "For", "Geeks"]

df = pd.DataFrame({
    'a':classes[:3],
    'b':e2,
    'c':e3
})
df
```

Out[312]:

	a	b	c
0	alt.atheism	Geeks	Geeks
1	alt.atheism	For	For
2	alt.atheism	Geeks	Geeks

In [313]:
```
# creating dateframe

df = pd.DataFrame({
    'label':classes,
    'processed_text': list_texts,
    'procesossed_subject': subjects,
    'procesossed_emails' : list_of_preproessed_emails })

#stacking data
# processed_data = np.hstack((label1, list_texts, subjects,list_of_preproessed_emails))

#converting to dataframe
# df = pd.DataFrame(processed_data, columns= ["label","procesossed_text", "procesossed_subject", "procesossed_emails" ])
```

In [284]:
```
# np.array2string(list_texts[1])
list_texts[4]
```

Out[284]:
```
array([' soc motss et al princeton axes matching funds for boy scouts in article  however i hate economic terrorism and political co
rrectness worse than i hate this policy  a more effective approach is to stop donating to any organizating that directly or indirect
ly supports gay rights issues until they end the boycott on funding of scouts can somebody reconcile the apparent contradiction betw
een and rob strom  ibm research saw mill river road p o box yorktown heights ny '],
      dtype='<U99350')
```

In [315]:
```
df['procesossed_subject']
```

Out[315]:
```
0                            Alt Atheism Atheist Resources
1                      Alt Atheism Introduction to Atheism
2                                             Gospel Dating
3                university violating separation of church
4
                                ...
18823                                Religion and marriage
18824     A Message for you Mr How do you know what hap...
18825                      Why did they behave as they did
18826                                   Info about New Age
18827                                                    I
Name: procesossed_subject, Length: 18828, dtype: object
```

In [300]:
```
# df2[0]
```

In [317]:
```
# 1. combine columns
df2 = df['procesossed_subject'] +df['procesossed_emails']+ df['processed_text']
# df2 = df.loc[:,['procesossed_subject', 'procesossed_emails','processed_text']]

# labelizing
label = label_binarize(classes,np.unique(df[['label']]) )

df2.shape, label.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:67: FutureWarning: Pass classes=['alt.atheism' 'comp.graphic
s' 'comp.os.ms' 'comp.sys.ibm.pc.hardware'
 'comp.sys.mac.hardware' 'comp.windows.x' 'misc.forsale' 'rec.autos'
 'rec.motorcycles' 'rec.sport.baseball' 'rec.sport.hockey' 'sci.crypt'
 'sci.electronics' 'sci.med' 'sci.space' 'soc.religion.christian'
 'talk.politics.guns' 'talk.politics.mideast' 'talk.politics.misc'
 'talk.religion.misc'] as keyword args. From version 0.25 passing these as positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
```

Out[317]:
```
((18828,), (18828, 20))
```

Code checking:

After Writing preprocess function. call that functoin with the input text of 'alt.atheism_49960' doc and print the output of the preprocess function

This will help us to evaluate faster, based on the output we can suggest you if there are any changes.

In [36]:
```
# # prepossing + chucking
# email = email_extractor(file[2])
# txt, subject = preprocessing(file[2])
# new_text = replace_remove_chunk_word(txt)
# new_text = preprocessing2(new_text)
# subject = preprocessing2(subject)
```

In [37]:
```
# file[2] # file name  - alt.atheism_49960
```

In [38]:
```
# print(email,'\n', subject,'\n', new_text[:100])
# new_text
```

In [228]:
```
import tensorflow as tf
from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense,Input,Activation

from tensorflow.keras.preprocessing.text import Tokenizer
import seaborn as sns

```

Spliting Data

In [319]:
```
# df2 = np.array(df2)

```

Out[319]: (18828,)

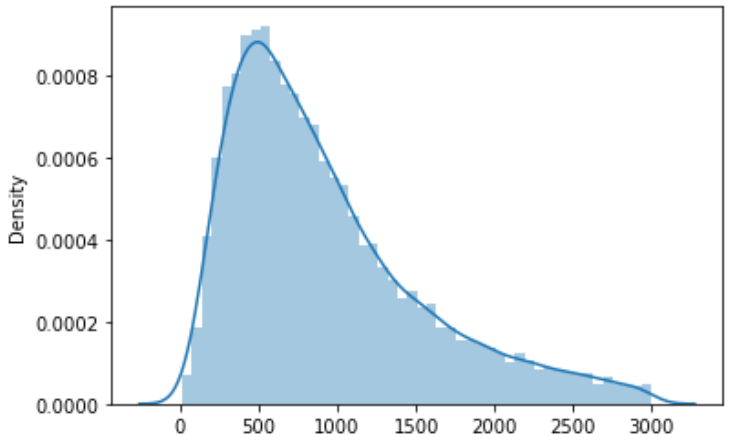
In [341]:
```
# 2. split the data into Train and test
from sklearn.model_selection import train_test_split
df2 = np.array(df2)
X_train, X_test, y_train, y_test = train_test_split(df2,label, test_size=0.2, stratify= label )

```

3. Analyze your text data and pad the sequnce

In [323]:
```
# 3. Analyze your text data and pad the sequnce

lens_txt =np.array([len(x) for x in X_train])

print(np.median([len(x) for x in X_train]) ,",", np.mean([len(x) for x in X_train]))
sns.distplot(lens_txt[lens_txt<=3000], kde = True)

