Text Classification:

Data

Import Section

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
import os
import glob
import pathlib
import datetime
from collections import Counter
from tqdm import tqdm
tqdm.pandas()
import re
import warnings
warnings.filterwarnings('ignore')
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

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

```
True
```

Download Data

```
#!pip install -U --no-cache-dir gdown --pre
!gdown 1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM
```

```
Downloading...
From: https://drive.google.com/uc?id=1rxD15nyeIPIAZ-J2VYPrDRZI66-TBWvM
To: /content/documents.rar
```

```
   0% 0.00/19.0M [00:00<?, ?B/s]  30% 5.77M/19.0M [00:00<00:00,
57.0MB/s] 100% 19.0M/19.0M [00:00<00:00, 134MB/s]

!unrar x /content/documents.rar
```

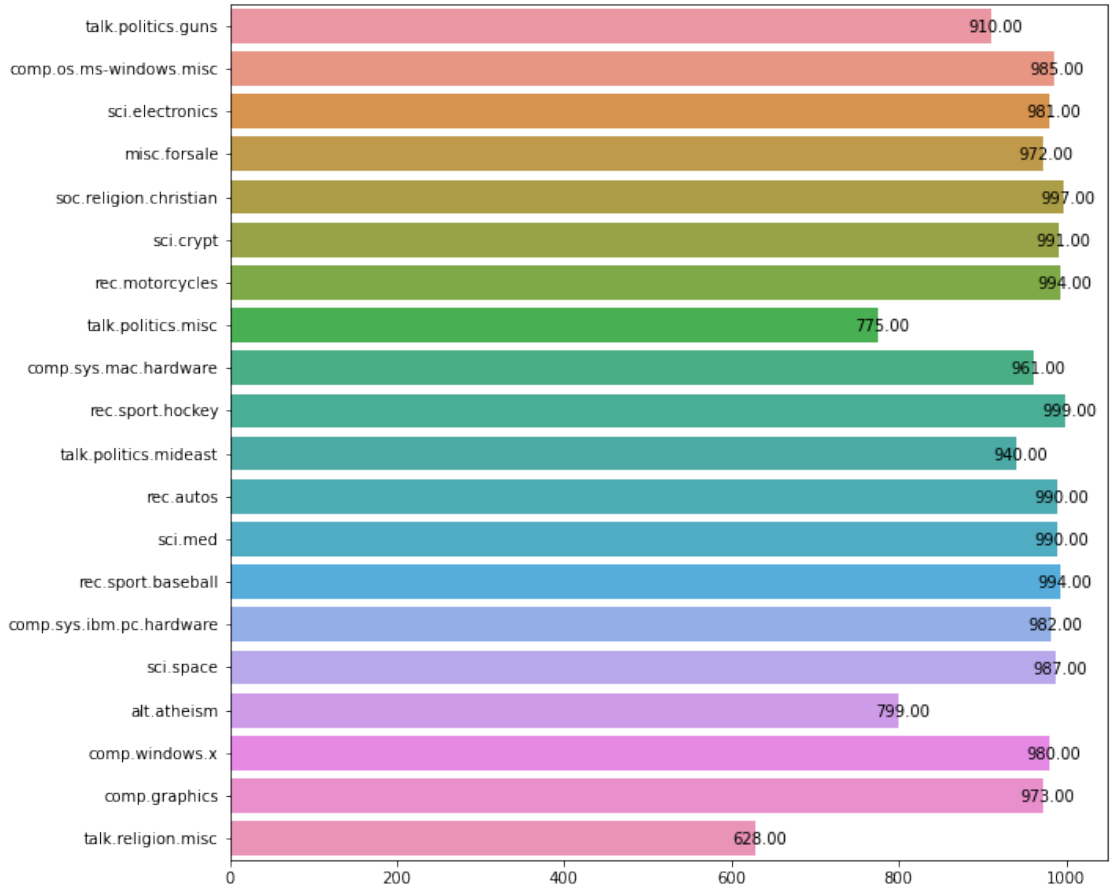
Lets Understand about the Data

```
data_dir_test = '/content/documents'
data_dir_test = pathlib.Path(data_dir_test)
Documents_list=list(data_dir_test.glob('*'))
print('Number of Documents : ',len(Documents_list))
```

```
Number of Documents :  18828
```

```
categories = []
for i in tqdm(Documents_list):
    categories.append((str(i).split('_')
[0]).replace('/content/documents/',''))
categories_dict = Counter(categories)
fig, ax = plt.subplots(figsize=(10,10))
plots = sns.barplot(y = list(categories_dict.keys()) , x =
list(categories_dict.values()),orient = 'h')
print('*'*50)
print("Number of Categories :" , len(np.unique(categories)))
print('*'*50)
print()
for p in ax.patches:
    width = p.get_width()
    plt.text(5+p.get_width(), p.get_y()+0.55*p.get_height(),
             '{:1.2f}'.format(width),
             ha='center', va='center')
plt.show()
```

```
100%|██████████| 18828/18828 [00:00<00:00, 510246.60it/s]

**************************************************
Number of Categories : 20
**************************************************
```



Lets Print Random Sample Document

```
with open(str(np.random.choice(Documents_list))) as f:
  print(f.read())
```

```
From: as010b@uhura.cc.rochester.edu (Tree of Schnopia)
Subject: Re: New Study Out On Gay Percentage

In <15440@optilink.COM> cramer@optilink.COM (Clayton Cramer) writes:

>In article <C5nAvn.F3p@murdoch.acc.Virginia.EDU>,
gsh7w@fermi.clas.Virginia.EDU (Greg Hennessy) writes:
>> In article <philC5n6D5.MK3@netcom.com> phil@netcom.com (Phil
Ronzone) writes:
>> #Tells you something about the fascist politics being
practiced ....
>>
>> Ah, ending discrimination is now fascism.
>>
>> -Greg Hennessy, University of Virginia

>When you force people to associate with others against their will,
>yes.
```

```
We're having to associate with you against our will.  This is fascism!


You don't have to associate with anyone against your will.  Go live in
a
cave.  We won't miss you.

Drewcifer


--
----bi    Andrew D. Simchik                      SCHNOPIA!
\ ----    as010b@uhura.cc.rochester.edu                    TreeWater
 \\  /
   \/     "Words Weren't Made For Cowards"--Happy Rhodes
```

Assignment:

sample document

Preprocessing:

```
data.columns
```

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

```
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
```

