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, in that you will 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.

In [1]:
```
# changing my jupyter theme
!jt -r
```

```
Reset css and font defaults in:
C:\Users\sai\.jupyter\custom &
C:\Users\sai\AppData\Roaming\jupyter\nbextensions
```

Read the Data zip file

In [ ]:
```
# Load the zip/rar file into colab
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving documents.zip to documents.zip
```

In [ ]:
```
# To unrar/unzip the rar/zip files of documents
!unzip "/content/documents.zip" -d "/content/output_folder"
```

Obtaining the class labels & the count distribution

In [ ]:

```
# extract class labels from the documents file name
import os ;
documents_path = '/content/output_folder/documents'
allfiles = sorted(os.listdir(documents_path))
print(allfiles[0])
print(type(allfiles))
print(len(allfiles))

# obtaining class labels
class_labels = []
for j in allfiles:
    class_labels.append(j.split("_")[0])
print(class_labels[0])
```

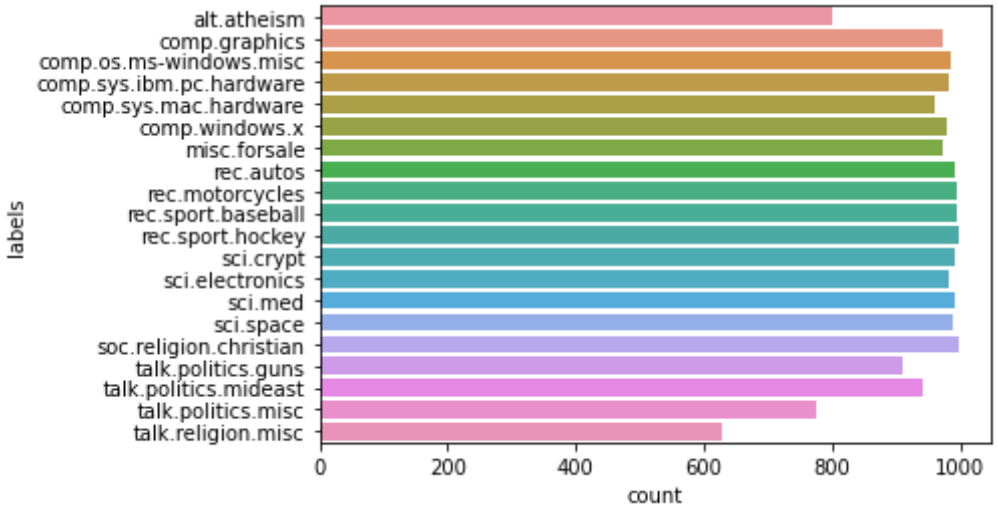
```
alt.atheism_49960.txt
<class 'list'>
18828
alt.atheism
```

In [ ]:

```
# lets create dataframe of class labels
import pandas as pd
data_labels = pd.DataFrame(class_labels,columns = ['labels'])
data_labels.to_csv('text_classlabels.csv')
```

In [ ]:

```
#count plot of labels
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(y = 'labels',data = data_labels,orient='h')
plt.show()
```



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?

Preprocessing:

useful links: 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.

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://stackoverflow.com/a/31837224/4084039
http://www.nltk.org/howto/tree.html
https://stackoverflow.com/a/44294377/4084039

In [ ]:

```
#i am living in the New York
print("i am living in the New York -->",list(chunks))
print(" ")
print("-"*50)
print(" ")
#My name is Srikanth Varma
print("My name is Srikanth Varma -->",list(chunks1))
```

```
i am living in the New York --> [('i', 'NN'), ('am', 'VBP'), ('living', 'VBG'), ('in', 'IN'), ('the', 'DT'), Tree('GP
E', [('New', 'NNP'), ('York', 'NNP')])]

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

My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERSON', [('Srikanth', 'NNP'),
('Varma', 'NNP')])]
```

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.

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.

Obtain file/doc contents

In [ ]:
```
# SOME CUSTOM TERMINOLOGIES
# f_contents - > all document contents
# mail_lst - > preprocessed mails
# fg - > each document's mail contents
# sub_lst - > subject details of each document
```

In [ ]:
```
# obtain all text data into f_contents list
import re
import os ;
documents_path = '/content/output_folder/documents'
allfiles = sorted(os.listdir(documents_path))
f_contents = []
for j in allfiles:
    with open(documents_path+'/'+j,'r',errors = 'ignore') as f:
        f_contents.append(f.read())

print("Total documents :- ",len(f_contents))
```

```
Total documents :-  18828
```

In [ ]:
```
# create a pandas series df from class labels & text data
import pandas as pd
```

```
df1 = pd.DataFrame(list(zip(f_contents,class_labels)),columns = ['text','labels'])
df1.head()
```

Out[ ]:

	text	labels
0	From: mathew <mathew@mantis.co.uk>\nSubject: A...	alt.atheism
1	From: mathew <mathew@mantis.co.uk>\nSubject: A...	alt.atheism
2	From: I3150101@dbstu1.rz.tu-bs.de (Benedikt Ro...	alt.atheism
3	From: mathew <mathew@mantis.co.uk>\nSubject: R...	alt.atheism
4	From: strom@Watson.Ibm.Com (Rob Strom)\nSubjec...	alt.atheism

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

In [ ]:

```
# required packages:-
nltk.download('popular');
```

In [ ]:

```
# MAIN
# check for any single doc/text data
import re; import nltk ;
from nltk import word_tokenize,pos_tag,ne_chunk;

#nltk.download('popular'); #--> run just one time

# PLEASE DO CHECK FOR ANY SINGLE DOCUMENT BY CHANGING INDEX NUMBER IN f_contents[]
#print("BEFORE PREPROCESSING:-\n",f_contents[40])

def preprocess_text(input_text):
    # emails
    fg = re.findall(r"[a-zA-Z0-9._-]+@[a-zA-z0-9._-]+",input_text) # list
    mail_lst = []
    empty = ''
    for each_mail in fg:
        temp = each_mail.split('@')[1].split('.')
        for word in temp:
            if (len(word) > 2) and (word != 'com') and \
```

```
            (word != 'Com') and (word != 'COM'):
                empty = empty + ' ' + word
    mail_lst.append(empty.strip()) # we get preprocessed mail

    replaced = (re.sub(r"[a-zA-Z0-9._-]+@[a-zA-z0-9._-]+",'',input_text))

    #subject extract
    sub_lst = []
    gt = re.findall(r'Subject:\s(.*)',replaced,flags = re.I)
    #pint(gt)  # ['DeltaPoints Animated Desktop']
    try :
        ind = gt[0].rindex(':')
        gta = gt[0][ind+1:].strip()
        gtb = re.findall(r'(\w+)',gta)
        gtb = str(" ".join(gtb))
        sub_lst.append(str(gtb)) # subject details extracted
    except:
        gtb = re.findall(r'(\w+)',gt[0])
        gtb = str(" ".join(gtb))
        sub_lst.append(str(gtb)) # subject details extracted

    # Replace all the subject & from (texts) in the original text by space:-
    replaced = re.sub(r'Subject:\s(.*)','',replaced,flags = re.I)
    replaced = re.sub(r'From:\s(.*)','',replaced,flags = re.I)

    # general cleaning of text data
    replaced = re.sub(r'(\<.*\>)','',replaced)
    replaced = re.sub(r'(\(.*\))','',replaced)
    replaced = re.sub('(\s+)',' ',replaced)
    replaced = re.sub('\w+\:+','',replaced)
    replaced = re.sub('[\,\!\/\)\(\-\<\.\?\>\@\*\_\#\;\|\+\=\$\&\:\"]+','',replaced)
    phrase = re.sub(r"won't", "will not",replaced,flags = re.I)
    phrase = re.sub(r"can\'t", "can not",phrase,flags = re.I)
    phrase = re.sub(r"n\'t", " not", phrase,flags = re.I)
    phrase = re.sub(r"\'re", " are", phrase,flags = re.I)
    phrase = re.sub(r"\'s", " is", phrase,flags = re.I)
    phrase = re.sub(r"\'d", " would", phrase,flags = re.I)
    phrase = re.sub(r"\'ll", " will", phrase,flags = re.I)
    phrase = re.sub(r"\'t", " not", phrase,flags = re.I)
    phrase = re.sub(r"\'ve", " have", phrase,flags = re.I)
    replaced = re.sub(r"\'m", " am", phrase,flags = re.I)
```

```
    # chunking
    replaced = " ".join(replaced.split()).strip()
    chunk = ne_chunk(pos_tag(word_tokenize(replaced)),binary = False)
    #print(chunk)
    named_entities = []
    for tagged_tree in chunk:
    #print(tagged_tree)
        if hasattr(tagged_tree,'label'):
            entity_name = ' '.join(c[0] for c in tagged_tree.leaves()) #
            entity_type = tagged_tree.label() # get NE category
            named_entities.append((entity_name,entity_type))
    #print(named_entities)
    for tag in named_entities:
        if tag[1]=='GPE':   #Specify any tag which is required
            replaced = replaced.replace(tag[0],tag[0].replace(' ','_'))
        if tag[1]=='PERSON':
            replaced = replaced.replace(tag[0],'')

    replaced = re.sub('\d+','',replaced) # remove digits
    # underscore pre-processing
    gh = re.findall('(\_\w+\_)',replaced)
    #print(gh)
    jk = re.findall('(\w+\_+\w+)',replaced)
    for m,n in enumerate(jk):
        if len(n.split('_')[0]) < 2 or len(n.split('_')[0]) == 2:
            replaced = re.sub(jk[m],n.split('_')[1],replaced,re.I)
    for m,n in enumerate(gh):
        replaced = re.sub(gh[m],gh[m].replace('_',''),replaced,re.I)

    #lowercase & length preprocessing (>=15 & <=2)
    replaced = replaced.lower()
    hj = re.findall(r"\w+",replaced)
    temp = ''
    for o in hj:
        if (len(o) > 2) and (len(o) < 15):
            temp = temp + ' ' + o
    replaced = temp

    #final preprocessing:-
    replaced = re.sub('\W+',' ',replaced)
    replaced = " ".join(replaced.split()).strip()
```

```
    return (mail_lst,sub_lst,replaced) # return results

# PLEASE DO CHECK FOR ANY SINGLE DOCUMENT BY CHANGING INDEX NUMBER IN f_contents[]
df2 = pd.DataFrame(columns=['preprocessed_emails','preprocessed_subject','preprocessed_text'])

for index,f in enumerate(f_contents):
    result = preprocess_text(f)
    df2 = df2.append({'preprocessed_emails' : result[0][0],
                      'preprocessed_subject' : result[1][0],
                      'preprocessed_text' : result[2]},ignore_index=True)
print(type(df2))
print(df2.shape)
print(df2.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
(18828, 3)
Index(['preprocessed_emails', 'preprocessed_subject', 'preprocessed_text'], dtype='object')
```

In [ ]:

```
# lets display our data
df2.head()
```

Out[ ]:

	preprocessed_emails	preprocessed_subject	preprocessed_text
0	mantis netcom mantis	Atheist Resources	archive resources last december atheist resour...
1	mantis mantis mantis	Introduction to Atheism	archive introduction last april begin pgp sign...
2	dbstu1 tu-bs mimsy umd edu umd edu	Gospel Dating	article has quite different not necessarily mo...
3	mantis kepler unh edu	university violating separation of church state	recently ras have been ordered and none have r...
4	Watson Ibm harder ccr-p ida org harder ccr-p i...	soc motss et al Princeton axes matching funds ...	article however hate economic terrorism and po...