# pad the text size
```

```
825.0 , 1435.4373257203558

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and wil
l be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[323]:
```
<AxesSubplot:ylabel='Density'>
```



observation: max_len for text = 1000, 1000 is good middle value

In [324]:
```

lens_txt =np.array([len(x) for x in X_train])
# lens_txt[lens_txt >= 1000]
lens_txt
```

Out[324]:
```
array([1850,  570,  432, ..., 1607, 1195, 2134])
```

Do Tokenizer And creating embedding matrix through glove file

```
i.e convert text( word and char) into numbers.
```

After writing Preprocess function, call the function for each of the document(18828 docs) and then create a dataframe as mentioned above.

In [325]:

```
# 4.1 Do Tokenizer i.e convert text into numbers.
# https://stackoverflow.com/questions/51956000/what-does-keras-tokenizer-method-exactly-do


t = Tokenizer(num_words = 1000, filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
t.fit_on_texts(X_train)

# convert each sentence to matrix(1000 x 1)
encoded_docs = t.texts_to_sequences(X_train)

print("sequences : ",len(encoded_docs),'\n')

print("word_index : ",len(t.word_index))


```

```
sequences :  15062

word_index :  112379
```

In [326]:
```
# 4.2 Do Tokenizer i.e convert char into numbers.

t2 = Tokenizer(num_words = 1000, filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n', char_level = True)
t2.fit_on_texts(X_train)

# convert each sentence to matrix(1000 x 1)
encoded_docs2 = t2.texts_to_sequences(X_train)
encoded_docs_test = t2.texts_to_sequences(X_test)
print("sequences : ",len(encoded_docs2),'\n')

print("word_index : ",(t2.word_index))

```

```
sequences :  15062

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

In [327]:
```
# char_list = list(t2.word_index.keys())
# char_ohe = label_binarize(char_list,char_list )
# char_ohe.shape
```

In [328]:
```
# padding every encoded txt file to (n x 1000)
from tensorflow.keras.utils import pad_sequences

print(len(encoded_docs[5]))

padded_docs = pad_sequences(encoded_docs, maxlen=1000, padding='post')
padded_docs_test = pad_sequences(encoded_docs_test, maxlen=1000, padding='post')
padded_docs_char = pad_sequences(encoded_docs2, maxlen=1000, padding='post')

print(padded_docs.shape, padded_docs_char.shape, padded_docs_test.shape)
```

```
77
(15062, 1000) (15062, 1000) (3766, 1000)
```

In [329]:

```
%%time
# load the whole glove file
embeddings_index = dict()
f = open('glove.6B.100d.txt', encoding="utf8")
for i,line in enumerate(f):
    try:
        #     pdb.set_trace()
        values = line.split(' ')
        word = values[0]
        coefs = np.asarray(values[1:])
        #pdb.set_trace()
        # storing vector of word into dict
        embeddings_index[word] = coefs
#         print(i)
    except:
        pdb.set_trace()

f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
Wall time: 26.4 s
```

In [330]:

```
# create a weight matrix for words in training docs
vocab_size = 1000
embedding_matrix_word = np.zeros((vocab_size, 100))
for word, i in t.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if i >= 1000:
#         pdb.set_trace()
        break
    if embedding_vector is not None:
        embedding_matrix_word[i] = embedding_vector


```

In [ ]:

```

```

In [331]:

```
%%time
# load the whole glove file for char vectorization
char_embed_val = dict()
f = open('glove.840B.300d-char.txt', encoding="utf8")
for i,line in enumerate(f):
    try:
        # pdb.set_trace()
        values = line.split(' ')
        char = values[0]
        coefs = np.asarray(values[1:])
        #pdb.set_trace()
        # storing vector of word into dict
        char_embed_val[char] = coefs
#         print(i)
    except:
        None

f.close()
print('Loaded %s word vectors.' % len(char_embed_val))
```

```
Loaded 94 word vectors.
Wall time: 40 ms
```

In [332]:

```
# create a weight matrix for char in training docs
char_vocab_size = len(char_embed_val.keys())
char_embed_matrix = np.zeros((char_vocab_size, 300))

for word, i in t2.word_index.items():
    embedding_vector = char_embed_val.get(word)
    if i >= 1000:
#
        break
    if embedding_vector is not None:
#         pdb.set_trace()
        char_embed_matrix[i] = embedding_vector

