Jeremy Reynolds
CPSC 4360
5/7/2019

Project Report:

Sentiment Analysis of Gaming Subreddits

Introduction

As someone who spends a fair amount of time on Reddit I was curious if there was a way to compare different subreddits to see if some are more positive or negative than others. I specifically wanted to compare gaming subreddits since these are best known for having toxicity. I drew inspiration for this project from Zach Stine's presentation about natural language processing. While his work is much more complicated I focused on sentiment analysis to draw comparisons between the subreddits. Two sentiment analysis software are used, TextBlob and VADER. I chose to use two analyzing software as they are not exact in their execution and often have misinterpretations which skew the results. This report outlines how data was collected and analyzed, along with detailed explanations into the programming aspect of those actions. It also goes into detail about future work that could be done in comparing subreddit sentiments.

Methodology

The first step in this process was to choose the subreddits to compare. I won't be listing any reference to a list of "the most toxic subreddits" as my own personal experience has shown me which are the most negative or positive. This choice is purely subjective just as any list would be since toxicity is highly biased to describe. I chose thirteen subreddits including an even mix of those that I've found to be slightly more or less toxic. I also included two other subreddits that I call my control values: r/CasualConversation and r/The_Donald. These two were chosen because the post length is typically much longer than those found on a gaming subreddit, and each represents one end of the sentiment spectrum (positive vs. negative). They act as a base ground for which to compare the gaming subreddits.

Once the targets were found the process of collecting the data began. I chose to collect all comments found on each subreddit beginning at January 1st 2019 and ending at midnight April 29th. This gave me a full four months of data to analyze. Reddit has an API service called the Pushshift API and a library called PRAW can easily access it. However another library, PSAW, was created which includes functions to search between two dates making my job much easier.

Using Python I created a script to search each subreddit for comments, and save those comments date, time, and text to a csv file. Since the API has a rate limit I chose to use the extra time to analyze each comment as they were being collected. This meant that it was slow enough to not hit the rate limit and little time was wasted.

To perform the sentiment analysis I used TextBlob's built in sentiment analyzer which returns a float value from -1.0 to +1.0. I immediately noticed that TextBlob would often assign a zero to a comment. This meant that it did not have enough information to assign a value which was troubling because nearly 70% of the comments from the gaming subreddits are less than 100 characters. As the character length increases, TextBlob is more likely to assign a sentiment value. To combat this problem I only looked at comments that were assigned a value other than 0.0 and contained more than 10 characters. This also made processing faster as there was less to analyze.

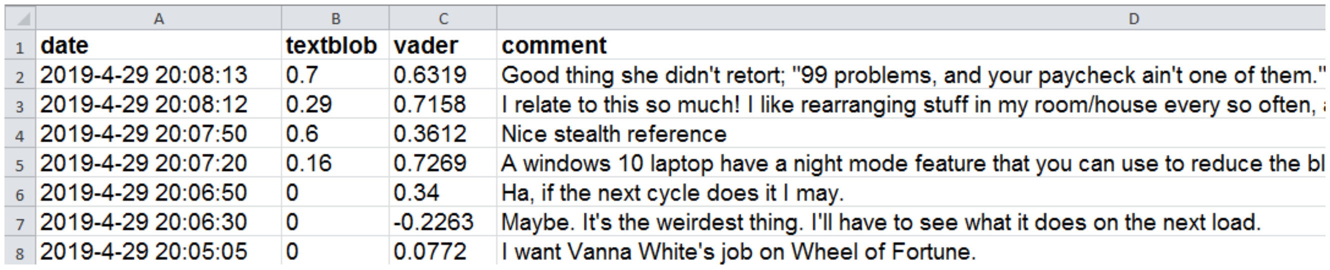
I did more research on sentiment analysis software and found another python package called VADER. Subjectively, this package seems to be more accurate and it rarely assigns a neutral zero value to a comment. To assign a final value for each subreddit I found the mean of the values assigned to each comment for both TextBlob and VADER. These final values were input into Excel and a chart was generated shown below (Figure 1).

