CIS 600 – Social Media and Data Mining

Project Report

"Exploring Twitter Data to Understand Communities: Clustering, Key-word Analysis & Sentiment Analysis".

Team:

Avantika Mahalingam Iyer
Gina Roh
Himanshu Churi
Janani Hemachandran
Sneha Sudheer Madathi Paramba
Yuheun Rachel Kim

Table of Contents

Abstract

This project presents a detailed and exhaustive study on understanding communities on Twitter. Twitter has become a great hub to share information, facts, and opinions on several topics. We have explored three communities in our project which are Technology, Economy, and Politics. We have implemented the project in three main phases which are clustering, keyword analysis, and sentiment analysis techniques to scrutinize communities based on tweets. The aim of this project is to examine tweets with the goal of comprehending various communities and groups of individuals more comprehensively. We used k-means clustering algorithms to group similar tweets from users based on the keywords of the tweets for each community. Further, to aid in better visualization of the clusters, the dimensionality is reduced using Multi Dimensionality scaling method. The clusters were analyzed using sentiment analysis techniques which are Vader, AFINN, Polarity and Subjectivity score to determine the overall sentiment or emotion expressed. NRC Lexicon and NRC VAD to understand the emotions associated with the sentiments of the clusters. Keyword analysis consisting of LDA Modelling, Word Cloud Analysis and Summarization enabled us to understand the subset of users inside the clusters. The project's findings indicate that sentiment in the Technology community is predominantly positive, while the Economics and Politics communities have mixed sentiments that lean towards negative. The analysis also revealed the top terms and themes in each cluster, as well as a significant presence of individuals from various backgrounds and domains across all communities.

1. Introduction

Twitter has become a notable social media platform with over 330 million active users across the world. It is widely used by people of diverse backgrounds, including various age groups, races, and locations. The platform offers a feature to share short messages known as "tweets," which can be up to 280 characters long. This concise format enables users to express their thoughts, opinions, and information rapidly. Due to its real-time and unfiltered nature, Twitter data analysis an essential tool for obtaining insights into different communities' interests, preferences, and emotions. The focus of this project is to comprehend diverse groups and communities on Twitter through the implementation of clustering, keyword analysis, and sentiment analysis techniques.

The project's primary objective is to achieve an in-depth comprehension of the various communities and groups of people on Twitter. By analysing Twitter data, we can identify frequently used keywords and topics, determine users' sentiments and emotions, and cluster similar users based on their characteristics. The project findings can help businesses, organizations, and individuals create targeted marketing campaigns, enhance social media management, and identify emerging trends in different industries.

The project incorporates k-means clustering algorithm to group similar Twitter users. Evaluation methods such as elbow method and silhouette analysis were used to determine the optimal 'K' value. Sentiment Analysis includes VADER and AFINN tools to identify the sentiments of the texts in the clusters. The NRC-Lexicon and NRC-VAD algorithms are used to identify the emotions associated with these tweets in each cluster. Additionally, the polarity and subjectivity of the tweets are calculated to get overall sentiment insights about each cluster. The keyword analysis uses LDA modelling to understand the type of users in each cluster of all communities. By analysing the most frequently occurring words and topics within each cluster, we are able to gain insights into the interests, beliefs, and motivations of the individuals within each community. Word clouds are generated to show the most frequently occurring words within each cluster, highlighting the most important themes and topics within each community. Further, summarization technique is applied to generate short summaries of the most important insights gleaned from the analysis, making it easier to communicate these insights.

1.1 Objectives

1. To identify the dominant topics and themes of discussion within each community on Twitter, which will help us understand the interests and concerns of people in these communities.
2. To explore the similarities and differences between the communities in terms of their language use, sentiment, and discourse patterns, which will help us gain insights into the patterns and behaviours of these communities on Twitter.

2. Literature Survey

There are four research papers that are relevant to the problem being addressed in this project. This work is influenced by and connected to multiple groups of research. There were two similar research that were close to the problem definition that is stated and is aimed to implement here in this project.

The first one is "TwiInsight: Discovering Topics and Sentiments from Social Media Datasets" by Wang et al., which proposes an algorithm that automatically detects topics and their corresponding sentiments in various categories. This algorithm uses different methods such as Skeyttle, RAKE, and GATE POS Tagger for sentiment analysis, topic detection, and correlation detection. They also compare the performance of three algorithms (StanfordCoreNLP, Havenondemand, and Monkeylearn) and visualize an information summarizer to view the location difference based on geo-location.[1]

The second research paper is "Clustering and Classification of Like-Minded People from Their Tweets" by Jaffali et al. This paper proposes two algorithms for grouping like-minded users. The first algorithm retrieves interest centers from users' posts and groups them with like-minded users. The second algorithm retrieves groups with maximum correlation between them using PCA. SVM is used to classify new users, and K-means is used for re-grouping users in like-minded categories. They evaluate their algorithms using the Sander corpus and TREC 2011 Microblog track and compare them with baseline models such as K-means, LDA, and SMSC. They also analyze user distribution per cluster.[2]

The third research paper is “Analysis and Visualization of Twitter data using k-means Clustering” by Neha Garg, Rinkle Rani et al. This paper concentrates on Twitter data and uses R language. Using the R programming language, the Twitter data is acquired, preprocessed, analyzed, and visualized. Initially, the data is extracted and then undergoes preprocessing before being clustered based on its geotagged information. The number of clusters is obtained by using the Elbow method. The clustered tweets are then visualized on the Indian Map.[3]

The fourth research paper is “Characterized communities of hashtag usage on twitter using during the 2020 covid-19 pandemic by multi-view clustering” by Iain J Cruickshank and Kathleen M Carley. The study aims at analyzing the mostly discussed topics on Twitter using hashtags. To achieve accurate clustering using hashtags, the paper has introduced an approach called Multi-view Clustering. The results show that there are distinct temporal and topical trends present within COVID-19 twitter discussion. The study finally concludes that there are three different types of clusters pertaining to COVID-19 twitter discussion.[4]

3. Methodology

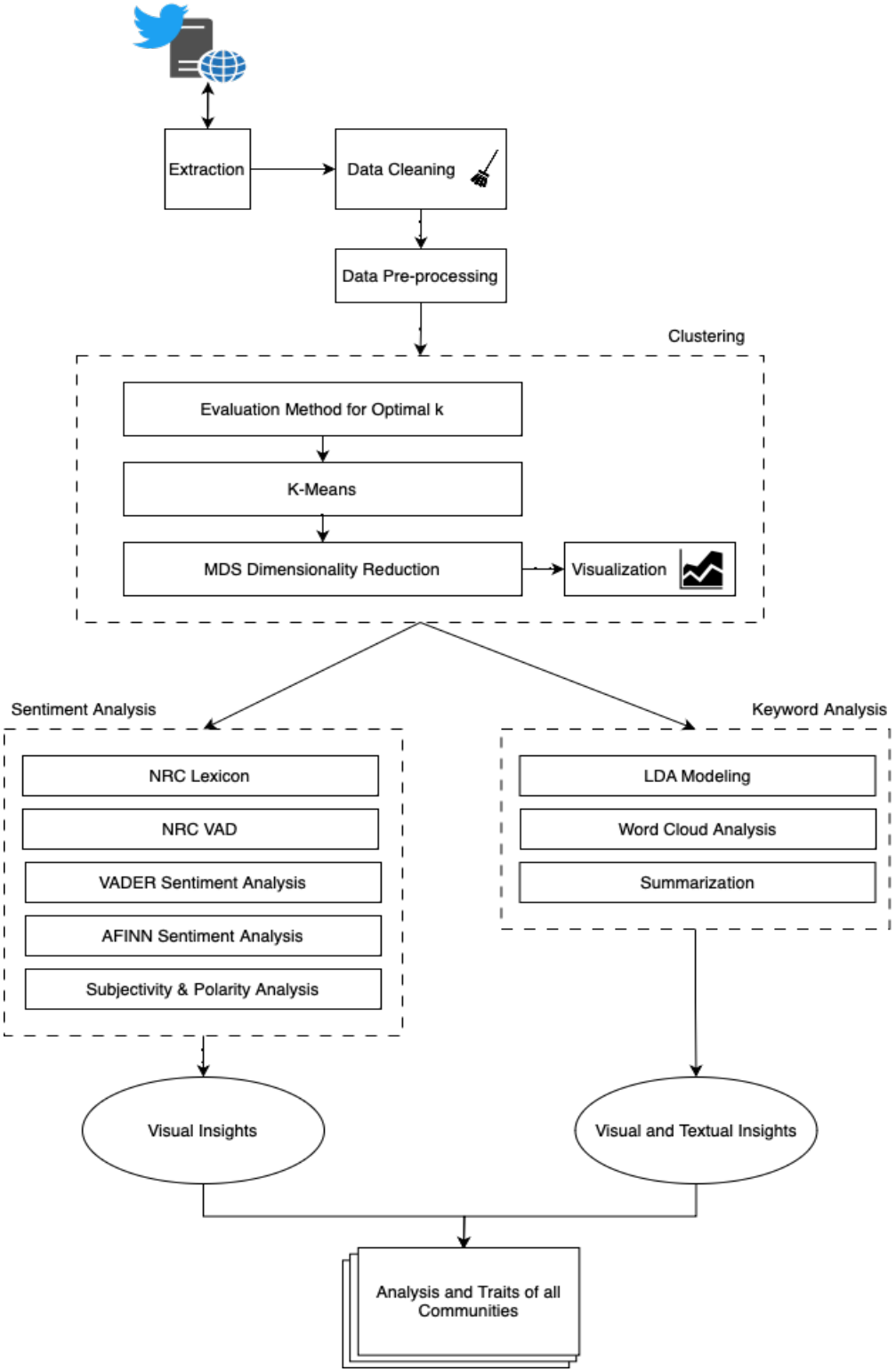


Fig 3.1: Methodology Flowchart

The methodology of our project typically involves three phases which are as followed:
Phase 0:

i) Extraction of Tweets.
ii) Data cleaning.
iii) Data Pre-Processing.

Phase 1 - Clustering:

i) k-means clustering.
ii) Dimensionality Reduction using MDS.

Phase 2 - Sentiment Analysis:

i) NRC Lexicon.
ii) NRC VAD Analysis.
iii) VADER Sentiment Analysis.
iv) AFINN Sentiment Analysis.
v) Polarity and Subjectivity Analysis.

Phase 3 - Key Word Analysis:

i) LDA Topic Modelling.
ii) Word Cloud Analysis.
iii) Summarization.

Phase 0:

3.1 Extraction of Tweets/Data Collection:

We have analysed three communities in our project. The tweets were extracted from twitter for each community using popular hashtags. The following is the list of communities with their hashtags.

i) Technology #Technology #tech #computerscience
ii) Economics #economy #capitalisim
iii) Politics #politics #elections #government

The data has been scraped from 'tweepy'[7] using the official Twitter API. The dataset consists of three columns which are User, Location and Tweets. We have extracted data across USA of about 1500-1600 tweets per community.

```
!pip install tweepy
import tweepy as tw
```

Fig. 3.1.1 : Code Snippet for importing tweepy.

As of May 2023, twitter has revoked our developer access.

	user	location	tweet
0	NASALaserComm	Greenbelt, MD	We're also at #AGU2021, representing @NASASCaN...
1	turntidetech	Sunnyvale, CA	When your technology can transform both busine...
2	TRUEUP90	Royal Oak, MI	@elonmusk Will you send me a Tesla in exchange...
3	Parents_Nook	San Francisco, CA	And the best part is working on a deep problem...
4	BlakeBeilue	Denver, CO	@daheels1 @therealdre_jack @HelmanDC Unless th...

Fig. 3.1.2: Downloaded Technology Dataset

	user	location	tweet
0	KDWS__	Atlanta, GA	@j0rdanr0binson hard disagree, most we would've had to worry about was his tweets. joe biden lac...
1	HairWeave	Philadelphia	@FredHungry @gotcankles @Cosmic_Andrew1 @latimes to be fair, financing whores is a critical part...
2	OneLadyOneVote	USA	WSJ: The Biden Stagflation Is Coming https://t.co/OeDLoLUdJk via @WSJOpinion #USA🇺🇸 #America #1A...
3	punchdownsuckup	United States	@ManFellow2 @TheBaconBrotato @KirstinMorrell @BuilderClaireKY Canada will always get away with u...
4	powanikutin	Bronx, NY	OSHA opens investigation after Amazon warehouse collapses during tornado, killing 6 https://t.co...

Fig. 3.1.3: Downloaded Economics Dataset

	user	location	tweet
0	Dmarlanawilson	KANSAS	Biden admin admits it left hundreds more US ci...
1	RepKristina	Green Bay, WI	I hope your mentorship of new leaders will pro...
2	rogersonkaren	USA	'It has gone too far': Listen to Cheney read o...
3	le_petit_trek	Vulcan, Alberta	@maxfawcett This might be the dumbest thing I ...
4	Plauterborn	San Francisco, CA	People are only conservative until it hits the...

Fig. 3.1.4: Downloaded Politics Dataset

3.2 Data Cleaning:

3.2.1 Remove URLs

This Python function takes a string of text as input and removes any URLs (links) present in the text. The function complies a regular expression using the re module and removes any text starting with "http://" or "https://" or "www." and ending with one or more non-whitespace characters.

```
# Remove url
def remove_url(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'',text)
```

Fig. 3.2.1: Code snippet to remove URL.

3.2.2 Remove punctuation

This Python function takes a string of text as input and removes any punctuation using a translation table “str.maketrans()” and maps each punctuation to None.

```
# Remove punct
def remove_punct(text):
    table = str.maketrans('', '', string.punctuation)
    return text.translate(table)
```

Fig. 3.2.1.1: Code snippet to remove punctuation.

3.2.3 Remove html

This Python function takes a string as input and compiles a regular expression using the re module to match any text between "<" and ">". It uses the sub() method to replace all matched HTML tags with empty string.

```
# Remove html
def remove_html(text):
    html=re.compile(r'<.*?>')
    return html.sub(r'',text)
```

Fig. 3.2.3.1 Code snippet to remove html:

3.2.4 Remove @username

This Python function takes a string of text as input, uses the re module to compile a regular expression pattern that matches any substring that begins with @ and ends with a whitespace character or the end of the string, and replaces all matched substrings with an empty string using the sub() method of the input text string.

```
# Remove @username
def remove_username(text):
    return re.sub('@[^\s]+','',text)
```

Fig. 3.2.4.1: Code snippet to remove @username:

3.2.5 Remove emojis

This Python function takes a string of text as input, compiles a regular expression pattern using the re module to match any Unicode characters in the range of emoji symbols and pictographs, transport and map symbols, flags (iOS), and other miscellaneous symbols, and uses the sub() method of the input text string to replace all matched characters with an empty string.

```
# Remove emojis
def remove_emoji(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)
```

Fig. 3.2.5.1: Code snippet to remove emoji

3.2.6 De-contraction

Contradictions like "isn't" are changed to "is not" throughout this procedure, while symbols like "y'all" are changed to "is". This aids in preparing the tweets for extraction. We want to limit the terms we use to only significant named entities and significant parts of speech tags while analysing the tweets.

```
# Decontraction text
def decontraction(text):
    text = re.sub(r"won\'t", " will not", text)
    text = re.sub(r"won\'t've", " will not have", text)
    text = re.sub(r"can\'t", " can not", text)
    text = re.sub(r"don\'t", " do not", text)

    text = re.sub(r"can\'t've", " can not have", text)
    text = re.sub(r"ma\'am", " madam", text)
    text = re.sub(r"let\'s", " let us", text)
    text = re.sub(r"ain\'t", " am not", text)
    text = re.sub(r"shan\'t", " shall not", text)
    text = re.sub(r"sha\n't", " shall not", text)
    text = re.sub(r"o\'clock", " of the clock", text)
    text = re.sub(r"y\'all", " you all", text)
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"n\'t've", " not have", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'d've", " would have", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'ll've", " will have", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    text = re.sub(r"\'re", " are", text)
    return text
```

Fig. 3.2.6.1: Code snippet for deconstruction text:

3.3 Data Pre-Processing

Data preprocessing is a valuable technique in Data Mining which involves the process of converting raw data into an organized format. This process consists of the following steps -

3.3.1 Remove stop words.

This code is a Python function that takes a string of text as input and removes all English stopwords from it. It returns the processed text as a single string. The function uses the Natural Language Toolkit (NLTK) library to obtain a list of English stopwords and filters out the stopwords from the input text using a list comprehension. The filtered words are then joined back into a single string with a space character separator.

```
import re
from nltk.corpus import stopwords
# Remove stop words
def remove_stopwords(text):
    text = ' '.join([word for word in text.split() if word not in (stopwords.words('english'))])
    return text
```

Fig 3.3.1.1: Code snippet to remove stop word.

3.3.2 Separate Alphanumeric

This code is a Python function which takes a string of text as input and returns the input as individual words in a new string as output by separating the alphanumeric characters in it.

```
# Seperate alphanumeric
def seperate_alphanumeric(text):
    words = text
    words = re.findall(r"[^\W\d_]+|\d+", words)
    return " ".join(words)
```

Fig 3.3.2.1: Code snippet to separate alphanumeric:

3.3.3 Remove contiguous repeated characters.

The first code is a Python function that replaces any contiguous repeated characters in a text string with two occurrences of that character which helps to simplify and standardize the text. The function obtains the matched substring returned by re.sub() method and checks if it has length greater than 1 which in turn returns the first two characters of the substring.

The second code is a Python function named unique_char that takes in two arguments which are rep and text. This function helps to replace any contiguous repeated characters in a text string with a single occurrence of that character.

```
def cont_rep_char(text):
    tchr = text.group(0)

    if len(tchr) > 1:
        return tchr[0:2]
```

Fig. 3.3.3.1: Code snippet for cont_rep_char.

```
def unique_char(rep, text):
    substitute = re.sub(r'(\w)\1+', rep, text)
    return substitute
```

Fig. 3.3.3.2: Code snippet for unique_char.

3.3.4 Lemmatization

This code is a Python function named lemmatize_word takes a string of text as input and performs lemmatization. We have imported “WordNetLemmatizer” module from Natural Language ToolKit (nltk) library. The input is tokenized into separate words using nltk.word_tokenizer(). Then, it uses the list comprehension to lemmatize each word. Finally, the lemmatized words are joined back using the join() method. This process helps in reducing each word to its base form.

```
lemmatizer = WordNetLemmatizer()
def lemmatize_word(text):
    words = nltk.word_tokenize(text)
    lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in words]
    lemmatized_tweet = ' '.join(lemmatized_words)
    return lemmatized_tweet
```

Fig. 3.3.4.1: Code snippet for lemmatization.

3.3.5 Tokenizing

The word_tokenize() function from the Natural Language Toolkit (NLTK) is used to convert a sentence or a piece of text into a list of tokens (words). This code uses the apply() function from pandas library to tokenize the text data. It applies this operation row wise, which is specified using the parameter "axis =1".

```
data['tokenized_sents'] = data.apply(lambda data: nltk.word_tokenize(data['tweet']), axis=1)
```

Fig. 3.3.5.1: Code snippet for Tokenization.

The pre-processed dataset for the three communities is given below.

```
0          [agu, represent, share, benefit, laser, commun...
1          [transform, business, fight, climate, change, ...
2          [send, tesla, exchange, theory, travel, inters...
3          [best, work, deep, problem, satisfy, ignore, c...
4          [substitute, type, android, real, dak, play, w...
                                ...
1675                 [aim, hose, vent, computer, case, unplug]
1676       [idea, trip, bring, work, computer, sound, ghe...
1677       [soo, mini, super, computer, tangible, half, w...
1678       [computer, screen, forward, conversations, syn...
1679       [reach, help, restart, computer, allow, device...
Name: tokenized_sents, Length: 1680, dtype: object
```

Fig. 3.3.5.2: Technology Tokens.

```
0                  [hard, disagree, worry, tweet, joe, biden, lack, competency, example, inflation, stock]
1       [fair, finance, whore, critical, totally, condone, finance, corporate, government, amp, mob, whore]
2       [wsj, biden, stagflation, usa, america, biden, trump, trump, freedom, patriots, democrats, repub...
3       [canada, underfunding, defense, defend, canada, defend, canada, country, neuter, europe, establi...
4                             [osha, open, investigation, amazon, warehouse, collapse, tornado, kill]
                                                    ...
903     [grade, essay, struggle, help, pay, physics, english, essaypay, math, pay, essaydue, history, as...
904     [grade, essay, struggle, help, pay, physics, english, essaypay, math, pay, essaydue, history, as...
905     [help, homework, biology, algebra, literature, essay, pay, finals, essaypay, essay, history, nur...
906     [help, homework, biology, algebra, literature, essay, pay, finals, essaypay, essay, history, nur...
907     [corporations, post, record, profit, american, families, force, pay, higher, food, price, grocer...
Name: tokenized_sents, Length: 908, dtype: object
```

Fig. 3.3.5.3: Economics Tokens.

```
  0          [biden, admin, admit, leave, hundreds, citizen...
  1          [hope, mentorship, leaders, provide, critical,...
  2          [listen, cheney, read, texts, trump, jr, hanni...
  3           [dumbest, thing, read, canadian, politics, well]
  4             [people, conservative, hit, personally, shock]
                                  ...
  894        [man, legal, background, knowledge, laws, bill...
  895        [well, prosecute, assange, espionage, publish,...
  896        [imagine, brainwash, mainstream, media, resort...
  897        [federal, government, radio, airwaves, lease, ...
  898           [kudlow, sure, government, spend, foxbusiness]
  Name: tokenized_sents, Length: 899, dtype: object
```

Fig. 3.3.5.4: Politics Tokens.

3.3.6 Vectorization using TF-IDF

The TF-IDF (Term Frequency-Inverse Document Frequency) statistic measures how essential a term is to a document in a collection or corpus. The vectorization process entails translating text into a numerical representation that machine learning algorithms may employ for analysis.

Why TF-IDF?
This vectorization method is chosen over other vectorization approaches because it aids in identifying the most essential and distinct terms in a document while filtering out common words that offer little significance.TF-IDF additionally mitigates the influence of document length on word frequency, as larger texts may include more instances of a term even if they are not necessarily relevant.