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

```
Input_Text= []
for document in tqdm(Documents_list):
    with open(str(document),'rb') as f:
      text = f.read().decode(errors='replace')
      Input_Text.append(text)
```

```
print('*'*30 +' Sample Doc ' + '*'*30)
print()
print('Number of Documents : ',len(Input_Text))
print('*'*30 +' Sample Doc ' + '*'*30)
print()
print(Input_Text[10])
```

```
100%|██████████| 18828/18828 [00:00<00:00, 55465.94it/s]

****************************** Sample Doc
******************************

Number of Documents :  18828
****************************** Sample Doc
******************************

From: asper@calvin.uucp (Alan E. Asper)
Subject: Re: Boom! Dog attack!

In article <BONG-230493121730@kfp-slac-mac.slac.stanford.edu>
BONG@slac.stanford.edu (Eric Bong) writes:
>  Nice ridin' Tex. I use the California DMV recommended technique:
>slow as you aproach said dog and wick it up as you pass. I've often

This must be the standard strategy that is taught, cuz that's what
they told
me to do in my Illinois MSF class. It works well, only you don't get
the
satisfaction of kicking the shit out of some rabid hell-beast.

Alan
```

```
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"""
    list_of_preprocessed_emails = []
    subject = []
    preprocessed_text = []
    for document in tqdm(Input_Text):
      splits = []
      reg = re.findall(r"[A-Za-z0-9_%+-.]+"
              r"@[A-Za-z0-9.-]+"
              r"\.[A-Za-z]{2,5}",document)
```

```
      for i in reg:
        document = document.replace(i,'')
        inter = [i for i in (i.lower().split('@')[1]).split('.')]
        inter_out = inter.copy()
        for word in inter:
          if len(word) <=2 or word == 'com':
            inter_out.remove(word)
        splits.append(inter_out)
      if 'Subject:' in document:
        document = document.replace('Subject: Subject:','Subject:')
        try:
          temp_subject = (document.split('Subject:')
[1].strip().splitlines()[0]).replace('Re:','').strip()
          subject.append(temp_subject)
          document =
document.replace(temp_subject,'').replace('Subject:','').replace('Re:'
,'')
        except :
          print(text)
      else:
        subject.append(np.nan)
      outsent = []
      for sentence in document.splitlines():
        if sentence.startswith('Write to:') or
sentence.startswith('From:'):
          pass
        else:
          outsent.append(sentence)
      document = (' '.join(outsent)).strip()
      reg_step6 = re.findall(r".*<(.*?)>.*",document)
      for i in reg_step6:
        document.replace(i,'')
      document = document.replace('<','').replace('>','').strip()
      reg_step7 = re.findall(r".*\((.*?)\).*",document)
      for i in reg_step7:
        document.replace(i,'')
      document = ' '.join((document.replace('\n',' ').replace('\t','
').replace('-',' ').replace('\\',' ').replace('\/','
').replace('(','').replace(')','').strip()).split())
      document_step9_split = document.split(':')[:-1]
      for i in document_step9_split:
        word = i.split(' ')[-1]+str(':')
        document = document.replace(word,'')
      # specific
      document = re.sub(r"won't", "will not", document)
      document = re.sub(r"can\'t", "can not", document)
      # general
      document = re.sub(r"n\'t", " not", document)
      document = re.sub(r"\'re", " are", document)
      document = re.sub(r"\'s", " is", document)
```

```
      document = re.sub(r"\'d", " would", document)
      document = re.sub(r"\'ll", " will", document)
      document = re.sub(r"\'t", " not", document)
      document = re.sub(r"\'ve", " have", document)
      document = re.sub(r"\'m", " am", document)
      document = ' '.join(document.split())
      #step 11
      for sent in nltk.sent_tokenize(document):
        for chunk in
nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
          if hasattr(chunk, 'label'):
            if chunk.label() == 'GPE':
              document = document.replace(' '.join(c[0] for c in
chunk),'_'.join(c[0] for c in chunk))
            elif chunk.label() == 'PERSON':
              document = document.replace(' '.join(c[0] for c in
chunk),'')
      document = ' '.join(document.split())
      #step 13 -17
      document = ''.join([i for i in document if not i.isdigit()])
      document_split_step13 = document.split()
      for word in document_split_step13:
        if word.startswith('_') or word.endswith('_') or
(word.startswith('_') or word.endswith('_')):
          document = document.replace(word, word.replace('_',''))
      for word in document_split_step13:
        if '_' in word:
          #print(word)
          splitwords = word.split('_')
          updatedword = '_'.join([i for i in splitwords if len(i)>2])
          #print(updatedword)
          document = document.replace(word,updatedword)
      document = ' '.join([i.lower() for i in document.split() if
(len(i) > 2 and len(i) < 15)])
      document = ' '.join([i for i in document.split() if
bool(re.match("^[A-Za-z_]*$",i))])
      preprocessed_text.append(document)
      list_of_preprocessed_emails.append(' '.join([item for sublist in
splits for item in sublist]))
    return (list_of_preprocessed_emails,subject,preprocessed_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.

```
sample_document = '/content/documents/alt.atheism_49960.txt'
```

```
with open(str(sample_document),'rb') as f:
      sample_text = f.read().decode(errors='replace')
preprocess([sample_text])
```

```
100%|██████████| 1/1 [00:00<00:00,  1.41it/s]
```

```
(['mantis netcom mantis'],
 ['Alt.Atheism FAQ: Atheist Resources'],
 ['archive alt atheism archive resources last december atheist
resources addresses atheist organizations usa freedom from religion
foundation fish bumper stickers and assorted other atheist
paraphernalia are available from the freedom from religion foundation
the evolution designs evolution designs sell the fish like the ones
stick their but with feet and the word written the deluxe moulded
plastic fish postpaid the people the san francisco bay area can get
fish from try mailing for net people who the price per american
atheist press aap publish various atheist books critiques the lists
biblical and one such book bible ball and american isbn bible contains
bible contradicts based the king version the prometheus books sell
books including see alternate address which may newer older prometheus
african americans for humanism organization promoting black secular
humanism and uncovering the history black they publish quarterly aah
united press 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 monthly magazine founded germany ibka bund der und berlin ibka
publish materialien und informationen zur politisches journal der und
ibka miz postfach berlin for atheist write der hannover fiction thomas
disch claus short the ultimate proof that all characters and events
are any similarity living dead gods walter canticle for one gem this
post atomic doomsday novel the monks who spent their lives copying
blueprints from filling the sheets paper with ink and leaving white
lines and edgar pangborn post atomic doomsday novel set clerical the
for forbids that anyone describe use any substance philip dick wrote
many philosophical and thought provoking short stories and his stories
are bizarre but very wrote mainly but wrote about truth and religion
rather than although often believed that had met some sort remained
amongst his the following are some fallible alien deity summons group
craftsmen and women remote planet raise giant cathedral from beneath
the when the deity begins demand faith from the pot healer unable
ironic and amusing maze noteworthy for its description technology
based the schizophrenic hero searches for the hidden mysteries gnostic
ity after reality fired into his brain pink laser beam unknown but
possibly divine accompanied his dogmatic and dismissively atheist
friend and assorted other odd divine invades making young woman
pregnant she returns from another star unfortunately she terminally
and must assisted dead man whose brain wired hour easy listening
margaret atwood handmaid story based the premise that the congress
mysteriously and quickly take charge the nation set the book the diary
woman life she tries live under the new women right own property and
their bank accounts are sinful luxuries are and the radio only used
```

for readings from the crimes are punished doctors who performed legal abortions the are hunted down and atwood writing style difficult get used but the tale grows more and more chilling goes various authors this somewhat dull and rambling work has often been probably worth only that you will know what all the fuss exists many different make sure you get the one true non fiction peter rosa although seems even catholic this very enlighting history papal fallacies german erste die dunkle seite des michael martin philosophical detailed and scholarly justification contains outstanding appendix defining terminology and usage this necessarily tendentious argues both for the belief the existence and also for belief the non existence includes great refutations the most challenging arguments for particular attention paid refuting contempory theists such and isbn paperback also available case against comprehensive critique which considers the best contemporary defences ity and ultimately demonstrates that they are unsupportable isbn turner the johns hopkins university usa subtitled origins unbelief examines the way which unbelief whether agnostic atheistic became mainstream alternative world focusses the period and while considering france and britain the emphasis and particularly new_england religious history secularization the intellectual history the fate single the belief that isbn hardcover paper george seldes editor great ballantine usa different concentrating statements and writings explicitly present the person philosophy and world includes obscure and often suppressed opinions from many for some popular traces the way which various people expressed and twisted the idea over the quite number the quotations are derived from cardiff and isbn paper richard swinburne existence revised oxford this book the second volume trilogy that began with coherence and was concluded with and this swinburne attempts construct series inductive arguments for the existence his which are somewhat tendentious and rely upon the imputation late century western values and aesthetics which supposedly simple can were decisively rejected miracle the revised edition existence swinburne includes appendix which makes somewhat incoherent attempt rebut mackie miracle oxford this posthumous volume contains comprehensive review the principal arguments for and against the existence ranges from the classical philosophical positions through the moral arguments kant and the recent restatements the classical theses and also addresses those positions which push the concept beyond the realm the such those and well for such the book delight read less formalistic and better written than and refreshingly direct when compared with the hand waving haught illustrated history religious murder and prometheus looks religious persecution from ancient times the present day and not only library congress catalog card number norm american see the listing for african americans for humanism gordon stein anthology atheism and prometheus anthology covering wide range including and and history comprehensive edmund cohen mind the bible prometheus study why people become and what effect has net resources there small mail based archive server which carries archives old articles and assorted other for more send mail saying help send and will mail back mathew'])

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

