Statistical NLP Part-1 - Text Classification

- **DOMAIN: Digital content management**

- **CONTEXT:**

Classification is probably the most popular task that you would deal with in real life. Text in the form of blogs, posts, articles, etc. is written every second. It is a challenge to predict the information about the writer without knowing about him/her. We are going to create a classifier that predicts multiple features of the author of a given text. We have designed it as a Multi label classification problem

- **DATA DESCRIPTION:**

Over 600,000 posts from more than 19 thousand bloggers The Blog Authorship Corpus consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. The corpus incorporates a total of 681,288 posts and over 140 million words - or approximately 35 posts and 7250 words per person. Each blog is presented as a separate file, the name of which indicates a blogger id# and the blogger's self-provided gender, age, industry, and astrological sign. (All are labelled for gender and age but for many, industry and/or sign is marked as unknown.) [Source]

All bloggers included in the corpus fall into one of three age groups:

- 8240 "10s" blogs (ages 13-17)
- 8086 "20s" blogs(ages 23-27) and
- 2994 "30s" blogs (ages 33-47)

For each age group, there is an equal number of male and female bloggers. Each blog in the corpus includes at least 200 occurrences of common English words. All formatting has been stripped with two exceptions. Individual posts within a single blogger are separated by the date of the following post and links within a post are denoted by the label url link.

- **PROJECT OBJECTIVE:**

The need is to build a NLP classifier which can use input text parameters to determine the label/s of the blog.

In [1]:

```
# !conda install dtale -c conda-forge
# if you want to also use "Export to PNG" for charts
# !conda install -c plotly python-kaleido
```

In [2]:

```
# imports

import os
import dtale
import random
import warnings
from time import time
from math import floor
from pathlib import Path
import pandas as pd, numpy as np
from pprint import pprint
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from collections import defaultdict
import tensorflow as tf
tqdm.pandas()
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [3]:

```
# reproducibility
seed = 7
random.seed(seed)
```

- **Import the data.**

In [4]:

```
blog_df = pd.read_csv('./data/blogtext.csv')
d = dtale.show(blog_df)
d
```

```
2021-07-25 18:25:51,567 - INFO     - Note: NumExpr detected 16 cores but "NUMEXPR_MAX_THR
EADS" not set, so enforcing safe limit of 8.
2021-07-25 18:25:51,568 - INFO     - NumExpr defaulting to 8 threads.
```

Out[4]:

In [5]:

```
# 681284 blog texts
blog_df.shape
```

Out[5]:

```
(681284, 7)
```

In [6]:

```
blog_df.isna().sum()   # no null values, dataset is good in terms of data completeness
```

Out[6]:

```
id        0
gender    0
age       0
topic     0
sign      0
```

```
date      0
text      0
dtype: int64
```

In [7]:

```
blog_df.info()  # topic, sing columns to be converted to categorical and date can be spli
t into day, month, year if required for analysis
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 681284 entries, 0 to 681283
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      681284 non-null  int64
 1   gender  681284 non-null  object
 2   age     681284 non-null  int64
 3   topic   681284 non-null  object
 4   sign    681284 non-null  object
 5   date    681284 non-null  object
 6   text    681284 non-null  object
dtypes: int64(2), object(5)
memory usage: 36.4+ MB
```

In [8]:

```
blog_df['id'] = pd.Categorical(blog_df.id)
blog_df['topic'] = pd.Categorical(blog_df.topic)
blog_df['sign'] = pd.Categorical(blog_df.sign)
blog_df['text'] = blog_df['text'].fillna('').apply(str)
```

In [9]:

```
blog_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 681284 entries, 0 to 681283
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      681284 non-null  category
 1   gender  681284 non-null  object
 2   age     681284 non-null  int64
 3   topic   681284 non-null  category
 4   sign    681284 non-null  category
 5   date    681284 non-null  object
 6   text    681284 non-null  object
dtypes: category(3), int64(1), object(3)
memory usage: 24.0+ MB
```

In [10]:

```
# Imbalanced classes!
blog_df.topic.value_counts()
```

Out[10]:

```
indUnk                     251015
Student                    153903
Technology                  42055
Arts                        32449
Education                   29633
Communications-Media        20140
Internet                    16006
Non-Profit                  14700
Engineering                 11653
Law                          9040
Publishing                   7753
Science                      7269
Government                   6907
Consulting                   5862
Religion                     5235
Fashion                      4851
```

```
Fashion                      4851
Marketing                    4769
Advertising                  4676
BusinessServices             4500
Banking                      4049
Chemicals                    3928
Telecommunications           3891
Accounting                   3832
Military                     3128
Museums-Libraries            3096
Sports-Recreation            3038
HumanResources               3010
RealEstate                   2870
Transportation               2326
Manufacturing                2272
Biotech                      2234
Tourism                      1942
LawEnforcement-Security      1878
Architecture                 1638
InvestmentBanking            1292
Automotive                   1244
Agriculture                  1235
Construction                 1093
Environment                   592
Maritime                      280
Name: topic, dtype: int64
```

In [11]:

```
blog_df.sign.value_counts()
```

Out[11]:

```
Cancer         65048
Aries          64979
Taurus         62561
Libra          62363
Virgo          60399
Scorpio        57161
Pisces         54053
Leo            53811
Gemini         51985
Sagittarius    50036
Aquarius       49687
Capricorn      49201
Name: sign, dtype: int64
```

In [12]:

```
# !pip install langdetect
```

In [13]:

```
# Looking for languages
from langdetect import detect_langs, LangDetectException
from collections import defaultdict

languages = []
lang_samples = defaultdict(list)
# pick random texts out of the dataset as langauge detection on every text is not feasibl
e
samples = np.random.choice(len(blog_df.text), size=5000, replace=False)
# Loop over the rows of the dataset and append
for row in tqdm(samples):
    try:
        text = blog_df.text[row]
        lang = detect_langs(text)
        clean_lang = str(lang).split(':')[0][1:]
        lang_samples[clean_lang].append(text)
        languages.append(clean_lang)
    except LangDetectException:
        pass
```

```python
print("Unique languages in the reviews: "
     f"{np.unique(languages)}")
```

```
100%|██████████████████████████████████████████████████| 5000/
5000 [00:28<00:00, 174.63it/s]
```

```
Unique languages in the reviews: ['af' 'cy' 'da' 'de' 'en' 'es' 'et' 'fi' 'fr' 'hu' 'id'
'it' 'ko' 'lt'
 'nl' 'no' 'pl' 'pt' 'ro' 'ru' 'sk' 'sl' 'so' 'sv' 'tl' 'tr' 'vi' 'zh-cn']
```

In [14]:

```python
for i in lang_samples:
    print(i)
    print(random.sample(lang_samples[i], 1))
    print('')
```