print(char_embed_matrix.shape)
```

```
(94, 300)
```

Training The models to Classify:

1. Combine "preprocessed_text", "preprocessed_subject", "preprocessed_emails" into one column. use that column to model.

2. Now Split the data into Train and test. use 25% for test also do a stratify split.

3. Analyze your text data and pad the sequnce if required.
Sequnce length is not restricted, you can use anything of your choice.
you need to give the reasoning

4. Do Tokenizer i.e convert text into numbers. please be careful while doing it.
if you are using tf.keras "Tokenizer" API, it removes the "_", but we need that.

5. code the model's (Model-1, Model-2) as discussed below
and try to optimize that models.

6. For every model use predefined Glove vectors.
Don't train any word vectors while Training the model.

7. Use "categorical_crossentropy" as Loss.

8. Use **Accuracy and Micro Avgeraged F1 score** as your as Key metrics to evaluate your model.

9. Use Tensorboard to plot the loss and Metrics based on the epoches.

10. Please save your best model weights in to **'best_model_L.h5' (L = 1 or 2).**

11. You are free to choose any Activation function, learning rate, optimizer.
But have to use the same architecture which we are giving below.

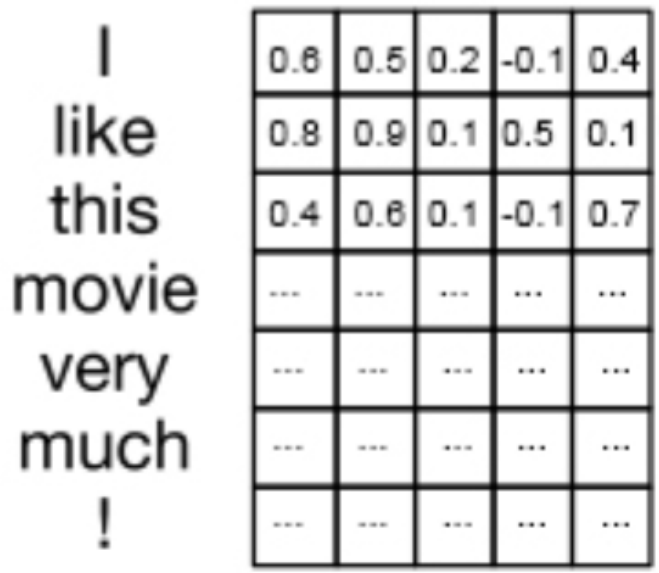
12. You can add some layer to our architecture but you **deletion** of layer is not acceptable.

13. Try to use **Early Stopping** technique or any of the callback techniques that you did in the previous assignments.

14. For Every model save your model to image (Plot the model) with shapes
and inlcude those images in the notebook markdown cell,
upload those imgages to Classroom. You can use "plot_model"
please refer [this (https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model)](https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model) if you don't know how to plot the mo
del with shapes.

Model-1: Using 1D convolutions with word embeddings

Encoding of the Text --> For a given text data create a Matrix with Embedding layer as shown Below.
In the example we have considered d = 5, but in this assignment we will get d = dimension of Word vectors we are using.
i.e if we have maximum of 350 words in a sentence and embedding of 300 dim word vector,
we result in 350*300 dimensional matrix for each sentance as output after embedding layer



Ref: https://i.imgur.com/kiVQuk1.png

Reference:
https://stackoverflow.com/a/43399308/4084039 (https://stackoverflow.com/a/43399308/4084039)
https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/ (https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/)

How EMBEDDING LAYER WORKS (https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work)

Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)

creating and training model

In [333]:
```
from tensorflow.keras import layers
from tensorflow.keras.initializers import RandomNormal
import tensorflow as tf
import os
import datetime
import tensorflow_addons as tfa
from sklearn.metrics import f1_score
from tensorflow.keras.layers import Embedding
```

In [334]:
```
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [335]:

```
# callback
# Printing costum made accuracy on validation data

class LossHistory(tf.keras.callbacks.Callback):

    def __init__(self,validation_data):
        self.x_test = validation_data[0]
        self.y_test= validation_data[1]


    def on_epoch_end(self, epoch, logs={}):

        # # we can get a list of all predicted values at the end of the epoch
        # # we can use these predicted value and the true values to calculate any custom evaluation score if it is needed for our
        # # Here we are taking log of all true positives and then taking average of it
        self.y_pred= self.model.predict(self.x_test)
        self.y_label_pred=np.argmax(self.y_pred,axis=1)

        #calcualting  f1_score through sklearn
        y_pred2 = [1 if x >=0.5 else 0 for x in self.y_pred[:,1]]
#         pdb.set_trace()
        f1 = f1_score( self.y_test[:,1], y_pred2, average = "micro")

        print('f1 score', f1)


```

In [391]:

```

def create_model_1():
    input_layer = Input(shape=(1000))
    # vocab_size = 1000, embedding_matrix.shape = (1000, 100)
    embeding = layers.Embedding(vocab_size, 100, weights=[embedding_matrix_word], input_length=1000, trainable=False)(input_layer

    n1 = layers.BatchNormalization()(embeding)

    l2 = layers.Conv1D(100, 4)(n1)
    l3 = layers.Conv1D(100, 4)(n1)
    l4 = layers.Conv1D(100, 4)(n1)

    # concatenate 3 output to 1 output
    l5 = layers.concatenate([l2,l3,l4])

    n2 = layers.BatchNormalization()(l5)

    l6 = layers.MaxPool1D()(n2)

    n3 = layers.Dropout(0.5)(l6)

    l7 = layers.Conv1D(62,5)(n3)
    l8 = layers.Conv1D(62,5)(n3)
    l9 = layers.Conv1D(62,5)(n3)


    # concatenate 3 output to 1 output
    L10 = layers.concatenate([l7,l8,l9])

    n4 = layers.BatchNormalization()(L10)

    L11 = layers.MaxPool1D()(n4)

    n5 = layers.Dropout(0.4)(L11)

    L12 = layers.Conv1D(62,3)(n5)
    L13 = layers.Flatten()(L12)
    L14 = layers.Dropout(0.4)(L13)
    L15 = layers.Dense(256, activation= "relu")(L14)

    n6 = layers.BatchNormalization()(L15)
    n7 = layers.Dropout(0.4)(n6)

    output_layer = layers.Dense(20, activation = 'softmax')(n7)