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 [ ]:

```
# lets call the preprocess function with just one text data :-
documents_path = '/content/output_folder/documents'
text_sample = 'alt.atheism_49960.txt'
p = [] #append the text of text_sample
with open(documents_path+'/'+text_sample,'r',errors = 'ignore') as f:
      p.append(f.read())
print(p)
print("Preprocessed email :-\n",preprocess_text(p[0])[0])
print("Preprocessed subject :-\n",preprocess_text(p[0])[1])
print("Preprocessed text :-\n",preprocess_text(p[0])[2])
```

```
['From: mathew <mathew@mantis.co.uk>\nSubject: Alt.Atheism FAQ: Atheist Resources\n\nArchive-name: atheism/resources
\nAlt-atheism-archive-name: resources\nLast-modified: 11 December 1992\nVersion: 1.0\n\n
Atheist Resources\n\n                      Addresses of Atheist Organizations\n\n
USA\n\nFREEDOM FROM RELIGION FOUNDATION\n\nDarwin fish bumper stickers and assorted other atheist paraphernalia are\n
available from the Freedom From Religion Foundation in the US.\n\nWrite to:  FFRF, P.O. Box 750, Madison, WI 53701.\n
Telephone: (608) 256-8900\n\nEVOLUTION DESIGNS\n\nEvolution Designs sell the "Darwin fish".  It\'s a fish symbol, lik
e the ones\nChristians stick on their cars, but with feet and the word "Darwin" written\ninside.  The deluxe moulded
3D plastic fish is $4.95 postpaid in the US.\n\nWrite to:  Evolution Designs, 7119 Laurel Canyon #4, North Hollywoo
d,\n          CA 91605.\n\nPeople in the San Francisco Bay area can get Darwin Fish from Lynn Gold --\ntry mailing <
figmo@netcom.com>.  For net people who go to Lynn directly, the\nprice is $4.95 per fish.\n\nAMERICAN ATHEIST PRESS\n
\nAAP publish various atheist books -- critiques of the Bible, lists of\nBiblical contradictions, and so on.  One suc
h book is:\n\n"The Bible Handbook" by W.P. Ball and G.W. Foote.  American Atheist Press.\n372 pp.  ISBN 0-910309-26-
4, 2nd edition, 1986.  Bible contradictions,\nabsurdities, atrocities, immoralities... contains Ball, Foote: "The Bib
le\nContradicts Itself", AAP.  Based on the King James version of the Bible.\n\nWrite to:  American Atheist Press, P.
O. Box 140195, Austin, TX 78714-0195.\n      or:  7215 Cameron Road, Austin, TX 78752-2973.\nTelephone: (512) 458-124
4\nFax:       (512) 467-9525\n\nPROMETHEUS BOOKS\n\nSell books including Haught\'s "Holy Horrors" (see below).\n\nWri
te to:  700 East Amherst Street, Buffalo, New York 14215.\nTelephone: (716) 837-2475.\n\nAn alternate address (which
may be newer or older) is:\nPrometheus Books, 59 Glenn Drive, Buffalo, NY 14228-2197.\n\nAFRICAN-AMERICANS FOR HUMANI
SM\n\nAn organization promoting black secular humanism and uncovering the history of\nblack freethought.  They publis
h a quarterly newsletter, AAH EXAMINER.\n\nWrite to:  Norm R. Allen, Jr., African Americans for Humanism, P.O. Box 66
4,\n          Buffalo, NY 14226.\n\n                                United Kingdom\n\nRationalist Press Association
National Secular Society\n88 Islington High Street              702 Holloway Road\nLondon N1 8EW
London N19 3NL\n071 226 7251                          071 272 1266\n\nBritish Humanist Association          South P
lace Ethical Society\n14 Lamb\'s Conduit Passage              Conway Hall\nLondon WC1R 4RH                    Red
Lion Square\n071 430 0908                          London WC1R 4RL\nfax 071 430 1271                    071 831 7
723\n\nThe National Secular Society publish "The Freethinker", a monthly magazine\nfounded in 1881.\n\n
Germany\n\nIBKA e.V.\nInternationaler Bund der Konfessionslosen und Atheisten\nPostfach 880, D-1000 Berlin 41. German
y.\n\nIBKA publish a journal:\nMIZ. (Materialien und Informationen zur Zeit. Politisches\nJournal der Konfessionslose
sn und Atheisten. Hrsg. IBKA e.V.)\nMIZ-Vertrieb, Postfach 880, D-1000 Berlin 41. Germany.\n\nFor atheist books, writ
e to:\n\nIBDK, Internationaler B"ucherdienst der Konfessionslosen\nPostfach 3005, D-3000 Hannover 1. Germany.\nTeleph
one: 0511/211216\n\n\n                              Books -- Fiction\n\nTHOMAS M. DISCH\n\n"The Santa Claus Compromi
se"\nShort story.  The ultimate proof that Santa exists.  All characters and \nevents are fictitious.  Any similarity
to living or dead gods -- uh, well...\n\nWALTER M. MILLER, JR\n\n"A Canticle for Leibowitz"\nOne gem in this post ato
```

mic doomsday novel is the monks who spent their lives\ncopying blueprints from "Saint Leibowitz", filling the sheets of paper with\nink and leaving white lines and letters.\n\nEDGAR PANGBORN\n\n"Davy"\nPost atomic doomsday novel set in clerical states. The church, for example,\nforbids that anyone "produce, describe or use any substance containing...\natoms". \n\nPHILIP K. DICK\n\nPhilip K. Dick Dick wrote many philosophical and thought-provoking short \nstories and novels. His stories are bizarre at times, but very approachable.\nHe wrote mainly SF, but he wrote about people, truth and religion rather than\ntechnology. Although he often believed that he had met some sort of God, he\nremained sceptical. Amongst his novels, the following are of some relevance:\n\n"Galactic Pot-Healer"\nA fallible alien deity summons a group of Earth craftsmen and women to a\nremote planet to raise a giant cathedral from beneath the oceans. When the\ndeity begins to demand faith from the earthers, pot-healer Joe Fernwright is\nunable to comply. A polished, ironic and amusing novel.\n\n"A Maze of Death"\nNoteworthy for its description of a technology-based religion.\n\n"VALIS"\nThe schizophrenic hero searches for the hidden mysteries of Gnostic\nChristianity after reality is fired into his brain by a pink laser beam of\nunknown but possibly divine origin. He is accompanied by his dogmatic and \ndismissively atheist friend and assorted other odd characters.\n\n"The Divine Invasion"\nGod invades Earth by making a young woman pregnant as she returns from\nanother star system. Unfortunately she is terminally ill, and must be \nassisted by a dead man whose brain is wired to 24-hour easy listening music.\n\nMARGARET ATWOOD\n\n"The Handmaid\'s Tale"\nA story based on the premise that the US Congress is mysteriously\nassassinated, and fundamentalists quickly take charge of the nation to set it\n"right" again. The book is the diary of a woman\'s life as she tries to live\nunder the new Christian theocracy. Women\'s right to own property is revoked,\nand their bank accounts are closed; sinful luxuries are outlawed, and the\nradio is only used for readings from the Bible. Crimes are punished\nretroactively: doctors who performed legal abortions in the "old world" are\nhunted down and hanged. Atwood\'s writing style is difficult to get used to\nat first, but the tale grows more and more chilling as it goes on.\n\nVARIOUS AUTHORS\n\n"The Bible"\nThis somewhat dull and rambling work has often been criticized. However, it\nis probably worth reading, if only so that you\'ll know what all the fuss is\nabout. It exists in many different versions, so make sure you get the one\ntrue version.\n\n Books -- Non-fiction\n\nPETER DE ROSA\n\n"Vicars of Christ", Bantam Press, 1988\nAlthough de Rosa seems to be Christian or even Catholic this is a very\nenlighting history of papal immoralities, adulteries, fallacies etc.\n(German translation: "Gottes erste Diener. Die dunkle Seite des Papsttums",\nDroemer-Knaur, 1989)\n\nMICHAEL MARTIN\n\n"Atheism: A Philosophical Justification", Temple University Press,\n Philadelphia, USA.\nA detailed and scholarly justification of atheism. Contains an outstanding\nappendix defining terminology and usage in this (necessarily) tendentious\narea. Argues both for "negative atheism" (i.e. the "non-belief in the\nexistence of god(s)") and also for "positive atheism" ("the belief in the\nnon-existence of god(s)"). Includes great refutations of the most\nchallenging arguments for god; particular attention is paid to refuting\ncontemporary theists such as Platinga and Swinburne.\n541 pages. ISBN 0-87722-642-3 (hardcover; paperback also available)\n\n"The Case Against Christianity", Temple University Press\nA comprehensive critique of Christianity, in which he considers \nthe best contemporary defences of Christianity and (ultimately)\ndemonstrates that they are unsupportable and/or incoherent.\n273 pages. ISBN 0-87722-767-5\n\nJAMES TURNER\n\n"Without God, Without Creed", The Johns Hopkins University Press, Baltimore,\n MD, USA\nSubtitled "The Origins of Unbelief in America". Examines the way in which\nunbelief (whether agnostic or atheistic) became a mainstream alternative\nworld-view. Focusses on the period 1770-1900, and while considering France\nand Britain the emphasis is on American, and particularly New England\ndevelopments. "Neither a religious history of secularization or atheism,\nWithout God, Without Creed is, rather, the intellectual history of the fate\nof a single idea, the belief that God exists." \n316 pages. ISBN (hardcover) 0-8018-2494-X (paper) 0-8018-3407-4\n\nGEORGE SELDES (Editor)\n\n"The great thoughts", Ballantine Books, New York, USA\nA "dictionary of quotations" of a different kind, concentrating on statements\nand writings which, explicitly or implicitly, present the person\'s philosophy\nand world-view. Includes obscure (and often suppressed) opinions from many\npeople. For some popular observations, traces the way in which various\npeople expressed and twisted the idea over the centuries. Quite a number of\nthe quotations are derived from Cardiff\'s "What Great Men Think of Religion"\nand Noyes\' "Views of Rel

