Sentiment Analysis_POA_Ngram

@Author : Woojin Park, Nidhi Bhaskar

@Email : woojinpa@andrew.cmu.edu , nidhibha@andrew.cmu.edu

@Status : In-Progress

In [1]:

```
### Import Relevant Libraries
import os
import pandas as pd
import numpy as np
import collections
import datetime as dt
import requests
import json
import re
import time

import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
from scipy.stats import norm

import string
import re
import nltk
from nltk.collocations import *
from nltk.util import ngrams
from nltk import pos_tag,word_tokenize
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer,PorterStemmer
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

In [ ]:

```
### Build a get_date function to convert date format
#### Build a data_creation function to read json data into pandas dataframe

def get_date(created):
    return dt.datetime.fromtimestamp(created)

def data_creation(subreddit) :
    with open('submissions_'+subreddit+'.json') as f:
        data = json.loads("[" +
            f.read().replace("}\n{", "},\n{") +
        "]")
    data =pd.DataFrame(data)
    reddit_data = data[['author','over_18','title','selftext','num_comments', 's
core', 'full_link','created_utc']]
    reddit_data = reddit_data.dropna()
    _timestamp = reddit_data["created_utc"].apply(get_date)
    reddit_data = reddit_data.assign(timestamp = _timestamp)
    reddit_data['over_18'] = reddit_data['over_18'].astype('str')
    reddit_data['subreddit']= subreddit
    # Build column have title + selftext
    reddit_data['title_with_selftext']= reddit_data['title'] +" " + reddit_data[
'selftext']

    # Do one more extra cleaning : keep updating this part
    reddit_data=reddit_data[~reddit_data['title_with_selftext'].isin([ '[removed
]', '[deleted]',''])]
    subreddit = reddit_data

    return subreddit

def empty_words_clean(text):
    text = text.replace('[removed]','')
    text= text.replace('[deleted]','')
    text= text.replace('\n','')
    return (text)
```

In [3]:

```
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
```

Classes : 25000 - 50000

Reddit_SuicideWatch_20200122_20200522 : 47701 Reddit_Depressed_20100522_20200522 : 26639
Reddit_happy_20170522_20200522 : 41420 Reddit_selfimprovement_20160522_20200522 : 44441

In [4]:

```
### Dataframing 4 subreddit Datasets
SuicideWatch_df = data_creation('SuicideWatch')
depressed_df = data_creation('depressed')
happy_df = data_creation('happy')
selfimprovement_df = data_creation('selfimprovement')

### Concat all 4 dataframes into one merged file
all_subreddit_df = pd.concat([SuicideWatch_df,depressed_df,happy_df,selfimprovem
ent_df])
all_subreddit_df.head(2)
```

Out[4]:

	author	over_18	title	selftext	num_comments	score	
0	DespressoCafe	False	I don't know where to go or what to do. I can'...	Let's make it quick. I'm almost 20. I've been ...	5	1	https://www.reddit.com/r/
1	LifeisCrumbling	False	I'm having an existencial crisis	If I only helped people either as a defense me...	1	1	https://www.reddit.com/r/

In [5]:

```
SuicideWatch_df["title_with_selftext_cleaned"] = SuicideWatch_df["title_with_sel
ftext"].apply(lambda x: empty_words_clean(x))
depressed_df["title_with_selftext_cleaned"] = depressed_df["title_with_selftext"
].apply(lambda x: empty_words_clean(x))
happy_df["title_with_selftext_cleaned"] = happy_df["title_with_selftext"].apply(
lambda x: empty_words_clean(x))
selfimprovement_df["title_with_selftext_cleaned"] = selfimprovement_df["title_wi
th_selftext"].apply(lambda x: empty_words_clean(x))
```

Sentiment Analysis

1.Text Preprocessing, 2.Part-Of-Speech(POS), 3.Ngram

1.Text Preprocessing

Text Preprocessing by following pipeline :

Raw text => Tokeninze/lowercase => Remove stop words => Remove non-alphabetic characters => Remove Extra Punctuations => Lemmatization => Apply Custom Stop words dictionary

SuicideWatch

In [6]:

```
### Because of relatively huge dataset, we need to perform random sampling of 10 % for now
sampleSuicideWatch_list = SuicideWatch_df.sample(frac=0.1, replace=True, random_state=1)
```

In [7]:

```
SuicideWatch_list = sampleSuicideWatch_list['title_with_selftext_cleaned'].tolist()
```

In [8]:

```
#1.lowercase words as they are tokenized
```

In [9]:

```
SuicideWatch_list_lower = [tok.lower() for i in SuicideWatch_list for tok in nltk.word_tokenize(i)]
print(len(SuicideWatch_list_lower))
print(SuicideWatch_list_lower[:30])
```

```
838804
['i', 'love', 'you', 'guys', 'on', 'this', 'subreddit', "y'all", 'made', 'me', 'feel', 'less', 'alone', '...', 'and', 'take', 'care', '...', 'i', 'will', 'be', 'gone', 'foreveri', 'love', 'you', 'all', '...', ".y'all", 'get', 'the']
```

In [10]:

```
#2.remove stopwords
```

In [11]:

```
nltk_stopwords = set(stopwords.words('english'))
```

In [12]:

```
SuicideWatch_list_lower_stop = [x for x in SuicideWatch_list_lower if not x in n
ltk_stopwords]
print(len(SuicideWatch_list_lower_stop))
print(SuicideWatch_list_lower_stop[:30])
```

```
434496
['love', 'guys', 'subreddit', "y'all", 'made', 'feel', 'less', 'alon
e', '...', 'take', 'care', '...', 'gone', 'foreveri', 'love', '...',
".y'all", 'get', 'idea', 'much', 'life', 'depression', "sucksy'all",
'understand', 'like', 'heavy', 'soul', 'one', 'understood', 'issues'
]
```

In [13]:

```
##3. remove non-alphabetic characters (ex) punctations)
```

In [14]:

```
# function that takes a word and returns true if it consists only
#   of non-alphabetic characters
def alpha_filter(w):
    pattern = re.compile('^[^a-z]+$')
    if (pattern.match(w)):
        return True
    else:
        return False
```

In [15]:

```
#4. remove punctuations
SuicideWatch_list_lower_stop_pun = [y for y in SuicideWatch_list_lower_stop if n
ot alpha_filter(y)]
print(len(SuicideWatch_list_lower_stop_pun))
print(SuicideWatch_list_lower_stop_pun[:30])
```

```
343704
['love', 'guys', 'subreddit', "y'all", 'made', 'feel', 'less', 'alon
e', 'take', 'care', 'gone', 'foreveri', 'love', ".y'all", 'get', 'id
ea', 'much', 'life', 'depression', "sucksy'all", 'understand', 'like
', 'heavy', 'soul', 'one', 'understood', 'issues', 'sometimes', 'fam
ily', 'sucks']
```

In [16]:

```
#4-1. remove extra punctuations -attached to words
SuicideWatch_list_lower_stop_pun_extra = [''.join(x for x in par if x not in str
ing.punctuation) for par in SuicideWatch_list_lower_stop_pun]
print(SuicideWatch_list_lower_stop_pun_extra[:30])
```

```
['love', 'guys', 'subreddit', 'yall', 'made', 'feel', 'less', 'alone
', 'take', 'care', 'gone', 'foreveri', 'love', 'yall', 'get', 'idea'
, 'much', 'life', 'depression', 'sucksyall', 'understand', 'like', '
heavy', 'soul', 'one', 'understood', 'issues', 'sometimes', 'family'
, 'sucks']
```

In [17]:

```
#5. Lemmatization : to reduce inflectional forms to a common base form.
## It uses lexical knowledge bases to get the correct base forms of words.

porter = WordNetLemmatizer()
SuicideWatch_list_lower_stop_pun_extra_lemmatized = []
for a in SuicideWatch_list_lower_stop_pun_extra :
    SuicideWatch_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a)
)

print(len(SuicideWatch_list_lower_stop_pun_extra_lemmatized))
print(SuicideWatch_list_lower_stop_pun_extra_lemmatized[:30])
```

```
343704
['love', 'guy', 'subreddit', 'yall', 'made', 'feel', 'le', 'alone',
'take', 'care', 'gone', 'foreveri', 'love', 'yall', 'get', 'idea', '
much', 'life', 'depression', 'sucksyall', 'understand', 'like', 'hea
vy', 'soul', 'one', 'understood', 'issue', 'sometimes', 'family', 's
uck']
```

In [63]:

```
#6. Buid Custom stopwords function : Remove newly generated stop words
stop_words = stopwords.words('english')
print(len(stop_words))
cachedStopWords = set(stopwords.words("english"))

####Keep Updating custom stop words
cachedStopWords.update(('nt', 'wo', 're', 'im', 'yall','u','ca','ive', 'wan','na
','gon','nov','x200b','amp',\
                        'wwwyoutubecomwatch','http','vbjkbl5olvm8','lt', 'br', '
gt', 'amp','tsp','tbsp','nbsp', 'le','foreveri'))
print(len(cachedStopWords))
```

```
179
203
```

In [66]:

```
print(stopwords.words("english"))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 't
heir', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this
', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'w
ere', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', '
does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', '
because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', '
about', 'against', 'between', 'into', 'through', 'during', 'before',
'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'h
ere', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'coul
dn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn
't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi
ghtn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "
shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't
", 'won', "won't", 'wouldn', "wouldn't"]
```

In [65]:

```
print(cachedStopWords)
```

```
{'br', 'had', 'as', 'other', 'not', 'has', 'should', 'own', "it's",
'ive', "you're", 'having', 'so', 'over', 'down', 're', 'after', 'whe
n', 'here', 'again', 'haven', 'if', 'there', "shan't", 'where', 'its
elf', "don't", 'shan', 'my', 'that', 'yours', 'being', 'against', 'i
t', 'during', 'no', 've', 'why', 'its', 'did', 'with', 'na', 'won',
'ours', 'any', 'do', 'about', 'me', 'myself', 'tbsp', 'wo', 'forever
i', 'your', 'by', 'each', 'wan', 'because', 'have', 'doesn', 'theirs
', 'all', 'than', "she's", 'mustn', "hasn't", 'yall', 'amp', 'wwwyou
tubecomwatch', 'y', 'ourselves', 'up', 'lt', "haven't", 'aren', 'has
n', 'weren', 'himself', 'herself', 'those', 'was', 'how', "mightn't"
, 'and', 'while', 'more', 'but', 'on', 'am', 'now', "mustn't", 'shou
ldn', 'him', 'his', 'to', 'themselves', 'ma', 'only', "that'll", 'is
', 'o', "needn't", 'vbjkbl5olvm8', 'you', 'hers', 'at', 'doing', 'u'
, 'off', 'le', 'both', 'before', 'above', "didn't", "doesn't", 'your
self', "you've", 'between', 'im', 'the', 'then', 'this', 'can', 'fro
m', "won't", 'he', 'until', 'their', 'been', 'few', 'which', "couldn
't", 'or', 'd', 'ca', 'whom', 'further', 'nor', 'don', 'some', 'will
', 'are', 'once', 'an', 'yourselves', 'we', "weren't", "you'd", 'a',
'through', 'out', 'such', 'under', "should've", "wouldn't", 'i', 'nb
sp', "isn't", "hadn't", "shouldn't", 'gt', "aren't", 'were', 'wouldn
', 'most', 'mightn', 'her', 'in', 'these', 'just', 'll', 'gon', 'htt
p', 'who', "wasn't", 'they', 'same', 's', 'm', 'for', 'does', "you'l
l", 't', 'of', 'our', 'into', 'what', 'isn', 'be', 'too', 'very', 'd
idn', 'below', 'she', 'wasn', 'x200b', 'needn', 'them', 'nov', 'ain'
, 'couldn', 'tsp', 'nt', 'hadn'}
```

In [64]:

```
SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop = [x for x in SuicideWatc
h_list_lower_stop_pun_extra_lemmatized if not x in cachedStopWords]
print(len(SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop))
print(SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop[:30])
```

```
319489
['love', 'guy', 'subreddit', 'made', 'feel', 'alone', 'take', 'care'
, 'gone', 'love', 'get', 'idea', 'much', 'life', 'depression', 'suck
syall', 'understand', 'like', 'heavy', 'soul', 'one', 'understood',
'issue', 'sometimes', 'family', 'suck', 'sometimes', 'everyone', 'su
ck', 'towards']
```

Final Comparison before & after the Text Preprocessing Pipeline

In [71]:

```
print("Before: " , SuicideWatch_list_lower[:50],"\n")
print("After: " , SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop[:50])
```

```
Before:  ['i', 'love', 'you', 'guys', 'on', 'this', 'subreddit', "y'
all", 'made', 'me', 'feel', 'less', 'alone', '...', 'and', 'take', '
care', '...', 'i', 'will', 'be', 'gone', 'foreveri', 'love', 'you',
'all', '...', ".y'all", 'get', 'the', 'idea', 'of', 'how', 'much', '
life', 'with', 'depression', "sucksy'all", 'understand', 'what', 'is
', 'it', 'like', 'to', 'have', 'a', 'heavy', 'soul', 'while', 'no']

After:  ['love', 'guy', 'subreddit', 'made', 'feel', 'alone', 'take'
, 'care', 'gone', 'love', 'get', 'idea', 'much', 'life', 'depression
', 'sucksyall', 'understand', 'like', 'heavy', 'soul', 'one', 'under
stood', 'issue', 'sometimes', 'family', 'suck', 'sometimes', 'everyo
ne', 'suck', 'towards', 'youthis', 'give', 'goodbye', 'last', 'long'
, 'hotline', 'sucksok', 'trying', 'understand', 'like', 'hotline', '
wtf', 'supposed', 'actually', 'ask', 'tell', 'sound', 'like', 'hate'
, 'job']
```

In [ ]:

In [ ]:

2. Part-Of-Speech(POS) Analysis

: Grammatical tagging is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context.

Generate Noun list and adjective list using Pos_tag

1. Noun pos_tag :

NN : noun, singular (cat, tree), NNS :noun plural (desks), NNP: proper noun, singular (sarah), NNPS :proper noun, plural (indians or americans)

2. Adjective pos_tag :

JJ : adjective (large), JJR: adjective, comparative (larger), JJS : adjective, superlative (largest)

In [21]:

```
SuicideWatch_pos = nltk.pos_tag(SuicideWatch_list_lower_stop_pun_extra_lemmatize
d_stop)
SuicideWatch_pos[:10]
```

Out[21]:

```
[('love', 'NN'),
 ('guy', 'NN'),
 ('subreddit', 'NN'),
 ('made', 'VBD'),
 ('feel', 'JJ'),
 ('le', 'JJ'),
 ('alone', 'RB'),
 ('take', 'VBP'),
 ('care', 'NN'),
 ('gone', 'VBN')]
```

In [22]:

```
# generate Noun list and adjective
NN_list = []
AJ_list = []
for i,j in SuicideWatch_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        AJ_list.append(i)
print('There are',len(NN_list),'nouns in the list ')
print('There are',len(AJ_list),'adjectives in the list')
```

```
There are 122830 nouns in the list
There are 59050 adjectives in the list
```

In [23]:

```
### Check the frequency of All words, Nouns, Adjectives
```

In [24]:

```
print("The top 30 mos frequent words are below : \n")
review_freq = nltk.FreqDist(SuicideWatch_pos)
review_freq_top = review_freq.most_common()
print(review_freq_top[:30])
```

```
The top 30 mos frequent words are below :

[(('like', 'IN'), 4160), (('want', 'VBP'), 3546), (('life', 'NN'), 3
487), (('even', 'RB'), 2506), (('time', 'NN'), 2474), (('people', 'N
NS'), 2386), (('would', 'MD'), 2201), (('one', 'CD'), 2194), (('know
', 'VBP'), 2157), (('year', 'NN'), 2049), (('really', 'RB'), 2038),
(('feel', 'NN'), 1981), (('thing', 'NN'), 1917), (('get', 'VB'), 184
2), (('day', 'NN'), 1787), (('never', 'RB'), 1766), (('going', 'VBG'
), 1713), (('friend', 'NN'), 1367), (('anymore', 'RB'), 1357), (('co
uld', 'MD'), 1324), (('think', 'VBP'), 1272), (('anything', 'NN'), 1
217), (('way', 'NN'), 1205), (('much', 'JJ'), 1171), (('feel', 'VB')
, 1168), (('always', 'RB'), 1142), (('help', 'NN'), 1141), (('everyt
hing', 'NN'), 1125), (('nothing', 'NN'), 1124), (('someone', 'NN'),
1104)]
```

In [25]:

```
print("The top 50 most frequent nouns are below : \n" )
noun_review_freq = nltk.FreqDist(NN_list)
noun_review_freq_top = noun_review_freq.most_common()
print(noun_review_freq_top[:50])
```

```
The top 50 most frequent nouns are below :

[('life', 3487), ('time', 2474), ('people', 2386), ('feel', 2065), (
'year', 2049), ('thing', 1917), ('day', 1787), ('friend', 1369), ('a
nything', 1217), ('way', 1205), ('help', 1142), ('everything', 1125)
, ('nothing', 1124), ('someone', 1104), ('family', 1074), ('anyone',
932), ('school', 905), ('everyone', 895), ('something', 885), ('end'
, 854), ('kill', 840), ('work', 799), ('parent', 795), ('care', 777)
, ('point', 757), ('die', 755), ('month', 754), ('get', 738), ('pers
on', 719), ('job', 704), ('pain', 698), ('talk', 692), ('hate', 668)
, ('reason', 665), ('world', 654), ('week', 587), ('depression', 530
), ('mom', 527), ('dont', 526), ('death', 488), ('lot', 482), ('home
', 473), ('suicide', 424), ('problem', 409), ('place', 399), ('shit'
, 387), ('night', 387), ('relationship', 386), ('try', 376), ('cry',
364)]
```

In [26]:

```
print("The top 50 most frequent adjectives are below : \n ")

adjective_review_freq = nltk.FreqDist(AJ_list)
adjective_review_freq_top = adjective_review_freq.most_common()
print(adjective_review_freq_top[:50])
```

```
The top 50 most frequent adjectives are below :

[('much', 1171), ('good', 1033), ('live', 830), ('last', 793), ('bad
', 785), ('want', 732), ('happy', 614), ('many', 514), ('worse', 512
), ('better', 485), ('hard', 483), ('wish', 470), ('able', 454), ('b
est', 450), ('know', 445), ('suicide', 441), ('right', 408), ('suici
dal', 401), ('die', 384), ('mental', 381), ('alive', 374), ('sure',
359), ('wrong', 356), ('little', 352), ('old', 344), ('long', 329),
('first', 326), ('new', 326), ('sad', 311), ('dead', 311), ('sick',
310), ('whole', 305), ('stupid', 280), ('give', 278), ('next', 278),
('cant', 276), ('high', 272), ('least', 270), ('real', 266), ('shit'
, 247), ('enough', 239), ('social', 231), ('sleep', 218), ('tired',
217), ('single', 216), ('fuck', 212), ('tried', 209), ('everyday', 2
03), ('great', 197), ('due', 196)]
```

3. Ngram Analysis -Bi,Tri,Quad gram

: a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application

In [27]:

```
finder = BigramCollocationFinder.from_words(SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop, window_size = 2)
#bigram_measures = nltk.collocations.BigramAssocMeasures()

for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:20] :
    print(k,v)
```

```
('feel', 'like') 1592
('want', 'die') 468
('get', 'better') 312
('every', 'day') 281
('kill', 'kill') 262
('suicidal', 'thought') 222
('dont', 'want') 203
('want', 'kill') 195
('year', 'old') 195
('even', 'though') 194
('year', 'ago') 193
('dont', 'know') 179
('want', 'end') 179
('best', 'friend') 179
('make', 'feel') 172
('wish', 'could') 168
('really', 'want') 166
('want', 'live') 160
('one', 'day') 154
('mental', 'health') 149
```

In [28]:

```
# Check the most fequently appeared contents with 3-grams

finder = TrigramCollocationFinder.from_words(SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop, window_size = 3)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:20] :
    print(k,v)
```

```
('kill', 'kill', 'kill') 253
('want', 'die', 'want') 62
('every', 'single', 'day') 48
('thing', 'get', 'better') 44
('make', 'feel', 'like') 35
('feel', 'like', 'shit') 35
('really', 'want', 'die') 33
('feel', 'like', 'life') 32
('feel', 'like', 'one') 31
('need', 'someone', 'talk') 31
('life', 'feel', 'like') 30
('want', 'live', 'anymore') 30
('feel', 'like', 'burden') 30
('know', 'much', 'longer') 29
('want', 'feel', 'like') 27
('getting', 'worse', 'worse') 26
('life', 'worth', 'living') 25
('make', 'feel', 'better') 25
('anymore', 'feel', 'like') 24
('time', 'feel', 'like') 24
```

In [29]:

```
# Check the most frequently appeared contents with quadgram

finder = QuadgramCollocationFinder.from_words(SuicideWatch_list_lower_stop_pun_extra_lemmatized_stop, window_size = 4)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:20] :
    print(k,v)
```

```
('kill', 'kill', 'kill', 'kill') 249
('never', 'good', 'enough', 'never') 12
('good', 'enough', 'never', 'good') 11
('enough', 'never', 'good', 'enough') 11
('thing', 'would', 'get', 'better') 9
('want', 'die', 'feel', 'like') 9
('know', 'much', 'longer', 'go') 8
('feel', 'like', 'burden', 'everyone') 8
('nothing', 'make', 'happy', 'anymore') 7
('dont', 'want', 'live', 'anymore') 7
('make', 'feel', 'like', 'shit') 7
('go', 'sleep', 'never', 'wake') 7
('really', 'want', 'die', 'want') 7
('want', 'die', 'want', 'live') 7
('want', 'die', 'want', 'stop') 7
('want', 'die', 'dont', 'want') 7
('anyone', 'else', 'feel', 'way') 6
('anyone', 'else', 'feel', 'like') 6
('world', 'would', 'better', 'without') 6
('keep', 'getting', 'worse', 'worse') 6
```

Depressed

In [30]:

```
## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now

sampledepressed_list = depressed_df.sample(frac=0.1, replace=True, random_state=
1)
depressed_list = sampledepressed_list['title_with_selftext_cleaned'].tolist()
depressed_list_lower = [tok.lower() for i in depressed_list for tok in nltk.word
_tokenize(i)]
nltk_stopwords = set(stopwords.words('english'))
depressed_list_lower_stop = [x for x in depressed_list_lower if not x in nltk_st
opwords]
depressed_list_lower_stop_pun = [y for y in depressed_list_lower_stop if not alp
ha_filter(y)]
depressed_list_lower_stop_pun_extra = [''.join(x for x in par if x not in string
.punctuation) for par in\
                                       depressed_list_lower_stop_pun]
porter = WordNetLemmatizer()
depressed_list_lower_stop_pun_extra_lemmatized = []
for a in depressed_list_lower_stop_pun_extra :
    depressed_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a))
depressed_list_lower_stop_pun_extra_lemmatized_stop = [x for x in depressed_list
_lower_stop_pun_extra_lemmatized if not x in cachedStopWords]
```

In [31]:

```
Depressed_pos = nltk.pos_tag(depressed_list_lower_stop_pun_extra_lemmatized_stop
)
# generate Noun list and adjective
NN_list = []
AJ_list = []
for i,j in Depressed_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        AJ_list.append(i)
print('There are',len(NN_list),'nouns in the list ')
print('There are',len(AJ_list),'adjectives in the list')
```

```
There are 78161 nouns in the list
There are 37022 adjectives in the list
```

In [32]:

```
print("The top 30 mos frequent words are below : \n")
review_freq = nltk.FreqDist(Depressed_pos)
review_freq_top = review_freq.most_common()
print(review_freq_top[:30])
```

```
The top 30 mos frequent words are below :

[(('like', 'IN'), 3154), (('life', 'NN'), 1981), (('time', 'NN'), 17
48), (('want', 'VBP'), 1625), (('feel', 'NN'), 1597), (('one', 'CD')
, 1572), (('even', 'RB'), 1489), (('know', 'VBP'), 1462), (('people'
, 'NNS'), 1451), (('really', 'RB'), 1425), (('year', 'NN'), 1416), (
('day', 'NN'), 1309), (('thing', 'NN'), 1303), (('friend', 'NN'), 12
33), (('would', 'MD'), 1226), (('get', 'VB'), 1116), (('never', 'RB'
), 1071), (('going', 'VBG'), 985), (('feel', 'VB'), 895), (('could',
'MD'), 859), (('always', 'RB'), 816), (('someone', 'NN'), 783), (('t
hink', 'VBP'), 774), (('anything', 'NN'), 766), (('got', 'VBD'), 756
), (('school', 'NN'), 751), (('everything', 'NN'), 708), (('much', '
JJ'), 697), (('still', 'RB'), 697), (('make', 'VBP'), 693)]
```

In [33]:

```
print("The top 50 most frequent nouns are below : \n" )
noun_review_freq = nltk.FreqDist(NN_list)
noun_review_freq_top = noun_review_freq.most_common()
print(noun_review_freq_top[:50])
```

```
The top 50 most frequent nouns are below :

[('life', 1981), ('time', 1748), ('feel', 1669), ('people', 1451), (
'year', 1416), ('day', 1309), ('thing', 1303), ('friend', 1236), ('s
omeone', 783), ('anything', 766), ('school', 751), ('everything', 70
8), ('person', 687), ('help', 676), ('way', 676), ('depression', 653
), ('something', 648), ('work', 616), ('nothing', 614), ('anyone', 6
11), ('family', 594), ('job', 557), ('talk', 557), ('month', 537), (
'everyone', 492), ('get', 462), ('week', 417), ('reason', 386), ('lo
t', 385), ('point', 384), ('care', 376), ('home', 365), ('parent', 3
64), ('hate', 350), ('relationship', 339), ('mom', 329), ('cry', 328
), ('problem', 326), ('night', 308), ('end', 286), ('feeling', 284),
('girl', 276), ('world', 274), ('try', 273), ('love', 264), ('colleg
e', 264), ('guy', 260), ('hour', 257), ('today', 253), ('anxiety', 2
52)]
```

In [34]:

```
print("The top 50 most frequent adjectives are below : \n ")

adjective_review_freq = nltk.FreqDist(AJ_list)
adjective_review_freq_top = adjective_review_freq.most_common()
print(adjective_review_freq_top[:50])
```

```
The top 50 most frequent adjectives are below :

[('much', 697), ('good', 667), ('happy', 586), ('last', 493), ('bad'
, 486), ('hard', 366), ('sad', 334), ('live', 332), ('wrong', 320),
('depressed', 319), ('want', 318), ('best', 316), ('little', 291), (
'old', 287), ('new', 283), ('worse', 279), ('many', 277), ('know', 2
72), ('first', 270), ('right', 264), ('better', 249), ('high', 240),
('sure', 222), ('wish', 221), ('real', 216), ('able', 207), ('shit',
202), ('mental', 189), ('long', 185), ('whole', 184), ('sleep', 175)
, ('great', 175), ('social', 174), ('normal', 167), ('friend', 163),
('different', 162), ('give', 160), ('empty', 149), ('least', 139), (
'due', 138), ('next', 134), ('everyday', 134), ('stupid', 131), ('si
ck', 121), ('nice', 118), ('enough', 114), ('suicide', 109), ('mean'
, 109), ('cant', 106), ('horrible', 104)]
```

In [35]:

```
finder = BigramCollocationFinder.from_words(depressed_list_lower_stop_pun_extra_
lemmatized_stop, window_size = 2)
#bigram_measures = nltk.collocations.BigramAssocMeasures()

for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('feel', 'like') 1178
('shit', 'person') 182
('person', 'shit') 181
('make', 'feel') 177
('year', 'old') 140
('even', 'though') 136
('high', 'school') 136
('every', 'day') 132
('best', 'friend') 129
('wish', 'could') 127
('year', 'ago') 111
('felt', 'like') 105
('anyone', 'else') 102
('last', 'year') 102
('first', 'time') 101
('want', 'die') 97
('get', 'better') 96
('mental', 'health') 93
('even', 'know') 91
('dont', 'know') 90
```

In [36]:

```
# Check the most fequently appeared contents with 3-grams

finder = TrigramCollocationFinder.from_words(depressed_list_lower_stop_pun_extra
_lemmatized_stop, window_size = 3)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('shit', 'person', 'shit') 179
('person', 'shit', 'person') 178
('make', 'feel', 'like') 39
('knew', 'knew', 'knew') 38
('feel', 'like', 'life') 33
('feel', 'like', 'shit') 33
('feel', 'like', 'going') 30
('make', 'feel', 'better') 24
('sometimes', 'feel', 'like') 23
('matter', 'hard', 'try') 23
('need', 'someone', 'talk') 22
('feel', 'like', 'nothing') 22
('anyone', 'else', 'feel') 21
('feel', 'like', 'failure') 21
('every', 'single', 'day') 19
('play', 'video', 'game') 18
('feel', 'like', 'one') 17
('day', 'feel', 'like') 17
('know', 'feel', 'like') 16
('else', 'feel', 'like') 16
```

In [37]:

```
# Check the most frequently appeared contents with quadgram

finder = QuadgramCollocationFinder.from_words(depressed_list_lower_stop_pun_extr
a_lemmatized_stop, window_size = 4)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('shit', 'person', 'shit', 'person') 178
('person', 'shit', 'person', 'shit') 178
('knew', 'knew', 'knew', 'knew') 24
('anyone', 'else', 'feel', 'like') 12
('wish', 'could', 'go', 'back') 11
('tired', 'tired', 'tired', 'tired') 9
('dying', 'knew', 'knew', 'knew') 8
('know', 'anymore', 'feel', 'like') 7
('knew', 'dying', 'knew', 'knew') 7
('anyone', 'else', 'feel', 'way') 6
('every', 'day', 'feel', 'like') 6
('second', 'full', 'blown', 'manic') 6
('knew', 'knew', 'dying', 'knew') 6
('feel', 'like', 'going', 'insane') 5
('make', 'feel', 'like', 'shit') 5
('knew', 'knew', 'knew', 'dying') 5
('told', 'halloween', 'got', 'job') 4
('halloween', 'got', 'job', 'today') 4
('got', 'job', 'today', '20th') 4
('job', 'today', '20th', 'got') 4
```

SelfImprovement

In [38]:

```
## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now

sampleselfimprovement_list = selfimprovement_df.sample(frac=0.1, replace=True, r
andom_state=1)
selfimprovement_list = sampleselfimprovement_list['title_with_selftext_cleaned']
.tolist()
selfimprovement_list_lower = [tok.lower() for i in selfimprovement_list for tok
in nltk.word_tokenize(i)]
nltk_stopwords = set(stopwords.words('english'))
selfimprovement_list_lower_stop = [x for x in selfimprovement_list_lower if not
x in nltk_stopwords]
selfimprovement_list_lower_stop_pun = [y for y in selfimprovement_list_lower_sto
p if not alpha_filter(y)]
selfimprovement_list_lower_stop_pun_extra = [''.join(x for x in par if x not in
string.punctuation) for par in\
                                             selfimprovement_list_lower_stop_pun]
porter = WordNetLemmatizer()
selfimprovement_list_lower_stop_pun_extra_lemmatized = []
for a in selfimprovement_list_lower_stop_pun_extra :
    selfimprovement_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize
(a))
selfimprovement_list_lower_stop_pun_extra_lemmatized_stop = [x for x in selfimpr
ovement_list_lower_stop_pun_extra_lemmatized if not x in cachedStopWords]
```

In [39]:

```
selfimprovement_pos = nltk.pos_tag(selfimprovement_list_lower_stop_pun_extra_lem
matized_stop)
# generate Noun list and adjective
NN_list = []
AJ_list = []
for i,j in selfimprovement_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        AJ_list.append(i)
print('There are',len(NN_list),'nouns in the list ')
print('There are',len(AJ_list),'adjectives in the list')
```

```
There are 172845 nouns in the list
There are 82329 adjectives in the list
```

In [40]:

```
print("The top 30 mos frequent words are below : \n")
review_freq = nltk.FreqDist(selfimprovement_pos)
review_freq_top = review_freq.most_common()
print(review_freq_top[:30])
```

```
The top 30 mos frequent words are below :

[(('like', 'IN'), 5029), (('life', 'NN'), 4095), (('time', 'NN'), 40
63), (('people', 'NNS'), 3603), (('thing', 'NN'), 3492), (('want', '
VBP'), 3014), (('year', 'NN'), 2686), (('really', 'RB'), 2677), (('d
ay', 'NN'), 2409), (('would', 'MD'), 2353), (('one', 'CD'), 2297), (
('even', 'RB'), 2254), (('get', 'VB'), 2194), (('know', 'VBP'), 2102
), (('feel', 'NN'), 2017), (('work', 'NN'), 1994), (('way', 'NN'), 1
856), (('something', 'NN'), 1789), (('good', 'JJ'), 1754), (('always
', 'RB'), 1620), (('going', 'VBG'), 1591), (('make', 'VBP'), 1575),
(('never', 'RB'), 1531), (('friend', 'NN'), 1494), (('think', 'VBP')
, 1491), (('also', 'RB'), 1426), (('much', 'JJ'), 1320), (('still',
'RB'), 1297), (('lot', 'NN'), 1282), (('person', 'NN'), 1254)]
```

In [41]:

```
print("The top 50 most frequent nouns are below : \n" )
noun_review_freq = nltk.FreqDist(NN_list)
noun_review_freq_top = noun_review_freq.most_common()
print(noun_review_freq_top[:50])
```

```
The top 50 most frequent nouns are below :

[('life', 4095), ('time', 4063), ('people', 3603), ('thing', 3492),
('year', 2686), ('day', 2409), ('feel', 2092), ('work', 1994), ('way
', 1856), ('something', 1789), ('friend', 1495), ('lot', 1286), ('pe
rson', 1254), ('job', 1252), ('help', 1149), ('anything', 1102), ('s
chool', 1073), ('someone', 1041), ('goal', 1009), ('month', 945), ('
problem', 941), ('get', 913), ('change', 901), ('everything', 853),
('others', 851), ('anyone', 842), ('week', 810), ('book', 766), ('re
lationship', 704), ('guy', 678), ('advice', 663), ('family', 660), (
'point', 657), ('everyone', 654), ('nothing', 652), ('habit', 643),
('self', 620), ('start', 597), ('college', 590), ('mind', 577), ('ta
lk', 555), ('hour', 552), ('part', 546), ('home', 539), ('money', 53
7), ('idea', 528), ('kind', 524), ('parent', 518), ('try', 515), ('n
eed', 513)]
```

In [42]:

```
print("The top 50 most frequent adjectives are below : \n ")

adjective_review_freq = nltk.FreqDist(AJ_list)
adjective_review_freq_top = adjective_review_freq.most_common()
print(adjective_review_freq_top[:50])
```

```
The top 50 most frequent adjectives are below :

[('good', 1754), ('much', 1320), ('new', 1100), ('social', 915), ('b
ad', 838), ('many', 764), ('hard', 764), ('happy', 641), ('best', 63
8), ('last', 630), ('great', 612), ('old', 606), ('better', 595), ('
little', 593), ('want', 593), ('right', 570), ('able', 566), ('first
', 544), ('sure', 453), ('high', 445), ('long', 439), ('live', 436),
('different', 419), ('next', 417), ('know', 415), ('real', 412), ('b
ig', 394), ('negative', 370), ('mental', 363), ('wrong', 356), ('lea
st', 349), ('important', 340), ('small', 336), ('positive', 335), ('
whole', 326), ('free', 308), ('give', 308), ('past', 284), ('low', 2
80), ('healthy', 279), ('possible', 278), ('personal', 272), ('enoug
h', 265), ('due', 263), ('friend', 256), ('full', 255), ('short', 24
4), ('daily', 244), ('stop', 239), ('normal', 235)]
```

In [43]:

```
finder = BigramCollocationFinder.from_words(selfimprovement_list_lower_stop_pun_
extra_lemmatized_stop, window_size = 2)
#bigram_measures = nltk.collocations.BigramAssocMeasures()

for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('feel', 'like') 1566
('year', 'old') 345
('every', 'day') 292
('even', 'though') 286
('self', 'improvement') 268
('high', 'school') 259
('make', 'feel') 240
('social', 'medium') 216
('year', 'ago') 212
('really', 'want') 195
('would', 'like') 193
('video', 'game') 191
('last', 'year') 169
('thing', 'like') 158
('go', 'back') 152
('felt', 'like') 145
('long', 'time') 140
('get', 'better') 137
('self', 'esteem') 132
('need', 'help') 132
```

In [44]:

```
# Check the most fequently appeared contents with 3-grams

finder = TrigramCollocationFinder.from_words(selfimprovement_list_lower_stop_pun
_extra_lemmatized_stop, window_size = 3)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('year', 'old', 'male') 47
('playing', 'video', 'game') 43
('make', 'feel', 'like') 43
('low', 'self', 'esteem') 39
('play', 'video', 'game') 36
('life', 'feel', 'like') 34
('sometimes', 'feel', 'like') 34
('feel', 'like', 'going') 30
('every', 'single', 'day') 29
('long', 'story', 'short') 29
('feel', 'like', 'shit') 29
('year', 'high', 'school') 28
('still', 'feel', 'like') 28
('self', 'help', 'book') 28
('meet', 'new', 'people') 26
('really', 'feel', 'like') 26
('successful', 'people', 'refuse') 26
('make', 'new', 'friend') 25
('feel', 'like', 'need') 25
('always', 'feel', 'like') 25
```

In [45]:

```
# Check the most frequently appeared contents with quadgram
finder = QuadgramCollocationFinder.from_words(selfimprovement_list_lower_stop_pu
n_extra_lemmatized_stop, window_size = 4)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('win', 'friend', 'influence', 'people') 10
('senior', 'year', 'high', 'school') 9
('middle', 'school', 'high', 'school') 7
('feel', 'like', 'good', 'enough') 7
('year', 'old', 'feel', 'like') 7
('anyone', 'else', 'feel', 'like') 7
('last', 'year', 'high', 'school') 7
('first', 'time', 'long', 'time') 7
('advice', 'would', 'greatly', 'appreciated') 6
('day', 'playing', 'video', 'game') 6
('get', 'one', 'word', 'reply') 6
('negative', 'thought', 'feeling', 'social') 6
('thought', 'feeling', 'social', 'isolation') 6
('year', 'old', 'college', 'student') 5
('information', 'year', 'old', 'middleeastern') 5
('year', 'old', 'middleeastern', 'guy') 5
('old', 'middleeastern', 'guy', 'figgity') 5
('middleeastern', 'guy', 'figgity', 'impulsive') 5
('guy', 'figgity', 'impulsive', 'really') 5
('figgity', 'impulsive', 'really', 'think') 5
```

Happy

In [46]:

```
## ### Because of relatively huge dataset, we need to perform random sampling of
10% for now

samplehappy_list = happy_df.sample(frac=0.1, replace=True, random_state=1)
happy_list = samplehappy_list['title_with_selftext_cleaned'].tolist()
happy_list_lower = [tok.lower() for i in happy_list for tok in nltk.word_tokeniz
e(i)]
nltk_stopwords = set(stopwords.words('english'))
happy_list_lower_stop = [x for x in happy_list_lower if not x in nltk_stopwords]
happy_list_lower_stop_pun = [y for y in happy_list_lower_stop if not alpha_filte
r(y)]
happy_list_lower_stop_pun_extra = [''.join(x for x in par if x not in string.pun
ctuation) for par in\
                                        happy_list_lower_stop_pun]
porter = WordNetLemmatizer()
happy_list_lower_stop_pun_extra_lemmatized = []
for a in happy_list_lower_stop_pun_extra :
    happy_list_lower_stop_pun_extra_lemmatized.append(porter.lemmatize(a))
happy_list_lower_stop_pun_extra_lemmatized_stop = [x for x in happy_list_lower_s
top_pun_extra_lemmatized if not x in cachedStopWords]
```

In [47]:

```
happy_pos = nltk.pos_tag(happy_list_lower_stop_pun_extra_lemmatized_stop)
# generate Noun list and adjective
NN_list = []
AJ_list = []
for i,j in happy_pos:
    #print(i)
    if j == 'NN' or j == 'NNS' or j == 'NNP' or j == 'NNPS':
        NN_list.append(i)
    elif j == 'JJ' or j == 'JJS' or j == 'JJR':
        AJ_list.append(i)
print('There are',len(NN_list),'nouns in the list ')
print('There are',len(AJ_list),'adjectives in the list')
```

```
There are 35265 nouns in the list
There are 17271 adjectives in the list
```

In [48]:

```
print("The top 30 mos frequent words are below : \n")
review_freq = nltk.FreqDist(happy_pos)
review_freq_top = review_freq.most_common()
print(review_freq_top[:30])

print("The top 50 most frequent nouns are below : \n" )
noun_review_freq = nltk.FreqDist(NN_list)
noun_review_freq_top = noun_review_freq.most_common()
print(noun_review_freq_top[:50])

print("The top 50 most frequent adjectives are below : \n ")
adjective_review_freq = nltk.FreqDist(AJ_list)
adjective_review_freq_top = adjective_review_freq.most_common()
print(adjective_review_freq_top[:50])
```

The top 30 mos frequent words are below :

[(('happy', 'JJ'), 1350), (('year', 'NN'), 1273), (('time', 'NN'), 7
91), (('day', 'NN'), 763), (('life', 'NN'), 744), (('got', 'VBD'), 7
33), (('today', 'NN'), 639), (('finally', 'RB'), 577), (('like', 'IN
'), 548), (('one', 'CD'), 481), (('really', 'RB'), 466), (('month',
'NN'), 464), (('first', 'JJ'), 448), (('friend', 'NN'), 404), (('job
', 'NN'), 397), (('thing', 'NN'), 378), (('good', 'JJ'), 364), (('ne
ver', 'RB'), 358), (('week', 'NN'), 340), (('could', 'MD'), 334), ((
'make', 'VBP'), 327), (('last', 'JJ'), 326), (('people', 'NNS'), 318
), (('would', 'MD'), 310), (('best', 'JJS'), 305), (('get', 'VB'), 3
04), (('going', 'VBG'), 296), (('new', 'JJ'), 291), (('even', 'RB'),
278), (('ago', 'RB'), 276)]
The top 50 most frequent nouns are below :

[('year', 1273), ('time', 791), ('day', 763), ('life', 744), ('today
', 639), ('month', 464), ('friend', 404), ('job', 397), ('thing', 37
8), ('week', 340), ('people', 318), ('work', 276), ('family', 262),
('school', 250), ('feel', 227), ('way', 215), ('love', 196), ('world
', 193), ('home', 190), ('something', 179), ('lot', 170), ('someone'
, 165), ('night', 160), ('guy', 157), ('everything', 154), ('wife',
151), ('relationship', 148), ('college', 144), ('share', 143), ('hou
r', 140), ('happiness', 133), ('depression', 131), ('moment', 129),
('dream', 128), ('person', 128), ('house', 125), ('get', 125), ('hel
p', 122), ('place', 120), ('mom', 119), ('everyone', 117), ('yesterd
ay', 116), ('girl', 115), ('man', 113), ('birthday', 108), ('dog', 1
08), ('kid', 105), ('game', 96), ('post', 95), ('video', 94)]
The top 50 most frequent adjectives are below :

[('happy', 1350), ('first', 448), ('good', 364), ('last', 326), ('be
st', 305), ('new', 291), ('little', 263), ('much', 194), ('great', 1
78), ('old', 169), ('long', 158), ('able', 150), ('hard', 131), ('ma
ny', 129), ('big', 122), ('right', 110), ('bad', 110), ('beautiful',
100), ('small', 94), ('full', 90), ('high', 89), ('whole', 86), ('pr
oud', 85), ('nice', 84), ('smile', 83), ('next', 82), ('live', 79),
('better', 78), ('free', 76), ('second', 75), ('amazing', 75), ('rea
l', 74), ('wonderful', 72), ('different', 71), ('sure', 69), ('happi
est', 63), ('favorite', 63), ('mental', 61), ('positive', 61), ('due
', 55), ('girl', 54), ('entire', 51), ('happier', 49), ('grateful',
49), ('single', 48), ('married', 48), ('perfect', 48), ('awesome', 4
8), ('super', 45), ('want', 44)]

In [49]:

```
finder = BigramCollocationFinder.from_words(happy_list_lower_stop_pun_extra_lemm
atized_stop, window_size = 2)
#bigram_measures = nltk.collocations.BigramAssocMeasures()

for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('first', 'time') 206
('year', 'ago') 177
('make', 'happy') 148
('feel', 'like') 137
('best', 'friend') 119
('month', 'ago') 86
('long', 'time') 82
('last', 'year') 76
('year', 'old') 71
('finally', 'got') 66
('high', 'school') 61
('felt', 'like') 58
('wanted', 'share') 51
('every', 'day') 50
('made', 'happy') 47
('make', 'feel') 46
('two', 'year') 46
('last', 'night') 45
('love', 'life') 43
('really', 'happy') 40
```

In [50]:

```
# Check the most fequently appeared contents with 3-grams

finder = TrigramCollocationFinder.from_words(happy_list_lower_stop_pun_extra_lem
matized_stop, window_size = 3)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('first', 'time', 'long') 20
('first', 'time', 'life') 19
('first', 'time', 'ever') 16
('first', 'time', 'year') 16
('year', 'ago', 'today') 16
('happy', 'first', 'time') 14
('time', 'long', 'time') 14
('happy', 'new', 'year') 13
('today', 'good', 'day') 11
('best', 'thing', 'ever') 9
('married', 'love', 'life') 9
('happy', 'long', 'time') 9
('married', 'best', 'friend') 9
('full', 'time', 'job') 9
('time', 'last', 'year') 9
('two', 'year', 'ago') 9
('got', 'new', 'job') 9
('dream', 'come', 'true') 8
('put', 'smile', 'face') 8
('finally', 'feel', 'like') 8
```

In [51]:

```
# Check the most frequently appeared contents with quadgram

finder = QuadgramCollocationFinder.from_words(happy_list_lower_stop_pun_extra_le
mmatized_stop, window_size = 4)
for k,v in sorted(finder.ngram_fd.items(), key=lambda t:t[-1], reverse=True)[0:2
0] :
    print(k,v)
```

```
('first', 'time', 'long', 'time') 14
('make', 'feel', 'warm', 'fuzzy') 5
('first', 'time', 'entire', 'life') 5
('met', 'reddit', 'year', 'ago') 4
('reddit', 'year', 'ago', 'mile') 4
('year', 'ago', 'mile', 'away') 4
('ago', 'mile', 'away', 'married') 4
('mile', 'away', 'married', 'happiness') 4
('away', 'married', 'happiness', 'indescribable') 4
('car', 'front', 'paid', 'meal') 4
('always', 'self', 'confidence', 'issue') 4
('test', 'post', 'please', 'ignore') 4
('senior', 'year', 'high', 'school') 4
('graduated', 'high', 'school', 'earned') 4
('high', 'school', 'earned', 'eagle') 4
('school', 'earned', 'eagle', 'scout') 4
('earned', 'eagle', 'scout', 'day') 4
('month', 'ago', 'felt', 'like') 4
('best', 'thing', 'ever', 'happen') 3
('never', 'knew', 'happy', 'would') 3
```