```

```
    model = Model(input_layer,output_layer, name= "text_df")

    return model

# tensorboard_callback
log_dir = os.path.join("logs",'fits', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)
# custom accuracy callback
history_own=LossHistory(validation_data=[padded_docs_test,y_test])

model = None
model = create_model_1()
```

In [392]:

```
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss = tf.keras.losses.BinaryCrossentropy(), metrics=[tf.keras.metrics.AUC(multi_label = True, num_labels= 20)
                                                                    ,tfa.metrics.F1Score(num_classes = 20, average = 'micro')
                                                                    ])

model.fit(padded_docs, y_train, batch_size=16, epochs =2, validation_data = [padded_docs_test, y_test], callbacks =[tensorboard_c
```

```
Epoch 1/2
942/942 [==============================] - 733s 776ms/step - loss: 0.2941 - auc_40: 0.4981 - f1_score: 0.0503 - val_loss: 0.3267 - v
al_auc_40: 0.5042 - val_f1_score: 0.0542
Epoch 2/2
942/942 [==============================] - 850s 902ms/step - loss: 0.2138 - auc_40: 0.4988 - f1_score: 0.0504 - val_loss: 0.2970 - v
al_auc_40: 0.4946 - val_f1_score: 0.0515
```

Out[392]:

```
<keras.callbacks.History at 0x15228b9b2b0>
```

In [396]:

```
pip install graphviz-2.38
```

```
Note: you may need to restart the kernel to use updated packages.

ERROR: Could not find a version that satisfies the requirement graphviz-2.38 (from versions: none)
ERROR: No matching distribution found for graphviz-2.38
```

In [393]:

```

from tensorflow.keras.utils import plot_model
plot_model(model, "mini_resnet2.png", show_shapes=True)

# y = tf.keras.layers.Conv2D(2, 3, activation='relu', dilation_rate=2, input_shape=input_shape[1:])(x)
```

```
You must install pydot (`pip install pydot`) and install graphviz (see instructions at https://graphviz.gitlab.io/download/) (https://graphviz.gitlab.io/download/)) for plot_model/model_to_dot to work.
```

In [369]:
```
model.summary()
```

```
Model: "text_df"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_28 (InputLayer)          [(None, 1000)]       0           []

 embedding_27 (Embedding)       (None, 1000, 100)    100000      ['input_28[0][0]']

 batch_normalization_4 (BatchNo  (None, 1000, 100)   400         ['embedding_27[0][0]']
 rmalization)

 conv1d_175 (Conv1D)            (None, 998, 164)     49364       ['batch_normalization_4[0][0]']

 conv1d_176 (Conv1D)            (None, 998, 164)     49364       ['batch_normalization_4[0][0]']

 conv1d_177 (Conv1D)            (None, 998, 164)     49364       ['batch_normalization_4[0][0]']

 concatenate_46 (Concatenate)   (None, 998, 492)     0           ['conv1d_175[0][0]',
                                                                  'conv1d_176[0][0]',
                                                                  'conv1d_177[0][0]']

 batch_normalization_5 (BatchNo  (None, 998, 492)    1968        ['concatenate_46[0][0]']
 rmalization)

 max_pooling1d_52 (MaxPooling1D  (None, 499, 492)    0           ['batch_normalization_5[0][0]']
 )

 dropout_27 (Dropout)           (None, 499, 492)     0           ['max_pooling1d_52[0][0]']

 conv1d_178 (Conv1D)            (None, 492, 62)      244094      ['dropout_27[0][0]']

 conv1d_179 (Conv1D)            (None, 492, 62)      244094      ['dropout_27[0][0]']

 conv1d_180 (Conv1D)            (None, 492, 62)      244094      ['dropout_27[0][0]']

 concatenate_47 (Concatenate)   (None, 492, 186)     0           ['conv1d_178[0][0]',
                                                                  'conv1d_179[0][0]',
                                                                  'conv1d_180[0][0]']

 batch_normalization_6 (BatchNo  (None, 492, 186)    744         ['concatenate_47[0][0]']
 rmalization)

 max_pooling1d_53 (MaxPooling1D  (None, 246, 186)    0           ['batch_normalization_6[0][0]']
 )
```

```
 dropout_28 (Dropout)           (None, 246, 186)     0           ['max_pooling1d_53[0][0]']

 conv1d_181 (Conv1D)            (None, 244, 62)      34658       ['dropout_28[0][0]']

 flatten_25 (Flatten)           (None, 15128)        0           ['conv1d_181[0][0]']

 dropout_29 (Dropout)           (None, 15128)        0           ['flatten_25[0][0]']

 dense_50 (Dense)               (None, 256)          3873024     ['dropout_29[0][0]']

 batch_normalization_7 (BatchNo  (None, 256)         1024        ['dense_50[0][0]']
 rmalization)

 dropout_30 (Dropout)           (None, 256)          0           ['batch_normalization_7[0][0]']

 dense_51 (Dense)               (None, 20)           5140        ['dropout_30[0][0]']

==================================================================================================
Total params: 4,897,332
Trainable params: 4,795,264
Non-trainable params: 102,068
__________________________________________________________________________________________________
```