igion".\n490 pages. ISBN (paper) 0-345-29887-X.\n\nRICHARD SWINBURNE\n\n"The Existence of God (Revised Edition)", Cla
rendon Paperbacks, Oxford\nThis book is the second volume in a trilogy that began with "The Coherence of\nTheism" (19
77) and was concluded with "Faith and Reason" (1981). In this\nwork, Swinburne attempts to construct a series of ind
uctive arguments for the\nexistence of God. His arguments, which are somewhat tendentious and rely\nupon the imputat
ion of late 20th century western Christian values and\naesthetics to a God which is supposedly as simple as can be co
nceived, were\ndecisively rejected in Mackie\'s "The Miracle of Theism". In the revised\nedition of "The Existence o
f God", Swinburne includes an Appendix in which he\nmakes a somewhat incoherent attempt to rebut Mackie.\n\nJ. L. MAC
KIE\n\n"The Miracle of Theism", Oxford\nThis (posthumous) volume contains a comprehensive review of the principal\nar
guments for and against the existence of God. It ranges from the classical\nphilosophical positions of Descartes, An
selm, Berkeley, Hume et al, through\nthe moral arguments of Newman, Kant and Sidgwick, to the recent restatements\nof
the classical theses by Plantinga and Swinburne. It also addresses those\npositions which push the concept of God be
yond the realm of the rational,\nsuch as those of Kierkegaard, Kung and Philips, as well as "replacements for\nGod" s
uch as Lelie\'s axiarchism. The book is a delight to read - less\nformalistic and better written than Martin\'s work
s, and refreshingly direct\nwhen compared with the hand-waving of Swinburne.\n\nJAMES A. HAUGHT\n\n"Holy Horrors: An
Illustrated History of Religious Murder and Madness",\n Prometheus Books\nLooks at religious persecution from ancient
times to the present day -- and\nnot only by Christians.\nLibrary of Congress Catalog Card Number 89-64079. 1990.\n\n
NORM R. ALLEN, JR.\n\n"African American Humanism: an Anthology"\nSee the listing for African Americans for Humanism a
bove.\n\nGORDON STEIN\n\n"An Anthology of Atheism and Rationalism", Prometheus Books\nAn anthology covering a wide ra
nge of subjects, including \'The Devil, Evil\nand Morality\' and \'The History of Freethought\'. Comprehensive bibli
ography.\n\nEDMUND D. COHEN\n\n"The Mind of The Bible-Believer", Prometheus Books\nA study of why people become Chris
tian fundamentalists, and what effect it\nhas on them.\n\n   Net Resources\n\nThere\'s a
small mail-based archive server at mantis.co.uk which carries\narchives of old alt.atheism.moderated articles and ass
orted other files. For\nmore information, send mail to archive-server@mantis.co.uk saying\n\n   help\n   send atheis
m/index\n\nand it will mail back a reply.\n\n\nmathew\n\n']
Preprocessed email :-
 ['mantis netcom mantis']
Preprocessed subject :-
 ['Atheist Resources']
Preprocessed text :-
 archive resources last december atheist resources addresses atheist organizations usa freedom from religion foundati
on fish bumper stickers and assorted other atheist paraphernalia are available from the freedom from religion foundat
ion the write ffrf box madison evolution designs evolution designs sell the fish fish symbol like the ones stick thei
r cars but with feet and the word written inside the deluxe moulded plastic fish postpaid the write evolution designs
north hollywood people the san francisco bay area can get fish from try mailing for net people who directly the price
per fish american atheist press aap publish various atheist books critiques the bible lists biblical contradictions a
nd one such book the bible handbook ball and foote american atheist press isbn edition bible contradictions absurditi
es atrocities immoralities contains ball the bible contradicts itself aap based the king version the bible write amer
ican atheist press box cameron road prometheus books sell books including write east amherst street york alternate ad
dress prometheus books for humanism organization promoting black secular humanism and uncovering the history black fr
eethought they publish quarterly newsletter aah examiner write norm allen african americans for humanism box united p
ress association national secular society street holloway road london london british humanist association south place
ethical society lamb wcr red lion square london wcr fax the national secular society publish the freethinker monthly
magazine founded germany_ibka bund der und berlin germany_ibka publish miz materialien und zur zeit politisches journ
al der und ibka mizvertrieb postfach berlin germany for atheist books write ibdk bucherdienst der hannover germany bo
oks fiction thomas disch the claus compromise short story the ultimate proof that exists all characters and events ar

e fictitious any similarity living dead gods well walter canticle for leibowitz one gem this post atomic doomsday nov
el the monks who spent their lives copying blueprints from filling the sheets paper with ink and leaving white lines
and letters edgar pangborn atomic doomsday novel set clerical states the church for example forbids that anyone produ
ce describe use any substance containing atoms philip dick wrote many philosophical and short stories and novels his
stories are bizarre times but very approachable wrote mainly but wrote about people truth and religion rather than te
chnology although often believed that had met some sort remained sceptical amongst his novels the following are some
galactic pothealer fallible alien deity summons group craftsmen and women remote planet raise giant cathedral from be
neath the oceans when the deity begins demand faith from the earthers pothealer unable comply polished ironic and amu
sing novel maze for its description religion valis the schizophrenic hero searches for the hidden mysteries gnostic i
ty after reality fired into his brain pink laser beam unknown but possibly divine origin accompanied his dogmatic and
dismissively atheist friend and assorted other odd characters the divine invasion invades making young woman pregnant
she returns from another star system unfortunately she terminally ill and must assisted dead man whose brain wired ho
ur easy listening music margaret atwood the handmaid story based the premise that the congress mysteriously assassina
ted and quickly take charge the nation set right again the book the diary woman life she tries live under the new the
ocracy women right own property revoked and their bank accounts are closed sinful luxuries are outlawed and the radio
only used for readings from the bible crimes are punished doctors who performed legal abortions the old world are hun
ted down and hanged writing style difficult get used first but the tale grows more and more chilling goes various aut
hors the bible this somewhat dull and rambling work has often been criticized however probably worth reading only tha
t you will know what all the fuss about exists many different versions make sure you get the one true version peter r
osa vicars christ bantam press although seems even catholic this very enlighting history papal immoralities adulterie
s fallacies etc german gottes erste dunkle seite des droemerknaur michael martin philosophical justification temple u
niversity press philadelphia usa detailed and scholarly justification atheism contains outstanding appendix defining
terminology and usage this tendentious area both for negative atheism the nonbelief the existence god and also for po
sitive atheism the belief the nonexistence god includes great refutations the most challenging arguments for god part
icular attention paid refuting contempory theists such and pages isbn the case against ity temple university press co
mprehensive critique ity which considers the best contemporary defences ity and demonstrates that they are unsupporta
ble andor incoherent pages isbn james turner without the johns hopkins university press baltimore usa subtitled the o
rigins unbelief america examines the way which unbelief became mainstream alternative worldview focusses the period a
nd while considering france and britain the emphasis american and particularly new_england developments neither relig
ious history secularization atheism without rather the intellectual history the fate single idea the belief that exis
ts pages isbn george seldes the great thoughts usa dictionary quotations different kind concentrating statements and
writings which explicitly implicitly present the person philosophy and worldview includes obscure opinions from many
people for some popular observations traces the way which various people expressed and twisted the idea over the cent
uries quite number the quotations are derived from cardiff what religion and views religion pages isbn richard swinbu
rne the existence clarendon paperbacks oxford this book the second volume trilogy that began with the coherence theis
m this work attempts construct series inductive arguments for the existence his arguments which are somewhat tendenti
ous and rely upon the imputation late century western values and aesthetics which supposedly simple can conceived wer
e decisively rejected mackie the miracle theism the revised edition the existence includes appendix which makes somew
hat 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 through the moral arguments
newman kant and the recent restatements the classical theses and also addresses those positions which push the concep
t beyond the realm the rational such those kierkegaard kung and well replacements for such axiarchism the book deligh
t read less formalistic and better written than works and refreshingly direct when compared with the handwaving james
haught holy illustrated history religious murder and madness prometheus books looks religious persecution from ancien
t times the present day and not only library congress catalog card number norm allen african american the listing for

```
african americans for humanism above gordon stein anthology atheism and anthology covering wide range subjects includ
ing nothe and morality and nothe history freethought comprehensive bibliography edmund cohen the mind the biblebeliev
er prometheus books study why people become and what effect has them net resources there small mailbased archive serv
er mantiscouk which carries archives old articles and assorted other files for more information send mail saying help
send atheismindex and will mail back reply mathew
```

After writing Preprocess function, call the function for each of the document(18828 docs) and then display the dataframe as mentioned in the assignment.

In [ ]:
```
# Display the final dataframe using concat()
final_df = pd.concat([df1,df2],axis = 1)
final_df.head()
```

Out[ ]:

	text	labels	preprocessed_emails	preprocessed_subject	preprocessed_text
0	From: mathew <mathew@mantis.co.uk>\nSubject: A...	alt.atheism	mantis netcom mantis	Atheist Resources	archive resources last december atheist resour...
1	From: mathew <mathew@mantis.co.uk>\nSubject: A...	alt.atheism	mantis mantis mantis	Introduction to Atheism	archive introduction last april begin pgp sign...
2	From: I3150101@dbstu1.rz.tu-bs.de (Benedikt Ro...	alt.atheism	dbstu1 tu-bs mimsy umd edu umd edu	Gospel Dating	article has quite different not necessarily mo...
3	From: mathew <mathew@mantis.co.uk>\nSubject: R...	alt.atheism	mantis kepler unh edu	university violating separation of church state	recently ras have been ordered and none have r...
4	From: strom@Watson.Ibm.Com (Rob Strom)\nSubjec...	alt.atheism	Watson Ibm harder ccr-p ida org harder ccr-p i...	soc motss et al Princeton axes matching funds ...	article however hate economic terrorism and po...

In [ ]:
```
# to check any null values
final_df.isnull().sum()
```

Out[ ]:
```
text                  0
labels                0
preprocessed_emails   0
preprocessed_subject  0
preprocessed_text     0
dtype: int64
```

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 if you don't know how to plot the model with shapes.

Final_text data for the models :-

In [ ]:
```
# Obtain the data for modelling:-
df2['final_txt'] = df2[['preprocessed_text', 'preprocessed_subject','preprocessed_emails']].apply(lambda x: ' '.join(
df2.drop(columns = ['preprocessed_text','preprocessed_subject','preprocessed_emails'],inplace = True)
df2.to_csv('final_preprocessed_text.csv')
df2.head()
```

Out[ ]:

	final_txt
0	archive resources last december atheist resour...
1	archive introduction last april begin pgp sign...
2	article has quite different not necessarily mo...
3	recently ras have been ordered and none have r...
4	article however hate economic terrorism and po...

