

Reddit Karen Shaming Analysis Report

Andrew McGrail, Brian Bell, Raymond Cosgrove

CS 4624

Multimedia, Hypertext, and Information Access

Virginia Tech

Blacksburg, VA 24061

8 December 2021

Instructor: Dr. Edward A. Fox

Client: Dr. Florian Zach

Table of Contents

Table of Contents.....	2
List of Figures.....	4
List of Tables.....	5
1. Executive Summary.....	6
2. Introduction.....	6
2.1 Objective.....	6
2.2 Client.....	7
2.3 Background.....	7
2.4 Motivation.....	7
2.5 Organization of Report.....	8
3. Requirements.....	8
3.1 Data Deliverables.....	8
3.2 Software Deliverables.....	9
4. Design.....	9
4.1 Overview.....	9
4.2 Database Creation & Web Scraping.....	10
4.3 Natural Language Processing.....	10
5. Implementation.....	11
5.1 Overview.....	11
5.2 Data Collection.....	11
5.3 Natural Language Processing.....	11
6. Testing.....	13
7. User Manual.....	15
7.1 User Manual Introduction.....	15
7.2 Program Arguments.....	15
7.3 Input File Specifications.....	15
8. Developer Manual.....	25
8.1 Developer Manual Introduction.....	25
8.2 File Summaries.....	26
8.3 File Developer Guides.....	26

8.4 Virtual Environments.....	30
9. Lessons Learned.....	31
9.1 Timeline and Schedule.....	31
9.2 Challenges.....	33
9.3 Takeaways.....	34
9.4 Future Work.....	36
10. Acknowledgements.....	38
11. References.....	39

List of Figures

1. Google Trends Karen Visualization.....	8
2. Architecture Diagram	9
3. UberEats Download and Graph Example	18
4. UberEats graph_options num_comments Example	20
5. Multi-subreddit Download and Graph Example	22
6. NLP Text File Output Example	23
7. NLP HTML Visualization Output Example	24
8. Timeline Waterfall Diagram.....	30

List of Tables

1 Detailed Timeline Table 30

1. Executive Summary

The Reddit Shaming Karen's Tool is designed to allow the user to discover potentially hidden data trends with a given data set in relation to a specified keyword. Our program is given a specified data set which is obtained using the Reddit Pushshift API [2], parses and passes the data to our data analysis script, and finally outputs the requested values onto a series of graphs. These values may be anything which relates to the Reddit post which had contained the previously given keyword. This then will act as a simplified connector between keyword usage trends and the user.

Our client, Dr. Florian Zach, had us design our program to suit his research needs which had the following restrictions set in place. The provided keyword, as our project title suggests, was to be set as Karen throughout the entire process. Additionally, our sourced subreddit data was only to be obtained from three groups which were as follows. Group 1: Delivery services, Group 2: Food employees in general and firm specific, and Group 3: Restaurants. Thirdly, our program was to only look at data within a set time period during the start of the Covid-10 Pandemic. Lastly, we were required to implement some form of topic modeling to group the posts and comments appropriately. These restrictions were given to allow our client to understand how the pandemic affected the usage of Karen over a time period spanning many months.

2. Introduction

2.1 Objective

The main goal of this project was to gain an initial understanding of online shaming behavior and also discover potential trends in the usages of the given keyword. As mentioned earlier, Karen will be the associated keyword, although the project will support Dr. Zach with measuring data trends with any keywords he may require. Reddit will act as our sole source of information. Reddit is a social media platform or forum, where unlike other platforms such as Facebook, users gather in given communities (known as subreddits), that discuss whatever the topic of that community is. For example, there are subreddits dedicated to discussing cats, music, and lots of other popular topics [6]. Our initial design will analyze a set size of subreddits which will be explained in greater detail in the Design section. In technical terms, the primary task of this project was to create a Python program that scrapes both current and historical Reddit data for specific keyword analysis using the PushShift API[3]. This would provide an avenue to

understanding both current and past online shaming behavior. This API is designed to provide programmers with enhanced functionality and search capabilities when analyzing Reddit post data. The scraped data is then converted into the CSV and JSON formats, and parsed for all relevant information. This parsed information is used to create various graphs and charts which show keyword use trends over a given period of time. For this specific project, we will be analyzing the use of the keyword Karen 18 months before and after the start of the pandemic in March 2020.

2.2 Client

Our client, Dr. Florian Zach, is an Assistant Professor within the Howard Feiertag Department of Hospitality and Tourism Management [5]. His Research Areas include Innovation, Interorganizational relationships & networks, and Technology. Our project relates directly to his research interests and is designed to aid in both current and future research.

2.3 Background

Online shaming behavior has become much more common due to the widespread adoption of online social media platforms such as Instagram and also large online forums such as Reddit. As a result, there have been various words which have been adapted to mean and represent something completely different than they originally did. This project focuses on one specific term which has shown an increase in popularity over the last decade.

Karen is defined as, "...a pejorative slang term for an obnoxious, angry, entitled, and often racist middle-aged white woman who uses her privilege to get her way or police other people's behaviors [2]." This term has become associated with various physical features and personality traits such as having a blonde bob haircut, rudely asking to speak to managers, and attempting to police other people's behaviors. Unfortunately, there have also been instances of racially motivated actions becoming associated with the word as well.

2.4 Motivation

The motivation for this project stems from a discovery by our client Florian Zach, who is discussed above. He discovered that during 2020, the first year of the Covid-19 pandemic, there was an increase in the number of "Karen" occurrences (discussed in Section 2.4 above). This is illustrated in Figure 1. Our client is interested in why this

increase occurred and what the ramifications might be. Useful inferences can be made from the analysis of this phenomenon such as how “Karen” occurrences can be prevented, the main causes of “Karen” events, and more.

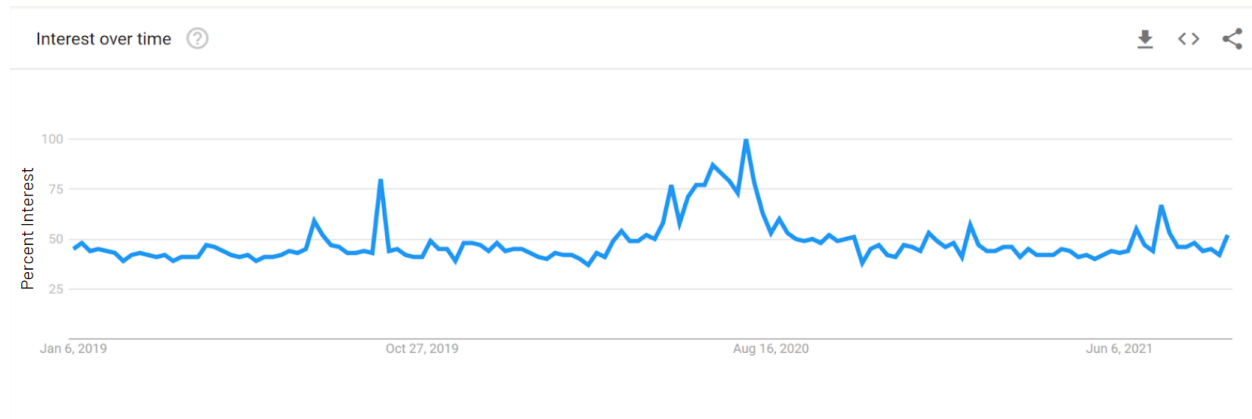


Figure 1: Google Trends Karen Visualization [4]

2.5 Organization of Report

This report contains a summary of the project as well as background information on our topic and client. It continues with Section 3 which describes the requirements of our project. This includes a detailed list of our project deliverables. Following is Section 4, which relates directly to the design of our project. This section includes our approach, motivation, and the technical strategy of our implementation. Section 5 builds upon the previous section by showcasing our implementation of the program. Section 6 then describes the testing of our program. The following two Sections, 7 and 8, act as guides for both the user and developer, respectively. Next, Section 9 describes what we learned during the project as well as possible future plans. Lastly, Sections 10 and 11 provide acknowledgements to our client, professor, and the various resources used.

3. Requirements

Our project had a set list of requirements that was requested by our client, Dr. Zach. This section will share the two subcategories of requirements which consist of our data & software deliverables.

3.1 Data Deliverables

1. Identification of quantifiable Reddit data.
2. Quantitative data from historical Reddit posts

- a. Comment and score outliers
 - b. Upvote data (per subreddit per month)
 - c. Keyword data (per subreddit per month)
 - d. Media data
 - e. Additional metadata
3. Qualitative data
 - a. Keyword analysis (25 words per subreddit per month)

3.2 Software Deliverables

The software deliverables include:

Programs to collect requested data and analyze it

- a. Python script to parse and process user input files
- b. Python script to collect historical reddit data
- c. Python script to analyze collected data
- d. Python script to perform NLP topic modeling on collected data

4. Design

4.1 Overview

The design of our project has a few different components. First, there is the parser which reads in the input file and controls the logical flow of the program based on the user's requests. There are three other components of our project that can be run based on the input file. The data collector component makes requests to both the PushShift API and the Reddit API in order to collect the historical post data specified by the user. The data analyzer component then uses this collected historical data to make some statistical analyses such as upvote trends over time. Lastly, the natural language processing component uses the historical data to complete keyword frequency analysis and topic modeling to determine trends in the titles and self texts of posts. A visualization of this architecture can be seen in Figure 2.

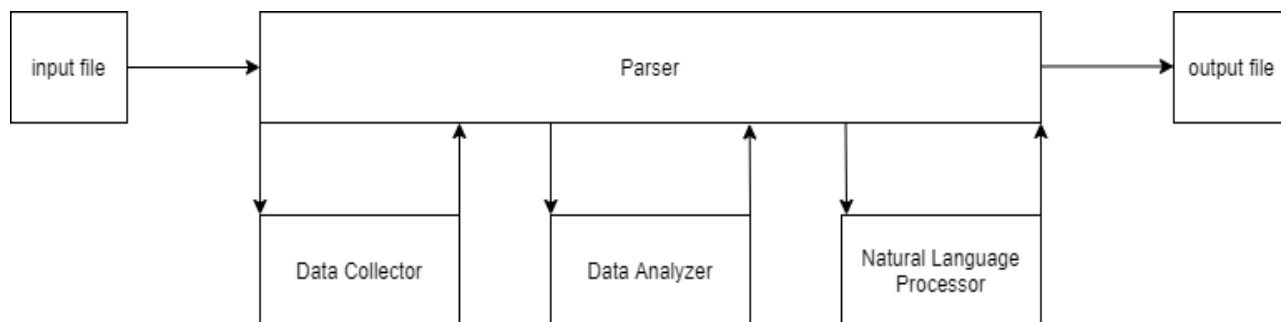


Figure 2: Project Architecture

4.2 Database Creation & Web Scraping

There is no explicit database for this project. Instead, a collection of JSON objects separated by subreddit, keyword, and month are used to store information from Reddit for access later, both for users and the graphing portion of the program. The same data is also stored in CSV format under the data/csv directory. No explicit web scraping is done by us, as Reddit data is accessed through a repository.

4.3 Natural Language Processing

The design of the natural language processing component of this project consisted of three major subcomponents, discussed below.

4.3.1 Data Preprocessing

This phase of natural language processing will help us clean our data and get it into a format that is best suited for appropriate algorithms. This preprocessing includes tasks such as stop word removal, stemming and lemmatization, and removal of non-alphabetic characters. We create bag-of-words structures for each document to better format the data for keyword frequency analyses. The bag of words model is a simplified representation of the text in a document. The grammar and order of words are disregarded, but the frequency of words is respected [11].

4.3.2 Keyword Frequency Analysis

In this phase of the natural language processing component, the bag of words models that were created are used to create a dictionary of all the unique words contained in the text data being analyzed. Once this dictionary has been created, the frequency of each unique

word can be determined. This is stored in a structure called a corpus that maps a word's unique identifier to its frequency.

4.3.3 Topic Modeling

This last section of the natural language processing code uses the corpus so Latent Dirichlet Allocation code can cluster and model topics within the provided text data [15]. This section of the code handles creating visualizations that better show the trends in these topic clusters.

5. Implementation

5.1 Overview

This section delves into the exact implementation of our application. Please refer to Section 4 (Design) to see an explanation of each of these processes.

5.2 Data Collection

The collection of data in this project is done using PMAW (Python Multithreaded API Wrapper) in conjunction with PRAW (Python Reddit API Wrapper), and requests a Python library for accessing and downloading webpages. Through these, Reddit data is downloaded locally through an iterative, month by month approach. For a given keyword and subreddit and month, a request is made to PMAW, the information is downloaded, and this process is repeated until all the lists of months are exhausted.

5.3 Natural Language Processing

There are a series of steps that are required when processing English text. Below were the steps taken to ensure that our processing was done correctly.

5.3.1 Text Normalization

All text was converted to their lowercase equivalents within the alphabet. This is a very popular preprocessing step and has been noted to be important for many NLP specific applications [10]. In the project, this is handled by the Python library gensim which includes a simple preprocessor [15].

5.3.2 Noise Removal

Noise removal consists of removing all non-alphanumeric characters, also known as special characters [10]. These characters added no notable context for our algorithm to understand, therefore acted as noise. This was handled using the `simple_preprocessor` object included in the `gensim` Python package [15].

5.3.3 Stop Word Removal

In addition to noise removal and text normalization, stop words were also removed from the text. Stop words are words that occur in high frequency and have very little meaning. In English, this includes words such as “a”, “an”, “the”, and “it”. Removing these words is a common preprocessing step used for natural language processing as it allows algorithms to focus on the most relevant words in the text [8]. The Python library `gensim` includes a list of common English stop words. In the stop word removal portion of the program, all the words in `gensim`'s list are removed as well as all words with a length of less than three characters.

5.3.4 Stemming & Lemmatization

Lemmatization is an advanced version of stemming and provides a solution to present inflectional forms and derivationally related forms of various words [9]. For an example, let us consider the English words “bike”, “bikes”, “bike’s” and “bikes’”. These words would all map to “bike”. The lemmatization of words in the data allows us to reduce the complexity of our dictionary/corpus and focus on the meaning of words rather than their variations. The stemming and lemmatization aspects of our program utilize the Python library `NLTK` to convert words to their root form [14].

5.3.5 Keyword Frequency Analysis

Keyword frequency analysis refers to calculating the frequency of relevant terms within document text. Once the above mentioned preprocessing steps are performed, our code uses `gensim` in order to create a dictionary containing every unique word in the proved text mapped to a unique ID. This dictionary is then converted into a corpus which maps each word's unique ID to its corresponding frequency within the text data. It is important that this step occurs after preprocessing has been performed otherwise the resulting dictionaries and frequencies will be inaccurate.

5.3.6 Topic Modeling

Once the corpus of the text data has been generated, it can be used with Latent Dirichlet Allocation code [15]. This code analyzes the frequency of specific words that occur

together often. Through this analysis, the code is able to cluster predicted groups of words that represent topics discussed. This project utilizes the Python package gensim in order to create a Latent Dirichlet Model and predict clusters within the documents. Initially, our implementation attempted to predict 10 topics per month. However, this was found to be too many topics due to the sparsity of data in some monthly files. The project now attempts to only predict 5 topics per month. This was easily changed as a parameter during model creation.

5.3.7 Topic Visualization

After generating the unsupervised Latent Dirichlet Model, the Python package pyLDAvis is used to generate visualizations of the predicted clusters [16]. Our implementation saves these interactive visualizations to html files so they can be accessed after the program runs. This is a feature built into the pyLDAvis library.

6. Testing

As shown in Figure 2 (Solution Architecture), the architecture of our project was composed of four main parts. We have one component, the parser, which handles the logical flow of our program and makes use of the other three components. Since we developed the project in these small parts, we were able to first unit test the individual components before adding them into the bigger overall architecture. Then, components were added into the larger architecture one at a time and integration tests were completed after each addition.

6.1 Data Collection Testing

In regards to the data collection component, testing helped us discover an error early on that was resulting in duplicate data files. Testing was conducted manually by making data collection requests and then checking the contents of the files to ensure data was collected correctly and not duplicated. In our current implementation, we have functionality that collects archived Reddit data, then makes requests to Reddit's API in order to validate and refresh this data. This ensures high quality data is collected.

6.2 Data Analysis Testing

For the data analysis component, testing was done manually with small sample sizes of data in order to check for accuracy. Since this section is responsible for calculating statistics, we were able to manually calculate the expected values and confirm that the program achieves the same statistics. The graphing component of this was tested in a

similar manner by confirming that the generated graphs accurately display the provided data.

6.3 Topic Modeling Testing

When developing and testing the natural language processing component, small sample data sets were used initially in order to ensure proper functionality. These small sample data sets allowed manual calculations to be performed in calculating keyword frequencies. Additionally, the data sets were constructed in a manner where the topic clusters could be predicted by a human, and then confirmed by LDA. Once the component was accurate in analyzing the sample datasets, Reddit data was inserted into the algorithms. We started by putting low numbers of posts into the algorithm to confirm they were being processed accurately. One issue we came across was trying to calculate too many clusters when low data was present. We adjusted parameters of the model to account for this, and were able to successfully scale up to larger amounts of data.

6.4 Integration Testing

After each of the components were tested and working properly individually, we began adding them one by one into the larger architecture. We first added the data collection component to our full architecture. Once we added it, we were able to use a variety of different input files to test that integration was successful. We would manually make the requests for data with the data collector component, then request the same data using the parser, and confirm that the data collected with both methods was accurate. Next, we added in the data analysis component. We followed the same methodology of how we tested the first integration: calculate manually and make sure results are the same. We also tested different combinations of both data collection and data analysis commands in the input files. Lastly, we integrated the natural language processing component into the parser. While testing different combinations of the three commands, we found an issue where natural language processing wouldn't complete successfully if graphing commands were also requested. This was due to the fact that the displayed graphs pause program execution. Thanks to testing, we were able to catch this error early and figure out a simple fix. Once this was fixed, testing continued until we were pleased with the integration of all four components.

7. User Manual

7.1 User Manual Introduction

This user manual should provide a deep understanding of how to use the program, and give a general idea of how certain features might be working behind the scenes. The manual has two main sections. The program arguments section describes the different ways our program can be executed. The input file specifications section describes how to construct an input file for our program to acquire your required results.

7.2 Program Arguments

There are several arguments that can be supplied to the program (with one being required, the parser file to be used). This help can be seen by running the program with the `-h` command:

usage: parser [-h] [--force-download] [--image-download] input_filename

positional arguments:

input_filename **Name of the input file.**

optional arguments:

-h, --help **show this help message and exit**

--force-download, -f **Flag to forcefully redownload all files.**

--image-download, -i **Flag to download images from the URL of reddit posts.**

Both of the optional arguments are fairly straightforward. Force-download will forcefully redownload all files that are already downloaded locally (potentially with new information, and overwriting whatever it was before, if it did already exist). Image-download downloads the image of Reddit image-posts to the images directory inside the data directory.

7.3 Input File Specifications

This subsection discusses the different ways an input file can be constructed.

7.3.1 Input File Commands

As for the input file itself, there are several notable commands:

download <subreddit> <keyword> <start_date> <end_date>

- Downloads data from a given subreddit that fit the given keyword, between a given start and end date into the data directory

graph <subreddit> <keyword> <start_date> <end_date>

- Graphs data already downloaded locally from a given subreddit on a given keyword, between a given start and end date

nlp <subreddit> <keyword> <start_date> <end_date>

- Generates keyword frequency analyses and topic modeling visualizations for each month with posts from provided subreddits containing the provided keyword within specified date range.

graph_options <options>

- The options for the graph

<comment>

- For placing comments in the parser file, note that the space between # and the <comment> is required.

Commands (Clarification)

Most of these commands are fairly straightforward, but here are some clarifications on what is expected of individual arguments. It should be noted that all arguments are required unless otherwise stated, and can be separated by any number of spaces or tabs:

<subreddit> should be the name of a subreddit.

Ex: ubereats, but NOT r/ubereats

<keyword> the keyword to be used in the results

Ex: karen, or alternatively “karen bad”, as double quotes can be used to search for keywords with spaces. Do NOT do Karen bad without the double quotes, or the behavior is undefined.

<start_date> the date to start searching for. Should be in the format yyyy-mm-dd.

<end_date> the date to stop searching for. Should be in the format yyyy-mm-dd.

7.3.2 Graph Options

A very specific format is required for <options>, one that will hopefully be changed in future iterations of the program:

```
graph_options "<value>:<display type>, <value>:<display type>, ..."
```

All options should be in the above format of a value and display type separated by a colon, with those two values delimited by a comma for the next pairs. **Note that the double quotes are required.**

<VALUE>

There are several options for <value>, of which will be listed a few here:

posts: Number of overall posts

num_comments: Number of overall comments

over_18: Whether a post is only for those over 18

only_media: Whether a post has only media and no self-text

score: The number of upvotes minus the number of downvotes the post has

upvote_ratio: The ratio of upvotes to downvotes

There are several more values that can be used, and it is recommended checking the developer's manual if you are looking for something more specific, as a somewhat more comprehensive guide will be there.

<DISPLAY TYPE>

There are currently three options for <display type>, with one option being a specific exception and the other two being more general:

len: Retrieves the number of something, has a specific exception to work with 'posts', but also works on a few other values such as 'subreddit_subscribers'. See the developer manual for reasoning on why.

sum: Retrieves the sum of that value across all posts in a given month, such as finding the total sum of comments.

per: Calculates the percentage of all posts that have a specific quality. For example, "over_18:per" calculates the percentage of posts that have the over_18 value. Here is an example of how to format graph options if you want the number of posts, the number of comments, and the collective score for each month:

```
graph_options "posts:len, num_comments:sum, score:sum" (note, the double quotes are required)
```

Here is an example of how to format graph options if you, for example, just want to see the score change over time:

```
graph_options "score:sum" (again, double quotes required)
```

7.3.3 Larger Examples

Here is an example of what you might use as your input file if you wanted to download all of the data from r/ubereats over the course of about two years, as well as the resulting graph:

```
# command subreddit keyword start_date end_date  
download UberEATS karen 2019-09-01 2021-07-01
```

And what if you wanted to graph all this data? Specifically the number of posts, comments, and score? (Note, as mentioned earlier, having the download command in the input file does **not** redownload files that have already been downloaded, so having it in an input file is perfectly fine. Having only the graph command will also work as long as you already have the file downloaded).

```
graph_options "posts:len, num_comments:sum, score:sum"  
# command subreddit keyword start_date end_date  
  
download UberEATS karen 2019-09-01 2021-07-01  
graph UberEATS karen 2019-09-01 2021-07-01
```

r/UberEATS

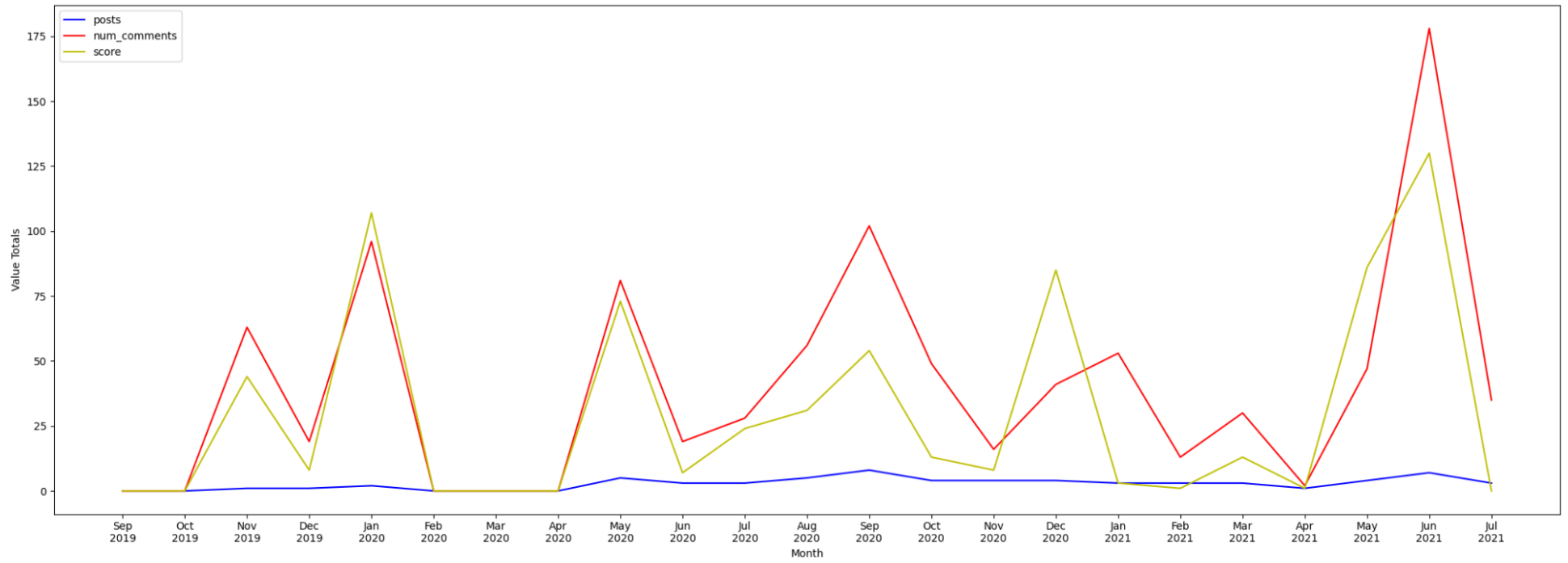


Figure 3: UberEats Download and Graph Example

But what if a user would like to graph just the number of comments and posts from ubereats over that time?

graph_options “posts:len, num_comments:sum”

command subreddit keyword start_date end_date

download UberEATS karen 2019-09-01 2021-07-01

graph UberEATS karen 2019-09-01 2021-07-01

r/UberEATS

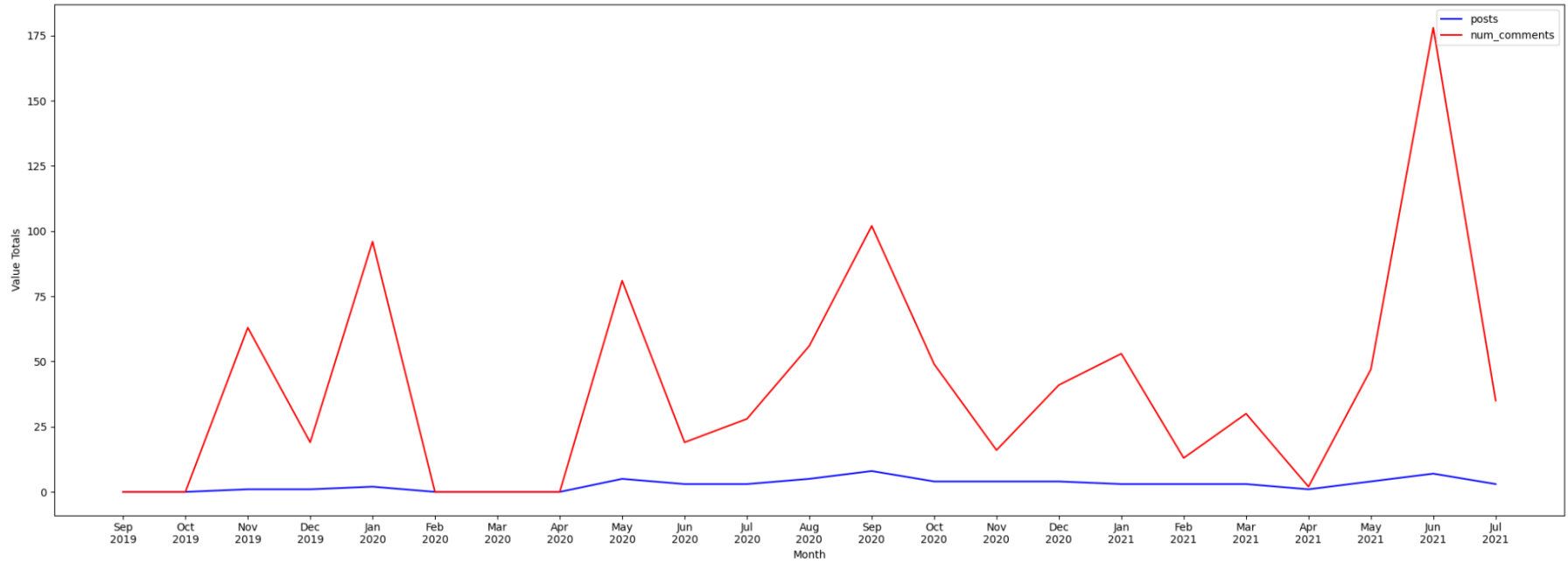


Figure 4: UberEats graph_options num_comments Example

That's cool, but what if a user would like to graph the data from two different subreddits?

graph_options "posts:len"

# command	subreddit	keyword	start_date	end_date
download	UberEATS	karen	2019-09-01	2021-07-01
download	doordash	karen	2019-09-01	2021-07-01
graph	UberEATS	karen	2019-09-01	2021-07-01
graph	doordash	karen	2019-09-01	2021-07-01

As a note, this will graph the data from r/UberEATS and r/doordash collectively, and there is currently no way to differentiate between the two unless you simply make two separate graphs with two separate input files.

r/UberEATS, r/doordash

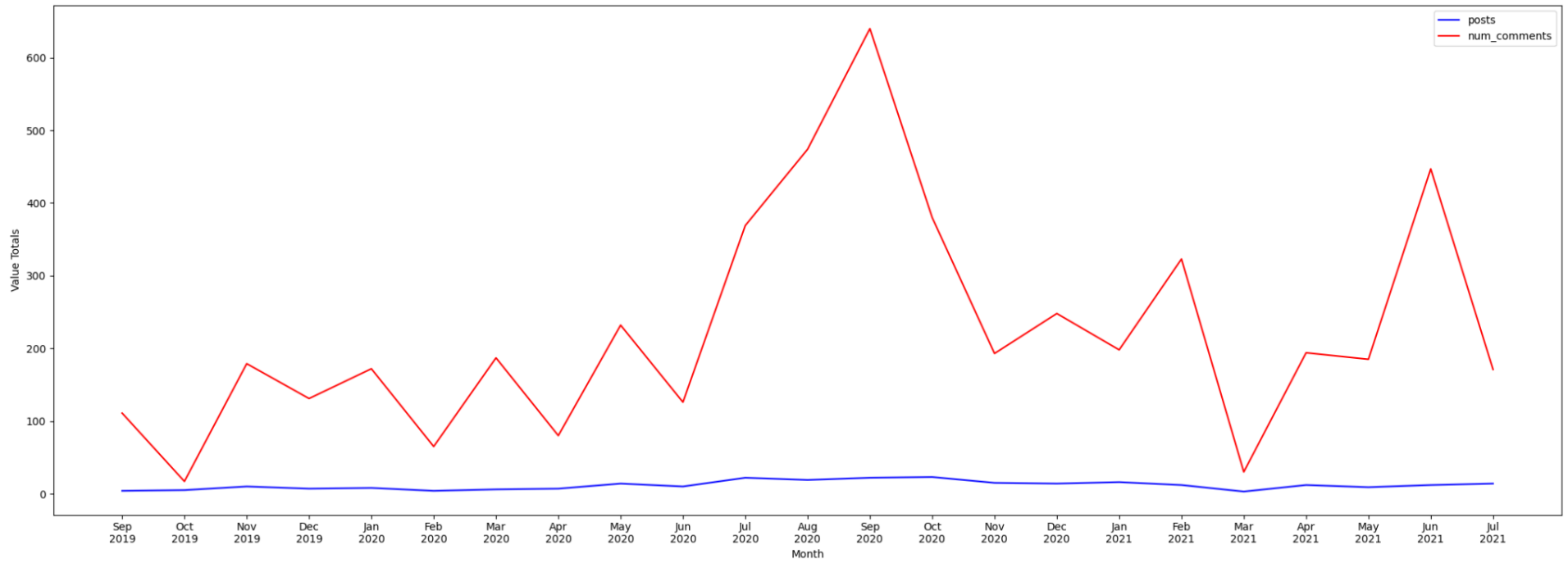


Figure 5: Multi-subreddit Download and Graph Example

Lastly, we can demonstrate the NLP capabilities, including keyword frequency analysis and topic modeling, of our program with the use of the nlp command as shown below.

```
graph_options "posts:len, num_comments:sum"  
# command subreddit keyword start_date end_date  
  
download UberEATS karen 2019-09-01 2021-07-01  
nlp UberEATS karen 2019-09-01 2021-07-01
```

This input file will produce outputs with contents similar to the following.

1. A text file for each month with data describing keyword frequency analysis as well as modeled topics.
2. An HTML file for each month with data containing an interactive visualization of the topics modeled.

The text file that is generated contains the top 25 most frequently used keywords after preprocessing in the text data specified. It also shows the frequency of each of these words. Below this keyword frequency analysis section is the topic modeling section of the file. This section shows the weighted representation of the generated topic clusters. Each word in the cluster is shown along with a coefficient weight that specifies its importance to the specific topic cluster.

The HTML file generated contains an interactive visualization of the aforementioned topic clusters. Within the visualization, the user is able to mouse over the various shown clusters to display the frequency of the keywords contained within them. Additionally, the clusters are visualized in two dimensions to show similarity between them. This can be useful when analyzing data to determine if the text contains many disjoint clusters or many similar ones.

Both of these outputs are shown in figures 6 and 7 below.

```

Natural_Language_Processing > data > json > UberEATS_karen_2020-05_nlp.txt
1 Keyword Frequency Analysis (Top 25 Most Common Keywords):
2 ('order', 10)
3 ('karen', 8)
4 ('mcdonald', 5)
5 ('go', 3)
6 ('give', 2)
7 ('thumb', 2)
8 ('take', 2)
9 ('call', 2)
10 ('wish', 2)
11 ('say', 2)
12 ('wasn', 2)
13 ('post', 1)
14 ('show', 1)
15 ('spit', 1)
16 ('abl', 1)
17 ('explain', 1)
18 ('meal', 1)
19 ('mean', 1)
20 ('second', 1)
21 ('wait', 1)
22 ('comin', 0)
23 ('ranti', 0)
24 ('warn', 0)
25 ('fight', 0)
26 ('remov', 0)
27
28 Topics Modeled From Monthly Posts (as shown in html visualization):
29 Topic: 0
30 Words: 0.033*"happen" + 0.024*"mcdonald" + 0.024*"say" + 0.024*"order" + 0.024*"deliveri" + 0.019*"time" + 0.019*"custom" + 0.015*"restaur" + 0.0
31
32 Topic: 1
33 Words: 0.019*"mcdonald" + 0.019*"karen" + 0.019*"take" + 0.019*"order" + 0.019*"counter" + 0.019*"hour" + 0.019*"aint" + 0.019*"mcfish" + 0.019*"
34

```

Figure 6: NLP Text File Output Example

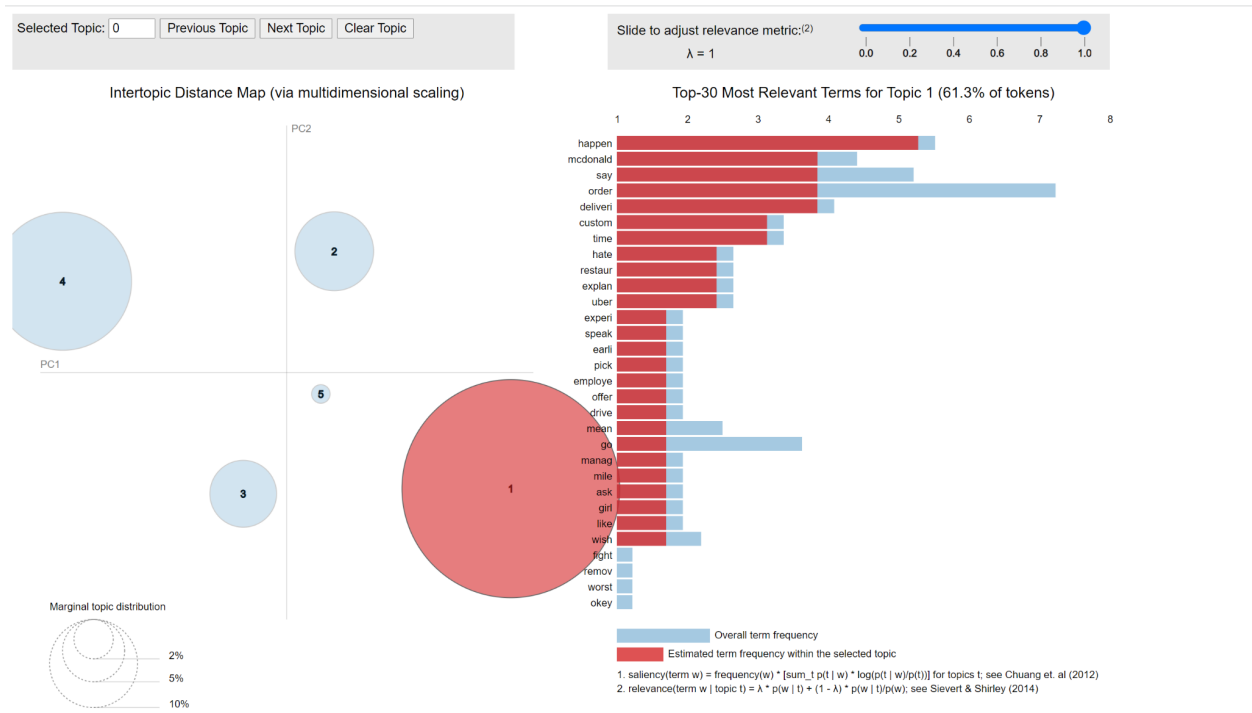


Figure 7: NLP HTML Visualization Output Example

As a note, there are also several example input files included with the project that may be of some help.

8. Developer Manual

8.1 Developer Manual Introduction

This developer manual should be seen mostly as an extension of the user manual; refer to here for reasoning on implementation and more information. The user manual how to use the program, and gives a general idea of how certain features might be working behind the scenes. This developer guide provides a deeper look into the respective files and their inner workings. This should help future teams determine where to look if they want to add or change a feature.

8.2 File Summaries

Most of the files and directories are fairly self explanatory, and is explained in its own mini section:

data/ - A directory that holds all downloaded information, with subdirectories such as `csv/`, `json/`, and `images/`.

data_analysis.py - Contains code related to analyzing the data already downloaded. To be more specific, this could be called `graphing.py`.

data_collector.py - Where data is actually collected from PRAW (Python Reddit API Wrapper) and PMAW (Pushshift Multithreaded API Wrapper)

misc.py - Is mostly for holding random values, classes, and methods that need to be used in multiple files, and to provide a level of indirection.

parser_files/ - While not necessary, should be used for holding input files for the parser

parser.py - Script that should be executed in order to parse and run the input files (with either `./parser.py` or `python3 parser.py` and followed by the name of the input file).

praw.ini - This file needs to be created by you, or supplied by someone else as it allows PMAW [3] enrichment to take place. A guide detailing how to create it from scratch is here: <https://new.pythonforengineers.com/blog/build-a-reddit-bot-part-1/>

README.md - It is the README.

topic_modeling.py - This is a script that contains functions for data preprocessing, topic modeling, and keyword frequency analysis. It is utilized by parser.py when an input file specifies the nlp command.

8.3 File Developer Guides

8.3.1 Data_analysis.py

This is for grouping and graphing data from files. In future work, improvements should be made, such as how colors for lines are used; at the moment it is just a handmade list.

Also, if you are wondering why there are newlines in the title and for the x values, it is because without them the table does not format correctly, and words overlap. Try removing them to see what happens if you are curious.

8.3.2 Data_collector.py

This is for collecting data using PMAW [3], and in two cases, the python requests library. Most of the code is about cleaning up and storing the data correctly, as PMAW has a few intricacies which will hopefully be fixed at some point. For example, submissions aren't actually okay to convert to JSON out of the box, several of their values have to be explicitly converted to a string before JSON allows it to be serialized. It should be noted that some of the values are of interest to users and are NOT just data objects being stored in the posts (which is just flatout strange, and done by the PMAW authors), as the 'subreddit' and 'author' value of submissions both have to be fixed according to what was just said, as they are vital information in posts.

The get_subscribers method exists because PMAW does not have a way to choose what values are enriched by PRAW, meaning that when data is updated using newer Reddit information, it updates everything it can, up to and including the subscriber count of subreddits. That means that without a second request to retrieve JUST the unenriched

data, there is no way to get the accurate value of subscribers. To fix this, a call is made directly to pushshift using requests.

There is also a method to download images, as that was requested by the client for a potential new project. It is not heavily tested or implemented in a particularly robust manner, but it works from the testing that has been done, albeit only with .jpg and .png pictures (though this can be easily expanded).

8.3.3 Misc.py

This is the file where everything that needed a layer of indirection went, such as methods that needed to be used across multiple files, or did not make sense to go in one place in particular. Such as a method for generating a list of files over a certain timeframe with a keyword and subreddit (and is useful for both referencing files already made or to be made).

8.3.4 Parser.py

This file is the ‘main’ file for the program, and should be used to execute input files. It does nothing but run other files, which should be called directly or indirectly through the given input file.

8.3.5 Praw.ini

This can be built fairly easily as said above, either with a given praw.ini file containing a reddit bot, or by making one yourself using a guide (which will not be detailed here).

8.3.6 Topic_modeling.py

This file is responsible for the keyword frequency analysis and natural language processing capabilities of the program. The file uses the libraries NLTK, gensim, and pyLDAvis [14][15][16]. Nltk and gensim are used for preprocessing the data. This includes stemming and “lemmatization” of the words, as well as removing stopwords [14][15]. Gensim is then used to calculate the frequency of the words and apply Latent Dirichlet Allocation to the data [15]. The model can be used to produce predictions for some N number of topics. The results can then be visualized by using the pyLDAvis library. In this implementation, if a file specified with the nlp command contains data, all of the above mentioned operations are performed on the data within the file. This results in two files. First, a file is created that contains the top N (default 25) most frequent keywords within the data as well as information about the topics that were modeled

within the data. The second file created is a static web page containing an interactive visualization of the modeled topics. The file can be opened in any standard web browser.

The processes followed in the implementation of this file are fairly standard and much documentation can be found online about each of the procedures followed [14][15]. Additionally, the libraries utilized and mentioned above have their own respective APIs documented well online. They are all relatively common and also have answers to common questions on StackOverflow and other websites.

This is a very simplistic topic modeling script that utilizes Latent Dirichlet Allocation. We chose to develop a simple topic modeling script because the professor requested only beginners topic modeling for the scope of this project. Additionally, this allowed us to focus our efforts on different portions of the project such as data analysis and data collection. This file provides a solid preprocessing function and framework for loading in posts to be processed.

8.3.7 /Data Directory

This directory holds all of the downloaded information. The current convention is that different file types go in their own folder, though this is not the case for images. Files inside the csv and json directory should have the format:

`SUBREDDIT_KEYWORD_YYYY-DD.EXT`

This format is REQUIRED for most of the functionality of this program. Beware if this is not adhered to, as there are already issues with capitalized/uncapitalized subreddit names causing issues.

Remember that no JSON file is really ever empty, although CSV files can be completely empty. Accounting for this is important, especially if random data is eventually stored in the JSON files (like, for example, subreddit subscribers).

8.4 Virtual Environments

Our team found that when working on this project, the easiest way to manage the necessary Python packages was to make use of virtual environments. We found the best way to develop on the project was to create a virtual environment and install the necessary packages within the environment. This resolved dependency issues by making sure all the necessary packages for the project were installed whenever development

work was being done. This is further discussed below. It was also beneficial to use the Anaconda, a data science platform built for Python. Anaconda can be downloaded [here](#). Anaconda will come with pandas and numpy which are python libraries used for data analysis in this project [12][13]. It also will come with virtual environment capabilities. Documentation on working with virtual environments in Anaconda can be found [here](#). Once a virtual environment has been created, you can use the following command within your environment to install the packages necessary to develop this project.

```
pip install gensim nltk pyLDAvis praw pmaw matplotlib
```

9. Lessons Learned

9.1 Timeline and Schedule

The project timeline was adjusted throughout the entire development process. This was mainly due to newly discovered roadblocks that were accounted for. Below is the final timeline which consists of our milestones, deadlines, and our clients needs.

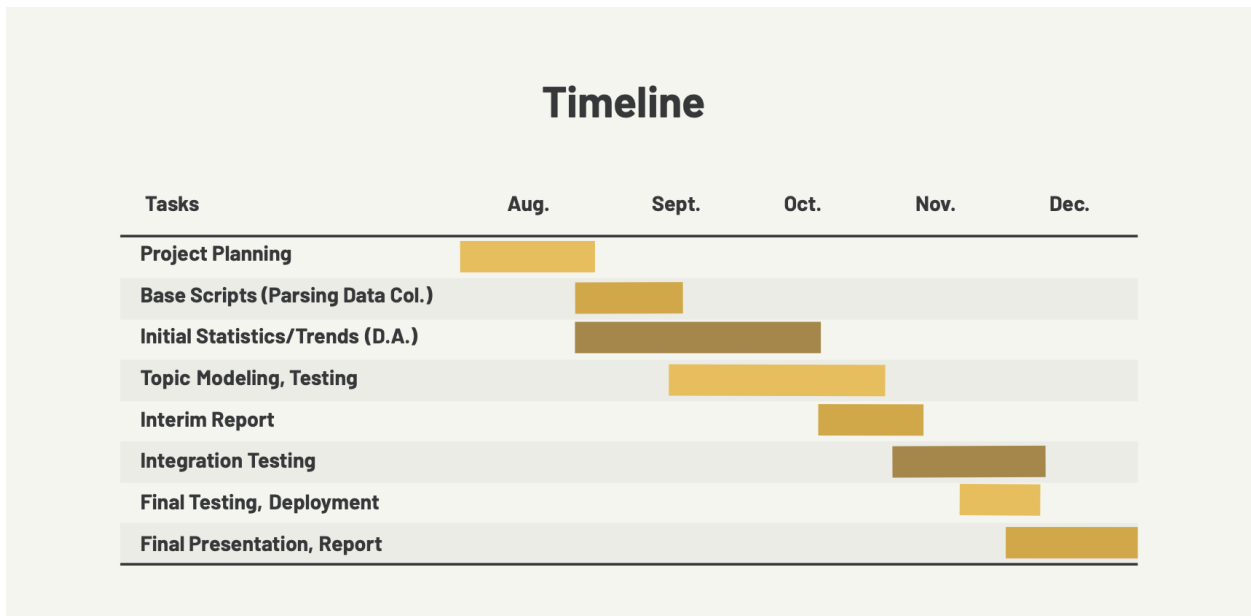


Figure 8: Timeline Waterfall

7 September 2021	<ul style="list-style-type: none"> ● Completed initial timeline ● Finalize project page ● Data Collection Prototype
14 September 2021	<ul style="list-style-type: none"> ● Begin building parsing component
16 September 2021	<ul style="list-style-type: none"> ● Presentation 1 - Project Overview
28 September 2021	<ul style="list-style-type: none"> ● Initial data pulldown ● Begin data analysis component
12 October 2021	<ul style="list-style-type: none"> ● Complete data analysis prototype
26 October 2021	<ul style="list-style-type: none"> ● Begin testing ● Feedback/Improvement Cycle
9 November 2021	<ul style="list-style-type: none"> ● Begin natural language processing component
14 November 2021	<ul style="list-style-type: none"> ● Interim Presentation/Report
15 November 2021	<ul style="list-style-type: none"> ● Complete NLP preprocessing functionality
30 November 2021	<ul style="list-style-type: none"> ● Complete NLP Topic Modeling
8 December 2021	<ul style="list-style-type: none"> ● Final Presentation and Report

Table 1 - Detailed Timeline Table

The architecture of our project allowed us to divide our tasks among our three members which allowed for concurrent progress on various objectives. Our tasks can be divided into eight main phases shown in Figure 6. A more granular version of the timeline containing specific tasks can also be viewed in Table 1. The phases of development are briefly described below.

9.1.1 Project Planning

In this phase of the project, we met with our client to discuss project requirements and expectations. We worked together as a team to research topics related to our project and develop a project plan, timeline, and architecture.

9.1.2 Base Scripts

This phase of the project focused on creating the Python scripts we needed first in order to continue work on the project. Our first course of action was to create the data collection and parser components of our architecture.

9.1.3 Initial Statistics

During this phase, our team focused our work on the data analysis component of our project. This included loading in collected data, performing statistical analysis on it, and generating visualizations to detect trends within the data.

9.1.4 Topic Modeling

While in the topic modeling phase of the timeline, the team's focus was on completing the natural language processing aspects of the project. This included keyword frequency analysis as well as modeling topics within the collected data.

9.1.5 Interim Report

In the middle of our timeline, the interim report phase shows the period of time where the team focused efforts on completing a draft outline of this final report, while also attempting to continue work on the project.

9.1.6 Integration Testing

This period on the timeline was when the team worked together to piece each of the working components together in order to complete our larger architecture. Extensive testing was done to confirm the integration of our components was successful. More can be read about this topic in Section 7.

9.1.7 Deployment

During the deployment phase in our timeline we focused on creating our finished code as well as creating a frozen windows executable to hand off to our client for easy use.

9.1.8 Final Deliverables

The last phase of our timeline describes the period of time where our team focused on finishing our final presentation, our final report, and provided our client with his requested deliverables.

9.2 Challenges

Some of our members had no prior experience developing applications such as this which did result in some unforeseen challenges. We have taken note of them below in their own subsections.

9.2.1 Inaccurate archived data

Our data collection process did not account for the constant changing of Reddit post metadata values in real times. For example, a post may have three upvotes, eight comments, and one keyword occurrence during the archiving phase. These values would permanently stay the same and ignore any changes that were a result of another upvote or comment. This only became apparent when our team noticed inaccuracies in the output file and our referenced post. We addressed this complication by simply refreshing the data set before running our tests. We were able to do this by utilizing the PRAW Python package and Reddit API. Reddit post IDs are sent in batches to the Reddit API via praw. The API then responds with the updated metadata values, which PRAW subsequently updates. This resulted in data that is accurate to the time that it is requested.

9.2.2 Obtaining subscriber counts

Our client requested that we obtain accurate subscriber counts for our clients requested subreddits. This came as an unforeseen challenge as we had difficulties obtaining the correct counts on a month to month basis. The reason we had difficulties was actually due to the inaccurate archived data issue (see Section 9.2.1). The data actually had the correct subscriber count at the time the post was made, which is what we wanted. However, when using PRAW and the Reddit API to update and refresh this data, there was no functionality to select which fields we wanted to update. This resulted in all of the subscriber count fields getting updated to the number of subreddit subscribers at the time of retrieval rather than their historic counterparts. We were eventually able to acquire historical subscriber counts, but it results in the program essentially having to request each post twice. We download the updated and refreshed data, then for each file containing data, we request the archived version of that post and add the historic subscriber count to the updated data.

9.2.3 Team member sickness

During the entirety of the semester, each of our team members unfortunately became sick at one point or another which made it difficult to complete certain tasks. When an individual was sick, the other two members notified our client and tried their best to complete the required work.

9.2.4 Duplicate Data

Early in the development of the data collector portion of the project, we had an issue where certain files contained data duplicated from the previous month. Luckily, this was

detected early on due to our testing regiment. This issue stemmed from a custom developed implementation of requesting archived Reddit posts from the PushShift API. This issue was eventually resolved by using an existing implementation for this purpose from the Python PMAW and PRAW libraries.

9.3 Takeaways

Throughout the semester, our team was able to identify some important takeaways from our work.

9.3.1 Set a Realistic Scope

Firstly, we realized that there are almost an infinite number of ways to analyze Reddit posts. This resulted in us making an important realization of the importance of scope when planning a project. It is important to not get in over one's head when planning out work over an extended period of time. As shown in Section 9.2, unexpected challenges will be faced and cause a delay in the completion of work. Additionally, when working on a new project, it is likely that there will be a large overhead time period at the beginning where not a lot of work is completed due to the initial learning curve. This is important to take into account when determining scope. If too large of a scope is chosen, this overhead period can creep too far into the project timeline, resulting in a time crunch where lower quality deliverables are produced, and likely not all of the promised deliverables are produced. One positive note from this takeaway is that by choosing a manageable scope for our project, we are allowing for many more projects with similar goals to be carried out by future students, possibly using the framework we developed this semester. Some of these are discussed in Section 9.4 below.

9.3.2 Do Not Reinvent the Wheel

Another takeaway our team discovered was that sometimes, it is best not to reinvent the wheel. Originally, when developing the data collection component of our project, we developed our own logic to request archived Reddit data within certain time ranges. This resulted in the duplicate data issue discussed in Section 9.3.4. We eventually changed our code to use an existing solution for the same task. Once we updated our code with this solution, it began working well. Oftentimes, if a solution is popular, there is a reason for that popularity. This was the case for this project. If we had decided to use this existing solution from the start rather than creating our own, we would have saved development, debugging, and testing time and possibly been able to implement more features at the end of our project timeline. This is an important lesson to learn and a good takeaway to make from this project.

9.3.3 Know Your Users

Additionally, our team learned that applications should be designed to best suit researchers such as Dr. Zach. Researchers tend to focus on the numbers and trends in the data and have a preference for comparisons done by graphs. Our Reddit data analysis application allowed us to see the value in the comparisons of current numbers and also the power of data collection and analysis. Since most researchers performing this kind of analysis are not formally trained as computer scientists or programmers, it is important to design an application that caters toward this type of user. This takeaway leads directly into our future work ideas, specifically Section 9.4.

9.4 Future Work

Throughout the semester, multiple potential avenues for future work were discussed. The future work for this project will be directed by Dr. Zach. Feedback and notes will be given to future teams according to his evaluation of the software. We will discuss our ideas for future plans in respective subsections below.

9.4.1 Improved Graphic Features

First, improved graphing features including higher granularity graphs would be a useful feature for trend identification. Graphing options are already supported, but having the ability to create per subreddit graphs for specific timeframes could be a useful feature down the line. This should be a relatively uncomplicated task for a future group of students to complete.

9.4.2 Image Analysis

A second future extension discussed was image analysis. Our group implemented functionality to download all of the pictures associated with a downloaded Reddit post. A future group could be responsible for using either a third party ocr or their own computer vision program to extract text from these downloaded images. Once the text is extracted from the images, natural language processing could be utilized to interpret some sort of meaning from the text. This would be especially useful in images that are screenshots of text message conversations. An interesting and difficult problem to tackle would be dethrading the conversation and keeping track of who is sending what text. Furthermore, an even harder task would be figuring out which message is in response to which. This could allow for flows of conversations to be studied and see how people interact when

talking about the specified topic. This would be an interesting human interaction study and we think many inferences could be drawn from such a study.

9.4.3 Expand to different industries

Another future extension is to use our framework to explore phenomena in different industries. We built an extensible framework that can be used to perform data analysis and natural language processing on any set of subreddits and any keyword. It will be interesting to see what applications Dr. Florian Zach and possibly other clients will find for this framework. We think this framework will allow for many other research questions and even capstone projects for future groups of students.

9.4.4 Graphic User Interface

One possible project that could help with extending this tool to other industries is the creation of a graphics user interface or front end for our code. If a front end were to be created for our framework, it would allow those who are less technically inclined to be able to utilize our tool for their own projects. This future user interface could allow users to select time ranges with calendars, click on specific subreddits, view example graphs so they know what their final visualizations would look like, and more. This could definitely be a semester-long project for a group of students, especially if the client and students go through multiple iterations of what the front end would look like and what features it would include.

9.4.5 Sentiment Analysis

A final extension could be adding more natural language processing features to the framework. Firstly, a different form of topic modeling could be used and different forms of Natural Language Processing could be performed. One particularly interesting feature would be the tracking of sentiment over specified time periods. We think a lot of research questions could be answered if a group of students were to study how the posts' positive and negative connotations changed over time. There may have been a large spike in the number of Karen posts, but what is the sentiment of that spike? Maybe there was an influx of posts because of a positive event. Right now, we have no way of knowing.

10. Acknowledgements

We would like to acknowledge our client Dr. Florian Zach from the Howard Feiertag Department of Hospitality and Tourism Management alongside our professor in our Multimedia/Hypertext course Dr. Edward A. Fox. We have provided each individual's contact information below.

Dr. Florian Zach
Howard Feiertag Department of Hospitality and Tourism Management
florian@vt.edu

Dr. Edward A. Fox
Department of Computer Science
fox@vt.edu

11. References

- [1] F. Zach, *F2021Project-RedditShamingKaren*, Aug-2021. [Online]. Available: <https://canvas.vt.edu/courses/136035/pages/f2021project-redditshamingkaren> [Accessed: 09-Sep-2021].
- [2] Dictionary.com, “slang Karen” *Dictionary*. [Online]. Available: <https://www.dictionary.com/e/slang/karen/> [Accessed: 09-Sept-2021].
- [3] Pushshift, “Pushshift API Documentation” *Pushshift*. [Online]. Available: <https://github.com/pushshift/api> [Accessed: 09-Sept-2021].
- [4] Google, “Google Trends” *Google*. [Online]. Available: <https://trends.google.com/trends/explore?geo=US&q=karen> [Accessed: 09-Sept-2021].
- [5] Florian Zach, “Florian Zach” *Virginia Tech Pamplin College of Business*. [Online]. Available: <https://htm.pamplin.vt.edu/directory/zach.html> [Accessed: 30-Nov-2021]
- [6] RedditInc, “Dive into anything,” *Reddit*. [Online]. Available: <https://www.redditinc.com/>. [Accessed: 08-Dec-2021].
- [7] Anaconda, “Managing environments,” [Online]. Available: <https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html> . [Accessed: 08-Dec-2021].
- [8] O’Reilly Media, “Stop word removal” *O’Reilly*. [Online]. Available: <https://www.oreilly.com/library/view/natural-language-processing/9781787285101/ch02s07.html> [Accessed: 08-Dec-2021]
- [9] C. D. Manning, P. Raghavan, and H. Schütze, *Stemming and lemmatization*, 2008. [Online]. Available: <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html> [Accessed: 13-Dec-2021].

- [10] Dinesh Yadav, "NLP: Building Text Cleanup and PreProcessing Pipeline" [Online]. Available: <https://towardsdatascience.com/nlp-building-text-cleanup-and-preprocessing-pipeline-eba4095245a0> [Accessed: 08-Dec-2021]
- [11] Sivic, Josef (April 2009). "Efficient visual search of videos cast as text retrieval". IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 31, NO. 4. IEEE. pp. 591–605.
- [12] Pandas, "Pandas," *pandas*. [Online]. Available: <https://pandas.pydata.org/> [Accessed: 13-Dec-2021].
- [13] NumPy, "About Us," *NumPy*, 2021. [Online]. Available: <https://numpy.org/about/> [Accessed: 13-Dec-2021].
- [14] NLTK Team, *NLTK*, 2021. [Online]. Available: <https://www.nltk.org/> [Accessed: 13-Dec-2021].
- [15] R. Řehůřek, "Gensim: Topic modelling for humans," *Radim Řehůřek: Machine learning consulting*, 30-Aug-2021. [Online]. Available: <https://radimrehurek.com/gensim/> [Accessed: 13-Dec-2021].
- [16] B. Mabey, "Pyldavis," *PyPI*, 2021. [Online]. Available: <https://pypi.org/project/pyLDavis/> [Accessed: 13-Dec-2021].