In [352]:

```
1 len(padded_docs)
```

Out[352]:

```
15062
```

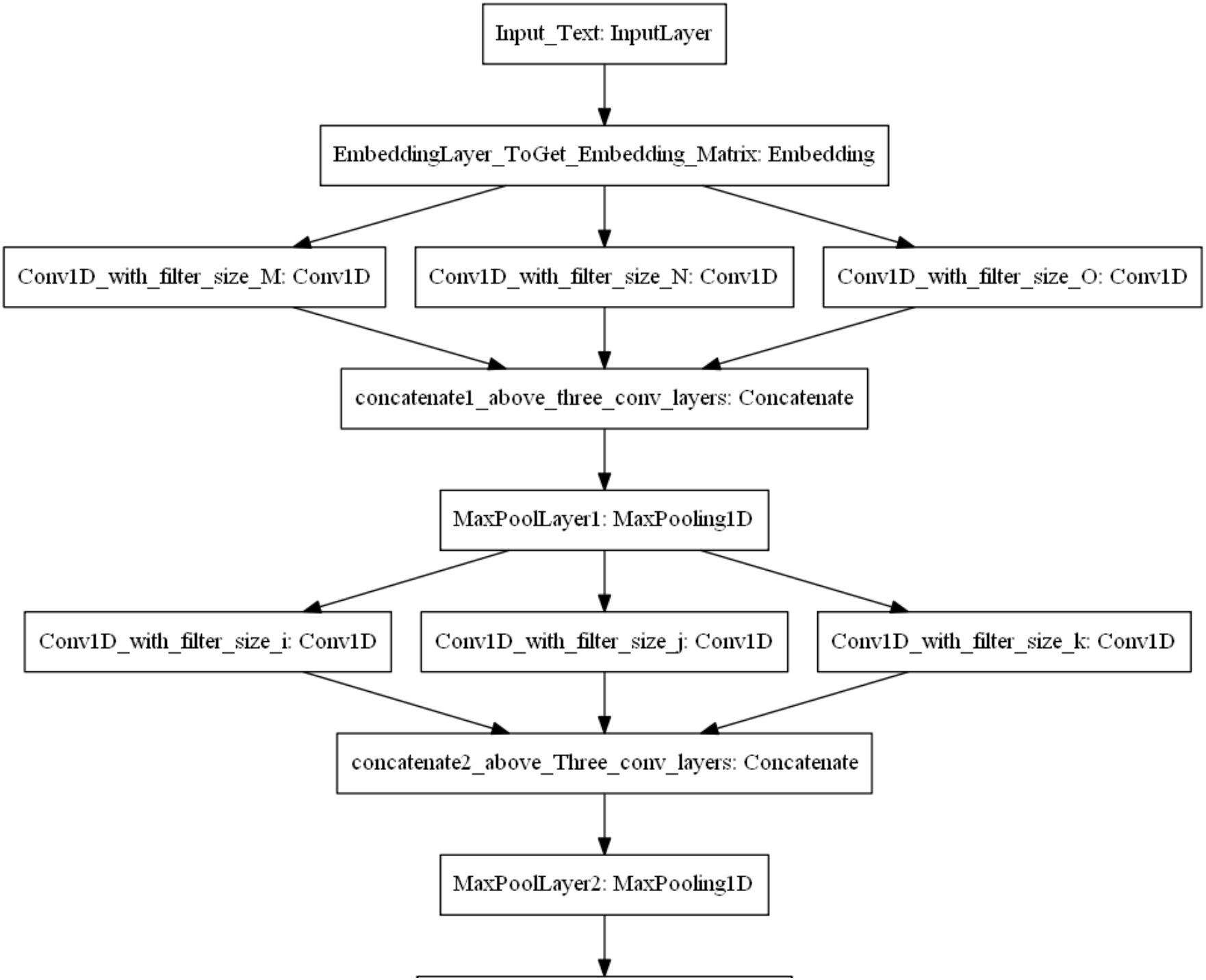
In [355]:
```
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss = tf.keras.losses.BinaryCrossentropy(), metrics=[tf.keras.metrics.AUC(multi_label = True, num_labels= 20)
                                                                    ,tfa.metrics.F1Score(num_classes = 20, average = 'micro')
                                                                    , tf.keras.metrics.Accuracy()])

model.fit(padded_docs, y_train, batch_size=16, epochs =4, validation_split=0.2, callbacks =[tensorboard_callback] )
```

```
Epoch 1/4
754/754 [==============================] - 209s 275ms/step - loss: 0.2132 - auc_33: 0.4987 - f1_score: 0.0510 - accuracy: 1.3694e-04
- val_loss: 0.2069 - val_auc_33: 0.4980 - val_f1_score: 0.0498 - val_accuracy: 0.0000e+00
Epoch 2/4
754/754 [==============================] - 232s 308ms/step - loss: 0.2061 - auc_33: 0.4975 - f1_score: 0.0500 - accuracy: 5.3946e-05
- val_loss: 0.2025 - val_auc_33: 0.4987 - val_f1_score: 0.0561 - val_accuracy: 0.0000e+00
Epoch 3/4
754/754 [==============================] - 212s 281ms/step - loss: 0.2034 - auc_33: 0.4946 - f1_score: 0.0493 - accuracy: 0.0000e+00
- val_loss: 0.2003 - val_auc_33: 0.5003 - val_f1_score: 0.0461 - val_accuracy: 0.0000e+00
Epoch 4/4
754/754 [==============================] - 242s 320ms/step - loss: 0.2014 - auc_33: 0.4960 - f1_score: 0.0499 - accuracy: 0.0000e+00
- val_loss: 0.1991 - val_auc_33: 0.4987 - val_f1_score: 0.0498 - val_accuracy: 0.0000e+00
```