```
en
["             NOTE: You may want to edit your posts so they don't include the HeppSTAW na
me, folks... would you really want (P)ank (Fr)ate finding out how much you weight? And si
nce blogs ARE included in most search engines....             "]

de
['     urlLink    Gerbera  urlLink    ']

lt
['         urlLink       urlLink            ']

nl
['         urlLink    costume jewelry  urlLink        ']

tl
['            Hintayin ko na tanong ni Debb para maging organized ang game.         ']

no
["         urlLink jonbetter.com's breakout     "]

af
['         urlLink    Room 309, The Premier Room  urlLink            ']

sv
['             urlLink    Tavallodi Digar - Shoja&#39;eddin Shafa  urlLink
']

da
['        *deleted*    ']

et
['        aaaaaaaaaaaaaw, kate finks mitch is one in a million. shame she still likes u-kn
ow-who.         ']

hu
['         urlLink    Get A Job    ']

pl
['              My two babies.         ']

id
["                   leaving dundee tomorrow. yeaah!!! waitin' to see kak minn cos she'll b
e leaving dundee in august for good.. isk isk... my sis angkat. One of the reasons why i
chose dundee over aberdeen and leeds.   anyways.. the following has been copied from www.
utusan.com.my. thought i'd paste it on my blog for my own personal reading in future.. hah
ahahah. MY DAD SHOULD REALLY READ THIS!!! especially  the last bit!  Doktor pakar diberi
layanan kelas kedua kenaikan pangkat Saudara Pengarang, KENYATAAN Menteri Kesihatan di Pa
rlimen pada 7 Julai 2004 yang disiarkan oleh akhbar Utusan Malaysia adalah dirujuk. Belia
u mendakwa sehingga 31 Mei 2004, masih terdapat 875 jawatan pegawai perubatan pakar pelba
gai bidang yang belum diisi daripada sejumlah 2,397 jawatan pegawai perubatan pakar di Ke
menterian Kesihatan Malaysia.   Di sini, saya dengan sebulat suara menyokong kenyataan be
liau dan ia amat berpatutan dengan iklan yang dikeluarkan oleh Kementerian Kesihatan yang
disiarkan di laman web kementerian (untuk sesiapa yang berminat membacanya, www.moh.gov.m
y).   Namun begitu, janganlah rakyat Malaysia terkejut, jika semakin ramai pegawai peruba
```

y). Namun begitu, janganlah rakyat Malaysia terkejut, jika semakin ramai pegawai peruba
tan pakar yang akan meletakkan jawatan mereka disebabkan iklan ini. Untuk pengetahuan se
mua, gelaran doktor perubatan adalah bersamaan dengan pegawai perubatan di dalam skim per
khidmatan kerajaan. Untuk pengetahuan pembaca, sebelum Skim SSM (Sistem Saraan Malaysia)
diperkenalkan, kenaikan pangkat untuk jawatan pegawai perubatan pakar di dalam Skim SSB (
Sistem Saraan Baru) adalah dari gred U3 ke gred U2. Gaji untuk pegawai perubatan pakar kl
inikal gred U3 untuk skim SSB adalah di dalam lingkungan RM5,012.81 (iaitu selepas mendap
at hadiah dua kenaikan gaji dan bayaran insentif pakar apabila seseorang pegawai perubata
n lulus ijazah sarjana kepakaran dan diwartakan sebagai pakar). Apabila dinaikkan pangk
at ke gred U2 di dalam SSB, seorang pegawai perubatan pakar gred U2 akan menerima imbuhan
gaji di dalam lingkungan RM6,988.51. Kenaikan pangkat ke gred U2 ini akan dinikmati oleh
pegawai perubatan pakar tersebut setelah dua hingga tiga tahun diwartakan sebagai pakar k
linikal di gred U3. Ini merupakan pertambahan gaji sebanyak RM1,975.70, satu jumlah yan
g agak berpatutan. Bagaimanapun kami hanya nikmati setelah terpaksa bertungkus lumus un
tuk menamatkan pengajian di peringkat sarjana kepakaran yang mengambil masa selama empat
hingga lima tahun (tidak termasuk ijazah sarjana muda doktor perubatan, M.D, M.B.B.S, yan
g telah diambil dalam masa enam hingga tujuh tahun). Tetapi lain pula halnya di dalam SS
M yang diperkenalkan pada penghujung tahun 2002, jawatan kami sebagai pegawai perubatan p
akar klinikal gred U41 hanya dinaikkan ke pegawai perubatan pakar gred U44 (sepatutnya ja
watan setara yang perlu ditawarkan untuk kenaikan pangkat di dalam SSM untuk pegawai peru
batan pakar adalah pada gred U48 iaitu bersamaan gred U2 untuk skim SSB). Gaji yang kam
i nikmati untuk jawatan pegawai perubatan pakar klinikal gred U41 adalah di dalam lingkun
gan RM5,177.81 dan apabila kami dinaikkan pangkat ke gred U44 (selepas diwartakan dan sem
estinya perlu lulus peperiksaan Penilaian Tahap Kecekapan, SSM), gaji baru yang akan kami
nikmati hanyalah RM5,631.23. iaitu kenaikan sebanyak RM453.42. Bayangkanlah pertambahan
gaji sebanyak itu yang kami akan dapat di dalam Skim SSM setelah bertungkus lumus belajar
untuk mendapatkan ijazah kepakaran. Untuk pengetahuan pembaca, gaji pegawai perubatan p
akar gred U48 skim SSM adalah bermula daripada RM7,303.51. Suatu perbezaan yang amat keta
ra di dalam skim SSM yang diperkenalkan berbanding skim SSB. Pada pendapat saya, jika se
seorang hanyalah pegawai perubatan gred U41 yang tidak mempunyai ijazah kepakaran dan din
aikkan ke gred U44, itu merupakan satu rezeki. Tetapi yang saya tidak faham, kenapalah ke
naikan pangkat ke gred U44 ini diperkenalkan kepada pegawai perubatan yang mempunyai kepa
karan klinikal. Kenapa, pihak kementerian tidak mahu memperkenalkan kembali skim kenaika
n pangkat ke gred kanan kepada pegawai perubatan, seperti yang diamalkan sewaktu Jawatank
uasa Kabinet 1976, yang memberikan kenaikan pangkat kepada semua pegawai perubatan yang t
elah berkhidmat selama lima tahun. Gred U43/U44 boleh digunakan sebagai gred kanan. Man
akala Gred U47/U48 dan ke atas dijadikan gred kenaikan pangkat untuk pegawai perubatan pa
kar dan juga kepada pegawai perubatan yang cemerlang dan sebagainya. Sebagai kesimpulan
dari tulisan saya ini, 1. Pihak kementerian sememangnya menggalakkan pegawai perubatan p
akar rakyat Malaysia untuk berhijrah, kerana mereka boleh mendapatkan khidmat pegawai per
ubatan pakar dari luar. Yang menariknya di sini untuk khidmat pakar dari luar, kementeria
n menawarkan Gred permulaan U48 sehingga ke gred U54 untuk pegawai-pegawai perubatan paka
r dari luar ini, yang sesetengahnya tidak diketahui asal-usul ijazah kepakaran mereka amb
il. Bukanlah pegawai perubatan pakar ini tidak mahu berkhidmat di hospital-hospital keraj
aan, tetapi sebenarnya kami merasakan yang kami ini dianak-tirikan dan diberi layanan kel
as kedua oleh Kementerian Kesihatan. 2. Hospital-hospital kerajaan akan sentiasa diangga
p sebagai hospital kelas kedua oleh rakyat Malaysia. Ini kerana kebanyakan penghijrahan p
akar ini merupakan pakar-pakar muda yang bercita-cita tinggi untuk menjadi seorang pakar
khusus/sub-kepakaran. Dengan itu, hospital kerajaan akan tinggal dengan pakar-pakar per
unding yang akan bersara dan tiada kelebihan di bidang sub-kepakaran berbanding pegawai p
erubatan pakar yang masih muda. Jikapun mereka ini pergi melanjutkan pengajian di bidan
g sub-kepakaran, berapa lama sangat yang mereka dapat berkhidmat. Selepas itu mereka pu
n bersara dan menyertai pihak swasta dengan sub-kepakaran mereka. Inilah fenomena yang ki
ta alami sekarang ini di mana hospital-hospital swasta disanjung tinggi oleh rakyat, wala
upun terpaksa membayar kos rawatan yang begitu tinggi. Dari manakah datangnya pakar yan
g berkhidmat di hospital swasta ini, jika bukan daripada penghijrahan dari sektor awam.
3. Untuk adik-adik yang terlalu amat meminati bidang perubatan. Bidang perubatan bukanlah
bidang yang `glamour'. Ia memerlukan jiwa yang kental dan masa yang amat-amat panjang unt
uk mengharungi bidang ini. Janganlah mengharapkan imbuhan yang luar biasa dari bidang i
ni (kecualilah jika anda berazam untuk meninggalkan sektor awam selepas tamat belajar). T
eruskan niat jika anda mencintai bidang perubatan yang sememangnya memerlukan ketabahan m
ental dan fizikal. - DOKTOR PAKAR KERAJAAN, Batu Ferringhi, Pulau Pinang.
"]

so
['   o man ooo man   ']

sk
['   zoom   ']

cy
['   hey o.d.   ']

```
fr
["        That's just noise, Dave.          "]

es
["               la vida es una cosa curiosa... y ms cuando uno acaba de comer, justo ahori
ta todo se siente 'lento' y no dan ganas de nada... solo de pensar... probablemente por e
so varios filosofos eran 'rellenitos'... se la pasaban comiendo y gozando de ese curioso
estado de letargo que te crea la sensacin de estar saciado por una buena comida...  lo ni
co que falto fue el vino... un buen vino tinto junto con esa lasagna que nos acabamos de
hechar hubiera sido la onda...  y es que bien dicen 'despus de comer, ni una carta leer'.
.. he he he...  as que yo solo y mis pensamientos nos la pasamos muy bien... no entiendo
como todo mundo se puede poner a trabajar luego luego... ESO NO ESTA BIEN!!... debe uno d
e dejar descansar la comida...  por cierto, hablando de comida y otros pecados, mi cintur
n que no creo que heche mentiras, ya me dice que definitivamente mi talla esta cambiando.
.. por ms que la bscula quiera verse condescendiente creo que definitivamente estoy subie
ndo de peso... muy triste!!... mas que nada por mi espalda y todas esas cosas...  el lune
s comenzar una especie de dieta, que ms bien va a constar de eliminar de mi hbitos todos
esos gansitos y cocas que he consumido ultimaente... me va a costar trabajo dejarlos... p
ero va a ser necesario...          "]

fi
['             How did you put yours on?  Tellllllllllllllllllll meeeeeeeeeeee.  MOOOOOOOO
OOOOOOORTAAAAAAAAAAAAAAAAR!          ']

pt
['                 PARENTAL      ADVISORY     ODREAMS0COME0TRUEO CONTAINS EXPLICIT LYRICS
Username:     From  urlLink Go-Quiz.com           ']

it
['                    Hello               ']

tr
['         urlLink    Smiley happy Delu    ']

ru
['              ꚍ ̤ª    ɰꚍǵћcoȽҡeɵeeƤ  „ ·ęȽӼ̬ᶉζg         ']

ro
['              Straight people  urlLink are so cute ...         ']

ko
['                   最終 只要對得起自己 就夠了吧 ...  對於那些 睜眼說瞎話的混蛋 ........  即使死不
承認說過哪些話 作過哪些事  其實他們心裡明明 深深記得的  呼嚨 只是他們保護自己的手段  裝死 只是他們最後防
衛的武器  在裝死的瞬間 才能變態的得到一點點解脫  可憐的是他們 活在自以為的世界裡 永遠逃不出來  在對不起良
心的地獄裡 死過千遍萬遍                 ']

vi
['              Tic Toc          ']

zh-cn
['        [ 亲近则易生悔慢之心 ]  有时候，我们会对别人授予的小惠[ 感激不尽 ]， 欲对亲人，父母的一辈子恩
情[ 视而不见 ]！ 因为，亲人对我们来说，太亲近了，说以我们看不见他们的好、看不见他们的牺牲！我们把亲人幸苦
的照顾、爱护，视为[ 理所当然 ]！有时为了小争执，还有[憎恨]他们，而不知[ 心存感激 ]！ 西方有句话说，[ 上
帝又两个住处，一个是在天堂，另一个是在感恩者的心里 ！] 是的，常[ 心存感谢 ]的人是快乐的；一颗[ 知恩的心
]，也就是[ 快乐的心 ]，上帝将住在其中！ 且让我们学习——[ 多想想别人的好，忘记别人的不好 ]、[ 对所得恩典的
谢意，能与争取恩典时一样地热心 ]，则我们心中将更充满爱，也将如在天堂一样快乐。         ']

sl
['        I need some sake soon.....ZzzzZzzz         ']
```

- ***There are multiple languages and character sets in the dataset!***

- **Visualize**

In [15]:

```
blog_df.columns
```

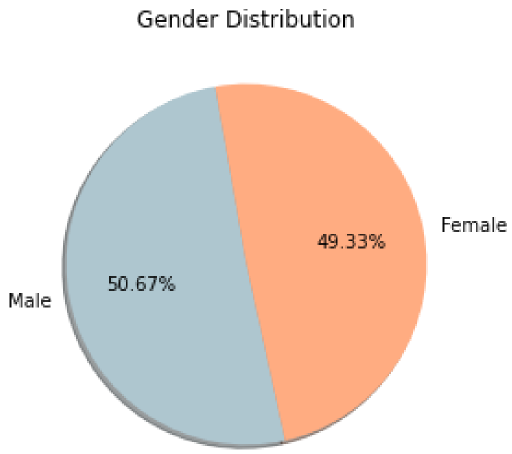
Out[15]:

```
Index(['id', 'gender', 'age', 'topic', 'sign', 'date', 'text'], dtype='object')
```

In [16]:

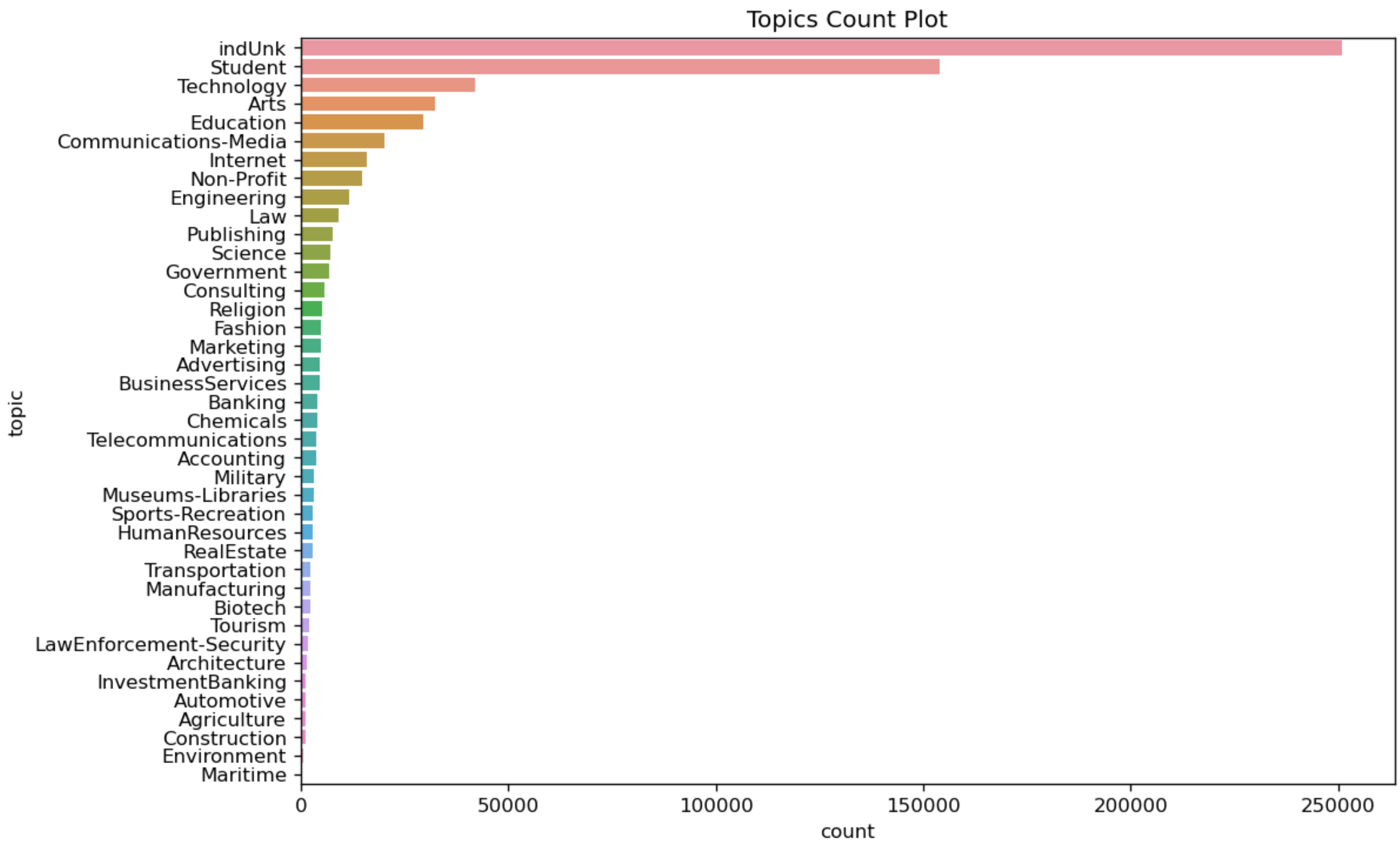
```
# plotting the gender distribution

males = len(blog_df[blog_df['gender'] == 'male'])
females = len(blog_df[blog_df['gender'] == 'female'])
plt.pie(x=[males, females], explode=(0, 0), labels=['Male', 'Female'], autopct='%1.2f%%'
,
        shadow=True, startangle=100, colors=['#aec6cf', '#ffac81'])
fig = plt.gcf()
fig.set_size_inches(4.5, 4.5)
plt.title('Gender Distribution')
plt.show()
```



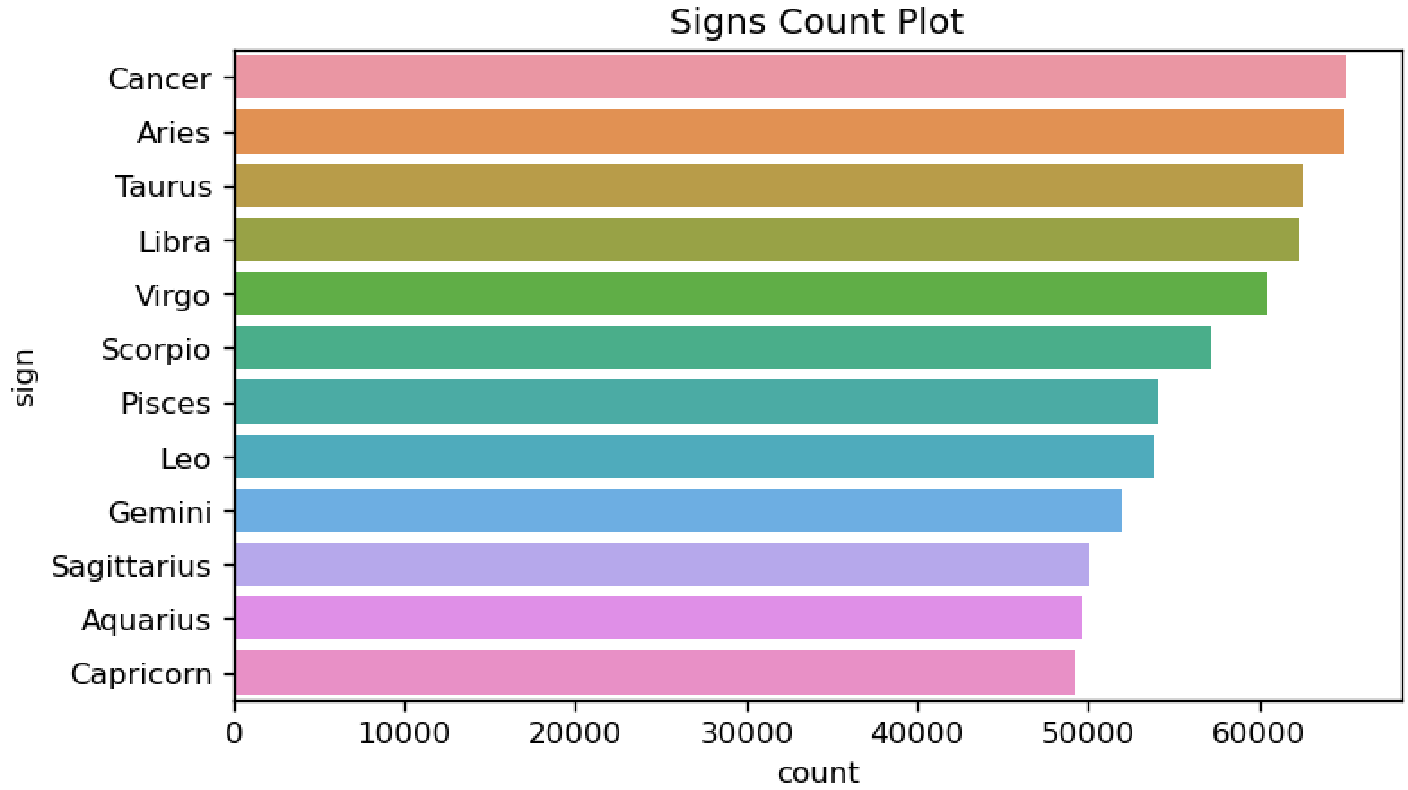
In [17]:

```
# plotting topic counts distribution
plt.figure(figsize=(10, 7), dpi=120)
sns.countplot(y='topic', data=blog_df, order=blog_df.topic.value_counts().index)
plt.title('Topics Count Plot')
plt.show()
```



In [18]:

```
# plotting sign counts distribution
plt.figure(figsize=(7, 4), dpi=120)
sns.countplot(y='sign', data=blog_df, order=blog_df.sign.value_counts().index)
plt.title('Signs Count Plot')
plt.show()
```



In [19]:

```
from collections import Counter

# top 50 most frequent words in text
top_N = 50

words = (blog_df.text.str.cat(sep=' ').split())
rslt = pd.DataFrame(Counter(words).most_common(top_N),
                    columns=['Word', 'Frequency']).set_index('Word')
```

In [20]:

```
rslt[:50].transpose()
```

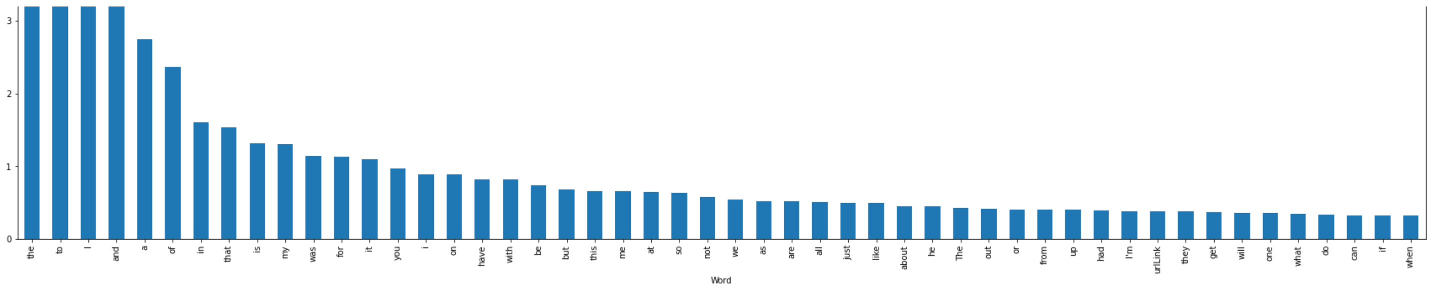
Out[20]:

Word	the	to	I	and	a	of	in	that	is	my	...	urlLink	they
Frequency	4785809	3842654	3546837	3303447	2750057	2362870	1600803	1540799	1318972	1300183	...	383203	375996

1 rows × 50 columns

In [21]:

```
rslt.plot.bar(rot=0, figsize=(30,8), width=0.55)
plt.xticks(rotation=90)
plt.show()
```



In [22]:

```
pprint(rslt.index.tolist(), compact=True)
```

```
['the', 'to', 'I', 'and', 'a', 'of', 'in', 'that', 'is', 'my', 'was', 'for',
 'it', 'you', 'i', 'on', 'have', 'with', 'be', 'but', 'this', 'me', 'at', 'so',
 'not', 'we', 'as', 'are', 'all', 'just', 'like', 'about', 'he', 'The', 'out',
 'or', 'from', 'up', 'had', "I'm", 'urlLink', 'they', 'get', 'will', 'one',
 'what', 'do', 'can', 'if', 'when']
```

- ***Many stopwords are occuring most frequently in the dataset, We will clean these in the preprocessing step***

In [23]:

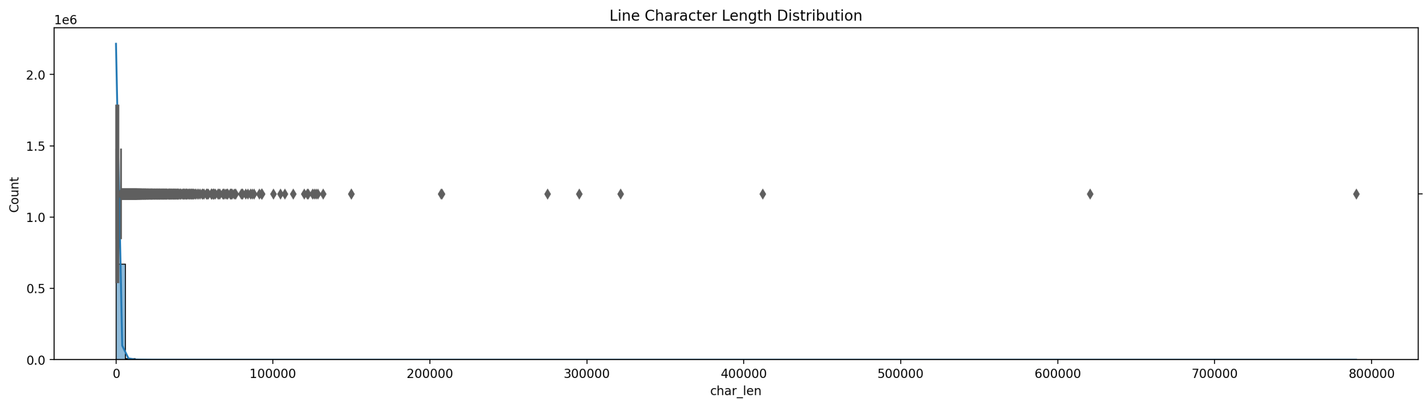
```
def get_text_len(row):
    row['char_len'] = len(row.text)
    row['word_len'] = len(row.text.split())
    return row

