Spring 5-2023

Analyzing the Overturning of Roe vs Wade on Twitter using Natural Language Processing Techniques

Gabriela Pinto
Chapman University, pinto@chapman.edu

Recommended Citation

G. Pinto, "Analyzing the overturning of Roe vs Wade on Twitter using natural language processing techniques," M. S. thesis, Chapman University, Orange, CA, 2023. https://doi.org/10.36837/chapman.000485

Analyzing the Overturning of Roe vs Wade on Twitter using Natural Language Processing Techniques

A Thesis by

Gabriela Pinto

Chapman University

Orange, CA

Schmid College of Science and Technology

Submitted in partial fulfillment of the requirements for the degree of

Masters of Science in Computational and Data Sciences

May 2023

The thesis of Gabriela Pinto is approved.

Erik Linstead Digitally signed by Erik Linstead Date: 2023.05.12 15:04:00 -07'00'

Erik Linstead, Ph.D., Chair

Elizabeth Stevens Digitally signed by Elizabeth Stevens Date: 2023.05.12 11:50:41 -07'00'

Elizabeth Stevens, Ph.D.

Elia Eiroa Lledo Digitally signed by Elia Eiroa Lledo Date: 2023.05.12 16:06:01 -07'00'

Elia Erioa Lledo, Ph.D.

April 2023

ACKNOWLEDGMENTS

I want to thank my family, friends, and colleagues for their support during my studies and research career thus far. I would also like to especially thank Dr. Erik Linstead for the opportunity to join the Machine Learning and Affiliated Technologies (MLAT) lab, where I felt inspired to pursue a career in research. I will always be grateful for the mentorship and experiences I gained during the past five years. With their support, I will pursue my Ph.D. in Computer Science at the University of Southern California.

VITA

EDUCATION

Master of Science in Computational and Data Science	**2023**
Chapman University	*Orange, CA*
Bachelor of Science in Computer Science	**2021**
Chapman University	*Orange, CA*

INDUSTRY EXPERIENCE

Masters Intern	**March 2023 -August 2018**
Research Computing Team	
Pacific Northwest National Laboratory	*Richland, WA*
Product Development Engineer Intern	**August 2021- January 2022**
NAND Team	
Intel	*Folsom, CA*
Technology Analyst Intern	**June 2021-August 2021**
Barclays	*Whippany, NJ*

RESEARCH EXPERIENCE

Graduate Research Assistant	**February 2022–May 2023**
Chapman University	*Orange, CA*
REU Program Participant	**March 2021–September 2021**
Worcester Polytechnic Institute	*Worcester, MA*
Undergraduate Research Assistant	**March 2020–February 2022**
Chapman University	*Orange, CA*

TEACHING EXPERIENCE

Graduate Teaching Assistant	**August 2022–May 2023**
Chapman University	*Orange, CA*

LIST OF PUBLICATIONS

Lledo-Eiroa, E., Ali Hamza, R., Pinto, G., Anderson, J., Linstead, E. **February 2023**
Large-Scale Identification and Analysis of Factors Impacting Simple Bug Resolution Times in Open Source Software Repositories
Applied Sciences

Atchison, A., Pinto, G., Woodward A., Stevens E., Dixon D., Linstead, E. **2022**
An Application of Document Embeddings to Indentifying Challenging Behaviors in Autism Spectrum Disorder from Clinical
2022 21st IEEE International Conference on Machine Learning and Applications

Ali Hamza A., Pinto, G., Lawrie, E., Linstead, E. **A Large-Scale Sentiment Analysis of Tweets Pertaining to the 2020 US Presidential Election** **June 2022**
Journal of Big Data

Atchison, A., Pinto, G., Woodward A., Stevens E., Dixon D., Linstead, E. **2021**
Classifying Challenging Behaviors in Autism Spectrum Disorder with Word Embeddings
20th IEEE International Conferences on Machine Learning and Applications

ABSTRACT

Analyzing the Overturning of Roe vs Wade on Twitter using Natural Language Processing Techniques

by Gabriela Pinto

In 1973, the historic U.S. Supreme Court (SCOTUS) case of Roe vs. Wade provided the constitutional right to abortion. However, on May 2, 2022, Politico magazine leaked the draft opinion on the Dobbs v. Jackson Women's Health Organization. The leak generated a surge of users to post their opinion on the case that would eliminate abortion as a constitutional right. Then, on June 24, 2022, SCOTUS overturned Roe vs. Wade. In this thesis, we aim to investigate the public opinion and reaction towards the overturning of Roe vs. Wade. We collected 20,640,166 tweets using Twitter API for Academic Research and an open-sourced dataset published during two periods. The first period was a week before Politico magazine leaked the SCOTUS decision and the week after. The second period was a week before and over a week after the overturning of Roe vs. Wade. Using natural language processing techniques, including sentiment analysis, emotion recognition, topic modeling, and bi-grams, we could develop insight into public opinion based on the posted tweets. Our research investigates if there is a change in sentiment over time, a change in the emotion expressed within the text over time, and which topics are most common within the collection of tweets. The results demonstrate a significant increase on the day of the Politico leak, which showed that most of the tweets published on that day expressed a positive sentiment. However, in the weeks before and after the overturning of Roe v. Wade, we witness a decrease from the beginning of the period up to the day of the overturn. Regarding emotion recognition, there is a significant decrease in the proportion of tweets classified as expressing optimism. There's also an increase in the proportion of tweets expressing anger when comparing the day of the Politico leak and the day of the overturn. The topic model we applied to the tweets published on the day of the Politico leak revealed that states' rights and children were discussed. Using bigram of the most negative tweets, we witnessed gun control and healthcare as words that frequently occurred within the collection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

In 1973, the Supreme Court ruled in the case of Roe v. Wade that the U.S. Constitution protected women the right to have an abortion. The ruling from this monumental court case became the legal precedent that upheld the constitutional right to abortion for almost 50 years. On May 2, 2022, Politico published a leaked draft of the U.S. Supreme Court majority decision in Dobbs v. Jackson Women's Health Organization that would overturn the decision made in Roe v. Wade [17]. As the news of the Politico leak reached millions of Americans, many users voiced their opinions on Twitter. Then on June 24, 2022, the U.S. Supreme Court overturned Roe v. Wade which revoked the federal right to abortion. The consequence of the decision caused people to voice their opinions on Twitter, either in favor or against. A policy change as significant as the overturning of Roe v. Wade has caused an emergence of topics such as birth control and women's rights within these tweets. Thus, it is imperative to analyze these conversations as they are essential in understanding the experiences and perceptions of the public, as they would aid in future policymaking and decision-making.

The abundance of data from tweets could provide insight into the discourse related to Roe v Wade and the topics that submerge from these conversations. In this thesis, we utilize natural language processing techniques to analyze social media data. Previous research has investigated the sentiment on tweets related to political events such as the U.S. Presidential Election, the government interference during the COVID-19 pandemic, and government elections in other countries [30, 49, 11, 13, 4]. Similarly in [1], has demonstrated that policy change affects political opinion towards issues that expand beyond the core issue. It holds even greater importance as a previous study has proven that hate speech, cyberbullying, and online harassment campaigns have significantly increased [14, 25, 23]. Moreover, a recent study has quantitatively proven that the since the purchase of Twitter by Elon Musk, there has been a significant increase in the occurrence of hate speech and in the prevalence of bots on the social media platform [22]. A topic such as abortion that ignites heated political debate would cause an emergence of hate speech. The approach outlined in this thesis could be used in analyzing future Supreme Court decisions that have an equal effect or are even greater.

In this novel application of natural language processing, we collected 20,640,166 tweets and examined how users linguistically communicated their opinion on Twitter. We look to determine in this thesis public opinion through the application of natural language processing techniques, including sentiment analysis, emotion detection, and topic modeling. We analyzed how the sentiment had changed over time, what topics were heavily discussed and how they changed over time, and how the emotion had changed over time. We sought to investigate an existing or lack of change a week before the Supreme Court decision leaked on

Politico and a week after. In addition, investigate a shift a week before the official overturning of Roe vs. Wade and a week later. The techniques, as mentioned earlier, would quantitatively demonstrate the opinion of Twitter users and any existing change. We demonstrated this by answering the following research questions:

RQ1. Is there a significant change in sentiment from the time the decision leaked in Politico and when Roe vs. Wade was officially overturned?

RQ2. Is there a significant change in emotion from the time the decision leaked in Politico and when Roe vs. Wade was officially overturned?

RQ3. What topics were commonly mentioned within these conversation related to Roe vs. Wade?

The application of sentiment analysis, emotion recognition, and topic modeling has given a deeper insight into the conversation on abortion from both sides of the political spectrum. In the following pages, we will detail the data used in our study in Chapter 2, the application of natural language processing techniques in Chapter 3, the results obtained in Chapter 4, and their practical significance in Chapter 5. We then outline related work in Chapter 6, future work in Chapter 7 and conclusions in Chapter 8.

2 Data

In this thesis, we collected the 20,640,166 tweets from two sources, the Twitter API and a dataset published by Chang et al. [9]. The tweets were published between two time periods: April 25, 2022, to May 10, 2022, and the second period from June 17, 2022, to July 5, 2022. The data we scraped using the Full-Archive Search provided by the Twitter API for Academic Research allowed us to collect more tweets faster and grab historical or real-time data. To gain a complete analysis of the public opinion on the overturning of Roe v Wade, we included keywords that we used by individuals that indicate a neutral stance, an anti-abortion stance, and a pro-abortion stance. The keywords were categorized respectively as ”neutral,” ”pro-choice,” and ”pro-life,” and their associated keywords are listed in Table 1.

Category	**Keywords**
Neutral	roe v wade, roevswade, Roeverturned, abortion
Pro-choice	prochoice, abortionrights, abortionishealthcare, mybodymychoice, reclaimroe, EndRoeVWade, reproductiverights, plannedparenthood, AbortionRightsAreHumanRights
Pro-life	prolife, AbortionIsMurder, LifeIsAHumanRight, EndAbortion, chooselife

Table 1: Keywords used in the query and their corresponding category

As shown in the Listing 1, we implemented the keywords in our query. The data retrieved from the API call were the author's id, the geo object, the tweet's id, the time stamp of when the tweet was created, the time stamp of when the user started their account, the author's location published in their profile, if the author is verified, the number of tweets the author published, the author's number of followers, the author's description published on their profile, the author's username, the language the tweet was written in, the number of times the tweet has been retweeted, the number of replies to the tweet, the number of likes the tweet received, the number of times the tweet has been quoted, and which device was used to publish the tweet. The geo object corresponds to the location of where the Tweet was posted. Within the tweets retrieved from the Twitter API, most didn't return a geo object and, thus, were omitted in our analysis.

```
query_params = {'query': keyword,
                'start_time': start_date,
                'end_time': end_date,
                'max_results': max_results,
                'expansions': 'author_id,in_reply_to_user_id,geo.place_id',
                'tweet.fields': 'id,text,author_id,in_reply_to_user_id,geo,conversation_id,\
                created_at,lang,public_metrics,referenced_tweets,reply_settings,\
                source',
                'user.fields': 'id,name,username,created_at,description,public_metrics,verified,\
                location',
                'place.fields': 'full_name,id,country,country_code,geo,name,place_type',
                'next_token': {}}
```

Listing 1: Query used to scrape tweets from the Twitter API

Keyword	**Count**
roe v wade	481461
roevswade	367445
Roeverturned	77
prochoice	52734
abortionrights	52993
abortionishealthcare	55409
mybodymychoice	50198
abortion	3453951
reclaimroe	33
EndRoeVWade	215
ChooseLife	2226
reproductiverights	11726
plannedparenthood	10044
AbortionRightsAreHumanRights	53906
prolife	98867
AbortionIsMurder	5682
LifeIsAHumanRight	506
EndAbortion	599

Table 2: Keywords used in the query and their corresponding count

After gathering the data from the API, we removed duplicate tweets. The number of tweets and their corresponding keyword as shown in Table 2, giving a total of 4,697,965 tweets. In addition to grabbing the 4,697,965 from the Twitter API, we utilized the tweet IDs provided in the dataset from [9]. The tweets were gathered similarly to ours, using the Full-Archive Search provided by the Twitter Academic API. In their data collection, they used an extensive set of keywords that extend to different stances on abortion. The keywords detailed were implemented in their search query and collected tweets published from January 1, 2022, to January 6, 2023. Using the Tweet lookup endpoint and with the Twitter IDs from the files as input, we collected 15,982,314 tweets published from April 25, 2022, to May 10, 2022, and from June 17, 2022, to July 5, 2022. Combined with the data we scraped and the data we grabbed using the data from [9], we obtained 20,640,166 after removing tweets without a creation time and tweets that didn't contain a sufficient amount of data for analysis. It is worth noting that while collecting the data from [9], the response from the Tweet Lookup endpoint returned errors for a fraction of the tweet IDs published. When grabbing the information for the tweet, two errors were returned: "Not Found Error" and "Authorization Error." When the tweet was unavailable or deleted, it would return "Not Found Error." If the tweet was protected and we could view it using the API, it produced an "Authorization Error." While grabbing a user's information, the response would return a "Forbidden" error when the user has been suspended and a "Not Found Error" if the API could not find the user's data. In Table 3, we listed the number of tweets that returned errors categorized by the error type. This is particularly noteworthy as it is shown that there were many tweets related to abortion were deleted on the listed dates.

The distributions for the number of tweets and the corresponding date they were published on Twitter are shown in Figures 1 and 2. While the leak occurred on May 2, 2022, Politico published the article leaking the Supreme Court decision in the evening; Politico published an edit in the morning on May 3, 2022. News outlets published the decision by the time the correction was published, causing a surge of tweet publications.

Date	TweetNotFound	UnauthorizedTweet	UserNotFound	ForbiddenUser	Total Errors	Total
04-25	7708	718	125	206	8757	15752
04-26	26037	1711	200	272	28220	30816
04-27	26124	1636	319	304	28383	31431
04-28	12191	1264	135	156	13746	23589
04-29	41235	4462	331	391	46419	46247
04-30	35353	3204	253	403	39213	44053
05-01	22972	2315	227	321	25835	32157
05-02	7463	919	123	144	8649	15324
05-03	2334136	188501	9094	9309	2541040	2326676
05-04	48766	3074	337	343	52520	86244
05-05	761317	55368	3639	3576	823900	627008
05-06	836114	62398	5659	5598	909769	733511
05-07	365858	33625	2514	2675	404682	420589
05-08	202395	27191	2036	2239	233861	302868
05-09	220893	33798	513	440	255644	333499
05-10	70864	6817	510	554	78745	131983
06-17	10695	1354	140	234	12423	20936
06-18	10132	1284	134	190	11740	20569
06-19	12386	1714	127	182	14409	26258
06-20	36881	2645	317	570	40413	48259
06-21	17990	1720	154	234	20098	34484
06-22	12743	1718	145	239	14845	27955
06-23	91339	5268	570	694	97871	97678
06-24	2805225	288285	6808	5983	3106301	2883872
06-25	2699385	314446	8179	7293	3029303	2806603
06-26	1453895	170186	7655	7421	1639157	1316685
06-27	1260965	131549	8577	9904	1410995	1098659
06-28	98065	10585	885	963	110498	191595
06-29	130319	58409	4297	5302	198327	508043
06-30	227809	82532	6089	6962	323392	789867
07-01	410996	41521	2679	3206	458402	352040
07-02	48395	9209	442	495	58541	92831
07-03	414556	45012	2987	3555	466110	297943
07-04	31405	3679	368	463	35915	59204
07-05	39466	4627	390	453	44936	67316

Table 3: Number of errors when retrieving tweet from [9]

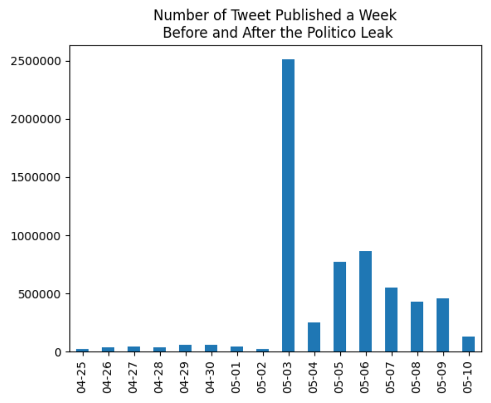

Figure 1: The Number of Tweets published a Week Before and After the Supreme Court Decision on Roe v Wade leaked on Politico

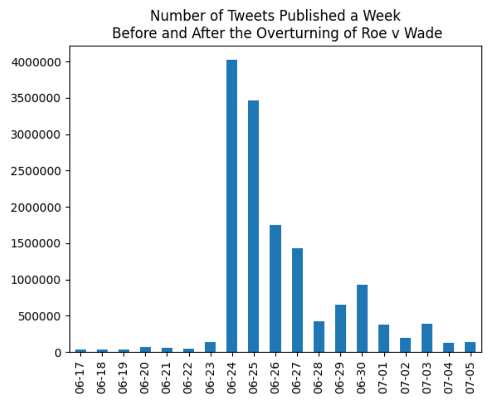

Figure 2: The Number of Tweets published a Week Before and After the Supreme Court Overturned Roe v Wade

3 Methods

To analyze the public opinion on the overturning of Roe v. Wade, we employed natural language processing techniques, including sentiment analysis using Valence Aware Dictionary for sEntiment Reasoner (VADER) and TweetEval's sentiment classification, TweetEval's emotion detection, and topic modeling using Gensim. In this paper, we implemented these techniques to obtain a quantitative representation of the public's reaction to the Supreme Court's decision on overturning Roe v. Wade and the topics arising from those conversations.

VADER is more efficient than other sentiment calculation approaches, such as the Textblob and Lexicon approach, because of its ability to handle various sentiment expressions. In addition to its sensibility of the plethora of sentiment expressions within a tweet that takes in the form of excessive use of exclamation marks, emoticons, and capitalization, it manages much of the processing in the backend so that it doesn't exclude any crucial characteristics. Other sentiment analysis approaches exclude vital data, such as excessive capitalization and punctuation, which typically indicates emotion. The quantitative representation of the sentiment expressed by the user could provide a general insight into how a large group of people reacted to a topical event. To gain a more in-depth analysis, we also implemented TweetEval's emotion recognition task on the collection of tweets. Emotion recognition provides a more detailed classification than sentiment analysis, which outputs either positive, negative, or neutral.

Emotion recognition through the TweetEval framework classifies each tweet as Anger, Joy, Sadness, or Optimism. In this study, we applied this classification as it aims to identify the complex emotions on an event that caused a plethora of responses to the overturning of Roe v. Wade. Unlike sentiment analysis, it ignores the complexity of emotions expressed in text as it only outputs negative, positive, and neutral. For instance, in a tweet that returns a negative sentiment, the emotion expressed could be fear or anger. While these emotions are technically 'negative,' they are significantly different in describing the user's emotional state. Emotion recognition provides a more in-depth classification that allows us to explore more complex emotions.

In addition to the sentiment and emotion classification, we implemented topic modeling through linear discriminant analysis (LDA) to extract the most prominent topics within online conversations. In this paper, we applied extensive data preprocessing and entered the cleaned data into an LDA model using Gensim. The topics outputted from the model will provide insight into other topics discussed in addition to the overturning of Roe v Wade.

Once these techniques were applied, we investigated whether there was an existing correlation between the sentiment and the time it was published, a correlation between the emotion detected and the time it was published, and the most prominent topics when the Supreme Court decision leaked versus when Roe v Wade was officially overturned. The following subsections provide a brief overview of Sentiment Analysis with VADER, Emotion Detection with TweetEval, and Topic Modeling.

3.1 Sentiment Analysis with VADER

Previous research integrated sentiment analysis to understand the public's perspective on a variety of topics, including COVID-19 booster vaccine shots [48], the stigma associated with monkeypox among LGBTQ+ community [15], and supplemental nutrition assistance program [10]. In this thesis, we apply sentiment analysis on the collected tweet using Valence Aware Dictionary for sEntiment Reasoning (VADER) [26]. While analyzing the sentiment on social media data, there are a series of challenges to consider. Social media text typically requires URLs, non-alphabetical characters, excessive punctuation, excessive application of capitalized letters, emoticons, slang, a limited amount of characters, degree modifiers (e.g., very, rather, sort of, kind of), and abbreviated language (e.g., LOL, FOMO). To account for the characteristics explicitly presented in a social media text, we utilized VADER to gain insight into the public's response to the Politico leak and the overturning of Roe vs. Wade.

A lexicon that has been applied to calculate the sentiment of social media data is the Linguistic Inquiry and Word Count (LIWC) [41]. LIWC, like VADER, is a dictionary-based approach to calculating the sentiment of a given text. It contains a set of words that express positive and negative sentiments. However, LIWC doesn't regard the challenges in calculating the sentiment expressed in social media text. Meanwhile, VADER considers the unique properties of social media data in its lexicon.

While there is a lexicon-based approach to calculating the sentiment of a given text, machine learning approaches have been developed to mitigate the intensive work of creating lexicons such as LIWC or VADER. Some machine learning approaches include Naive Bayes (NB) classifiers, Maximum Entropy, Support Vector Machines Classification models, and Support Vector Machine Regression models. However, machine learning approaches have their own set of challenges to consider. They typically require excessive amounts of validated training data, which are difficult to acquire. In addition, applying machine learning models to large amounts of social media data is computationally expensive. Meanwhile, VADER doesn't require any training data and doesn't require an extensive amount of computational power.

The features, consisting of words or emoticons, were applied a rating on a scale from -4 to 4. Where -4 indicates "extremely negative," 4 is "extremely positive," and 0 indicates neutral. For example, their lexicon contained a sad emoticon, ":(," which had a valence of -2.2, and the word "good" with the associated valence of 1.9. Because VADER is validated by humans, making its results more accurate than other lexicons, including Linguistic Inquiry Word Count (LIWC), General Inquirer (GI), Affective Norms for English Words (ANEW), and Hu-Lui04 lexicon.

For every tweet in our dataset, we applied the lexicon provided by VADER and its tools for computing the sentiment score using the vaderSentiment module [26]. The engine outputs four scores: compound, pos, neu, and neg. The pos (positive), neu (neutral), and neg (negative) scores represent the proportion of the text that expresses its respective sentiment, which all add up to 1. These metrics provide a method for analyzing how sentiment is conveyed in a given text. While this metric is helpful, it wasn't applied for this thesis as the compound score was deemed most appropriate for analyzing the overall sentiment of a given tweet as it considered the linguistic components seen in a social media text.

The compound score is computed with the tweet as input, and for each word in the tweet, the module grabs its associated valence scores. Once all the valence scores of each word in the input were obtained, the summed of the scores were calculated and adjusted according to a set of rules. The rules, which account for the sentiment intensity of the text, include the amount of punctuation (e.g., The view from here is amazing!!), use of ALL-CAPS (e.g., The view here is AMAZING!), degree modifiers (e.g., The view here is absolutely amazing), contrastive conjunction (e.g., The view here is great but there was so much traffic), and the tri-gram preceding a sentiment-laden lexical feature (e.g., The view here isn't really that amazing). Afterward, the score was normalized between -1 and 1, the most extreme negative and extreme positive, respectively. The compound score is the most helpful metric within this subfield of natural language processing, as previous literature uses the following thresholds for classifying texts as positive, neutral, and negative [39, 6]. A text is classified as expressing positive sentiment if the compound score is greater than or equal to 0.05; negative sentiment if the compound score is less than or equal to -0.05; neutral sentiment if the compound score is greater than -0.05 and less than 0.05. In Listing 2, I've provided a visual representation of the output for a given body of text that can also be found in [26].

```
Input:
VADER is VERY SMART, handsome, and FUNNY!!!

Output:
{'pos': 0.767, 'compound': 0.9342, 'neu': 0.233, 'neg': 0.0}
```

Listing 2: The Pos, Neg, Neu, and Compound Scores from a given body of text using VADER

3.2 Emotion Recognition

In addition to the sentiment analysis, we applied emotion recognition to our tweets collection to gain a deeper insight into the public's reaction to the Politico leak and the overturning of Roe v Wade. We used the TweetEval benchmark, which consists of seven tasks designed explicitly for English tweets [2]. The seven tasks were the following: emotion recognition, emoji prediction, irony detection, hate speech detection, offensive language identification, sentiment analysis, and stance detection. For this thesis, we applied the emotion recognition task [36] for our collection of tweets. The emotion recognition task takes in a tweet as input and outputs four emotions: Anger, Joy, Sadness, and Optimism. The task utilizes a dataset from [36], which included 11 emotions. In the integration into TweetEval, the four emotions were selected. Before inputting their own data set, the data preprocessing consisted of anonymizing user mentions, removing line breaks, and removing URL links. The language model chosen for constructing the TweetEval benchmark, which includes the emotion task, was RoBERTa [31] considering that its training data of 38GB of Reddit data exposed it to the language typically written on social media.

When inputting our dataset to TweetEval's emotion recognition task, we preprocessed each tweet by removing user mentions and URLs. Afterward, we downloaded the RoBERTa model and created an instance of the model. We applied the AutoTokenizer from Huggingface's Transformers to tokenize the text. The tokenized, or encoded text, contained two fields: input ids and attention mask. The input ids represented the tweets as numbers, and the attention mask defined which tokens provided a value. Once the text was tokenized, it was entered into our model. The model's output would generate the four scores for each emotion, which add up to one. The highest score indicated the emotion that was most expressed in the given text. An example is shown in Listing 3, where the highest ranking score is joy, indicating that the text is most likely expressing that emotion.

```
Input:
"Celebrating my promotion :)"


Output:
1) joy 0.9382
2) optimism 0.0362
3) anger 0.0145
4) sadness 0.0112
```

Listing 3: The Joy, Optimism, Anger, and Sadness scores from a given body of text using TweetEval

3.3 Topic Modeling

3.3.1 Dirichlet Distribution

The Dirichlet distribution is a multivariate generalization of beta distribution that is parameterized by a vector α of positive real numbers [29]. The application of this distribution is used in text mining techniques, which expands to the developmet of the Latent Dirichlet Allocation model and the Dirichlet process. The probability density function of the Dirichlet distribution is most conveniently written in the following way:

$p(q|\alpha) = \frac{\Gamma(\alpha_1+...+\alpha_J)}{\Gamma(\alpha_1)...\Gamma(\alpha_J)} \prod_{j=1}^{J} q_j^{\alpha_j-1} (q_j \geq 0; \sum_j q_j = 1)$

The application of this distributions is vital for the computations that occur when applying the Latent Dirichlet Allocation (LDA) model.

3.3.2 Poisson distribution

The Poisson distribution used in LDA models is a discrete distribution that measures the probability of a given number of events happening in a fixed period of time [21]. The probability density function is defined as the following: $f(x;u) = \frac{\mu^x e^{-\mu}}{x!} x = 0, 1, ...$, where x represents the discrete random variable.

3.3.3 Multinomial Distribution

Within the context of LDA, the Multinomial distribution serves as it conjugate prior. The Multinomial distribution provides the probability of any particular combination of numbers of success for the various categories possible [38]. The probability mass function is defined as the following:

$f(x_1, ..., x_k; n, p_1, ..., p_k) = \frac{\Gamma(\sum_i x_i+1)}{\Pi_i \Gamma(x_i+1)} \Pi_{i=1}^{k} p_i^{x_i}.$

3.3.4 Latent Dirichlet Allocation Model

To gain a more in-depth analysis of the tweets related to Roe vs. Wade, we applied topic modeling as a method of gathering hidden themes or topics from our large collection of tweets. A brief overview of the process is to extract each word in each document, or tweet, and assign it into one of K possible topics. K is a number of topics that we can decide to produce. In previous research that implements topic modeling on tweets [28], the standard that we know today was introduced by Blei et al. is Latent Dirichlet Allocation (LDA) [3]. In its essence, LDA clusters the words in a given collection into topics, which are distributions over words, while simultaneously classifying the texts as mixtures of topics. For the remainder of this section we will described the process of LDA described in [3].

A word is defined, within this thesis, as an item from a vocabulary indexed by $\{1,....,V\}$. A document is a sequence of N words denoted by $w = (w_1, w_2, ..., w_N)$. Meanwhile, a corpus is a collection of M documents denoted by $D = \{w_1, w_2, ..., w_M\}$.

The LDA is founded on the idea that documents are represented as a random mixture over latent topics–a collection of words–where each topic is a distribution over words. For each document,w, in a corpus D, LDA assumes a process that consists of three components. The first being choose $N \sim Poisson(\xi)$, second being choose $\theta \sim Dir(\alpha)$, and for each of the N word w_n it chooses a topic $z_n \sim Mulitnomial(\theta)$ and chooses a word w_n from $p(w_n|z_n, \beta)$, which is a multinomial probability conditioned on the topic z_n. The model makes two assumptions, the first assumption being the dimensionality k of the Dirichlet distribution and the dimensionality of the topic variable z is assumed known and fixed. And, the second being is the word probabilities are parameterized by a $k \times V$ matrix β where $\beta_{i,j} = p(w^j = 1|z^i = 1)$, which are treated as a fixed quantity, which will be estimates. N is independent of all other data generating variables (θ and z).

With these assumptions in mind, the making of LDA model considers a k-dimensional Dirichlet random variable θ takes in the values in the (k-1)-simplex and has the probability density on this simplex defined as $p(\theta|\alpha) = \frac{\Gamma(\Sigma_{i=1}^{k}\alpha_i)}{\Pi_{i=1}^{k}\Gamma(\alpha_i})\theta_1^{\alpha_1-1}...\theta_k^{\alpha_k-1}$,

where α represents a k-vector with component α_i is greater than 0 , where $\Gamma(x)$ is the Gamma function. The Dirichlet is a convenient distribution on the simplex. With the given α and β parameters, the joint distribution of a topic mixture θ, a set of N topic z, and a set of N words w is given by the following equation:

$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha)\Pi_{n=1}^{N}p(z_n|\theta)p(w_n|z_n, \beta)$,

where $p(z_n|\theta)$ is θ_i for the unique i such that $z_n^i = 1$. When the integration over θ is performed and summing over z, the marginal distribution of a document resulted in:

$p(w|\alpha, \beta) = \int p(\theta|\alpha)(\Pi_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n, \beta))d\theta$.

The product of the marginal probabilities of single documents results in the following probability of a corpus:

$p(D|\alpha, \beta) = \Pi_{d=1}^{M}\int p(\theta_d|\alpha)(\Pi_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn}, \beta))d\theta_d$.

Thus, we can consider LDA into three levels, where first the parameters α and β are corpus-level parameters, assumed to be sampled once in the process of generating a corpus. The variables θ_d are document-level variables, sampled once per document. And, the variables z_{dn} and w_{dn} are word-level variables and are sampled once for each word in each document. With the application of this model, documents can be associated with multiple topics. We applied the LDA model using the model provided by Gensim, a free open-source Python library, to our dataset [43].

3.3.5 Preprocessing

Before inputting our tweets into our model, we performed a series of text-cleaning procedures. First, we removed emojis in each tweet, removed the user mentions, and removed URLs. This was accomplished using the regex, re,emoji, string. Afterward, we tokenized each tweet using the spaCy module, which split the tweets into words. All words were then lowercase, and punctuation was removed. We also removed any words that were less than three characters long. Numbers and stop words were also extracted. Stop words are words that have little or no meaning within a sentence. The set of stop words were generated from the spaCy, wordcloud, and gensim modules.

Once tokenization was completed, the text was lemmatized, aiding the LDA model to produce better topics. The text was also stemmed, and any existing punctuation was removed. The lemmatized text was used to create a dictionary, where the words are mapped to their integer ids. Thus, any words in the third person are then changed to the first person; verbs in past and future tenses are changed into the present and reduced to their root form. Essentially, each word in our collection of tweets is now associated with its unique id.

3.3.6 Constructing the LDA Model

With the lemmatized text, we constructed a dictionary object representing the amount of how many times a word token occurs within the tweet. The bag of words was implemented with the dictionary as input on Gensim's doc2bow method. Once implemented, the doc2bow method outputted a vector for each tweet where the id of the token and the frequency of words in the document were included. The dictionary was then filtered so that any token that appeared less than three times and any token that contained more than 99 percent of the corpus was removed. In our LDA model, we initialized the number of topics we wanted to extract and the number of passes we wanted to iterate through the corpus during training. We set five topics and five passes as a starting point. We plan to perform hyper parameterization on these parameters to investigate if there is an optimal number of topics and passes for our LDA model. Considering the size of our corpus, we applied the LdaMulticore model with 12 workers. The LdaMultiCore model uses all CPU cores to perform parallelization to decrease the execution time of the model. Training is the same as applying Gensim's LdaModel, but LdaMulticore performs multiprocessing.

3.3.7 Coherence Model

To evaluate the LDA models we constructed, we utilized Gensim's Coherence Model, which calculates the coherence score. The coherence score allows us to judge the quality of the topics outputted from our model. As described in [44], the topic coherence is applied through segmentation, probability estimation, confirmation measure, and aggregation. We obtained our coherence scores based on the LDA model we constructed, the lemmatized tokens, and our dictionary containing the token and its unique identifier. There are currently several coherence scores available through the Gensim Python package; in this thesis, we applied the Cross Validated score, which is ranged between 0 and 1, inclusively. A score closer to 1 indicates that the topics outputted from the model are of high quality.

3.3.8 Perplexity Score

In this thesis, we applied the perplexity score through Gensim's log_preplexity method as another method of evaluating the performance of our model. The perplexity score represents how well the model predicts new data [52]. In contrast to the coherence score, the lower the perplexity score, the more confident and accurate its predictions are. In contrast, a higher perplexity score means the model produces inaccurate predictions. However, we decided to evaluate the model's accuracy using the outputted coherence scores, which have a standard range.

4 Results

Previous research has shown methods to analyze public opinion on various topics within online conversations can be done through computational linguistic approaches such as sentiment analysis and topic modeling [34, 12]. However, emotion recognition has not been applied, which can provide a deeper insight than classifying a text as positive, negative, or neutral. Regarding topic extraction in tweets, the sparse text and lack of content within tweets have presented a challenge in accurately extracting topics hidden within a collection of tweets. Previous work has proven that aggregating similar tweets and applying the LDA model provides the most feasible way to obtain topics [45]. Below we discuss the results of applying the three main components discussed in Chapter 3: sentiment analysis, emotion recognition, and topic modeling. We hope these findings will identify if there is a correlation with the opinion expressed regarding abortion and identify any prejudice against marginalized communities.

4.1 Sentiment Analysis with VADER

To analyze the sentiment expressed in our dataset of tweets, we applied VADER's SentimentIntensityAnalyzer to obtain the sentiment score for each tweet. The implementation of VADER was done using the programming language Python. Using the compound score described in Chapter 3, we investigated if there was an existing correlation between the tweet publication date and the average compound sentiment score.

The application of the SentimentIntensityAnalyzer outputted four scores for each tweet: positive, negative, neutral, and compound. The results were analyzed utilizing the compound score as it provided the best quantitative representation of the sentiment expressed in the text. In Figure 3 and 4, we illustrated the average compound scores for all the tweets for their corresponding publication date. Figure 3 shows how the average compound score for each day a week before the Politico leak and a week after the Politico leak. In Figure **??**, there's a significant decrease in the average compound sentiment score from May 2, 2022, to May 3, 2022. While Politico leaked the Supreme Court's decision on May 2, 2022, it was published at 8:32 PM EDT; hence by the next day, on May 3, 2022, more people were aware of the leak and posted their opinions online. Therefore, we see a decrease from approximately 0.100 to 0.05. We can deduce that on May 2, 2022, most tweets expressed a positive sentiment. Still, it does not show if the tweets were expressing in favor of the possibility of abortion rights no longer being a constitutional right. Since the day of the Politico, we can observe a trend of the average compound sentiment score decreasing, with the lowest average compound score below -0.050 on May 8, 2022. Figure 4 demonstrates the average of the compound score for each day a week before the overturning and a week after the overturning of Roe vs. Wade. Within the specific period, there is a trend of the average compound score for tweets published from June 17, 2022, to June 24, 2022, indicating that most tweets expressed positive sentiment. An interesting observation is the lowest average compound score throughout the respective period. On July 3, 2022, a day before the U.S.'s national holiday Fourth of July, the average compound score was -0.075.

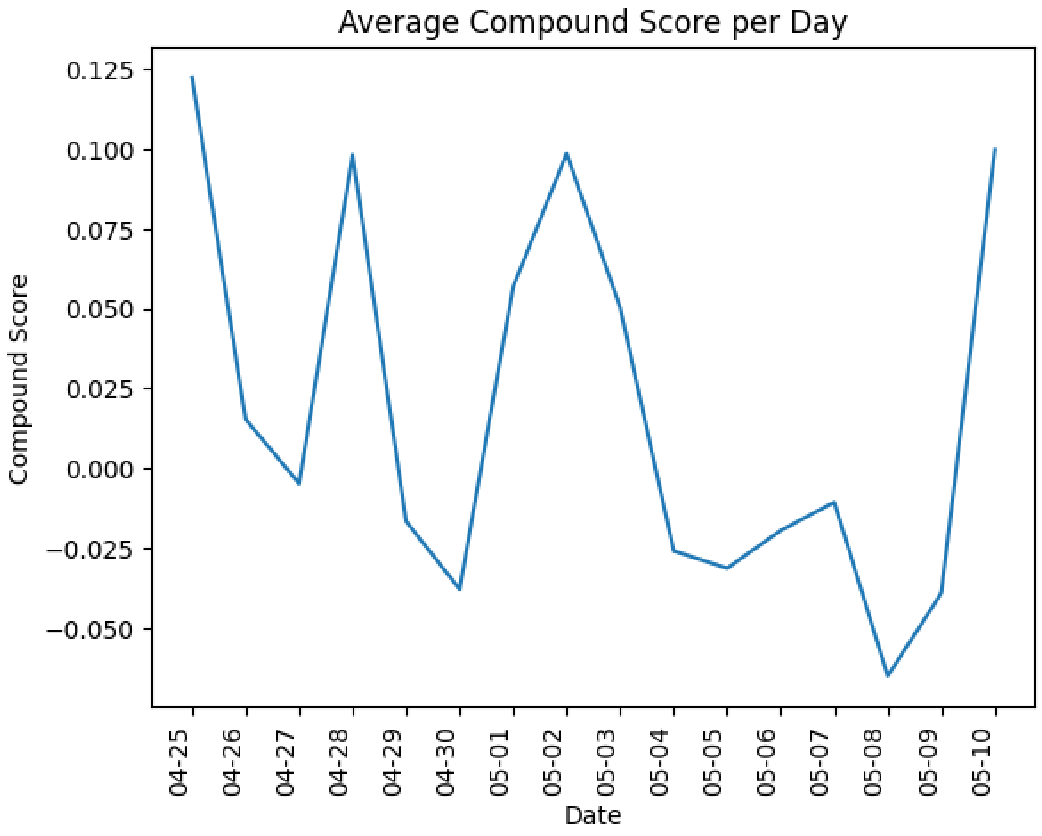

Figure 3: Average Sentiment Score per Day during the Politico Leak

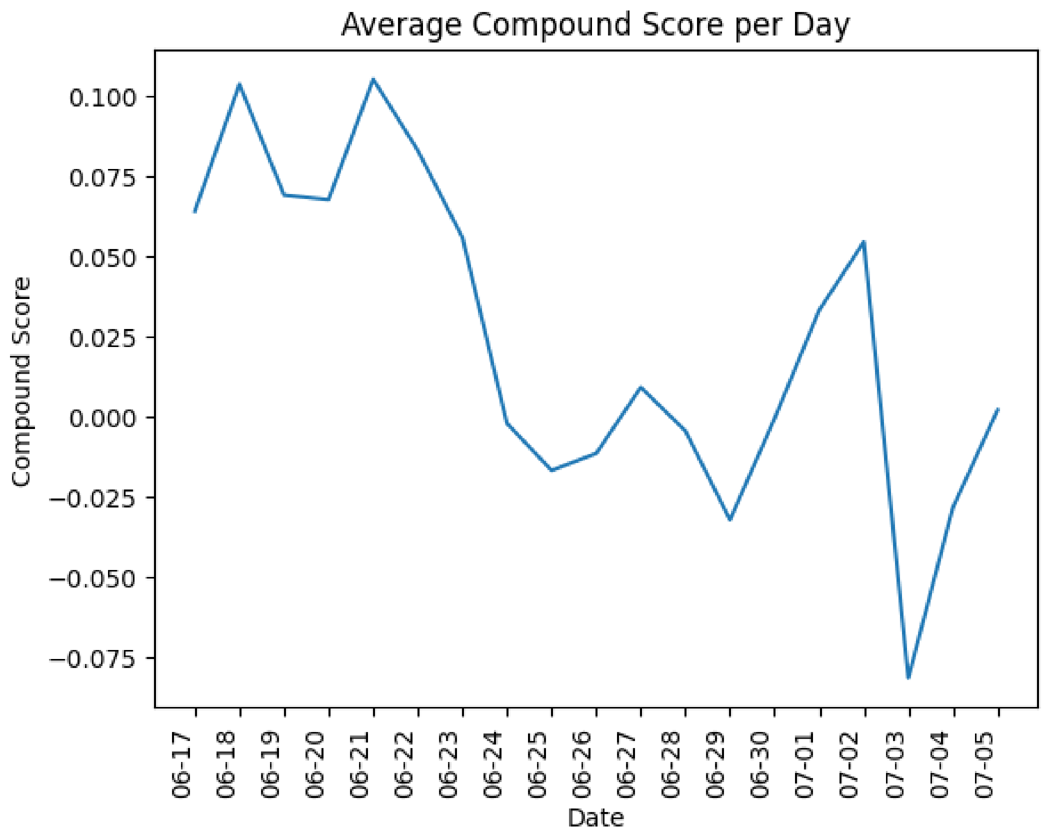

Figure 4: Average Sentiment Score per Day during the Overturning of Roe v Wade

Figure 5 demonstrated the proportion of tweets published each day during the week leading up to the Politico leak and a week after the Politic leak that was classified as positive, negative, and neutral. As mentioned in Chapter 3, a compound score generated from a given text greater than or equal to 0.05 is classified as positive, and less than or equal to -0.05 is classified as negative or neutral. The proportion of approximately 0.5 tweets published on May 2, 2022, classified as positive corresponds to the average compound score on the same date in Figure 3. Similarly, on May 8, 2022, we see that approximately 0.5 of the tweets published on that date were classified as negative, corresponding to the average compound score for May 8, 2022, in Figure **??** as it falls below -0.05. Figure 6 illustrates the proportion of tweets published each day during the week leading up to the overturning of Roe vs. Wade and a week after the overturning of Roe vs. Wade classified as positive, negative, and neutral. The proportion of the tweets published on July 3, 2022, classified as negative, approximately 0.50, outweighs those classified as positive. On the day of Roe v Wade being overturned, there is an average compound score that leans towards a neutral sentiment being expressed, as shown in Figure 4. In 6, we observed an almost equal proportion of the tweets published that day expressing positive and negative sentiments regarding being in favor or not in favor of the SCOTUS decision.

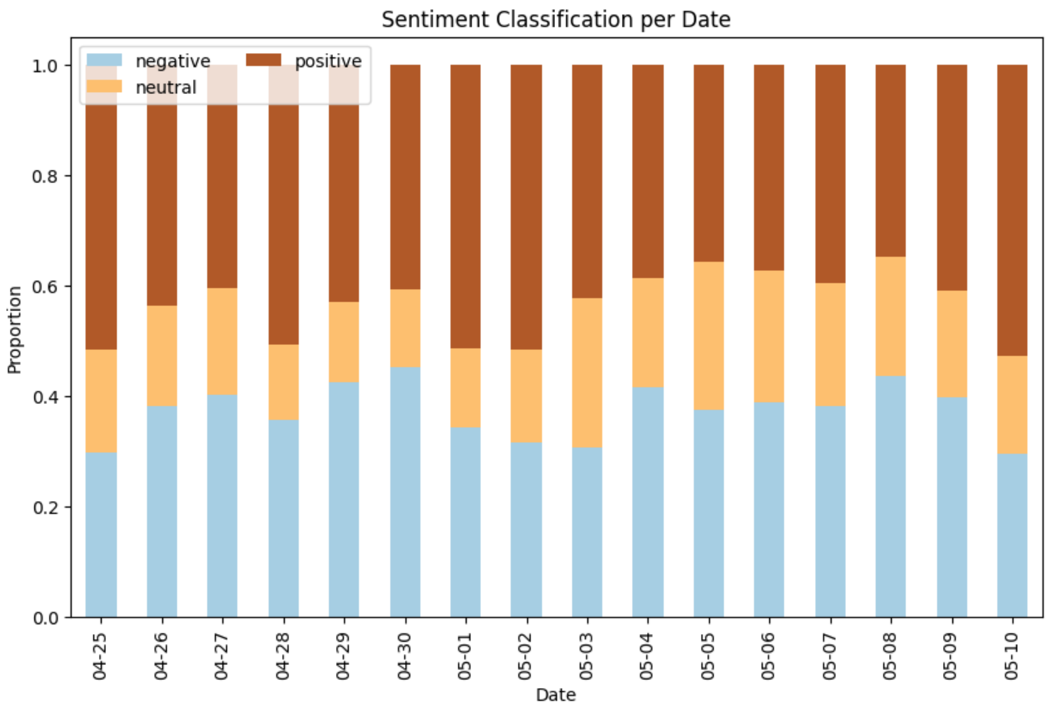

Figure 5: Sentiment classification of Tweets during the Politico leak

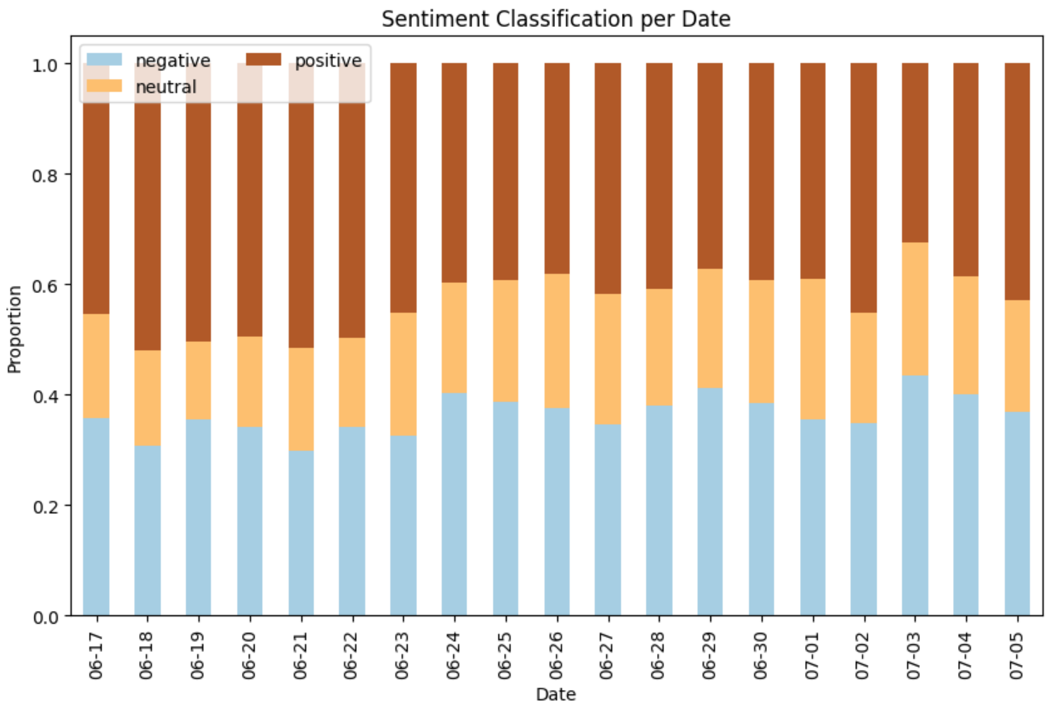

Figure 6: Sentiment Classification of Tweets during the Overturning of Roe v Wade

4.2 Emotion Recognition with TweetEval

Given the computational cost of efficiency in large language models (LMs) such as the one described in Chapter 3, we couldn't compute the emotion expressed for every tweet in our dataset. To increase the speed of execution for the roberta-based-emotion model, we applied Huggingface's BetterTransformers using fused kernels. For the 25,758 tweets published on the day of the Politico leak, the computation took approximately 72 hours to complete. The tweets posted on the day of the overturn, throughout the Politico leak, and then throughout the overturning of Roe vs. Wade period were 4,022,267, 6,300,676, and 14,339,490 tweets, respectively. We applied random sampling to gain preliminary insight into the emotions expressed on those dates. In this analysis, we compared the emotions expressed during the day of the Politico leak with those depicted on the day of the overturning of Roe v Wade, where each set had 25,758 tweets. We applied Panda's random sample method to grab 25,758 tweets at random. While comparing the two time periods, we randomly sampled 10,000 tweets from their respective datasets.

The results of applying emotion recognition are shown in Figure 7. For the tweets published during the day of the Politico leak, the chart is shown on the left, and for the tweets posted on the day of the overturning of Roe vs. Wade is shown on the right. We can observe that the vast majority of tweets published on the day of the Politico leak expressed a positive emotion. More specifically, 42%, 29%, 17%, ad 12% expressed anger, optimism, joy, and sadness, respectively. Out of the 25,758 tweets published that day, 3,148 expressed joy, 3,531 expressed optimism, 15,888 expressed anger, and 3,189 expressed sadness.

Meanwhile, on the day of the overturning of Roe vs. Wade, we witnessed a significant proportion of tweets that expressed anger. For tweets published on the day of the overturning of Roe vs. Wade, we can observe that 62%, 14%, 12%, and 12% expressed anger, optimism, joy, and sadness, respectively. Of the randomly sampled 25,758 tweets published on the day of the overturn, 4,420 expressed joy, 7,373 expressed optimism, 10,758 expressed anger, and 32,07 expressed sadness. The classification results showed a significant increase in tweets expressing anger between the two days, indicating a negative response to the overturn. The decreased proportion of tweets expressing optimism is also noteworthy, as that reduces from 29% to 14%. We can deduce that many users were either optimistic that the leak was not legitimate or agreed with the SCOTUS decision in the Dobbs v. Jackson's Women's Health Organization case.

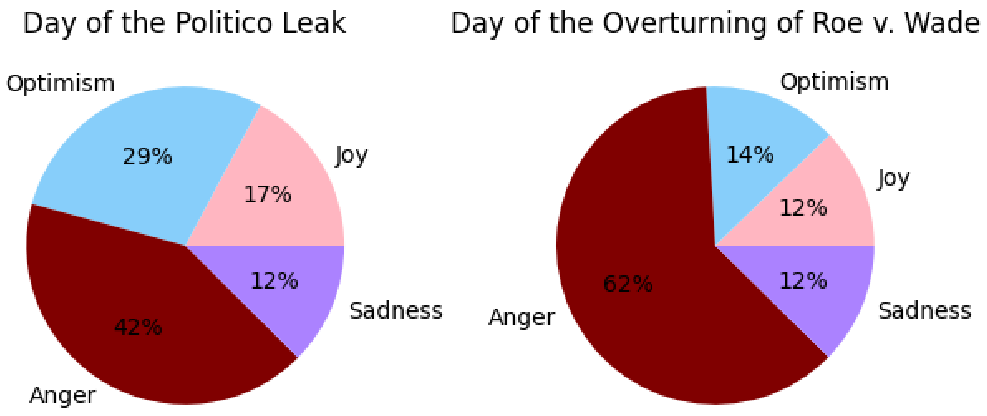

Figure 7: Proportion of Tweets Expressing a Specific Emotion during the Day of Politico magazine Leaking the SCOTUS decision on Dobbs v. Jackson's Women's Health Organization and durinng the Day of the Overturning of Roe vs. Wade, respectively

In Figure 8, we compared the proportion of tweets classified as expressing a specific emotion between the two time periods, where each collection contains 10,000 randomly sampled tweets. On the left, we demonstrated the proportion of tweets published throughout the Politico leak period. On the right, we observed the proportion of tweets posted during the overturning of the Roe v Wade period, classified as expressing anger, optimism, joy, or sadness. In contrast to Figure 7, we don't see a significant difference. During the Politico leak period, 6,255, 1,742, 1,077, and 926 tweets were classified as expressing anger, optimism, sadness, and joy, respectively. The tweets published during the overturning of Roe vs. Wade period, 6,370, 1,618, 1,257, and 755, were classified as expressing anger, optimism, sadness, and joy, respectively. It is possible that due to the limited amount of tweets, we cannot observe a difference. We plan to classify the rest of our data collection in future research.

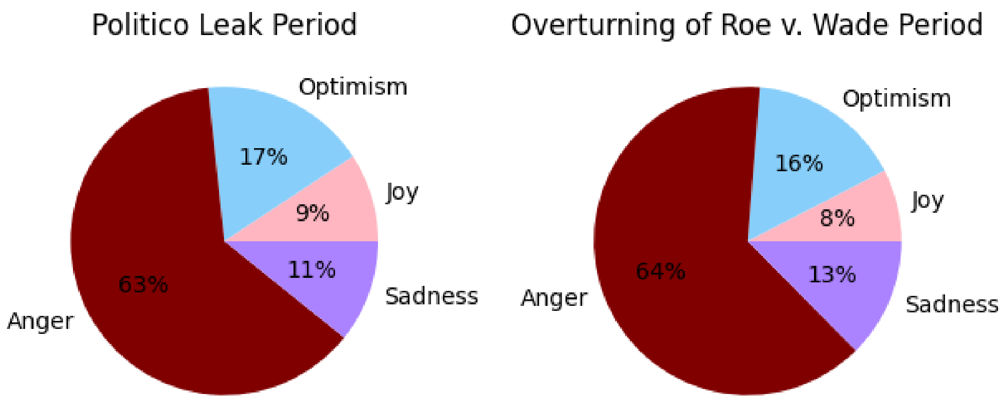

Figure 8: Proportion of Tweets Expressing a Specific Emotion during the Time Period of Politico magazine Leaking the SCOTUS decision on Dobbs v. Jackson's Women's Health Organization and the Overturning of Roe vs. Wade, respectively

4.3 Topic Modeling with Gensim

We used Gensim's LdaMultiCore topic model to implement topic modeling on our tweets collection. In Figures 9 and 10, we present the tweet length distribution of the tweets and the 30 most common words in the dataset where tweets were published on the day of the Politico leak. Similarly, we present Figures 11 and 12, we show the tweet length distribution of the tweets and the 30 most common words in the dataset where tweets were published on the day of the overturning of Roe v Wade. The tweet distribution shows that most of the tweets posted on that day were 17 words long, which juxtaposes the distribution of the tweets for the tweets published on the day of the overturn, which were mainly 11 words long. Given the large scale of the dataset for the tweets posted during the periods of the Politico leak ad the overturning of Roe vs. Wade and the memory constraints when constructing the necessary preprocessing for the LDA model, we perform our LDA on those sets of tweets. For this thesis, we applied the Topic Modeling with Gensim's LDA model on the tweets published on the day of the Politico leak and the overturning of Roe vs. Wade.

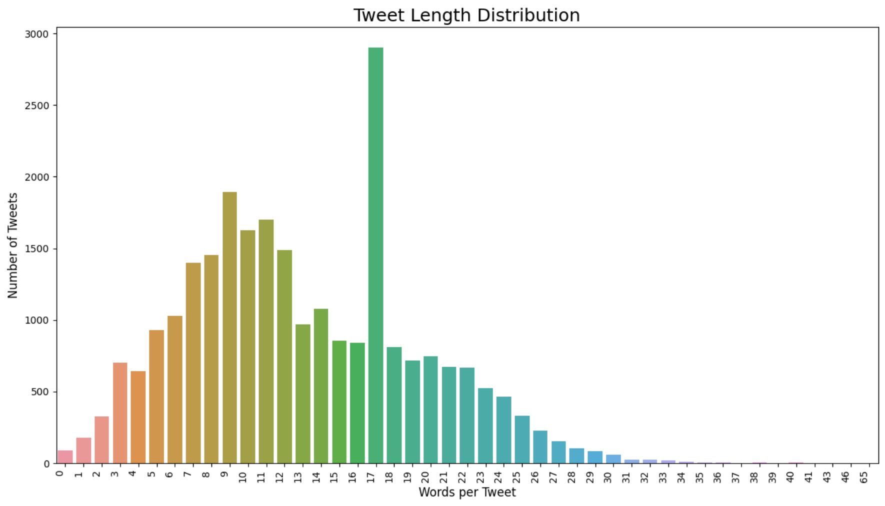

Figure 9: Tweet Length Distribution of the Tweets Published during the Day of Politico magazine Leaking the SCOTUS decision on Dobbs v. Jackson's Women's Health Organization

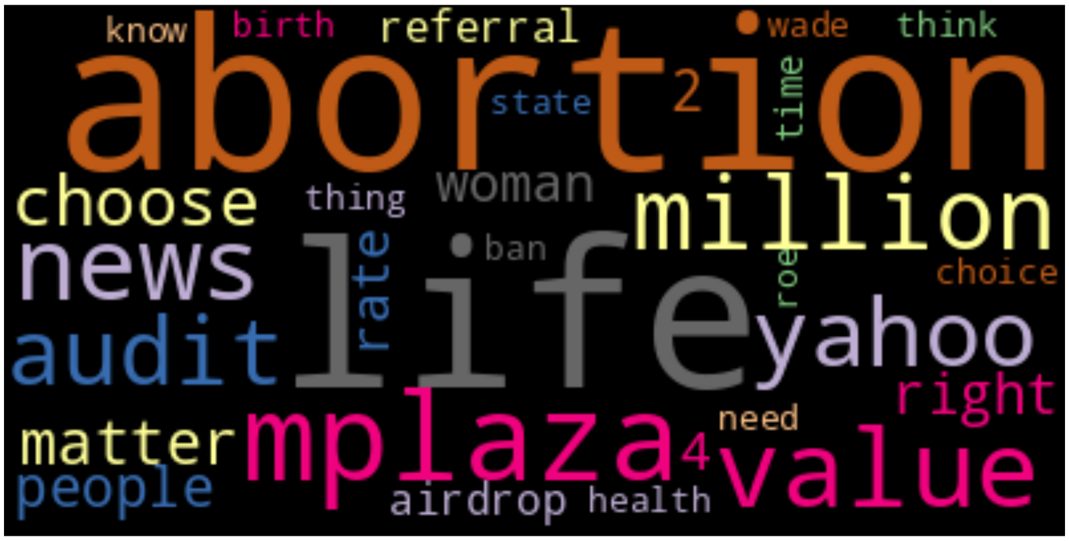

Figure 10: Word Cloud of the 30 Most Frequent Words within the Tweets Published during the Day of Politico magazine Leaking the SCOTUS decision on Dobbs v. Jackson's Women's Health Organization

The topics generated from the LDA model are shown in 4. The results demonstrated five underlying topics within this specific collection of tweets. We can see that ideas such as "women", "choose","life","state","right", are mentioned within our collection. Allowing us to assume that the tweets are referring to states rights, the right to choose, women, women's rights, children, and life. Because our corpus only contained a word total of 374011 and a vocabulary size of 34164, the topics outputted don't seem to output cohesive topics. The coherence score resulted in 0.307 in Table 4, and while this isn't a high score, there is room for improvement through hyper-parameterization. With hyper-parameterization, we still observe some repeated words that include numbers, abortion, life, and woman in Tables 5,6,7; thus indicating that we need to modify how tokens are filtered before entering them into the model. At execution, tokens that occurred less than three times were removed, and tokens contained in more than 99 percent of the corpus were removed. In future work, we would modify filtering so that tokens in 50 percent or more of the corpus are removed.

Topic #	**Terms**
Topic 1	abortion,life,right,state,choose,ban,kill,woman,say,day
Topic 2	life,abortion,choose,matter,woman,value,right,choice,pro,people
Topic 3	mplaza,news,million,yahoo,audit,value,2,4,rate,airdrop
Topic 4	abortion,people,life,value,woman,right,know,time,think,ban
Topic 5	abortion,law,life,woman,roe,people,wade,birth,issue,today

Table 4: Topics generated from LDA model with the Tweets published on the day of the Politico Leak as the Input

Topic #	**Terms**
Topic 1	life,abortion,choose,matter,roe,wade,value,woman,right,ban
Topic 2	abortion,life,right,choose,woman,people,state,live,value,ban
Topic 3	abortion,woman,right,birth,control,life,roe,choose,baby,wade
Topic 4	mplaza,million,life,news,yahoo,audit,value,abortion,2,4
Topic 5	mplaza,news,yahoo,million,audit,value,2,4,rate,airdrop
Topic 6	mplaza,news,million,audit,yahoo,abortion,value,2,rate,4
Topic 7	abortion,life,pro,choice,people,go,right,let,pass,end
Topic 8	abortion,life,choose,right,woman,mplaza,say,roe,people,value
Topic 9	abortion,life,people,choose,matter,woman,right,value,law,say
Topic 10	abortion, life,choose,right,woman,people,child,go,roe,value

Table 5: Topics from Hyperparameterized LDA Model for Tweets published on the day of the Politico Leak with Coherence Score 0.302

Topic #	Terms
Topic 1	abortion,ban,life,woman,mplaza,news,pregnancy,right,million,birth
Topic 2	mplaza,abortion,news,million,audit,yahoo,2,right,value,4
Topic 3	life,mplaza,million,news,audit,yahoo,choose,abortion,value,2
Topic 4	life,matter,choose,human,mplaza,value,million,news,yahoo,choice
Topic 5	abortion,life,people,choose,matter,human,right,think,value,law
Topic 6	life,u,value,abortion,child,people,woman,need,think,year
Topic 7	mplaza,news,audit,million,yahoo,abortion,value,2,cryptotowneu,referral
Topic 8	mplaza,yahoo,million,news,audit,value,4,cryptotowneu,rate,airdrop
Topic 9	abortion,life,right,woman,wade,roe,choose,help,people,human
Topic 10	abortion,mplaza,life,news,audit,million,yahoo,love,right,rate

Table 6: Topics from Hyperparameterized LDA Model for Tweets published on the day of the Politico Leak with Coherence Score of 0.304

Topic #	Terms
Topic 1	abortion,life,woman,right,risk,let,choose,state,grow,day
Topic 2	mplaza,news,million,audit,yahoo,abortionn,value,2,4,life
Topic 3	mplaza,news,audit,yahoo,million,value,2,referral,rate,airdrop
Topic 4	abortion,life,woman,murder,right,child,people,law,choose,help
Topic 5	mplaza,million,yahoo,audit,news,2,value,referral,airdrop,4
Topic 6	abortion,choose,life,today,roe,wade,health,right,help,day
Topic 7	abortion,right,life,woman,choose,time,control,value,human,child
Topic 8	abortion,mplaza,million,news,yahoo,audit,right,value,2,cryptotowneu
Topic 9	life,abortion,change,value,kid,margin,support,feel,today,woman
Topic 10	abortion,life,woman,right,anti,pro,mplaza,value,million,people

Table 7: Topics from Hyperparameterized LDA Model for Tweets published on the day of the Politico Leak with Coherence Score of 0.30659

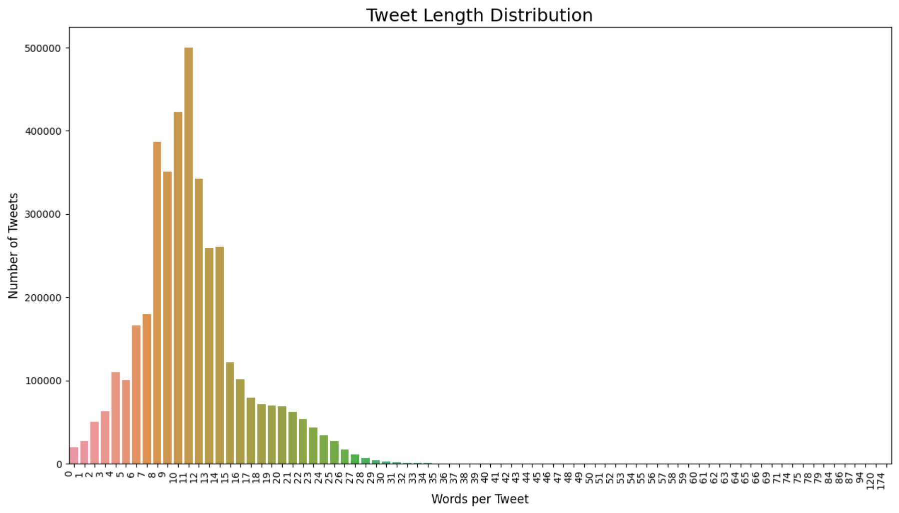

Figure 11: Tweet Length Distribution of the Tweets Published during the Day of the Overturning of Roe vs. Wade

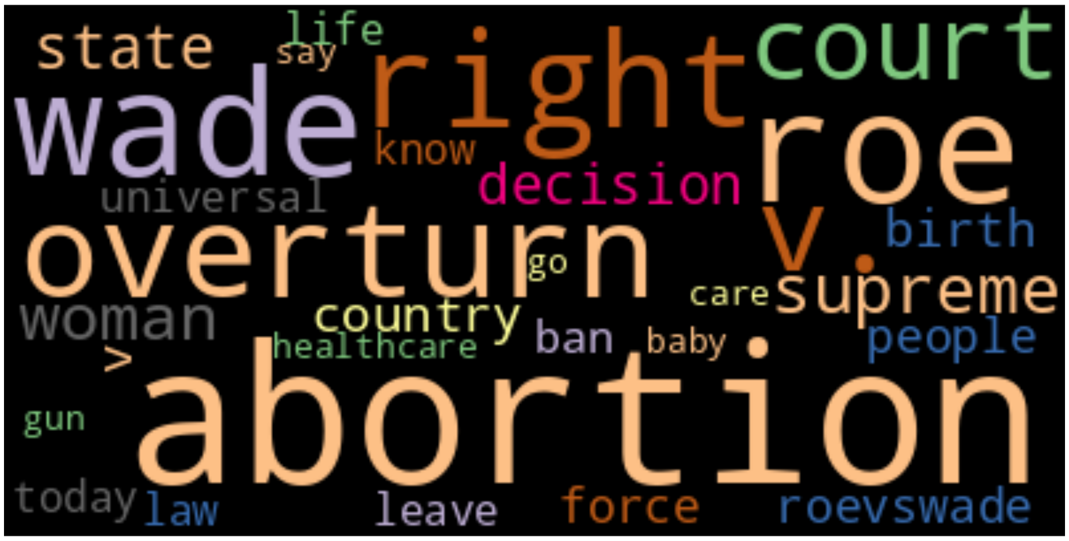

Figure 12: Word Cloud of the 30 Most Frequent Words within the Tweets Published during the Day of the Overturning of Roe vs. Wade

5 Discussion

Analyzing the tweets published during a historic moment in American history can provide a deeper insight into public opinion on abortion rights. However, there are some challenges in analyzing social media data, such as the character limit on tweets, which provides limited context to the tweet. Other challenges presented in social media data are non-alphabetical characters, excessive punctuation, misspellings, slang, and excessive capitalization [27, 47, 35]. Despite these challenges, we could determine the sentiment throughout the presented periods of when the Politico leak occurred and when SCOTUS overturned Roe vs. Wade. We determined dates where the average compound sentiment score differs significantly from the other dates. Still, we have not analyzed what words are used to convey highly negative tweets. In Figure 13, we can observe the 30 most frequent terms that outputted a compound score less than or equal to -0.9. The most frequent words included: ”rape,” ”guns,” ”children,” ”women,” and ”murder.” Thus, indicating words related to guns, women, and murder contributed to producing a negative compound score for the tweet. We also present Figure 14, which analyzes the bigrams in tweets that output a compound sentiment less than or equal to -0.9. The most notable bigrams included: ”gun violence,” ”ban guns,” ”stops abortions,” and ”forced to.” The bigrams present that the topic of guns or the mention of guns is significant within our entire corpus. Reviewing the linguistic components of these tweets holds value as it provides a method to understand other prevalent concerns from the general public during a conversation related to Roe vs. Wade.

In addition to the sentiment, we obtained insight into what specific emotions were expressed within the social media data and which topics were mentioned within the tweets about Roe vs. Wade. Using emotion recognition, we deduced that during the day of the Politico leak, most tweets expressed anger, presumably regarding the possibility of abortion being revoked as a constitutional right. On the day of the overturning of Roe vs. Wade, we witnessed a decrease in tweets expressing optimism, as anger dramatically outweighs the other emotions. Therefore, the comparison between the emotions expressed in the tweets posted on the day of the Politico versus the day of the overturning of Roe vs. Wade provides great motivation towards classifying all of the tweets in our larger datasets that comprise tweets published during the weeks before and after the overturning of Roe vs. Wade and the Politico leak. To our best knowledge, this is the first study of its kind to implement a multiclass classification of the emotion expressed by a given tweet.

In the LDA model implementation, we demonstrated the challenges of analyzing social media data. The results indicate that more preprocessing is needed to obtain more coherent topics. It is worth noting that in Chapter 2, a significant portion of the dataset provided by [9] was unavailable. Given the results from this study, there is value in investigating the linguistic nature of these tweets for a larger-scale analysis. Overall the importance of this work is to provide a deeper insight into the conversations published during historically significant events that had ensued emotional responses from both sides of the political spectrum. The application of natural language processing methods has illustrated varied public reactions to the overturning of Roe vs. Wade.

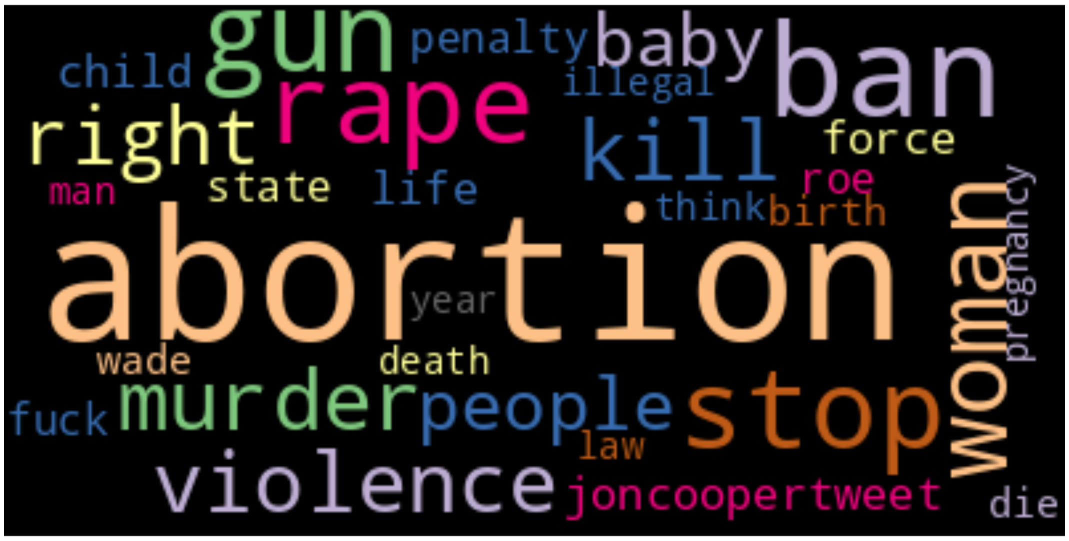

Figure 13: 30 Most Frequent Words in Tweets Less than or Equal to -0.9 Compound Sentiment Score

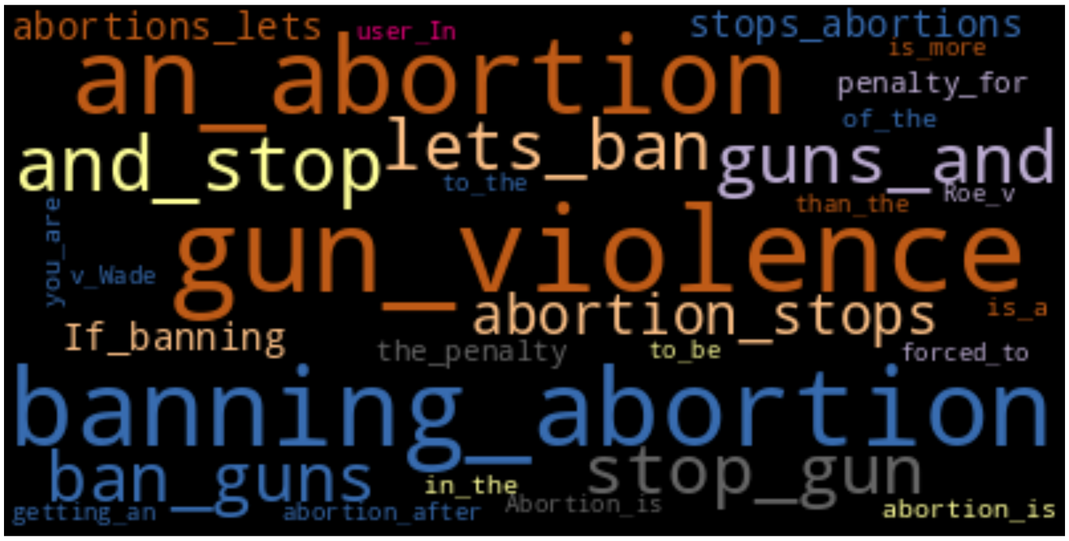

Figure 14: 30 Most Frequent Bigrams in Tweets Less than or Equal to -0.9 Compound Sentiment Score

6 Related Work

Unsupervised machine learning has been applied broadly across multiple domains, including content analysis of textbooks, social media analysis, and Autism research [32, 40, 24]. Natural language processing, a sub-field of unsupervised machine learning, has been used to analyze large volumes of social media data to understand public opinion on specific topics. Recent research has shown how social media data can be used to analyze online conversations to gain better insight into public perception on various topics. In [5], 107,990 tweets related to the COVID-19 pandemic were collected to analyze the themes and trends in the conversations. Frequency of keywords, sentiment analysis with emotion quotients, and topic modeling were used to conduct the study. In this thesis, we used state-of-the-art approaches for performing sentiment analysis using VADER, designed explicitly for analyzing Twitter data. Other studies focused on study conversations in social media focus analyzed conversations such as mental health during the COVID-19 pandemic, eating disorders, and online education during the pandemic [51, 7, 37]. While [51, 37] provided unique and more profound insight for their respective field, it does not demonstrate the specific emotion expressed as they classified their tweets as positive, negative, or neutral. Using TweetEval, we applied emotion recognition to our collection of tweets to calculate the emotions expressed.

A related study examined the public opinion on the overturning of Roe v Wade using 227,161 tweets published between May 1, 2021, and July 15, 2022 [33]. While the study scales to almost a year-long span of tweets, our large dataset will provide the ability to use Language Model to perform more advanced and accurate analysis. [1] used 5,996,741 tweets and apply sentiment analysis on tweets related to Obergefell v. Hodges decision. Similarly, we applied sentiment analysis on a collection of 20,640,166 tweets regarding a Supreme Court case of significant and immediate impact, Dobbs v. Jackson's Women's Health Organization. However, in this thesis, we did not consider the existence of bots within our dataset, as bot detection is significant in social media and one of the many challenges social media data inherits [46].

7 Future Work

Chang et al. describe how the abundance of manipulation caused by social media bots causes unhealthy discourse, thus affecting how the public perceives important political issues [8]. In future work, we would like to detect bots and analyze how bots manipulate the conversation regarding abortion and women's rights. To detect bots within our dataset, we will utilize the Botometer dataset and investigate what content is typically published by these bots as a preliminary step [18].

In this thesis, we only analyzed tweets written in English, but in [20], they address the concern for online manipulation in low-resource languages, such as Tagalog. We would like accurately analyze how users writing in low-resource languages perceived trending events online and how these conversations are prone to manipulation. [42] investigated the conversation on Twitter and Facebook on the lack of regulation on removing Russian propaganda. The growing presence of propaganda affects how users communicate with each other online, as misinformation can cause a ripple effect of distorted perspectives. Thus, we will aim to explore how propaganda and misinformation affect how the public perceives events online within the context of women's rights and abortion rights.

Concerning improving the methodology in this study, we plan to continue investigating n-gram in tweets with a compound sentiment score greater than or equal to 0.9 and less than or equal to 0.9. In addition, we plan to obtain the resources to implement TweetEval's emotion recognition to our sets of tweets within the two larger periods: a week before and after the overturning of Roe vs. Wade and the Politico leak. Using TweetEval, we aim to implement their stance detection, sentiment analysis, and hate speech detection tasks. To improve our topic modeling application on the tweets, we would want to continue hyper parameterizing our models for the dataset containing tweets published on the day of the Politico leak and the day Roe vs. Wade was overturned. While we could not apply the topic models to our large datasets, we plan to do so with the same methodology. Compared to other topic modeling approaches–LDA, Top2Vec, NMF, and BertTopic–NMF and BertTopic have proven to be most efficient in analyzing Twitter data [16]. Thus, in addition to Gensim's LDAMultiCore model, we hope to apply NMF and BertTopic modeling to improve the accuracy of the topics obtained from our corpus [19].

To gain a more expansive view of the public response to the overturning of Roe v Wade, we hope to develop a new and novel methodology that can perform sentiment analysis, emotion recognition, and topic modeling on tweets written in Spanish. The primary motivation is the known prevalence of Spanish-speaking individuals in the United States and online and its lack of attention within the research community. While a previous study has investigated classifying the labels of pro-choice and pro-life on social media, there is still room for improvement [50]. With the implementation of methods utilized in this thesis, we can obtain a cultural analysis of how people from different cultures perceive these events.

8 Conclusion

We analyzed public opinion through sentiment analysis, emotion recognition, and topic modeling. Despite the challenges with social media data, we were able to apply state-of-the-art techniques to tweets related to issues that pose a significant effect on marginalized communities. Using VADER, we applied sentiment analysis and observed that the strongly negative tweets with a sentiment score less than or equal to -0.9 frequently mentioned gun control and healthcare. An insight that has not been investigated in previous research. In addition, we looked at the average compound sentiment score for each day during two distinct periods: a week before and after the Politico leak and the overturning of Roe vs. Wade. We demonstrated that the sentiment of the tweets was largely positive on the day of the Politico leak.

Meanwhile, on the days leading to the overturning of Roe vs. Wade, we demonstrate that there has been an overall decrease in the sentiment score to a neutral one, indicating the strong presence of tweets expressing both positive and negative sentiment. We can see how the sentiment on the day of the overturning of Roe vs. Wade corresponds to the emotion recognition results, as the vast majority of tweets published on the day of the overturning of Roe vs. Wade expressed anger. Applying natural language processing on social media has demonstrated insight into the public's reaction regarding abortion.

9 Bibliography

References

[1] Nicholas Joseph Adams-Cohen. “Policy change and public opinion: measuring shifting political sentiment with social media data”. In: *American Politics Research* 48.5 (2020), pp. 612–621.

[2] Francesco Barbieri et al. “Tweeteval: Unified benchmark and comparative evaluation for tweet classification”. In: *arXiv preprint arXiv:2010.12421* (2020).

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[4] Johan Bollen, Alberto Pepe, and Huina Mao. “Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena”. In: *CoRR* abs/0911.1583 (2009). arXiv: `0911.1583`. URL: `http://arxiv.org/abs/0911.1583`.

[5] Sakun Boon-Itt, Yukolpat Skunkan, et al. “Public perception of the COVID-19 pandemic on Twitter: sentiment analysis and topic modeling study”. In: *JMIR Public Health and Surveillance* 6.4 (2020), e21978.

[6] Raphael Kwaku Botchway et al. “A review of social media posts from UniCredit bank in Europe: a sentiment analysis approach”. In: *Proceedings of the 3rd international conference on business and information Management.* 2019, pp. 74–79.

[7] Patricia A Cavazos-Rehg et al. ““I just want to be skinny.”: A content analysis of tweets expressing eating disorder symptoms”. In: *PloS one* 14.1 (2019), e0207506.

[8] Ho-Chun Herbert Chang et al. “Liberals Engage with More Diverse Policy Topics and Toxic Content Than Conservatives on Social Media”. In: (2023).

[9] Rong-Ching Chang et al. “# RoeOverturned: Twitter Dataset on the Abortion Rights Controversy”. In: *arXiv preprint arXiv:2302.01439* (2023).

[10] Miriam Chappelka et al. “Food for thought: Analyzing public opinion on the supplemental nutrition assistance program”. In: *arXiv preprint arXiv:1710.02443* (2017).

[11] Priyavrat Chauhan, Nonita Sharma, and Geeta Sikka. “The emergence of social media data and sentiment analysis in election prediction”. In: *Journal of Ambient Intelligence and Humanized Computing* 12 (2021), pp. 2601–2627.

[12] Biraj Dahal, Sathish AP Kumar, and Zhenlong Li. “Topic modeling and sentiment analysis of global climate change tweets”. In: *Social network analysis and mining* 9 (2019), pp. 1–20.

[13] Amanda D. Damiano and Jennifer R. Allen Catellier. “A Content Analysis of Coronavirus Tweets in the United States Just Prior to the Pandemic Declaration”. In: *Cyberpsychology, Behavior, and Social Networking* 23.12 (2020). PMID: 32813572, pp. 889–893. DOI: 10.1089/cyber.2020.0425. eprint: https://doi.org/10.1089/cyber.2020.0425. URL: https://doi.org/10.1089/cyber.2020.0425.

[14] Nemanja Djuric et al. “Hate Speech Detection with Comment Embeddings”. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15 Companion. Florence, Italy: Association for Computing Machinery, 2015, pp. 29–30. ISBN: 9781450334730. DOI: 10.1145/2740908.2742760. URL: https://doi.org/10.1145/2740908.2742760.

[15] Viola Savy Dsouza et al. “A sentiment and content analysis of tweets on monkeypox stigma among the LGBTQ+ community: A cue to risk communication plan”. In: *Dialogues in Health* 2 (2023), p. 100095.

[16] Roman Egger and Joanne Yu. “A topic modeling comparison between lda, nmf, top2vec, and bertopic to demystify twitter posts”. In: *Frontiers in sociology* 7 (2022).

[17] *Exclusive: Supreme Court has voted to overturn abortion rights, draft opinion shows*. URL: https://www.politico.com/news/2022/05/02/supreme-court-abortion-draft-opinion-00029473.

[18] Shangbin Feng et al. “TwiBot-22: Towards graph-based Twitter bot detection”. In: *arXiv preprint arXiv:2206.04564* (2022).

[19] Maarten Grootendorst. “BERTopic: Neural topic modeling with a class-based TF-IDF procedure”. In: *arXiv preprint arXiv:2203.05794* (2022).

[20] Samar Haider et al. “Detecting social media manipulation in low-resource languages”. In: *arXiv preprint arXiv:2011.05367* (2020).

[21] Frank A Haight. “Handbook of the Poisson distribution”. In: 1967.

[22] Daniel Hickey et al. “Auditing Elon Musk’s Impact on Hate Speech and Bots”. In: *arXiv preprint arXiv:2304.04129* (2023).

[23] Gabriel Hine et al. “Kek, cucks, and god emperor trump: A measurement study of 4chan’s politically incorrect forum and its effects on the web”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 11. 1. 2017, pp. 92–101.

[24] Esther Hong et al. “Topography and function of challenging behaviors in individuals with autism spectrum disorder”. In: *Advances in Neurodevelopmental Disorders* 2 (2018), pp. 206–215.

[25] Homa Hosseinmardi et al. “Analyzing labeled cyberbullying incidents on the instagram social network”. In: *Social Informatics: 7th International Conference, SocInfo 2015, Beijing, China, December 9-12, 2015, Proceedings 7*. Springer. 2015, pp. 49–66.

[26] Clayton Hutto and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media.* Vol. 8. 1. 2014, pp. 216–225.

[27] Gerald C Kane et al. "What's different about social media networks? A framework and research agenda". In: *MIS quarterly* 38.1 (2014), pp. 275–304.

[28] Qaisar Khan and Hui Na Chua. "Comparing topic modeling techniques for identifying informative and uninformative content: A case study on COVID-19 tweets". In: *2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET).* IEEE. 2021, pp. 1–6.

[29] Samuel Kotz, Narayanaswamy Balakrishnan, and Norman L Johnson. In: *Continuous Multivariate Distributions.* Vol. 1. Models and Applications. Wiley, 2000, pp. 485–523.

[30] Yachao Li et al. "Constructing and Communicating COVID-19 Stigma on Twitter: A Content Analysis of Tweets during the Early Stage of the COVID-19 Outbreak". In: *International Journal of Environmental Research and Public Health* 17.18 (2020). ISSN: 1660-4601. DOI: `10.3390/ijerph17186847`. URL: `https://www.mdpi.com/1660-4601/17/18/6847`.

[31] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[32] Li Lucy et al. "Content analysis of textbooks via natural language processing: Findings on gender, race, and ethnicity in Texas US history textbooks". In: *AERA Open* 6.3 (2020), p. 2332858420940312.

[33] Heran Mane et al. "Examination of the Public's Reaction on Twitter to the Over-Turning of Roe v Wade and Abortion Bans". In: *Healthcare.* Vol. 10. 12. MDPI. 2022, p. 2390.

[34] J Mantas et al. "Application of topic modeling to tweets as the foundation for health disparity research for COVID-19". In: *The Importance of Health Informatics in Public Health during a Pandemic* 272 (2020), p. 24.

[35] Diana Maynard, Kalina Bontcheva, and Dominic Rout. "Challenges in developing opinion mining tools for social media". In: *Proceedings of the@ NLP can u tag# usergeneratedcontent* (2012), pp. 15–22.

[36] Saif Mohammad et al. "Semeval-2018 task 1: Affect in tweets". In: *Proceedings of the 12th international workshop on semantic evaluation.* 2018, pp. 1–17.

[37] Muhammad Mujahid et al. "Sentiment analysis and topic modeling on tweets about online education during COVID-19". In: *Applied Sciences* 11.18 (2021), p. 8438.

[38] Vladimir Ostrovski. "Testing equivalence to families of multinomial distributions with application to the independence model". In: *Statistics & Probability Letters* 139 (2018), pp. 61–66.

[39] Toni Pano and Rasha Kashef. "A complete VADER-based sentiment analysis of bitcoin (BTC) tweets during the era of COVID-19". In: *Big Data and Cognitive Computing* 4.4 (2020), p. 33.

[40] Chelsea M Parlett-Pelleriti et al. "Applications of Unsupervised Machine Learning in Autism Spectrum Disorder Research: a Review". In: *Review Journal of Autism and Developmental Disorders* (2022), pp. 1–16.

[41] James W Pennebaker, Martha E Francis, and Roger J Booth. "Linguistic inquiry and word count: LIWC 2001". In: *Mahway: Lawrence Erlbaum Associates* 71.2001 (2001), p. 2001.

[42] Francesco Pierri et al. "Propaganda and Misinformation on Facebook and Twitter during the Russian Invasion of Ukraine". In: *arXiv preprint arXiv:2212.00419* (2022).

[43] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[44] Michael Röder, Andreas Both, and Alexander Hinneburg. "Exploring the space of topic coherence measures". In: *Proceedings of the eighth ACM international conference on Web search and data mining*. 2015, pp. 399–408.

[45] Asbjørn Steinskog, Jonas Therkelsen, and Björn Gambäck. "Twitter topic modeling by tweet aggregation". In: *Proceedings of the 21st nordic conference on computational linguistics*. 2017, pp. 77–86.

[46] Stefan Stieglitz et al. "Do social bots dream of electric sheep? A categorisation of social media bot accounts". In: *arXiv preprint arXiv:1710.04044* (2017).

[47] Stefan Stieglitz et al. "Social media analytics–Challenges in topic discovery, data collection, and data preparation". In: *International journal of information management* 39 (2018), pp. 156–168.

[48] Praveen Sv et al. "Twitter-Based Sentiment Analysis and Topic Modeling of Social Media Posts Using Natural Language Processing, to Understand People's Perspectives Regarding COVID-19 Booster Vaccine Shots in India: Crucial to Expanding Vaccination Coverage". In: *Vaccines* 10.11 (2022), p. 1929.

[49] Hiroki Takikawa and Kikuko Nagayoshi. "Political polarization in social media: Analysis of the "Twitter political field" in Japan". In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 3143–3150. DOI: `10.1109/BigData.2017.8258291`.

[50] Danny Valdez et al. "Identifying accurate pro-choice and pro-life identity labels in Spanish: Social media insights and implications for comparative survey research". In: *Perspectives on Sexual and Reproductive Health* (2022).

[51] Danny Valdez et al. “Social media insights into US mental health during the COVID-19 pandemic: Longitudinal analysis of Twitter data”. In: *Journal of medical Internet research* 22.12 (2020), e21418.

[52] Hanna M Wallach et al. “Evaluation methods for topic models”. In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 1105–1112.

10 Appendices

Appendix A: Scraping data from the Twitter API

```
import requests
import os
import csv
import dateutil.parser
import time

#insert your bearer token here
def setTokens():
    bearer_token = "INSERT BEARER TOKEN HERE"
    os.environ['TOKEN'] = 'INSERT BEARER TOKEN HERE'

#set the bearer token here - necessary for authorization
def auth():
    return os.getenv('TOKEN')

def create_headers(bearer_token):
    headers = {"Authorization": "Bearer {}".format(bearer_token)}
    return headers

#create the call to grab data from the API
def create_url(keyword, start_date, end_date, max_results = 20):
    search_url = "https://api.twitter.com/2/tweets/search/all"

    #change params based on the endpoint you are using
    query_params = {'query': keyword,
                    'start_time': start_date,
                    'end_time': end_date,
                    'max_results': max_results,
                    'expansions': 'author_id,in_reply_to_user_id,geo.place_id',
                    'tweet.fields': 'id,text,author_id,in_reply_to_user_id,geo,conversation_id,\
                    created_at,lang,public_metrics,referenced_tweets,reply_settings,\
                    source',
                    'user.fields': 'id,name,username,created_at,description,public_metrics,verified,\
                    location',
                    'place.fields': 'full_name,id,country,country_code,geo,name,place_type',
                    'next_token': {}}
    return (search_url, query_params)
```

Figure 15: Applying Authorization and Creating the Header

```
#response is retrieved here
def connect_to_endpoint(url, headers, params, next_token = None):
    params['next_token'] = next_token   #params object received from create_url function
    response = requests.request("GET", url, headers = headers, params = params)
    print("Endpoint Response Code: " + str(response.status_code))
    if response.status_code == 503:
        print("Status Code 503: \n HyperText Transfer Protocol (HTTP) 503 Service Unavailable Error")
        raise Exception(response.status_code, response.text)
    if response.status_code == 429:
        print("Status Code 429")
        raise Exception(response.status_code, response.text)
    if response.status_code != 200:
        raise Exception(response.status_code, response.text)
    return response.json()
```

Figure 16: Generating a response

```
def append_to_csv(json_response, fileName):
    #dictionary of all the user and user information
    user_dict= {}

    #A counter variable for counting the number of tweets the response
    counter = 0

    #create/open the csv file containing the information grabbed from the response
    csvFile = open(fileName, "a", newline="", encoding='utf-8')
    csvWriter = csv.writer(csvFile)

    #grabbinng user's information from the response
    #each key is the user, the value is the user's information
    for user in json_response['includes']['users']:
        user_dict[user['id']] = {'username': user['username'],
                                 'followers': user['public_metrics']['followers_count'],
                                 'tweets': user['public_metrics']['tweet_count'],
                                 'description': user.get("description",""),
                                 'location': user.get("location",""),
                                 'verified':user.get("verified",""),
                                 'created_at':user.get("created_at","")
                                }
```

```
#iterate through each tweet
for tweet in json_response['data']:
    #grab the author id
    author_id = tweet['author_id']
    #for each tweet we grab the author's information
    authorInfo = user_dict[author_id]
    #author's username
    username = authorInfo["username"]
    #author's number of followers
    author_followers = authorInfo["followers"]
    #author's number of published tweets
    author_tweets = authorInfo["tweets"]
    #author's bio description
    author_description = authorInfo["description"]
    #author's location published on his/her/their profile
    author_location = authorInfo["location"]
    #a boolean value that indicates if the user is verified
    verified = authorInfo["verified"]
    #the time stamp when the author created his/her/their account
    author_created_at = authorInfo["created_at"]
    #when the tweet was created
    created_at = dateutil.parser.parse(tweet['created_at'])

    #grabbing the geo object and its associated attributes
    geoObj = None
    geo_placeid = ""
    geo_url = ""
    geo_placeType=""
    geo_name=""
    geo_fullName=""
    geo_countryCode=""
    geo_country=""
    geo_boundingBox=None

    if ('geo' in tweet):
        geoObj = tweet['geo']
        geo_placeid = "" if 'place_id' not in geoObj else geoObj['place_id']
        geo_url = "" if 'url' not in geoObj else geoObj['url']
        geo_placeType = "" if 'place_type' not in geoObj else geoObj['place_type']
        geo_name = "" if 'name' not in geoObj else geoObj['name']
        geo_fullName = "" if 'full_name' not in geoObj else geoObj['full_name']
        geo_countryCode = "" if 'country_code' not in geoObj else geoObj['country_code']
        geo_country = "" if 'country' not in geoObj else geoObj['country']
        geo_boundingBox = None if 'bounding_box' not in geoObj else geoObj['bounding_box']
    else:
        geoObj = ""
     #grabbing the tweet id
     tweet_id = tweet['id']

     #indicates the language the user used to write the tweet
     lang = tweet['lang']

     #the number of times the tweet was retweeted
     retweet_count = tweet['public_metrics']['retweet_count']
     #the number of replies
     reply_count = tweet['public_metrics']['reply_count']
     #the number of likes
     like_count = tweet['public_metrics']['like_count']
     #the number of times someone quoted the tweet
     quote_count = tweet['public_metrics']['quote_count']

     #the source of the tweet (example: Mobile Phone)
     source = tweet['source'] if 'source' in tweet else ''

     #the text of the tweet
     text = tweet['text']

     #all the values we collected from the response
     res = [author_id, username, author_created_at, verified, author_followers, author_tweets, \
     author_description,author_location, tweet_id, text, created_at,lang, source, like_count, \
     quote_count, reply_count, retweet_count,geoObj,geo_placeid,geo_url,geo_placeType, geo_name,\
     geo_fullName,geo_countryCode,geo_country,geo_boundingBox]

     #appendinng the information to the csv file
     csvWriter.writerow(res)
     counter += 1

#close the csv file once we went through the entire response
csvFile.close()

#Print the number of tweets for this iteration
print("Number of Tweets added from this response: ", counter)
```

Figure 17: Saving scraped data to a file

```
if __name__ == '__main__':
    setTokens()
    bearer_token = auth()
    headers = create_headers(bearer_token)
    KEYWORD_Q = "insert the keyword you want to include in your query"
    keyword = "KEYWORD_Q lang:en -is:retweet"


    #list of the start dates for the query
    start_list =    ['YEAR-MONTH-DAYT00:00:00.000Z']
    #list of the end dates for the query
    end_list =      ['YEAR-MONTH-DAYT0:00:00.000Z']

    #list of the start and end dates for the file name, in string format
    datesStart = []
    datesEnd =   []

    #while running this script we included the following dates
    #04/25/2022-05/10/2022
    #06/17/2022-07/05/2022

    max_results = 500 #the max amount of tweets we want from the response
    total_tweets = 0 #Total number of tweets we collected from the loop

    for i in range(0,len(start_list)):
        startDate = datesStart[i]
        endDate = datesEnd[i]

        fileName = "KEYWORD_Q"+startDate+"_"+endDate+".csv"
        # Create file
        csvFile = open(fileName, "a", newline="", encoding='utf-8')
        csvWriter = csv.writer(csvFile)

        #Creating the columns in the dataset
        csvWriter.writerow(['author_id', 'username', 'author_created_at', 'verified',\
        'author_followers', 'author_tweets','author_description', 'author_location', 'tweet_id',\
        'text', 'created_at', 'lang', 'source', 'like_count', 'quote_count', 'reply_count',\
        'retweet_count','geoObj','geo_placeid','geo_url','geo_placeType', 'geo_name','geo_fullName',\
        'geo_countryCode','geo_country','geo_boundingBox'])
        csvFile.close()


        count = 0 # Counting tweets per time period
        max_count = 500000 # Max tweets per time period
        scrape = True
        next_token = None
    while scrape:
        # Check if the number of tweets collected is greater than
        # the max_count of tweets for that time period
        if count >= max_count:
            #move on to the next time period
            break
        print("Token: ", next_token)
        url = create_url(keyword, start_list[i],end_list[i], max_results)
        json_response = connect_to_endpoint(url[0], headers, url[1], next_token)
        result_count = json_response['meta']['result_count']

        if 'next_token' in json_response['meta']:
            # Save the token to use for next call
            next_token = json_response['meta']['next_token']
            print("NEXT_TOKEN: ", next_token)
            if result_count is not None and result_count > 0 and next_token is not None:
                print("Start Date: ", start_list[i])
                append_to_csv(json_response, fileName)
                count += result_count
                total_tweets += result_count
                print("Total # of Tweets added: ", total_tweets)
                print("-------------------")
                time.sleep(5)
        # If no next token exists
        else:
            if result_count is not None and result_count > 0:
                print("NO NEXT_TOKEN")
                print("Start Date: ", start_list[i])
                append_to_csv(json_response, fileName)
                count += result_count
                total_tweets += result_count
                print("Total # of Tweets added: ", total_tweets)
                print("-------------------")
                time.sleep(5)

            #Turn the scrape variable
            #to false to move to the next time period.
            scrape = False
            next_token = None
        time.sleep(5)

print("Total number of results: ", total_tweets)
```

Figure 18: Main function to gather data

Appendix B: Collecting tweets from an open-sourced dataset

```
def read_data(dataFile):
    allIds = pd.read_csv(dataFile)
    allIds = [ str(id) for id in allIds['id']]
    return allIds
```

Figure 19: Collecting all tweet IDs from the file

```
def create_url(curIds):
    #adding tweet fields
    tweet_fields = "tweet.fields=lang,author_id,id,text,in_reply_to_user_id,geo, \
    created_at,withheld,source,possibly_sensitive,public_metrics,referenced_tweets,\
    conversation_id,reply_settings"
    #adding user fields
    user_fields = "user.fields=id,name,username,created_at,description,public_metrics,\
    verified,location"
    #adding expansions
    expansions = "expansions=author_id,in_reply_to_user_id,geo.place_id"
    #adding place fields
    place_fields = "place.fields=full_name,id,country,country_code,geo,\
    name,place_type"
    #valid twitter ids from our data
    ids = "ids="+curIds
    # You can adjust ids to include a single Tweets.
    # Or you can add to up to 100 comma-separated IDs
    url = "https://api.twitter.com/2/tweets?{}&{}&{}&{}&{}".format \
    (ids, tweet_fields,user_fields,place_fields,expansions)
    return url

def bearer_oauth(r):
    """
    Method required by bearer token authentication.
    """
    r.headers["Authorization"] = f"INSERT BEARER TOKEN HERE"
    r.headers["User-Agent"] = "v2TweetLookupPython"
    return r

def connect_to_endpoint(url):
    response = requests.request("GET", url, auth=bearer_oauth)
    if response.status_code != 200:
        raise Exception(
            "Request returned an error: {} {}".format(
                response.status_code, response.text
            )
        )
    return response.json()
```

Figure 20: Authentication and Generating a Response

```
def append_to_csv(json_response,fileName):
    #dictionary of users grabbed
    user_dict= {}
    #A counter variable for keeping track the number of responses
    counter = 0
    csvFile = open(fileName, "a", newline="", encoding='utf-8')
    csvWriter = csv.writer(csvFile)

    if 'errors' in json_response:
        for errorRes in json_response['errors']:
            isError = True
            error_detail = errorRes['detail']
            error_parameter = errorRes['parameter']
            tweet_id = errorRes['resource_id']
            error_resource_type = errorRes['resource_type']
            error_title = errorRes['title']
            error_type = errorRes['type']
            error_value = errorRes['value']

            author_id = ""
            username = ""
            author_created_at = ""
            verified = ""
            author_created_at = ""
            author_followers = ""
            author_tweets = ""
            author_description = ""
            author_location = ""
            text = ""
            created_at = ""
            lang = ""
            source = ""
            like_count = ""
            quote_count = ""
            reply_count = ""
            retweet_count = ""
            geoObj = None
            geo_boundingBox = ""
            geo_placeid = ""
            geo_url = ""
            geo_placeType=""
            geo_name = ""
            geo_fullName = ""
            geo_country=""
            geo_countryCode=""
            res = [author_id, username, author_created_at, verified, author_followers,\
            author_tweets,author_description,author_location, tweet_id, text, \
            created_at,lang, source, like_count, quote_count, reply_count, retweet_count,\
            geoObj,geo_placeid,geo_url,geo_placeType, geo_name,geo_fullName,geo_countryCode,\
            geo_country,geo_boundingBox,isError,error_detail, error_parameter, error_resource_type,\
            error_title, error_type, error_value ]

            csvWriter.writerow(res)
```

Figure 21: Dealing with error objects from the open-sourced dataset

```
#grab each user's information
if 'includes' in json_response:
    for user in json_response['includes']['users']:
        user_dict[user['id']] = {'username': user['username'],
                                'followers': user['public_metrics']['followers_count'],
                                'tweets': user['public_metrics']['tweet_count'],
                                'description': user.get("description",""),
                                'location': user.get("location",""),
                                'verified':user.get("verified",""),
                                'created_at':user.get("created_at","")
                                }
```

Figure 22: Collecting the user objects returned

```
#Loop through each tweet
if 'data' in json_response:
    for tweet in json_response['data']:
        #grabbing the tweets' associated information
        #author information
        author_id = tweet['author_id']
        authorInfo = user_dict[author_id]
        username = authorInfo["username"]
        author_followers = authorInfo["followers"]
        author_tweets = authorInfo["tweets"]
        author_description = authorInfo["description"]
        author_location = authorInfo["location"]
        verified = authorInfo["verified"]
        author_created_at = authorInfo["created_at"]

        #time the tweet was created
        created_at = dateutil.parser.parse(tweet['created_at'])

        # geo location object and attributes
        geoObj = None
        geo_placeid = ""
        geo_url = ""
        geo_placeType=""
        geo_name=""
        geo_fullName=""
        geo_countryCode=""
        geo_country=""
        geo_boundingBox=None

        if ('geo' in tweet):
            geoObj = tweet['geo']
            geo_placeid = "" if 'place_id' not in geoObj else geoObj['place_id']
            geo_url = "" if 'url' not in geoObj else geoObj['url']
            geo_placeType = "" if 'place_type' not in geoObj else geoObj['place_type']
            geo_name = "" if 'name' not in geoObj else geoObj['name']
            geo_fullName = "" if 'full_name' not in geoObj else geoObj['full_name']
            geo_countryCode = "" if 'country_code' not in geoObj else geoObj['country_code']
            geo_country = "" if 'country' not in geoObj else geoObj['country']
            geo_boundingBox = None if 'bounding_box' not in geoObj else geoObj['bounding_box']
        else:
            geoObj = ""

        #id of the tweet
        tweet_id = tweet['id']

        #the language the user wrote the tweet
        lang = tweet['lang']

        #tweet's public metrics
        retweet_count = tweet['public_metrics']['retweet_count']
        reply_count = tweet['public_metrics']['reply_count']
        like_count = tweet['public_metrics']['like_count']
        quote_count = tweet['public_metrics']['quote_count']
        #how the tweet was published
        source = tweet['source'] if 'source' in tweet else ''

        #text of the tweet
        text = tweet['text']

        #the error object and associated attributes
        isError = False
        error_detail = ""
        error_parameter = ""
        error_resource_type = ""
        error_title = ""
        error_type  = ""
        error_value =""

        #add data to the csv file
        res = [author_id, username, author_created_at, verified, author_followers,\
        author_tweets,author_description,author_location, tweet_id, text, created_at,lang,\
        source, like_count, quote_count, reply_count, retweet_count,geoObj,geo_placeid,\
        geo_url,geo_placeType, geo_name,geo_fullName,geo_countryCode,geo_country,\
        geo_boundingBox,isError,error_detail, error_parameter, error_resource_type,\
        error_title,error_type, error_value ]
        csvWriter.writerow(res)
        counter += 1
csvFile.close()
```

Figure 23: Grabbing tweet information

```
def main():
    for filename in os.listdir(r"FILEPATH"):
        if filename.endswith(".csv"):
            #create the data file
            resultsFile = filename.split('.')[0]+"_data.csv"
            dataFile = filename
            allTweets = read_data(dataFile)

            #create the csv file and its columns
            csvFile = open(resultsFile, "a", newline="", encoding='utf-8')
            csvWriter = csv.writer(csvFile)
            csvWriter.writerow(['author_id', 'username', 'author_created_at', 'verified',\
            'author_followers', 'author_tweets','author_description','author_location',\
            'tweet_id','text', 'created_at','lang', 'source', 'like_count', 'quote_count',\
            'reply_count', 'retweet_count','geoObj','geo_placeid','geo_url','geo_placeType',\
            'geo_name','geo_fullName','geo_countryCode','geo_country','geo_boundingBox',\
            'isError','error_detail', 'error_parameter', 'error_resource_type', 'error_title',\
            'error_type', 'error_value' ])
            csvFile.close()

            responseCount=0 #the number of responses
            i = 0
            while i < len(allTweets):
                tweet = allTweets[i:i+100] #we incremented by 100 before
                i+=100 #grab every 100 tweets in the file
                responseCount +=1
                if responseCount % 250 == 0:
                    time.sleep(900) #sleep to prevent 429 Error
                tweet = ','.join(tweet)
                url = create_url(tweet)
                json_response = connect_to_endpoint(url)
                append_to_csv(json_response,resultsFile) #add response to data file
                time.sleep(5)

if __name__ == "__main__":
    main()
```

Figure 24: Main function to commence data collection

Appendix C: VADER implementation

```
import regex as re
import pandas as pd
import os
import csv
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

#preprocess text here, remove urls
def preprocess(text):
    #removing urls
    text = re.sub(r'http\S+', '', text)
    return text

#apply sentiment classification based on the compound score
def sentimentLabel(compound):
    #positive sentiment, compound >= 0.05
    #neutral sentiment, compound > -0.05 and compound <0.05
    #negative sentimennt, compound <= -0.05

    classifiedSentiment = ""
    if compound >= 0.05:
       classifiedSentiment = "positive"
    elif compound <= -0.05:
       classifiedSentiment = "negative"
    elif compound > -0.05 and compound < 0.05:
       classifiedSentiment = "neutral"
    return classifiedSentiment

#read file as a pandas data frame
df=pd.read_csv("FILEPATH",index_col=0)
analyzer = SentimentIntensityAnalyzer()
#preprocess each tweet
df['vader_preprocessed_text'] = df['text'].apply(preprocess)

#grab the positive, neutral, negative, and compound scores
df['vader_compound'] = [analyzer.polarity_scores(x)['compound'] \
for x in df['vader_preprocessed_text']]

df['vader_neg'] = [analyzer.polarity_scores(x)['neg'] for x in df['vader_preprocessed_text']]

df['vader_neu'] = [analyzer.polarity_scores(x)['neu'] for x in df['vader_preprocessed_text']]

df['vader_pos'] = [analyzer.polarity_scores(x)['pos'] for x in df['vader_preprocessed_text']]

#grab the sentiment label
df['sentiment'] = df['vader_compound'].apply(sentimentLabel)
#save results into a csv file
df.to_csv("FILENAME.csv")
```

Figure 25: Generating the Sentiment Scores for each Tweet in our Data Collection

Appendix D: Emotion Recognition

```
from transformers import AutoModelForSequenceClassification
from optimum.bettertransformer import BetterTransformer
from transformers import AutoTokenizer
import numpy as np
from scipy.special import softmax
import csv
import urllib.request
import pandas as pd
import os

# Preprocess text (username and link placeholders)
def preprocess(text):
    new_text = []
    if text is np.nan:
        return None
    for t in text.split(" "):
        t = '@user' if t.startswith('@') and len(t) > 1 else t
        t = 'http' if t.startswith('http') else t
        new_text.append(t)
    return " ".join(new_text)


def grabOutputs(text):
    if text is np.nan:
        return None,None,None,None,None,None,None,None
    else:
        encoded_input = tokenizer(text, return_tensors='pt')
        output = model(**encoded_input)
        scores = output[0][0].detach().numpy()
        scores = softmax(scores)
        ranking = np.argsort(scores)
        ranking = ranking[::-1]
        emotionLabel1 = labels[ranking[0]]
        emotionScore1 = scores[ranking[0]]
        emotionLabel2 = labels[ranking[1]]
        emotionScore2 = scores[ranking[1]]
        emotionLabel3 = labels[ranking[2]]
        emotionScore3 = scores[ranking[2]]
        emotionLabel4 = labels[ranking[3]]
        emotionScore4 = scores[ranking[3]]
        return emotionLabel1,emotionScore1,emotionLabel2,emotionScore2,emotionLabel3,\
            emotionScore3,emotionLabel4,emotionScore4
```

Figure 26: Generating the Emotion label and score for each Tweet in our Data Collection

```
task='emotion'
#name of our model
MODEL = f"cardiffnlp/twitter-roberta-base-{task}"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
# download label mapping
mapping_link = \
f"https://raw.githubusercontent.com/cardiffnlp/tweeteval/main/datasets/{task}/mapping.txt"

with urllib.request.urlopen(mapping_link) as f:
    html = f.read().decode('utf-8').split("\n")
    csvreader = csv.reader(html, delimiter='\t')
labels = [row[1] for row in csvreader if len(row) > 1]

#creating the model for the first time
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
#applying BetterTransformer for faster execution
model = BetterTransformer.transform(model, keep_original_model=True)
#saving tokenizer
tokenizer.save_pretrained(MODEL)
#read data file
df = pd.read_csv("FILENAME.csv",index_col=0)
df = df.sample(n=25758) #random sample
#inserting columnns
df.insert(0,"preprocessed_roberta_emotion"," ")
df.insert(1,"emotion_label1"," ")
df.insert(2,"emotion_score1"," ")
df.insert(3,"emotion_label2"," ")
df.insert(4,"emotion_score2"," ")
df.insert(5,"emotion_label3"," ")
df.insert(6,"emotion_score3"," ")
df.insert(7,"emotion_label4"," ")
df.insert(8,"emotion_score4"," ")
#preprocess text
df['preprocessed_roberta_emotion'] = df['text'].map(preprocess)

#generate emotion labels and score
df['emotion_label1'], df['emotion_score1'],df['emotion_label2'],df['emotion_score2'],\
df['emotion_label3'],df['emotion_score3'],df['emotion_label4'],df['emotion_score4'] = \
zip(*df['preprocessed_roberta_emotion'].map(grabOutputs))


df.to_csv("FILENAME.csv")
model = BetterTransformer.reverse(model)
model.save_pretrained(MODEL)
```

Figure 27: Continuation of the Code used for Emotion Recognition Python Implementation

Appendix E: LDA Topic Modeling

```
import gensim
import pandas as pd
from nltk.probability import FreqDist
from gensim.corpora import Dictionary
from gensim.models.ldamodel import LdaModel
from gensim.models.ldamulticore import LdaMulticore
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from gensim.models.coherencemodel import CoherenceModel
import spacy
from spacy.tokenizer import Tokenizer
from gensim.parsing.preprocessing import STOPWORDS as SW
from wordcloud import STOPWORDS
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import pickle

#data cleaning modules
import emoji
import regex
import re
import string

#importing graph and visualization modules
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud


#generating stop words
stopwords = set(STOPWORDS)
#loading en_codeweb_lg
nlp = spacy.load('en_core_web_lg')
```

Figure 28: Loading the Necessary Modules for Topic Modeling

```
def removeURLs(text):
    text = re.sub(r"http\S+","",text)
    return text

def removeEmojis(text):
    #UNICODE_EMOJI to EMOJI_DATA
    if not(pd.isnull(text)):
        emojiList = [char for char in text if char in emoji.EMOJI_DATA]
        cleanText = ' '.join([str for str in text.split() if not any(i in str for i in emojiList)])
    else:
        cleanText = ''
    return cleanText

def removeAt(text):
    newText = " ".join(filter(lambda x:x[0]!='@', text.split()))
    return newText


def tokenizeText(df):
    tokenizer = Tokenizer(nlp.vocab)

    stopWordsList = ['hi','\n','\n\n', '&amp;', ' ', '.', '-', 'got', "it's", 'it's', "i'm", 'i'm', 'im
    STOP_WORDS = nlp.Defaults.stop_words.union(stopWordsList)

    stop_words_punc = list(string.punctuation)
    # ALL_STOP_WORDS = spacy + gensim + wordcloud
    ALL_STOP_WORDS = STOP_WORDS.union(SW).union(stopwords).union(stop_words_punc)
    ALL_STOP_WORDS.union(['0','1','2','3','4','5','6','7','8','9'])
    tokens = []

    for doc in tokenizer.pipe(df['noURLTweets'],batch_size = 500):
        doc_tokens = []
        for token in doc:
            if token.text.lower() not in ALL_STOP_WORDS:
                doc_tokens.append(token.text.lower())
        tokens.append(doc_tokens)

    df['tokens'] = tokens
    return df
```

Figure 29: Functions for Preprocessing our Tweets for our LDA Model

```
def lemmatize(text):
    lemmas = []
    doc = nlp(text)

    for token in doc:
        if ((token.is_stop == False) and (token.is_punct == False)):
            lemmas.append(token.lemma_)
    return lemmas

def lemmaTokenize(text):

    #remove urls
    pattern = r"http\S+"
    tokens = re.sub(pattern,"",text) #remove url
    tokens = re.sub('[^a-zA-Z 0-9]', '', text)
    tokens = re.sub('[%s]' % re.escape(string.punctuation), '', text) # Remove punctuation
    tokens = re.sub('\w*\d\w*', '', text) # Remove words containing numbers
    tokens = re.sub('@*!*\$*', '', text) # Remove @ ! $
    tokens = tokens.lower().split() #make the text lowercase and create a list of tokens
    return tokens
```

Figure 30: Lemmatization of our Data

```
def preprocessText(df):
    #remove emojis - apply the funnction and remove emojis for all tweets
    # x is all tweets
    emojiFreeText = lambda x: removeEmojis(x)
    df['noEmojiTweets'] = df['text'].apply(emojiFreeText)

    #remove @ from tweets
    noAtTweet =lambda x: removeAt(x)
    df['noAt'] = df['noEmojiTweets'].apply(noAtTweet)

    #remove urls
    urlFreeText = lambda x: removeURLs(x)
    df['noURLTweets'] = df['noAt'].apply(urlFreeText)
    # df['noURLTweest'] = df['noEMojiTweets'].apply(removeURLs)


    #tokenize each tweet
    df = tokenizeText(df)

    #make tokens a string again
    df['tokens_back_to_text'] = [' '.join(map(str,l)) for l in df['tokens']]

    #lemmatize text
    # lemmaText = lambda x: lemmatize(x)
    df['lemmas'] = df['tokens_back_to_text'].apply(lemmatize)

    # #make the lemmas a string again
    df['lemma_back_to_text'] = [' '.join(map(str,l)) for l in df['lemmas']]

    # #remove stop words, punctuation and lowercase annd any special characters
    df['lemma_tokens'] = df['lemma_back_to_text'].apply(lemmaTokenize)

    return df
```

Figure 31: Application of the Preprocessing Functions

```
def examineCorpus(df):
    allWords= [word for tokens in df['lemma_tokens'] for word in tokens]
    tweetLengths = [len(tokens) for tokens in df['lemma_tokens']]
    vocabList = sorted(list(set(allWords)))
    df['lemma_tokens_length'] = tweetLengths

    print(len(allWords),'words total, with a vocabulary size of',len(vocabList)))
    print('Max tweet length is',max(tweetLengths))
    return df

def plotTweetLength(df):
    plt.figure(figsize = (15,8))
    sns.countplot(data=df,x='lemma_tokens_length')
    plt.title('Tweet Length Distribution',fontsize = 18)
    plt.xlabel('Words per Tweet',fontsize = 12)
    plt.ylabel('Number of Tweets',fontsize=12)
    plt.xticks(rotation=90, ha='right')
    #save the plot
    plt.savefig("TweetLengthDistribution.png")


def freqTweets(df):
    flatWords = [item for sublist in df['lemma_tokens'] for item in sublist]
    wordFreq = FreqDist(flatWords)
    #grab the most commonn words
    wordFreq.most_common(30)

    #grab the word/count pairs from wordFreq
    mostCommonCount = [x[1] for x in wordFreq.most_common(30)]
    mostCommonWord = [x[0] for x in wordFreq.most_common(30)]

    #create the dictionary for word/count
    top30WordsDict = dict(zip(mostCommonWord,mostCommonCount))

    #generate a word cloud
    wordcloudVis = WordCloud(colormap = 'Accent',background_color = 'black')\
    .generate_from_frequencies(top30WordsDict)

    #plot with matplotlib
    plt.figure(figsize=(12,8))
    plt.imshow(wordcloudVis,interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.savefig('top20WordsCloudv2.png')
    plt.show()
```

Figure 32: Examining the Number of Tokens, Tweet Length Distribution, and the Most Frequent Words in our Data Collection

```
def bagOfWords(df):
    #create dictionary
    corpus=df['lemma_tokens']
    textDict = Dictionary(corpus)

    #filter out extremes
    textDict.filter_extremes(no_below=3,no_above=0.99)

    #tweetsBOW contains a vector for each tweet =
    # (word id, frequency of word occurrence in document)
    tweetsBOW = [textDict.doc2bow(tweet) for tweet in corpus]

    return tweetsBOW,textDict

def ldaModelBuild(df,tweetsBOW,textDict,k):
    tweetsLDA = LdaMulticore(corpus=tweetsBOW,num_topics=k,id2word=textDict,workers=12,passes=5)

    #print the topics
    print(tweetsLDA.show_topics(),file=sourceFile)

    #compute coherence score
    coherence_model= CoherenceModel(model=tweetsLDA,texts=df['lemma_tokens'],
                                dictionary=textDict,coherence='c_v')
    coherence_lda_model_base = coherence_model.get_coherence()
    print('\nCoherence Score: ',coherence_lda_model_base,file=sourceFile)
    return tweetsLDA
```

Figure 33: Generating Bag of Words and LDA Model Implementation

```
def hyperparameterTuning(df):
    vectorizer = CountVectorizer()
    data_vectorized = vectorizer.fit_transform(df['lemma_back_to_text'])

    #define search param
    searchParams = {'n_components': [10,15,20,25,30], 'learning_decay': [.5,.7,.9]}

    #init the model
    lda = LatentDirichletAllocation()

    #init grid search class
    model = GridSearchCV(lda,param_grid=searchParams)

    #do the grid search
    model.fit(data_vectorized)
    GridSearchCV(cv=None,error_score='raise',
                 estimator=LatentDirichletAllocation(batch_size=128,
                                                     doc_topic_prior=None,
                                                     evaluate_every=-1,
                                                     learning_decay=0.7,
                                                     learning_method=None,
                                                     learning_offset=10.0,
                                                     max_doc_update_iter=100,
                                                     max_iter=10,
                                                     mean_change_tol=0.001,
                                                     n_components=10,
                                                     n_jobs=1,
                                                     perp_tol=0.1,
                                                     random_state=None,
                                                     topic_word_prior=None,
                                                     total_samples=1000000.0,
                                                     verbose=0),
            n_jobs=1,
            param_grid={'n_topics':[10,15,20,30],
                        'learning_decay':[0.5,0.7,0.9]},
            pre_dispatch='2*n_jobs',refit=True,return_train_score='warn',
            scoring=None,verbose=0)
    #best model
    bestLDAModel = model.best_estimator_

    #model Parameters
    print("Best Model's Params: ",model.best_params_,file=sourceFile)

    #log likelihood score
    print("Best Log Likelihood Score: ",model.best_score_,file=sourceFile)

    #perplexity
    print("Model Perplexity: ",bestLDAModel.perplexity(data_vectorized),file=sourceFile)
    return model
```

Figure 34: Hyperparameterization of the LDA Model

```
def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):

    coherence_values_topic = []
    model_list_topic = []
    for num_topics in range(start, limit, step):
        model = LdaMulticore(corpus=corpus, num_topics=num_topics, id2word=dictionary)
        model_list_topic.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v
        coherence_values_topic.append(coherencemodel.get_coherence())

    return model_list_topic, coherence_values_topic
```

Figure 35: Computing the Coherence Values of our Models Generated from the Hyperparameterization Process

```
#print results
sourceFile = open('results.txt', 'w')

#read the file
df = pd.read_csv("FILENAME.csv",index_col=0)

#call preprocess text
df = preprocessText(df)

#examine the corpus
df = examineCorpus(df)
#convert df to a pickle file
df.to_pickle("FILENAME.pkl")
df = pd.read_pickle("FILENAME.pkl")

#generate a plot of the tweet length distribution
plotTweetLength(df)

#generate a word cloud of the most frequent tweets
freqTweets(df)

#generate bags of words
bow,textDict = bagOfWords(df)

#save bag of words to pickle file
bowOut = open("bowFile.pickle","wb")
pickle.dump(bow,bowOut)

#save id2wordcount to pickle file
textDictOut = open("textDictFile.pickle","wb")
pickle.dump(textDict,textDictOut)

#open corresponding pickle files
bowObj = open("bowFile.pickle","rb")
bow = pickle.load(bowObj)
textDictObj = open("textDictFile.pickle","rb")
textDict = pickle.load(textDictObj)
```

Figure 36: Data Preprocessing for the LDA model on Tweets

```
k=5
tweetsLDA = ldaModelBuild(df,bow,textDict,k) #this will print the topics

orgModelFile = open('baseModel.pickle','wb')
pickle.dump(tweetsLDA,orgModelFile,protocol=pickle.HIGHEST_PROTOCOL)


#hyperparameter tuning
bestMod = hyperparameterTuning(df)
bestModFile = open('bestMod.pickle','wb')
pickle.dump(bestMod,bestModFile,protocol=pickle.HIGHEST_PROTOCOL)


# Can take a long time to run.
model_list_topic, coherence_values_topic = compute_coherence_values(dictionary=textDict,
                                                        corpus=bow,
                                                        texts=df['lemma_tokens'],
                                                        start=2, limit=200, step=6)

count = 1
for model in model_list_topic:
    tempModFile = open("model{}.pickle".format(count),'wb')
    pickle.dump(model,tempModFile,protocol=pickle.HIGHEST_PROTOCOL)
    print(model.show_topics(),file=sourceFile)
    count+=1
    print("-----------------------------------",file=sourceFile)
print("",file=sourceFile)

print("coherence_values_topic",coherence_values_topic,file=sourceFile)

sourceFile.close()
```

Figure 37: Constructing and Evaluating our LDA

Appendix F: Bigram Generation

```
import nltk
import re
import pandas as pd
import string
nltk.download('punkt')

#remove urls and user mentions
def removeUrl(text):
    #remove url
    text = re.sub(r"http\S+","",text)

    #replace At sign
    new_text = []
    for t in text.split(" "):
        t = '@user' if t.startswith('@') and len(t) > 1 else t
        new_text.append(t)
    text = " ".join(new_text)

    #remove punctuation
    text = re.sub(r'[^\w\s]', '', text)
    return text

#calculate bigrams using nltk
def grabBigrams(text):
    #return bigrams for the tweet
    nltk_tokens = nltk.word_tokenize(text)
    return list(nltk.bigrams(nltk_tokens))



df = pd.read_csv("negativePointNine.csv",index_col=0)

df['clean_pre_bigrams'] = df['text'].apply(removeUrl)
df['bigrams'] = df['clean_pre_bigrams'].apply(grabBigrams)
df.to_csv("mostNegBigrams.csv")
```

Figure 38: Generating the Bigrams of Tweets that had a Compound Sentiment Score Less than or Equal to -0.9