Since I now had access to a large amount of data I also wanted to find the frequency of certain words found each day throughout the four month time range. To do this I developed another python script to search each post for a given word and add it to a frequency counter resetting each day. This allowed me to see when certain topics gain large momentum and then drop off as people lose interest. Figure 3 is a great real world example as it directly translates to a social/political event in Paris. Initially this frequency counter was going to be used to draw comparisons to people's sentiments when a game would update. It could show if the general consensus was positive or negative during the days of an update by searching for the words 'good' or 'bad' during that time range. Ultimately I was not successful in producing a recognizable trend that correlated to game update days. Perhaps with more data or larger, more controversial updates I may be able to see a trend.

Results

After twelve hours of scraping I had collected 13.8 million comments from the 13 subreddits listed below between January 1st 2019 and April 29th. This text was stored in csv files along with the date, time, TextBlob value, and VADER value.

These 13 files totaled 2.4 gigabytes of data which I compressed by 62% to 0.9 gigabytes. Table 1 depicts the data from the r/CasualConversation subreddit and shows how TextBlob and VADER compare in their analysis.



	A	B	C	D
1	date	textblob	vader	comment
2	2019-4-29 20:08:13	0.7	0.6319	Good thing she didn't retort; "99 problems, and your paycheck ain't one of them."
3	2019-4-29 20:08:12	0.29	0.7158	I relate to this so much! I like rearranging stuff in my room/house every so often,
4	2019-4-29 20:07:50	0.6	0.3612	Nice stealth reference
5	2019-4-29 20:07:20	0.16	0.7269	A windows 10 laptop have a night mode feature that you can use to reduce the bl
6	2019-4-29 20:06:50	0	0.34	Ha, if the next cycle does it I may.
7	2019-4-29 20:06:30	0	-0.2263	Maybe. It's the weirdest thing. I'll have to see what it does on the next load.
8	2019-4-29 20:05:05	0	0.0772	I want Vanna White's job on Wheel of Fortune.

Table 1: Example of collected data.

My comparison findings are shown in Figure 1 below. Notice the two control subreddits on opposite ends and the gaming subreddits in between them. I have ordered these smallest to largest based on the sentiment values from VADER for each subreddit.

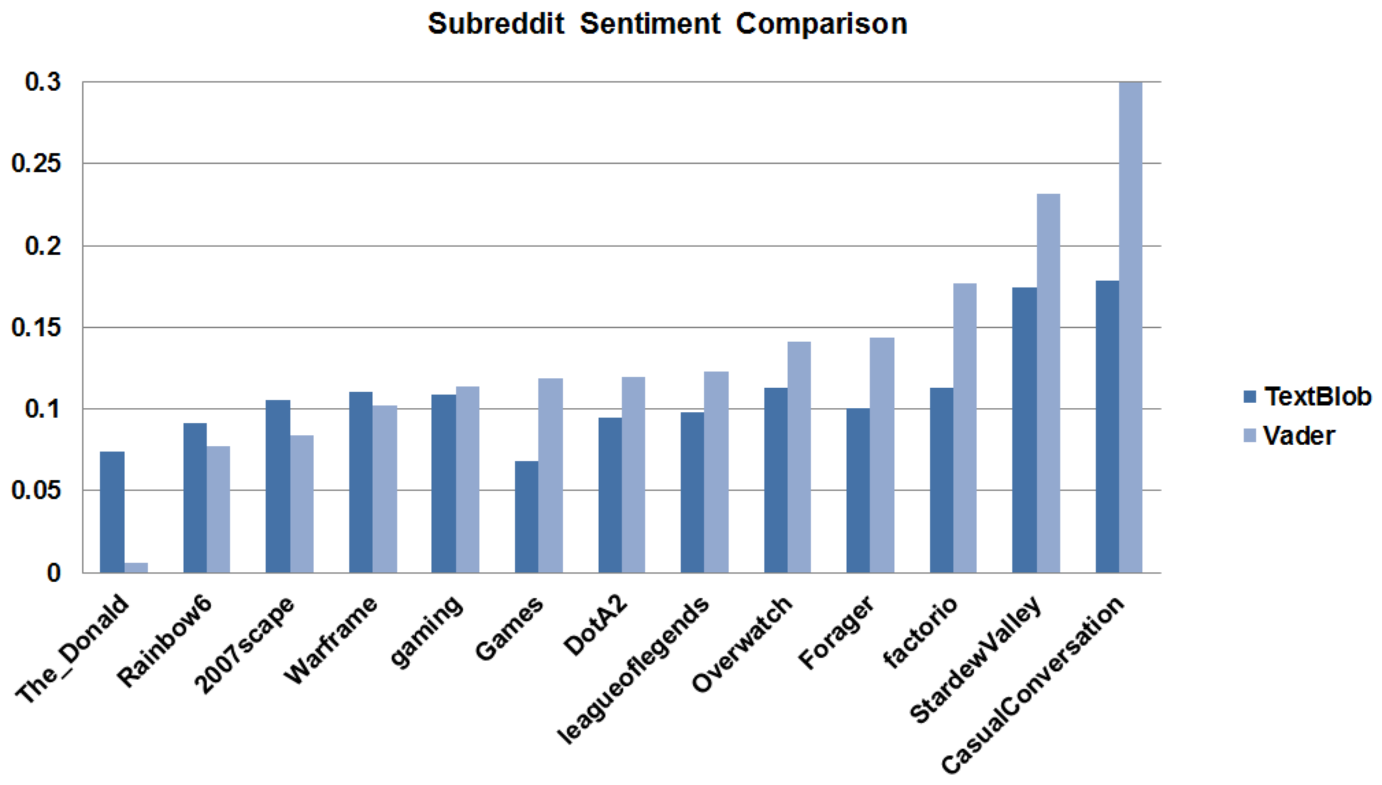


Figure 1: Sentiment analysis comparisons.

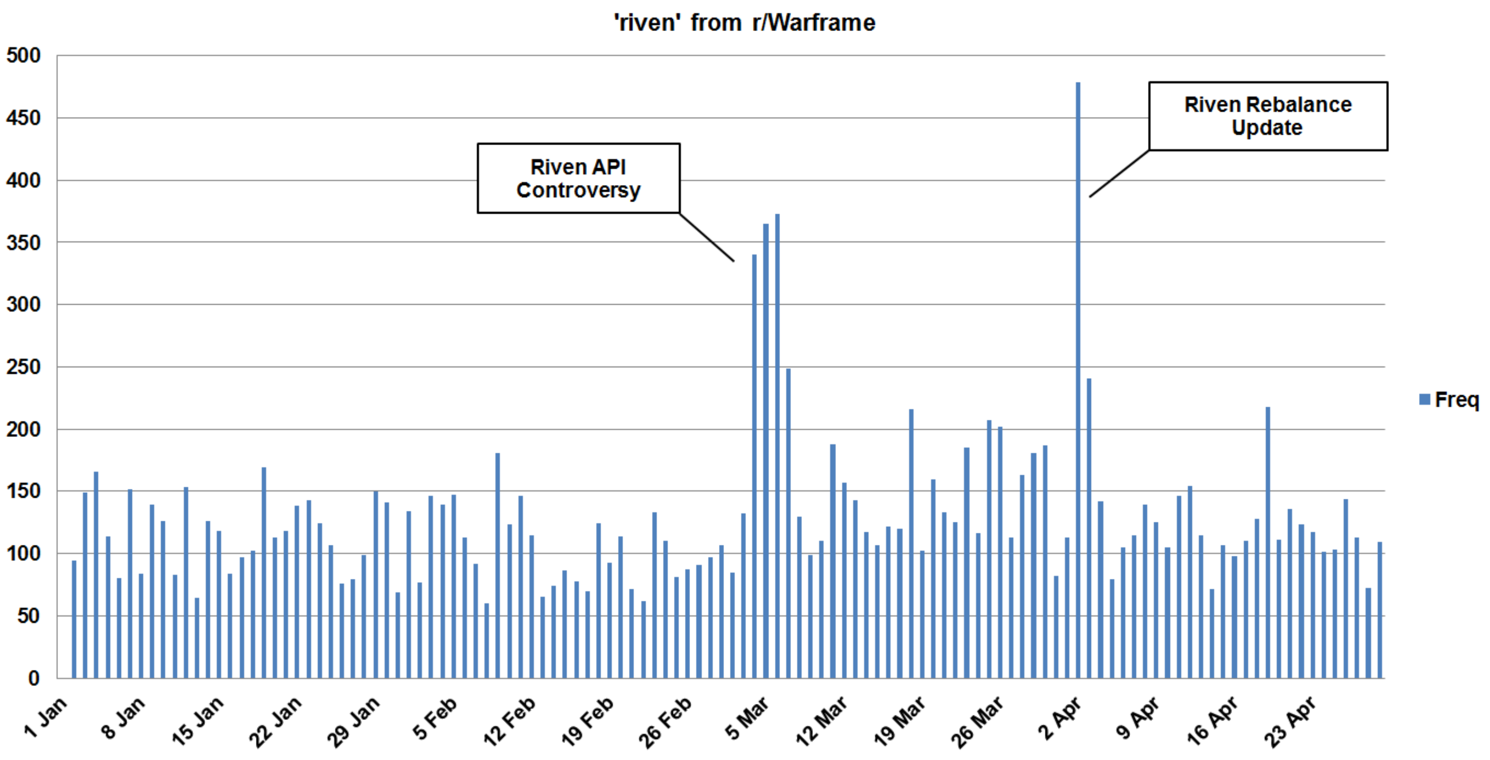


Figure 2: Word frequency per day example 1.

Figure 2 shows the frequency of the word ‘riven’ for each day on the r/Waframe subreddit.

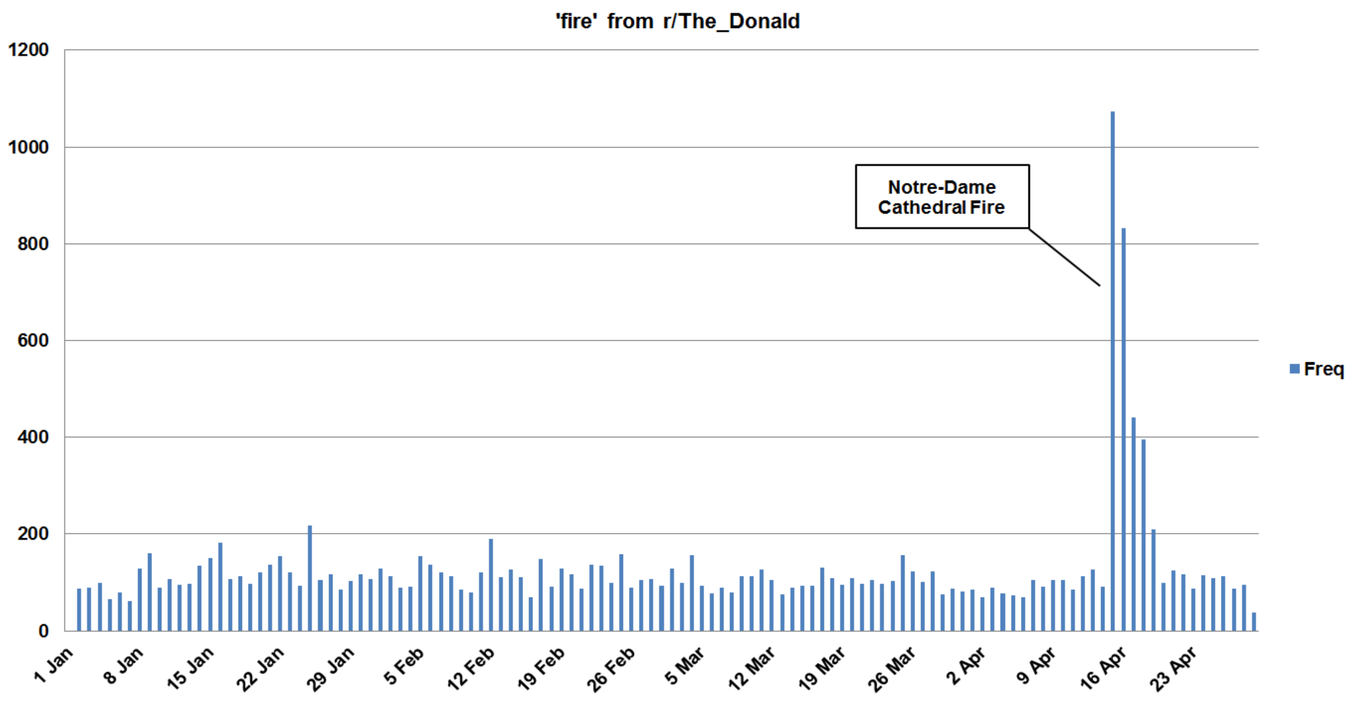


Figure 3: Word frequency per day example 2.

Figure 3 shows the frequency of the word ‘fire’ for each day on the r/The_Donald subreddit.

Discussion and Future Work

Sentiment analysis was a very common topic this semester with the other presentations. Nearly all mentioned the notable discrepancies with TextBlob and other similar software. It is obviously not an exact science. The VADER package allows for difference lexicons to be used to train the algorithm and possibly result in more accurate values. The largest limitation for these software is clearly the algorithms. It is difficult for machines to understand our language and it is made worse by analyzing text that is often short or abbreviated in some way.

There are several improvements that could be made to this project in the future. The most obvious is to simply collect more data. If I were to collect comments from several years I could easily run the sentiment analysis software on comments that are greater than 100 characters and likely get much more accurate results. I could also look at comments that have been upvoted or downvoted as these represent posts that are polarizing in some way. Trend lines would also become more apparent with a larger dataset. Perhaps people’s mood changes around the holidays or with the seasons each year.

Analyzing reading level is also something I would like to do in the future. There are several python packages, such as Textstat, that include readability and estimated reading grade level analyzers that would be simple to implement.