blog_df = blog_df.progress_apply(get_text_len, axis=1)
```

```
100%|██████████████████████████████████████████| 681284/681
284 [11:02<00:00, 1028.75it/s]
```

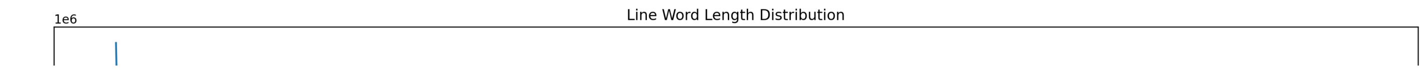
In [24]:

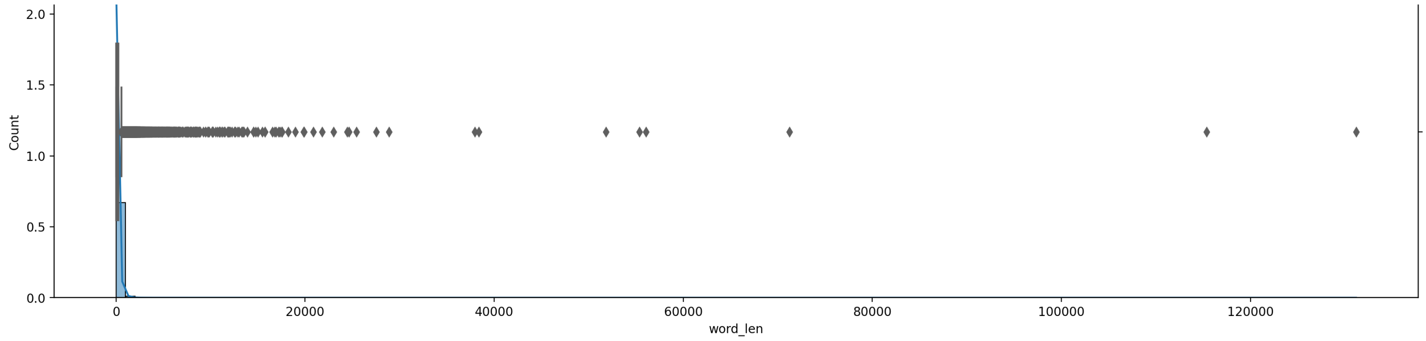
```
plt.figure(figsize=(20, 5), dpi=200)
ax = sns.histplot(x=blog_df.char_len, kde=True, bins=130)
ax2 = ax.twinx()
sns.boxplot(x=blog_df.char_len, ax=ax2, color='#EE6050')
ax2.set(ylim=(-.75, .75))
plt.title('Line Character Length Distribution')
plt.show()
```



In [25]:

```
plt.figure(figsize=(20, 5), dpi=200)
ax = sns.histplot(x=blog_df.word_len, kde=True, bins=130)
ax2 = ax.twinx()
sns.boxplot(x=blog_df.word_len, ax=ax2, color='#EE6050')
ax2.set(ylim=(-.75, .75))
plt.title('Line Word Length Distribution')
plt.show()
```





In [26]:

```
blog_df.describe()
```

Out[26]:

	id	age	char_len	word_len
count	6.812840e+05	681284.000000	681284.000000	681284.000000
mean	2.397802e+06	23.932326	1120.730698	200.786742
std	1.247723e+06	7.786009	2328.437003	415.160622
min	5.114000e+03	13.000000	4.000000	0.000000
25%	1.239610e+06	17.000000	230.000000	37.000000
50%	2.607577e+06	24.000000	637.000000	112.000000
75%	3.525660e+06	26.000000	1407.000000	255.000000
max	4.337650e+06	48.000000	790123.000000	131169.000000

- ***Most blog texts have between 0 and 500 with median at 112 with relatively few outliers raning till 1,31,169 words!***
  This imblance in text lengths will cause problem to classify the texts and we'll have to deal with it in preprocessing.

- ## Data Preprocessing

- ***Data cleansing by removing unwanted characters, spaces, stop words etc. Convert text to lowercase***

In [27]:

```
# !pip install contractions
# !pip install beautifulsoup4
```

In [28]:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\surya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\surya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[28]:

```
True
```

In [29]:

```
# utility functions for text preprocesing
```

```python
import re
import string
import unicodedata
import contractions
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer

CUSTOM = True

stemmer = SnowballStemmer('english')
if CUSTOM:
    stop_words = set(nltk.corpus.stopwords.words('english'))
    # custom stopwords added from the most frequent words which are generic
    # and might not relate to the sentiment of the review
    stop_words.update(['urllink'])
else:
    stop_words = set(nltk.corpus.stopwords.words('english'))

def replace_accented_chars(review_text):
    '''normalizes and replaces accented characters'''
    unaccented_text = unicodedata.normalize('NFKD', review_text).encode('ascii', 'ignore
').decode('utf-8', 'ignore')
    return unaccented_text



def strip_html_tags(review_text):
    '''strips html tags like <h4> ..etc'''
    soup = BeautifulSoup(review_text, "html.parser")
    [s.extract() for s in soup(['iframe', 'script'])]
    stripped_text = soup.get_text()
    stripped_text = re.sub(r'[\r|\n|\r\n]+', '\n', stripped_text)
    return stripped_text


def expand_contractions(review_text):
    review_text = contractions.fix(review_text)
    return review_text


def remove_special_characters(review_text):
    '''
    Remove special characters but preserve digits and excalamation marks
    as they indicate emotionally charged review '''
    review_text = re.sub(r"[^A-Za-z0-9!?\'\`]", " ", review_text)
    return review_text


def strip_stops(text, is_lower_case=False, stop_words=stop_words):
    '''strip stopwrds'''
    tokens = word_tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stop_words]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text


def snowball_stem(text, stemmer=stemmer):
    '''stemming using snowball stemmer'''
    words = text.split()
    stemmed_words = [stemmer.stem(word) for word in words]
    review_text = " ".join(stemmed_words)
    return review_text
```

In [30]:

```python
def preprocess_text(text: str, lower=True, srip_punctuation=True) -> str:
```

```
    text = replace_accented_chars(text)
    text = strip_html_tags(text)
    text = expand_contractions(text)
    text = remove_special_characters(text)

    if lower:
        text = text.lower()
    text = strip_stops(text)
    text = snowball_stem(text)
    return str(text.strip())

def preprocess(row):
    text = row.text
    if isinstance(text, str):
        text = preprocess_text(text)
    else:
        text = np.nan
    row['cleaned_text'] = text
    return row
```

In [31]:

```
blog_df = blog_df.progress_apply(preprocess, axis=1)
```

```
100%|██████████████████████████████████████████████████| 681284/68
1284 [29:17<00:00, 387.60it/s]
```

In [32]:

```
# all texts cleaned successfully
blog_df.isna().any()
```

Out[32]:

```
id              False
gender          False
age             False
topic           False
sign            False
date            False
text            False
char_len        False
word_len        False
cleaned_text    False
dtype: bool
```

In [33]:

```
blog_df.sample(20)
```

Out[33]:

	id	gender	age	topic	sign	date	text	char_len	word_len	clea
114792	3543301	female	24	Non-Profit	Gemini	03,August,2004	urlLink for real . i need to go h...	155	27	rea lun ma
599440	3535546	male	14	indUnk	Sagittarius	10,June,2004	....	634	80	so r r
241859	529513	male	33	Internet	Taurus	16,November,2003	This was the 4th or 5th time in my ...	122	17	4 wv m
321892	963380	male	24	Student	Cancer	20,January,2003	Good morning sunshine, as it's put in t...	1001	188	g s pa driv
							I must say, Barack			

	id	gender	age	topic	sign	date	text	char_len	word_len	clea
551120	1125546	male	27	Law	Aries	26,July,2004	I must say, Barack Obama was impressive...	318	48	
30505	3687738	male	47	indUnk	Aries	12,agosto,2004	¿Acaso, el escenario descrito en el D...	648	103	des
552488	3056329	male	17	Government	Libra	15,May,2004	hey!! I havent blogged in awhile!...	704	125	h a
380569	1417798	female	35	indUnk	Scorpio	15,September,2003	Hottie Alert! Greg K. just g...	698	119	g
61256	2871824	male	24	Engineering	Leo	04,May,2004	Last 3 days quite happy lor because...	970	190	qu fa
150862	2427534	male	25	indUnk	Leo	02,July,2004	Where's my poker? Just finishe...	1143	176	pok w
492401	3736084	male	25	Non-Profit	Leo	25,June,2004	Re: Laying to rest the idea of a pre-trib '...	1507	294	la pre '
127139	3541168	male	34	Government	Gemini	06,July,2004	and now has it's own blog at Zoe...	351	65	p
32151	1538911	female	35	indUnk	Libra	04,June,2004	urlLink All the mercy is right here...	228	32	d
242087	2790498	female	23	indUnk	Aries	12,February,2004	I am moderately hung over. And ...	2044	381	m rain fri
616582	4296124	male	17	indUnk	Gemini	21,August,2004	urlLink Look at my boredom tonight!...	77	8	look to
110967	3798616	female	26	indUnk	Pisces	29,June,2004	hey , just seeing if this works so i k...	855	176	hey
586904	944569	female	14	Student	Leo	09,February,2004	Eeeexcellent...more Prince of Tenni...	890	140	epi m
624468	3653978	female	16	indUnk	Virgo	01,July,2004	I dunno y... whenever Im bullied by...	3589	677	wh o re
388427	3168970	female	25	HumanResources	Cancer	19,July,2004	been slacking a bit the last few days...	4303	837	sl
456911	798653	male	26	indUnk	Aquarius	09,July,2004	You suck.	20	2	

In [34]:

```
top_N = 50

# top 50 most frequent words in cleaned text
words = (blog_df.cleaned_text.str.cat(sep=' ').split())
rslt = pd.DataFrame(Counter(words).most_common(top_N),
                    columns=['Word', 'Frequency']).set_index('Word')
```

In [35]:

```
rslt[:50].transpose()
```

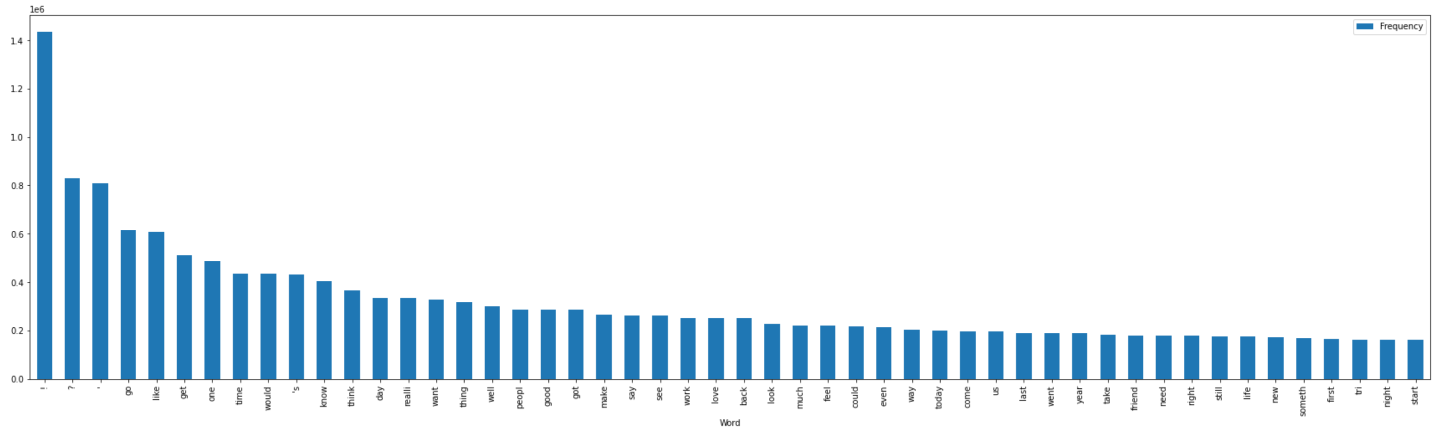
Out[35]:

Word	!	?	'	go	like	get	one	time	would	's	...	need	right	still
Frequency	1433386	827744	807814	615740	606031	511351	487906	435766	433882	431960	...	176987	176689	175512

1 rows × 50 columns

In [36]:

```
rslt.plot.bar(rot=0, figsize=(30,8), width=0.55)
plt.xticks(rotation=90)
plt.show()
```



In [37]:

```
pprint(rslt.index.tolist(), compact=True)
```

```
['!', '?', "'", 'go', 'like', 'get', 'one', 'time', 'would', "'s", 'know',
 'think', 'day', 'realli', 'want', 'thing', 'well', 'peopl', 'good', 'got',
 'make', 'say', 'see', 'work', 'love', 'back', 'look', 'much', 'feel', 'could',
 'even', 'way', 'today', 'come', 'us', 'last', 'went', 'year', 'take', 'friend',
 'need', 'right', 'still', 'life', 'new', 'someth', 'first', 'tri', 'night',
 'start']
```

- ***Target/label merger and transformation***

In [80]:

```
# merge all labels together as we want to do multi-label classification
blog_df['labels'] = blog_df[['gender','age','topic','sign']].values.tolist()
```

In [81]:

```
dataset = blog_df.drop(columns = ['id','gender','age','topic','sign','date', 'text'])
```

- ***Split dataset into train-test cuts***

In [82]:

```
from sklearn.model_selection import train_test_split
```

```
# Train-Test split of 80-20
X_train, X_test, y_train, y_test = train_test_split(dataset['cleaned_text'], dataset['la
bels'], test_size=0.20)
```

In [83]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[83]:

```
((545027,), (136257,), (545027,), (136257,))
```

- ***Vectorisation***

In [113]:

```
NGRAM_RANGE = (1, 2)
TOP_K = 30000
TOKEN_MODE = 'word'
MIN_DOC_FREQ = 2

kwargs = {
    'ngram_range' : NGRAM_RANGE,
    'dtype' : 'int32',
    'strip_accents' : 'unicode',
    'decode_error' : 'replace',
    'analyzer' : TOKEN_MODE,
    'min_df' : MIN_DOC_FREQ,
    'max_features': TOP_K
}
```

In [114]:

```
# vectorize the texts to get features
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectKBest, f_classif

vectorizer = TfidfVectorizer(**kwargs)

X_train_vec = vectorizer.fit_transform(X_train)
```

In [115]:

```
X_train_vec.shape
```

Out[115]:

```
(545027, 30000)
```

In [116]:

```
# do not fit on X_test to avoid data leakakge, only transform
X_test_vec = vectorizer.transform(X_test)
```

In [117]:

```
X_train_vec[0].shape
```

Out[117]:

```
(1, 30000)
```

In [118]:

```
X_test_vec[0].shape
```

Out[118]:

```
(1, 30000)
```

- ***Transform labels***

In [119]:

```
y_train[0]
```

Out[119]:

```
['female', '23', 'Advertising', 'Taurus']
```

In [120]:

```
# use MultiLabelBinarizer to transform labels in a binary form so that the prediction wil
l be a mask of 0s and 1s
from sklearn.preprocessing import MultiLabelBinarizer

binarizer = MultiLabelBinarizer()

# convert all labels to str
y_train = [[str(i) for i in j] for j in y_train]
y_test = [[str(i) for i in j] for j in y_test]

y_train_labels = binarizer.fit_transform(y_train)

# only transform test data to avoid data leakage
y_test_labels = binarizer.transform(y_test)

y_train_labels.shape, y_test_labels.shape
```

Out[120]:

```
((545027, 80), (136257, 80))
```

In [121]:

```
# converted to one hot vectors, each category here is a combination of labels from possib
le combinations of labels
y_train_labels[0]
```

Out[121]:

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

- **Train Text Classifiers**

In [122]:

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression


from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, clas
sification_report
```

- **Logistic Regression**

In [123]:

```
# wrapper for lr classfier to be able to predict among classes
lr = OneVsRestClassifier(LogisticRegression(solver='lbfgs'), n_jobs=8)
```

In [124]:

```
lr.fit(X_train_vec, y_train_labels)
```

Out[124]:

```
OneVsRestClassifier(estimator=LogisticRegression(), n_jobs=8)
```

In [127]:

```python
pred = lr.predict(X_test_vec)
```

In [128]:

```python
# classification report
print(classification_report(y_test_labels, pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.04      0.07      2696
           1       0.66      0.06      0.11      5397
           2       0.59      0.06      0.12      8288
           3       0.68      0.11      0.19     14565
           4       0.66      0.11      0.18     16374
           5       0.69      0.03      0.05     14517
           6       0.81      0.04      0.07     16103
           7       0.69      0.02      0.05     13237
           8       0.76      0.03      0.05     11033
           9       0.74      0.03      0.05      9232
          10       0.98      0.03      0.05      3456
          11       0.90      0.11      0.20      4373
          12       0.75      0.05      0.10      3481
          13       0.87      0.07      0.12      2930
          14       0.98      0.07      0.13      1835
          15       0.91      0.03      0.06      1472
          16       0.92      0.01      0.02      1072
          17       0.98      0.22      0.36      1000
          18       1.00      0.00      0.01       741
          19       0.58      0.02      0.04       596
          20       0.93      0.03      0.06       831
          21       1.00      0.01      0.02       390
          22       0.63      0.02      0.04       914
          23       0.42      0.06      0.11       527
          24       0.88      0.10      0.18       455
          25       0.72      0.06      0.11       742
          26       0.44      0.04      0.07       811
          27       0.83      0.01      0.01       931
          28       0.00      0.00      0.00       249
          29       0.84      0.03      0.06      9879
          30       1.00      0.00      0.01       347
          31       0.76      0.04      0.07     13024
          32       0.82      0.03      0.06      6467
          33       0.00      0.00      0.00       246
          34       1.00      0.00      0.01       830
          35       1.00      0.03      0.05       437
          36       0.88      0.05      0.09       905
          37       0.86      0.06      0.10     12959
          38       0.56      0.01      0.01      9755
          39       0.00      0.00      0.00       751
          40       0.71      0.01      0.02      3964
          41       1.00      0.03      0.06       222
          42       0.00      0.00      0.00      1195
          43       0.81      0.04      0.07      5970
          44       0.87      0.03      0.05      2325
          45       0.00      0.00      0.00       119
          46       0.97      0.16      0.27       966
          47       0.77      0.02      0.04     10503
          48       0.95      0.03      0.05      1422
          49       0.50      0.00      0.00       627
          50       0.81      0.02      0.05      3302
          51       0.00      0.00      0.00       240
          52       0.62      0.02      0.04      1782
          53       1.00      0.00      0.01       385
          54       0.90      0.02      0.04     10820
          55       0.66      0.02      0.03     12313
          56       1.00      0.00      0.01       442
```

```
          56       1.00      0.00      0.01       442
          57       0.00      0.00      0.00        48
          58       0.64      0.01      0.01       979
          59       1.00      0.00      0.01       601
          60       0.56      0.02      0.04       600
          61       0.55      0.03      0.05      3007
          62       0.79      0.04      0.08     10768
          63       0.92      0.07      0.14      1546
          64       0.29      0.01      0.01       586
          65       0.40      0.02      0.03      1084
          66       0.69      0.02      0.03     10187
          67       0.50      0.00      0.01      1406
          68       0.76      0.02      0.04     11389
          69       0.83      0.01      0.02       583
          70       0.59      0.20      0.30     30722
          71       0.87      0.03      0.05     12484
          72       0.64      0.05      0.09      8401
          73       0.87      0.03      0.05       800
          74       1.00      0.02      0.04       377
          75       1.00      0.13      0.23       457
          76       0.83      0.03      0.05     12176
          77       0.70      0.68      0.69     67216
          78       0.55      0.21      0.30     50125
          79       0.70      0.72      0.71     69041

   micro avg       0.68      0.23      0.34    545028
   macro avg       0.71      0.06      0.09    545028
weighted avg       0.71      0.23      0.26    545028
 samples avg       0.68      0.23      0.33    545028
```

- **Metrics:**

Metrics for the model	Precision	Recall
Micro Average	0.68	0.23
Macro Average	0.71	0.06
Weighted Average	0.71	0.23

- **Micro-averaged Precision is calculated as precision of Total values:**
- **all samples equally contribute to the final averaged metric**
- **Macro-averaged Precision is calculated as an average of Precisions of all classes:**
- **all classes equally contribute to the final averaged metric**
- **Weighted-averaged Precision is also calculated based on Precision per class but takes into account the number of samples of each class in the data:**
- **each classes's contribution to the average is weighted by its size**

Which metric is relevant depends on If there is a class-imbalanced dataset? Is one class more important to get right than others? If you have an under-represented class which is important to your problem, macro-averaging may better, as it will highlight the performance of a model on all classes equally. On the other hand, if the assumption that all classes are equally important is not true, macro-averaging will over-emphasize the low performance on an infrequent class. Micro-averaging may be preferred in multilabel settings, including multiclass classification where a majority class is to be ignored.

- **So, our model acheived a micro-avg f1-score of 0.34, macro-avg f1-score of 0.09. If we care about the minoority classes as well in the final classification, our model isn't up to the mark and might be further improved by cleaning the texts with multiple languages and performing oversampling/undersampling or SMOTE to deal with the class imbalance. Further, we can use BERT or other Attention-based classifiers which are much better at text classification tasks.**

- **Print the true vs predicted labels for any 5 entries from the dataset**

In [140]:

```
from random import sample

# Getting real labels from transformed predicted labels
pred_classes = binarizer.inverse_transform(pred)
```

```python
#Picking 5 random records from y_test and comparing actual labels vs predicted labels for those 5 records
for i in sample(range(len(pred)),  5):
    print(i)
    print("Actual labels: ", y_test[i])
    print("Predicted labels:", pred_classes[i])
    print()
```

```
78708
Actual labels:  ['female', '14', 'Student', 'Virgo']
Predicted labels: ('female',)

129791
Actual labels:  ['male', '27', 'Technology', 'Libra']
Predicted labels: ('male',)

90040
Actual labels:  ['male', '24', 'Technology', 'Libra']
Predicted labels: ('male',)

117659
Actual labels:  ['male', '24', 'Education', 'Leo']
Predicted labels: ('female', 'indUnk')

75481
Actual labels:  ['female', '33', 'Communications-Media', 'Scorpio']
Predicted labels: ('male',)
```