Out[355]:
```
<keras.callbacks.History at 0x151edce8310>
```

In [140]:
```
%tensorboard --logdir logs
```

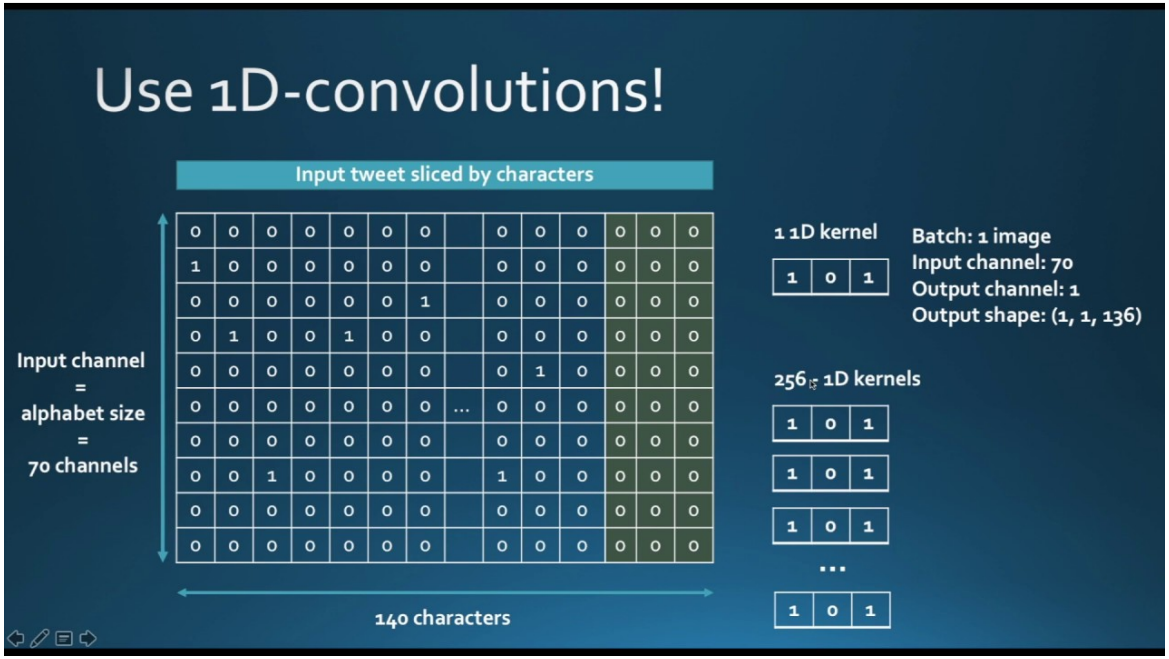
```
Reusing TensorBoard on port 6006 (pid 8704), started 0:46:11 ago. (Use '!kill 8704' to kill it.)
```


Input_Text: InputLayer
EmbeddingLayer_ToGet_Embedding_Matrix: Embedding
Conv1D_with_filter_size_M: Conv1D
Conv1D_with_filter_size_N: Conv1D
Conv1D_with_filter_size_O: Conv1D
concatenate1_above_three_conv_layers: Concatenate
MaxPoolLayer1: MaxPooling1D
Conv1D_with_filter_size_i: Conv1D
Conv1D_with_filter_size_j: Conv1D
Conv1D_with_filter_size_k: Conv1D
concatenate2_above_Three_conv_layers: Concatenate
MaxPoolLayer2: MaxPooling1D

ref: 'https://i.imgur.com/fv1GvFJ.png'