In [ ]:
```
# to check gpu is running or not
import tensorflow as tf
tf.test.gpu_device_name()
```

Out[ ]:
```
'/device:GPU:0'
```

In [ ]:
```
# Load the classlabels file into colab
from google.colab import files
files.upload()  # upload text_classlabels
# NOTE : As the uploaded file is very big, I cleared the output because of easy readeability purposes
```

In [ ]:
```
# Load the final_csv file into colab
from google.colab import files
files.upload() # upload final_preprocessed_text.csv
# NOTE : As the uploaded file is very big, I cleared the output because of easy readeability purposes
```

Read the Final preprocessed datasets

In [2]:
```
# lets read the dataset :-
import pandas as pd
df1 = pd.read_csv('text_classlabels.csv')
df2 = pd.read_csv('final_preprocessed_text.csv')
```

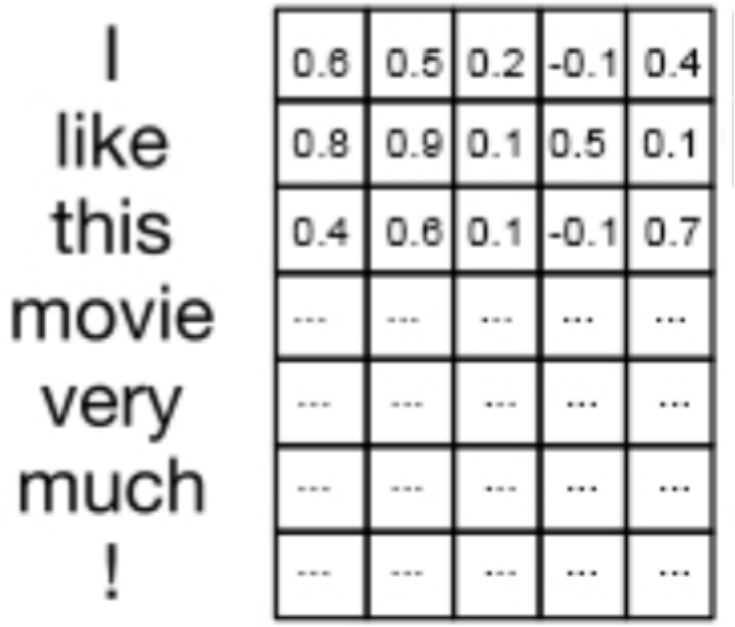
In [3]:
```
# Train test split is done
from sklearn.model_selection import train_test_split
y = df1['labels']
X = df2['final_txt']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=170,stratify=y)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [4]:
```
# lets convert labels to categorical
from tensorflow.keras.utils import to_categorical
ytrain_labels = pd.get_dummies(y_train)
ytest_labels = pd.get_dummies(y_test)
```

TASK 1

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

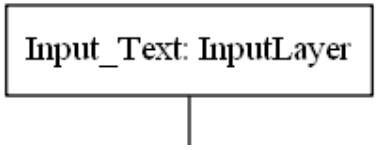


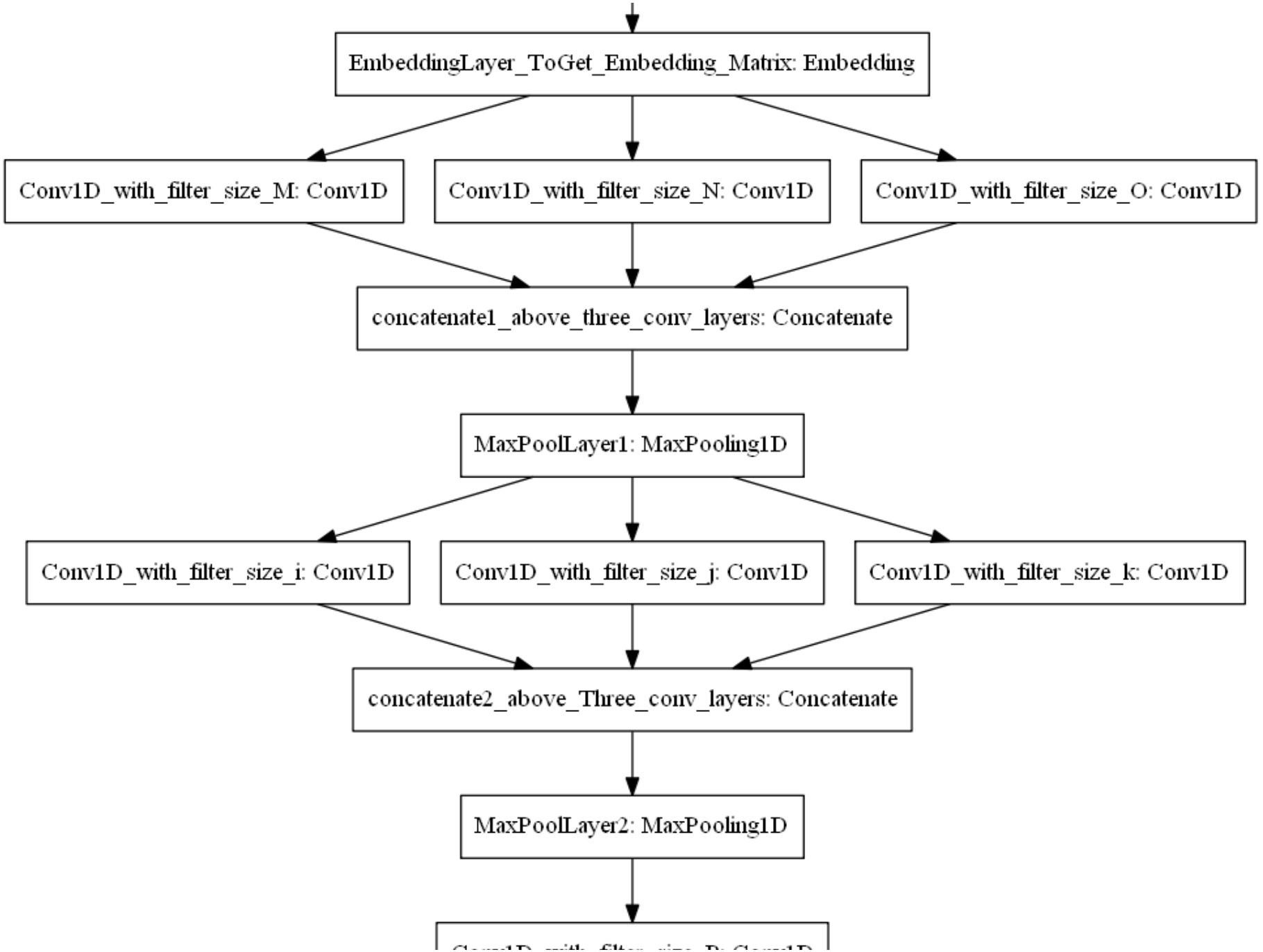
Ref: https://i.imgur.com/kiVQuk1.png

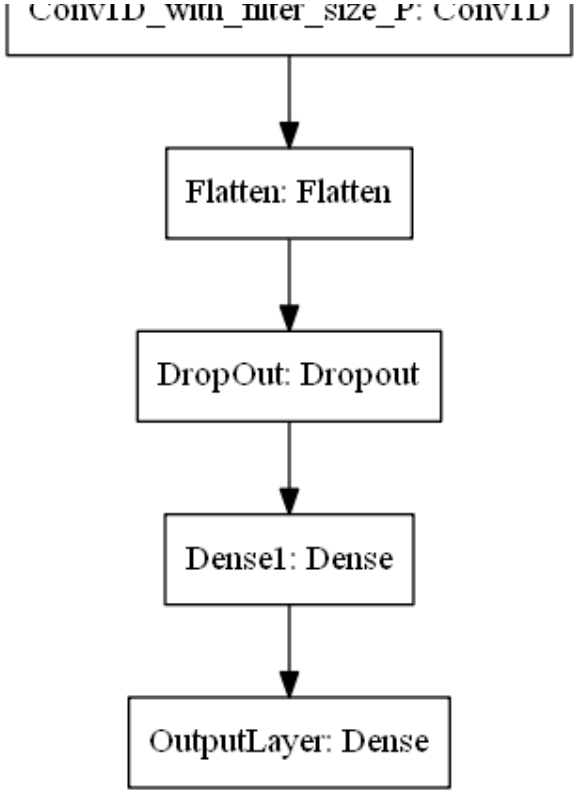
Reference:
https://stackoverflow.com/a/43399308/4084039
https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

How EMBEDDING LAYER WORKS

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/







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.
```

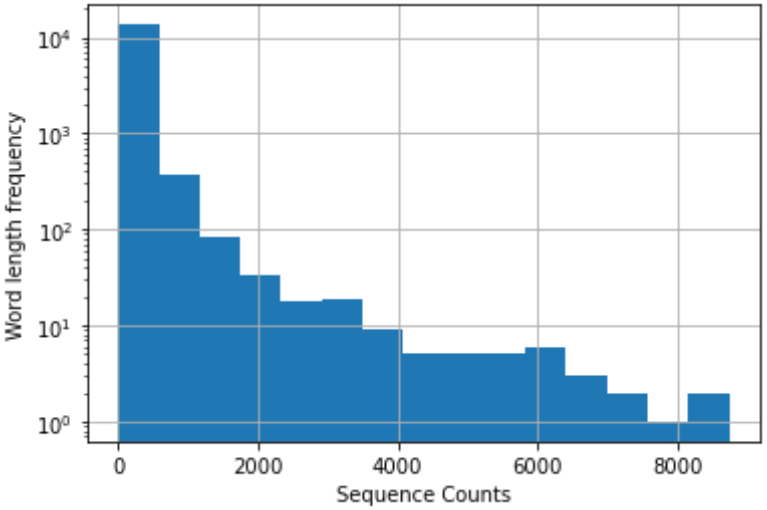
Tokenizing & Padding for MODEL -1

In [5]:

```
# reference : https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html
# lets pad & tokenize the data:-
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

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

# Get our training data word index
word_index = tokenizer.word_index
print("Length of word indices :-",len(word_index))
print('\n')
# Encode training data sentences into sequences
train_sequences = tokenizer.texts_to_sequences(X_train)

# Get max training sequence length
import statistics ; import matplotlib.pyplot as plt
lengths = [len(x) for x in train_sequences]
print("The mean length of words in the sequences :-\n",statistics.mean(lengths))
plt.hist(lengths,bins = 15)
plt.xlabel('Sequence Counts')
plt.ylabel('Word length frequency')
plt.grid()
plt.yscale('log')
plt.show()

# count of lengths
count_gr600 = 0
count_ls600 = 0
for j in lengths:
    if j > 600:
        count_gr600 += 1
    else:
        count_ls600 += 1

print("No of sentences having words > 600 :- \n",count_gr600)
print("No of sentences having words < 600 :- \n",count_ls600)
print("Contribution of lengthier sentences having > 600 words :-\n",(count_gr600)/len(lengths))

# Pad the training sequences
```

```
train_padded = pad_sequences(train_sequences,padding = 'post',maxlen= 600,truncating='post')

# tokenize & Pad the testing sequences
test_sequences = tokenizer.texts_to_sequences(X_test)
test_padded = pad_sequences(test_sequences,padding='post',maxlen = 600,truncating='post')

# Output the results of our work
print("\nPadded training shape:", train_padded.shape)
print("Training sequences data type:", type(train_sequences))
print("Padded Training sequences data type:", type(train_padded))

print("\nPadded testing shape:", test_padded.shape)
print("Testing sequences data type:", type(test_sequences))
print("Padded Testing sequences data type:", type(test_padded))
```

```
Length of word indices :- 111511


The mean length of words in the sequences :-
 187.86700658593585
```




```
No of sentences having words > 600 :-
 543
No of sentences having words < 600 :-
 13578
Contribution of lengthier sentences having > 600 words :-
 0.03845336732526025
```

```
Padded training shape: (14121, 600)
Training sequences data type: <class 'list'>
Padded Training sequences data type: <class 'numpy.ndarray'>

Padded testing shape: (4707, 600)
Testing sequences data type: <class 'list'>
Padded Testing sequences data type: <class 'numpy.ndarray'>
```

Observation :

From the results of tokenizing the sequence, we can understand that only 3% of total sentences contribute for greater than 600 words & also its clearly evident from the histogram plotted. therefore i chose padding length to be 600.