This code creates a TF-IDF vectorizer object with a maximum number of features set to 200,000. It then fits the vectorizer on a list of top words and transforms the tweets in the dataset to a sparse TF-IDF matrix. Finally, it retrieves the feature names for each column in the matrix. The resulting TF-IDF matrix can be used as input to train a machine learning model.

```
tfidf_vectorizer = TfidfVectorizer(max_features=200000)
tfidf_vectorizer.fit(top_words)
tfidf_matrix =tfidf_vectorizer.transform(data.tweet)

feature_names = tfidf_vectorizer.get_feature_names_out()
```

Fig. 3.3.6.1: Code snippet for Vectorization using TFIDF.

```
pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out(), index = data.user)
```

user	abc	ability	absolutely	academy	access	accessible	accord	account	action	actual	...	worth	wow	wrap	write	wrong	yeah	year	years
NASALaserComm	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
turntidetech	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TRUEUP90	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Parents_Nook	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
BlakeBeilue	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
aerohistorian	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N8Talk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
joshhwesterman	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LuisM_Oliveira	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AppleSupport	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.235702	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1680 rows × 1000 columns

Fig. 3.3.6.2: Technology TFIDF matrix.

```
pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out(), index = data.user)
```

user	abudhabigp	academic	accept	acceptable	access	accomplish	accord	account	add	admin	...	writers	writingreliable	wrong	wtf	xlf	yeah
KDWS__	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
HairWeave	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
OneLadyOneVote	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
punchdownsuckup	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
powanikutin	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
BestOnlineWrit4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
EGradners	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
BestOnlineWrit4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
BestOnlineWrit4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
wordpower2018	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

908 rows × 1000 columns

Fig. 3.3.6.2: Economics TFIDF matrix.

```
pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out(), index = data.user)
```

user	abide	ability	abortion	absolutely	accord	accountable	achieve	action	actual	address	...	worry	wow	write	wrong	ya	yeah	year	yea
Dmarianawilson	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
RepKristina	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.277
rogersonkaren	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
le_petit_trek	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
Plauterborn	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Ga21Renee	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
POmi7798	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
tydelrosedrums	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.258199	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
ericowensdc	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
Robinsm86398738	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000

899 rows × 1000 columns

Fig. 3.3.6.3: Politics TFIDF matrix.

Phase 1 - Clustering:

3.4. k-means Clustering

Clustering is an unsupervised learning approach that groups comparable data points into clusters based on their commonalities. K-Means is a well-known clustering technique that divides data into K clusters, where K is a user-specified value. The tweets are clustered based on the similarity between the keywords of the vectorized tweets. Further, clusters are analysed based on the sentiments and keywords of the users.

3.4.1 Evaluation of optimal clusters:

The K-Means approach is widely used for clustering tasks; however, it must first determine the ideal number of groups. The elbow approach and silhouette analysis are two typical assessment methods for identifying the ideal number of clusters in K-Means.

The elbow approach includes charting the sum of squared distances from each location to its allocated cluster center versus the number of clusters. Another approach for determining the ideal number of clusters is silhouette analysis. It compares each data point's similarity to its own cluster to other clusters. The mean intra-cluster distance (distance between a point and other points within the same cluster) and the mean closest-cluster distance (distance between a point and the nearest cluster it does not belong to) are used to compute a silhouette score for each data point.

Here, we go ahead with the Elbow method for determining the optimal K by plotting a graph for the 10 clusters to determine a drop in the SSE. The within-cluster sum of squares (SSE) for various cluster counts is computed and shown versus cluster count. The ideal number of clusters is determined by the number of clusters with the greatest decrease in SSE. The scikit-learn library's KMeans class is used to conduct K-means clustering with varying numbers of clusters. To ensure repeatability, the number of clusters is set to 1 and the random state is set to 0. A line plot is used to depict the outcome.

```
SSE_list = []
for i in range(1, 10):
    kmeans_sse = KMeans(
        n_clusters=i, init='random', max_iter=300,
        tol=1e-04, random_state=0
    )
    kmeans_sse.fit(tfidf_matrix)
    SSE_list.append(kmeans_sse.inertia_)
plt.plot(range(1, 10), SSE_list, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.show()
```

Fig. 3.4.1.1: Code snippet for Elbow method.

This code snippet performs silhouette analysis to determine the optimal number of clusters for K-means clustering. The silhouette score is calculated for different numbers of clusters and printed for each iteration. The KMeans class from the scikit-learn library is used to perform K-means clustering with different numbers of clusters. The range of the number of clusters is from 2 to 9. The silhouette score measures how similar an object is to its own cluster compared to other clusters.

```
silhouette = []
for n_clusters in range(2,10):
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(tfidf_matrix)
    score = silhouette_score(tfidf_matrix, preds)
    silhouette.append(score)
    print("For n_clusters = {}, silhouette score is {})".format(n_clusters, score))
```

Fig. 3.4.1.2: Code snippet for silhouette analysis.

Here, K-means clustering with a specified number of clusters, 5 in this case. The KMeans class from scikit-learn is used to perform the clustering. The 'init' parameter specifies the initialization method, 'random' in this case. The maximum number of iterations is set to 100 and the number of times the algorithm will be run with different centroid seeds is set to 1 using the 'n_init' parameter. The clustering is applied to the tf-idf matrix and the resulting cluster labels are assigned to each tweet in the form of a list using the 'labels_' attribute of the KMeans object.

```
num_clusters = 5

km = KMeans(n_clusters=num_clusters, init='random', max_iter=100, n_init=1)

km.fit(tfidf_matrix)
clusters = km.labels_.tolist()
```

Fig. 3.4.1.3: Code snippet for K-means clustering.

Below is the list of top terms per cluster for each community:

Technology	Economy	Politics
Top terms per cluster: Cluster 0: science essay help people pay Cluster 1: computer work time day phone Cluster 2: security source hack cybersecurity cyber Cluster 3: high school science bartow boys Cluster 4: start meet women computer market	Top terms per cluster: Cluster 0: essay pay javascript biology physics Cluster 1: essay pay paper class biology Cluster 2: amp understand money work biden Cluster 3: essay psychology finance homework class Cluster 4: inflation people money years americans	Top terms per cluster: Cluster 0: government people overthrow state work Cluster 1: trump meadows jan news gop Cluster 2: politics people talk agree news Cluster 3: money government tax pay jan Cluster 4: time government people rebel control

Fig. 3.4.1.4: Top terms per cluster as per community

3.5 Dimensionality Reduction Using MDS[13]:

To handle the higher dimensionality issue in the TF_IDF matrix we also define an error term for distance matrix using cosine similarity. Cosine similarity is used here because it is generally used for measuring how similar the documents are irrespective of their size. Since we need to reduce the dimension of the matrix, we take the error matrix produced from subtracting 1 from the cosine similarity of the TF-IDF matrix.

```
dist = 1 - cosine_similarity(tfidf_matrix)
```

Fig. 3.5.1: Code snippet for cosine- similarity.

For aiding the visualization of the structure of the dataset, this error matrix is helpful because it can help reduce the dimension in some way. To help solve this dimensionality problem the simple way would be to take a random projection of the data. Although it provides for certain visibility of such data structure, the choice's random nature offers a lot to be desired. But this random projection makes the interesting part of the data to be lost.

So now, using this error matrix we apply Multidimensionality scaling method (MDS) which aims for a low-dimensional representation of the data in which the distances in the original high-dimensional space are well preserved.

```
MDS()
mds = MDS(n_components=2, dissimilarity="precomputed", random_state=1)
pos = mds.fit_transform(dist)
xs, ys = pos[:, 0], pos[:, 1]
```

Fig. 3.5.2: Code snippet for Multidimensionality scaling method (MDS)

Phase 2 - Sentiment Analysis:
3.6 NRC Lexicon[9]:

The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The Sentiment and Emotions include different manually created and automatically created lexicons. Available in 40 different languages, it has 14,182 unigrams (words), 25,000-word senses.[6]

At first the lexicon from a file is loaded into a variable, and each line is iterated the extracts each word, the emotion, scores after stripping blank spaces and stored in a object named “data_emotion” is made.

```
data_emotion = open("/content/sample_data/NRC-Lexicon.txt")
#data_emotion = open("NRC-Lexicon.txt")
next(data_emotion)
emotion_lexicon = {}
for line in data_emotion:
    word, emotion, score = line.strip().split("\t")
    if word in emotion_lexicon:
        emotion_lexicon[word].update({emotion: float(score)})
    else:
        emotion_lexicon.update({word: {emotion:float(score)}})
```

Fig. 3.6.1: Code Snippet to prepare data for NRC Lexicon.

For each tweet in “top_tweets”, a zero initialized of the length of the number of emotions is created and index of tweet is calculated. For each word of cleaned tweet from “store_tech_clusters”, if the word appears in the data_emotion dictionary then, for all emotions its score is updated in the ‘temp’ list and then added to the ‘emotion_score’ list. Finally, when all words are processed, the score is added to “top_tweets_el” and cumulative scores are stored in “res_list”.

```
emotions = list(emotion_lexicon['kill'].keys())

# computing emotional scores for tweets
# top tweets index and score = top_tweets
# Non Normalized here
res_list = [0]*len(emotions)
top_tweets_el = []
#top_tweets_ad = []
#top_tweets_dv = [](
for tweet in top_tweets:
    emotion_score = [0]*len(emotions)
    index = tweet[0]
    clean_tweet = store_tech_clusters[0][index]
    #print("tweet: ",clean_tweet)
    for each_word in clean_tweet.split():
        if each_word in emotion_lexicon:
            temp = []
            for each_emotion in emotion_lexicon[each_word]:
                temp.append(emotion_lexicon[each_word][each_emotion])
            emotion_score = [temp[i] + emotion_score[i] for i in range(len(emotions))]
    top_tweets_el.append(emotion_score)
    res_list = [emotion_score[i] + res_list[i] for i in range(len(emotions))]
```

Fig. 3.6.1: Code Snippet for NRC Lexicon.

3.7 NRC – VAD[10]:

National Research Council VAD is a model that is used to classify tweets based on three parameters namely Valance, Arousal and Dominance.

Valance refers to the positivity of the tweet. It typically measures on a scale, the higher the value the positive the tweet is and vice-a-versa. A negative score is a representation of emotions like sadness, anger, fear, and a positive one is a representation of emotions like joy, happiness, love.

Arousal refers to the level of excitement, or energy associated with the tweet. A lower value of arousal indicates the tweet's nature is calmed and relaxed, whereas a higher value indicates that the tweets are excited/agitated.

Dominance refers to the degree of power that an individual has over, emotional state. It reflects the extent to which an individual perceives themselves as overseeing their emotions, rather than being controlled by them.

At first VAD lexicon from a file is loaded into a variable, and each line is iterated the extracts each word, the valance, dominance, arousal scores after stripping blank spaces and a dictionary named “vad” is made.

```
# Sentimental analysis using Valance Arousal and Dominace Data

# loading the file into a dictionary
data_lexicon = open("/content/sample_data/NRC-VAD-Lexicon.txt")
#data_lexicon = open("NRC-VAD-Lexicon.txt")
next(data_lexicon)
vad = {}
for line in data_lexicon:
    word, var, aro, dom = line.strip().split("\t")
    vad[word] = [float(var),float(aro),float(dom)]
```

Fig. 3.7.1: Code Snippet for load NRC VAD.

Then, the valance-arousal, arousal-dominance and dominance-valence scores for each word are calculated for each word in the top tweets list. For each word its index and text are extracted then for each word of the cleaned tweet, it is checked if that word belongs in the list of words and scores extracted earlier and the scores are updated in the corresponding list. Once all the words in a tweet are processed, the average scores are calculated for each word in the tweet. After all the words in all the tweets are processed, the results are added to the valance-arousal, arousal-dominance, and dominance-valence result lists. This process is repeated for the remaining four clusters.

```
# calculating the valance arousal and dominance scores
# top tweets index and score = top_tweets
top_tweets_va1 = []
top_tweets_ad1 = []
top_tweets_dv1 = []
for tweet in top_tweets:
    index = tweet[0]
    clean_tweet1 = store_tech_clusters[0][index]
    #print("tweet: ",clean_tweet)
    count = 0
    val_score1 = 0
    arousal_score1 = 0
    dominance_score1 = 0
    for each_word in clean_tweet1.split():
        if each_word in vad:
            val_score1 += vad[each_word][0]
            arousal_score1 += vad[each_word][1]
            dominance_score1 += vad[each_word][2]
            count += 1
    # averaging the score over the words in tweets to get appropriate scores for each tweet
    if count!= 0:
        top_tweets_va1.append([val_score1/count, arousal_score1/count])
        top_tweets_ad1.append([arousal_score1/count, dominance_score1/count])
        top_tweets_dv1.append([dominance_score1/count, val_score1/count])
```

Fig. 3.7.2: Code Snippet for calculating NRC Lexicon for one cluster.

3.8 VADER[11] Implementation:

Vader (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool specifically designed for analyzing sentiment. VADER analyses sentiments primarily based on key points such as punctuation, capitalization, degree modifiers and conjunctions, which enables it to precisely predict the text's sentiments.

The below code performs sentiment analysis using the VADER tool on a dataframe column called 'tokenized_sents' which contains pre-processed tokenized sentences. The resulting sentiment scores for each word in the tokenized sentences are added to a new dataframe column called 'vader_scores'. It creates an instance of the SentimentIntensityAnalyzer class from the NLTK library. This analyzer is then used to calculate sentiment scores for individual words in each tokenized sentence. The apply() method is then used to apply a lambda function to each element in the 'tokenized_sents' column of the dataframe. The lambda function applies the polarity_scores() method of the SentimentIntensityAnalyzer class to each word in the sentence, generating a dictionary containing sentiment scores for each word. The resulting list of sentiment score dictionaries is assigned to a new column called 'vader_scores' in the dataframe, providing sentiment scores for each word in the tokenized sentences. Finally, the last snippet calculates the mean sentiment score for each cluster in a dataframe named 'data' and stores the results in a list called 'mean_cluster_polarity'. For each cluster, the mean sentiment score for all rows with that cluster label is calculated using the mean() method on the 'agg_vader_score' column. This mean score is then appended to the 'mean_cluster_polarity' list.

```
analyzer = SentimentIntensityAnalyzer()
data['vader_scores'] = data['tokenized_sents'].apply(lambda x: [analyzer.polarity_scores(word) for word in x])
```

Fig. 3.8.1: Code Snippet to load VADER Analyzer.

```
def agg_vader_score(data2):
  return sum([data2[i]['compound'] for i in range(len(data2))])/len(data2)
temp =[]
for i in range(len(data)):
  temp.append(agg_vader_score(data['vader_scores'][i]))
```

Fig. 3.8.2: Code Snippet for calculating VADER Value.

```
mean_cluster_polarity = []
for i in range(0,5):
  #print(i, data['agg_vader_score'][(data.cluster == i)].mean())
  mean_cluster_polarity.append(data['agg_vader_score'][(data.cluster == i)].mean())
print(mean_cluster_polarity)
```

Fig. 3.8.3: Code Snippet for calculating VADER Values for each cluster.

3.9 AFINN[12] Implementation

AFINN is a word list-based sentiment analysis tool that assigns a score to words based on their sentiment polarity.
The above code calculates the mean sentiment score for each cluster in a DataFrame named 'data', using the Afinn sentiment analysis tool. an instance of the Afinn class is created, which is used to calculate sentiment scores for individual words in each tokenized sentence. Then, the apply() method is used to apply a lambda function to each element in the 'tokenized_sents' column of the dataframe. The lambda function applies the score() method of the Afinn class to each word in the sentence, generating a list of sentiment scores for each word. A function called 'agg_afinn_score' is defined, which takes a list of sentiment scores as input, and returns the average score for the list. For each

cluster, the mean sentiment score for all rows with that cluster label is calculated using the mean() method on the 'agg_afinn_score' column. This mean score is then appended to the 'mean_cluster_polarity' list. The resulting list of mean sentiment scores for each cluster is printed to the console.

```
afinn = Afinn()

data['afinn_scores'] = data['tokenized_sents'].apply(lambda x: [afinn.score(word) for word in x])
```

Fig. 3.9.1: Code Snippet to load AFINN Analyzer.

```
def agg_afinn_score(data2):
  return sum([data2[i] for i in range(len(data2))])/len(data2)

temp = []
for i in range(len(data)):
  temp.append(agg_afinn_score(data['afinn_scores'][i]))
```

Fig. 3.9.2: Code Snippet for calculating AFINN Value.

```
mean_cluster_polarity = []
for i in range(0,5):
  #print(i, data['agg_vader_score'][(data.cluster == i)].mean())
  mean_cluster_polarity.append(data['agg_afinn_score'][(data.cluster == i)].mean())
print(mean_cluster_polarity)
```

Fig. 3.9.3: Code Snippet for calculating AFINN Values for each cluster.

3.10 Polarity & Subjectivity:

The polarity score typically ranges from -1 to 1, where negative scores indicate negative sentiment, positive scores indicate positive sentiment, and scores closer to zero indicate a more neutral sentiment. The subjectivity score typically ranges from 0 to 1, where scores closer to 0 indicate more objective language and scores closer to 1 indicate more subjective language.[5]

In our project, polarity and subjectivity scores are relevant in understanding the sentiment and subjective nature of tweets within a particular community. The code below first creates two dictionaries, polaritycluster and subjectivitycluster, where each key is a unique cluster number from the cluster column of a dataframe.

Next, the code loops through each dataframe and calculates the polarity and subjectivity of all tweets in that dataframe using the TextBlob library and stores the resulting values in the corresponding dictionary for the relevant cluster.

Therefore, this code is used to calculate the polarity and subjectivity scores for each cluster, based on the tweets contained within each cluster's dataframe. The resulting scores can then be used to compare the sentiment and subjectivity of different clusters, providing insight into the attitudes and beliefs of the communities within each cluster.

```
UniqueNames = data['cluster'].unique()
```

Fig. 3.10.1: Code snippet for storing unique names.

```
polaritycluster={elem : pd.DataFrame for elem in UniqueNames}
subjectivitycluster={elem : pd.DataFrame for elem in UniqueNames}
for i in DataFrameDict.keys():
    polaritycluster[i]=TextBlob(' '.join(DataFrameDict[i]['tweet'].astype('str'))).sentiment.polarity
    subjectivitycluster[i]=TextBlob(' '.join(DataFrameDict[i]['tweet'].astype('str'))).sentiment.subjectivity
```

Fig. 3.10.2: Code snippet for calculating polarity & subjectivity.

Phase 3 – Keyword Analysis:
3.11 LDA[8] Topic Modelling

We conduct a keyword analysis on the clusters for each topic (technology, economics, and politics). The first method we use to do so is topic modeling. Topic modeling is an unsupervised model that uses statistics to uncover the implicit structure within a collection of texts. In specific we use a generative probabilistic model, Latent Dirichlet Allocation (LDA), a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities.

Our LDA model is made using the gensim library for Python. First, we create a dictionary of all documents using corpora. Dictionary() module. Second, we create a corpus based on the term-document frequency. Next, we calculate the coherence score of minimum 1 to maximum 10 topics through 50 iterations. The coherence score calculates the degree of semantic similarity between high-scoring words in the topic, and we choose the highest score as our final number of topics. The code used to calculate the coherence score and visualize the result is below.

```
# Compute coherence score
number_of_topics = []
coherence_score = []
for i in range(1,10):
  lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                              id2word=id2word,
                                              iterations=50,
                                              num_topics=i)
  coherence_model_lda = CoherenceModel(model=lda_model,
                                       texts=data_tech.loc[mask, 'tokenized_sents'],
                                       dictionary=id2word,
                                       coherence='c_v')
  coherence_lda = coherence_model_lda.get_coherence()
  number_of_topics.append(i)
  coherence_score.append(coherence_lda)

# Create a dataframe of coherence score by number of topics
topic_coherence = pd.DataFrame({'number_of_topics':number_of_topics,
                                'coherence_score':coherence_score})

# Print a line plot
sns.lineplot(data=topic_coherence, x='number_of_topics', y='coherence_score')
```

Fig. 3.11.1: Code Snippet to compute coherent score.

After we finalize the number of topics, we use that number to figure out specific topics for each cluster. Each topic contains weighted keywords that influenced the LDA model, and we will use

these keywords to describe groups of users in each cluster. The specifics of the LDA model are illustrated in the following screenshot: We iterate the model 100 times with a chunk size 10. The number of topics is manually modified according to the coherence score.

```
# Run the LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=n_topics,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=10,
                                            passes=10,
                                            alpha='symmetric',
                                            iterations=100,
                                            per_word_topics=True)
```

Fig. 3.11.2: Code Snippet to run LDA model.

We visualize the topic modeling result using pyldavis library. We draw the inter-topic distance map via multidimensional scaling and display top 30 salient terms. It is a responsive visualization so when one hovers their mouse over to each topic, we can examine each topic and their keywords. The code for visualization is below.

```
# Import and enable notebook to run visualization
pyLDAvis.enable_notebook()

vis = pyLDAvis.gensim_models.prepare(lda_model,
                                     corpus,
                                     dictionary=lda_model.id2word)
vis
```

Fig. 3.11.3: Code Snippet to run visualization.

3.12 Word Cloud Analysis

To analyze the keywords in each community (technology, economics, and politics), we used word clouds. By creating a word cloud for each cluster in each community, we were able to identify the dominant words in each cluster, which helped us understand the unique characteristics of each cluster.

```
from wordcloud import WordCloud
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator


for i in range(len(store_economy_clusters)):
  print("For Cluster in economy: \n",i+1)
  word_string=listToString(np.unique(np.hstack(store_economy_clusters[i])).tolist())
  wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110, background_color="white").generate(word_st
  plt.figure(figsize=(10, 7))
  plt.imshow(wordcloud, interpolation="bilinear")
  plt.axis('off')
  plt.show()
  print("\n")
```

Fig. 3.12.1: Code snippet to generate word cloud.