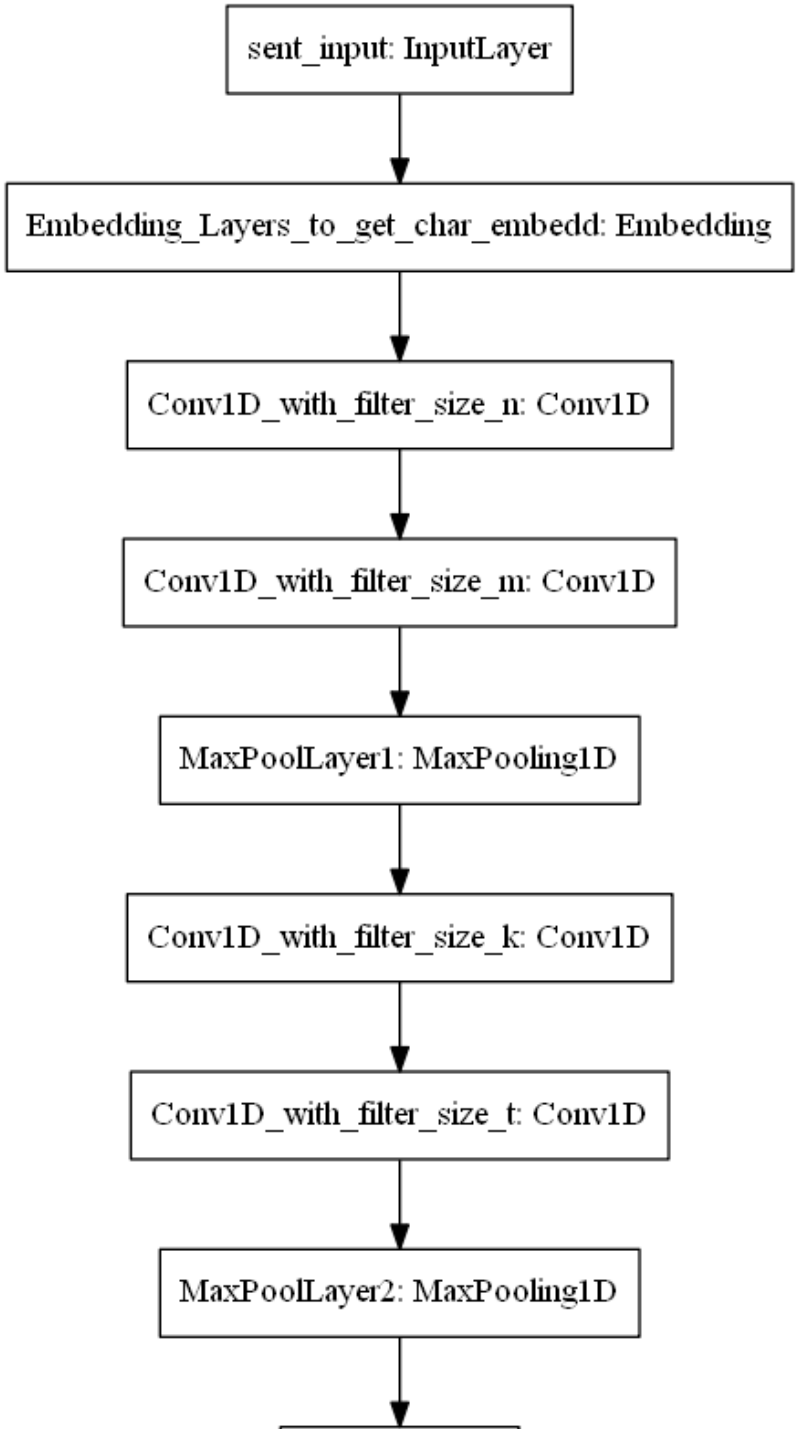

```
1. all are Conv1D layers with any number of filter and filter sizes, there is no restriction on this.

2. use concatenate layer is to concatenate all the filters/channels.

3. You can use any pool size and stride for maxpooling layer.

4. Don't use more than 16 filters in one Conv layer becuase it will increase the no of params.
( Only recommendation if you have less computing power )

5. You can use any number of layers after the Flatten Layer.
```

Model-2 : Using 1D convolutions with character embedding



Here are the some papers based on Char-CNN

1. Xiang Zhang, Junbo Zhao, Yann LeCun. [Character-level Convolutional Networks for Text Classification (http://arxiv.org/abs/1509.01626)](http://arxiv.org/abs/1509.01626).NIPS 2015
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. [Character-Aware Neural Language Models (https://arxiv.org/abs/1508.06615)](https://arxiv.org/abs/1508.06615). AAAI 2016
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. [An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling (https://arxiv.org/pdf/1803.01271.pdf)](https://arxiv.org/pdf/1803.01271.pdf)
4. Use the pratrained char embeddings [https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt (https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt)](https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt)

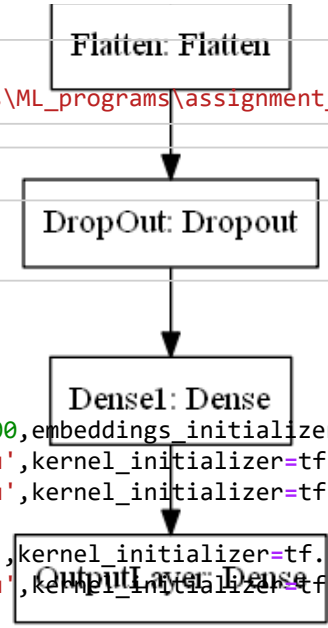

sent_input: InputLayer
Embedding_Layers_to_get_char_embedd: Embedding
Conv1D_with_filter_size_n: Conv1D
Conv1D_with_filter_size_m: Conv1D
MaxPoolLayer1: MaxPooling1D
Conv1D_with_filter_size_k: Conv1D
Conv1D_with_filter_size_t: Conv1D
MaxPoolLayer2: MaxPooling1D