In [ ]:
```
# Load the glove zip/rar file into colab
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving glove.6B.50d.txt.zip to glove.6B.50d.txt.zip
```

In [ ]:
```
# To unrar/unzip the rar/zip files of documents
!unzip "/content/glove.6B.50d.txt.zip" -d "/content/output_folder"
```

```
Archive:  /content/glove.6B.50d.txt.zip
   creating: /content/output_folder/glove.6B.50d.txt/
  inflating: /content/output_folder/glove.6B.50d.txt/glove.6B.50d.txt
```

Model-1: Creating word embedding layer

In [ ]:
```
# reference :- https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

# create map word indices to embeddings/vectors
import numpy as np
embeddings_index = {}
with open('/content/output_folder/glove.6B.50d.txt/glove.6B.50d.txt') as f:
    for line in f:
        values = line.split()
```

```
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

print('No of word vectors :-\n',len(embeddings_index))
emb_dim = coefs.shape[0]
print("Dimension of word vector :-\n",emb_dim)

# embedding matrix
embeddings_matrix = np.zeros((len(word_index) + 1,emb_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embeddings_matrix[i] = embedding_vector

# embedding layer
from tensorflow.keras.layers import Embedding
embedding_layer = Embedding(len(word_index) + 1,
                            emb_dim,
                            weights = [embeddings_matrix],
                            input_length = 600,
                            trainable = False)

print('shape of embedding_matrix :-\n',embeddings_matrix.shape)
```

```
No of word vectors :-
 400000
Dimension of word vector :-
 50
shape of embedding_matrix :-
 (111512, 50)
```

In [ ]:

```
# for add-on metrics we need this library
!pip install tensorflow-addons
```

```
Collecting tensorflow-addons
  Downloading https://files.pythonhosted.org/packages/66/4b/e893d194e626c24b3df2253066aa418f46a432fdb68250cde14bf9bb0
700/tensorflow_addons-0.13.0-cp37-cp37m-manylinux2010_x86_64.whl (679kB)
     |████████████████████████████████| 686kB 3.8MB/s
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/dist-packages (from tensorflow-addons) (2.
7.1)
```

```
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.13.0
```

Tensorboard results

In [ ]:
```
# %load_ext tensorboard
%reload_ext tensorboard
```

In [ ]:
```
# Clear any logs of previous executions/runs
import shutil
#/content/best_model_1 #./logfiles_model1
path='./logfiles_model1'
shutil.rmtree(path,ignore_errors = True)
```

In [ ]:
```
import shutil
#/content/best_model_1 #./logfiles_model1
path='/content/best_model_1'
shutil.rmtree(path,ignore_errors=True)
```

In [ ]:
```
!kill 558
```

In [ ]:
```
# Tensorboard outputs
import os;import datetime ; #Reference :--> https://www.dlology.com/blog/how-to-run-tensorboard-in-jupyter-notebook/
logs_base_dir = "./logfiles_model1"

os.makedirs(logs_base_dir,exist_ok=True)# Creating a directory to store logs
# Launch the tensorboard  & then refresh it after executing the model.
%tensorboard --logdir {logs_base_dir}
```

In [ ]:
```
# Decay learning rate scheduler
import tensorflow
def scheduler(epoch,lr):
    if ((epoch) % 3 == 0):
        lr = ((lr)-(lr*(0.05))) # 0.05
```

```
    else:
        pass

    return lr
```

APPLY MODEL 1 & DISPLAY RESULTS

In [ ]:

```
# functional API
import tensorflow as tf
tf.keras.backend.clear_session()# For easy reset of notebook state
tf.random.set_seed(1234)
import tensorflow_addons as tfa ; import os
from tensorflow.keras import layers,Input
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dropout,Flatten,Dense,Conv1D,MaxPooling1D,BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.initializers import HeNormal

# inputs
x = Input(shape=(600,))
initializer = HeNormal(seed = 103)
embedded_sequences = embedding_layer(x)

x1 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(embedded_sequences)
x2 = Conv1D(32,3,activation='relu',padding = 'same',kernel_initializer = initializer)(embedded_sequences)
x3 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(embedded_sequences)

concat_x1 = layers.concatenate([x1,x2,x3],axis = -1)

y = MaxPooling1D(3,strides = 2)(concat_x1)

y1 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(y)
y2 = Conv1D(32,3,activation='relu',padding = 'same',kernel_initializer = initializer)(y)
y3 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(y)

concat_x2 = layers.concatenate([y1,y2,y3],axis = -1)
```

```
z = MaxPooling1D(3,strides = 2)(concat_x2)

h1 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(z)
h2 = Conv1D(32,3,activation='relu',padding = 'same',kernel_initializer = initializer)(z)
h3 = Conv1D(28,3,activation='relu',padding = 'same',kernel_initializer = initializer)(z)

concat_x3 = layers.concatenate([h1,h2,h3],axis = -1)

h = MaxPooling1D(3,strides = 2)(concat_x3)

z1 = Conv1D(24,4,activation='relu',padding = 'same',kernel_initializer = initializer)(h)

z2 = Flatten()(z1)
z3 = Dropout(0.3)(z2)
z4 = Dense(32, activation='relu',kernel_initializer = initializer)(z3)
z5 = Dense(32, activation='relu',kernel_initializer = initializer)(z4)
z6 = Dense(20,activation='softmax',kernel_initializer = 'glorot_normal')(z5)

# model
model = Model(inputs = x,outputs = z6)

# optimizer
optimizer = Adam(learning_rate = 0.001)

#save model at every epoch end if validation accuracy is improved
filepath = "best_model_1/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath,monitor='val_acc',verbose=1,save_best_only=True,mode='auto')

#early stopping
earlystop = EarlyStopping(monitor='val_acc',patience=2,verbose=1)

#tensorboard callbacks # reference :-:--> https://www.dlology.com/blog/how-to-run-tensorboard-in-jupyter-notebook/
log_dir = os.path.join(logs_base_dir,datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tb_callback = TensorBoard(log_dir=log_dir,histogram_freq=1)

# Lr schedule
lrschedule = LearningRateScheduler(scheduler,verbose = 1)

# metric
f1score = tfa.metrics.F1Score(num_classes = 20,average = 'micro')

# Model compile
```

```
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['acc',f1score])
# Model fit
model.fit(train_padded,ytrain_labels, validation_data=(test_padded,ytest_labels),
          epochs=40, batch_size=64,callbacks = [earlystop,checkpoint,lrschedule,tb_callback])
```

```
Epoch 1/40

Epoch 00001: LearningRateScheduler reducing learning rate to 0.0009500000451225787.
221/221 [==============================] - ETA: 0s - loss: 2.6579 - acc: 0.1270 - f1_score: 0.1270
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/metrics.py:257: UserWarning: Metric F1Score implements
a `reset_states()` method; rename it to `reset_state()` (without the final "s"). The name `reset_states()` has been d
eprecated to improve API consistency.
  'consistency.' % (self.__class__.__name__,))
221/221 [==============================] - 12s 42ms/step - loss: 2.6579 - acc: 0.1270 - f1_score: 0.1270 - val_loss:
2.1459 - val_acc: 0.2233 - val_f1_score: 0.2233

Epoch 00001: val_acc improved from -inf to 0.22328, saving model to best_model_1/weights-01-0.2233.hdf5
Epoch 2/40

Epoch 00002: LearningRateScheduler reducing learning rate to 0.0009500000160187483.
221/221 [==============================] - 7s 32ms/step - loss: 1.9126 - acc: 0.3067 - f1_score: 0.3067 - val_loss:
1.6560 - val_acc: 0.3952 - val_f1_score: 0.3952

Epoch 00002: val_acc improved from 0.22328 to 0.39516, saving model to best_model_1/weights-02-0.3952.hdf5
Epoch 3/40

Epoch 00003: LearningRateScheduler reducing learning rate to 0.0009500000160187483.
221/221 [==============================] - 7s 32ms/step - loss: 1.5747 - acc: 0.4287 - f1_score: 0.4287 - val_loss:
1.4351 - val_acc: 0.4918 - val_f1_score: 0.4918

Epoch 00003: val_acc improved from 0.39516 to 0.49182, saving model to best_model_1/weights-03-0.4918.hdf5
Epoch 4/40

Epoch 00004: LearningRateScheduler reducing learning rate to 0.0009025000152178108.
221/221 [==============================] - 7s 34ms/step - loss: 1.3398 - acc: 0.5229 - f1_score: 0.5229 - val_loss:
1.3026 - val_acc: 0.5509 - val_f1_score: 0.5509

Epoch 00004: val_acc improved from 0.49182 to 0.55088, saving model to best_model_1/weights-04-0.5509.hdf5
Epoch 5/40

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0009025000035762787.
221/221 [==============================] - 7s 31ms/step - loss: 1.1517 - acc: 0.5912 - f1_score: 0.5912 - val_loss:
```

```
1.2488 - val_acc: 0.5730 - val_f1_score: 0.5730

Epoch 00005: val_acc improved from 0.55088 to 0.57298, saving model to best_model_1/weights-05-0.5730.hdf5
Epoch 6/40

Epoch 00006: LearningRateScheduler reducing learning rate to 0.0009025000035762787.
221/221 [==============================] - 7s 32ms/step - loss: 1.0247 - acc: 0.6363 - f1_score: 0.6363 - val_loss:
1.1455 - val_acc: 0.6065 - val_f1_score: 0.6065

Epoch 00006: val_acc improved from 0.57298 to 0.60654, saving model to best_model_1/weights-06-0.6065.hdf5
Epoch 7/40

Epoch 00007: LearningRateScheduler reducing learning rate to 0.0008573750033974648.
221/221 [==============================] - 7s 34ms/step - loss: 0.9183 - acc: 0.6742 - f1_score: 0.6742 - val_loss:
1.1127 - val_acc: 0.6297 - val_f1_score: 0.6297

Epoch 00007: val_acc improved from 0.60654 to 0.62970, saving model to best_model_1/weights-07-0.6297.hdf5
Epoch 8/40

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0008573749801144004.
221/221 [==============================] - 7s 32ms/step - loss: 0.8398 - acc: 0.7012 - f1_score: 0.7012 - val_loss:
1.0648 - val_acc: 0.6401 - val_f1_score: 0.6401

Epoch 00008: val_acc improved from 0.62970 to 0.64011, saving model to best_model_1/weights-08-0.6401.hdf5
Epoch 9/40

Epoch 00009: LearningRateScheduler reducing learning rate to 0.0008573749801144004.
221/221 [==============================] - 7s 31ms/step - loss: 0.7608 - acc: 0.7246 - f1_score: 0.7246 - val_loss:
1.0477 - val_acc: 0.6550 - val_f1_score: 0.6550

Epoch 00009: val_acc improved from 0.64011 to 0.65498, saving model to best_model_1/weights-09-0.6550.hdf5
Epoch 10/40

Epoch 00010: LearningRateScheduler reducing learning rate to 0.0008145062311086804.
221/221 [==============================] - 8s 34ms/step - loss: 0.6859 - acc: 0.7533 - f1_score: 0.7533 - val_loss:
1.0527 - val_acc: 0.6637 - val_f1_score: 0.6637

Epoch 00010: val_acc improved from 0.65498 to 0.66369, saving model to best_model_1/weights-10-0.6637.hdf5
Epoch 11/40

Epoch 00011: LearningRateScheduler reducing learning rate to 0.0008145062020048499.
221/221 [==============================] - 7s 31ms/step - loss: 0.6317 - acc: 0.7669 - f1_score: 0.7669 - val_loss:
1.0784 - val_acc: 0.6586 - val_f1_score: 0.6586

Epoch 00011: val_acc did not improve from 0.66369
```

```
Epoch 12/40

Epoch 00012: LearningRateScheduler reducing learning rate to 0.0008145062020048499.
221/221 [==============================] - 7s 34ms/step - loss: 0.5992 - acc: 0.7796 - f1_score: 0.7796 - val_loss:
1.0554 - val_acc: 0.6792 - val_f1_score: 0.6792

Epoch 00012: val_acc improved from 0.66369 to 0.67920, saving model to best_model_1/weights-12-0.6792.hdf5
Epoch 13/40

Epoch 00013: LearningRateScheduler reducing learning rate to 0.0007737808919046074.
221/221 [==============================] - 7s 32ms/step - loss: 0.5176 - acc: 0.8144 - f1_score: 0.8144 - val_loss:
1.0565 - val_acc: 0.6849 - val_f1_score: 0.6849

Epoch 00013: val_acc improved from 0.67920 to 0.68494, saving model to best_model_1/weights-13-0.6849.hdf5
Epoch 14/40

Epoch 00014: LearningRateScheduler reducing learning rate to 0.000773780862800777.
221/221 [==============================] - 7s 32ms/step - loss: 0.4671 - acc: 0.8301 - f1_score: 0.8301 - val_loss:
1.0962 - val_acc: 0.6936 - val_f1_score: 0.6936

Epoch 00014: val_acc improved from 0.68494 to 0.69365, saving model to best_model_1/weights-14-0.6936.hdf5
Epoch 15/40

Epoch 00015: LearningRateScheduler reducing learning rate to 0.000773780862800777.
221/221 [==============================] - 7s 32ms/step - loss: 0.4348 - acc: 0.8417 - f1_score: 0.8417 - val_loss:
1.1541 - val_acc: 0.6818 - val_f1_score: 0.6818

Epoch 00015: val_acc did not improve from 0.69365
Epoch 16/40

Epoch 00016: LearningRateScheduler reducing learning rate to 0.0007735091819660738 2.
221/221 [==============================] - 7s 32ms/step - loss: 0.3879 - acc: 0.8601 - f1_score: 0.8601 - val_loss:
1.1185 - val_acc: 0.6966 - val_f1_score: 0.6966

Epoch 00016: val_acc improved from 0.69365 to 0.69662, saving model to best_model_1/weights-16-0.6966.hdf5
Epoch 17/40

Epoch 00017: LearningRateScheduler reducing learning rate to 0.0007735091802198439 8.
221/221 [==============================] - 7s 32ms/step - loss: 0.3483 - acc: 0.8739 - f1_score: 0.8739 - val_loss:
1.1962 - val_acc: 0.6924 - val_f1_score: 0.6924

Epoch 00017: val_acc did not improve from 0.69662
Epoch 18/40

Epoch 00018: LearningRateScheduler reducing learning rate to 0.0007735091802198439 8.
```