3.13 Summarization

In the final stage of our analysis, we performed text summarization for each cluster by selecting the five most important sentences from the entire set of tweets within each cluster. We utilized the same clean data set from our previous analysis for two reasons. First, we wanted to ensure that the summarization results were consistent with our previous findings by using the same data set. Second, we aimed to eliminate any noise that might have affected the summarization outcome, such as emojis or stop words. To create a single document for each cluster, we concatenated all the tweets within each cluster. We then used the Sumy library LexRank to perform the text summarization, as it is suitable for summarizing multiple documents. We had some similar tweets in the original data set, such as repeated advertisements, which resulted in duplicated sentences in a summarization outcome. To avoid this issue, we summarized a document containing seven tweets and selected five un-duplicated tweets.

```
#written by Gina Roh
pd.set_option('display.max_colwidth',100)

data_sum_econ_copy = data_sum_econ.copy()

#concatenate all the tweets within each cluster to form a single document for each cluster
data_sum_econ_copy['clean_tweet'] = data_sum_econ_copy.groupby(['cluster'])['clean_tweet'].transform(lambda x : '. '.join(x))
data_sum_final = data_sum_econ_copy.drop_duplicates(subset=['clean_tweet'])
data_sum_final
```

Fig. 3.13.1: Code snippet to make one document with multiple tweets

```
#using Luhn
summarizer_luhn = LuhnSummarizer()
#summarize the document with seven sentence
summary_1 =summarizer_luhn(parser.document,7)

#The first five unduplicated sentences are printed
num = 0
sum_result = []
for sentence in summary_1:
  if str(sentence) not in sum_result:
    sum_result.append(str(sentence))
    num += 1
  if num == 5:
    break

# Wrap this text
wrapper = textwrap.TextWrapper(width=100)
word_list = wrapper.wrap(text=" ".join(sum_result))

# Print each line
for element in word_list:
    print(element)
```

Fig. 3.13.2: Code snippet for text summarization.

One limitation of summarizing tweets is that the sentences in the result may have less cohesiveness since each tweet is typically about a different subject and is written by a different person. This makes the dataset less cohesive compared to a document that was written by a single person on a single subject, such as a news article. Additionally, when creating a single document from multiple tweets, we concatenated them in time-order. However, in summarization, where each sentence is placed is considered, so the result may vary depending on how the multiple tweets are concatenated. Despite this limitation, we were still able to obtain meaningful summarization results.

4. Results

4.1 K-means Clustering

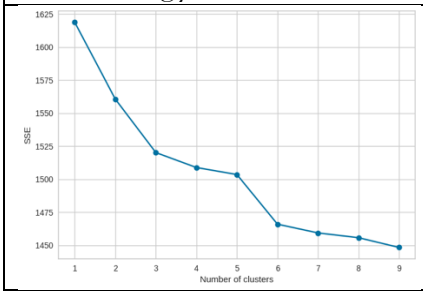
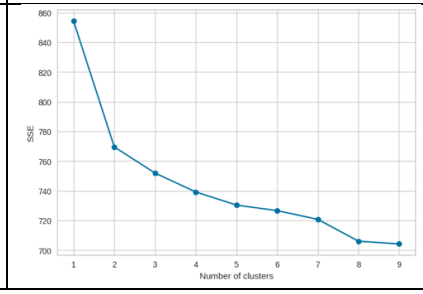
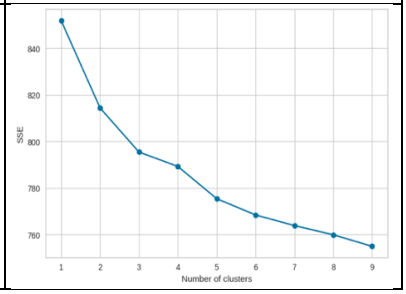
Technology	Economics	Politics
		

Fig. 4.1.1: Elbow graph for all communities

We use the Elbow method to determine the optimal number of k here. The method gets its name from the resulting plot, which typically looks like an arm with an elbow. The point of the elbow on the plot represents the optimal number of clusters, as it indicates the point where the SSE begins to level off and the improvement in clustering performance decreases significantly with additional clusters.

Technology	Economics	Politics
0.0297	0.0836	0.0430
0.0398	0.0420	0.0507
0.0403	0.0421	0.0585
0.0410	0.0548	0.0381
0.0419	0.0674	0.0448
0.0425	0.0621	0.0404
0.0424	0.0705	0.0503
0.0416	0.0750	0.0472
Avg - 0.0399	**Avg – 0.0622**	**Avg – 0.0466**

Table 4.1.1: Silhouette scores for the communities and their average.

For the technology community, the elbow gets formed at k=5. The silhouette analysis also gives the same observation. Between k=3,4,5 we can infer that k = 5 is the optimal no. of clusters for the technology community because good cluster has about a silhouette above 0.4 which is the average silhouette value here. Also, the cluster densities are optimized.
Therefore, both the analysis states the same for k = 5.

For the economics community, there is a sharp drop at k=2 and then it eventually falls steadily and forms a slight elbow at k=5. The silhouette value also is well above 0.4 as per the average silhouette value and clusters appear to be well defined.

For the politics community, there is a sharp drop at k=4 but from k=5 the SSE graph starts looking linearly reducing. The silhouette value is well above the threshold of average silhouette value of 0.45. Hence, we take into consideration k=5 which makes the cluster densities optimized.

4.2 Clustering after MDS Dimensionality Reduction.

These graphs represent the clustering of the three communities using the k-means algorithm with 5 clusters. Each color represents a different cluster, and the points indicates the tweets by the users in each cluster. The clusters are based on the similarity of the keywords in the users' tweets, with users in the same cluster potentially having similar interests, preferences, and sentiments.

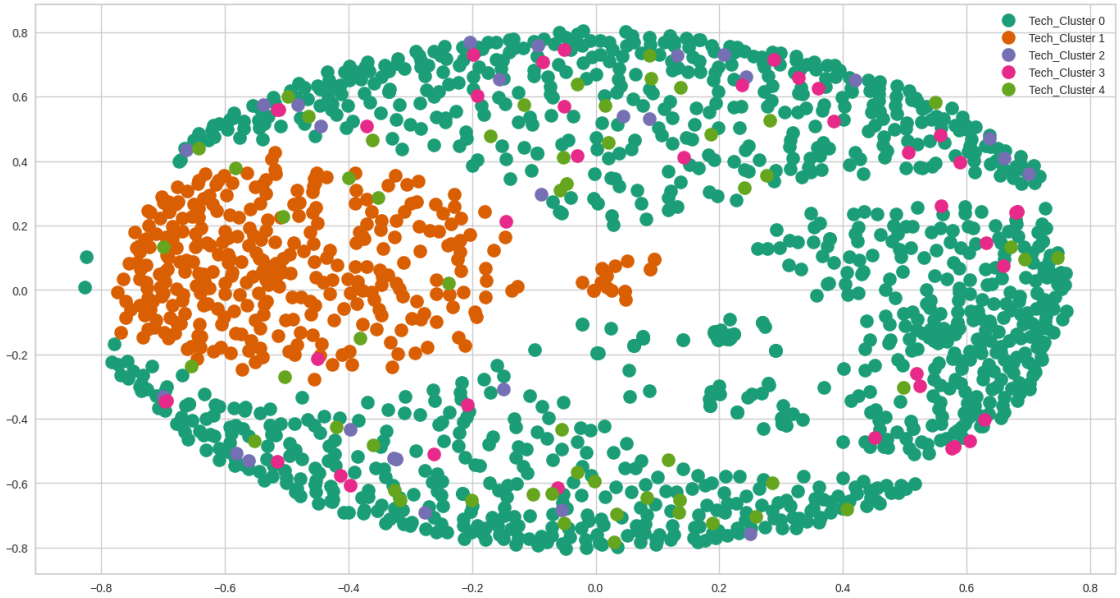


Fig. 4.2.1: Technology Community cluster after MDS.

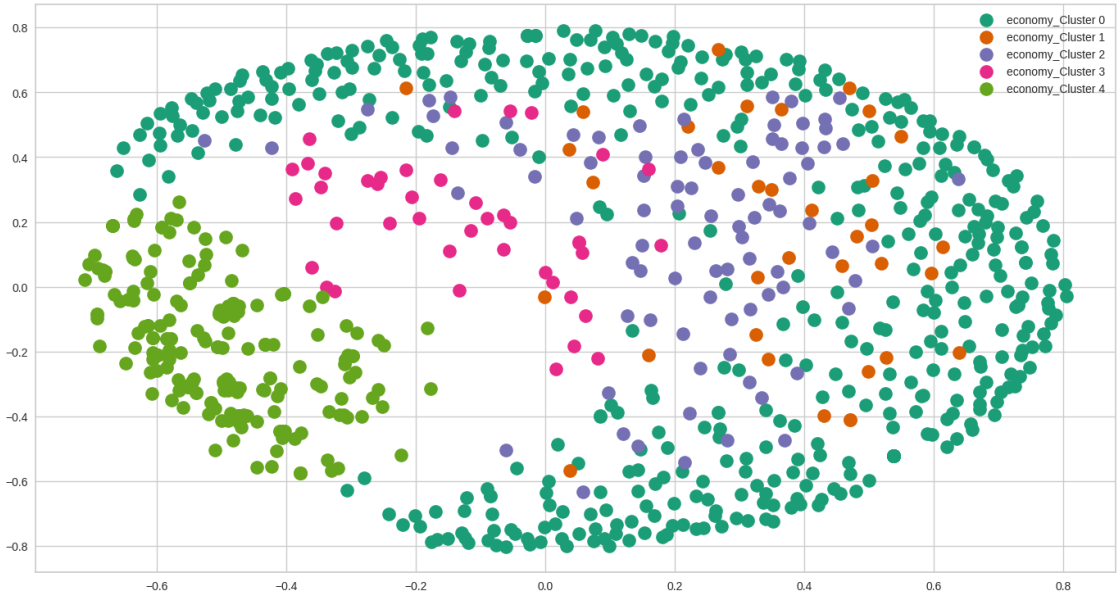


Fig. 4.2.2: Economics Community cluster after MDS.

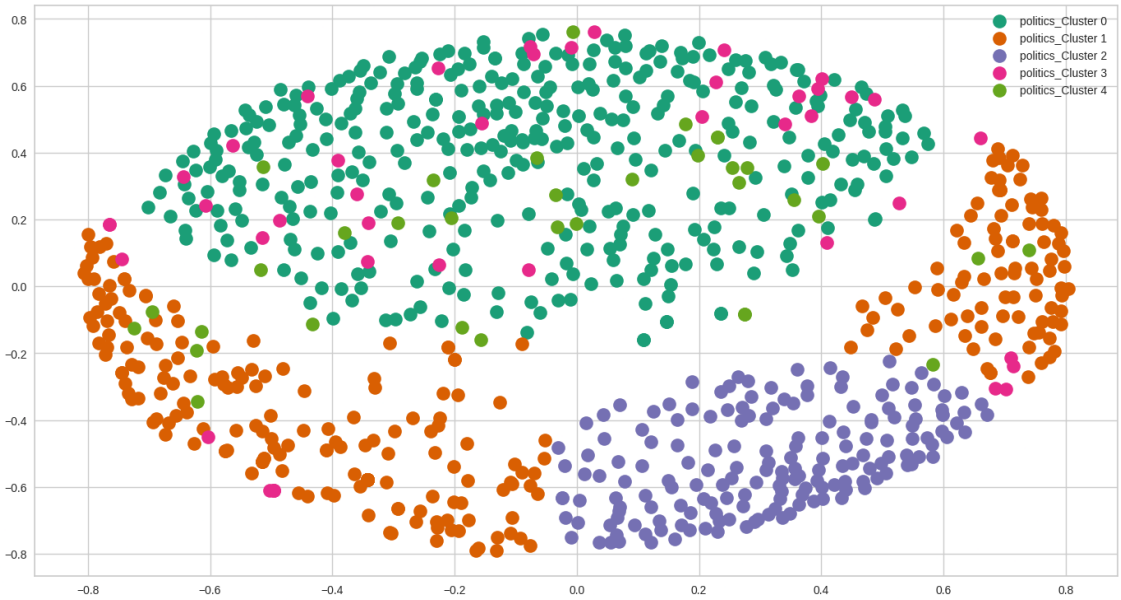
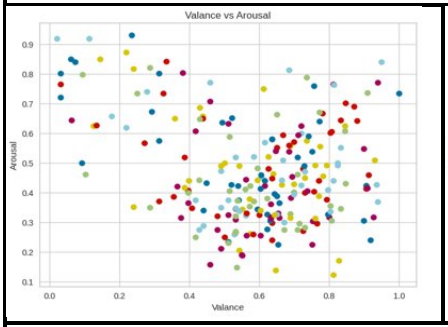
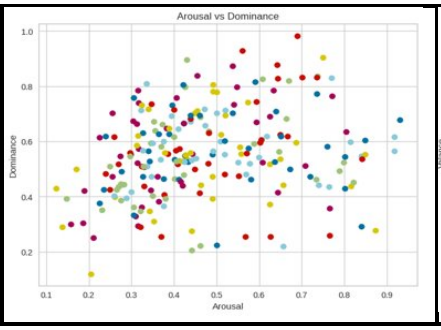
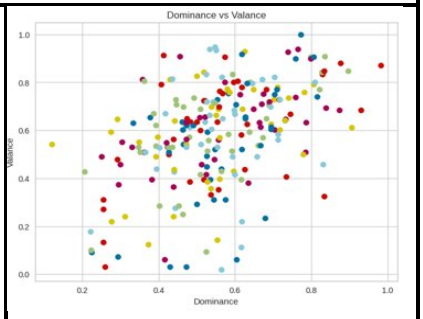
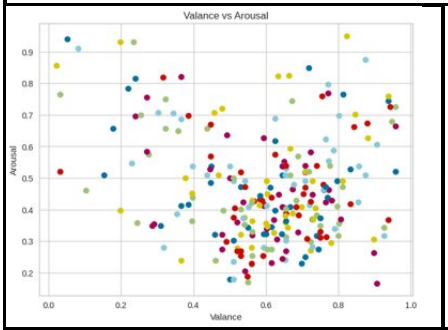
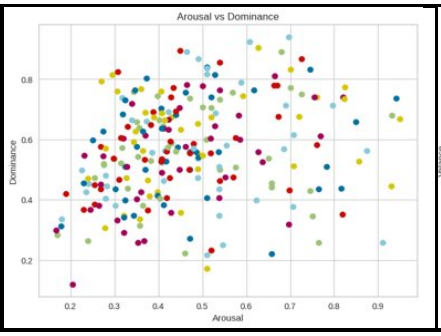
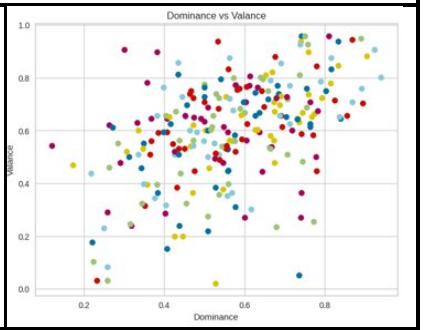


Fig. 4.2.3: Politics Community cluster after MDS.

4.3 NRC VAD – Technology Community

Valance Vs Arousal	Arousal Vs Dominance	Dominance Vs Valance
Cluster 1		
		
Cluster 2		
		
Cluster 3		

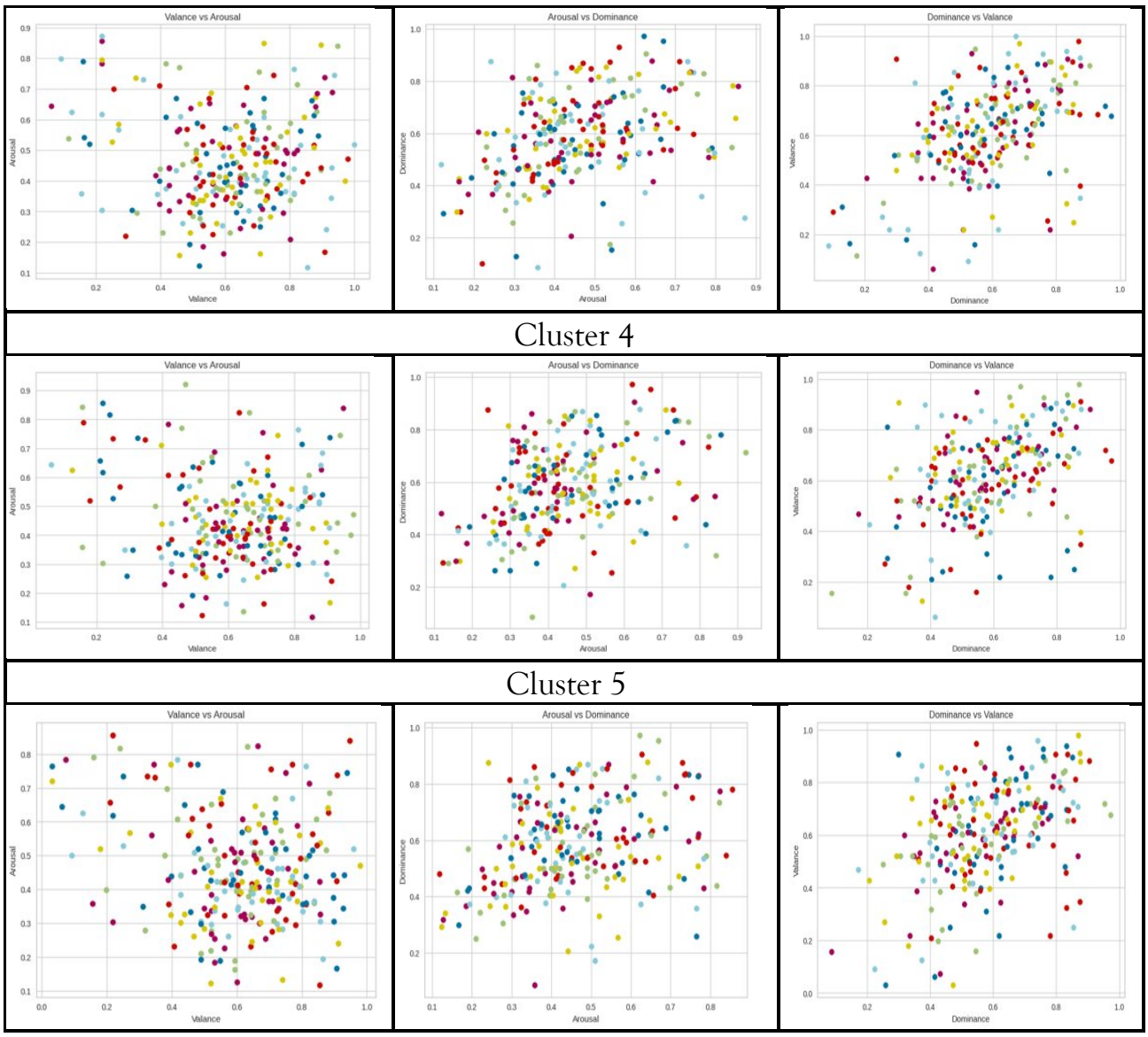


Fig. 4.3.1: NRC VAD result for all clusters of Technology Community.

Upon analyzing the above table, it is evident that, from the Valance vs Arousal graphs across all the five clusters, the tweets contain a mostly positive sentiment, very few tweets having below 0.5 on the valance scale and mostly concentrating after the 0.5 mark and being consistent after that, this suggests that the overall tone of the tweets is optimistic. If we compare the Dominance axis, i.e., y-axis Arousal vs Dominance and x-axis on the Dominance vs Valance graphs, we can clearly see that the tweets are neither extremely submissive nor extremely dominant, are mostly distributed along their respective axes concentrating around the halfway mark, indicating that they do not contain any excessive expressions of power or subordination. This suggests that the authors of these tweets were trying to convey their thoughts in neither an assertive nor a passive way. Furthermore if we look along the Arousal axis which is the y-axis on the Valance vs Arousal graph and the x-axis on the Arousal vs Dominance axis, then it is evident that, although the tweets are spread along the Arousal axis, the distribution lies more concentrated on the lower side with a few sparse points on the extreme positives (left side for x-axis and top side for y-axis), which makes the tone of the tweets on

the very clam side. This indicates that the language used is not emotionally charged or intense, and the tweets are merely statements of fact or opinion. From the graphs that we have we can make a firm assumption that the authors of the tweets are more informing about a particular subject/topic rather than forcing an idea or influencing or even trying to influence other twitter users. In summary, the analyzed tweets appeared to be well-balanced, positive, and informative, with a relatively neutral emotional tone.