In [356]:
```
import shutil
shutil.rmtree(r'C:\Users\ashutosh tiwari\Documents\ML_programs\assignment_aai\21_document with cnn with text data\logs')
```

In [357]:
```
vocab_size, char_embed_matrix.shape
```

Out[357]:
```
(1000, (94, 300))
```

In [360]:
```

def create_model_2():

    input_layer = tf.keras.Input(shape=(1000,))
    char_emb = layers.Embedding(char_vocab_size,300,embeddings_initializer=tf.keras.initializers.Constant(char_embed_matrix),trai
    convlay1 = layers.Conv1D(32,5,activation='relu',kernel_initializer=tf.keras.initializers.HeNormal(seed=5))(char_emb)
    convlay2 = layers.Conv1D(32,3,activation='relu',kernel_initializer=tf.keras.initializers.HeNormal(seed=10))(convlay1)
    maxpool1 = layers.MaxPool1D(2,2)(convlay2)
    convlay3 = layers.Conv1D(8,3,activation='relu',kernel_initializer=tf.keras.initializers.HeNormal(seed=15))(maxpool1)
    convlay4 = layers.Conv1D(16,5,activation='relu',kernel_initializer=tf.keras.initializers.HeNormal(seed=20))(convlay3)
    maxpool2 = layers.MaxPool1D(2,2)(convlay4)
    flat = layers.Flatten()(maxpool2)
    drop = layers.Dropout(0.2)(flat)
    dlay = layers.Dense(40,activation='relu',kernel_initializer=tf.keras.initializers.HeNormal(seed=25))(drop)
    output_layer = layers.Dense(20,activation='softmax',kernel_initializer=tf.keras.initializers.HeNormal(seed=30))(dlay)

    model = Model(inputs=input_layer,outputs=output_layer)
    return model

model = None
model = create_model_2()


model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss = tf.keras.losses.BinaryCrossentropy(), metrics=[tf.keras.metrics.AUC(),tfa.metrics.F1Score(num_classes = 3,

```



In [361]:
```
model.summary()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_24 (InputLayer)       [(None, 1000)]            0

 embedding_23 (Embedding)    (None, 1000, 300)         28200

 conv1d_152 (Conv1D)         (None, 996, 32)           48032

 conv1d_153 (Conv1D)         (None, 994, 32)           3104

 max_pooling1d_46 (MaxPoolin  (None, 497, 32)          0
 g1D)

 conv1d_154 (Conv1D)         (None, 495, 8)            776

 conv1d_155 (Conv1D)         (None, 491, 16)           656

 max_pooling1d_47 (MaxPoolin  (None, 245, 16)          0
 g1D)

 flatten_23 (Flatten)        (None, 3920)              0

 dropout_23 (Dropout)        (None, 3920)              0

 dense_46 (Dense)            (None, 40)                156840

 dense_47 (Dense)            (None, 20)                820

=================================================================
Total params: 238,428
Trainable params: 210,228
Non-trainable params: 28,200
_________________________________________________________________
```

In [362]:

```
model.fit(padded_docs_char, y_train, batch_size=16, epochs =5, validation_split=0.2, callbacks =[tensorboard_callback] )
```

```
Epoch 1/5
754/754 [==============================] - 87s 113ms/step - loss: 0.2044 - auc_35: 0.5099 - f1_score: 0.0532 - val_loss: 0.1995 - va
l_auc_35: 0.5121 - val_f1_score: 0.0461
Epoch 2/5
754/754 [==============================] - 90s 119ms/step - loss: 0.1996 - auc_35: 0.5083 - f1_score: 0.0520 - val_loss: 0.1987 - va
l_auc_35: 0.5110 - val_f1_score: 0.0514
Epoch 3/5
754/754 [==============================] - 85s 112ms/step - loss: 0.1992 - auc_35: 0.5116 - f1_score: 0.0511 - val_loss: 0.1993 - va
l_auc_35: 0.5151 - val_f1_score: 0.0558
Epoch 4/5
754/754 [==============================] - 87s 116ms/step - loss: 0.1990 - auc_35: 0.5183 - f1_score: 0.0524 - val_loss: 0.1991 - va
l_auc_35: 0.5085 - val_f1_score: 0.0561
Epoch 5/5
754/754 [==============================] - 83s 110ms/step - loss: 0.1987 - auc_35: 0.5290 - f1_score: 0.0602 - val_loss: 0.1996 - va
l_auc_35: 0.5043 - val_f1_score: 0.0478
```

Out[362]:

```
<keras.callbacks.History at 0x151e99ebb20>
```

In [363]:
```
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 8704), started 18:13:22 ago. (Use '!kill 8704' to kill it.)
```

tensorboard analysis for both model

Link :- https://docs.google.com/document/d/1z74uiJtv1MOyFIt3SJRuVwK8LwjD0teXdg-_4I92T7E/edit?usp=sharing (https://docs.google.com/document/d/1z74uiJtv1MOyFIt3SJRuVwK8LwjD0teXdg-_4I92T7E/edit?usp=sharing)