Conclusion

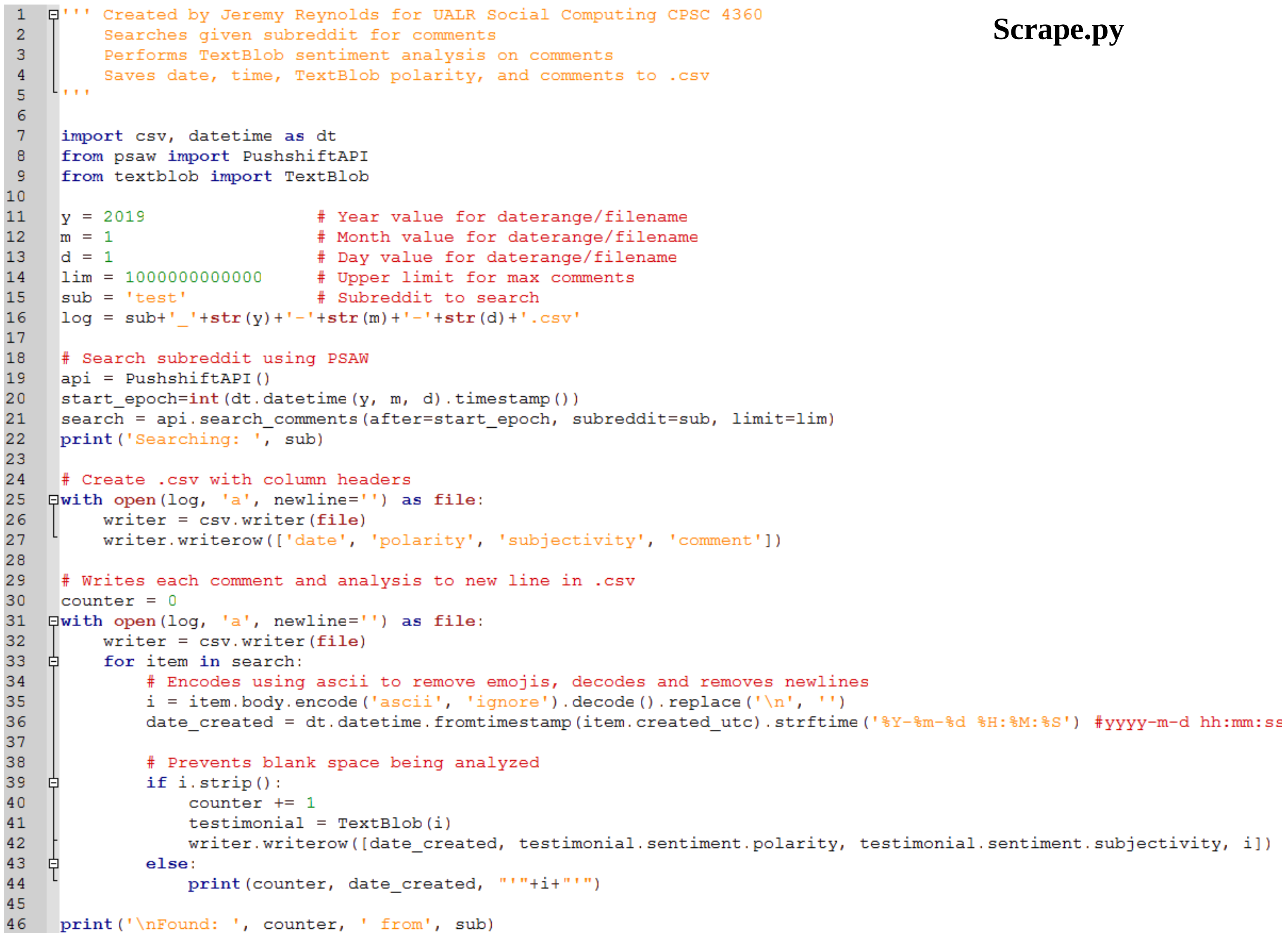
This project was designed to perform sentiment analysis on various gaming subreddits and compare their sentiments. The Python language was heavily used to collect and analyze the data. The PSAW package was used to connect to Reddit’s Pushshift API to collect the data, while both TextBlob and VADER were used to perform the sentiment analysis. Ultimately I succeeded in comparing the subreddits sentimentality and it lined up with what I subjectively predicted. This project has several areas in which it can be expanded upon for future work. Software that performs sentiment analysis could also be improved, perhaps by using tailored lexicons for the specific type of text it will be analyzing. Overall this was a very interesting project. It strengthened my programming skills and was sufficiently difficult to challenge me.

References

1. Python 3 https://www.python.org/download/releases/3.0/
2. PRAW https://praw.readthedocs.io/en/latest/
3. PSAW https://github.com/dmarx/psaw
4. TextBlob https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis
5. VADER https://github.com/cjhutto/vaderSentiment
6. Teststat https://pypi.org/project/textstat/

Code

Scrape.py



```python
''' Created by Jeremy Reynolds for UALR Social Computing CPSC 4360
    Searches given subreddit for comments
    Performs TextBlob sentiment analysis on comments
    Saves date, time, TextBlob polarity, and comments to .csv
'''

import csv, datetime as dt
from psaw import PushshiftAPI
from textblob import TextBlob

y = 2019                    # Year value for daterange/filename
m = 1                       # Month value for daterange/filename
d = 1                       # Day value for daterange/filename
lim = 1000000000000         # Upper limit for max comments
sub = 'test'                # Subreddit to search
log = sub+'_'+str(y)+'-'+str(m)+'-'+str(d)+'.csv'

# Search subreddit using PSAW
api = PushshiftAPI()
start_epoch=int(dt.datetime(y, m, d).timestamp())
search = api.search_comments(after=start_epoch, subreddit=sub, limit=lim)
print('Searching: ', sub)

# Create .csv with column headers
with open(log, 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['date', 'polarity', 'subjectivity', 'comment'])

# Writes each comment and analysis to new line in .csv
counter = 0
with open(log, 'a', newline='') as file:
    writer = csv.writer(file)
    for item in search:
        # Encodes using ascii to remove emojis, decodes and removes newlines
        i = item.body.encode('ascii', 'ignore').decode().replace('\n', '')
        date_created = dt.datetime.fromtimestamp(item.created_utc).strftime('%Y-%m-%d %H:%M:%S') #yyyy-m-d hh:mm:ss

        # Prevents blank space being analyzed
        if i.strip():
            counter += 1
            testimonial = TextBlob(i)
            writer.writerow([date_created, testimonial.sentiment.polarity, testimonial.sentiment.subjectivity, i])
        else:
            print(counter, date_created, "'"+i+"'")

print('\nFound: ', counter, ' from', sub)
```

Sentiment.py

```python
''' Created by Jeremy Reynolds for UALR Social Computing CPSC 4360
    Reads the comments produced from scrape.py and
    performs VADER sentiment analysis.

    Expects .csv with comments in fourth column.
    Outputs mean value of TextBlob and VADER values.
'''

import csv, glob, numpy
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

tb = []  # TextBlob Mean Sentiment
vd = []  # Vader Mean Sentiment
#log = 'test_2019-1-1.csv'

for log in glob.glob('*.csv'):
    print('searching', log[:-13])

    reader = csv.reader(open(log))
    analyzer = SentimentIntensityAnalyzer()
    for item in reader:
        vs = analyzer.polarity_scores(item[3])
        vd.append(vs['compound'])
        if item[1] != '0.0' and len(item[3]) > 10:
            tb.append(item[1])
    tb = [float(i) for i in tb[1:]]
    print('TB: ', numpy.mean(tb), log[:-13])
    print('VD: ', numpy.mean(vd), log[:-13])
```

Frequency.py

```python
''' Created by Jeremy Reynolds for UALR Social Computing CPSC 4360
    Searches each comment for the 'query' and totals it's
    frequency for each date.

    Expects .csv with date in first column and comments in forth column.
    Outputs the frequency of 'query' for each date.
'''

import csv, re

count = 0       # Initial frequency count
date1 = '0'     # Initialize first date to be 0 (for logic)
date2 = '0'     # Initialize second date to be 0 (for logic)
query = 'good'  # The word to search each comment for
log = 'test_2019-1-1.csv'
print('Searching: ', query, ' from', log[:-13])

# If the current date is greater/equal than the next date, find 'query' and iterate count
# If the current date is less than the next date, reset and counter and continue
reader = csv.reader(open(log))
for item in reader:
    date1 = item[0][0:10]
    if date1 >= date2:
        comment = [item[3].split()]
        for words in comment:
            for word in words:
                word = re.sub(r'[^a-z0-9]','',word.lower())
                if word == query:
                    count += 1
    else:
        print(date2, count)
        count = 0
    date2 = item[0][0:10]
print('Finished: ', query, ' from', log[:-13])
```

Size.py

```python
''' Created by Jeremy Reynolds for UALR Social Computing CPSC 4360
    Searches for all .csv in a directory and totals
    their size in MB and counts the number of lines (comments).

    Expects directory with .csv files.
    Outputs the total size of all files.
'''

import glob, os

print('Printing file sizes and number of lines:\n')
for log in glob.glob('*.csv'):
    size = os.path.getsize(log)/(1024*1024.0) # Converts from bytes to megabytes
    with open(log) as f:
        lines = sum(1 for line in f)
    print(log[:-13].ljust(20), str(round(size,2)).ljust(6), 'MB\t', lines)
```