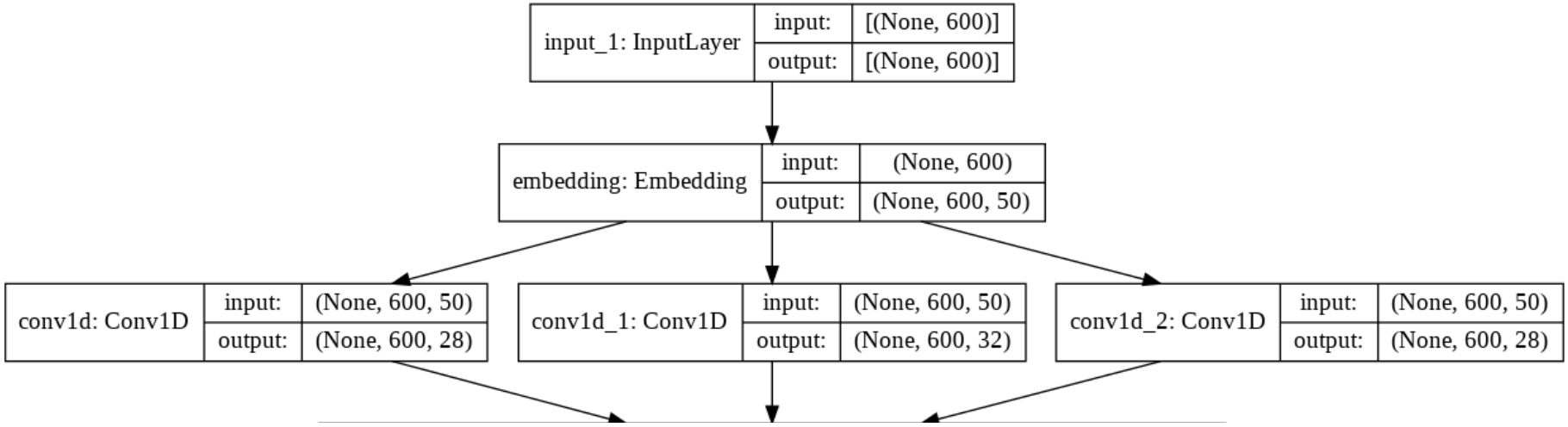
```
221/221 [==============================] - 7s 32ms/step - loss: 0.3308 - acc: 0.8807 - f1_score: 0.8807 - val_loss:
1.2661 - val_acc: 0.7043 - val_f1_score: 0.7043

Epoch 00018: val_acc improved from 0.69662 to 0.70427, saving model to best_model_1/weights-18-0.7043.hdf5
Epoch 19/40

Epoch 00019: LearningRateScheduler reducing learning rate to 0.0006983372120885178.
221/221 [==============================] - 8s 34ms/step - loss: 0.2883 - acc: 0.8989 - f1_score: 0.8989 - val_loss:
1.3144 - val_acc: 0.6985 - val_f1_score: 0.6985

Epoch 00019: val_acc did not improve from 0.70427
Epoch 20/40

Epoch 00020: LearningRateScheduler reducing learning rate to 0.0006983372149989009.
221/221 [==============================] - 7s 32ms/step - loss: 0.2609 - acc: 0.9050 - f1_score: 0.9050 - val_loss:
1.3664 - val_acc: 0.6956 - val_f1_score: 0.6956

Epoch 00020: val_acc did not improve from 0.70427
Epoch 00020: early stopping
```

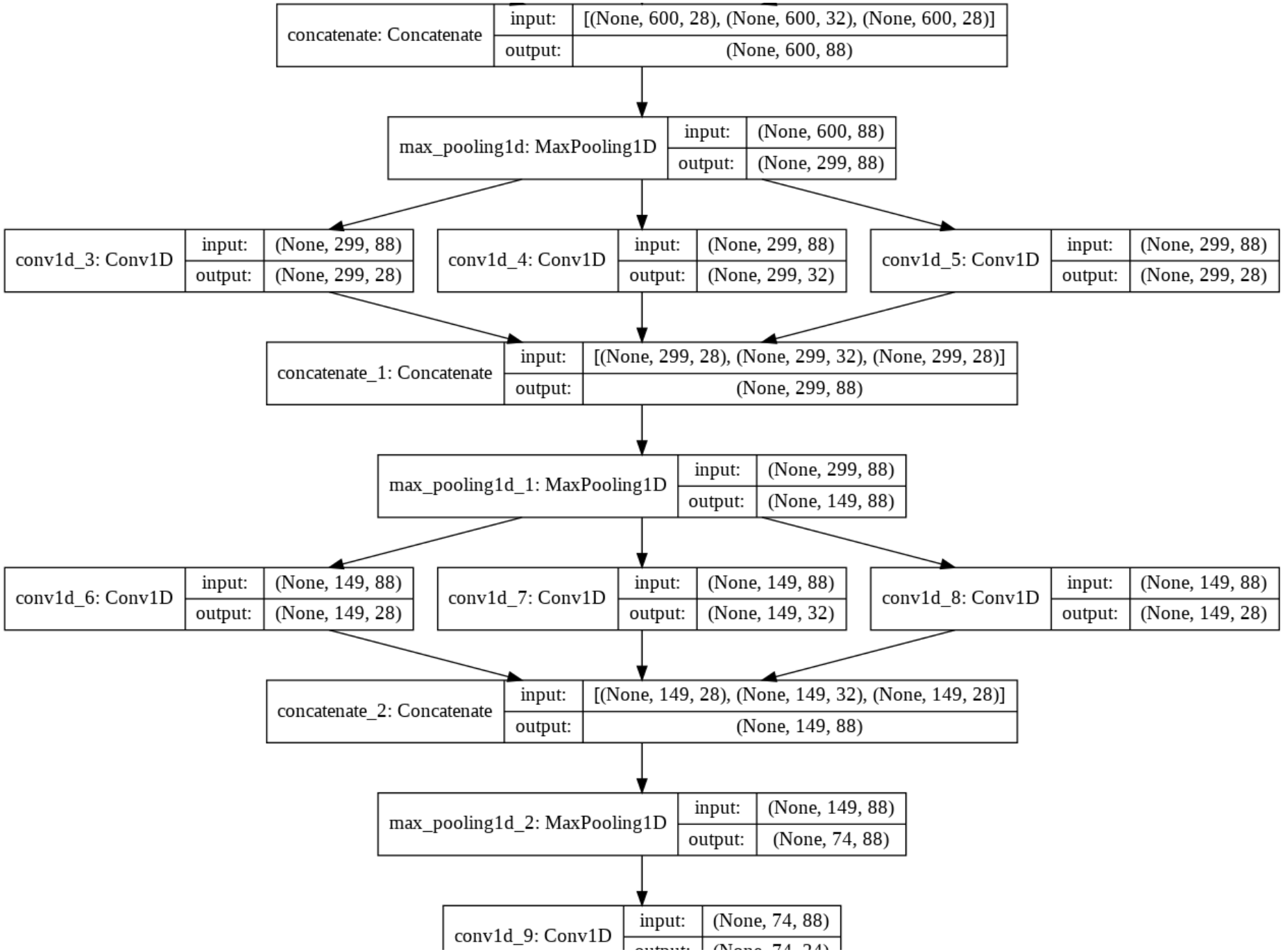
Out[ ]: <tensorflow.python.keras.callbacks.History at 0x7f8ffa6c5290>