4.4 NRC Lexicon - Technology Community

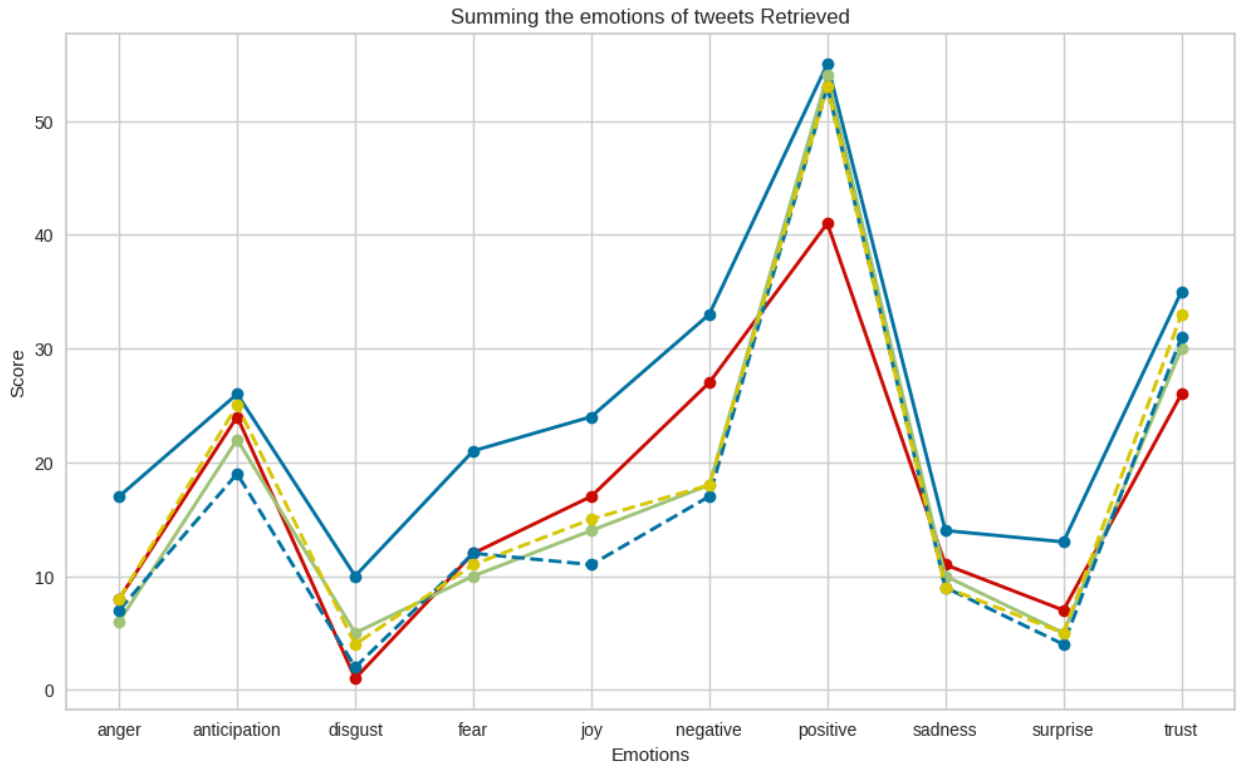


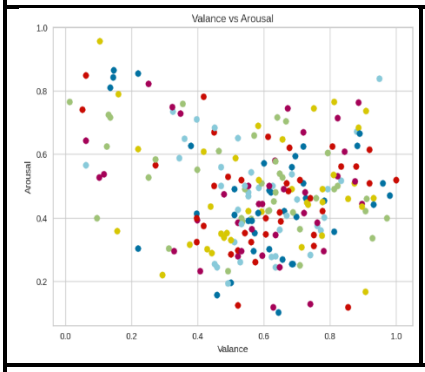
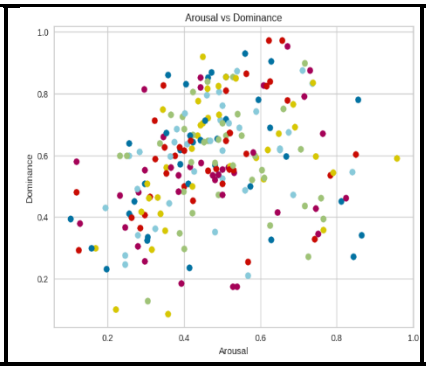
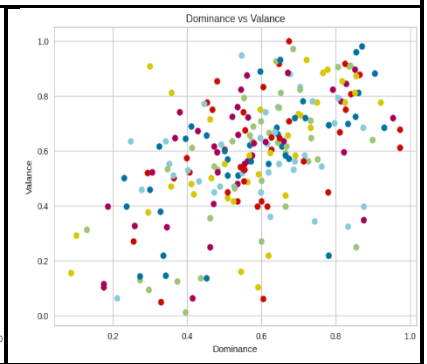
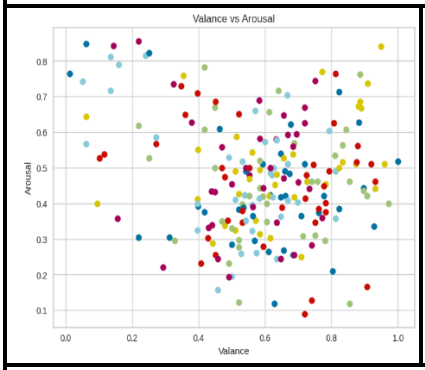
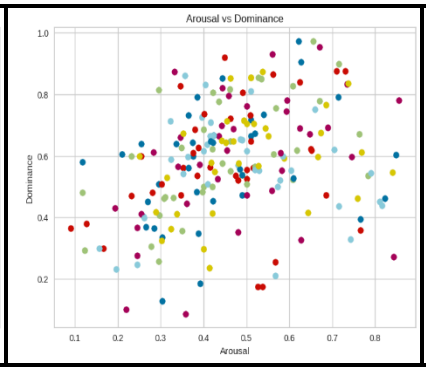
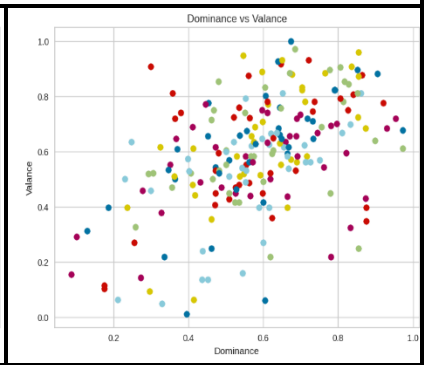
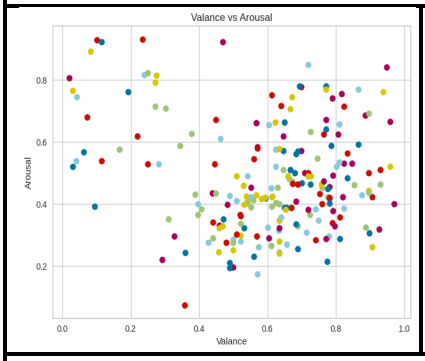
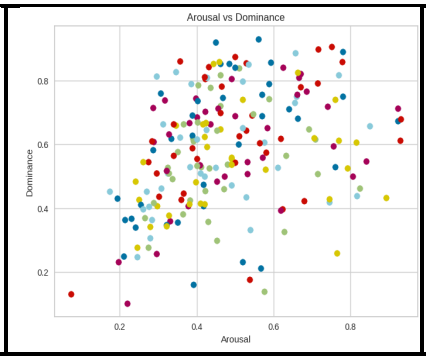
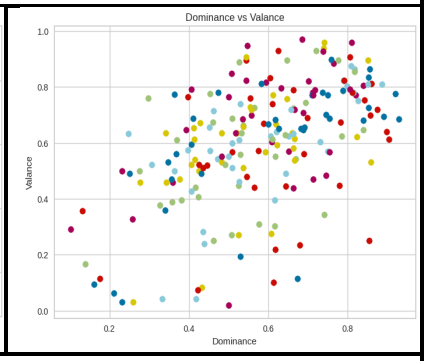
Fig. 4.4.1: NRC Lexicon for all clusters of Technology Community.

We further validated the results of NRC VAD by using NRC Lexicon. By applying NRC lexicon to the dataset used to generate the NRC VAD in short, we can conclude that result in the earlier analysis were consistent with the dataset and accurate.

The results after running this analysis indicate, the tweets are mostly positive, as indicated by the peak score for the positive emotion. This analysis confirms that the tweets are not particularly exciting as the case with the arousal values of VAD analysis. Both the first and the second statement combined give us an understanding that the data/tweets are optimistically favorable and not intense or emotionally charged.

Both results of both the VAD analysis and the NRC lexicon analysis together suggest that the tweets are positive and relatively neutral in terms of emotional intensity. This can be useful in understanding the attitudes and opinions expressed in the tweets and providing insights into the emotional impact of the language used.

4.5 NRC VAD – Economy Community

Valance Vs Arousal	Arousal Vs Dominance	Dominance Vs Valance
Cluster 1		
		
Cluster 2		
		
Cluster 3		
		
Cluster 4		

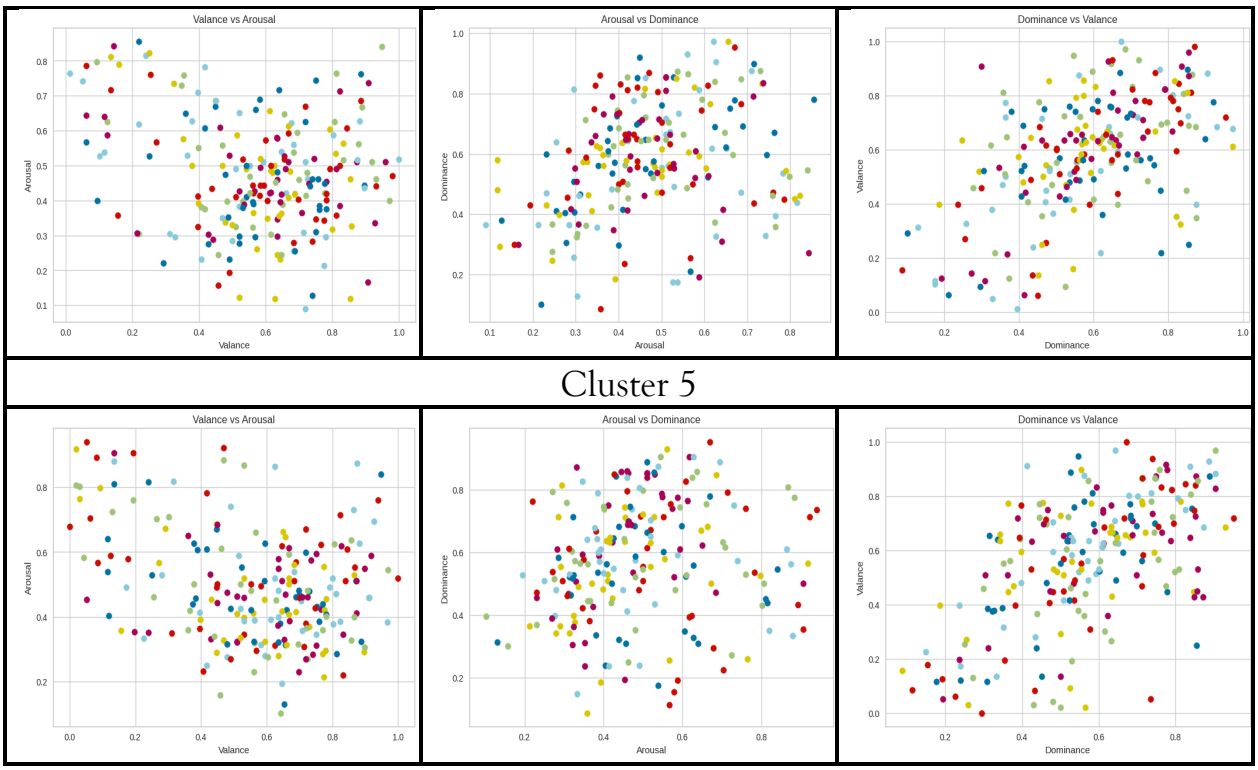


Fig. 4.5.1: NRC VAD result for all clusters of Economics Community.

From the above table that represents the NRC VAD graphs for all the clusters of the Economics cluster, it is clearly seen that the tweets belonging to the economic community have a higher degree of valance. This is due to the enormous number of emotions present in the tweets as seen in the graph below and the lack of consistency in sentiments. If we look at the Valance vs Arousal graph and the Arousal vs Dominance graph, the tweets are spread across the middle of the arousal axis. This indicates they are neutral, neither too exciting nor calm. This suggests that the language used in the tweets is not too emotional or charged, but it conveys a sense of neutrality and balance. Furthermore, there is little dispersion along the arousal axis (y-axis of the Valance vs Arousal graph and x-axis of Dominance Arousal graph), this is an indication that the tweets are consistent in level of excitement or arousal. Despite the overall neutrality in the language used, we found that most tweets are dominant in nature. This suggests the authors of these tweets are expressing strong opinions, which may be the reason for the higher variance in the data.

4.6 NRC Lexicon - Economy Community

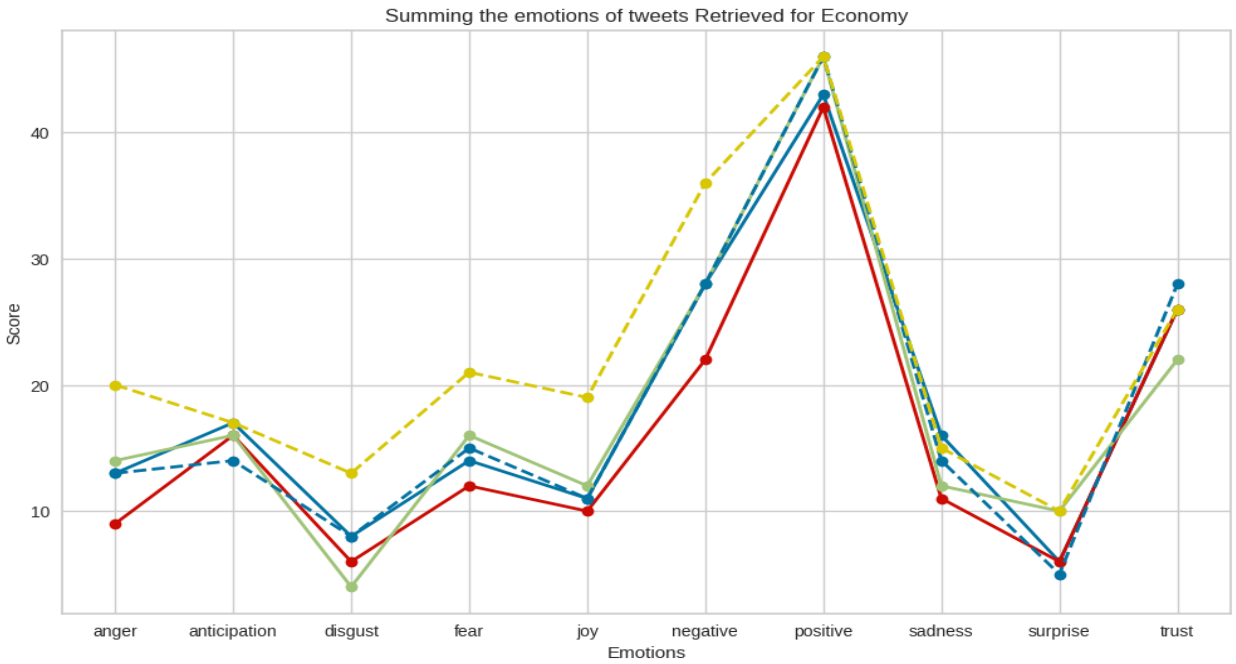
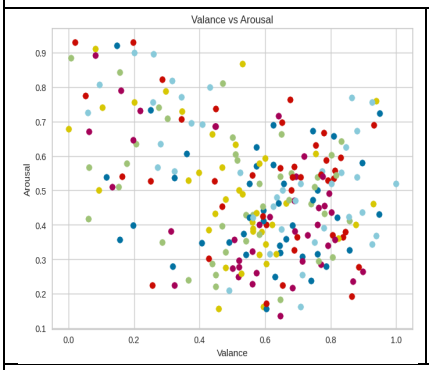
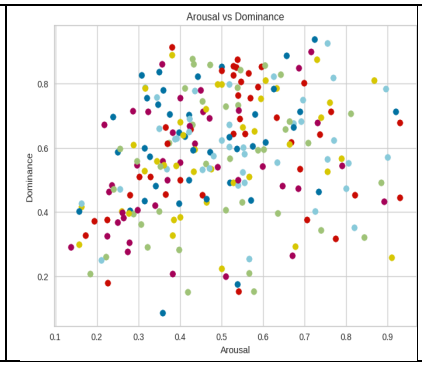
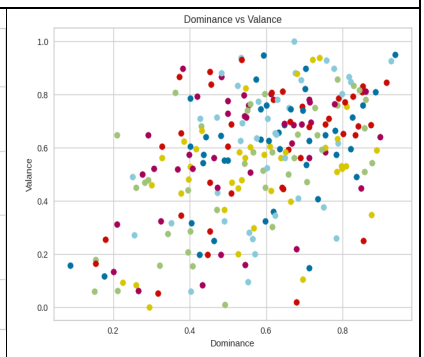
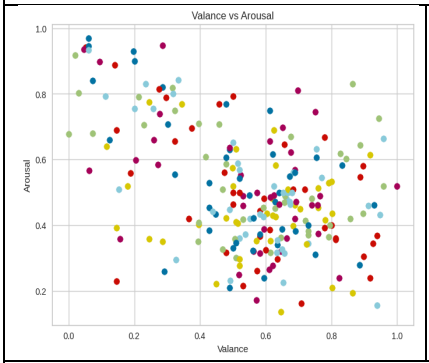
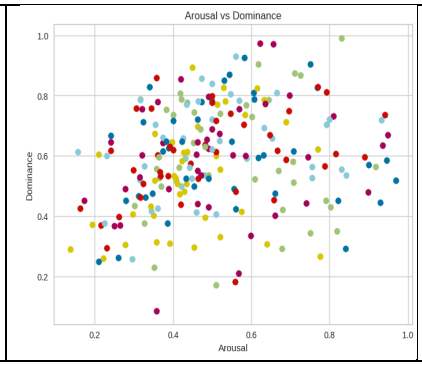
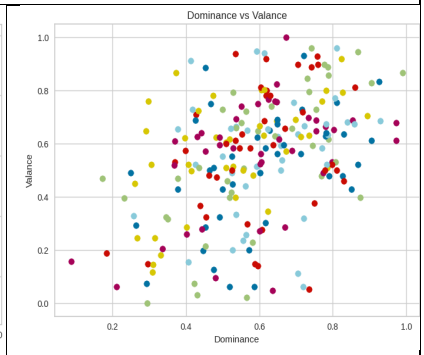
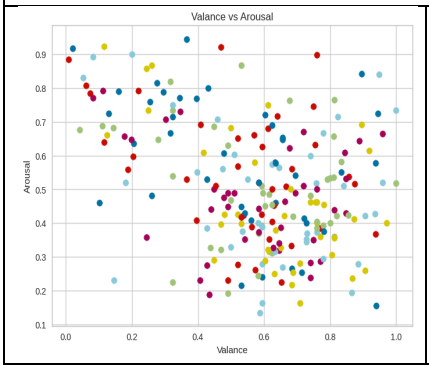
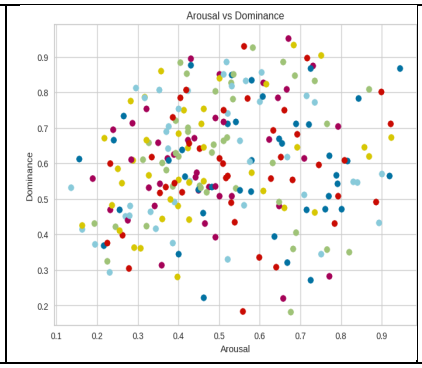
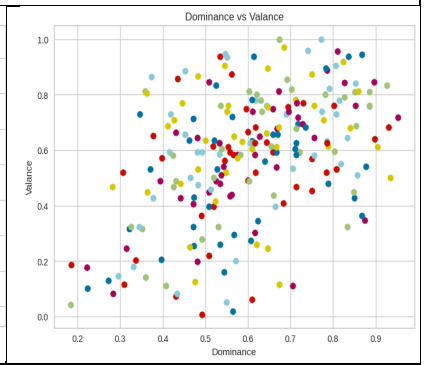


Fig. 4.6.1: NRC Lexicon result for all clusters of Economics Community.

The NRC lexicon analysis was used to verify that the tweets in the economics cluster and at first glance can be said that they are generally positive in sentiment. This was also supported by the anticipation scores, if we compare the anticipation score of the technology community and the economics community, the score of the economics community was at a slightly lower level. But, when examined closely, it was observed that the anticipation scores in the economics cluster were lower due to the distribution of tweets along the arousal axis. Specifically, the tweets in the economics cluster were found to be less varied in terms of their level of arousal or excitement, with most of them being distributed in the middle of the arousal axis. This lack of variability in the arousal axis likely contributed to the lower anticipation scores in the economics cluster, as anticipation is often associated with high levels of arousal or excitement. In contrast, the technology clusters may have had a greater distribution of tweets along the arousal axis, leading to higher anticipation scores. Overall, while the NRC lexicon analysis supported the initial finding that tweets in the economics cluster are positive, the lower anticipation scores were likely influenced by the distribution of tweets along the arousal axis.

4.7 NRC VAD - Politics Community

Valance Vs Arousal	Arousal Vs Dominance	Dominance Vs Valance
Cluster 1		
		
Cluster 2		
		
Cluster 3		
		
Cluster 4		

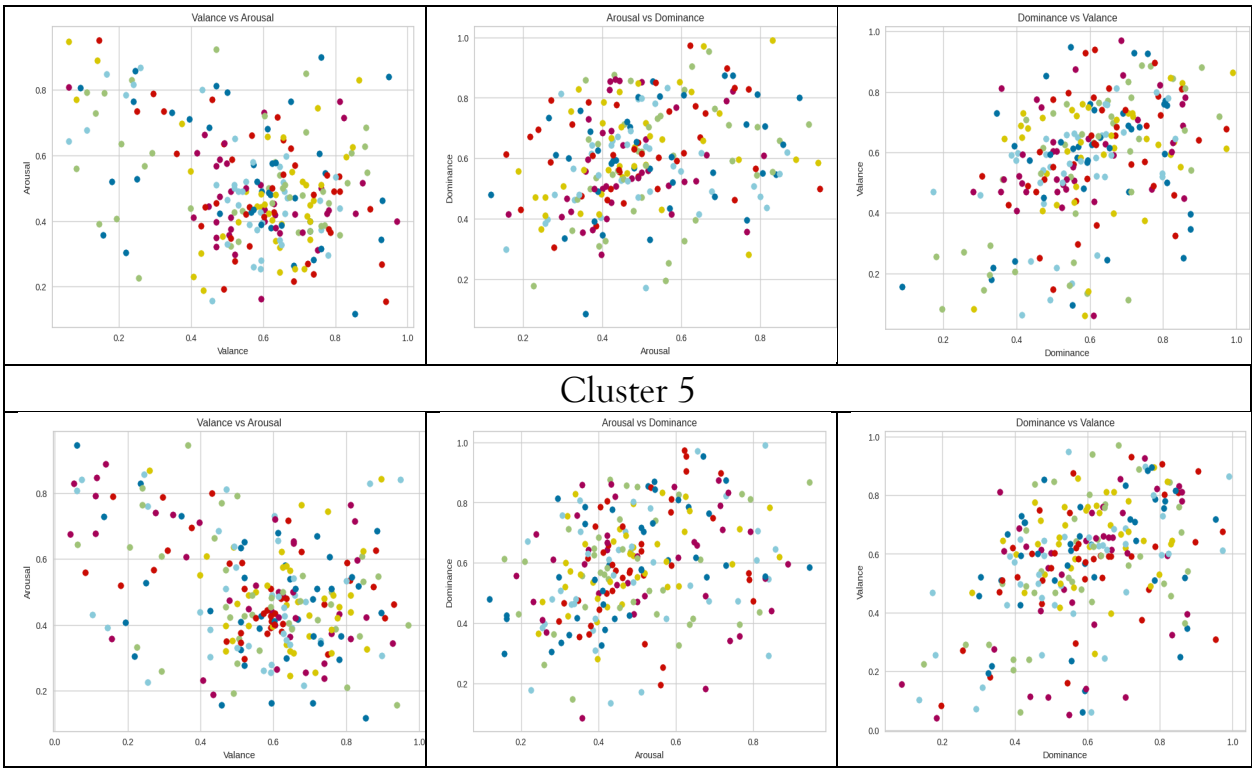


Fig. 4.7.1: NRC VAD result for all clusters of Politics Community.

When we analyze the clusters belonging to the politics community, from the valance axis in the Valance vs Arousal and Arousal vs Dominance graph, the tweets have a positive sentiment with most tweets concentrated above the mid of the axis. A quick look at the arousal axis reveals that, the tweets range from calm to exciting with being evenly distributed along the axis, this indicated that the emotions in the tweets consist of a good mix. Looking at the Dominance axis in the Arousal vs Dominance and Dominance vs Valance graph, the tweets within this community lie on the dominant side of the graph which indicates that strong opinions or beliefs are expressed in the tweets. So, we can say that the Politics cluster is characterized by a generally positive sentiment, with a mix of emotions expressed and a significant presence of dominant language.

4.8 NRC Lexicon - Politics Community

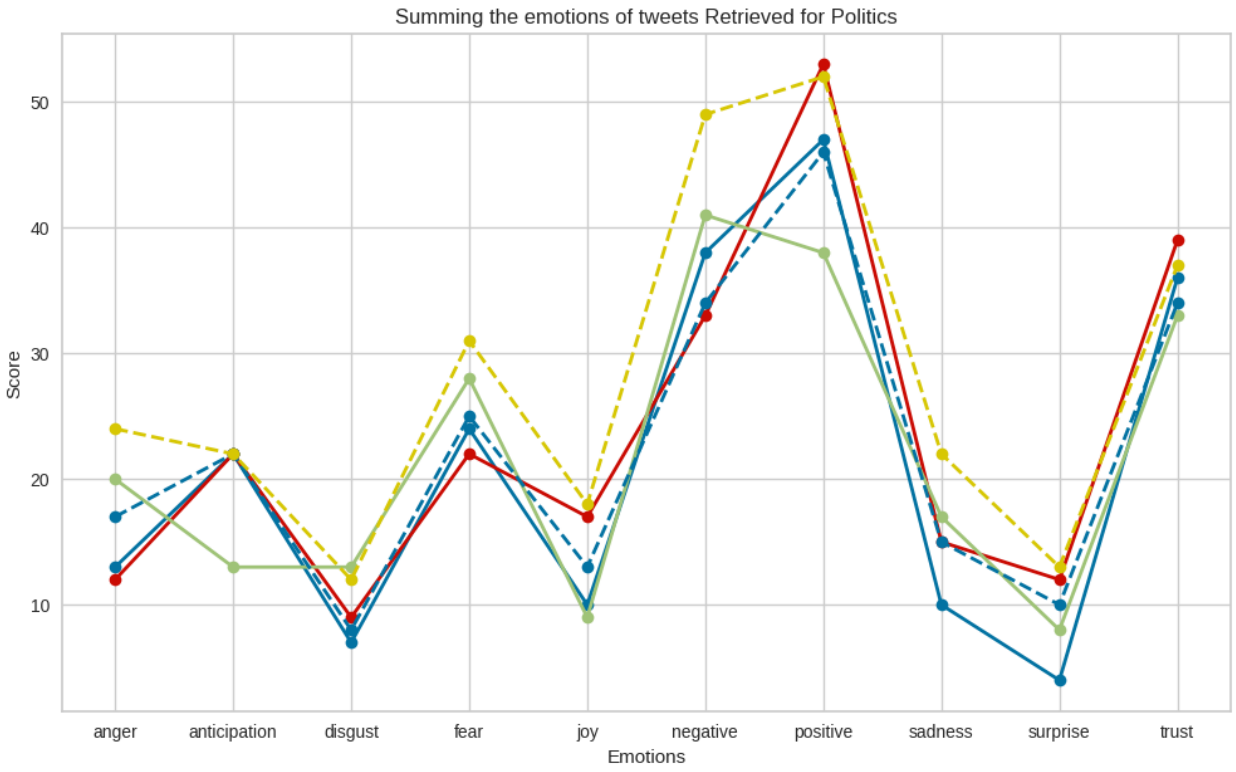
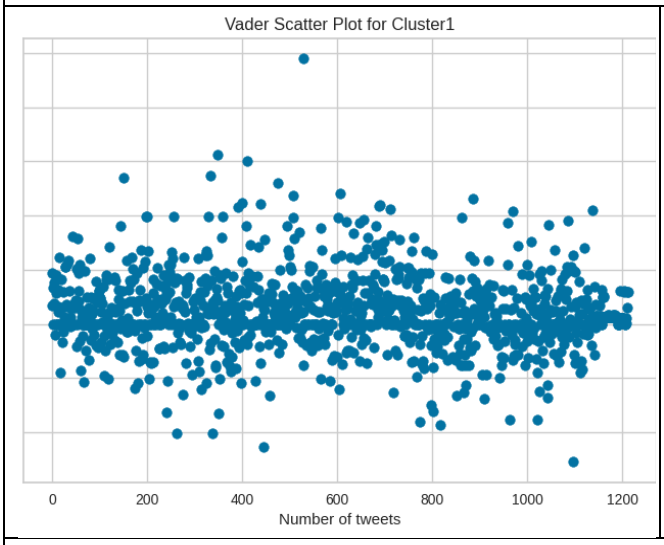
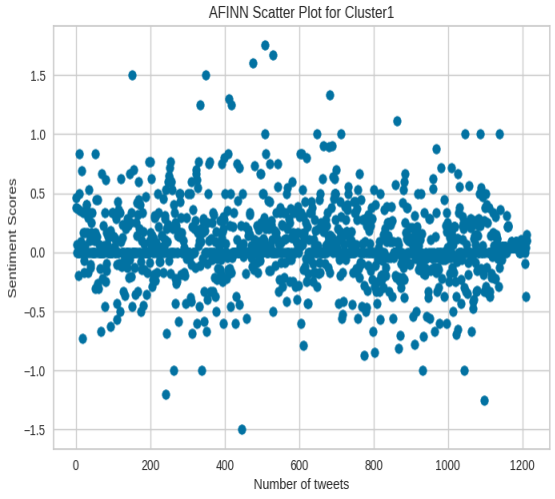
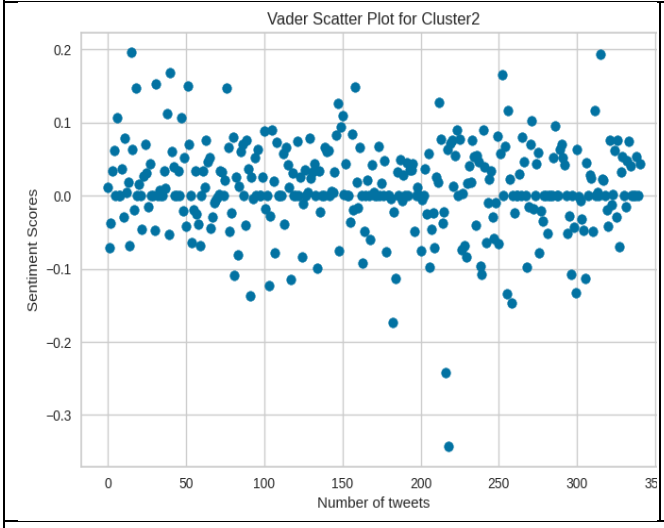
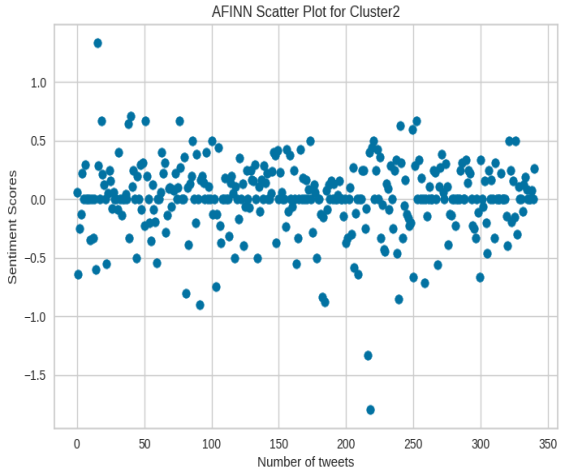


Fig. 4.5.1: NRC Lexicon result for all clusters of Politics Community.

Generating the NRC lexicon analysis on the Politics clusters, it was found that though the tweets in this cluster were positive, there was a lower amount of joy expressed in them as compared to other positive emotions of the Economics or Technology community. Apart from this dip, there was some elevated presence of anger and fear sentiments within these clusters.

It was observed that the tweets of Economics and Politics community had a higher negative value as compared to Technology community. This suggests that the tweets within these clusters were although positive, they also contained a significant amount of negative sentiment, this might be the reason for the overall higher negative value. It is worth noting the fact that the elevated negative sentiment present in these tweets may be due to the polarizing nature of the topics being discussed in the Politics and Economics community, which often tend to express strong, extreme emotions and opinions. Overall, the NRC lexicon analysis suggests that while the Politics and Economics clusters contained mostly positive tweets, there was a notable presence of negative sentiment within these clusters.

4.9 VADER & AFINN – Technology Community

VADER	AFINN
Cluster 1	
	
Cluster 2	
	
Cluster 3	

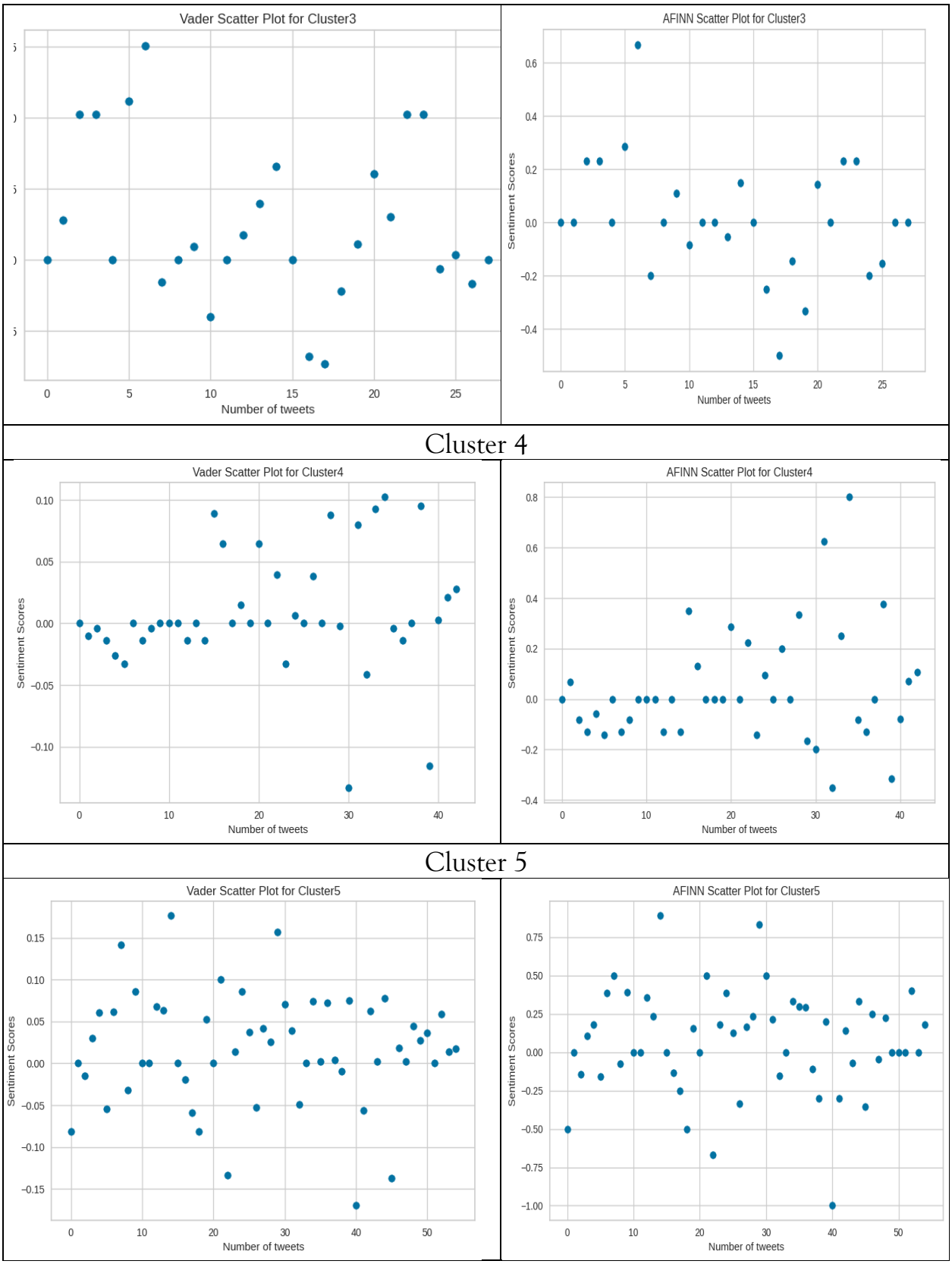
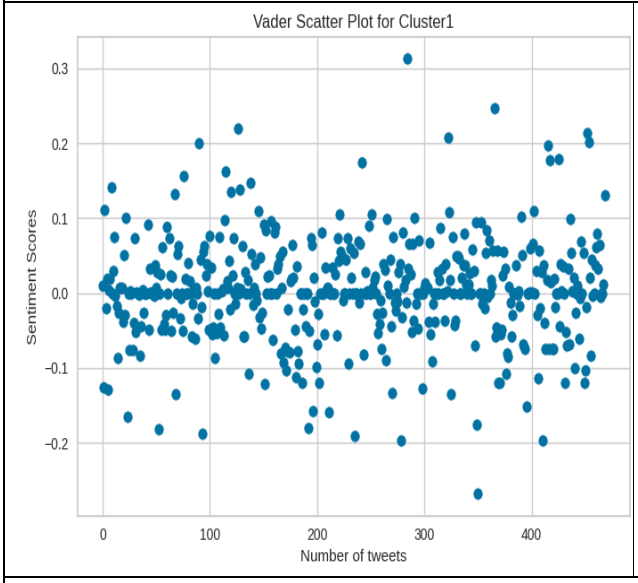
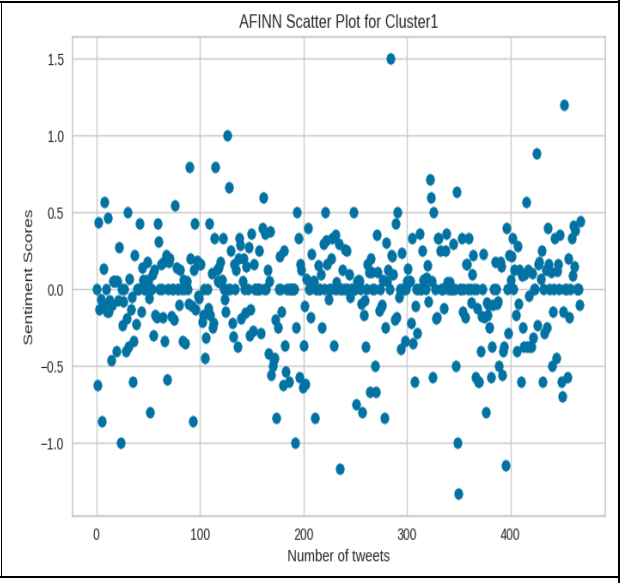
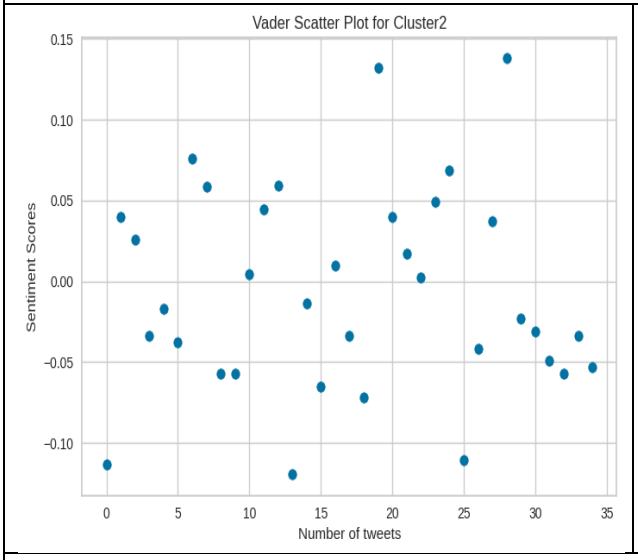
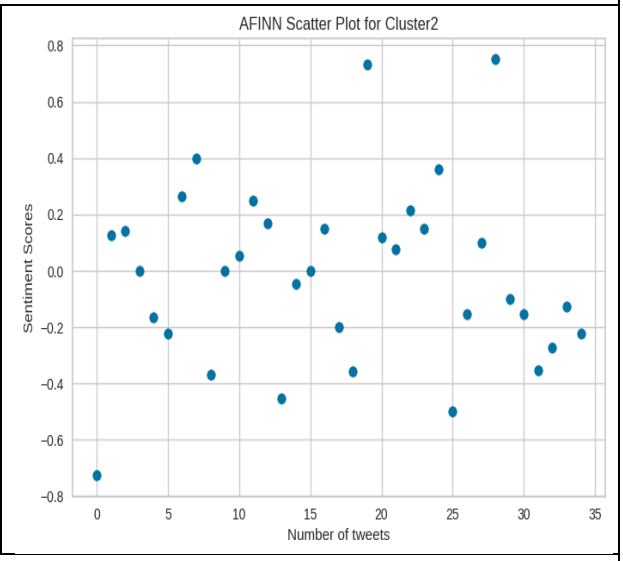


Fig. 4.9.1: VADER & AFINN for all clusters of Technology Community

In the Technology community, after VADER analysis, 2 clusters (1 and 2) are densely populated while the other three clusters (3,4,5) are sparsely populated. Also, a lot of data points are scattered all over and have slight inclination towards positive side. But particularly in cluster 1 and 2, we can observe that there are points that are scattered in the negative side as well. This can be verified with the AFINN analysis which gives out similar results since both algorithms are lexicon-based tools. Thus, we can conclude that overall, the tweets are inclined towards positive sentiments.

4.10 VADER & AFINN – Economics Community

VADER	AFINN
Cluster 1	
	
Cluster 2	
	
Cluster 3	

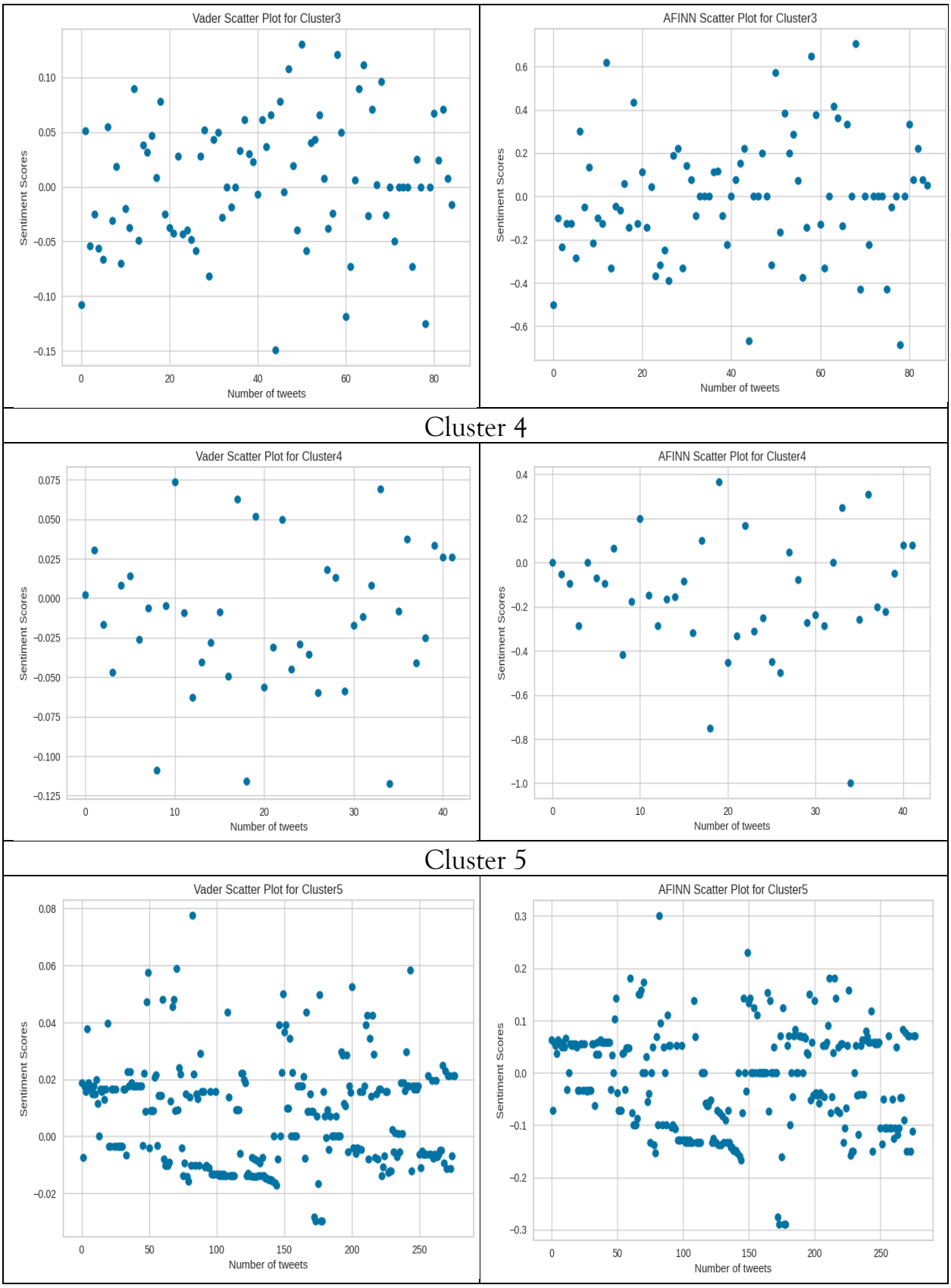
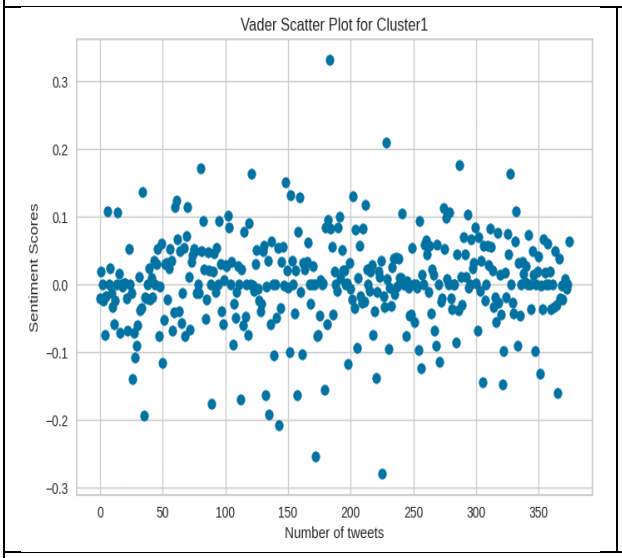
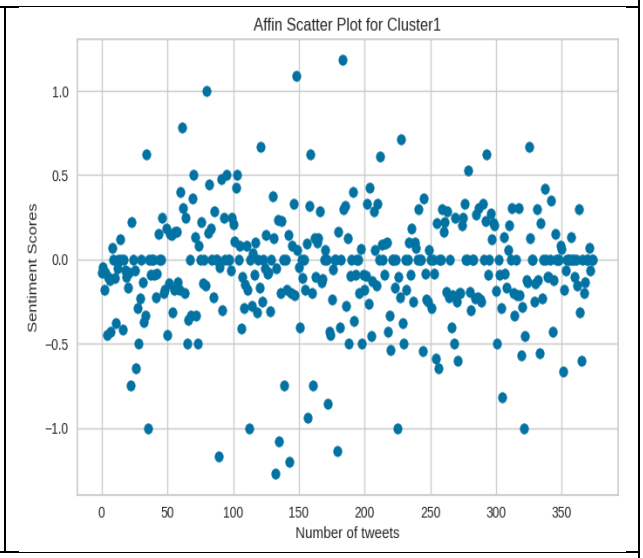
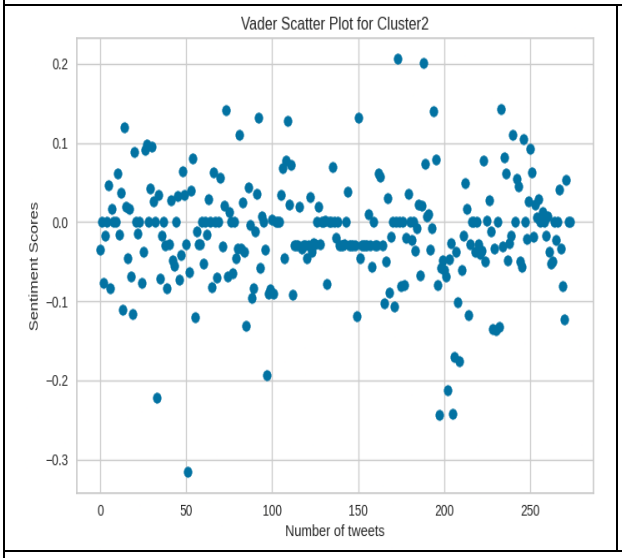
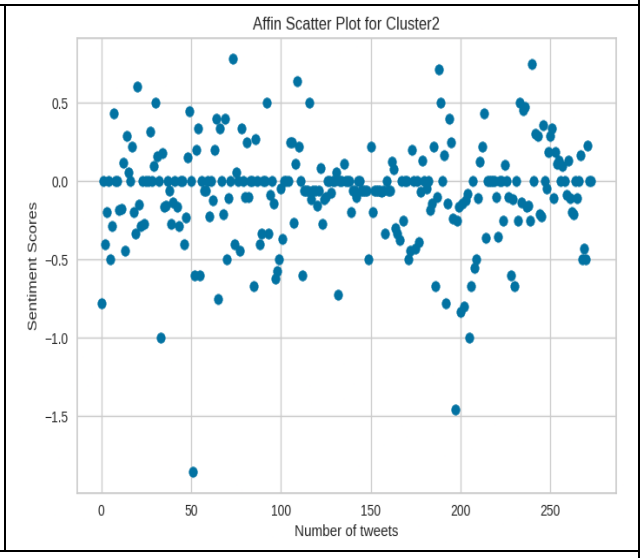


Fig. 4.10.1: VADER & AFINN for all clusters of Economics Community

In the Economics community, after VADER analysis, 2 clusters (1 and 5) are densely populated while the other three clusters (2,3,4) are sparsely populated. In this community also, a lot of data points are scattered all over and have slight inclination towards positive side. Almost all clusters have an equal number of points in this community. But particularly in cluster 1 and cluster 5, some data points are more over the negative side. These results were ensured using AFINN analysis as well. Thus, we can conclude that overall, the tweets are inclined towards positive sentiments.

4.11 VADER & AFINN – Politics Community

VADER	AFINN
Cluster 1	
	
Cluster 2	
	
Cluster 3	

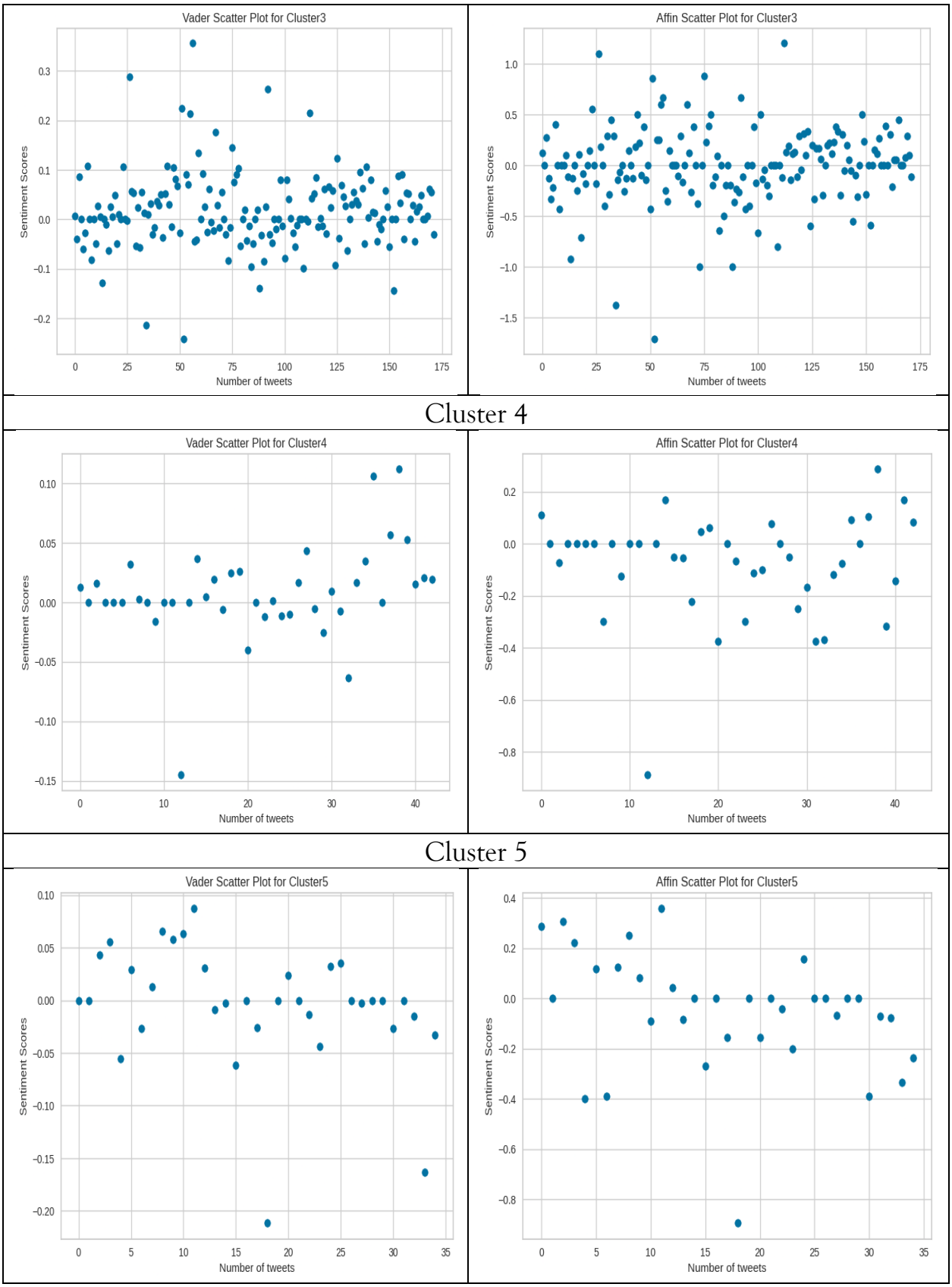


Fig. 4.11.1: VADER & AFINN for all clusters of Politics Community

In the Politics community, after VADER, 3 clusters (1,2,3) are densely populated while the other 2 clusters (4,5) are sparsely populated. In this community, even though a lot of data points are scattered all over, as compared to other 2 communities a lot of data points are slightly inclined towards the negative side. Particularly in cluster 5 there are data points on the negative side. Also, in cluster 1 and cluster 2 the data points are scattered on both positive and negatives sides equally. These results were ensured using AFINN analysis as well. Thus, we can conclude that overall, the tweets are inclined towards negative sentiments.

4.12 Subjectivity and Polarity Analysis – Technology Community

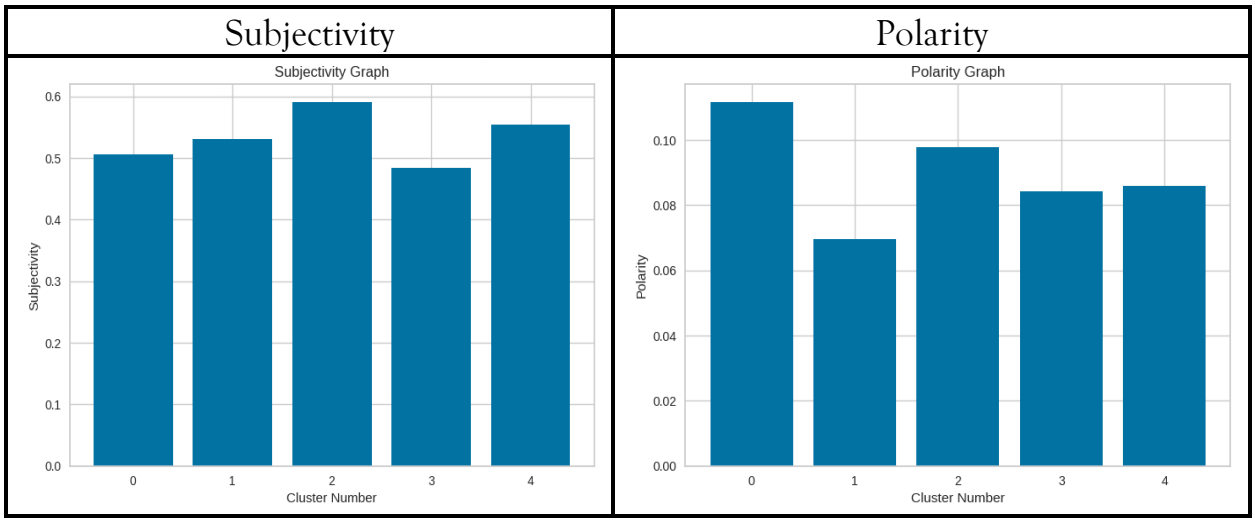


Fig. 4.12.1: Plot of Subjectivity and Polarity for all clusters of Technology community

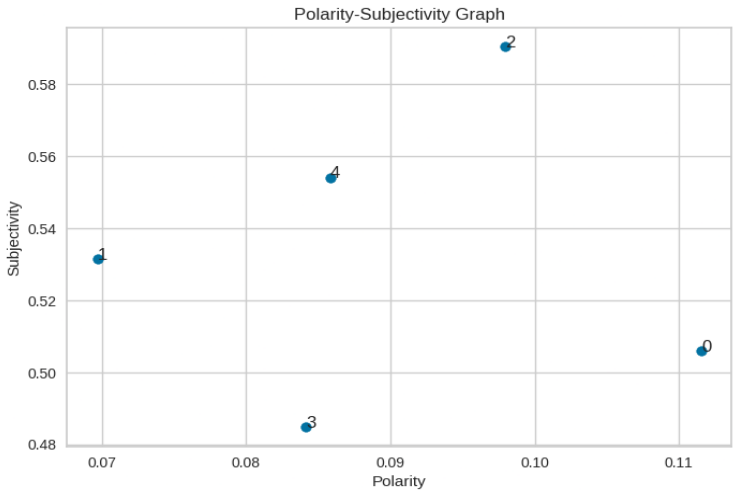


Fig. 4.12.2: Plot of Subjectivity vs Polarity for all clusters of Technology community

Note: 0th cluster refers to Cluster 1, 1st cluster refers to Cluster 2, 2nd cluster refers to cluster 3, 3rd cluster refers to Cluster 4, 4th Cluster refers to Cluster 5

Upon analyzing the subjectivity graph for the technology community, we can observe that the tweets from all clusters are almost inclined to be subjective which means that tweets from all clusters are more inclined in expressing feelings and sentiment. We can also notice that cluster 2 has higher sentiment compared to the remaining cluster.

Upon analyzing the polarity graph for technology community, we can infer that all the clusters are leaned towards being on the positive sentiment with cluster 1 leaned on slightly towards being neutral with sentiment.

We have derived a graph with the relationship between polarity and subjectivity with x-axis representing the polarity score and y-axis representing the subjectivity score. We can observe that cluster 2 has the highest subjectivity score since the placement of cluster 2 is the highest in the y-axis (positive of y- axis) and cluster 3 has the lowest since it is placed closer to the x-axis. Also, cluster 1 has the lowest polarity score since it is closer to the y axis and cluster 0 has the highest polarity score.

4.13 Subjectivity and Polarity Analysis – Economics Community

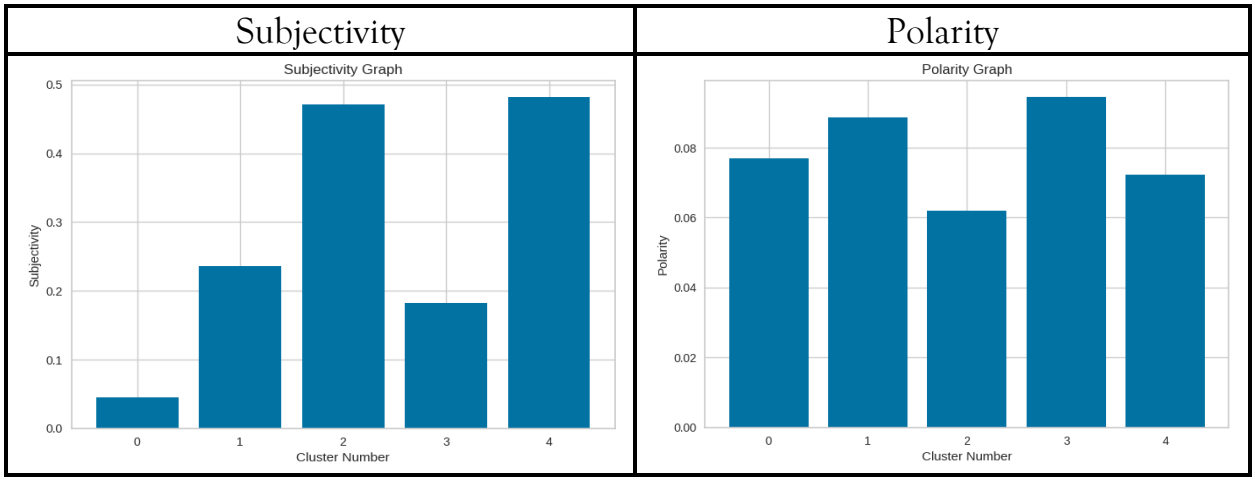


Fig. 4.13.1: Plot of Subjectivity and Polarity for all clusters of Economics community

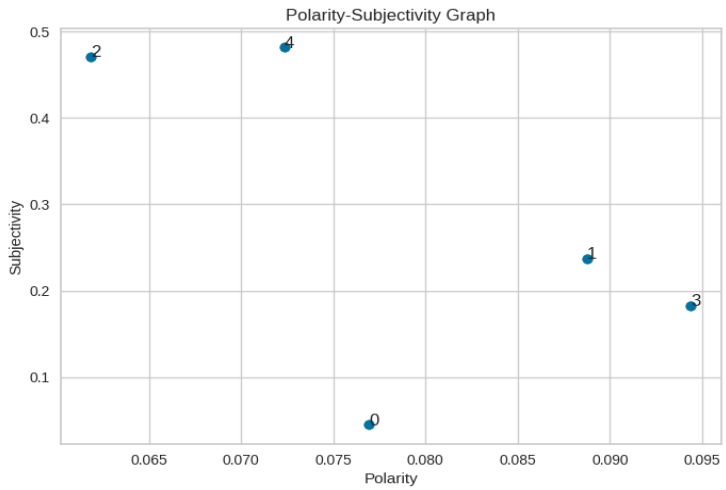


Fig. 4.13.2: Plot of Subjectivity vs Polarity for all clusters of Economics community

Note: 0th cluster refers to Cluster 1, 1st cluster refers to Cluster 2, 2nd cluster refers to cluster 3, 3rd cluster refers to Cluster 4, 4th Cluster refers to Cluster 5

Upon analyzing the subjectivity graph for economics community, we can observe that cluster 0 has the lowest subjectivity score as compared with the other clusters which conveys that the tweets in cluster 0 are more inclined in being objective and factual. Also, cluster 2 and cluster 4 being almost equal conveys that they are more inclined in expressing feelings and sentiments.

Upon analyzing the polarity graph for economics community, we can observe that all the clusters lean towards the positive sentiment with cluster 2 being lesser as compared to the other clusters.

Upon analyzing the polarity-subjectivity graph, we can observe that cluster 2 and cluster 4 are more subjective since they're present at the top (positive of y axis) in the graph. Cluster 0 is closer to the x axis conveying that it's more objective. We can also observe that all the clusters are placed more towards the right (positive of x axis) meaning that all the tweets have positive sentiment with cluster 2 being closest to the y axis meaning that it has lesser polarity score than the rest.

4.14 Subjectivity and Polarity Analysis – Politics Community

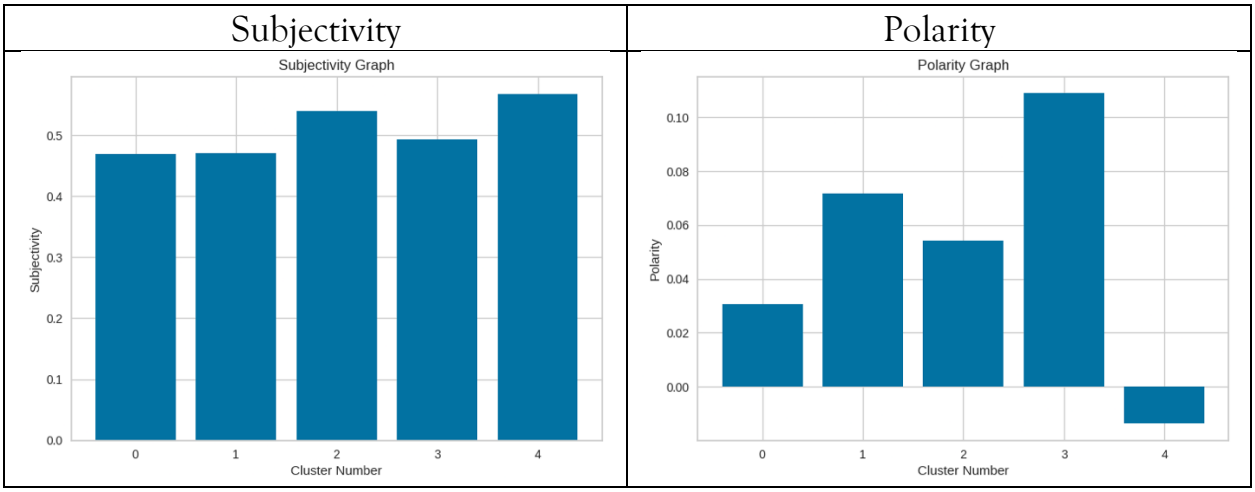


Fig. 4.14.1: Plot of Subjectivity and Polarity for all clusters of Politics community

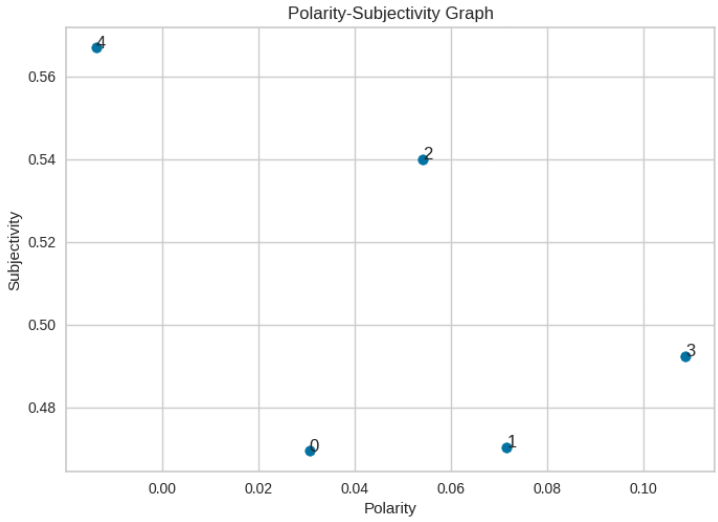


Fig. 4.14.2: Plot of Subjectivity and Polarity for all clusters of Politics community

Note: 0^{th} cluster refers to Cluster 1, 1^{st} cluster refers to Cluster 2, 2^{nd} cluster refers to cluster 3, 3^{rd} cluster refers to Cluster 4, 4^{th} Cluster refers to Cluster 5

Upon analyzing the subjectivity score for the politics community, we observe that the tweets from all clusters lean towards being more subjective with a lot of feelings and sentiments with cluster 4 being the highest as compared to the other clusters. We can also observe that the subjectivity score of clusters 0 and cluster 1 are the same.

Upon analyzing the polarity score of politics community, we can observe very evident results of cluster 4 being inclined to negative sentiments. We can also observe the fact that cluster 0 is on the positive sentiment yet it's closer to the value 0 indicating neural sentiments.

Upon analyzing the polarity subjectivity graph, we can observe that cluster 4 has the highest subjectivity score, since it's placed to the extreme positive of the y axis. We can also observe that it's placed in the negative extreme of x axis inferring that it's more subjective and has negative sentiments.

4.15 LDA Topic Modeling – Technology Community

Cluster 1:

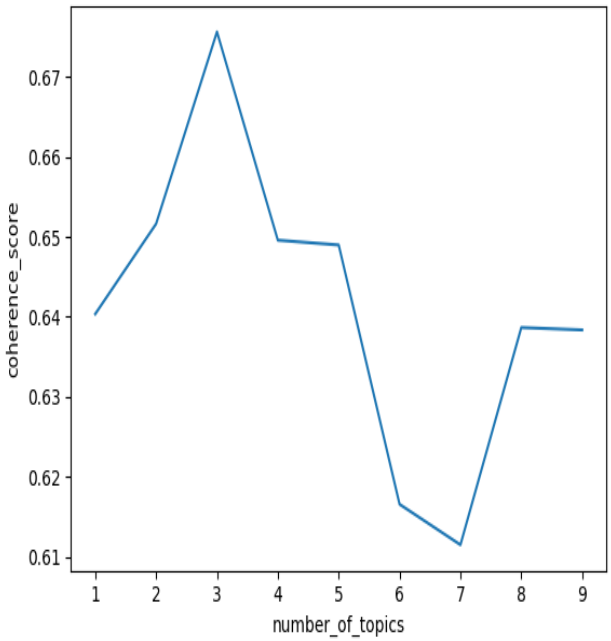


Fig. 4.15.1: Coherence score for Technology cluster 1.

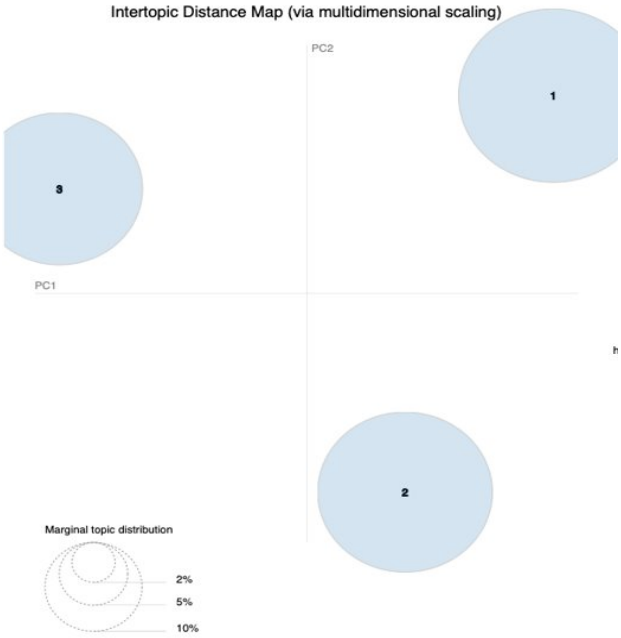


Fig. 4.15.2: Visual representation of LDA for cluster 1 of Technology community

The first cluster was divided into 3 topics with the following keywords. Based on the distribution of keywords, the first topic indicates users who spend most of their time on the computer, the second on users who use science and technology as a means of making money and the third on users who use computers to assist them in work. From these keywords we can infer that the first cluster of users are those who use technology medium in their work.

Topic	Keywords
1	Computer, time, day, spend, game
2	Science, business, money, digital
3	Computer, help, pay, write, work, analysis

Table 4.15.1: Keywords per topics for Technology cluster 1

Cluster 2:

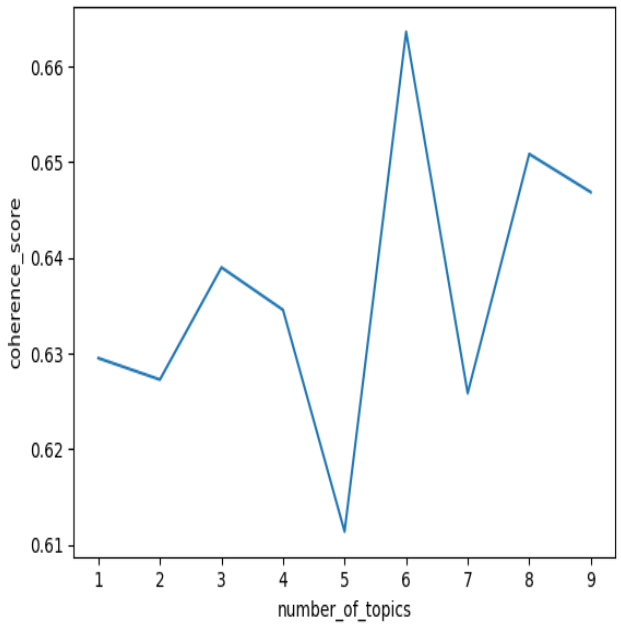


Fig 4.15.3: Coherence score for technology cluster 2.

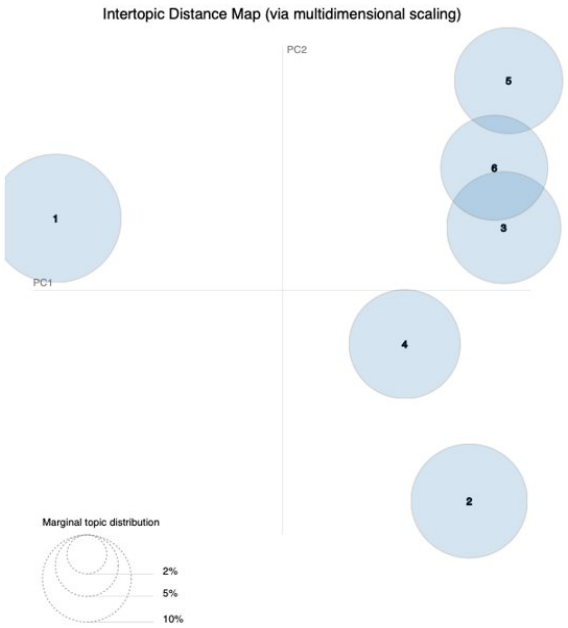


Fig. 4.15.4: Visual representation of LDA for cluster 2 of Technology community

Note: The coherence score for cluster 2 peaks at 6. Hence, the number of topics chosen for cluster 2 is 6.

The second cluster was divided into 6 topics. Based on the Inter topic distance map, topics 3, 5, and 6 are closely located to each other and their keywords such as 'phone', 'game', 'pay', 'work', 'analysis' suggest users who use technology in their business as jobs. Topic 4 is also close and implies technological jobs with keywords such as 'money' and 'engineer'. Topic 1 implies users who use technological applications as they have unique keywords such as 'account' and 'post'. Topic 2 seems to be users who use technology in school, in particular with design. This cluster has a variety of users but normally those who are the core users of technology in school or in business.

Topic	Keywords
1	Computer, account, clean, stuff, post
2	People, design, learn, fun, authorization, school
3	Data, phone, time, spend, program, study
4	Science, money, engineer, mine
5	Help, pay, write, work, analysis
6	Game, screen, business, system

Table 4.15.3: Keywords per topics for Technology cluster 2

We also examined cluster 3, 4, and 5 but when taking into consideration how the clusters were distributed most of their keywords overlapped with cluster 1 and 2. For instance, words such as 'computer', 'game', 'science', 'program' repeatedly overlapped making it hard to distinguish the user demographic.

4.16 LDA Topic Modeling – Economics Community

Cluster 1:

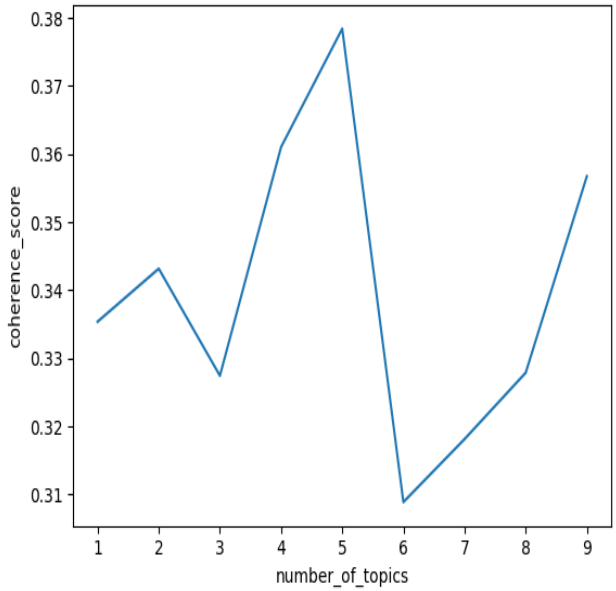


Fig. 4.16.1: Coherence score for economics cluster 1.

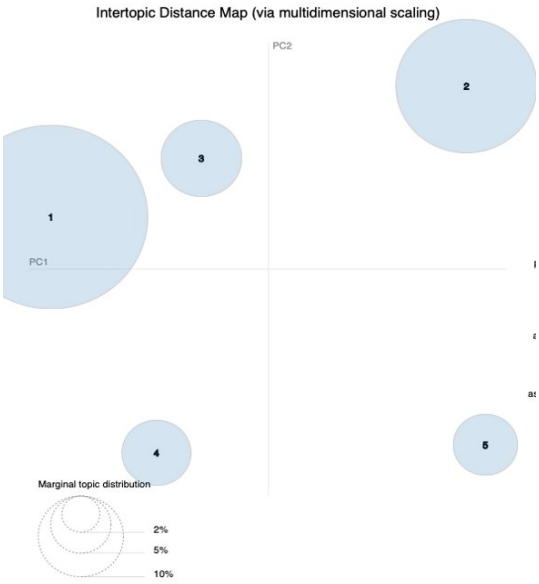


Fig. 4.16.2: Visual representation of LDA Modeling for cluster1 of Economics community.

Note: The coherence score for cluster 1 peaks at 5. Hence, the number of topics chosen for cluster 1 is 5.

The first cluster is divided into 5 topics. Overall, the user demographic seems likely to be students. While each topic has distinctive keywords that differentiate themselves from other topics, they all include general student related keywords such as 'essay', 'assignments', 'course'. They were all proportionately distributed in the inter topic distance map as well.

Topic	Keywords
1	Essay, biology, assignments, homework
2	Grade, data, work, best, course
3	Essaypay, essaydue, statistics, mathematics
4	Exams, final, book, review, coding
5	Math, class, write, finance, online

Table. 4.16.1: Keywords for Economics cluster 1

Cluster 4:

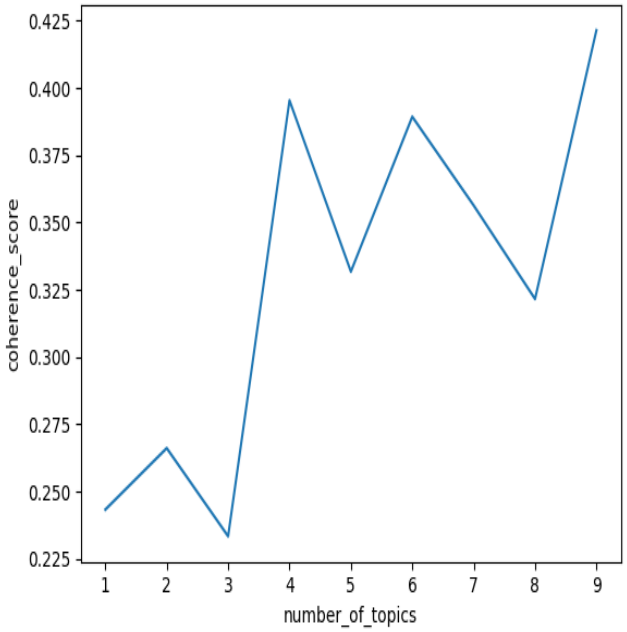


Fig. 4.16.3: Coherence score for economics cluster 4

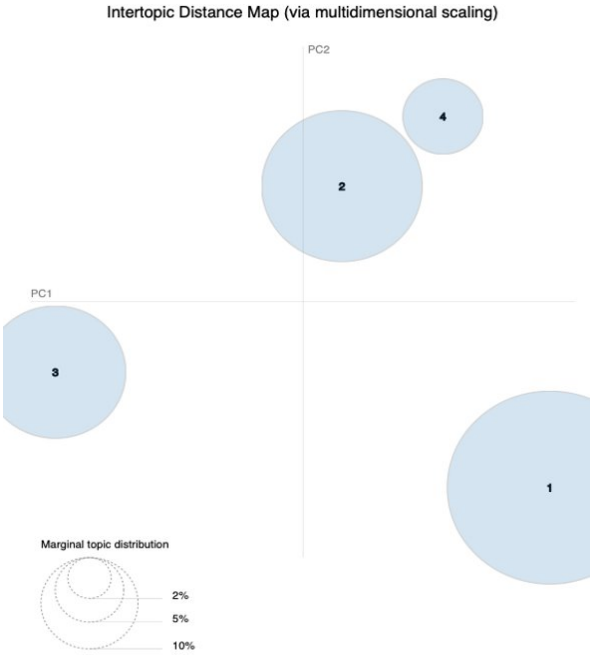


Fig. 4.16.4: Visual representation of LDA Modeling for cluster 1 of Economics community.

Note: The coherence score for cluster 4 peaks at 4. Hence, the number of topics chosen for cluster 4 is 4.

The fourth cluster also indicates student users. As the first cluster, keywords selected for each topic include 'class', 'discussion', 'canvas'. The details of each topic keyword are in the table and the picture depicts the intertopic distance, which is fairly distributed just like the first cluster. However, unlike the first cluster this seems to include students from different majors, not just economics but nursing, computer science, biology and more.

Topic	Keywords
1	Essay, biology, homework, paper, thesis
2	Discussion, term, data, analysis
3	Class, finance, online, fall, classes
4	JavaScript, assignments, history, nursing

Table 4.16.2: Keywords for Economics cluster 4.

Cluster 2:

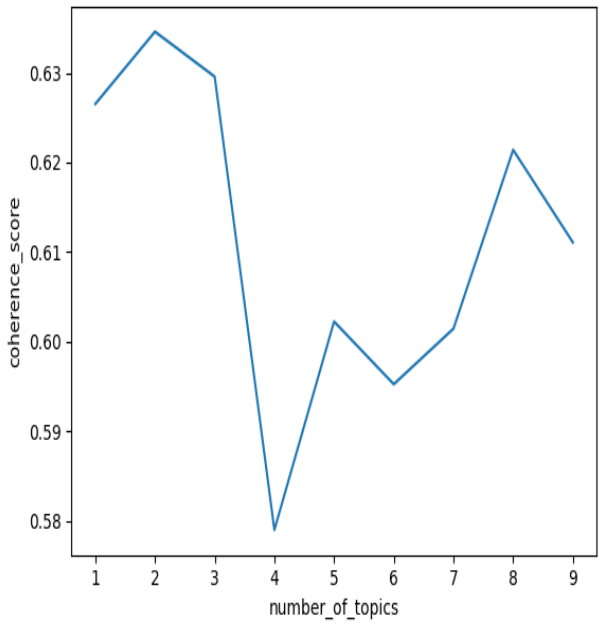


Fig. 4.16.4: Coherence score for economy cluster 2

Fig. 4.16.5: Visual representation of LDA Modeling for cluster 2 of Economics community.

Note: The coherence score for cluster 2 peaks at 2. Hence, the number of topics chosen for cluster 2 is 2. In order to get a broader idea about the cluster we picked the second highest peak which is 3.

The second cluster is divided into 3 topics. Based on the keywords for each topic, users in this cluster mostly discuss economy in general. Topic 1 users are concerned with the effect on economics the pandemic may have on while topic 2 users discuss how to analytically approach economic effects. The last topic includes yet again student users, but they also discuss finance and law which are generally considered in real-life economics.

Topic	Keywords
1	Prices, higher, pandemic, food, profit
2	Class, finance, business, law, political
3	Pay, quality, data, post, analysis

Table 4.16.3: Keywords for Economics cluster 2.

The third and fifth cluster are pretty similar but also at the same time they both partially have overlapping users from other clusters. In general, both clusters include users who talk about American economics. The table below describes keywords that depict users concerned with American economics mainly.

Cluster	Keywords
3	Pandemic, gas, american, political, profit, business, policy
5	Families, work, profit, american, corporation, business, cost

Table 4.16.4: Keywords for Economics cluster 3 and 5.

4.17 LDA Topic Modeling – Politics Community

Cluster 1:

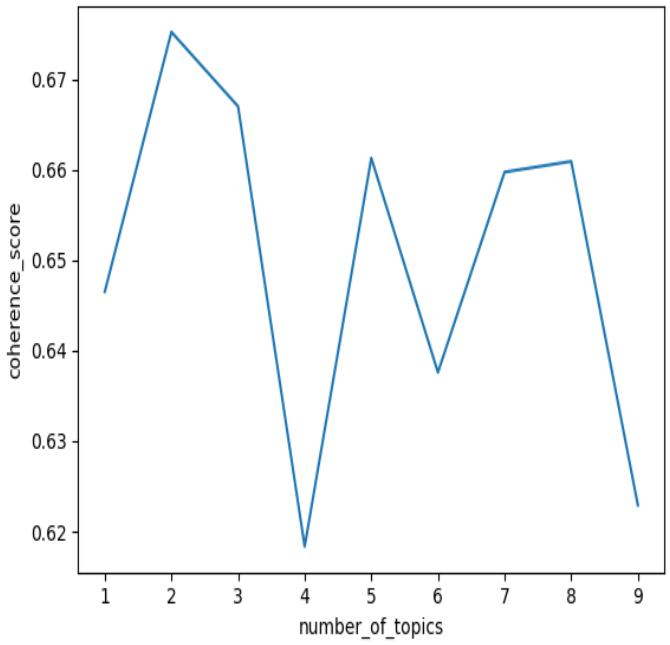


Fig. 4.17.1: Coherence score for Politics cluster 1

Fig. 4.17.2: Visual representation of LDA Modeling for cluster 1 of Politics community.

Note: The coherence score for cluster 1 peaks at 2. Hence, the number of topics chosen for cluster 1 is 2.

The first cluster was divided into 2 topics. Based on the keywords, they were both indicative of the political issue regarding covid-19. We can also see that since the tweets we collected are English, most of the political issues are related to American politics with the keyword such as 'trump' and 'fox'.

Topic	Keywords
1	Government, people, federal, covid, state
2	Overthrow, news, law, party, trump, fox

Table 4.17.1: Keywords for Politics cluster 1.

Cluster 2:

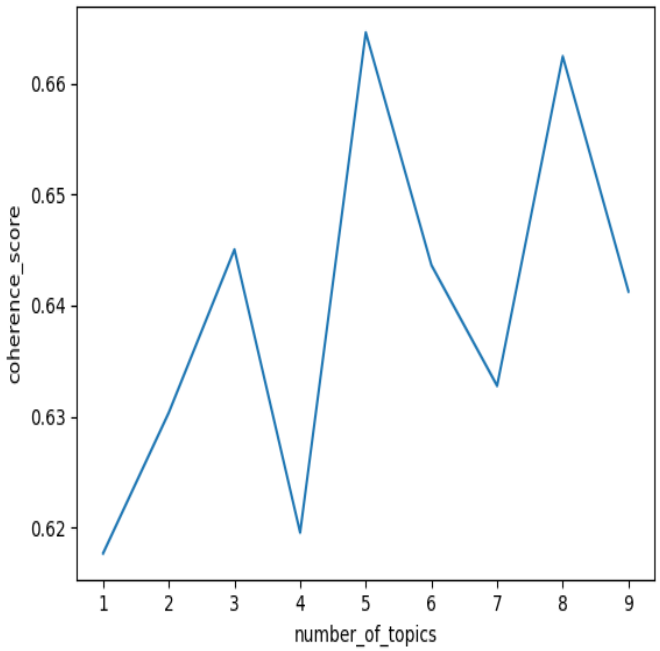


Fig. 4.17.3: Coherence score for Politics cluster 2

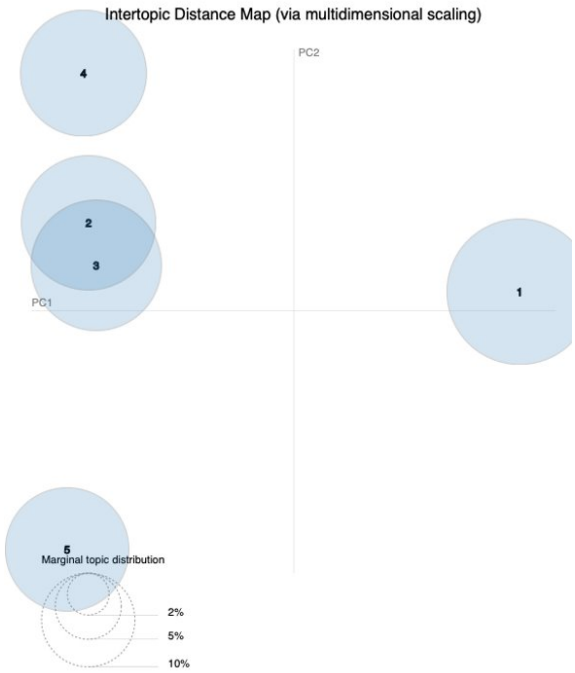


Fig. 4.17.4: Visual representation of LDA Modeling for cluster 2 of Politics community

Note: The coherence score for cluster 2 peaks at 5. Hence, the number of topics chosen for cluster 2 is 5.

The second cluster is divided into 5 different topics and each topic represents different political issues. While some talk about political discussions regarding the pandemic, some talk about the US presidential election and some talk about politics in general. In general, similar to cluster1, they are all talking about American politics.

Topic	Keywords
1	State, party, support, political
2	Overthrow, private, vote, american
3	Tax, america, trump, biden, democracy
4	Pay, covid, news, control, fox
5	Government, public, man, politicians, better

Table 4.17.1: Keywords for Politics cluster 2.

The third, fourth and fifth cluster have overlapping keywords with the first two clusters. The third cluster includes topic keywords where users share their thoughts on tax information unlike any other clusters though.

4.18 Word Cloud Analysis – Technology Community

In the technology community's word cloud, we noticed that "Problem" appeared frequently in several clusters (Clusters 1, 3, 4, and 5). This suggests that people in the technology community associate technology with problems (e.g., problem-solving). While some clusters had similar words, we were able to categorise them into two major clusters. Clusters 1 and 2 were focused on academia-related topics like "friend," "math," and "assignment," while Clusters 3, 4, and 5 were more professionally oriented with words like "nft," "app," "product," and "scientist."

Fig. 4.18.1: Word Cloud for all clusters of Technology community

4.19 Word Cloud Analysis – Economics Community

Moving on to the economics community, we noticed a wider range of topics than in the technology community. "Payment" was a common keyword in most clusters (Clusters 1, 2, 3, and 4), which aligns with people's association of economics with payment. We categorized the five clusters into two groups: Clusters 1 and 2 were related to lockdown, while Clusters 4 and 5 had a mix of academic words (e.g., "assignment," "math") and business-related words (e.g., "worker," "assets," "bitcoin").

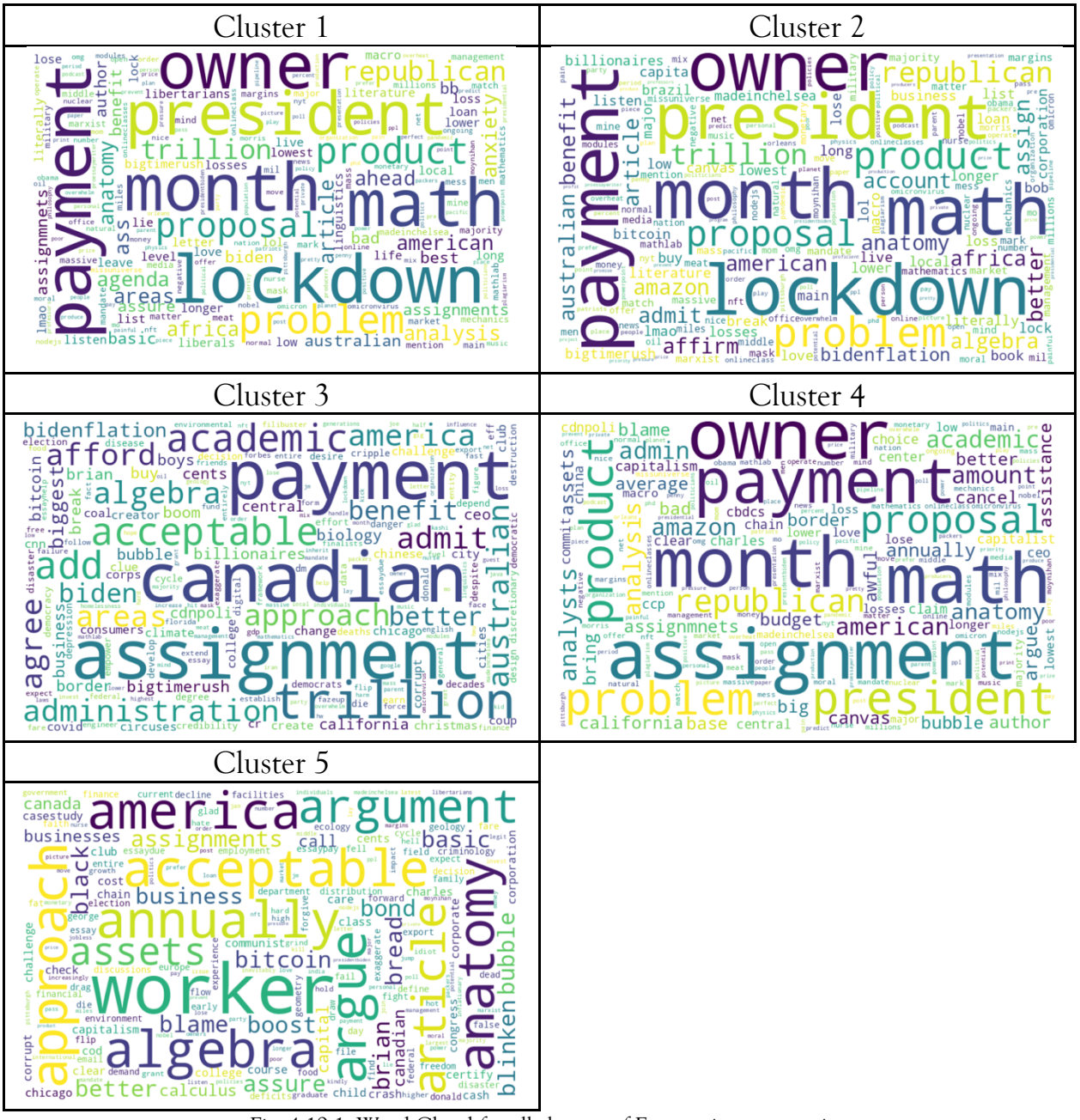


Fig. 4.19.1: Word Cloud for all clusters of Economics community

4.20 Word Cloud Analysis – Politics Community

In the politics community, "Opinion" was a common keyword in most clusters (Clusters 2, 4, and 5), indicating that people associated politics with opinions. One notable observation was that politics was more influenced by current issues. For instance, we found multiple tweets about "Prison," which was a trending topic about the Scottish government's effective ban on trans prisoners who committed sexual and violent crimes against women being moved to a women-only prison. We also noticed a higher frequency of Republican-related keywords than Democratic-related ones.

Fig. 4.20.1: Word Cloud for all clusters of Politics community.

4.21 Summarisation – Technology Community

The technology community is vast, as technology can be applied in various areas such as economics, politics, and more. Nonetheless, Cluster 3 stands out with a cohesive summarization result that revolves around cyber security. Cluster 1, 2, 4 and 5 shows mixed statements and opinions on Technology community and is somewhat domain specific with some set of words like “innovation”, “market”, “hackathon”.

```
----------
• Cluster1
agu represent share benefit laser communications scientists public. transform business fight climate
change tend notice recognition feel good innovation sustainability turnthetide. send tesla exchange
theory travel interstellar. best work deep problem satisfy ignore community work solve problem help
satisfy life work guess. substitute type android real dak play week advance days.
----------
• Cluster2
imply china forget keep copy hear birth certificate hear computer help chavez elect exist hear vote.
horrible surprise hear consider typical knowledge gap generations computer weird ass effort creep
child. quote call dr el hayek lebanese ignorant premise germany mongol khazars speak hebrew german
moron fall computer israeli investigate complicate. cookie terrible kid move computer daughter
scrappy hacker digital implant hair dad untraceable free hand punch. pop staple correction tape
browse laptops fun convo computer brand years agreement hps suck toshiba sony vaio superior.
----------
• Cluster3
whitepaper mitre att ck framework source cybersecurity. security officer caribe royale hotel orlando
fl caribe royale orlando itjobs orlandojobs. team trash hackers treasure find cyber security expert
share safest dispose computerhacking hack. health source evidence base practice health source.
better speed security wait time national airports.
----------
• Cluster4
roe happen medical sonograms high image roe happen laws regard medical issue update. humans curse
fear evidence existence general high average live better keep roll dice better bet. nasa newest ray
observatory rocket orbit thursday light explode star black hole violent high energy events unfold
universe abc news detail science space exploration. nasa newest ray observatory rocket orbit
thursday light explode star black hole violent high energy events unfold universe abc news detail
science space. high filter reason hologram listen podcast talk conspiracy bomber hologram shoot
second tower real hologram tweet lol.
----------
• Cluster5
moderna unicorn company start desperate market patent mrna trials fail abandon exotic disease market
chase vaccines. underappreciated deflationary force decades accelerate pandemic zoom call replace
meet gather lecture conferences superpower summit reunions deflation nation. toyota pay subscription
fee start car remotely. correction empire build smart technological aspect clothe market game whilst
subdue young women color treacherously rap spirit body tightenup. hackathon idea improve government
efficiency state controller office start issue wire transfer send check class mail real low bar
improve governmental efficiency.
```

Fig. 4.21.1: Summarization all clusters of Technology community.

4.22 Summarization – Economics Community

The economics community has more focused subjects, resulting in more cohesive summarization compared to the technology community. Specifically, the summarization of Cluster 1, 2, and 4 is assignment-related, while Cluster 3 and 5 are more centered on government, policy, market, and macroeconomics.

- Cluster1

help assignments pay phy maths stats finals essay chemistry history nurse javascript calculus geography thesis trigonometry. grade nurse physics chemistry literature english essay pay account essay history assignments microbiology biology. help assignments pay physics biology english essay finals essay chemistry history nurse javascript calculus geography thesis trigonometry. hand assignments physics biology english essay pay maths stats essay chemistry history physical science calculus essay help geography thesis txsu su su txsu nccu ssu lsu asutwitte su txsu txsu tamu aamu jsu. well handle assignments dm biology psychology sociology paper literature essay pay maths hw statistics exams essay history nurse physics javascript calculus algebra excel essaypay thesis discussion umhb dm.

- Cluster2

trigonometry chemistry physics help thesis maths biology powerpoint sociology account essay essay pay casestudy literature term paper whatsapp text. online class help account essay pay maths essay thesis essay help assignment dissertation paper law excel algebra calculus finance exam help expert class kick ass theflash bachelorette. trigonometry chemistry physics help thesis maths biology powerpoint sociology account essay essay pay casestudy term paper. trigonometry chemistry physics help thesis maths biology powerpoint paper homework sociology account essay essay pay casestudy literature. chemistry physics help thesis maths biology powerpoint paper homework sociology account essay essay pay casestudy literature term paper.

- Cluster3

fair finance whore critical totally condone finance corporate government amp mob whore. wsj biden stagflation usa america biden trump trump freedom patriots democrats republicans inflation gasprices bidenflation politics. canada underfunding defense defend canada defend canada country neuter europe establish global roles reverse. osha open investigation amazon warehouse collapse tornado kill. blinken develop comprehensive indo pacific economic framework pursue share objectives include trade digital biden administration flesh framework entail clue today speech.

- Cluster4

pay assignment tutor homework amp academic task online class finance essay biology chemistry physics history math paper stats calculus acc algebra english sociology excel final semester anatomy ethics email custompapers dm. hire term paper case study nurse fall semester online class admission post thesis project paper dissertation discussion quiz assignment homework essay chemistry psychology philosophy algebra finance calculus. week ahead exams fall student friendly price college biology essay report chemistry statistics daysofcode exams hesitate reach ace paper. type assignment discipline pay essay sociology english history geology math statistics homework algebra calculus online class finance test exam psychology chemistry fall semester finals dm usa uk canada uae finalsweek. type assignment discipline pay essay sociology english history geology math statistics homework algebra calculus online class finance test exam psychology chemistry fall semester finals

---

- Cluster5

hard disagree worry tweet joe biden lack competency example inflation stock. laughable inflation mandate freedom toilet joke fuks. people gratitude people keep years deal general garbage people shop. number americans file initial jobless claim unexpectedly fell lowest level years fare better economists expect forbes detail employment. president congress protect global inflation presidents credit blame authority set wag price interest rat despite dow unprecedented highs.

Fig. 4.22.1: Summarization all clusters of Economics community.

4.23 Summarization – Politics Community

In terms of the politics community, the summarization result is also cohesive. Cluster 1 and 2 discuss government-related topics such as politics, government, Biden, and Trump. Cluster 3 is a general representation of criticism among North American politics. Cluster 4 talks about business-related topics such as stock and money, while Cluster 5 is more focused on technology-related topics such as smartphones and laptops.

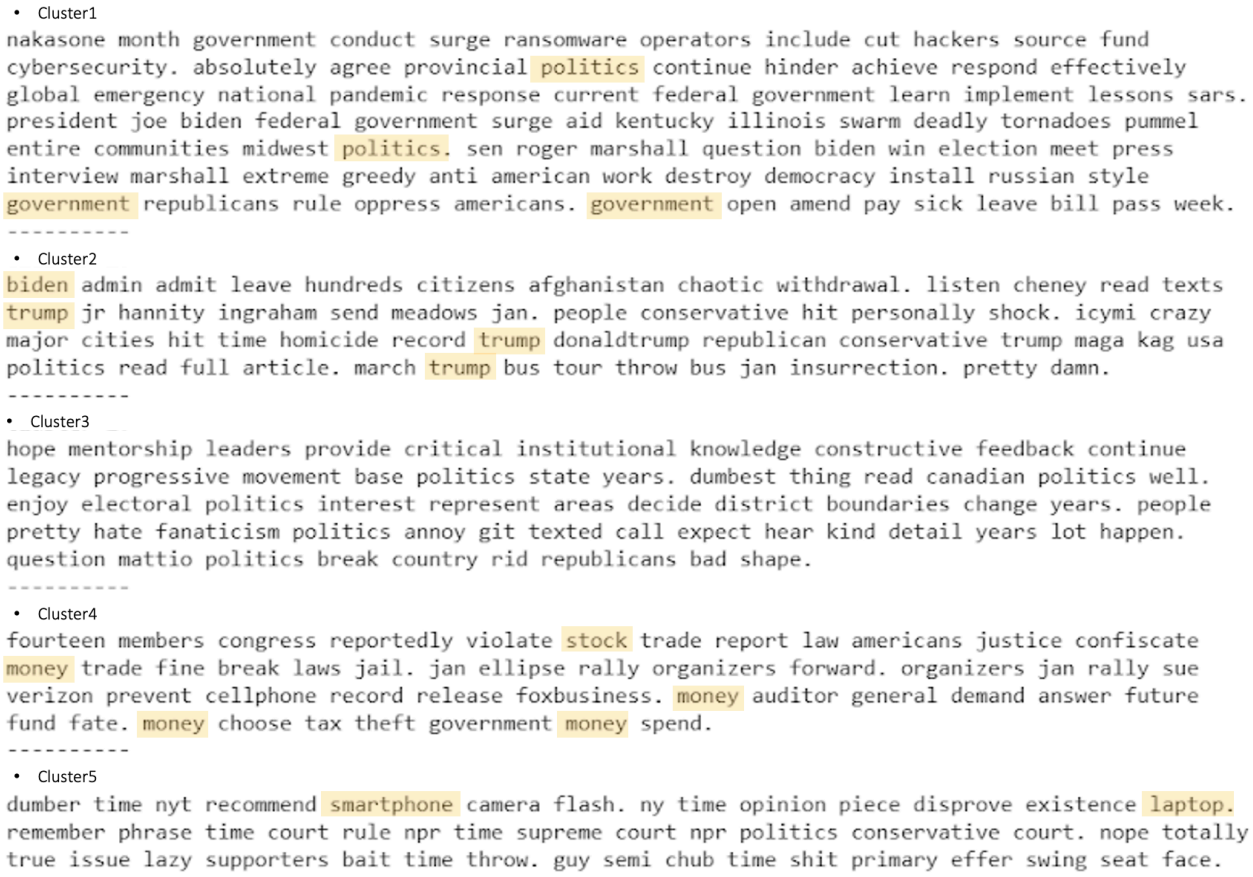


Fig. 4.23.1: Summarization all clusters of Politics community.

4.24 Key findings from all the analysis performed:

Technology Community

The analysis of the Technology community revealed several interesting facts. Firstly, the sentiments expressed in the clusters were predominantly positive, with no strong opinions expressed. This could be indicative of a community that is generally supportive and open-minded.
One of the clusters in the Technology community was focused on trending topics and users in the domains of science, hacking, blockchain, and bitcoin. This could indicate that these are popular and highly discussed topics in the Technology community. Another cluster was more domain-specific, with a focus on keywords related to literature, psychology, sociology, and their impact on technology. An interesting finding was that one of the clusters was representative of economic activities and bills in the tech industry. This could indicate that members of the Technology community are interested in the business and economic aspects of the tech industry, in addition to the technological aspects. Finally, a significant portion of all the clusters could be inferred as being related to tech academia (university or school), based on the prevalence of keywords like "essay," "school," "university," and "work."

Economics Community

The analysis of the Economics community revealed that the first two clusters were focused on individuals facing financial losses due to inflation, with keywords like "loss," "lockdown," "loan," and "payment problem." The third cluster appeared to represent the demographics of the capitalist North American economy.
Another cluster contained a mix of individuals from academia as well as people with opinions on the American economy. This could indicate that the Economics community is not limited to just professionals in the field, but also includes individuals with a general interest in economic issues.
The last cluster in the Economics community consisted of working-class individuals and business owners who own assets. This cluster could represent the voices of individuals who are directly impacted by economic policies and decisions.

Politics Community

The analysis of the Politics community revealed that the first cluster was representative of the Republican agenda on various issues, such as healthcare, defense, elections, taxes, and government. The second cluster represented ideas currently being discussed in the Senate and Congress.
One of the clusters contained sentiments related to elections and American foreign affairs. The fourth and fifth clusters also had views on political leadership and government operations in the country. Additionally, the inclusion of words such as "abortion," "assault," and "arrest" implies that the clusters may also be discussing social justice issues and criminal justice reform.
Overall, the analysis of the Twitter data revealed unique insights into the sentiments and topics of discussion in the Technology, Economics, and Politics communities. These findings can be useful for understanding the priorities and concerns of these communities and can inform future research and decision making.

5. Challenges

- One challenge we encountered with location data is the presence of irrelevant geotagging. For example, some users had tagged their locations as Mars, Moon, or other unrealistic places, which are not actually their physical locations. Hence, we were not able to analyze the clusters based on location tagging.

- Manual filtering was required to eliminate fraudulent or scam tweets in order to ensure the validity and reliability of the collected data.

- Stemming resulted in a loss of meaning for certain words, resulting in over-stemmed words so we did not implement this technique.

- The final challenge we encountered was that we needed to remove stop words twice to identify the meaningful words and eliminate noise from the text data. We defined a custom stop words list to perform this.

6. Conclusion

In conclusion, the project successfully explored Twitter data to understand different communities through clustering, keyword analysis, and sentiment analysis techniques. The project findings provide valuable insights into the interests, beliefs, and emotions of diverse groups of people on Twitter. The sentiment analysis results suggest that most of the clusters in the Technology community had positive sentiments and Economics, and Politics communities both have mixed sentiments inclined toward negative. Additionally, the keyword analysis highlighted the top terms and themes within each cluster, providing a deeper understanding of the users' characteristics in each community. There is a strong presence of individuals from academia in all the communities. Apart from these, the technology community also has a lot of individuals who belong in various domains of technology like “blockchain”, “ethical hacking”, “scientists”, etc. Similarly in economics users are mostly from the capitalist and working-class background. The politics community users are mostly interested in senate house debates and government/opposition agendas. These insights can be used to inform targeted marketing campaigns, enhance social media management, and identify emerging trends in various industries.

The project's implementation of k-means clustering algorithm, evaluation methods, sentiment analysis tools, and LDA modeling techniques showcased the power of data analysis in uncovering meaningful insights from social media data. The use of summarization techniques further facilitated the communication of insights in a concise and actionable manner. In the future, we can include expanding the analysis to include more communities and exploring other advanced techniques such as network analysis and deep learning to further improve the understanding of different communities on Twitter.

Overall, the project has demonstrated the potential of data analysis in gaining insights into diverse communities and enhancing decision-making in various industries.

7. Future Scope

Our work analysed each community by dividing them into clusters and doing a thorough sentiment analysis as well as keyword analysis. With this initial analysis as a foreground, we introduce some ideas for a deeper analysis that can be conducted in the future.

First, a thorough analysis based on location can be implemented. As for now we have excluded the location information of users as there were several outliers such as 'mars' set as a location. In the future, we can exclude those unrealistic locations and do a thorough analysis based on location. By conducting a geo-location analysis, we can use that to map out different opinions from different regions which may provide a new insight into each community.

Second, an inter comparison of each community can be conducted. As for now, we did a sentiment analysis within the community but, each community may influence each other whether positively or negatively. It may be crucial to pinpoint how each community poses influence on one another. For instance, if a president makes a political decision this does not impact the politics community but may impact the economic community and the technology community one way or another. Figuring this inter-relationship between the community is crucial to getting realistic insight from the information we collect.

Finally, we can utilise the nature of Twitter and find a user who influences the most in a community. For instance, when Elon Musk mentioned the Doge coin the value skyrocketed. In this case, Elon Musk is the influencer in the coin market. When we figure out who the influencer is in each community, it will be easier to track the flow of information. With the information disseminated faster and faster, it is crucial to pinpoint who the influencer is to avoid the spread of misinformation and locate where the trend is starting.

References

1. Wang, Zhengkui, et al. "Twiinsight: Discovering topics and sentiments from social media datasets." arXiv preprint arXiv:1705.08094 (2017).

2. Jaffali, Soufiene, et al. "Clustering and classification of like-minded people from their tweets." 2014 IEEE International Conference on Data Mining Workshop. IEEE, 2014.

3. Garg, Neha, and Rinkle Rani. "Analysis and visualization of Twitter data using k-means clustering." 2017 International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2017.

4. Cruickshank, Iain J., and Kathleen M. Carley. "Characterizing communities of hashtag usage on twitter during the 2020 COVID-19 pandemic by multi-view clustering." Applied Network Science 5 (2020): 1-40.

5. "Getting Started with Textblob for Sentiment Analysis." Top Website Designers, Developers, Freelancers for Your Next Project, www.topcoder.com/thrive/articles/getting-started-with-textblob-forsentimentanalysis#:~:text=Polarity%20is%20the%20output%20that,to%20personal%20opinions%20and%20judgments. Accessed 5 March 2023.

6.sitesh431."Sitesh431/YouTube-Comment-Sentiment-Analysis." GitHub, github.com/sitesh431/Youtube-Comment-Sentiment-Analysis. Accessed 5 March 2023.

7. Tweepy, www.tweepy.org/. Accessed 15 March 2023.

8. "Gensim: Topic Modelling for Humans." Models.Ldamodel – Latent Dirichlet Allocation - Gensim, 21 Dec. 2022, radimrehurek.com/gensim/models/ldamodel.html.

9. Mohammad, and Turney. "Mohammad, Saif M." Canada.Ca, 6 July 2022, nrc-publications.canada.ca/eng/view/object/?id=0b6a5b58-a656-49d3-ab3e-252050a7a88c.

10. Mohammad. "Mohammad, Saif." Canada.Ca, 17 Sept. 2021, nrc-publications.canada.ca/eng/view/object/?id=166f8642-f65c-4b69-b736-1d2eaac3a094.

11. "Vadersentiment." PyPI, pypi.org/project/vaderSentiment/. Accessed 17 April 2023.

12. "Afinn." PyPI, pypi.org/project/afinn/. Accessed 17 April 2023.

13. "Multi-Dimensional Scaling." Scikit, scikit-learn.org/stable/auto_examples/manifold/plot_mds.html. Accessed 2 April 2023.