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 [0]:

```
!wget --header="Host: doc-0o-5c-docs.googleusercontent.com" --header="User-Agent: Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.149
Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9,hi;q=0.8" --header="Referer: http
s://drive.google.com/" --header="Cookie:
AUTH_7eng5j7giaoveajo06v1k9i47ll2htp9_nonce=9j30e190pmrs0; _ga=GA1.2.463994940.1577099354" --heade
r="Connection: keep-alive" "https://doc-0o-5c-
docs.googleusercontent.com/docs/securesc/s15g6dj6p7fp8j9va0avmglmji71ccgo/08v25lgfpo6gre5qpnmb6vfj1
ugm/1585291950000/00484516897554883881/15791985801860507697/1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM?e=do
wnload&authuser=0&nonce=9j30e190pmrs0&user=15791985801860507697&hash=ndg1fpvqf4npc2kdhdeccbe2rmt4o
-O "documents.rar" -c
```

```
--2020-03-27 06:53:09--  https://doc-0o-5c-
docs.googleusercontent.com/docs/securesc/s15g6dj6p7fp8j9va0avmglmji71ccgo/08v25lgfpo6gre5qpnmb6vfj1
ugm/1585291950000/00484516897554883881/15791985801860507697/1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM?
e=download&authuser=0&nonce=9j30e190pmrs0&user=15791985801860507697&hash=ndg1fpvqf4npc2kdhdeccbe2rm
kt
Resolving doc-0o-5c-docs.googleusercontent.com (doc-0o-5c-docs.googleusercontent.com)...
74.125.197.132, 2607:f8b0:400e:c03::84
Connecting to doc-0o-5c-docs.googleusercontent.com (doc-0o-5c-
docs.googleusercontent.com)|74.125.197.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/rar]
Saving to: ‘documents.rar’

documents.rar           [  <=>                 ]  18.16M  69.9MB/s    in 0.3s

2020-03-27 06:53:10 (69.9 MB/s) - ‘documents.rar’ saved [19038123]
```

In [0]:

```
get_ipython().system_raw("unrar x documents.rar")
```

In [35]:

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

In [0]:

```
import nltk
```

```
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('punkt')
from nltk.chunk import conlltags2tree, tree2conlltags
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

In [0]:

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

In [0]:

```
files = os.listdir('documents')
files[:5]
files[0].split('_')[0]
labels = {}
for f in files:
  labels[f] = f.split('_')[0]
labels['comp.sys.ibm.pc.hardware_61072.txt']
```

Out[0]:

```
'comp.sys.ibm.pc.hardware'
```

In [0]:

```
# The function for decontracting the common contracted words
# Reference: https://stackoverflow.com/a/47091490/4084039

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

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

# Function for removing various punctuation marks and special characters

def cleanpunc(sentence): #function to clean the word of any punctuation or special characters
    cleaned = re.sub(r'[\n|\t|\r]', r'', sentence)
    cleaned = re.sub(r'[?|!|\'|"|#|-|>|<|=]',r'',cleaned)
    cleaned = re.sub(r'[,|)|(|\\|]',r'',cleaned)
    cleaned = re.sub("[\(\[].*?[\)\]]", "", cleaned)
    cleaned = re.sub(r'[/]', r' ', cleaned)
    cleaned = re.sub(r'--', r'', cleaned)
    return  cleaned
```

In [0]:

```
len(labels)
```

Out[0]:

```
18828
```

In [0]:

```
# Function for creating chunks of the sentence and performing required operations on the name enti
ties

def chunking_process(expr):
  ne_tree = nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(expr)))
  iob_tagged = tree2conlltags(ne_tree)
  ne_tree = conlltags2tree(iob_tagged)
  people = []
  exp = []
  for t in list(ne_tree):
    #If the data type is tree then it can be a named entity
    if type(t) == nltk.tree.Tree:
      #If the label that we find is 'GPE', it means we have a place
      if t.label() == 'GPE':
        #print(list(t))
        #We are joining all the words in a place name using '_'
        ne = t[0][0]
        for i in range(1, len(t)):
          ne += '_'+t[i][0]
        exp.append(ne)
      #In this case we have a person
      elif t.label() == 'PERSON':
        for i in list(t):
          #For every word that is part of a name, we are appanding it to a list
          people.append(i[0])
    else:
      exp.append(t[0])
  #If the word is not a person, we are adding it to the output sentence
  exp = list(filter(lambda x: x not in people, exp))
  exp = " ".join(exp)
  exp = cleanpunc(exp)
  return exp

#As you can see, this function removes names of people and joins all the words in the name of a pl
ace by '_'
line = chunking_process('His name is Marco Rodriguez and he lives in New York')
print(line)
```

```
His name is and he lives in New_York
```

In [0]:

```
files[:5]
```

Out[0]:

```
['talk.politics.guns_55272.txt',
 'rec.motorcycles_105096.txt',
 'talk.politics.mideast_76449.txt',
 'talk.politics.guns_54273.txt',
 'misc.forsale_75936.txt']
```

In [0]:

```
with open('documents/misc.forsale_76880.txt', encoding='utf-8', errors='ignore') as f:
  lines = f.readlines()
lines
```

Out[0]:

```
['From: belvilad@dunx1.ocs.drexel.edu (A. Belville)\n',
 'Subject: Re: waterbed for sale\n',
 '\n',
 'In article <1993Apr25.135853.5725@magnus.acs.ohio-state.edu> kwmiller@magnus.acs.ohio-state.edu
(Kenneth W Miller) writes:\n',
 '>Ken\n',
 '\n',
 '\tAgain, tell us about it Ken!\n',
 '\n',
 '-=- Andy -=-\n',
 '\n',
 '\n',
 '______________________________________________________________________________\n',
 "Andy Belville                              || It's taken me a long time, but I've\n",
 'belvilad@dunx1.ocs.drexel.edu             || fallen in Love with a beautiful woman.\n',
 '______________________________________________________________________________\n']
```

In [0]:

```python
#This funtion is for removing all the words that are sized less than 3 and more than 14

def len_check(sent):
  new = ""
  for s in sent.split():
    if len(s) > 2 and len(s) < 15:
      new += s + " "
  return new.rstrip()

len_check('Happy Birthday T jbekhbkbdkfhwbefkshvdqj').lower()
```

Out[0]:

```
'happy birthday'
```

In [0]:

```python
#This function removes the underscore from words where the substring on the left and/or right of t
he underscore is of length less than 3

sent = "dwjf fjjbkh _dnjh jwnd_ jdnch _jdwbn_ wdhb d_berlin dr_berlin jqsd_asbjh"
def _handler(sent):
  new = ""
  for word in sent.split():
    w = ""
    if word[0] == '_' and word[-1] == '_':
      w = word[1:-1]
    elif word[0] == '_':
      w = word[1:]
    elif word[-1] == "_":
      w = word[:-1]
    else:
      if '_' in word:
        l = word.split('_')
        t = []
        for i in l:
          if len(i) > 2:
            t.append(i)
        t = '_'.join(t)
        w = t
      else:
        w = word
    new += str(w) + " "
  return new.rstrip()
q = _handler(sent)
q
```

Out[0]:

```
'dwjf fjjbkh dnjh jwnd jdnch jdwbn wdhb berlin berlin jqsd_asbjh'
```

In [0]:

```python
### count plot of all the class labels.
```

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 thos
e texts by '.'
after that remove the words whose length is less than or equal to 2 and also remove'com' wo
rd 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 li
st 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 char
s.
**Eg: if we have sentance like "Subject: Re: Gospel Dating @ \r\r\n" --> You have to get "Gos
pel 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 [0]:

```
#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'), ('th
e', 'DT'), Tree('GPE', [('New', 'NNP'), ('York', 'NNP')])]

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

My name is Srikanth Varma --> [('My', 'PRP$'), ('name', 'NN'), ('is', 'VBZ'), Tree('PERSON', [('Sr
ikanth', '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.

In [0]:

```
data.columns
```

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

In [0]:

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

```
text                    From: arc1@ukc.ac.uk (Tony Curtis)\r\r\r\nSubj...
class                                                         alt.atheism
preprocessed_text       said re is article if followed the quoting rig...
preprocessed_subject                                  christian morality is
preprocessed_emails                                     ukc mac macalstr edu
Name: 567, dtype: object
```

In [0]:

```
#This function completes various preprocessing steps as explianed below

def cleaning(preprocessed):
  preprocessed = decontracted(preprocessed)  #Performs decontractions using the function we defined above
  preprocessed = chunking_process(preprocessed)  #Performs chunking using the function we defined above
  preprocessed = cleanpunc(preprocessed) #Cleans punctuation marks and special characters
  preprocessed = re.sub(r'<[^>]+>', r'', preprocessed) #Removes all the expressions of the kind '<anything>'
  preprocessed = preprocessed.lstrip() #Removes spaces in the start of the sentence
  preprocessed = re.sub("[\(\[].*?[\)\]|&|$|#|@]", "", preprocessed) #Removing some extra special characters
  preprocessed = re.sub(r'[.|-]', r' ', preprocessed) #Removing the '.' and '-'
  preprocessed = re.sub(r'[0-9]', r' ', preprocessed) #Removing numbers
  preprocessed = _handler(preprocessed) #Applies the necessary changes of the words containg '_'
  preprocessed = len_check(preprocessed).lower() #Filters the words according to their length and converts all characters to lower case
  preprocessed = re.sub('[^A-Za-z_]+', ' ', preprocessed) #Keeps only the characters that are in (a, z) or in (A, Z) or '_'
  return preprocessed
```

In [0]:

```
#This function is practically the same as the funtion above but without the chunking process, mean
t for the emails and the subjects

def cleaning_t(preprocessed):
  preprocessed = decontracted(preprocessed)
  preprocessed = cleanpunc(preprocessed)
  preprocessed = re.sub(r'<[^>]+>', r'', preprocessed)
  preprocessed = preprocessed.lstrip()
  preprocessed = re.sub("[\(\[].*?[\)\]|&|$|#|@]", "", preprocessed)
  preprocessed = re.sub(r'[.|-]', r' ', preprocessed)
  preprocessed = re.sub(r'[0-9]', r' ', preprocessed)
  preprocessed = _handler(preprocessed)
  preprocessed = len_check(preprocessed).lower()
  preprocessed = re.sub('[^A-Za-z_]+', ' ', preprocessed)
  return preprocessed
```

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

In [0]:

```
def preprocessing(filename):
  with open(filename, encoding='utf-8', errors='ignore') as f:
    lines = f.readlines()
  #We define three empty strings for the email, subject and text data
  email = ""
  subject = ""
  preprocessed = ""
  #We iterate through the list of lines in the file
  for line in lines:
    s = 0 #This is a flag to mark that the line has been included (s = 1, if the line gets
included)
    if "Subject:" in line:
      #We have entered the condition for the subject line
      s = 1 #This line will now be included so we mark s as 1
      line = re.sub(r'[@|%|#|&|*]', r'', line)
      lis = line.split(" ")
      #We will check for all words in this line for the presence of ':'
      for l in lis:
        if ':' not in l:
          #If ':' is not present in the word, we add it to the subject
          subject += " " + l
      #We add all the words to the final text
      preprocessed += " "
    if 'From:' in line or 'Write to:' in line:
      #Here we don't do anything for the lines that contain these two words (virtually removing th
em from the text)
      s = 1;
    for l in line.split(' '):
      if '@' in l:
        #Here we are creating the email data
        preprocessed += " "
        ind = l.index('@')
        temp = l[ind+1:].split(' ')[0].split('.')
        #We are adding all the words in the email that have length > 2 and are not 'com'
        for t in temp:
          if len(t) >= 3 and t is not 'com':
            email += " " +t
      #All the words that have not been take care of above (s=0) and don't contain ':' are added t
o the final text
      elif s == 0 and ':' not in l:
        preprocessed += " " + l
  #All three subparts of the text are now cleaned
  subject = cleaning_t(subject)
  email = cleaning_t(email)
  preprocessed = cleaning(preprocessed)
  f.close()
  return (email, subject, preprocessed)

p = preprocessing('documents/'+'alt.atheism_49960.txt')
print(p[2])
print(p[1])
```

```
atheism resources resources december usa freedom from foundation fish bumper stickers and assorted
other atheist paraphernalia are available from the the designs evolution designs sell the fish fis
h symbol like the ones christians stick their cars but with feet and the word written inside the d
eluxe moulded plastic fish postpaid the people the area can get from try mailing for net people wh
o directly the price per fish press aap publish various atheist books critiques the lists biblical
contradictions and one such book the and edition bible contradictions absurdities atrocities immor
alities contains the based the king version the austin books including holy horrors see below alte
rnate address which may newer older prometheus african americans humanism organization promoting b
lack secular humanism and uncovering the history black freethought they publish quarterly
newsletter examiner association national secular society street holloway british society lamb red
lion square fax the publish the freethinker monthly magazine founded der und berlin publish und zu
r zeit politisches journal der und miz vertrieb berlin for atheist books write ucherdienst der han
nover disch the short story the ultimate proof that exists all characters and events are
fictitious any similarity living dead gods well miller canticle for one gem this post atomic dooms
day novel the monks who spent their lives copying blueprints from saint leibowitz filling the
sheets paper with ink and leaving white lines and letters pangborn davy post atomic doomsday novel
set clerical states the church for example forbids that anyone produce describe use any substance
containing atoms dick wrote many philosophical and thought provoking short stories and novels his
stories are bizarre times but very approachable wrote mainly but wrote about people truth and reli
gion rather than technology although often believed that had met some sort remained sceptical amon
gst his novels the following are some galactic pot healer fallible alien deity summons group craft
smen and women remote planet raise giant cathedral from beneath the oceans when the deity begins d
emand faith from the earthers pot healer unable comply polished ironic and amusing novel maze deat
h noteworthy for its description technology based religion valis the schizophrenic hero searches f
or the hidden mysteries christianity after reality fired into his brain pink laser beam unknown bu
t possibly divine origin accompanied his dogmatic and dismissively atheist friend and assorted oth
er odd characters the invades making young woman pregnant she returns from another star system unf
ortunately she terminally ill and must assisted dead man whose brain wired hour easy listening mus
ic atwood the story based the premise that the mysteriously assassinated and quickly take charge t
he nation set right again the book the diary woman life she tries live under the new theocracy wom
en right own property revoked and their bank accounts are closed sinful luxuries are outlawed and
the radio only used for readings from the crimes are punished doctors who performed legal abortion
s the old world are hunted down and hanged writing style difficult get used first but the tale gro
ws more and more chilling goes authors the this somewhat dull and rambling work has often been cri
ticized however probably worth reading only that you will know what all the fuss about exists many
different versions make sure you get the one true version non fiction rosa vicars christ although
seems even this very enlighting history papal immoralities adulteries fallacies etc german gottes
erste dunkle des droemer knaur martin philosophical justification detailed and scholarly
justification atheism contains outstanding appendix defining terminology and usage this
necessarily tendentious area argues both for negative atheism the non belief the existence god and
also for positive atheism the belief the non existence god includes great refutations the most cha
llenging arguments for god particular attention paid refuting contempory theists such and
swinburne pages hardcover paperback also available the christianity comprehensive critique
christianity which considers the best contemporary defences and ultimately demonstrates that they
are unsupportable and incoherent pages turner the baltimore subtitled the unbelief america
examines the way which unbelief whether agnostic atheistic became mainstream alternative world vie
w focusses the period and while considering france and britain the emphasis american and particula
rly new_england developments neither religious history secularization atheism rather the
intellectual history the fate single idea the belief that exists pages isbn hardcover paper seldes
editor the great thoughts new_york dictionary quotations different kind concentrating statements a
nd writings which explicitly implicitly present the person philosophy and world view includes obsc
ure and often suppressed opinions from many people for some popular observations traces the way wh
ich various people expressed and twisted the idea over the centuries quite number the quotations a
re derived from cardiff what religion and views religion pages isbn paper the revised edition oxfo
rd this book the second volume trilogy that began with the theism and was concluded with and this
work swinburne attempts construct series inductive arguments for the existence his arguments which
are somewhat tendentious and rely upon the imputation late century western values and aesthetics w
hich supposedly simple can conceived were decisively rejected the theism the revised edition the s
winburne includes appendix which makes somewhat incoherent attempt rebut mackie the theism oxford
this posthumous volume contains comprehensive review the principal arguments for and against the e
xistence ranges from the classical philosophical positions descartes anselm through the moral argu
ments newman and the recent restatements the classical theses and swinburne also addresses those p
ositions which push the concept beyond the realm the rational such those kierkegaard and well repl
acements for such axiarchism the book delight read less formalistic and better written than works
and refreshingly direct when compared with the hand waving swinburne haught holy illustrated histo
ry and religious persecution from ancient times the present day and not only library number allen
anthology see the listing for for humanism above stein anthology atheism and rationalism anthology
covering wide range subjects including the devil and and the history freethought comprehensive bib
liography cohen the the bible believer study why people become and what effect has them there smal
l mail based archive server mantis which carries archives old alt atheism moderated articles and a
ssorted other files for more information send mail saying help send atheism index and will mail ba
ck reply mathew
alt atheism atheist resources
```

In [0]:

```
def preprocess(Input_Text):
    """Do all the Preprocessing as shown above and
    return a tuple contain preprocess_email,preprocess_subject,preprocess_text for that
Text_data"""
    return (list_of_preproessed_emails,subject,text)
```

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.

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

In [0]:

```
from tqdm import tqdm_notebook as tqdm

p_emails = [] #list for preprocessed emails
p_subs = []   #list for preprocessed subjects
p_text = []   #list for preprocessed text
text = []     #list for text without preprocessing
label = []    #list for labels

for file in tqdm(files):
  with open('documents/' + file, encoding='utf-8', errors='ignore') as f:
    t = f.readlines()
  ' '.join(t)
  text.append(t)
  label.append(labels[file])
  p = preprocessing('documents/'+file)
  #Appending the preprocessed elements to their respective lists
  p_emails.append(p[0])
  p_subs.append(p[1])
  p_text.append(p[2])
  f.close()

print("No of files skipped for utf-8 error: ", c)
df = pd.DataFrame(list(zip(text, label, p_text, p_subs, p_emails)), columns=['Text', 'Class', 'prep
rocessed_text', 'preprocessed_subject', 'preprocessed_emails'])
print(df.shape)
df.head()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: TqdmDeprecationWarning: This
function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  if __name__ == '__main__':
```

```
No of files skipped for utf-8 error:  0
(18828, 5)
```

Out[0]:

	Text	Class	preprocessed_text	preprocessed_subject	preprocessed_emails
0	[From: PA146008@UTKVM1.UTK.EDU (David Veal)\n,...	talk.politics.guns	article article everyone discussing why the di...	batf achieved objective wants move	utkvm utk edu cronkite central sun com clesun ...
1	[From: s851708@minyos.xx.rmit.OZ.AU (John Edmo...	rec.motorcycles	dogs will chase anything that moves have two d...	dogs bikes	minyos rmit minyos rmit
2	[From: tichauer@valpso.hanse.de (Manfredo Tich...	talk.politics.mideast	writes apr the following two quite normal why ...	israeli terrorism	valpso hanse virginia edu virginia edu virgini...
3	[From: cdt@sw.stratus.com (C. D. Tavares)\n, S...	talk.politics.guns	article would like have handgun would have get...	gun like american express card	stratus com surt ifi uio ifi uio rocket stratu...

	Text	Class	preprocessed_text	preprocessed_subject	preprocessed_emails
4	[From: mark@ardsley.business.uwo.ca (Mark Bram...	misc.forsale	hope you realize that for cellular phone you n...	cellular phone portable for sale	ardsley business uwo ardsley business uwo

In [0]:

```
#Merging all of the elements of the text into one column
df['merged_text'] = df[['preprocessed_text', 'preprocessed_subject', 'preprocessed_emails']].apply
(lambda x: " ".join(x), axis = 1)
df.head(10)
```

Out[0]:

	Text	Class	preprocessed_text	preprocessed_subject	preprocessed_emails	merged_text
0	[From: PA146008@UTKVM1.UTK.EDU (David Veal)\n,...	talk.politics.guns	article article everyone discussing why the di...	batf achieved objective wants move	utkvm utk edu cronkite central sun com clesun ...	article article everyone discussing why the di...
1	[From: s851708@minyos.xx.rmit.OZ.AU (John Edmo...	rec.motorcycles	dogs will chase anything that moves have two d...	dogs bikes	minyos rmit minyos rmit	dogs will chase anything that moves have two d...
2	[From: tichauer@valpso.hanse.de (Manfredo Tich...	talk.politics.mideast	writes apr the following two quite normal why ...	israeli terrorism	valpso hanse virginia edu virginia edu virgini...	writes apr the following two quite normal why ...
3	[From: cdt@sw.stratus.com (C. D. Tavares)\n, S...	talk.politics.guns	article would like have handgun would have get...	gun like american express card	stratus com surt ifi uio ifi uio rocket stratu...	article would like have handgun would have get...
4	[From: mark@ardsley.business.uwo.ca (Mark Bram...	misc.forsale	hope you realize that for cellular phone you n...	cellular phone portable for sale	ardsley business uwo ardsley business uwo	hope you realize that for cellular phone you n...
5	[From: pharvey@quack.kfu.com (Paul Harvey)\n, ...	talk.religion.misc	article article this brings another question s...	kind and loving god not	quack kfu com sandvik kent apple com newton ap...	article article this brings another question s...
6	[From: luriem@alleg.edu The Liberalizer (Micha...	rec.sport.baseball	article walter works the yankees are now one g...	yankkes game closer	alleg edu axe acadiau axe acadiau	article walter works the yankees are now one g...
7	[From: probert@ucsb.edu (Dave Probert)\n, Subj...	comp.windows.x	posting this for friend please respond working...	image data format question	ucsb edu hub ucsb edu	posting this for friend please respond working...
8	[From: mccullou@snake2.cs.wisc.edu (Mark McCul...	alt.atheism	turn jump article reference line trimmed there...	political atheists	snake wisc edu gap caltech edu cco caltech edu...	turn jump article reference line trimmed there...
9	[From: garrett@Ingres.COM\n, Subject: Re: Retu...	talk.politics.misc	article writes article article drieux just dri...	return the know nothing party	ingres com wetware com wetware com organpipe u...	article writes article article drieux just dri...

In [0]:

```
df = pd.read_csv('required.csv')
df = df.drop(columns = ['Unnamed: 0'])
df.head()
```

Out[0]:

	Class	merged_text
0	talk.politics.guns	article article everyone discussing why the di...
1	rec.motorcycles	dogs will chase anything that moves have two d...
2	talk.politics.mideast	writes apr the following two quite normal why ...
3	talk.politics.guns	article would like have handgun would have get...
4	misc.forsale	hope you realize that for cellular phone you n...

In [0]:

```
df.shape
```

Out[0]:

```
(4777, 2)
```

In [0]:

```
#Checking the maximum length of text in a file and the number of words in the vocabulary

from tqdm import tqdm_notebook as tqdm
max = 0
word_list = []
lenghts = []
for l in tqdm(df['merged_text'].values):
  lenghts.append(len(l.split()))
  if len(l.split()) > max:
    max = len(l.split())
  for w in l.split():
    if w not in word_list:
      word_list.append(w)
print(max, len(word_list))
print(median(lenghts))
```

In [0]:

```
import statistics
#printing the mean length and the median length
print(statistics.mean(lenghts), statistics.median(lenghts))
```

```
183.1612284069098 110.0
```

In [0]:

```
#Saving the data frame into a csv file for later use
df.to_csv('data.csv')
```

In [0]:

```
short = df.drop(columns=['Text', 'preprocessed_text', 'preprocessed_subject', 'preprocessed_emails'
])
short.head()
short.to_csv('required.csv')
```

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 t
he 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.
```

In [0]:

```
#Creating a custom callback for measuring the micro averaged F1 score

from sklearn.metrics import f1_score

class Metrics(tf.keras.callbacks.Callback):
  def __init__(self, validation_data):
    self.x_val = validation_data[0]
    self.y_val = validation_data[1]

  def on_train_begin(self, logs={}):
    self.metrics = {'f1_score': []}
    print(self.x_val.shape, self.y_val.shape)

  def on_epoch_end(self, epoch, logs={}):
    y_pred = (np.asarray(self.model.predict(self.x_val))).round()
    y_true = self.y_val
    #Using the f1_score function provided by sickit learn, to get the score
    F1s = f1_score(y_true, y_pred, average = 'micro')
    self.metrics['f1_score'].append(F1s)
    print("The F1 score is: ", F1s)
```

In [0]:

```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard
```

In [0]:

```
import pandas as pd
import numpy as np
```

In [0]:

```
df = pd.read_csv('required.csv')
df.head()
```

Out[0]:

	Unnamed: 0	Class	merged_text
0	0	talk.politics.guns	article article everyone discussing why the di...
1	1	rec.motorcycles	dogs will chase anything that moves have two

	Unnamed: 0	Class	merged_text
2	2	talk.politics.mideast	writes apr the following two quite normal why ...
3	3	talk.politics.guns	article would like have handgun would have get...
4	4	misc.forsale	hope you realize that for cellular phone you n...

In [0]:

```
X = df['merged_text'].values
Y = df['Class']
```

In [0]:

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass

# Load the TensorBoard notebook extension.
%load_ext tensorboard
```

In [0]:

```
#Creating one-hot-encoded vectors of the labels

from sklearn.preprocessing import OneHotEncoder
Y = Y.values.reshape(-1, 1)
onehotencoder = OneHotEncoder()
Y = onehotencoder.fit_transform(Y).toarray()
```

In [0]:

```
Y[0]
```

Out[0]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 0., 0.])
```

In [0]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test  = train_test_split(X, Y, stratify = Y, test_size = 0.25)
```

In [0]:

```
print(X_train.shape, Y_train.shape)
```

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

In [0]:

```
#Downloading the pretrained GloVe vectors to be used in the embedding layer

!wget --header="Host: downloads.cs.stanford.edu" --header="User-Agent: Mozilla/5.0 (Windows NT 10.
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36" --header
="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
d-exchange;v=b3;q=0.9" --header="Accept-Language: en-US,en;q=0.9,hi;q=0.8" --header="Cookie: _ga=G
A1.2.442721336.1583413614; _gid=GA1.2.481596713.1584432856" --header="Connection: keep-alive" "htt
p://downloads.cs.stanford.edu/nlp/data/glove.42B.300d.zip" -O "glove.42B.300d.zip" -c
```

```
--2020-03-28 13:11:30--  http://downloads.cs.stanford.edu/nlp/data/glove.42B.300d.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu (downloads.cs.stanford.edu)|171.64.64.22|:80... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: 1877800501 (1.7G) [application/zip]
Saving to: ‘glove.42B.300d.zip’

glove.42B.300d.zip    0%[                    ]       0  --.-KB/s               ^C
```

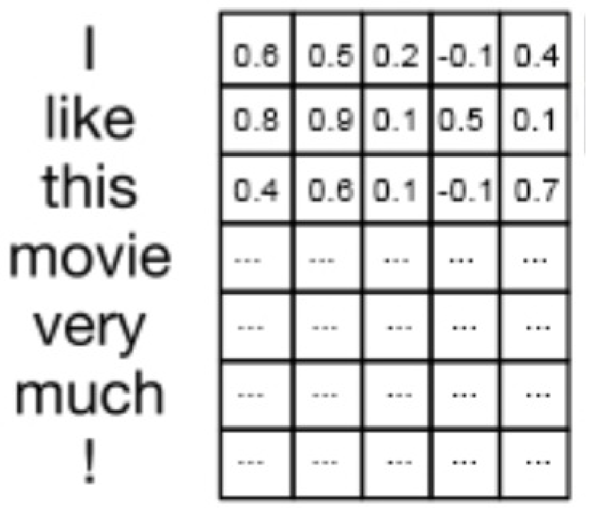
In [0]:

```
!unzip glove.42B.300d.zip -d glove_vectors
```

```
Archive:  glove.42B.300d.zip
  inflating: glove_vectors/glove.42B.300d.txt
```

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 o
f 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/

In [0]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Conv1D, MaxPool1D
from tensorflow.keras.initializers import he_normal, glorot_normal
from tqdm import tqdm_notebook as tqdm
from datetime import datetime
```

In [0]:

```
# reference : https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html
#Using the keras tokenizer to convert the words into integers

t = Tokenizer(filters='')
#Fitting on the training data
t.fit_on_texts(X_train)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_train = t.texts_to_sequences(X_train)
encoded_test = t.texts_to_sequences(X_test)
#Taking the maximimum length as that of the longest document and applying post-padding
padded_train = pad_sequences(encoded_train, padding='post')
input_length = padded_train.shape[1]
print("input length",input_length)

padded_test = pad_sequences(encoded_test, maxlen = input_length, padding='post')
output_length = padded_test.shape[1]
print("output length",output_length)

glove_size = 300

#Creating a dictionary of word vectors using the glove vectors file that we downloaded above
embeddings_index = dict()
f = open('glove_vectors/glove.42B.300d.txt')
for line in tqdm(f):
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()
```

```
input length 8962
output length 8962
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: TqdmDeprecationWarning: This
function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

In [0]:

```
#Demonstrating that the tokenizer does not remove the '_'
docs  = ['i live in New_york']
t = Tokenizer(num_words=500, filters='')
t.fit_on_texts(docs)
encoded_docs = t.texts_to_sequences(docs)
encoded_docs
```

Out[0]:

```
[[1, 2, 3, 4]]
```

In [0]:

```
X_train = padded_train
X_test = padded_test
print(X_test.shape)
```

```
(4707, 8962)
```

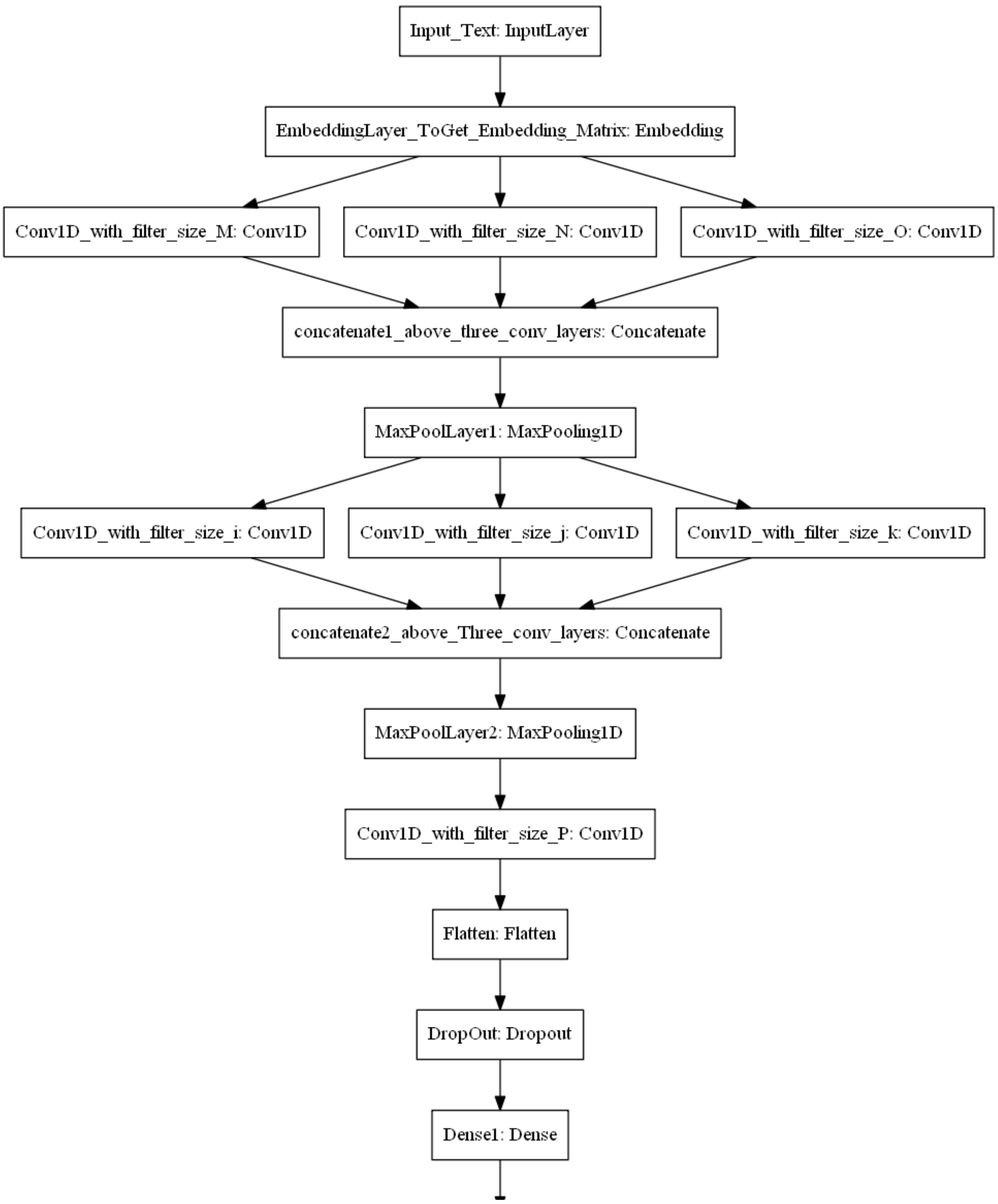
In [0]:

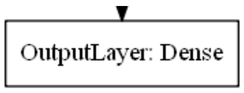
```
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 300))
for word, i in tqdm(t.word_index.items()):
 embedding_vector = embeddings_index.get(word)
 if embedding_vector is not None:
  embedding_matrix[i] = embedding_vector
```

```
Loaded 1917494 word vectors.
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: TqdmDeprecationWarning: This
function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  after removing the cwd from sys.path.
```


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
Conv1D_with_filter_size_P: Conv1D
Flatten: Flatten
DropOut: Dropout
Dense1: Dense



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

In [0]:

```
from tensorflow.keras.layers import Input, Concatenate, Dropout
from tensorflow.keras import Model
from tensorflow.keras.optimizers import Adam, SGD, Adadelta
```

In [0]:

```
#Initializing various optimizers

delta = Adadelta(learning_rate=1.0, rho=0.95)
adam = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
sgd = SGD(learning_rate=0.01, momentum=0.0, nesterov=False)
```

In [0]:

```
embedding_matrix[10]
```

In [0]:

```
#BUILDING MODEL 1
import random as rn
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)

input_ = Input(shape=(input_length))
e = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length = input_length, trainable =
False)(input_)

c13 = Conv1D(filters=16, kernel_size=32, kernel_initializer=he_normal(seed=0), activation='relu')(e
)
c15 = Conv1D(filters = 16, kernel_size=32, kernel_initializer=he_normal(seed=0), activation='relu')
(e)
c17 = Conv1D(filters = 16, kernel_size=32, kernel_initializer=he_normal(seed=0), activation='relu')
(e)
conc1 = Concatenate()([c13, c15, c17])

m1 = MaxPool1D(pool_size=2, strides=2)(conc1)
c23 = Conv1D(filters=16, kernel_size=16, kernel_initializer=he_normal(seed=0), activation='relu')(m
1)
c25 = Conv1D(filters = 16, kernel_size=16, kernel_initializer=he_normal(seed=0), activation='relu')
(m1)
c27 = Conv1D(filters = 16, kernel_size=16, kernel_initializer=he_normal(seed=0), activation='relu')
(m1)
conc2 = Concatenate()([c23, c25, c27])

m2 = MaxPool1D(pool_size = 4, strides=2)(conc2)
c3 = Conv1D(filters=16, kernel_size=32, kernel_initializer=he_normal(seed=0), activation='relu')(m2
)

f = Flatten()(c3)
d = Dropout(rate = 0.5)(f)
d = Dense(units = 100, kernel_initializer=glorot_normal(seed=32), activation='relu')(d)
output = Dense(units = 20, name = 'output', kernel_initializer=glorot_normal(seed=32), activation='
softmax')(d)

model = Model(inputs = input_, outputs = output)
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 8962)]       0
```

```
________________________________________________________________________________________________
embedding (Embedding)           (None, 8962, 300)    21186300    input_1[0][0]
________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 8931, 16)     153616      embedding[0][0]
________________________________________________________________________________________________
conv1d_1 (Conv1D)               (None, 8931, 16)     153616      embedding[0][0]
________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 8931, 16)     153616      embedding[0][0]
________________________________________________________________________________________________
concatenate (Concatenate)       (None, 8931, 48)     0           conv1d[0][0]
                                                                 conv1d_1[0][0]
                                                                 conv1d_2[0][0]
________________________________________________________________________________________________
max_pooling1d (MaxPooling1D)    (None, 4465, 48)     0           concatenate[0][0]
________________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 4450, 16)     12304       max_pooling1d[0][0]
________________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 4450, 16)     12304       max_pooling1d[0][0]
________________________________________________________________________________________________
conv1d_5 (Conv1D)               (None, 4450, 16)     12304       max_pooling1d[0][0]
________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 4450, 48)     0           conv1d_3[0][0]
                                                                 conv1d_4[0][0]
                                                                 conv1d_5[0][0]
________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 2224, 48)     0           concatenate_1[0][0]
________________________________________________________________________________________________
conv1d_6 (Conv1D)               (None, 2193, 16)     24592       max_pooling1d_1[0][0]
________________________________________________________________________________________________
flatten (Flatten)               (None, 35088)        0           conv1d_6[0][0]
________________________________________________________________________________________________
dropout (Dropout)               (None, 35088)        0           flatten[0][0]
________________________________________________________________________________________________
dense (Dense)                   (None, 100)          3508900     dropout[0][0]
________________________________________________________________________________________________
output (Dense)                  (None, 20)           2020        dense[0][0]
================================================================================================
Total params: 25,219,572
Trainable params: 4,033,272
Non-trainable params: 21,186,300
________________________________________________________________________________________________
```

In [0]:

```
#Setting the filepath for the ModelCheckpoint callback to save the files
filepath="weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=
True, mode='auto')

#We will use the EarlyStopping callback to prevent the model from training after the optimal condi
tion has been met
earlystop = EarlyStopping(monitor = 'val_loss', patience = 2, verbose=1, mode = 'min')

#Passing validation data into the custom Metrics callback that we have created
validation_data=(X_test,Y_test)
metrics = Metrics(validation_data)

model.compile(optimizer=adam, loss='categorical_crossentropy',metrics=['accuracy'])
```

In [0]:

```
print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(14121, 8588) (14121, 20)
(4707, 8588) (4707, 20)
```

In [0]:

```
!rm -rf ./logs/fit
tf.keras.backend.clear_session()
```

In [0]:

```
#Creating logs direcctory to store information about the fits
log_dir = "logs/fit/" + datetime.now().strftime("%Y%m%d - %H%M%S")

tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True)

tbCallBack = TensorBoard(log_dir=log_dir, histogram_freq=1, write_graph=True, write_grads=True, wri
te_images=True)

allCs = [checkpoint, earlystop, tbCallBack, metrics]

model.fit(X_train,Y_train,epochs=12, validation_data=validation_data, batch_size=64,
callbacks=allCs)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
(4707, 8962) (4707, 20)
Epoch 1/12
221/221 [==============================] - ETA: 0s - loss: 2.5541 - accuracy: 0.1562
Epoch 00001: val_accuracy improved from -inf to 0.27109, saving model to weights-01-0.2711.hdf5
The F1 score is:  0.11819595645412131
221/221 [==============================] - 131s 591ms/step - loss: 2.5541 - accuracy: 0.1562 - val
_loss: 2.0650 - val_accuracy: 0.2711
Epoch 2/12
221/221 [==============================] - ETA: 0s - loss: 1.7432 - accuracy: 0.3645
Epoch 00002: val_accuracy improved from 0.27109 to 0.45039, saving model to weights-02-0.4504.hdf5
The F1 score is:  0.34089795918367344
221/221 [==============================] - 129s 584ms/step - loss: 1.7432 - accuracy: 0.3645 - val
_loss: 1.5556 - val_accuracy: 0.4504
Epoch 3/12
221/221 [==============================] - ETA: 0s - loss: 1.2694 - accuracy: 0.5384
Epoch 00003: val_accuracy improved from 0.45039 to 0.57489, saving model to weights-03-0.5749.hdf5
The F1 score is:  0.551743220807969
221/221 [==============================] - 129s 583ms/step - loss: 1.2694 - accuracy: 0.5384 - val
_loss: 1.2904 - val_accuracy: 0.5749
Epoch 4/12
221/221 [==============================] - ETA: 0s - loss: 0.9478 - accuracy: 0.6588
Epoch 00004: val_accuracy improved from 0.57489 to 0.62503, saving model to weights-04-0.6250.hdf5
The F1 score is:  0.6173508907823393
221/221 [==============================] - 129s 584ms/step - loss: 0.9478 - accuracy: 0.6588 - val
_loss: 1.1709 - val_accuracy: 0.6250
Epoch 5/12
221/221 [==============================] - ETA: 0s - loss: 0.7257 - accuracy: 0.7366
Epoch 00005: val_accuracy improved from 0.62503 to 0.64861, saving model to weights-05-0.6486.hdf5
The F1 score is:  0.6524857177585998
221/221 [==============================] - 129s 582ms/step - loss: 0.7257 - accuracy: 0.7366 - val
_loss: 1.1541 - val_accuracy: 0.6486
Epoch 6/12
221/221 [==============================] - ETA: 0s - loss: 0.5522 - accuracy: 0.7999
Epoch 00006: val_accuracy improved from 0.64861 to 0.68111, saving model to weights-06-0.6811.hdf5
The F1 score is:  0.6866510538641687
221/221 [==============================] - 129s 583ms/step - loss: 0.5522 - accuracy: 0.7999 - val
_loss: 1.1804 - val_accuracy: 0.6811
Epoch 7/12
221/221 [==============================] - ETA: 0s - loss: 0.4471 - accuracy: 0.8455
Epoch 00007: val_accuracy improved from 0.68111 to 0.68961, saving model to weights-07-0.6896.hdf5
The F1 score is:  0.695371862767672
221/221 [==============================] - 129s 583ms/step - loss: 0.4471 - accuracy: 0.8455 - val
_loss: 1.1893 - val_accuracy: 0.6896
Epoch 00007: early stopping
```

Out[0]:

```
<tensorflow.python.keras.callbacks.History at 0x7feeac053cc0>
```

In [0]:

```
import os

#Iterating through the list of the files to see which one has given the best score

files = os.listdir()
fnames = []
scores = []

for f in files:
  if 'weights' in f:
    fnames.append(f)
    f = f.split('.')
    scores.append(f[1])

scores = np.array(scores)
ind = np.argmax(scores)
best = fnames[ind]
print(best)
```

```
weights-07-0.6896.hdf5
```

In [0]:

```
#Saving the best model
model.load_weights(best)
model.save('best_model_1.h5')
```

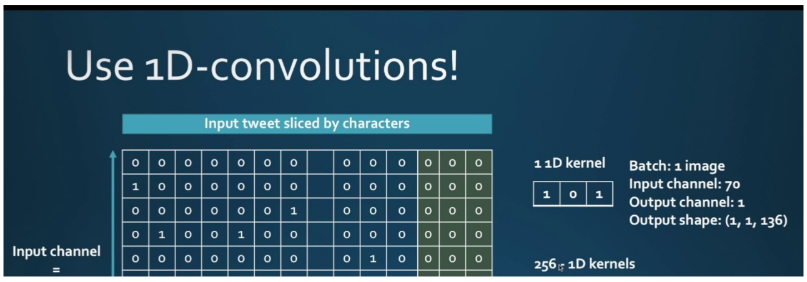
```
1. all are Conv1D layers with any number of filter and filter sizes, there is no
restriction on this.

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.
```

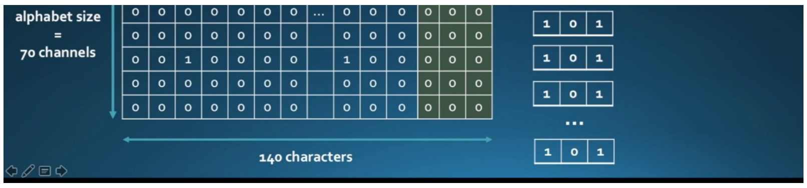
Observations

```
- The best validation accuracy is around 0.69. The micro F1 score is also decent (around
0.69)
```

- The optimizer that works best for this model is Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
- There is overfitting in this model as the train accuracy is almost 0.85
- On the whole it can be said that pre-trained word embeddings work decently well with 1-D convolutions

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

In [0]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test  = train_test_split(X, Y, stratify = Y, test_size = 0.25)
```

In [0]:

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

t = Tokenizer(filters='', char_level = True)
t.fit_on_texts(X_train)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_train = t.texts_to_sequences(X_train)
encoded_test = t.texts_to_sequences(X_test)
padded_train = pad_sequences(encoded_train, padding='post')
input_length = padded_train.shape[1]
print("input length",input_length)

padded_test = pad_sequences(encoded_test, maxlen = input_length, padding='post')
output_length = padded_test.shape[1]
print("output length",output_length)

glove_size = 300

embeddings_index = dict()
f = open('CharacterEmbeddings.txt')
for line in tqdm(f):
  values = line.split()
  char = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[char] = coefs
f.close()
```

```
input length 56788
output length 56788
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: TqdmDeprecationWarning: This
function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

In [0]:

```
X_train = padded_train
X_test = padded_test
print(X_test.shape)
```

```
(4707, 56788)
```

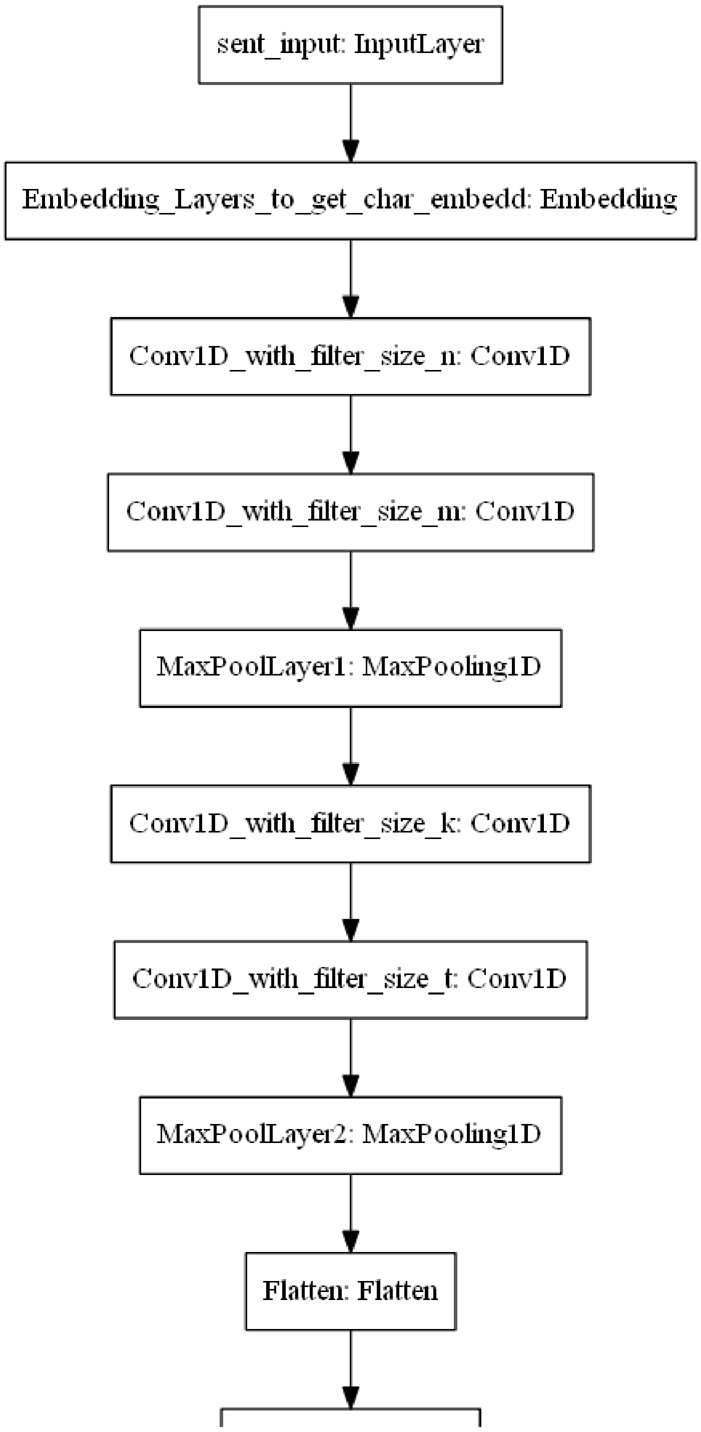
```
(4707, 30788)
```

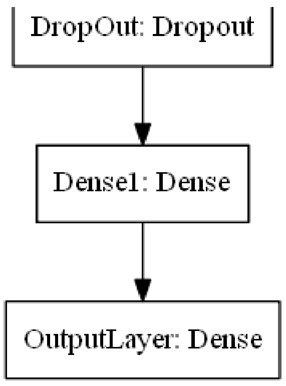
In [0]:

```
print('Loaded %s word vectors.' % len(embeddings_index))
# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 300))
for word, i in tqdm(t.word_index.items()):
 embedding_vector = embeddings_index.get(word)
 if embedding_vector is not None:
  embedding_matrix[i] = embedding_vector
```

```
Loaded 94 word vectors.
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: TqdmDeprecationWarning: This
function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  after removing the cwd from sys.path.
```


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
Flatten: Flatten



In [0]:

```
delta = Adadelta(learning_rate=1.0, rho=0.95)
adam = Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, amsgrad=False)
sgd = SGD(learning_rate=0.01, momentum=0.0, nesterov=False)
```

In [0]:

```
from tensorflow.keras.regularizers import l2
```

In [0]:

```
#BUILDING MODEL 1
import random as rn
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)

input_ = Input(shape=(input_length))
e = Embedding(vocab_size, 300, weights=[embedding_matrix], input_length = input_length, trainable =
False)(input_)

c13 = Conv1D(filters=16, kernel_size=32, kernel_initializer=he_normal(seed=0), kernel_regularizer=l
2(0.01), activation='relu')(e)
c15 = Conv1D(filters = 16, kernel_size=32, kernel_initializer=he_normal(seed=0), kernel_regularizer
=l2(0.01), activation='relu')(c13)

m1 = MaxPool1D(pool_size=2, strides=2)(c15)
c23 = Conv1D(filters=16, kernel_size=16, kernel_initializer=he_normal(seed=0), kernel_regularizer=l
2(0.01), activation='relu')(m1)
c25 = Conv1D(filters = 16, kernel_size=16, kernel_initializer=he_normal(seed=0), kernel_regularizer
=l2(0.01), activation='relu')(c23)

m2 = MaxPool1D(pool_size = 4, strides=2)(c25)
f = Flatten()(m2)
d = Dropout(rate = 0.5)(f)
d = Dense(units = 30, kernel_initializer=glorot_normal(seed=32), kernel_regularizer=l2(0.01), activ
ation='relu')(d)
output = Dense(units = 20, name = 'output', kernel_initializer=glorot_normal(seed=32), activation='
softmax')(d)

model2 = Model(inputs = input_, outputs = output)
model2.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 56788)]           0         
_________________________________________________________________
embedding (Embedding)        (None, 56788, 300)        8700      
_________________________________________________________________
conv1d (Conv1D)              (None, 56757, 16)         153616    
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 56726, 16)         8208      
_________________________________________________________________
```

```
max_pooling1d (MaxPooling1D) (None, 28363, 16)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 28348, 16)          4112
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 28333, 16)          4112
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 14165, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 226640)             0
_________________________________________________________________
dropout (Dropout)            (None, 226640)             0
_________________________________________________________________
dense (Dense)                (None, 30)                 6799230
_________________________________________________________________
output (Dense)               (None, 20)                 620
=================================================================
Total params: 6,978,598
Trainable params: 6,969,898
Non-trainable params: 8,700
_________________________________________________________________
```

In [0]:

```
filepath="weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_accuracy',  verbose=1, save_best_only=
True, mode='auto')

earlystop = EarlyStopping(monitor = 'val_loss', patience = 2, verbose=1, mode = 'min')

validation_data=(X_test,Y_test)
metrics = Metrics(validation_data)

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

In [0]:

```
!rm -rf ./logs/fit
tf.keras.backend.clear_session()
```

In [0]:

```
log_dir = "logs/fit/" + datetime.now().strftime("%Y%m%d - %H%M%S")

tf.keras.utils.plot_model(model2, to_file='model2.png', show_shapes=True)

tbCallBack = TensorBoard(log_dir=log_dir, histogram_freq=1, write_graph=True, write_grads=True, wri
te_images=True)

allCs = [checkpoint, earlystop, tbCallBack, metrics]

model2.fit(X_train,Y_train,epochs=30, validation_data=validation_data, batch_size=64, callbacks=all
Cs)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard` Callback.
(4707, 56788) (4707, 20)
Epoch 1/30
221/221 [==============================] - ETA: 0s - loss: 3.9873 - accuracy: 0.0677
Epoch 00001: val_accuracy improved from -inf to 0.06416, saving model to weights-01-0.0642.hdf5
The F1 score is:  0.0
221/221 [==============================] - 484s 2s/step - loss: 3.9873 - accuracy: 0.0677 - val_lo
ss: 3.6805 - val_accuracy: 0.0642
Epoch 2/30
221/221 [==============================] - ETA: 0s - loss: 3.5354 - accuracy: 0.0770
Epoch 00002: val_accuracy improved from 0.06416 to 0.07606, saving model to weights-02-0.0761.hdf5
The F1 score is:  0.0
221/221 [==============================] - 480s 2s/step - loss: 3.5354 - accuracy: 0.0770 - val_lo
ss: 3.4171 - val_accuracy: 0.0761
Epoch 3/30
221/221 [==============================] - ETA: 0s - loss: 3.3342 - accuracy: 0.0775
Epoch 00003: val_accuracy did not improve from 0.07606
The F1 score is:  0.0
221/221 [==============================] - 480s 2s/step - loss: 3.3342 - accuracy: 0.0775 - val_lo
ss: 3.2624 - val_accuracy: 0.0741
Epoch 4/30
Epoch 4/30
221/221 [==============================] - ETA: 0s - loss: 3.2092 - accuracy: 0.0833
Epoch 00004: val_accuracy did not improve from 0.07606
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 3.2092 - accuracy: 0.0833 - val_lo
ss: 3.1617 - val_accuracy: 0.0754
Epoch 5/30
221/221 [==============================] - ETA: 0s - loss: 3.1272 - accuracy: 0.0846
Epoch 00005: val_accuracy did not improve from 0.07606
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 3.1272 - accuracy: 0.0846 - val_lo
ss: 3.0997 - val_accuracy: 0.0744
Epoch 6/30
221/221 [==============================] - ETA: 0s - loss: 3.0768 - accuracy: 0.0830
Epoch 00006: val_accuracy did not improve from 0.07606
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 3.0768 - accuracy: 0.0830 - val_lo
ss: 3.0556 - val_accuracy: 0.0733
Epoch 7/30
221/221 [==============================] - ETA: 0s - loss: 3.0347 - accuracy: 0.0851
Epoch 00007: val_accuracy improved from 0.07606 to 0.08201, saving model to weights-07-0.0820.hdf5
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 3.0347 - accuracy: 0.0851 - val_lo
ss: 3.0210 - val_accuracy: 0.0820
Epoch 8/30
221/221 [==============================] - ETA: 0s - loss: 3.0150 - accuracy: 0.0876
Epoch 00008: val_accuracy improved from 0.08201 to 0.08455, saving model to weights-08-0.0846.hdf5
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 3.0150 - accuracy: 0.0876 - val_lo
ss: 3.0023 - val_accuracy: 0.0846
Epoch 9/30
221/221 [==============================] - ETA: 0s - loss: 2.9932 - accuracy: 0.0859
Epoch 00009: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9932 - accuracy: 0.0859 - val_lo
ss: 2.9928 - val_accuracy: 0.0775
Epoch 10/30
221/221 [==============================] - ETA: 0s - loss: 2.9845 - accuracy: 0.0835
Epoch 00010: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9845 - accuracy: 0.0835 - val_lo
ss: 2.9767 - val_accuracy: 0.0816
Epoch 11/30
221/221 [==============================] - ETA: 0s - loss: 2.9719 - accuracy: 0.0876
Epoch 00011: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9719 - accuracy: 0.0876 - val_lo
ss: 2.9710 - val_accuracy: 0.0803
Epoch 12/30
221/221 [==============================] - ETA: 0s - loss: 2.9628 - accuracy: 0.0857
Epoch 00012: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9628 - accuracy: 0.0857 - val_lo
ss: 2.9656 - val_accuracy: 0.0795
Epoch 13/30
221/221 [==============================] - ETA: 0s - loss: 2.9587 - accuracy: 0.0863
Epoch 00013: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9587 - accuracy: 0.0863 - val_lo
ss: 2.9602 - val_accuracy: 0.0799
Epoch 14/30
221/221 [==============================] - ETA: 0s - loss: 2.9548 - accuracy: 0.0872
Epoch 00014: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9548 - accuracy: 0.0872 - val_lo
ss: 2.9634 - val_accuracy: 0.0824
Epoch 15/30
221/221 [==============================] - ETA: 0s - loss: 2.9520 - accuracy: 0.0857
Epoch 00015: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9520 - accuracy: 0.0857 - val_lo
ss: 2.9537 - val_accuracy: 0.0780
Epoch 16/30
221/221 [==============================] - ETA: 0s - loss: 2.9504 - accuracy: 0.0869
Epoch 00016: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9504 - accuracy: 0.0869 - val_lo
ss: 2.9542 - val_accuracy: 0.0784
```

```
ss: 2.9542 - val_accuracy: 0.0784
Epoch 17/30
221/221 [==============================] - ETA: 0s - loss: 2.9498 - accuracy: 0.0853
Epoch 00017: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9498 - accuracy: 0.0853 - val_lo
ss: 2.9486 - val_accuracy: 0.0803
Epoch 18/30
221/221 [==============================] - ETA: 0s - loss: 2.9442 - accuracy: 0.0874
Epoch 00018: val_accuracy did not improve from 0.08455
The F1 score is:  0.0
221/221 [==============================] - 479s 2s/step - loss: 2.9442 - accuracy: 0.0874 - val_lo
ss: 2.9476 - val_accuracy: 0.0788
Epoch 19/30
221/221 [==============================] - ETA: 0s - loss: 2.9428 - accuracy: 0.0883
Epoch 00019: val_accuracy improved from 0.08455 to 0.08519, saving model to weights-19-0.0852.hdf5
The F1 score is:  0.0004247186239116585
221/221 [==============================] - 478s 2s/step - loss: 2.9428 - accuracy: 0.0883 - val_lo
ss: 2.9501 - val_accuracy: 0.0852
Epoch 20/30
 46/221 [=====>........................] - ETA: 4:25 - loss: 2.9378 - accuracy: 0.0873


---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-23-7b536f279e31> in <module>()
      7 allCs = [checkpoint, earlystop, tbCallBack, metrics]
      8 
----> 9 model2.fit(X_train,Y_train,epochs=30, validation_data=validation_data, batch_size=64, callb
acks=allCs)

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py in
_method_wrapper(self, *args, **kwargs)
     63   def _method_wrapper(self, *args, **kwargs):
     64     if not self._in_multi_worker_mode():  # pylint: disable=protected-access
---> 65       return method(self, *args, **kwargs)
     66 
     67     # Running inside `run_distribute_coordinator` already.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py in fit(self, x,
y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle,
class_weight, sample_weight, initial_epoch, steps_per_epoch, validation_steps,
validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing, **kwargs)
    781                 batch_size=batch_size):
    782               callbacks.on_train_batch_begin(step)
--> 783               tmp_logs = train_function(iterator)
    784               # Catch OutOfRangeError for Datasets of unknown size.
    785               # This blocks until the batch has finished executing.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py in __call__(self, *
args, **kwds)
    578         xla_context.Exit()
    579     else:
--> 580       result = self._call(*args, **kwds)
    581 
    582     if tracing_count == self._get_tracing_count():

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py in _call(self,
*args, **kwds)
    609       # In this case we have created variables on the first call, so we run the
    610       # defunned version which is guaranteed to never create variables.
--> 611       return self._stateless_fn(*args, **kwds)  # pylint: disable=not-callable
    612     elif self._stateful_fn is not None:
    613       # Release the lock early so that multiple threads can perform the call

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in __call__(self,
*args, **kwargs)
   2418     with self._lock:
   2419       graph_function, args, kwargs = self._maybe_define_function(args, kwargs)
-> 2420     return graph_function._filtered_call(args, kwargs)  # pylint: disable=protected-access
   2421 
   2422   @property

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in _filtered_call(self,
args, kwargs)
   1663          if isinstance(t, (ops.Tensor,
   1664                            resource_variable_ops.BaseResourceVariable))),
-> 1665         self.captured_inputs)
```

```
   1666
   1667   def _call_flat(self, args, captured_inputs, cancellation_manager=None):

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in _call_flat(self,
args, captured_inputs, cancellation_manager)
   1744       # No tape is watching; skip to running the function.
   1745       return self._build_call_outputs(self._inference_function.call(
-> 1746           ctx, args, cancellation_manager=cancellation_manager))
   1747     forward_backward = self._select_forward_and_backward_functions(
   1748         args,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in call(self, ctx,
args, cancellation_manager)
    596               inputs=args,
    597               attrs=attrs,
--> 598               ctx=ctx)
    599         else:
    600           outputs = execute.execute_with_cancellation(

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/execute.py in
quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     58     ctx.ensure_initialized()
     59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
---> 60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:
     62     if name is not None:

KeyboardInterrupt:
```

In [0]:

```
%tensorboard --logdir logs --host localhost
```

```
Output hidden; open in https://colab.research.google.com to view.
```

In [36]:

```
import os

#Iterating through the list of the files to see which one has given the best score

files = os.listdir()
fnames = []
scores = []

for f in files:
  if 'weights' in f:
    fnames.append(f)
    f = f.split('.')
    scores.append(f[1])

scores = np.array(scores)
ind = np.argmax(scores)
best = fnames[ind]
print(best)
```

```
weights-19-0.0852.hdf5
```

In [0]:

```
model2.load_weights(best)
model2.save('best_model_2.h5')
```

Observations

```
 - The validation accuracy is around 0.0852. The f1 score is terrible (0.000424)
```

- The best optimizer is Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999,

```
amsgrad=False)
```

- As we have used l2 regularization in the model, and the score is very less to begin with , there is not much overfitting.
- On the whole pretrained charcacter embeddings do not work well with pretr

In [0]:

```
https://www.geeksforgeeks.org/join-two-text-columns-into-a-single-column-in-pandas/
https://www.geeksforgeeks.org/create-a-pandas-dataframe-from-lists/
https://stackoverflow.com/questions/22216076/unicodedecodeerror-utf8-codec-cant-decode-byte-0xa5-in-position-0-invalid-s?lq=1
https://nlp.stanford.edu/projects/glove/
```