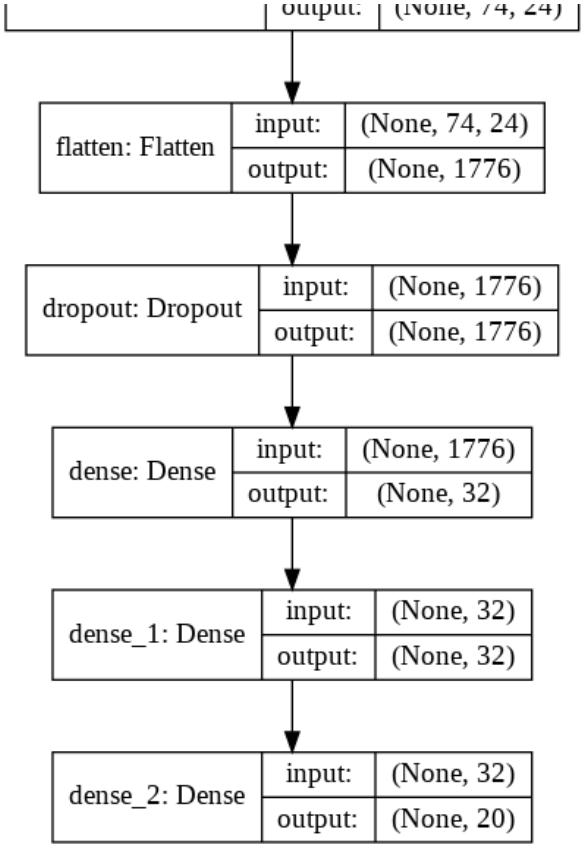
Model shapes & structure plots

In [ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_1_plot.png',show_shapes=True,show_layer_names=True)
#model.summary()
```

Out[ ]:

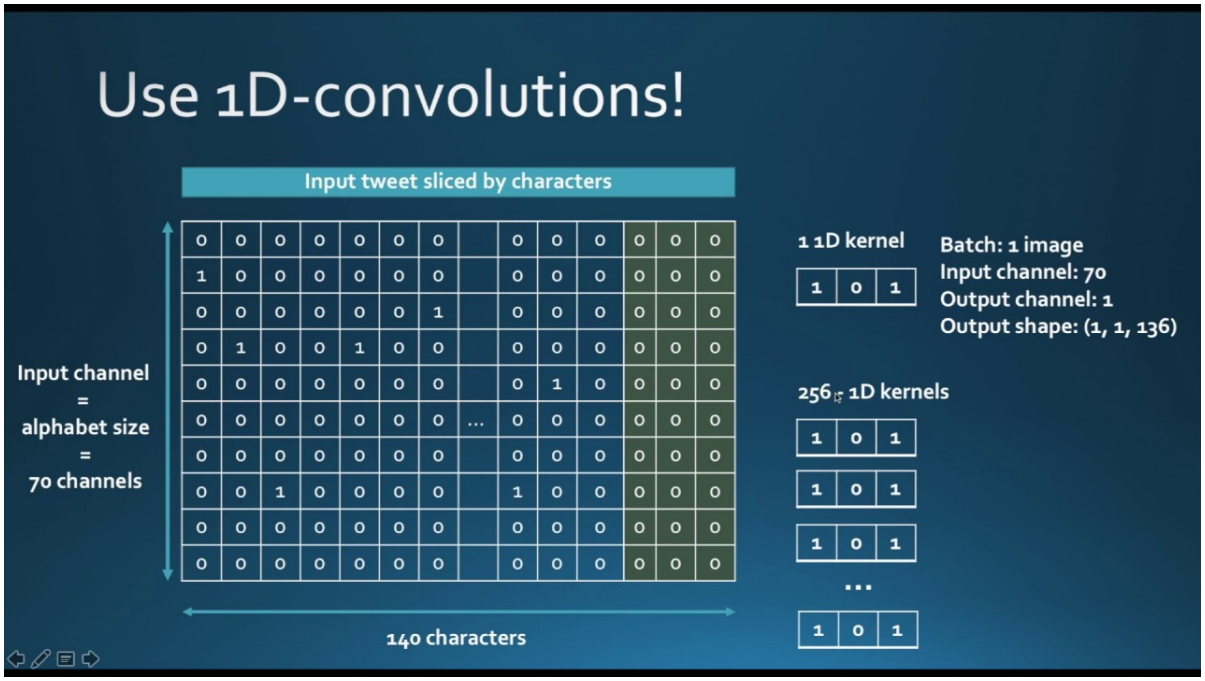






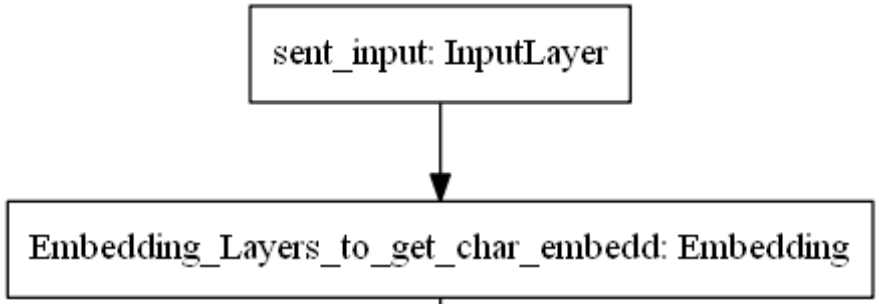
TASK 2

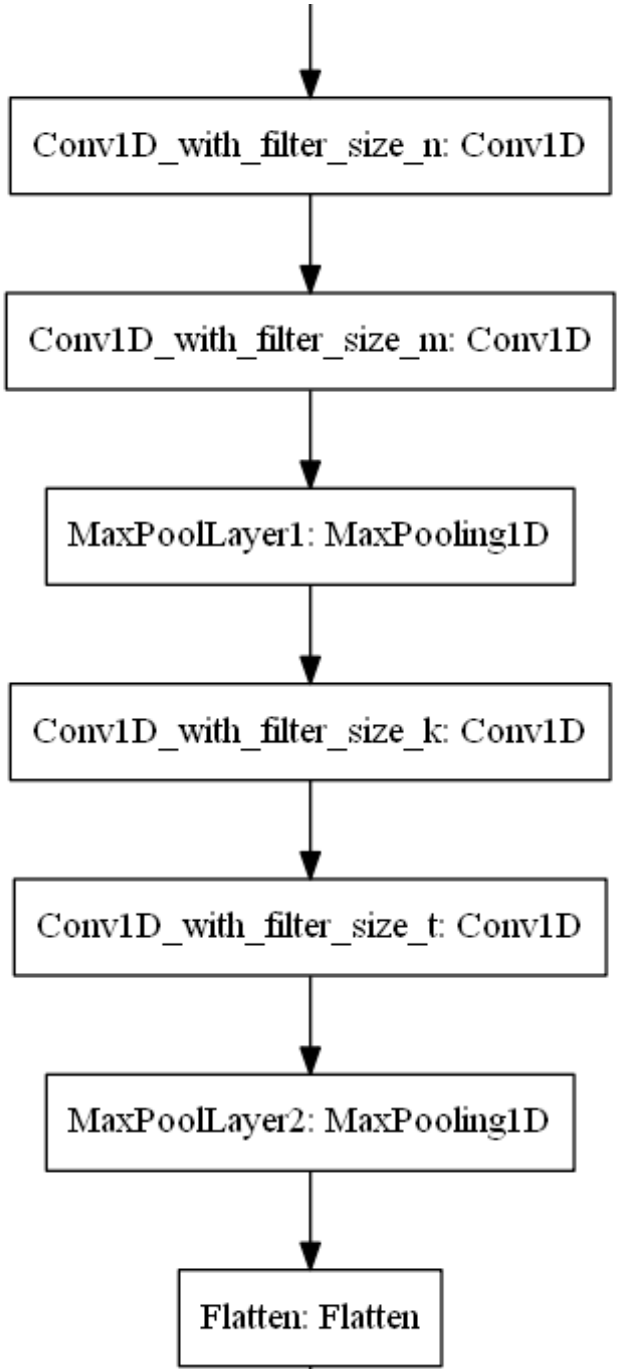
Model-2 : Using 1D convolutions with character embedding

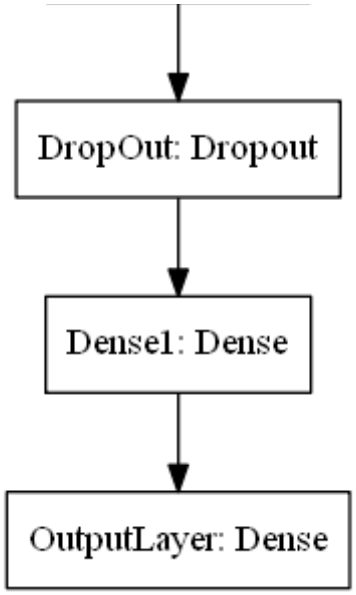


Here are the some papers based on Char-CNN

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







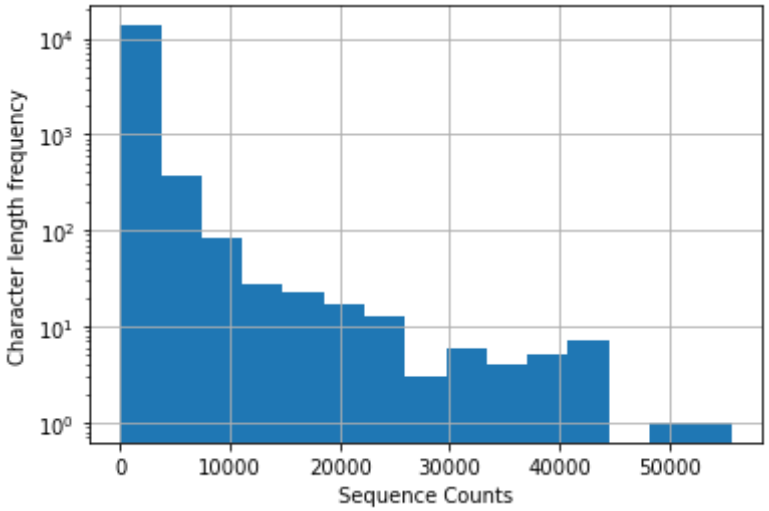
Tokenizing & Padding for Model - 2

In [6]:

```
# reference : https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html
# lets pad & tokenize the data:-
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenize our training data
oov_token = '<UNK>' ;
tokenizer = Tokenizer(oov_token=oov_token,filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n',char_level=True)
tokenizer.fit_on_texts(X_train)

# Get our training data character index
char_index = tokenizer.word_index
print("Length of character indices :-",len(char_index))
print('\n')
# Encode training data sentences into sequences
train_sequences = tokenizer.texts_to_sequences(X_train)

# Get max training sequence length
import statistics ; import matplotlib.pyplot as plt
lengths = [len(x) for x in train_sequences]
```

```
print("The median length of characters in the sequences :-\n",statistics.median(lengths))
plt.hist(lengths,bins = 15)
plt.xlabel('Sequence Counts')
plt.ylabel('Character length frequency')
plt.grid()
plt.yscale('log')
plt.show()

# count of lengths
count_gr2500 = 0
count_ls2500 = 0
for j in lengths:
    if j > 2500:
        count_gr2500 += 1
    else:
        count_ls2500 += 1

print("No of sentences having characters > 2500 :- \n",count_gr2500)
print("No of sentences having characters < 2500 :- \n",count_ls2500)
print("Contribution of lengthier sentences having > 2500 words :-\n",(count_gr2500)/len(lengths))

# Pad the training sequences
train_padded = pad_sequences(train_sequences,padding = 'post',maxlen= 2500,truncating='post')

# tokenize & Pad the testing sequences
test_sequences = tokenizer.texts_to_sequences(X_test)
test_padded = pad_sequences(test_sequences,padding='post',maxlen = 2500,truncating='post')