```
list_of_preprocessed_emails,subject,text = preprocess(Input_Text)
document_dataset = pd.DataFrame(data =
list(zip(Input_Text,categories,text,subject,list_of_preprocessed_email
s)),columns = ['text', 'class', 'preprocessed_text',
'preprocessed_subject','preprocessed_emails'])
document_dataset.head()
```

```
100%|██████████| 18828/18828 [35:54<00:00,  8.74it/s]
```

```
                                                   text
class  \
0  From: crphilli@hound.dazixca.ingr.com (Ron Phi...
talk.politics.guns
1  From: FL2G@gandalf.fl.bs.dlr.de (Reiner Suikat...  comp.os.ms-
windows.misc
2  From: schuster@panix.com (Michael Schuster)\nS...
sci.electronics
3  From: whit@carson.u.washington.edu (John Whitm...
sci.electronics
4  From: oeth6050@iscsvax.uni.edu\nSubject: ****C...
misc.forsale

                                  preprocessed_text  \
0  this was posted the firearms politics mailing ...
1  having problem with truetype fonts windows hav...
2  article neil gandler how does the radio electr...
3  article suppose have boolean function which mi...
4  name and have the following comic books for sa...

                     preprocessed_subject  \
0              Randy Weaver Trail - Day 3
1     TrueType font mix-up Times=>Cyrillic
2  Radio Electronics Free information card
3                  minimal boolean circuit
4                  ****COMIC BOOK SALE****

                                preprocessed_emails
0             hound dazixca ingr hound dazixca ingr
1                          gandalf dlr gandalf dlr
2  panix acsu buffalo edu ubvmsb buffalo edu pani...
3  carson washington edu ringer utsa edu ringer u...
4    iscsvax uni edu iscsvax uni edu iscsvax uni edu
```

```
document_dataset.to_csv('final_dataset.csv')
```

```
document_dataset = pd.read_csv('/content/final_dataset.csv')
document_dataset = document_dataset.iloc[:,1:]
document_dataset.head()
```

```
                                                text
class  \
0  From: crphilli@hound.dazixca.ingr.com (Ron Phi...
talk.politics.guns
1  From: FL2G@gandalf.fl.bs.dlr.de (Reiner Suikat...  comp.os.ms-
windows.misc
2  From: schuster@panix.com (Michael Schuster)\nS...
sci.electronics
3  From: whit@carson.u.washington.edu (John Whitm...
sci.electronics
4  From: oeth6050@iscsvax.uni.edu\nSubject: ****C...
misc.forsale

                                    preprocessed_text  \
0  this was posted the firearms politics mailing ...
1  having problem with truetype fonts windows hav...
2  article neil gandler how does the radio electr...
3  article suppose have boolean function which mi...
4  name and have the following comic books for sa...

                      preprocessed_subject  \
0               Randy Weaver Trail - Day 3
1     TrueType font mix-up Times=>Cyrillic
2  Radio Electronics Free information card
3                  minimal boolean circuit
4                  ****COMIC BOOK SALE****

                                 preprocessed_emails
0             hound dazixca ingr hound dazixca ingr
1                           gandalf dlr gandalf dlr
2  panix acsu buffalo edu ubvmsb buffalo edu pani...
3  carson washington edu ringer utsa edu ringer u...
4    iscsvax uni edu iscsvax uni edu iscsvax uni edu
```

Training The models to Classify:

Model-1: Using 1D convolutions with word embeddings

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'

```
document_dataset.head()
```

```
                                                text
class  \
0  From: crphilli@hound.dazixca.ingr.com (Ron Phi...
talk.politics.guns
```

```
1  From: FL2G@gandalf.fl.bs.dlr.de (Reiner Suikat...  comp.os.ms-
windows.misc
2  From: schuster@panix.com (Michael Schuster)\nS...
sci.electronics
3  From: whit@carson.u.washington.edu (John Whitm...
sci.electronics
4  From: oeth6050@iscsvax.uni.edu\nSubject: ****C...
misc.forsale

                                  preprocessed_text  \
0  this was posted the firearms politics mailing ...
1  having problem with truetype fonts windows hav...
2  article neil gandler how does the radio electr...
3  article suppose have boolean function which mi...
4  name and have the following comic books for sa...

                     preprocessed_subject  \
0              Randy Weaver Trail - Day 3
1     TrueType font mix-up Times=>Cyrillic
2  Radio Electronics Free information card
3                  minimal boolean circuit
4                  ****COMIC BOOK SALE****

                                preprocessed_emails
0             hound dazixca ingr hound dazixca ingr
1                           gandalf dlr gandalf dlr
2  panix acsu buffalo edu ubvmsb buffalo edu pani...
3  carson washington edu ringer utsa edu ringer u...
4    iscsvax uni edu iscsvax uni edu iscsvax uni edu
```

```
def add_requiredcols(document):
  return str(document[2]) + ' ' + str(document[3]) + ' ' +
str(document[4])
document_dataset['complete_text'] =
document_dataset.progress_apply(add_requiredcols,axis = 1)
document_dataset.head()
```

```
100%|██████████| 18828/18828 [00:00<00:00, 72636.52it/s]

                                                  text
class  \
0  From: crphilli@hound.dazixca.ingr.com (Ron Phi...
talk.politics.guns
1  From: FL2G@gandalf.fl.bs.dlr.de (Reiner Suikat...  comp.os.ms-
windows.misc
2  From: schuster@panix.com (Michael Schuster)\nS...
sci.electronics
3  From: whit@carson.u.washington.edu (John Whitm...
sci.electronics
4  From: oeth6050@iscsvax.uni.edu\nSubject: ****C...
```

```
misc.forsale

                                    preprocessed_text  \
0  this was posted the firearms politics mailing ...
1  having problem with truetype fonts windows hav...
2  article neil gandler how does the radio electr...
3  article suppose have boolean function which mi...
4  name and have the following comic books for sa...

                      preprocessed_subject  \
0               Randy Weaver Trail - Day 3
1     TrueType font mix-up Times=>Cyrillic
2  Radio Electronics Free information card
3                  minimal boolean circuit
4                  ****COMIC BOOK SALE****

                                 preprocessed_emails  \
0             hound dazixca ingr hound dazixca ingr
1                           gandalf dlr gandalf dlr
2  panix acsu buffalo edu ubvmsb buffalo edu pani...
3  carson washington edu ringer utsa edu ringer u...
4    iscsvax uni edu iscsvax uni edu iscsvax uni edu

                                       complete_text
0  this was posted the firearms politics mailing ...
1  having problem with truetype fonts windows hav...
2  article neil gandler how does the radio electr...
3  article suppose have boolean function which mi...
4  name and have the following comic books for sa...
```

```
document_dataset = document_dataset[['complete_text','class']]
document_dataset.head()
```

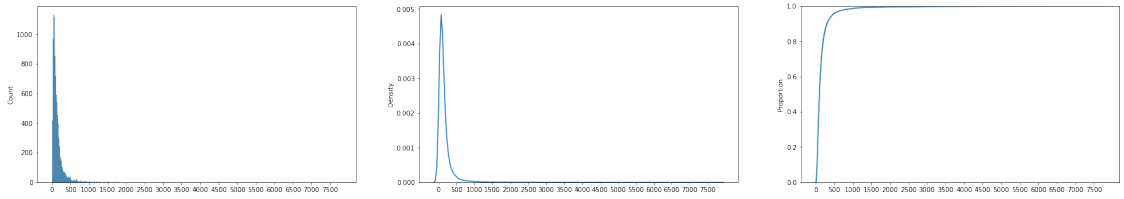
```
                                       complete_text
class
0  this was posted the firearms politics mailing ...
talk.politics.guns
1  having problem with truetype fonts windows hav...  comp.os.ms-
windows.misc
2  article neil gandler how does the radio electr...
sci.electronics
3  article suppose have boolean function which mi...
sci.electronics
4  name and have the following comic books for sa...
misc.forsale
```

Lets understand the word length of the data

```
words_length = []
for document in tqdm(document_dataset.complete_text):
  words_length.append(len(document.split()))
```

```
fig, axs = plt.subplots(ncols=3,figsize=(30,5))
a = sns.histplot(data = words_length,ax=axs[0])
a.set_xticks(range(0,8000,500))
b = sns.kdeplot(data = words_length,ax=axs[1])
b.set_xticks(range(0,8000,500))
c = sns.ecdfplot(data = words_length,ax=axs[2])
c.set_xticks(range(0,8000,500))
plt.show()
```

```
100%|██████████| 18828/18828 [00:00<00:00, 119618.16it/s]
```



We can clearly see that 99% of the sentences have words less than 1000.

Lets only consider maximum common words as 20000

```
max_sequence_len = 1000
max_words = 20000
```

```
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(nb_words =max_words,filters='!"#$
%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
tokenizer.fit_on_texts(document_dataset.complete_text)
sequences =
tokenizer.texts_to_sequences(document_dataset.complete_text)

word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 71002 unique tokens.
```

```
data = pad_sequences(sequences, maxlen=max_sequence_len)

map_dict = {}
for k in enumerate(list(np.unique(document_dataset['class'].values))):
  map_dict[k[1]] = k[0]
print(map_dict)
```

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

```
document_dataset['class'] = document_dataset['class'].map(map_dict)
document_dataset.head()
```

	complete_text	class
0	this was posted the firearms politics mailing ...	16
1	having problem with truetype fonts windows hav...	2
2	article neil gandler how does the radio electr...	12
3	article suppose have boolean function which mi...	12
4	name and have the following comic books for sa...	6

```
from keras.utils import to_categorical
labels = to_categorical(document_dataset['class'].values)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
```

```
Shape of data tensor: (18828, 1000)
Shape of label tensor: (18828, 20)
```

```
# split the data into a training set and a validation set
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
nb_validation_samples = int(0.25 * data.shape[0])

x_train = data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]
```

Lets Prepare the embedding layer

```
!gdown https://nlp.stanford.edu/data/glove.840B.300d.zip
```

```
Downloading...
From: https://nlp.stanford.edu/data/glove.840B.300d.zip
To: /content/glove.840B.300d.zip
100% 2.18G/2.18G [08:31<00:00, 4.26MB/s]
```

```
!unzip /content/glove.840B.300d.zip
```

```
Archive:  /content/glove.840B.300d.zip
  inflating: glove.840B.300d.txt
```

```
embeddings_index = {}
f = open(r'/content/glove.840B.300d.txt',  "r",encoding="utf8")
for line in tqdm(f):
    values = line.split()
    word = ''.join(values[:-300])
    #print(len(values[-300:]))
    coefs = np.asarray(values[-300:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
```

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

```
2196017it [02:03, 17762.06it/s]

Found 2195892 word vectors.
```

```
embedding_dim = 300

embedding_matrix = np.zeros((len(word_index) + 1, embedding_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.
        embedding_matrix[i] = embedding_vector

del embeddings_index


import pickle

# Open a file and use dump()
with open('embedded_matrix.pkl', 'wb') as file:
    # A new file will be created
    pickle.dump(embedding_matrix, file)

with open('/content/embedded_matrix.pkl', 'rb') as file:
    # Call load method to deserialze
    embedding_matrix = pickle.load(file)

from keras.layers import Embedding

embedding_layer = Embedding(len(word_index) + 1,
                            embedding_dim,
                            weights=[embedding_matrix],
                            input_length=max_sequence_len,
                            trainable=False)
```

Lets Train Model

```
output_shape = 20

from keras.layers import
Input,Conv1D,Concatenate,MaxPooling1D,Flatten,Dropout,Dense
from keras import Model
sequence_input = Input(shape=(max_sequence_len,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
layer1_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(embedded_sequences)
layer1_conv2 = Conv1D(16, 5,
```

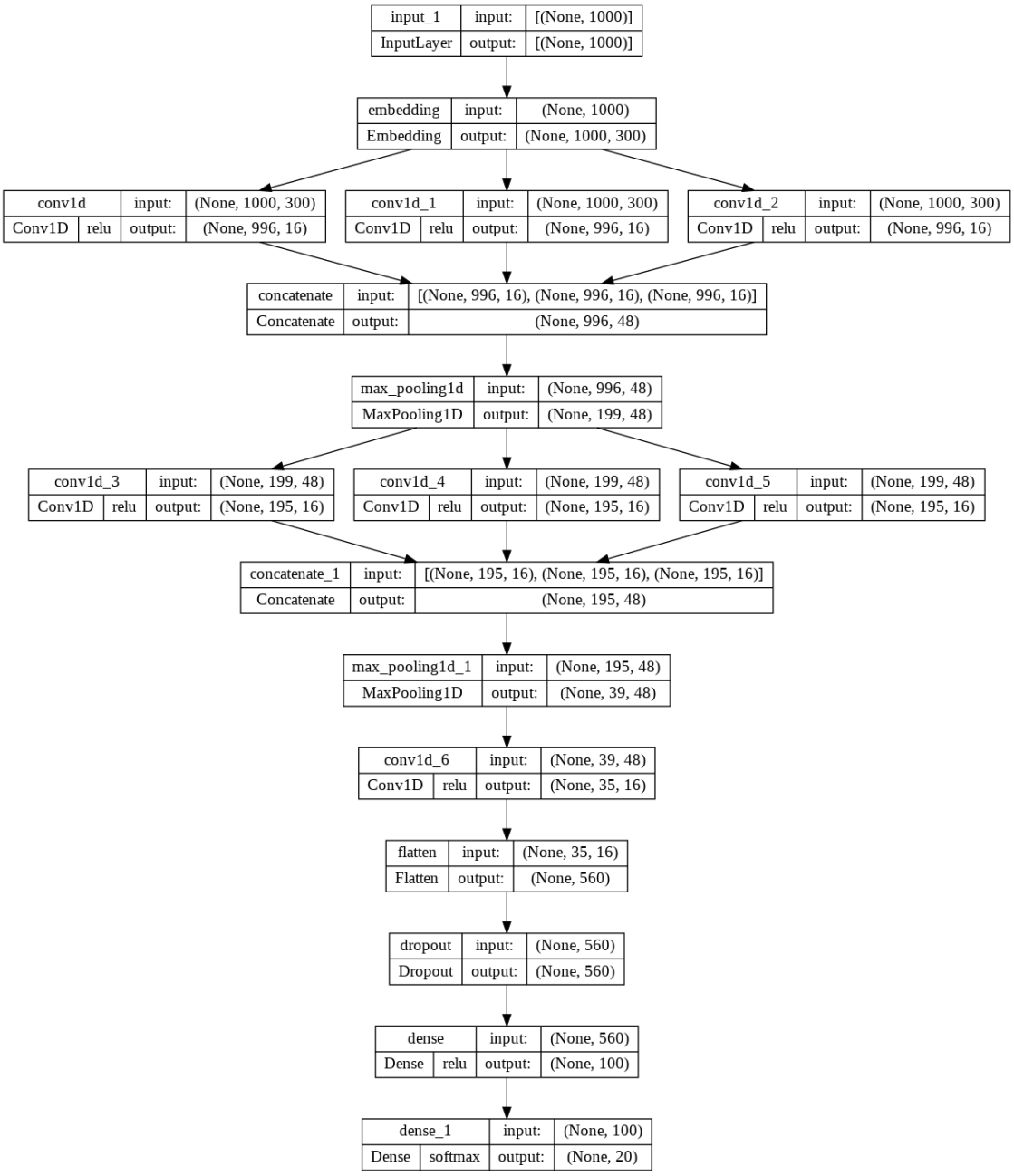
```
activation='relu',kernel_initializer="he_normal")(embedded_sequences)
layer1_conv3 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(embedded_sequences)
concatenate_1 = Concatenate()([layer1_conv1,
layer1_conv2,layer1_conv3])
maxpooling_1 = MaxPooling1D(5)(concatenate_1)
layer2_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(maxpooling_1)
layer2_conv2 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(maxpooling_1)
layer2_conv3 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(maxpooling_1)
concatenate_2 = Concatenate()([layer2_conv1,
layer2_conv2,layer2_conv3])
maxpooling_2 = MaxPooling1D(5)(concatenate_2)
layer3_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(maxpooling_2)
flatten_1 = Flatten()(layer3_conv1)
dropout_1 = Dropout(0.5)(flatten_1)
dense_1 = Dense(100,activation='relu',kernel_initializer="he_normal")
(dropout_1)
output = Dense(output_shape, activation='softmax')(dense_1)
model = Model(sequence_input, output)


model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================
==========================
 input_1 (InputLayer)        [(None, 1000)]            0
[]



 embedding (Embedding)       (None, 1000, 300)         21300900
['input_1[0][0]']


 conv1d (Conv1D)             (None, 996, 16)           24016
['embedding[0][0]']


 conv1d_1 (Conv1D)           (None, 996, 16)           24016
['embedding[0][0]']
```

```
 conv1d_2 (Conv1D)              (None, 996, 16)      24016
['embedding[0][0]']


 concatenate (Concatenate)      (None, 996, 48)      0
['conv1d[0][0]',

'conv1d_1[0][0]',

'conv1d_2[0][0]']


 max_pooling1d (MaxPooling1D)   (None, 199, 48)      0
['concatenate[0][0]']


 conv1d_3 (Conv1D)              (None, 195, 16)      3856
['max_pooling1d[0][0]']


 conv1d_4 (Conv1D)              (None, 195, 16)      3856
['max_pooling1d[0][0]']


 conv1d_5 (Conv1D)              (None, 195, 16)      3856
['max_pooling1d[0][0]']


 concatenate_1 (Concatenate)    (None, 195, 48)      0
['conv1d_3[0][0]',

'conv1d_4[0][0]',

'conv1d_5[0][0]']


 max_pooling1d_1 (MaxPooling1D)  (None, 39, 48)      0
['concatenate_1[0][0]']


 conv1d_6 (Conv1D)              (None, 35, 16)       3856
['max_pooling1d_1[0][0]']


 flatten (Flatten)              (None, 560)          0
['conv1d_6[0][0]']
```

```
 dropout (Dropout)              (None, 560)          0
['flatten[0][0]']


 dense (Dense)                  (None, 100)          56100
['dropout[0][0]']


 dense_1 (Dense)                (None, 20)           2020
['dense[0][0]']


==================================================================================
============================
Total params: 21,446,492
Trainable params: 145,592
Non-trainable params: 21,300,900
__________________________________________________________________________________
_____________________________
```

```
from tensorflow.keras.utils import plot_model
plot_model(model,to_file = 'model_1.png',show_shapes =
True,show_layer_activations = True)
```




```
from tensorflow.keras.callbacks import
ModelCheckpoint,EarlyStopping,LearningRateScheduler,ReduceLROnPlateau,
TensorBoard
import datetime
import os

earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.01,
patience=3, verbose=1)

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

```
log_dir = os.path.join("logs",'fits',
datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback =
TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)
%reload_ext tensorboard

filepath="model_1/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath,
monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')

!pip install tensorflow-addons
import tensorflow_addons as tfa
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow-addons in
/usr/local/lib/python3.8/dist-packages (0.19.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.8/dist-packages (from tensorflow-addons) (21.3)
Requirement already satisfied: typeguard>=2.7 in
/usr/local/lib/python3.8/dist-packages (from tensorflow-addons)
(2.7.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.8/dist-packages (from packaging->tensorflow-
addons) (3.0.9)
```

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics
=['accuracy',tfa.metrics.F1Score(num_classes=20,average='micro',thresh
old=0.5)])

model.fit(x_train, y_train, validation_data=(x_val, y_val),epochs=10,
batch_size=128,callbacks=[earlystop,checkpoint,tensorboard_callback])
```

```
Epoch 1/10
111/111 [==============================] - ETA: 0s - loss: 2.8672 -
accuracy: 0.0864 - f1_score: 0.0077
Epoch 1: val_accuracy improved from -inf to 0.13469, saving model to
model_1/weights-01-0.1347.hdf5
111/111 [==============================] - 24s 135ms/step - loss:
2.8672 - accuracy: 0.0864 - f1_score: 0.0077 - val_loss: 2.5434 -
val_accuracy: 0.1347 - val_f1_score: 0.0118
Epoch 2/10
111/111 [==============================] - ETA: 0s - loss: 2.3516 -
accuracy: 0.1924 - f1_score: 0.0645
Epoch 2: val_accuracy improved from 0.13469 to 0.27810, saving model
to model_1/weights-02-0.2781.hdf5
111/111 [==============================] - 11s 104ms/step - loss:
2.3516 - accuracy: 0.1924 - f1_score: 0.0645 - val_loss: 2.0193 -
val_accuracy: 0.2781 - val_f1_score: 0.1276
Epoch 3/10
111/111 [==============================] - ETA: 0s - loss: 1.9430 -
```

```
accuracy: 0.2973 - f1_score: 0.1562
Epoch 3: val_accuracy improved from 0.27810 to 0.39919, saving model
to model_1/weights-03-0.3992.hdf5
111/111 [==============================] - 9s 83ms/step - loss: 1.9430
- accuracy: 0.2973 - f1_score: 0.1562 - val_loss: 1.6443 -
val_accuracy: 0.3992 - val_f1_score: 0.2221
Epoch 4/10
110/111 [============================>.] - ETA: 0s - loss: 1.6412 -
accuracy: 0.4078 - f1_score: 0.2754
Epoch 4: val_accuracy improved from 0.39919 to 0.47419, saving model
to model_1/weights-04-0.4742.hdf5
111/111 [==============================] - 8s 76ms/step - loss: 1.6405
- accuracy: 0.4080 - f1_score: 0.2757 - val_loss: 1.3966 -
val_accuracy: 0.4742 - val_f1_score: 0.3407
Epoch 5/10
111/111 [==============================] - ETA: 0s - loss: 1.3880 -
accuracy: 0.4873 - f1_score: 0.3893
Epoch 5: val_accuracy improved from 0.47419 to 0.53601, saving model
to model_1/weights-05-0.5360.hdf5
111/111 [==============================] - 8s 76ms/step - loss: 1.3880
- accuracy: 0.4873 - f1_score: 0.3893 - val_loss: 1.2481 -
val_accuracy: 0.5360 - val_f1_score: 0.4526
Epoch 6/10
110/111 [============================>.] - ETA: 0s - loss: 1.2070 -
accuracy: 0.5513 - f1_score: 0.4862
Epoch 6: val_accuracy improved from 0.53601 to 0.62694, saving model
to model_1/weights-06-0.6269.hdf5
111/111 [==============================] - 9s 77ms/step - loss: 1.2072
- accuracy: 0.5512 - f1_score: 0.4861 - val_loss: 1.0778 -
val_accuracy: 0.6269 - val_f1_score: 0.5819
Epoch 7/10
111/111 [==============================] - ETA: 0s - loss: 1.0537 -
accuracy: 0.6139 - f1_score: 0.5746
Epoch 7: val_accuracy improved from 0.62694 to 0.63650, saving model
to model_1/weights-07-0.6365.hdf5
111/111 [==============================] - 8s 76ms/step - loss: 1.0537
- accuracy: 0.6139 - f1_score: 0.5746 - val_loss: 1.0349 -
val_accuracy: 0.6365 - val_f1_score: 0.6070
Epoch 8/10
111/111 [==============================] - ETA: 0s - loss: 0.9531 -
accuracy: 0.6553 - f1_score: 0.6266
Epoch 8: val_accuracy improved from 0.63650 to 0.68048, saving model
to model_1/weights-08-0.6805.hdf5
111/111 [==============================] - 8s 76ms/step - loss: 0.9531
- accuracy: 0.6553 - f1_score: 0.6266 - val_loss: 0.9676 -
val_accuracy: 0.6805 - val_f1_score: 0.6579
Epoch 9/10
110/111 [============================>.] - ETA: 0s - loss: 0.8705 -
accuracy: 0.6812 - f1_score: 0.6615
Epoch 9: val_accuracy improved from 0.68048 to 0.69662, saving model
```

```
to model_1/weights-09-0.6966.hdf5
111/111 [==============================] - 9s 77ms/step - loss: 0.8707
- accuracy: 0.6810 - f1_score: 0.6614 - val_loss: 0.9477 -
val_accuracy: 0.6966 - val_f1_score: 0.6794
Epoch 10/10
110/111 [============================>.] - ETA: 0s - loss: 0.7876 -
accuracy: 0.7148 - f1_score: 0.6989
Epoch 10: val_accuracy improved from 0.69662 to 0.70130, saving model
to model_1/weights-10-0.7013.hdf5
111/111 [==============================] - 9s 78ms/step - loss: 0.7888
- accuracy: 0.7147 - f1_score: 0.6990 - val_loss: 0.9404 -
val_accuracy: 0.7013 - val_f1_score: 0.6954

<keras.callbacks.History at 0x7fe88e558820>
```

```
values = model.evaluate(x_val, y_val,verbose =0)
print('Loss of the model : ',values[0])
print('accuracy of the model : ',values[1])
print('f1-score of the model : ',values[2])
```

```
Loss of the model :  0.9404045939445496
accuracy of the model :  0.7012959718704224
f1-score of the model :  0.6954314708709717
```

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

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

Model-2 : Using 1D convolutions with character embedding

```
import tensorflow as tf
tf.compat.v1.reset_default_graph()
```

```
document_dataset = document_dataset[['complete_text','class']]
document_dataset.head()
```

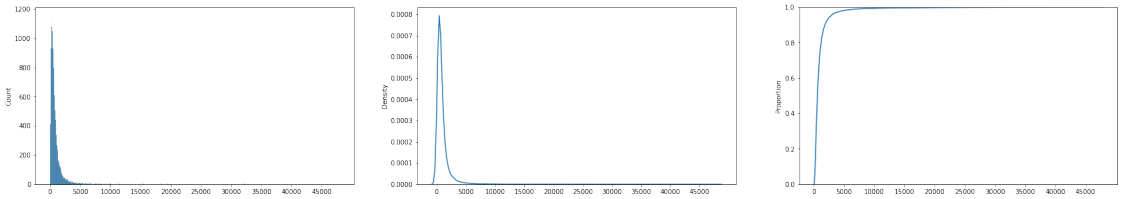
	complete_text	class
0	this was posted the firearms politics mailing ...	16
1	having problem with truetype fonts windows hav...	2
2	article neil gandler how does the radio electr...	12
3	article suppose have boolean function which mi...	12
4	name and have the following comic books for sa...	6

Lets understand the charector length of the data

```
char_length = []
for document in tqdm(document_dataset.complete_text):
  char_length.append(len(str(document)))
fig, axs = plt.subplots(ncols=3,figsize=(30,5))
a = sns.histplot(data = char_length,ax=axs[0])
a.set_xticks(range(0,50000,5000))
```

```
b = sns.kdeplot(data = char_length,ax=axs[1])
b.set_xticks(range(0,50000,5000))
c = sns.ecdfplot(data = char_length,ax=axs[2])
c.set_xticks(range(0,50000,5000))
plt.show()
```

```
100%|██████████| 18828/18828 [00:00<00:00, 632952.80it/s]
```



We can clearly see that 99% of the sentences have charectors less than 5000.

```
max_char_len = 5000

tokenizer_char = Tokenizer(nb_words =None,filters='!"#$
%&()*+,-./:;<=>?@[\\]^`{|}~\t\n',char_level=True)
tokenizer_char.fit_on_texts(document_dataset.complete_text)
char_sequences =
tokenizer_char.texts_to_sequences(document_dataset.complete_text)

np.array(char_sequences).shape
```

```
(18828,)
```

```
char_data = pad_sequences(char_sequences, maxlen=max_char_len)

char_data.shape
```

```
(18828, 5000)
```

```
map_dict = {}
for k in enumerate(list(np.unique(document_dataset['class'].values))):
  map_dict[k[1]] = k[0]
print(map_dict)
```

```
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 8: 8, 9: 9, 10: 10,
11: 11, 12: 12, 13: 13, 14: 14, 15: 15, 16: 16, 17: 17, 18: 18, 19:
19}
```

```
document_dataset['class'] = document_dataset['class'].map(map_dict)
document_dataset.head()
```

	complete_text	class
0	this was posted the firearms politics mailing ...	16
1	having problem with truetype fonts windows hav...	2
2	article neil gandler how does the radio electr...	12
3	article suppose have boolean function which mi...	12
4	name and have the following comic books for sa...	6

```
labels = to_categorical(document_dataset['class'].values)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
```

```
Shape of data tensor: (18828, 1000)
Shape of label tensor: (18828, 20)
```

```
# split the data into a training set and a validation set
indices = np.arange(char_data.shape[0])
np.random.shuffle(indices)
char_data = char_data[indices]
labels = labels[indices]
nb_validation_samples = int(0.25 * char_data.shape[0])

x_train = char_data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = char_data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]

print(x_train.shape)
print(x_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(14121, 5000)
(4707, 5000)
(14121, 20)
(4707, 20)
```

Lets Prepare the embedding layer

```
!gdown https://github.com/minimaxir/char-
embeddings/archive/refs/heads/master.zip
```

```
Downloading...
From:
https://github.com/minimaxir/char-embeddings/archive/refs/heads/master
.zip
To: /content/master.zip
11.4MB [00:02, 5.23MB/s]
```

```
!unzip '/content/master.zip'
```

```
Archive:  /content/master.zip
860c92a0af3b13c525c33d7257ef0204aaf80e1c
   creating: char-embeddings-master/
 extracting: char-embeddings-master/.gitignore
  inflating: char-embeddings-master/LICENSE
  inflating: char-embeddings-master/README.md
  inflating: char-embeddings-master/char-tsne-embed.png
  inflating: char-embeddings-master/create_embeddings.py
  inflating: char-embeddings-master/create_magic_text.py
```

```
  inflating: char-embeddings-master/glove.840B.300d-char.txt
  inflating: char-embeddings-master/magic_cards.txt
  inflating: char-embeddings-master/model.png
   creating: char-embeddings-master/output/
  inflating: char-embeddings-master/output/char-embeddings.txt
  inflating: char-embeddings-master/output/iter-01-0_9204.txt
  inflating: char-embeddings-master/output/iter-02-0_6075.txt
  inflating: char-embeddings-master/output/iter-03-0_5305.txt
  inflating: char-embeddings-master/output/iter-04-0_4860.txt
  inflating: char-embeddings-master/output/iter-05-0_4553.txt
  inflating: char-embeddings-master/output/iter-06-0_4315.txt
  inflating: char-embeddings-master/output/iter-07-0_4132.txt
  inflating: char-embeddings-master/output/iter-08-0_3970.txt
  inflating: char-embeddings-master/output/iter-09-0_3840.txt
  inflating: char-embeddings-master/output/iter-10-0_3728.txt
  inflating: char-embeddings-master/output/iter-11-0_3626.txt
  inflating: char-embeddings-master/output/iter-12-0_3536.txt
  inflating: char-embeddings-master/output/iter-13-0_3459.txt
  inflating: char-embeddings-master/output/iter-14-0_3381.txt
  inflating: char-embeddings-master/output/iter-15-0_3316.txt
  inflating: char-embeddings-master/output/iter-16-0_3257.txt
  inflating: char-embeddings-master/output/iter-17-0_3199.txt
  inflating: char-embeddings-master/output/iter-18-0_3145.txt
  inflating: char-embeddings-master/output/iter-19-0_3097.txt
  inflating: char-embeddings-master/output/iter-20-0_3052.txt
  inflating: char-embeddings-master/output/log.csv
  inflating: char-embeddings-master/output/model.hdf5
  inflating: char-embeddings-master/output/text_sample.txt
  inflating: char-embeddings-master/output/text_sample_1.txt
  inflating: char-embeddings-master/text_generator_keras.py
  inflating: char-embeddings-master/text_generator_keras_sample.py
```

```
embeddings_index_char = {}
with open('/content/char-embeddings-master/glove.840B.300d-char.txt',
'r') as f:
    for line in f:
      #print(line)
      line_split = line.strip().split(" ")
      #print(line_split)
      char = line_split[0]
      coefs = np.asarray(line_split[1:], dtype='float32')
      embeddings_index_char[char] = coefs
print('Found %s word vectors.' % len(embeddings_index_char))
```

```
Found 94 word vectors.
```

```
embeddings_index_char['$'].shape
```

```
(300,)
```

```
char_index = tokenizer_char.word_index
print('Found %s unique tokens.' % len(char_index))
```

```
Found 73 unique tokens.
```

```
embedding_dim_char = 300
```

```
embedding_matrix_char = np.zeros((len(char_index) + 1,
embedding_dim_char))
for char, i in char_index.items():
    embedding_vector = embeddings_index_char.get(char)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix_char[i] = embedding_vector[0]

# Open a file and use dump()
with open('embedding_matrix_char.pkl', 'wb') as file:
    # A new file will be created
    pickle.dump(embedding_matrix_char, file)

with open('/content/embedding_matrix_char.pkl', 'rb') as file:
    # Call load method to deserialze
    embedding_matrix_char = pickle.load(file)

from keras.layers import Embedding

embedding_layer = Embedding(len(char_index) + 1,
                            embedding_dim,
                            weights=[embedding_matrix_char],
                            input_length=max_sequence_len,
                            trainable=False)
```

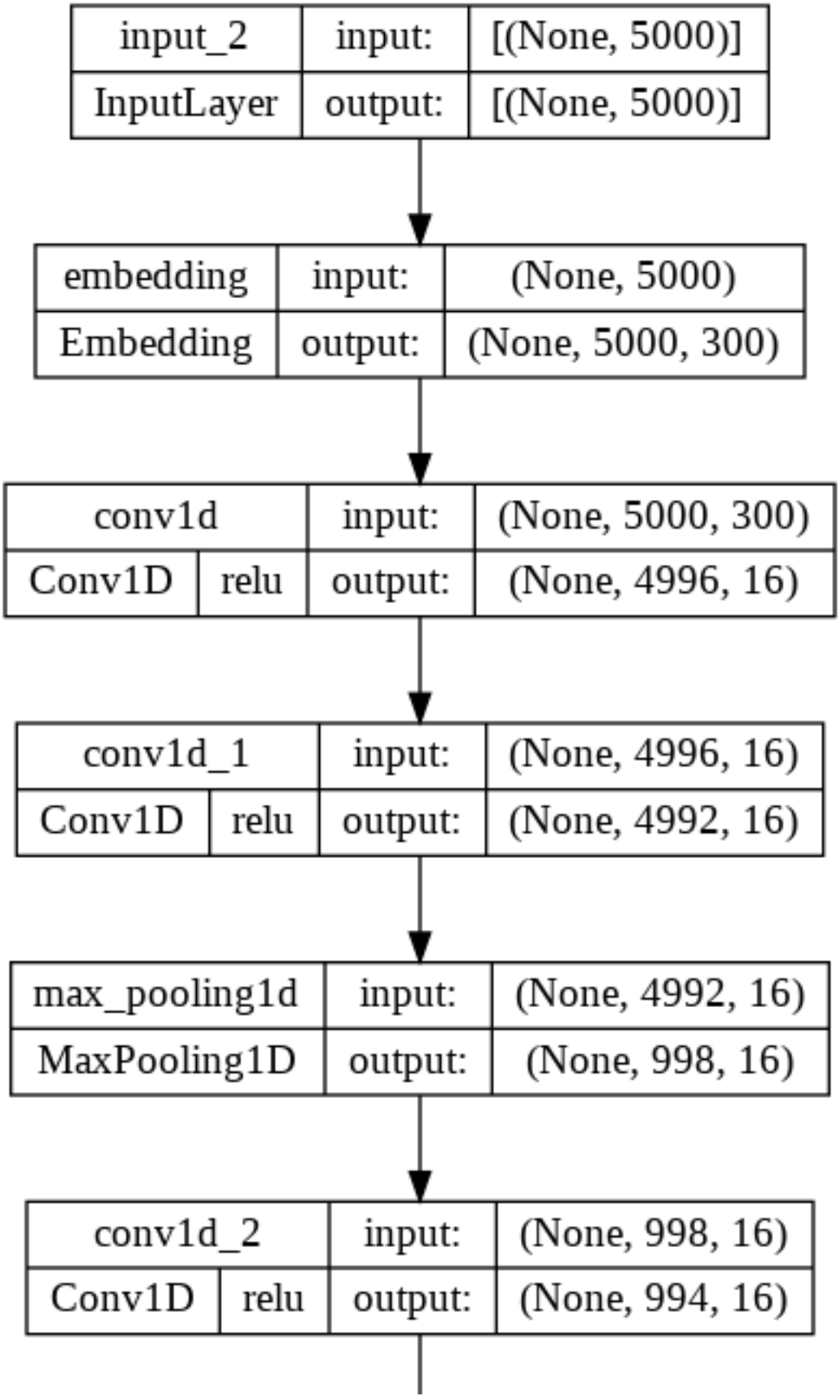
Lets Train Model_2

```
output_shape = 20

sequence_input = Input(shape=(max_char_len,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
layer1_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(embedded_sequences)
layer2_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(layer1_conv1)
maxpooling_1 = MaxPooling1D(5)(layer2_conv1)
layer3_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(maxpooling_1)
layer4_conv1 = Conv1D(16, 5,
activation='relu',kernel_initializer="he_normal")(layer3_conv1)
maxpooling_2 = MaxPooling1D(5)(layer4_conv1)
flatten_1 = Flatten()(maxpooling_2)
dropout_1 = Dropout(0.5)(flatten_1)
dense_1 = Dense(100,activation='relu',kernel_initializer="he_normal")
```

```
(dropout_1)
output = Dense(output_shape, activation='softmax')(dense_1)
model_char = Model(sequence_input, output)


model_char.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 5000)]            0

 embedding (Embedding)       (None, 5000, 300)         22200

 conv1d (Conv1D)             (None, 4996, 16)          24016

 conv1d_1 (Conv1D)           (None, 4992, 16)          1296

 max_pooling1d (MaxPooling1D  (None, 998, 16)          0
 )

 conv1d_2 (Conv1D)           (None, 994, 16)           1296

 conv1d_3 (Conv1D)           (None, 990, 16)           1296

 max_pooling1d_1 (MaxPooling  (None, 198, 16)          0
 1D)

 flatten (Flatten)           (None, 3168)              0

 dropout (Dropout)           (None, 3168)              0

 dense (Dense)               (None, 100)               316900

 dense_1 (Dense)             (None, 20)                2020

=================================================================
Total params: 369,024
Trainable params: 346,824
Non-trainable params: 22,200
_________________________________________________________________
```

```
plot_model(model_char,to_file = 'model_2.png',show_shapes =
True,show_layer_activations = True)
```


input_2 | input: | [(None, 5000)]
InputLayer | output: | [(None, 5000)]
embedding | input: | (None, 5000)
Embedding | output: | (None, 5000, 300)
conv1d | input: | (None, 5000, 300)
Conv1D | relu | output: | (None, 4996, 16)
conv1d_1 | input: | (None, 4996, 16)
Conv1D | relu | output: | (None, 4992, 16)
max_pooling1d | input: | (None, 4992, 16)
MaxPooling1D | output: | (None, 998, 16)
conv1d_2 | input: | (None, 998, 16)
Conv1D | relu | output: | (None, 994, 16)

```
model_char.compile(loss='categorical_crossentropy',optimizer='adam',me
trics=['accuracy',tfa.metrics.F1Score(num_classes=20,average='micro',t
hreshold=0.5)])


filepath="model_2/weights-{epoch:02d}-{val_accuracy:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath,
monitor='val_accuracy',  verbose=1, save_best_only=True, mode='auto')

model_char.fit(x_train, y_train, validation_data=(x_val,
y_val),epochs=20,
batch_size=128,callbacks=[earlystop,checkpoint,tensorboard_callback])

Epoch 1/20
110/111 [============================>.] - ETA: 0s - loss: 2.7225 -
accuracy: 0.1550 - f1_score: 0.0295
Epoch 1: val_accuracy improved from 0.08668 to 0.08859, saving model
to model_2/weights-01-0.0886.hdf5
111/111 [==============================] - 11s 100ms/step - loss:
2.7226 - accuracy: 0.1551 - f1_score: 0.0294 - val_loss: 2.9647 -
val_accuracy: 0.0886 - val_f1_score: 0.0029
Epoch 2/20
110/111 [============================>.] - ETA: 0s - loss: 2.6940 -
accuracy: 0.1658 - f1_score: 0.0409
Epoch 2: val_accuracy improved from 0.08859 to 0.09475, saving model
to model_2/weights-02-0.0948.hdf5
111/111 [==============================] - 11s 102ms/step - loss:
2.6942 - accuracy: 0.1658 - f1_score: 0.0408 - val_loss: 2.9672 -
val_accuracy: 0.0948 - val_f1_score: 0.0046
Epoch 3/20
110/111 [============================>.] - ETA: 0s - loss: 2.6418 -
accuracy: 0.1793 - f1_score: 0.0575
Epoch 3: val_accuracy did not improve from 0.09475
111/111 [==============================] - 10s 90ms/step - loss:
2.6422 - accuracy: 0.1793 - f1_score: 0.0573 - val_loss: 2.9814 -
val_accuracy: 0.0877 - val_f1_score: 0.0038
Epoch 4/20
110/111 [============================>.] - ETA: 0s - loss: 2.6105 -
accuracy: 0.1876 - f1_score: 0.0645
Epoch 4: val_accuracy did not improve from 0.09475
111/111 [==============================] - 11s 100ms/step - loss:
2.6105 - accuracy: 0.1876 - f1_score: 0.0645 - val_loss: 3.0094 -
val_accuracy: 0.0903 - val_f1_score: 0.0071
Epoch 4: early stopping

<keras.callbacks.History at 0x7fe89194dac0>

values = model_char.evaluate(x_val, y_val,verbose =0)
print('Loss of the model : ',values[0])
print('accuracy of the model : ',values[1])
print('f1-score of the model : ',values[2])
```

```
Loss of the model :  3.0094172954559326
accuracy of the model :  0.09029105305671692
f1-score of the model :  0.007067137863487005
```

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

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