# Output the results of our work
print("\nPadded training shape:", train_padded.shape)
print("Training sequences data type:", type(train_sequences))
print("Padded Training sequences data type:", type(train_padded))

print("\nPadded testing shape:", test_padded.shape)
print("Testing sequences data type:", type(test_sequences))
print("Padded Testing sequences data type:", type(test_padded))
```

```
Length of character indices :- 45


The median length of characters in the sequences :-
 671
```




```
No of sentences having characters > 2500 :-
 1089
No of sentences having characters < 2500 :-
 13032
Contribution of lengthier sentences having > 2500 words :-
 0.07711918419375398

Padded training shape: (14121, 2500)
Training sequences data type: <class 'list'>
Padded Training sequences data type: <class 'numpy.ndarray'>

Padded testing shape: (4707, 2500)
Testing sequences data type: <class 'list'>
Padded Testing sequences data type: <class 'numpy.ndarray'>
```

Observation :

From the results of tokenizing the sequence, we can understand that only 7% of total sentences contribute for greater than 2500 characters & also its clearly evident from the histogram plotted. therefore i chose padding length to be 2500.

In [ ]:

```
# Load the glove zip/rar file into colab
from google.colab import files
files.upload() # glove file for characters
# NOTE : As the uploaded file is very big, I cleared the output because of easy readeability purposes
```

Model-2: Creating character embedding layer

In [ ]:

```
# reference :- https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

# create map character indices to embeddings/vectors
import numpy as np
embeddings_index = {}
with open('/content/glove_300d_char.txt') as f:
    for line in f:
        values = line.split()
        _char = values[0]
        coefs = np.asarray(values[1:],dtype='float32')
        embeddings_index[_char] = coefs

print('No of character vectors :-\n',len(embeddings_index))
emb_dim = coefs.shape[0]
print("Dimension of character vector :-\n",emb_dim)

# embedding matrix
embeddings_matrix = np.zeros((len(char_index) + 1,emb_dim))
for char,i in char_index.items():
    embedding_vector = embeddings_index.get(char)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embeddings_matrix[i] = embedding_vector

# embedding layer
from tensorflow.keras.layers import Embedding
embedding_layer = Embedding(len(char_index) + 1,
                            emb_dim,
                            weights = [embeddings_matrix],
                            input_length = 2500,
                            trainable = False)

print('shape of embedding_matrix :- \n',embeddings_matrix.shape)
```

```
No of character vectors :-
 94
Dimension of character vector :-
 300
shape of embedding_matrix :-
 (46, 300)
```

Tensorboard Results

In [ ]:
```
# %load_ext tensorboard
%reload_ext tensorboard
```

In [ ]:
```
# Clear any logs of previous executions/runs
import shutil
#/content/best_model_2 #./logfiles_model2
path='./logfiles_model2'
shutil.rmtree(path,ignore_errors=True)
```

In [ ]:
```
import shutil
#/content/best_model_1 #./logfiles_model1
path='/content/best_model_2'
shutil.rmtree(path,ignore_errors=True)
```

In [ ]:
```
!kill 418
```

In [ ]:
```
# Tensorboard outputs
import os;import datetime ; #Reference :--> https://www.dlology.com/blog/how-to-run-tensorboard-in-jupyter-notebook/
logs_base_dir = "./logfiles_model2"
os.makedirs(logs_base_dir,exist_ok=True)# Creating a directory to store logs
# Launch the tensorboard  & then refresh it after executing the model.
%tensorboard --logdir {logs_base_dir}
```

In [ ]:
```
# decay learning rate scheduler
import tensorflow
def scheduler(epoch,lr):
    if ((epoch) % 4 == 0):
        lr = ((lr)-(lr*(0.05))) # 0.05
    else:
        pass
```

```
    return lr
```

In [ ]:

```
# for add-on metrics we need this library
!pip install tensorflow-addons
```

```
Collecting tensorflow-addons
  Downloading https://files.pythonhosted.org/packages/66/4b/e893d194e626c24b3df2253066aa418f46a432fdb68250cde14bf9bb0
700/tensorflow_addons-0.13.0-cp37-cp37m-manylinux2010_x86_64.whl (679kB)
     |████████████████████████████████| 686kB 8.8MB/s
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/dist-packages (from tensorflow-addons) (2.
7.1)
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.13.0
```

APPLY MODEL - 2 & DISPLAY RESULTS

In [ ]:

```
# functional API
import tensorflow as tf
tf.keras.backend.clear_session()# For easy reset of notebook state
tf.random.set_seed(1234)
import tensorflow_addons as tfa ; import os
from tensorflow.keras import layers,Input
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dropout,Flatten,Dense,Conv1D,MaxPooling1D,BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.initializers import HeNormal

# Inputs
x = Input(shape=(2500,))
initializer = HeNormal(seed = 32)
embedded_sequences = embedding_layer(x)

x1 = Conv1D(32,3,activation='relu',padding = 'same',kernel_initializer = initializer)(embedded_sequences)
x2 = Conv1D(32,3,activation='relu',padding = 'same',kernel_initializer = initializer)(x1)
```

```
y = MaxPooling1D(3,strides = 2)(x2)

y1 = Conv1D(24,3,activation='relu',padding = 'same',kernel_initializer = initializer)(y)
y2 = Conv1D(24,3,activation='relu',padding = 'same',kernel_initializer = initializer)(y1)

z = MaxPooling1D(3,strides = 2)(y2)

z1 = Conv1D(24,3,activation='relu',padding = 'same',kernel_initializer = initializer)(z)
z2 = Conv1D(24,3,activation='relu',padding = 'same',kernel_initializer = initializer)(z1)

z3 = Flatten()(z2)
z4 = Dropout(0.3)(z3)
z5 = Dense(36, activation='relu',kernel_initializer = initializer)(z4)
z6 = Dense(20,activation='softmax',kernel_initializer = 'glorot_normal')(z5)

# model
model = Model(inputs = x,outputs = z6)

# optimizer
optimizer = Adam(learning_rate = 0.001)

#save model at every epoch end if validation accuracy is improved
filepath = "best_model_2/weights-{epoch:02d}-{val_acc:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath,monitor='val_acc',verbose=1,save_best_only=True,mode='auto')

#early stopping
earlystop = EarlyStopping(monitor='val_acc',patience=2,verbose=1)

#tensorboard callbacks # reference :-:--> https://www.dlology.com/blog/how-to-run-tensorboard-in-jupyter-notebook/
log_dir = os.path.join(logs_base_dir,datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tb_callback = TensorBoard(log_dir=log_dir,histogram_freq=1)

# Lr schedule
lrschedule = LearningRateScheduler(scheduler,verbose = 1)

# metric
f1score = tfa.metrics.F1Score(num_classes = 20,average = 'micro')

model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['acc',f1score])
# Model fit
```

```
model.fit(train_padded,ytrain_labels, validation_data=(test_padded,ytest_labels),
          epochs=50, batch_size=64,callbacks = [earlystop,checkpoint,lrschedule,tb_callback])
```

```
Epoch 1/50

Epoch 00001: LearningRateScheduler reducing learning rate to 0.0009500000451225787.
221/221 [==============================] - ETA: 0s - loss: 2.9545 - acc: 0.0795 - f1_score: 0.0795
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/metrics.py:257: UserWarning: Metric F1Score implements
a `reset_states()` method; rename it to `reset_state()` (without the final "s"). The name `reset_states()` has been d
eprecated to improve API consistency.
  'consistency.' % (self.__class__.__name__,))
221/221 [==============================] - 19s 84ms/step - loss: 2.9545 - acc: 0.0795 - f1_score: 0.0795 - val_loss:
2.9418 - val_acc: 0.0758 - val_f1_score: 0.0758

Epoch 00001: val_acc improved from -inf to 0.07584, saving model to best_model_2/weights-01-0.0758.hdf5
Epoch 2/50

Epoch 00002: LearningRateScheduler reducing learning rate to 0.0009500000160187483.
221/221 [==============================] - 19s 87ms/step - loss: 2.9276 - acc: 0.0868 - f1_score: 0.0867 - val_loss:
2.9312 - val_acc: 0.0850 - val_f1_score: 0.0850

Epoch 00002: val_acc improved from 0.07584 to 0.08498, saving model to best_model_2/weights-02-0.0850.hdf5
Epoch 3/50

Epoch 00003: LearningRateScheduler reducing learning rate to 0.0009500000160187483.
221/221 [==============================] - 17s 78ms/step - loss: 2.8978 - acc: 0.0969 - f1_score: 0.0969 - val_loss:
2.9213 - val_acc: 0.0865 - val_f1_score: 0.0865

Epoch 00003: val_acc improved from 0.08498 to 0.08647, saving model to best_model_2/weights-03-0.0865.hdf5
Epoch 4/50

Epoch 00004: LearningRateScheduler reducing learning rate to 0.0009500000160187483.
221/221 [==============================] - 19s 87ms/step - loss: 2.8288 - acc: 0.1200 - f1_score: 0.1200 - val_loss:
2.8884 - val_acc: 0.1050 - val_f1_score: 0.1050

Epoch 00004: val_acc improved from 0.08647 to 0.10495, saving model to best_model_2/weights-04-0.1050.hdf5
Epoch 5/50

Epoch 00005: LearningRateScheduler reducing learning rate to 0.0009025000152178108.
221/221 [==============================] - 17s 78ms/step - loss: 2.7206 - acc: 0.1535 - f1_score: 0.1535 - val_loss:
2.9262 - val_acc: 0.1058 - val_f1_score: 0.1058

Epoch 00005: val_acc improved from 0.10495 to 0.10580, saving model to best_model_2/weights-05-0.1058.hdf5
Epoch 6/50
```

```
Epoch 00006: LearningRateScheduler reducing learning rate to 0.0009025000035762787.
221/221 [==============================] - 19s 87ms/step - loss: 2.5906 - acc: 0.1959 - f1_score: 0.1959 - val_loss:
2.9637 - val_acc: 0.1060 - val_f1_score: 0.1060

Epoch 00006: val_acc improved from 0.10580 to 0.10601, saving model to best_model_2/weights-06-0.1060.hdf5
Epoch 7/50

Epoch 00007: LearningRateScheduler reducing learning rate to 0.0009025000035762787.
221/221 [==============================] - 17s 79ms/step - loss: 2.4381 - acc: 0.2309 - f1_score: 0.2309 - val_loss:
3.1042 - val_acc: 0.0992 - val_f1_score: 0.0992

Epoch 00007: val_acc did not improve from 0.10601
Epoch 8/50

Epoch 00008: LearningRateScheduler reducing learning rate to 0.0009025000035762787.
221/221 [==============================] - 19s 87ms/step - loss: 2.2726 - acc: 0.2879 - f1_score: 0.2879 - val_loss:
3.2127 - val_acc: 0.0958 - val_f1_score: 0.0958

Epoch 00008: val_acc did not improve from 0.10601
Epoch 00008: early stopping
```

Out[ ]: `<tensorflow.python.keras.callbacks.History at 0x7f120f75be50>`

Model shapes & structure plots

In [ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_2_plot.png',show_shapes=True,show_layer_names=True)
#model.summary()
```

Out[ ]: